2-бөлім Информатика

Раздел 2 Информатика

Section 2 Computer science

UDK 004.382

Aidarov K.A.*, Akhmed-Zaki D.Zh.

Al-Farabi Kazakh National University, Republic of Kazakhstan, Almaty
*E-mail: kanataidarov@yahoo.com

Conducting computational experiments for a module of integration of several computational clusters into single computational complex building unified communication environment

Given paper describes the new instrument of multicluster organization of distributed computations and gives its detailed description using key components and implementation issues as autonomous program module. The purpose of given module and attached broker is to provide framework and execution environment of parallel applications in multicluster computational environments for conducting computational experiments with applied problems of the numerical modeling of the filtration theory merging into unified computational complex several computational clusters. Therefore, built unified communication environment becomes multicluster distributed computational system for solving large scale computational problems. Library part of the module, used in program code of applied problems, is MPJ-Express framework add-on and developed using Java programming language. Using Java in developed add-on allows integration into wide range of computer platforms starting form standard desktop Windows systems and finishing with large Linux clusters, which ensures general rule of the ubiquitous computing. The work presented in given article is a part of grant funding project from Kazakhstan government.
Key words: distributed computing, multicluster systems, cluster unification, cluster merge, parallel programming, MPJ-Express add-on, MPI, Java.

Айдаров К.А., Ахмед-Заки Д.Ж.

Біріккен ақпараттық коммуникациялық ортаны құру арқылы бірнеше есептеу кластерлерін бірлескен есептеу кешеніне интеграциялау үшін есептеу тәжірибелерін іске-асыру

Берілген модульдің және оған қосылатын брокердің мақсаты болып параллельді қолданбалардың фреймворкін және орындалу ортасын ұсыну болып табылады. Бұл бірнеше есептеу кластерлерін біріккен есептеу кешенінің көмегімен сандық модельдеудің фильтрация теориясының қолданбалы есептерімен есептеу тәжірибелерін мультикластерлі есептеу орталарында іске асыру арқылы жасалынады. Осылай құрылған біріккен коммуникациялық орта ірі ауқымды есептеу мәселелерін шешуге арналған мультикластерлі үлестірілген есептеу жүйесін болып табылады. Қолданбалы есептердің бағдарламалық кодында қолданылатын модульдің кітапханалық бөлігі Java бағдарламалау тілінде құрастырылып, MPJ-Express атты фреймворкқа қосымша болып табылады. Бұл қосымшада Java платформасын қолдану арқылы оны компьютерлік орталардың кең ауқымына енгізуге мүмкіндік береді стандартты Windows жүйелерден бастап, ірі ауқымды Linux кластерлермен аяқтап. Осы арқылы жаппай есептеу қағидасының негізгі ережесі қанағаттандырылады.
Түйін сөздер: үлестірілген есептеулер, мультикластерлік жүйелер, көпкластерлік жүйелер, кластерлердің бірігуі, параллельді бағдарламалау, MPJ-Express қосымшасы, MPI, Java.

Айдаров К.А., Ахмед-Заки Д.Ж.

Проведение вычислительных экспериментов для модуля интеграции в единый вычислительный комплекс нескольких вычислительных кластеров путем построения единой коммуникационной среды

Задачей разработанного модуля и подключаемого к ней брокера является предоставление фреймворка и среды исполнения параллельных приложений в мультикластерных вычислительных средах для проведения вычислительных экспериментов с прикладными задачами численного моделирования теории фильтраций с помощью объединения в единый вычислительный комплекс нескольких вычислительных кластеров. Построенная, таким образом, единая коммуникационная среда представляет собой мультикластерную распределенную вычислительную систему для решения крупномасштабных вычислительных задач. Библиотечная часть модуля, использующаяся в программном коде прикладных задач, была разработана на языке программирования Java и является надстройкой над фреймворком MPJ-Express. Использование Java в надстройке позволяет интегрировать ее в широкий набор компьютерных платформ, начиная со стандартных настольных Windows систем, и заканчивая крупными Linux кластерами, что обеспечивает основное правило принципа повсеместных вычислений.
Ключевые слова: распределенные вычисления, мультикластерные системы, многокластерные системы, объединение кластеров, параллельное программирование, надстройка над MPJ-Express, MPI, Java.

