IRSTI 50.41.25 https://doi.org/10.26577/JMMCS-2019-3-m4

Parallel implementation of Thomas algorithm for the 2D heat equation

Kenzhebek Y.G., Al-Farabi Kazakh National University
Almaty, Kazakhstan, E-mail: kenzhebekyerzhan@gmail.com
Imankulov T.S., Al-Farabi Kazakh National University
Almaty, Kazakhstan, E-mail: imankulov_ts@mail.ru
Matkerim B., Al-Farabi Kazakh National University
Almaty, Kazakhstan, E-mail: bazargulmm@gmail.com
Akhmed-Zaki D.Zh., University of International Business
Almaty, Kazakhstan, E-mail: darhan_a@mail.ru

In this paper was considered a parallel implementation of the Thomas algorithm for the 2D heat equation. MPI was chosen as the technology for parallelization. The numerical solution of the two-dimensional heat conduction problem was solved using the two step iteration process of alternating direction implicit method (ADI). The Thomas algorithm is simple to implement a sequential program, but difficult to parallelize due to dependent data transfers. When applying this method to solve the 2D heat equation, there is a need to implement Thomas algorithm along each direction x-axis and y-axis. It was implemented the parallelization of this problem using the Yanenko method using 1D and 2D data decomposition. In particular, in 2D data decomposition, the Yanenko method was used along each x-axis and y-axis direction. In the article the speedup and efficiency of parallel programs using 1D and 2D data decomposition were shown in the form of tables and graphs. The presented algorithm was tested on a cluster of the computing center of Novosibirsk State University for a different number of points in the computational domain (from 512x512 to 4096x4096). The obtained test results are presented and analyzed, on the basis of which the features of the used decompositions are described.
Key words: high-performance computing, Thomas algorithm, Yanenko method, parallel computing, ADI method, MPI.

2D жылуөткізгіштік теңдеуі үшін қуалау әдісін паралельді жүзеге асыру

Кенжебек Е.Ғ., Әл-Фараби атындағы Қазақ ұлттық университеті
Алматы қ., Қазақстан, E-mail: kenzhebekyerzhan@gmail.com
Иманкулов Т.С., Әл-Фараби атындағы Қазақ ұлттық университеті
Алматы қ., Қазақстан, E-mail: imankulov_ts@mail.ru
Мәткерім Б., Әл-Фараби атындағы Қазақ ұлттық университеті
Алматы қ., Қазақстан, E-mail: bazargulmm@gmail.com
Ахмед-Заки Д.Ж., Халықаралық бизнес университеті
Алматы қ., Қазақстан, E-mail: darhan_a@mail.ru

Бұл жұмыста 2D жылуөткізгіштік теңдеуін қуалау әдісімен параллельді есептеуді жүзеге асыру қарастырылған. Параллельдеу технологиясы ретінде MPI хат жіберу интерфейсі қолданылды. Жылуөткізгіштіктің екі өлшемді есебінің сандық шешімі ADI әдісі арқылы шешілді. Бұл әдіс тізбекті бағдарламаны іске асыруда қарапайым болып табылады, алайда деректерді жіберудің тәуелді болуына байланысты параллельдеу қиын болып табылады. Осы әдісті қолдану кезінде 2D жылуөткізгіштік теңдеуін шешу үшін x және y ось бағыттары бойынша қуалауды орындау қажеттілігі туындайды. Зерттеу жұмысында таңдалған мысал есеп үшін Яненко әдісін қолдануда деректерді 1D және 2D декомпозициялау арқылы параллельденуіне сипаттама берілген. Атап айтқанда, 2D декомпозициясы кезінде, қуалаудың әрбір x және y ось бағыты бойынша Яненко әдісі қолданылды. Мақалада 1D және 2D декомпозициялары бойынша параллельді алгоритмдердің үдеуі және тиімділігі кестелер

мен графиктер түрінде көрсетілген. Ұсынылған алгоритм Новосибирск Мемлекеттік Университетінің есептеу орталығының кластерінде есептеу облысының әртүрлі нүктелері үшін (512x512-ден 4096x4096-ға дейін) сыналды. Тестілеу нәтижелері алынған және талдау жасалынған, сонымен қатар пайдаланылған декомпозициялардың ерекшеліктері сипатталған.
Түйін сөздер: жоғары өнімді есептеулер, қуалау әдісі, Яненко әдісі, параллельді есептеулер, ADI әдісі, MPI.

