IRSTI 519.687.1

Development of a hybrid parallel algorithm (MPI + OpenMP) for solving the Poisson equation

Kenzhebek Y.G., Al-Farabi Kazakh National University
Almaty, Kazakhstan, E-mail: kenzhebekyerzhan@gmail.com
Baryssova S.B., Al-Farabi Kazakh National University
Almaty, Kazakhstan, E-mail: sandugash.baryssova@gmail.com
Imankulov T.S., Al-Farabi Kazakh National University
Almaty, Kazakhstan, E-mail: imankulov_ts@mail.ru

This article presents the development of a hybrid parallel algorithm for solving the Dirichlet problem for the two-dimensional Poisson equation. MPI and OpenMP were chosen as the technology for parallelization. For the numerical sequential solution of the Poisson equation, an explicit "cross" scheme was used (the Jacobi iterative method). A parallel algorithm was implemented by the method of decomposition of regions, namely, one-dimensional decomposition. In the article in the form of tables and graphs shows the acceleration and efficiency of parallel algorithms using MPI and OpenMP technologies separately and were compared with the acceleration and efficiency of the MPI + OpenMP hybrid algorithm. Also, the choice of the hybrid program architecture is justified and the distribution of data between processes is explained. The results show the effectiveness of using a hybrid algorithm for solving such problems and show the acceleration of time by 1.5-2 times. The presented algorithm was tested on a cluster of the computing center of the Novosibirsk State University for a different number of points in the computational domain (from 64x64 to 1024x1024). The results of the presented work can be applied to the simulation of problems of hydrodynamics, ecology, aerodynamics, the spread of chemical reagents, the propagation of heat and other physical processes.

Keywords: high-performance computing, hybrid technologies, parallel computing, MPI, OpenMP.

MPI және OpenMP технологиялары негізінде Пуассон теңдеуін шешуге арналған гибрид параллельді алгоритм құру

Кенжебек Е.Ғ., Әл-Фараби атындағы Қазақ Ұлттық Университеті
Алматы қ., Қазақстан, E-mail: kenzhebekyerzhan@gmail.com
Барысова С.Б., Әл-Фараби атындағы Қазақ Ұлттық Университеті
Алматы қ., Қазақстан, E-mail: sandugash.baryssova@gmail.com
Иманкулов Т.С., Әл-Фараби атындағы Қазақ Ұлттық Университеті
Алматы қ., Қазақстан, E-mail: imankulov_ts@mail.ru

Бұл мақалада екі өлшемді Пуассон теңдеуі үшін Дирихле мәселесін шешуге арналған гибридті параллельді алгоритм ұсынылған. Параллельдеу технологиясы ретінде MPI және OpenMP таңдалды. Пуассон теңдеуінің сандық жүйелі шешімі үшін айқын «крест» схемасы қолданылды(Якоби итерациялық әдісі). Параллельді алгоритм облысты декомпозициялау әдісі бойынша жүзеге асырылды. Мақалада параллельді алгоритмдердің үдеуі және тиімділігі кестелер мен графиктер түрінде көрсетілген және гибридті алгоритмнің үдеуі және тиімділігімен салыстырулар жүргізілді. Сондай-ақ, гибридті бағдарлама архитектурасын таңдау себебі және процесаралық деректердің үлестірілуі түсіндіріледі. Алынған нәтижелер, гибридті алгоритмді осыған ұқсас есептерде қолдану тиімді екенін және уақыттың жеделдетілуі 1,5-2 есе артатынын көрсетеді. Бұл алгоритм Новосибирск Мемлекеттік Университетінің есептеуіш орталығының кластерінде есептеу облысының әртүрлі нүктелерінде (64x64-тен 1024x1024-ге дейін) сыналды. Жасалған жұмыстың нәтижелерін гидродинамиканың, экологияның, аэродинамиканың, химиялық реагенттердің таралуының, жылу мен басқа да физикалық үрдістердің таралуының мәселелерін модельдеуге қолдануға болады.

Түйін сөздер: жоғары өнімді есептеулер, гибрид технологиялар, параллельді есептеулер, MPI, OpenMP.