1. Introduction

Development of reliable and efficient distributed applications was scientific-engineering problem for several decades, and recently achieved its new dimension with development of multicluster distributed large scale systems. Applied examples of such systems can be parallel scientific-computational application with use of the MPI for public computational cluster; scientific work process for high-level distributed computational environments; application of network monitoring, which allows scaling of thousands of nodes into multicluster distributed large scale system; or peering network of file exchange run on thousands of unstable Internet hosts. At development of given applications it is necessary not only develop appropriate distributed algorithms, but also according strategies of resource allocation, which utilizes good sides and avoids exploits of underlying computational platform. Unfortunately, implementation of all of it spawns many issues. Therefore, estimation of application performance in complex computational platforms very difficult and analytical models often based on unrealistic or simplified assumptions. One way is to conduct direct experiments. However, executing actual experiments on real platforms also spawns some complexities. At first, it is necessary to have complete developed application, at the same time comparison of different models and algorithms usually conducts at design and development stage. At second, experiments often restricted by platform configurations, which constrains their general value and applicability. At last, in majority of occasions experiments impossible to repeat (for example, because of non-deterministic distribution of resources), which makes it very difficult to correctly compare alternative approaches.

In the light of these complexities researchers and developers usually using computational simulation. Although, in order to develop accurate and valid computation model, it is necessary to conduct more complex and long-term computations, which makes them expensive by time/resources. Given issue increases with deployment of large scale, long-term application on supercomputer platform. Moreover, studies show that the program code written exclusively for simulation significantly differs from real application code. In given circumstances there are

two issues arising: both codes, sequential and parallel, can behave differently and efforts spent on writing of code of computational model, will be ineffective. Given work provides module of multicluster distributed system, consisting of following components: applied MPI program add-on, written in Java programming language and using MPJ-Express library as the basis of distributed resources broker, merging several computational clusters into united pool of MPI processes. More importantly for applied program using add-on, calling MPI processes located on the other computational cluster no different from usual MPI call. Therefore, MPI program code written to execute in standard MPI environment requires minimal changes to header information of MPI code and does not require change of structure of MPI code calls at all.

2. Short description of the structure the module of multicluster distributed computations

Architecture of multicluster environment in which MPJ-Express add-on deployed shown in the Figure 1. A tasks distribution broker located outside a facility allocating computational resources, however if there is appropriate rights, Broker can assign tasks on unified clusters. Scheduled tasks for connected to broker clusters seems as usual tasks of these clusters. Local task schedulers installed on clusters manage execution of separate tasks using their state scheduling mechanisms, and completely independent for broker actions. However, efficient management of task execution inside multicluster distributed network very complex because of restrictions of cluster system organization and network infrastructure [1]. Key question when simulating parallel programs using multiclustered distributed platforms is to distribute resources, i.e. such fragments of provided resources (for example, processor clock rate, through put in bytes/sec) will be allocated to each process of application using given resources in parallel. One of implementation approaches of such resources distribution is to use low level distribution models such as simulation of network packets [2] and execution of parallel application code on virtualization layer [3] in a way resources distribution will take place automatically (through packet transfer). Unfortunately, given approaches expensive in mainte nance, which is very undesirable for simulation of applications on large scale, especially multi-clustered, platforms with large time/resource expenses. Instead, developed module considers multiserver platform as unified computational resource, consisting of set of independent computational resources/nodes. At present user must specify number of resources to allocate on each cluster by himself through special configuration file of applied user task execution. Further the functionality to compute available resources fragment will be implemented, which allows to automatically define amount of resources required for each process of the user program.

Naturally, clusters participating represent heterogeneous environment, i.e. geographically distributed and self-sufficient for administration. Network connection provided by Internet usually unsafe and unpredictable in respect of performance. Successful completion of the application sent for execution depends of completion of all program tasks assigned on different clusters. Most unproductive computational cluster becomes bottleneck when executing user task in multicluster environment. Execution of the application also has high risk of system failure call inside one of deployed clusters.