Параллельная реализация метода прогонки для 2D уравнения теплопроводности

Кенжебек Е.Г., Казахский национальный университет имени аль-Фараби
г. Алматы, Казахстан, E-mail: kenzhebekyerzhan@gmail.com
Иманкулов Т.С., Казахский национальный университет имени аль-Фараби
г. Алматы, Казахстан, E-mail: imankulov_ts@mail.ru
Маткерим Б., Казахский национальный университет имени аль-Фараби
г. Алматы, Казахстан, E-mail: bazargulmm@gmail.com
Ахмед-Заки Д.Ж., Университет международного бизнеса
г. Алматы, Казахстан, E-mail: darhan_a@mail.ru

В данной работе была рассмотрена параллельная реализация метода прогонки для 2D уравнения теплопроводности. В качестве технологии для распараллеливания был выбран интерфейс передачи сообщения MPI. Численное решение двумерной задачи теплопроводности был решен с помощью метода продольно-поперечной прогонки. Метод прогонки является простым в реализации последовательной программы, но сложно распараллеливаемым из-за зависимых пересылок данных. При применении данного метода для решения 2D уравнения теплопроводности возникает потребность реализации прогонки вдоль каждого направления x и y оси. В работе приведена описания распараллеливания данной задачи с помощью метода Яненко при использований 1D и 2D декомпозиции данных. В частности, при 2D декомпозиции, вдоль каждого направления прогонки был использован метод Яненко. В статье в виде таблиц и графиков показаны ускорения и эффективность параллельных программ при использовании 1D и 2D декомпозиции данных. Представленный алгоритм протестирован на кластере вычислительного центра Новосибирского Государственного Университета для различного количества точек расчетной области (от 512x512 до 4096x4096). Полученные результаты тестирования представлены и проанализированы, на основании чего описаны особенности использованных декомпозиций.
Ключевые слова: высокопроизводительные вычисления, метод прогонки, метод Яненко, параллельные вычисления, метод ADI, MPI.

1 Introduction

Currently, modeling of processes using the numerical solution of differential equations is finding wider application in various branches of science. Since, the development of computer technology and numerical methods contributes to the solution of such equations. The most common methods reduce a differential problem to a system of linear algebraic equations (SLAE). There are various Thomas algorithm types such as direct sweep, inverse sweep and two-sided sweep to solve SLAE systems.

The Thomas algorithm is a direct method and attracts with its ease of implementation in a sequential solution. The appearance and development of computing systems using multi-core processors and graphics accelerators, actualizes the task of parallelizing Thomas algorithm.

In this paper was described the numerical solution of the heat equation and parallelization of this problem for 1D and 2D decomposition using the Yanenko method, at the second stage

of which the Thomas algorithm was used. The features of their implementation for working on a computer system with parallel processes are also given.

2 Literature review

Currently, many problems describing physical processes are reduced to the need for a numerical solution of systems of linear algebraic equations (SLAE). There are many works on the solvability and convergence of difference schemes, among which the work [1] can be noted. Methods such as the Thomas algorithm and the cyclic reduction method are used to find solutions to such SLAEs. The cyclic reduction method is more difficult to implement, but it is less affected, compared to Thomas algorithm, by rounding errors [2].