Разработка гибридного параллельного алгоритма (MPI+OpenMP) для решения уравнения Пуассона

Кенжебек Е.Г., Казахский национальный университет имени аль-Фараби
Алматы, Казахстан, E-mail: kenzhebekyerzhan.com
Барысова С.Б., Казахский национальный университет имени аль-Фараби
Алматы, Казахстан, E-mail: sandugash.baryssova@gmail.com
Иманкулов Т.С., Казахский национальный университет имени аль-Фараби
Алматы, Казахстан, E-mail: imankulov_ts@mail.ru

В данной статье представлена разработка гибридного параллельного алгоритма для решения задачи Дирихле для двумерного уравнения Пуассона. В качестве технологии для распараллеливания были выбраны MPI и OpenMP. Для численного последовательного решения уравнения Пуассона использовалась явная схема «крест» (итерационный метод Якоби). Параллельный алгоритм был реализован методом декомпозицией областей, а именно одномерная декомпозиция. В статье в виде таблиц и графиков показаны ускорения и эффективности параллельных алгоритмов при использовании технологий MPI и OpenMP по отдельности и были сравнены с ускорением и эффективностью гибридного алгоритма MPI + OpenMP. Так же, обоснован выбор архитектуры гибридной программы и объяснены распределения данных между процессами. Полученные результаты говорят об эффективности использования гибридного алгоритма для решения подобных задач и показывают ускорение времени в 1,5-2 раза. Представленный алгоритм протестирован на кластере вычислительного центра Новосибирского Государственного Университета для различного количества точек расчетной области (от 64x64 до 1024x1024). Результаты представленной работы можно применить для моделирования задач гидродинамики, экологии, аэродинамики, распространение химических реагентов, распространение тепла и других физических процессов.

Ключевые слова: высокопроизводительные вычисления, гибридные технологии, параллельные вычисления, MPI, OpenMP.

1 Introduction

Currently, parallel programming and high-performance computing systems are relevant in various fields of science and technology. High-performance computing uses parallel technologies, such as MPI, OpenMP and CUDA. The greatest productivity can be achieved by creating hybrids of the above technologies. The most high-performance, under a certain range of tasks, will be the merging of CUDA, MPI and OpenMP technologies into a single whole. Therefore, at present the development of hybrid parallel programs is very relevant.

The most complex of the parallel types are hybrid tasks. Particular interest to them is the trend towards the use of multi-core architectures and SMP-clusters for high-performance computing. One of the most effective programming approaches for such clusters is the hybrid, based on the combined use of MPI and OpenMP. The hybrid approach assumes that the algorithm is split into parallel processes, each of which is itself multi-threaded. Thus, there are two levels of parallelism: parallelism between MPI processes and parallelism within the MPI process at the thread level [1].

2 Literature review

There are many works devoted to the research of the MPI / OpenMP approach [2-4]. As practice shows [5-7], by consolidating MPI processes and reducing their number, a hybrid

model can eliminate a number of MPI deficiencies, such as large overhead for message transmission and poor scalability with an increase in the number of processes [8]. However, the performance of a hybrid technology depends very much on the mode of its launch and execution, which determines the ratio of MPI processes and OpenMP threads on one computing node [9]. Chan and Yang [10] argue that MPI can be more favorable with the scalability of clusters. However, OpenMP can favor the speed of shared memory. In addition, the application performance can be affected by the type of problem that is being solved and its size. They show that the effect of MPI communication is the main weakness of this programming model. And finally, they conclude that OpenMP prevails over MPI especially with using a multi-core processor.

It is well known that the implementation of MPI for algorithms in which data is naturally distributed across processes demonstrates very high efficiency (almost linear scaling in time from the number of MPI processes). As it was shown, for example, in [11], in order to achieve comparable performance on one compute node in the case of OpenMP implementation, it is required to implement OpenMP using the concept on which MPI technology is based, but taking into account the presence of shared memory on the node.

Hybrid parallel programming enables to explore the best that is offered by distributed and shared architecture in HPC [12]. Hybrid programming models can match better the architecture characteristics of an SMP cluster, and that would replace message passing communication with synchronized thread-level memory access [13-15]. However, the hybrid programming model can not be regarded as the ideal for all codes [16, 17].

3 Materials and methods

3.1 Purpose of the work and formulation of the problem

The purpose of this work was the creation of a hybrid program that solves the two-dimensional Poisson equation using Jacobi's iterative method in the C ++ programming language using MPI and OpenMP technologies.