3. Computational experiments

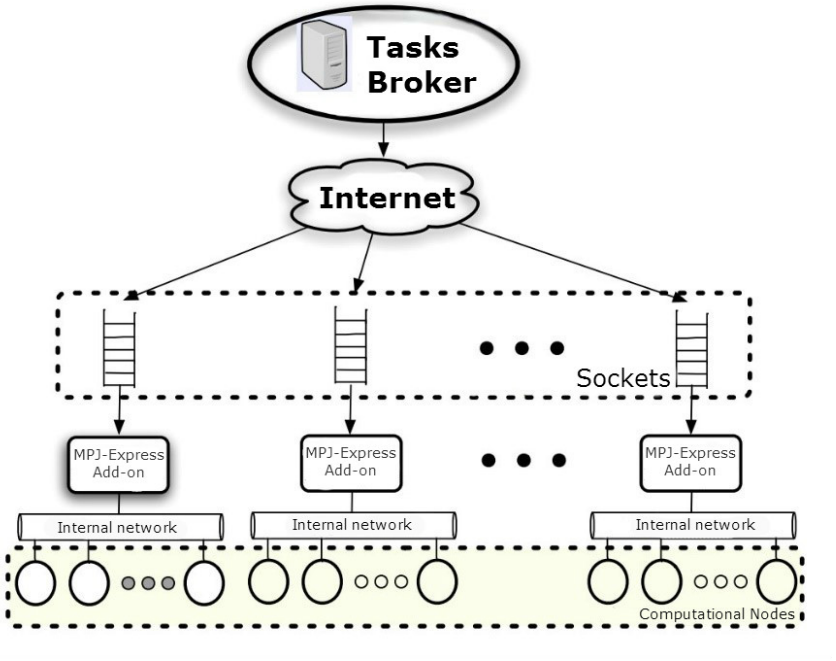

Figure 1 – Architecture of multicluster add-on for MPJ-Express and the User tasks Broker connected to it

Numerical simulation theory of filtration taken as test task. Given task is the standard Laplace equation in two-dimensional (2D) and three-dimensional (3D) case with different boundary coefficients, approximated in finite-difference scheme, solved by red-black strings alternation method in 2D case, and explicit Jacobi method in 3D case. Further part of this paper describes applied task in two- and three- dimensional case, its solution, as well as implementation in sequential and parallel algorithm, which then slightly modified in order to execute in multicluster distributed computational environment.

4. The problem of Heat transfer in 2D domain

Given problem belongs to problems with local synchronization. Let us consider square metallic plate that has fixed temperature on each of its sides except top side. Temperature of internal surface will be dependable of temperature around it on outside borders. Distribution of temperature can be found by dividing the domain into grid of points, $u_(i,j)$. Temperature of the inside point can be taken as an average of four neighbour points. In order to do computations it is convenient to describe boundaries by points adjacent with internal points. Internal points, $u_(i,j$ lie in interval $1 \leq i \leq n$, $1 \leq j \leq n$ (internal domain $[n-1]x[n-1]$). Boundary points at $i = 1, i = m, j = 1, j = n$, and equal to values, corresponding to temperature on the boundary. Temperature of each point can be calculated iteratively for equation

$$u_{i,j} = \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i-1,j}}{4} \tag{1}$$

where $1 \leq i \leq n$, $1 \leq j \leq n$ for fixed number of iterations or while difference between iteration values less than some very small specified number. This iterative equation appears in other

similar problems, for example, with pressure and concentration. Actually, system of linear algebraic equations solving. Given method known as finite difference equations method. Appropriate equation has the following form:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial x^2} = 0 \tag{2}$$

and called two dimensional Laplace equation. Boundary conditions for given problem defined as

$$\begin{array}{l} u\big|_{x=0} = u\big|_{x=L} = u\big|_{y=0} = 0,2 \\ \frac{\partial^u}{\partial y}\Big|_{y=L} = 0 \end{array} \tag{3}$$

Appropriate graphical representation of the solution of the heat transfer in 2D domain problem shown in Figure 2.

Obtained finite difference equation can be solved with different algorithm called red-black string alternation. Given approach allows to get rid of mutual dependencies of parallel threads. In given algorithm, grid of points divided to "red"and "black"points. Given scheme assumes that on each iteration of method execution the grid divides by two sequential stages. At first stage only strings with even indices processed, second stage processes strings with odd indices. Given scheme can be generalized for parallel execution to strings and columns (chess scheme of decomposition) of the computational domain.

5. The problem of heat transfer in 3D domain

The above problem can be expanded for three dimensions, and its solving method takes the average of six neighbour points, two in each dimension. Appropriate Laplace equation has the following form:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2} = 0 \tag{4}$$

Approximation in partial derivatives of given equation similar to two-dimensional version, only two additional points added according to coordinate grid:

$$u_{i,j,k} = \frac{u_{i-1,j,k} + u_{i+1,j,k} + u_{i,j-1,k} + u_{i-1,j,k} + u_{i,j,k-1} + u_{i,j,k+1}}{6} \tag{5}$$