There are currently many articles on this subject. Among them, the following works can be distinguished: in [3, 4], a combination of parallel cyclic reduction [5] and the Thomas algorithm was proposed to ensure parallelism and computational complexity. In [6], a computational solver based on the SPIKE algorithm [7, 8] was proposed. In [9], parallel algorithms were formulated and analyzed for solving SLAEs using the counter-running method. Also, the method proposed by N.N. Yanenko [10], which allows to reduce the original system with a large number of unknowns to a system with the number of unknowns equal to the number of processors. Such a system, consisting of parametric boundary processor points, is solved by the Thomas algorithm. It is also possible to apply such methods as the counter-sweep method, the parallel-cyclic reduction method. Also known is the parallel pipeline method [11, 12] for solving many three-diagonal systems.

In [13, 14, 15] was shown the application of the cyclic reduction method for implementation on a computer system with graphic accelerators, and in [16] was considered a solver based on PCR. A hybrid method was developed in [17], which includes the Thomas algorithm and parallel cyclic reduction. In [18], two-level parallelization of the Thomas algorithm (on shared memory using OpenMP and on distributed memory using MPI) was considered to solve three-diagonal systems that arise when modeling two-dimensional and three-dimensional physical processes. Another option for parallelizing the Thomas algorithm is the dichotomy method proposed by Terekhov [19]. The essence of the method is to divide the original area in half at each step. For the two-dimensional and three-dimensional cases, a parallel matrix sweep algorithm is used [20]. A parallel version of the alternating direction method is also known [21]. In [22], a parallelization method for the Thomas algorithm on hybrid computers using accelerators was considered. And also, in this work, the Yanenko method, the parallel pipeline method, and various methods for the second stage of the Yanenko method are described in detail. In [23], a comparison of the parallel pipeline method and the Yanenko method was shown, at the second stage of which the Thomas algorithm was used, the results of parallelization efficiency are also presented.

3 Materials and methods

3.1 Parallelization of Thomas algorithm

The parallelization of Thomas algorithm which was proposed by N. Yanenko called parametric sweep method. The parametric sweep method of N. Yanenko was implemented by distributing

a system of grid equations where at the boundary of processor elements are distinguished so-called parametric unknowns [10]. Consider a system of linear equations of the following form:

$$\begin{aligned} a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i,\ i = 1, ..., n-1, \\ b_0 x_0 + c_0 x_1 = d_0,\ a_n x_{n-1} + b_n x_n = d_n. \end{aligned} \tag{1}$$

Let each processor have the same number of points m=N/size, where N is the number of unknowns and size is the number of processors. Thus, only part of the equations of system (1) with numbers from (j-1)*m+1 to j*m, where j is the processor number, will be located on the processor with number j. Denote x_{j*m} by z_j and we will look for a solution to system (1) in the following form:

$$x_{(j-1)*m+i} = u_i z_{j-1} + v_i z_j + w_i,\ j = 1, ..., size \tag{2}$$

where u,v,w are solutions of the following systems:

$$\begin{aligned} a_i u_{i-1} + b_i u_i + c_i u_{i+1} = 0,\ u_{(j-1)*m} = 1,\ u_{j*m} = 0; \\ a_i v_{i-1} + b_i v_i + c_i v_{i+1} = 0,\ v_{(j-1)*m} = 0,\ v_{j*m} = 1; \\ a_i w_{i-1} + b_i w_i + c_i w_{i+1} = d_i,\ w_{(j-1)*m} = 0,\ w_{j*m} = 0; \\ j = 1, ..., size,\ i = (j-1)*m+1, ..., j*m-1. \end{aligned} \tag{3}$$