The two-dimensional Poisson equation of the form:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -f(x, y) \tag{1}$$

Where x, y are the coordinates; u (x, y) is the desired function; f (x, y) is a continuous function on a rectangular domain with Dirichlet boundary conditions.

$$f\,(x, y) = 2x\,(1 - x) + 2y\,(1 - y) \tag{2}$$

The Dirichlet boundary conditions for the problem under consideration are:

$$\begin{aligned} &u\,(0, y) = 0; \\ &u\,(1, y) = 0; \\ &u\,(x, 0) = 0; \\ &u\,(x, 1) = 0; \end{aligned} \tag{3}$$

3.2 Methods of solution

The most common approach for the numerical solution of differential equations is the method of finite differences. Following this method, the solution domain is represented as a discrete set of points [18]. For sampling internal grid points, a five-point pattern is used, thus using the Jacobi method to perform iterations, the equation takes the following form:

$$u_{i,\,j}^{n+1} = 0.\,25\left(u_{i+1,\,j}^{n} + u_{i-1,\,j}^{n} + u_{i,\,j+1}^{n} + u_{i,\,j-1}^{n} + h^2 f_{ij}\right) \tag{4}$$

Here, $u_{i,\,j}^{n+1}$ is a new layer of Jacobi iterations, and $u_{i,\,j}^{n}$is the previous iteration layer. As shown in Figure 1, to calculate the value of each point of the new layer $u_{i,\,j}^{n+1}$, we need the values of four neighboring points of the previous layer $u_{i+1,\,j}^{n}$, $u_{i-1,\,j}^{n}$, $u_{i,\,j+1}^{n}$, $u_{i,\,j-1}^{n}$.

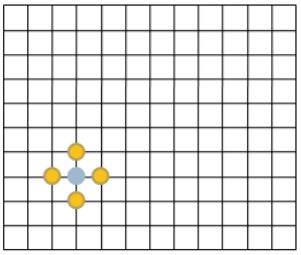

Figure 1: Jacobi method

3.3 Parallelizing a task in MPI

The first thing to solve when parallelizing such tasks is the way to share data between compute nodes. In the problem under consideration for solving the Poisson equation, a tape scheme was used to separate the data. With this division of data, the computing area can be broken down into several horizontal bands. For each process that performs processing of any band, the boundary lines of the previous and next bands were duplicated. The resulting enlarged bands are shown in Figure 2 with dashed frames. Calculations in each band are performed independently of each other and before each new iteration of Jacobi it is necessary to update the duplicated boundary lines.

The exchange of boundary lines between processes consists of two data transfers. First, each process passes its lower boundary to the next process and receives the upper boundary of the line of this process. In the second case, the transfer of boundary lines is performed in the opposite direction, that is, each process passes its upper boundary line to the previous process and receives the lower boundary line from that process. For this operation, combined reception and transmission of MPI_SendRecv messages was used.

3.4 Parallelizing a task in OpenMP

When organizing the problem in question using OpenMP (Open Multi-Processing) technology, the compiler directives were added to the sequential program code. Within this technology, these directives are used to allocate several parallel areas in which processing is performed

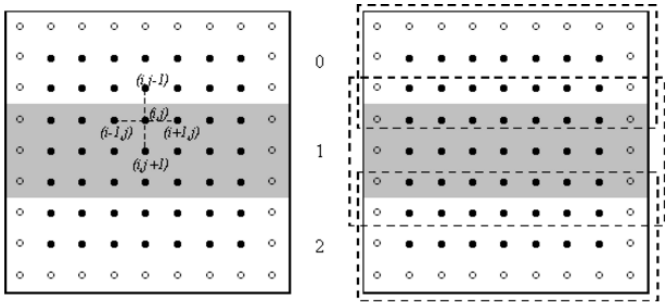

Figure 2: Data distribution

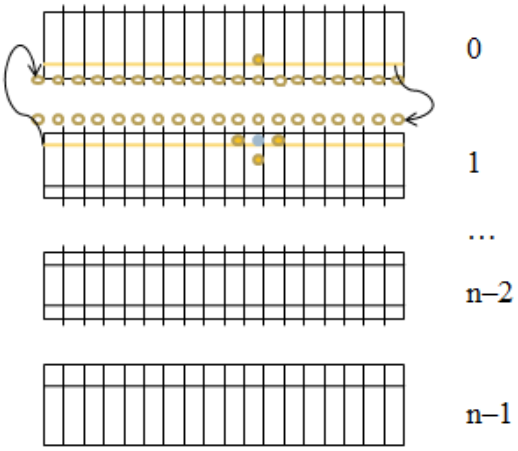

Figure 3: Exchange of boundary lines between processes

using threads. The processors used are multi-core, in order to optimally load all the kernels, there should be several parallel threads in the program. The number of threads that are specified in the program should not exceed the number of cores [19].