Let us produce from equation (5), the linear equation of the following form:

$$-6u_{i-1,j,k} + u_{i+1,j,k} + u_{i,j-1,k} + u_{i-1,j,k} + u_{i,j,k-1} + u_{i,j,k+1} = 0 \tag{6}$$

Computation domain D specified for approximating Laplace equation within this domain. For this, it is necessary to specify boundary values along all six limiting domain planes,

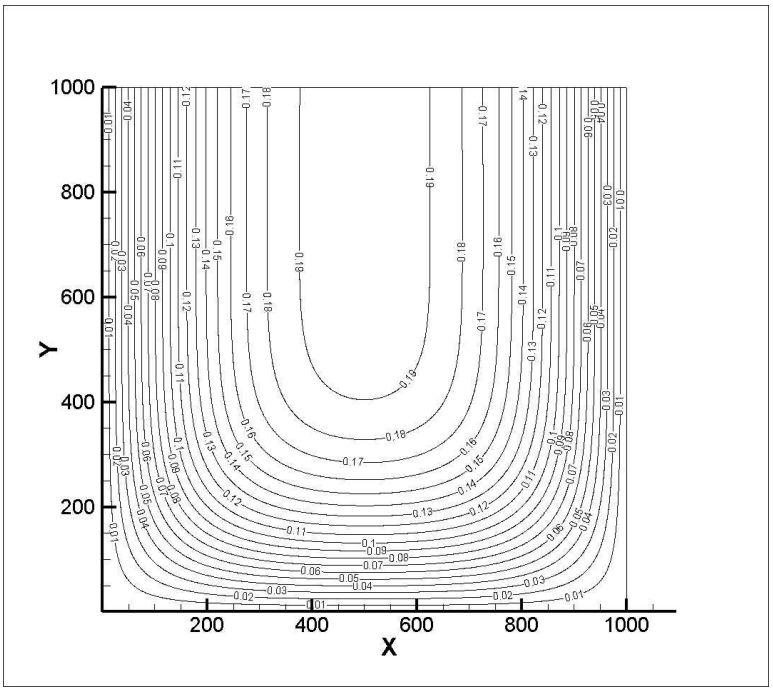

Figure 2 – Graphical representation of the solution of the heat transfer in 2D domain problem with 1000x1000 grid size

in Laplace equation case it is cube. Domain boundaries making impact on temperature distribution within specified cube contour. Obviously, in the center of studied domain the temperature equals to average value of temperature of all cube edges. Therefore, boundary values for solved problem looks like

In the 1D case our model problem becomes:

$$\begin{array}{l} u\big|_{x=0} = u\big|_{x=L} = 0,2 \\ u\big|_{y=0} = u\big|_{y=L} = 0,2 \\ u\big|_{z=0} = 0,2 \\ \frac{\partial^u}{\partial z}\big|_{z=L} = 0 \end{array} \tag{7}$$

Corresponding graphical representation of calculation result of the problem of heat transfer in 3D domain shown in Figure 3. Figure shows three intersecting slices of the solution of simulation domain in each of three directions of diffusion distribution. Parallel implementation of 3D problem. Parallel algorithms for solving Laplace equation for isolated systems can be divided into following groups:

1. Approximation of potentials on the boundary of the finite domain and parallel solution of finite difference scheme [4];

2. Parallel convolution using Fourier transformations [5];

3. Method of local corrections [6].

Explicit Jacobi method for parallel implementation of described above sequential algorithm chosen. Like in sequential case with the Jacobi algorithm solution at each point depends on six neighbour points. Consequently, parallel algorithm using explicit Jacobi method requires numerous steps. Considering equation (5) most important steps include dividing the domain and exchanging messages between neighbour boundary points. Domain D can be divided to internal domains across any of axes. Schematic single dimensional domain of dividing given problem shown in Figure 4. After dividing the domain and distributing it between parallel processes, neighbour points, located on boundaries of two neighbour processes require exchange of boundary values at each iteration. Hence, calculations require local usage of parallel processes.