The solutions of these three systems (3) can be found by Thomas algorithm, and independently on each processor. We will call this stage of solving this problem - the stage of finding pre-solutions. In the equations with numbers j*m from system (1), we substitute combinations (2) instead of x. Thus, we obtain a system of three-diagonal equations for finding z_j having the following form:

$$A_j z_{j-1} + B_j z_j + C_j z_{j+1} = D_j,\ j = 1, ..., size-1 \tag{4}$$

with coefficients:

$$\begin{aligned} &B_0 = b_0 + c_0 u_1,\ C_0 = c_0 v_1,\ D_0 = d_0 - c_0 w_1, \\ &A_j = a_{j*m} u_{j*m-1},\ B_j = a_{j*m} v_{j*m-1} + b_{j*m} + c_{j*m} u_{j*m+1},\ C_j = c_{j*m} v_{j*m+1}, \\ &D_j = d_{j*m} - a_{j*m} w_{j*m-1} - c_{j*m} w_{j*m+1},\ j = 1, ..., size-1, \\ &A_{size} = a_{size*m} u_{size*m-1},\ B_{size} = b_{size} * m + a_{size*m} v_{size*m-1}, \\ &D_{size} = d_{size*m} - a_{size*m} w_{size*m-1}. \end{aligned}$$

We will call this stage the stage of finding boundary processor solutions. The dimension of this system of equations is equal to the number of processors [15].

After finding the boundary-processor solutions, we restore the final solution by the formula (2).

The Yanenko method consists of three stages: 1) finding the pre-solutions of u, v, w, 2) finding the boundary-processor solutions of z_j, 3) restoring the solutions of x_j.

This computational algorithm has a high degree of parallelism, since the first and third stages are performed in parallel on each processor, but the system solution for parametric

unknowns (4) is performed sequentially by only one processor and requires communication between MPI processes.

In this paper, the Thomas algorithm was used to solve the second stage of the Yanenko method, which is consisting of boundary processor points. The formulas for the Thomas algorithm are shown as follows:

$$\alpha_0 = -\frac{c_0}{b_0},\ \beta_0 = \frac{d_0}{b_0}$$
$$\alpha_i = -\frac{c_i}{b_i + \alpha_i\alpha_{i-1}},\ \beta_i = \frac{d_i - \alpha_i\beta_{i-1}}{b_i + \alpha_i\alpha_{i-1}},\ i = 1, ..., size - 1,$$
$$x_{size} = \frac{d_{size} - \alpha_{size}\beta_{size-1}}{b_{size} + \alpha_{size}\alpha_{size-1}},\ x_i = \alpha_i x_{i+1} + \beta_i,\ i = size - 1, ..., 0.$$

3.2 Implementation of parallelization of the Thomas algorithm

The implementation on distributed memory computing system was performed using the MPI standard. As we know, when solving two-dimensional problems using the method of ADI, many systems of three-diagonal equations will be arise. It is used the Yanenko method to solve such systems, and at the second stage of which the Thomas algorithm was used. This task was performed with one-dimensional and two-dimensional data decomposition.

When using one-dimensional decomposition, the grid area is divided into horizontal stripes (Figure 1).