3.5 Hybrid method MPI + OpenMP

After creating parallel MPI and OpenMP algorithms, a hybrid method MPI + OpenMP was developed for parallel computation. When creating a hybrid program, a suitable architecture was considered to solve the problem under consideration and the advantages of each technology were taken into account.

MPI technology is used to parallelize a task between SMP nodes for processes, which allows using address spaces and processor computing resources. When performing calculations, each node does not take advantage of the shared memory between the cores, so OpenMP technology is used to parallelize the cores of each of these SMP nodes. MPI was run in a clustered configuration that uses the computational resources of several processors and runs several separate MPI processes on each used node [20].

The distribution of data between the processes was carried out using MPI, and parallel

calculation of data within each of the processes was handled by OMP threads.

The architecture used is shown in Figure 4. This architecture has several modes for working with MPI processes and OpenMP threads. In a hybrid technology, the multiplication of MPI processes and threads specified in the program should correspond to the total number of cores used in the computer system. This is used to correctly load all the kernels. If you exceed a certain number of threads assigned to each MPI process, this can lead to a collision of threads between them. Thus, this will lead to an increase in the execution time of the work.

As shown in Figure 4, for example, if we have two nodes with eight cores on each, the multiplication of the processes and threads used should not exceed 16. This will ensure the balancing of work between processes and threads.

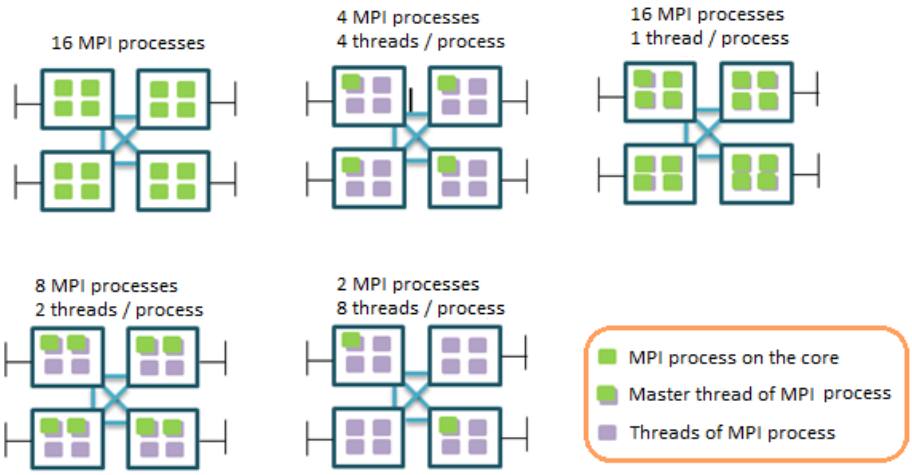

Figure 4: The architecture of the hybrid program MPI + OpenMP

4 Results and discussion

All parallel programs were tested on the Novosibirsk State University (NSU) cluster, which had 2 nodes available. Each node has two 4-core Intel (R) Xeon processor (R) CPU E5-2603 v2 1.80GHz. The tables show the averaged values of time based on several measurements.

Table 1. Time of parallel program execution using MPI technologies(p-process)

Grid size	The execution time of the sequential program, sec	The execution time of the MPI parallel program, sec			
		p = 2	p = 4	p = 8	p = 16
64x64	0,11	0,099	0,092	0,1	0,13
128x128	1,41	0,9	0,59	0,49	0,52
256x256	17,08	9,12	5,1	3,81	2,88
512x512	176,6	90,3	47,3	28,82	191,2
1024x1024	1549,3	781,74	335,7	191,2	135,2