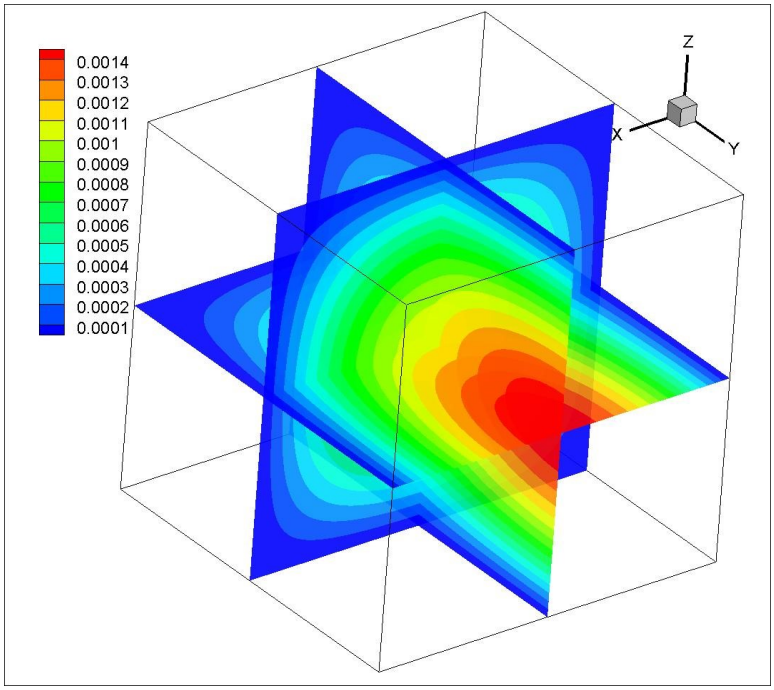

Figure 3 – Graphical representation of the solution of the heat transfer problem in 3D domain

6. Analysis of efficiency of parallel algorithms

Time spent on sequential algorithm execution can be calculated with the following formula:

$$T_1 = t(2mn - n) \tag{8}$$

where t – number of performed iterations, m and n grid size of simulation domain. Time spent on parallel algorithm execution on p processors without taking into account data transfer, can be expressed as

$$T_p = \frac{t(2mn - n)}{p} \tag{9}$$

Then overall speedup equals to

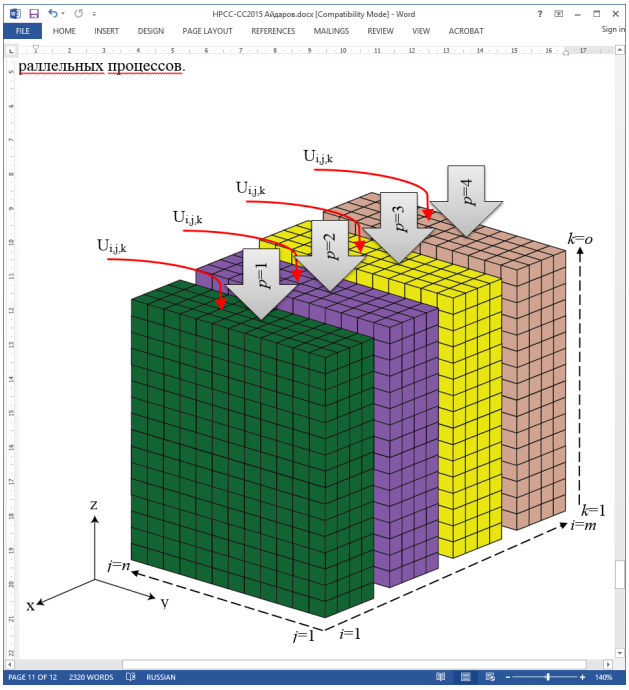

Figure 4 – Schematic 1D decomposition across x axis distributed on p=4 processes to calculate of the heat transfer in 3D domain problem

$$S = \frac{T_1}{T_p} \tag{10}$$

And appropriate efficiency:

$$E = \frac{T_1}{pT_p} \tag{11}$$

However given estimations correspond to ideal (linear) increase of speedup and efficiency which in practice unachievable cause of initialization and data transfer cost. Given costs, arise at exchange of boundary values of neighbour processes. Let us define time spent on data transfer. In order to do this the Hockney model used [7].

Initial data transfer requires following time:

$$T_{C_1} = (p-1)(4\theta_l + \frac{\frac{mn}{p} + n}{\theta_t}) \tag{12}$$

where θ_l – latency, θ_t – network throughput. Data transfer conducted in iteration process affects following time:

$$T_{C_2} = t(p-1)(3\theta_l + \frac{\frac{m}{p} + n}{\theta_t}) \tag{13}$$

where t – number of completed iterations. As a result overall data transfer time can expressed by following formula:

$$T_C = (p-1)(4\theta_l + \frac{\frac{mn}{p} + n}{\theta_t}) + t(p-1)(3\theta_l + \frac{\frac{m}{p} + n}{\theta_t}) \tag{14}$$

Given time depends on number of iterations. Generally, number of iterations less than grid size of calculation domain, mn, which means that time spent on data transfer can be estimated as:

$$T_C = O(mn) \tag{15}$$

Consequently, same estimation can be applied to execution time of the algorithm. If number of iterations will be comparable to m then for execution time of the algorithm another estimation will be valid:

$$T_C = O(m^2 n) \tag{16}$$

7. Results of computation experiments and their analysis

In given section will be given the estimation of numerical simulation results and parallel system performance for single sequential and two parallel implementations of the heat transfer problems algorithm. Numerical profiles of solutions depend on variable u and its gradient shown in Figure 3. Numerical slices of contour looks realistic and correctly represents transition of solutions values from high values to low ones. At more careful discretization obtained solution refine formidably. Results of numerical simulation shows healthy scalability in either for 2D domain case and for 3D domain case. This shows significant efficiency of the explicit method for resources economy as well as parallel execution quality. However, scalability by processes in both cases has its own limit, when efficiency stops its growth at large number of processes, which means than further increase of computational power does not lead to growth of speedup. Nevertheless, there is still space to grow computational resources on other types of distributed memory and new architectures, such as GPU.

Figures 5–8 shows comparisons of speedups and efficiencies in 2D and 3D domain for algorithms of the heat transfer in closed contour for sequential, parallel with use of MPJ-Express library, and parallel with use of MPJ-Express add-on, cases. As a result it can be seen that add-on insignificantly behind in efficiency from standard MPJ-Express library, as well as both parallel implementations show consistent growth when number of parallel processes increase. In conclusion, it can be shown that developed add-on applicable enough as replacement to MPJ-Express library in case of necessity to merge clusters for large scale computations.

8. Conclusion

Given paper describes the new instrument of multicluster organization of distributed computations and gives its detailed description using key components and implementation issues. Given instrument applied to applicable applications, specifically to parallel implementations