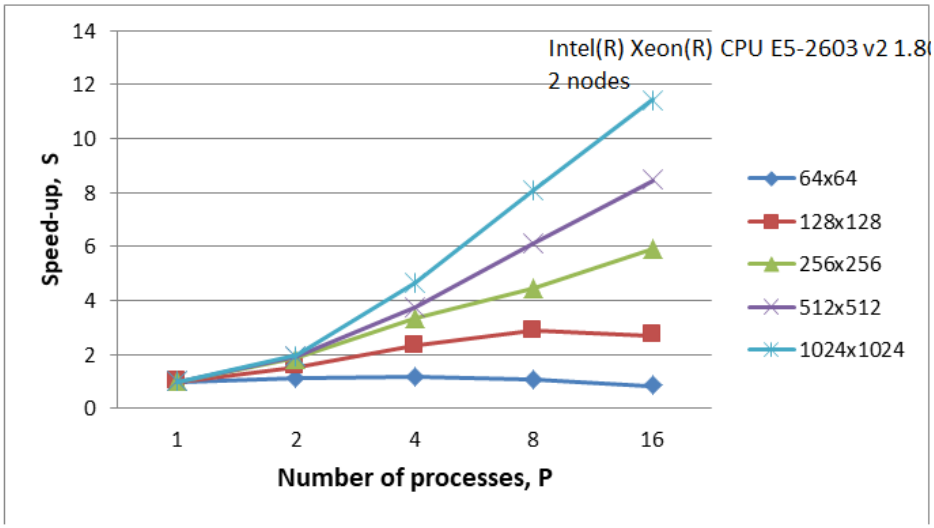

Figure 5: Speed-up of parallel version of the MPI program

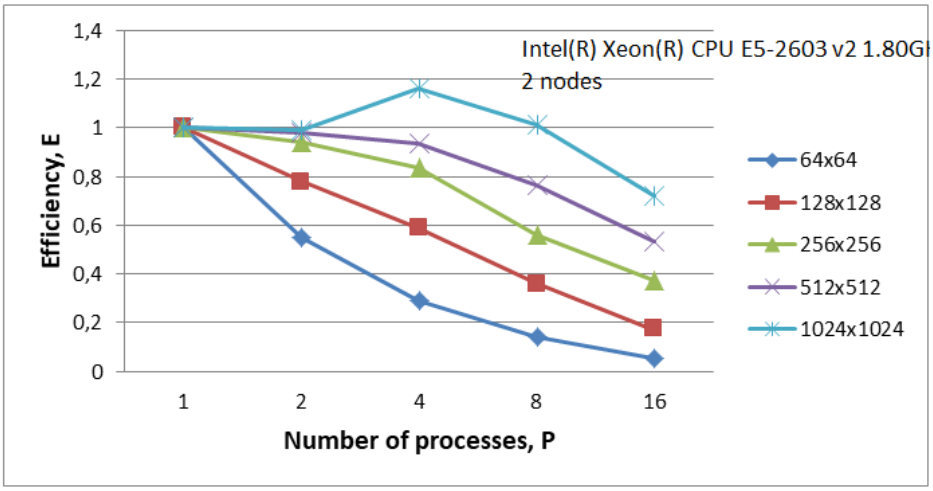

Figure 6: The efficiency of parallel version of the MPI program

Table 2. Time of execution of the parallel program using OpenMP technologies (thread)

Grid size	The execution time of the sequential program,sec(thread=1)	The execution time of the OpenMP parallel program, sec		
		thread=2	thread=4	thread=8
64x64	0,08	0,06	0,04	0,05
128x128	1,03	0,55	0,34	0,26
256x256	12,25	6,2	3,4	1,9
512x512	320,6	187,4	95,5	708,3
1024x1024	2317,3	1381,3	708,3	406,3