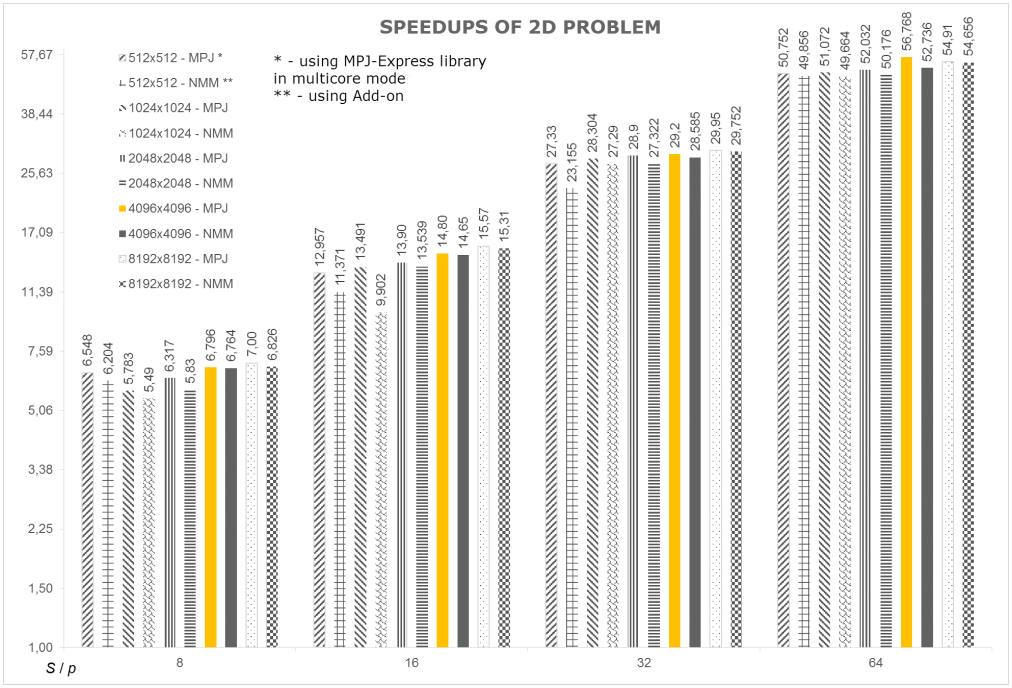

Figure 5 – Comparison of speedups for the heat transfer in 2D domain problem

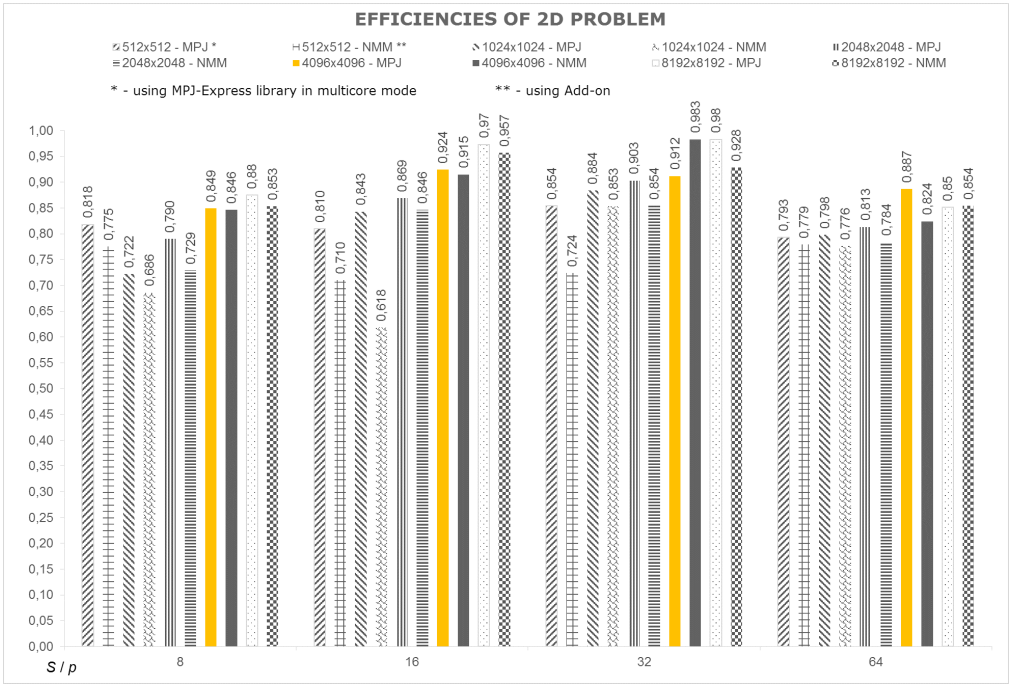

Figure 6 – Comparison of efficiencies in the heat transfer in 2D domain problem

of MPI technology – in particular under its extended Java implementation called MPJ-Express. It has been shown that multicluster interaction model can be successfully adopted to filtration theory problems while insignificantly lacking in performance because of communication delays. At the same time surpassing them in scale of available computational resources.