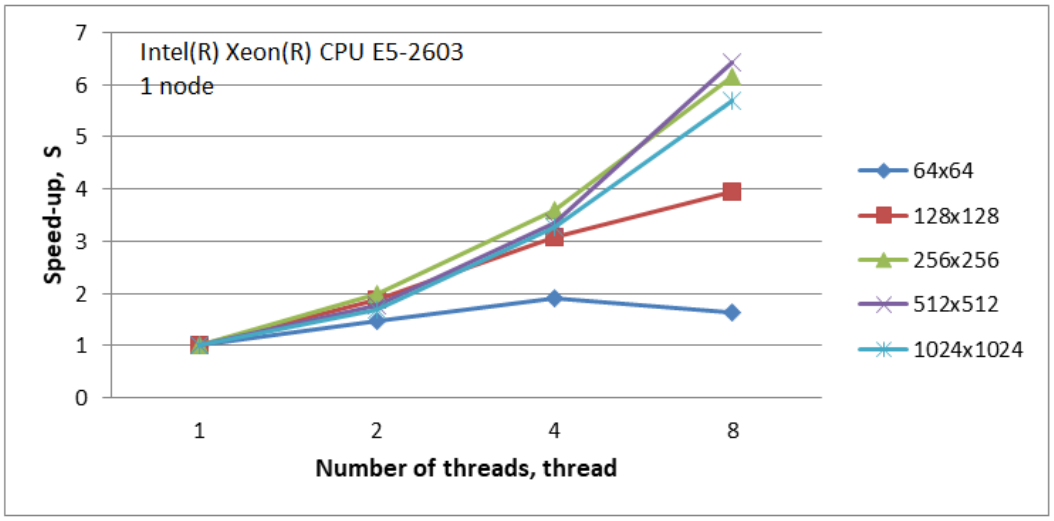

Figure 7: Speed-up of parallel version of the OpenMP program

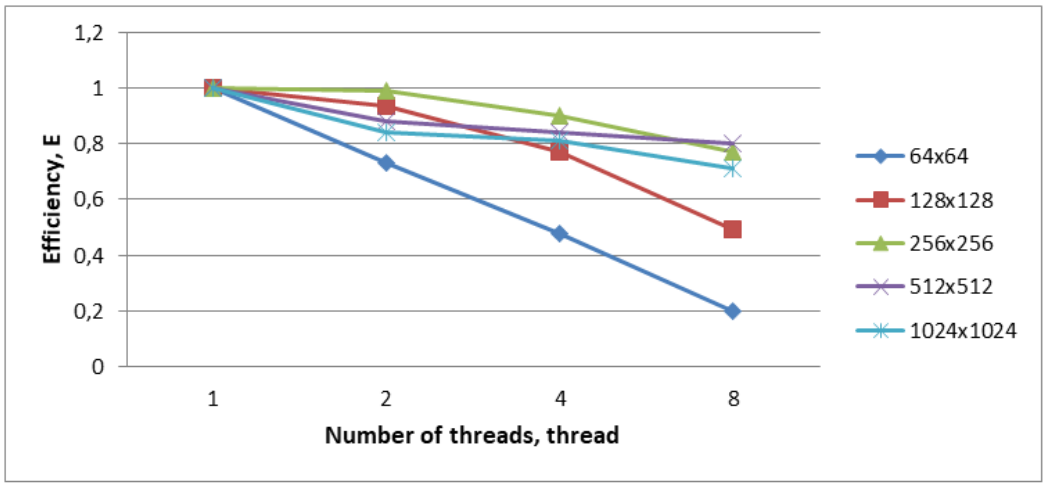

Figure 8: The efficiency of parallel version of the OpenMP program

Table 3. The execution time of the hybrid parallel program using MPI and OpenMP technologies (p-process, thread-thread)

Grid size	The execution time of the sequential program,sec (p=1)	The execution time of the hybrid parallel program MPI + OpenMP, sec			
		p=2, thread=8	p=4, thread=4	p=8, thread=2	p=16, thread=1
64x64	0,11	0,1	0,08	0,095	0,12
128x128	1,41	0,69	0,43	0,42	0,51
256x256	17,08	5,65	2,87	2,81	2,96
512x512	176,6	53,4	23,4	19,45	21,73
1024x1024	1549,3	406,8	150,6	127,1	132,17

Based on the obtained data, the average speed-up and efficiency of parallel programs MPI and MPI + OpenMP (Hybrid) were calculated for solving the Poisson equation. The results are shown in Figures 9 and 10.