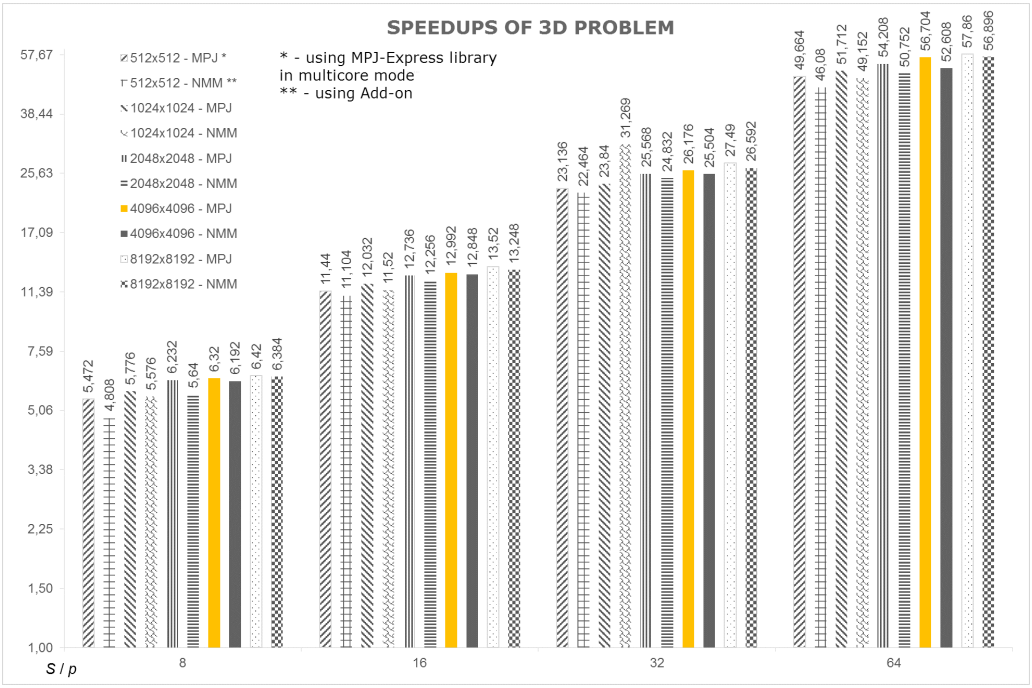

Figure 7 – Comparison of speedups in the heat transfer in 3D domain problem

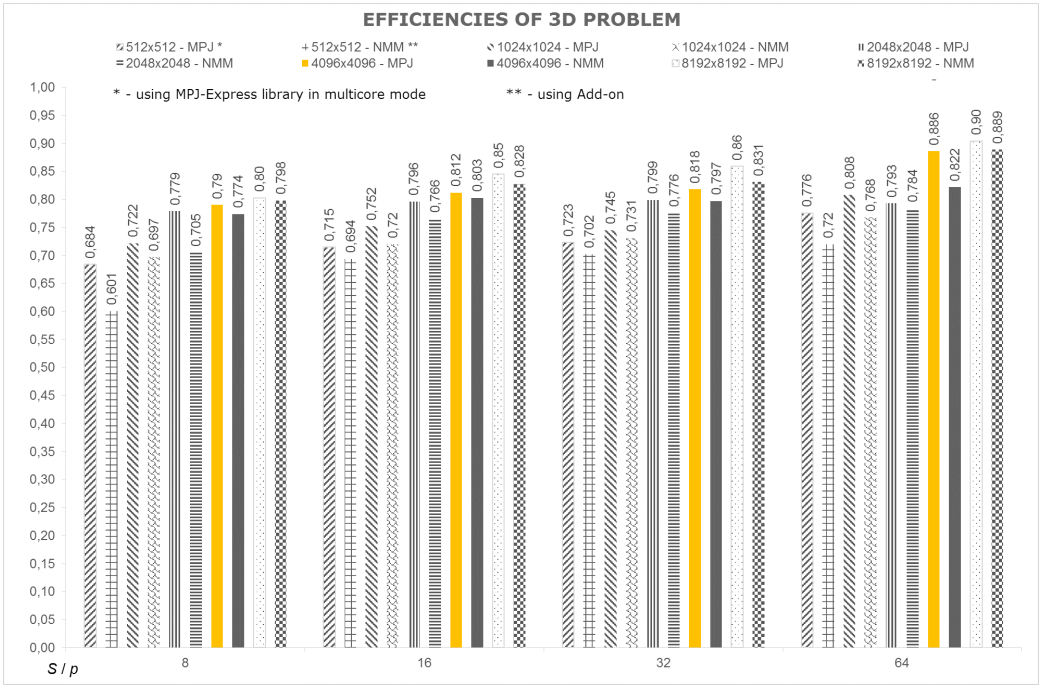

Figure 8 – Comparison of efficiencies of the heat transfer in 3D domain problem

The work presented in given article is a part of grant funding project of the Ministry of Education and Science of the Republic of Kazakhstan (No. 1549/ГФ3 from 15 Apr 2013).

References

[1] *Foster I., Kesselman C., Tuecke S.* The anatomy of the Grid: enabling scalable virtual organizations //International Journal of High Performance Computing Applications. – 2001. – Vol. 15(3). – P.200-222.

[2] *Lamanna M.* The LHC computing grid project at CERN //Nuclear Instruments and Methods in Physics Research A: Accelerators, Spectrometers, Detectors and Associated Equipmen. – 2004. – Vol. 534, No. 1-2. – P. 1-6.

[3] *Pordes R.* The Open Science Grid – its status and implementation architecture //JPCS Proceedings of International Conference on Computing in High Energy and Nuclear Physics (CHEP07). – 2007. – P.1-20.

[4] *Snytnikov N., Vshivkov V., Snytnikov V.* Study of 3D dynamics of gravitating systems using supercomputers: methods and applications //Parallel Computing Technologies. – 2007. – P. 162-173.

[5] *Springel V., Yoshida N., White S.D.M.* GADGET: a code for collisionless and gasdynamical cosmological simulation //New Astronomy. – 2001. – Vol. 6. – P. 79-117.

[6] *Balls G.T., Colella P.* A finite difference domain decomposition method using local corrections for the solution of Poissons equation //Journal of Computational Physics. – 2002. – Vol. 180. – P. 25-53.

[7] *Hockney, R.W., Jesshope, C.R.* Parallel Computers 2: Architecture, Programming and Algorithms. CRC Press. – 1988. – 642 p.