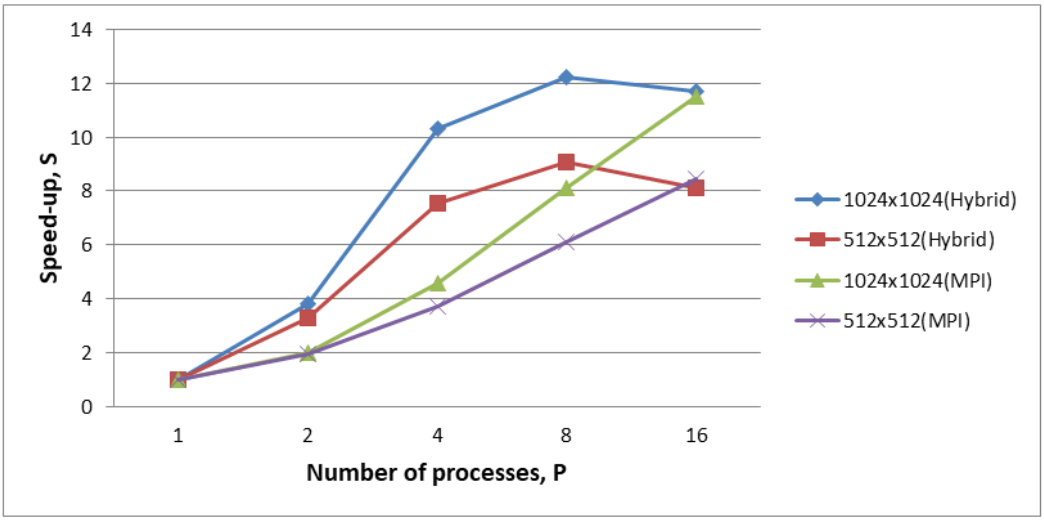

Figure 9: Speed-up for parallel versions of programs

On the efficiency figure of the parallel MPI program, it is noticeable that at the number of 1024x1024 points the efficiency at four processes becomes higher than one. The main reason for this is the total cache size available for the parallel program. With a large number of processors (or cores), one has access to more cache memory. At some point, most of the data fits into the cache memory, which greatly speeds up the calculation. Another way to take this into account is that the more processors are used, the less data that each gets until this part can fit inside the cache of a separate processor. For example, in our case, the cache memory of the Intel (R) Xeon processor (R) CPU E5-2603 has 10 MB of capacity. And in the Poisson problem under consideration there are 3 quantities of the double type. Therefore, when executing the program on one processor, the data did not fit into the cache memory. And

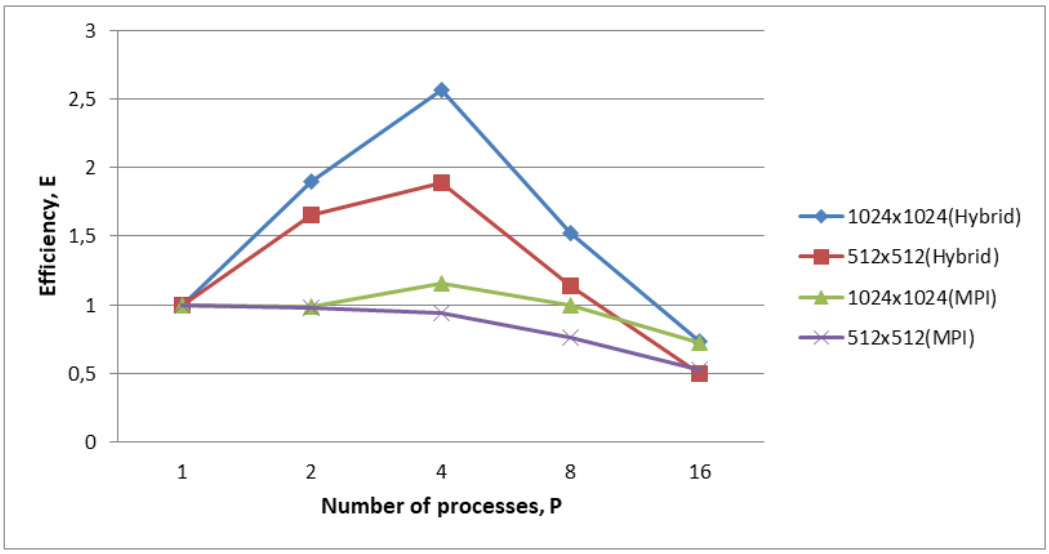

Figure 10: The efficiency of parallel versions of programs

when the program used all the processors in the computer system, each processor had a piece of data that was placed in the cache memory, thereby ensuring speed-up of the calculation.

The resulting superlinear acceleration using hybrid technology MPI + OpenMP is explained by the fact that MPI processes and threads use all the performance of the cores of the computer system. Because the threads assigned to each MPI process effectively use the computing resources of several computers. This hybrid program uses MPI technology to distribute data between processes, and OpenMP separates the iterations of the loop between threads of the program. This ensured the acceleration of work on each compute node.

5 Conclusion

This work was devoted to the development of a hybrid program using MPI and OpenMP technologies. Hybrid implementation of the program is more efficient when working with a large number of nodes and using multi-core processors, because this hybrid technology has the ability to use the cores of several computing nodes. The hybrid program MPI + OpenMP for solving the Poisson equation accelerated the performance of the work by 1.5-2 times in comparison with the MPI program.

References

[1] Gorobets A.V., Sukov S.A., Zheleznyakov A.O. Rasshirenie dvukhurovnevogo rasparallelivaniya MPI+OpenMP posredstvom OpenCL dlya gazodinamicheskikh raschetov na geteregennykh sistemakh [Expansion of two-level parallelization of MPI + OpenMP by means of OpenCL for gas-dynamic calculations on heterogeneous systems]. Vestnik YuUrGU, no. 9 (2009): 76-86.

[2] Martin J., Chorley W., David W. "Performance analysis of a hybrid mpi/openmp application on multi-core clusters."Journal of Computational Science, no. 1 (2010):168–174. doi.org/10.1016/j.jocs.2010.05.001.

[3] Adhianto L., Chapman B. "Performance modeling of communication and computation in hybrid mpi and openmp applications"(12th International Conference on, 2006).

[4] Jin H., Jespersen D., Mehrotra P. "High performance computing using MPI and OpenMP on multicore parallel systems."Parallel computing, no. 37 (2011): 562-575. doi.org/10.1016/j.parco.2011.02.002.

[5] Makris I. "Mixed Mode Programming on Clustered SMP systems"(The University of Edinburgh, 2005).

[6] Rane A., Stanzione D. "Experiences in tuning performance of hybrid MPI/OpenMP applications on quad-core systems."Proceedings 10th LCI International Conference on High-Performance Clustered Computing, (2009).

[7] Rabenseifner R., Hager G., Jost G. "Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes"(proc. 17 Euromicro Internat. Conf. on Parallel, Distributed and Network-based Processing, Weimar (2009): 427–436).

[8] Rabenseifner R., Wellein G. "Communication and Optimization Aspects of Parallel Programming Models on Hybrid Architectures."International Journal of High Performance Computing Applications 17 (2003): 49–62.

[9] Kryukov A.P., Stepanova M.M. Effektivnyi zapusk gibridnykh parallel'nykh zadach [Effective launch of hybrid parallel tasks]. Vestnik YuUrGU, no. 3 (2013): 32-48.

[10] Chan M.K., Yang L. "Comparative analysis of openmp and mpi on multi-core architecture."Proceedings of the 44th Annual Simulation Symposium, 11 (2011).

[11] Mitin I., Kalinkin A., Laevsky Y. "A parallel iterative solver for positive-definite systems with hybrid MPI-OpenMP parallelization for multi-core clusters."Journal of Computer Science, no. 3 (2012): 463–468. doi.org/10.1016/j.jocs.2012.08.010.

[12] Diaz J., Munoz-Caro C., Nino A. "A survey of parallel programming models and tools in the multi and many-core era."Parallel and Distributed Systems, IEEE Transactions on, 23(8):1369–1386, 2012.

[13] Drosinos N., Koziris N. "Performance comparison of pure mpi vs hybrid mpi-openmp parallelization models on smp clusters."Proceedings of the 18th International, 15 (2004).

[14] Chow E., Hysom D. "Assessing performance of hybrid mpi/openmp programs on smp clusters"(2001).

[15] Hager G., Jost G., Rabenseifner R. "Communication characteristics and hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes."Proceedings of Cray User Group Conference, (2009).

[16] Cappello F., Etiemble D. "Mpi versus mpi+openmp on the ibm sp for the nas benchmarks."(In Supercomputing, ACM/IEEE 2000 Conference, 2000).

[17] Smith L., Bull M. "Development of mixed mode mpi / openmp applications."Scientific Programming, 9 (2001): 83–98.

[18] Ryndin E. A. Metody resheniya zadach matematicheskoi fiziki [Methods for solving problems of mathematical physics]. Taganrog: Izd-vo TRTU, 2003.

[19] Antonov A. S. Parallel'noe programmirovanie s ispol'zovaniem tekhnologii OpenMP [Parallel programming using OpenMP technology]. Moscow : Izd-vo MGU, 2009.

[20] Akhmed-Zaki D.Zh., Borisenko M.B. Razrabotka vysokoproizvoditel'nykh prilozhenii s ispol'zovaniem gibridnykh tekhnologii parallel'nykh vychislenii [Developing high-performance applications using hybrid parallel computing technologies]. Almaty: NII Institut KazNU (2013): 7.