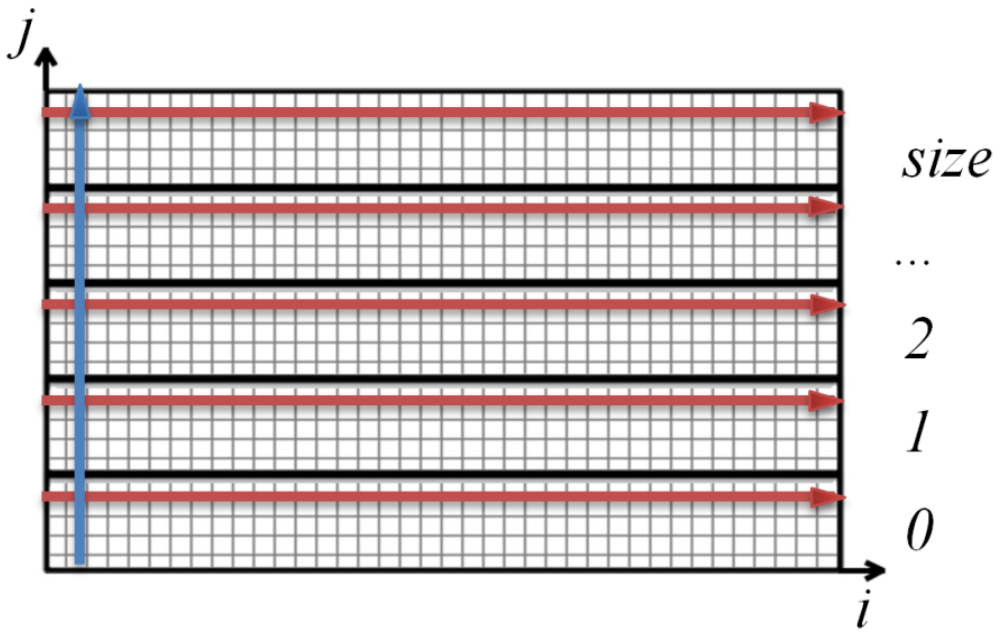

Figure 1: One-dimensional decomposition

The implementation of the ADI method consists of two steps of iteration process. The Thomas algorithm was used along each x-axis and y-axis. In Figure 1, the first step of iteration process is indicated in red color, and the second step of iteration process is indicated in blue

color. In the first step, the method is performed for each row independently on each MPI process. Therefore, when using the Thomas algorithm at the first step, the boundary data exchange between MPI processes are not required. However, at the second step, because of the data decomposition between MPI processes it is used Yanenko method.

An example of decomposition along each direct sweep is shown in Figure 2. The boundary processor parametric unknowns, which are determined in the second stage of the Yanenko method, are indicated in green [24].

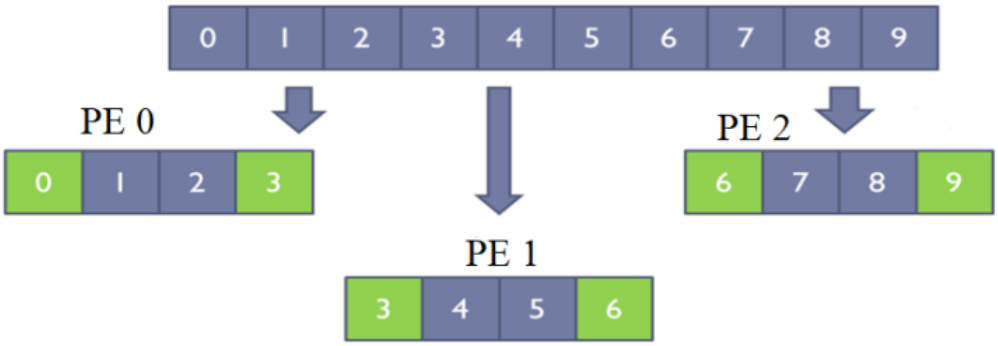

Figure 2: An example of decomposition between three MPI processes

The second stage of the Yanenko method requires additional MPI communication. In order to complete the second stage, the boundary elements of the pre-solutions u,v,w were collected from each MPI process using the MPI_Gather function. After sequentially finding the boundary-processor solutions on the root process using the Thomas algorithm, we broadcast the data using the MPI_Bcast function. Then, each process having its own parametric boundary unknowns and elements of the pre-solutions u,v,w and calculate final solutions by equation (2) independently in parallel.

In a distributed memory computing system using two-dimensional data decomposition, the grid area is divided into blocks (Figure 3). For this, we have constructed a two-dimensional MPI Cartesian topology.

In Figure 3, the first step is indicated in red color, and the second step is indicated in blue color. At the first and second step, where it is necessary to solve the three-diagonal equations along the grid lines i and j, it is used the Yanenko method. That means the Yanenko method was used along each x-axis and y-axis direction of the Thomas algorithm. When using two-dimensional data decomposition, GridSize is the number of data blocks along each x-axis and y-axis direction.

3.3 Numerical experiments

As a test problem, the two-dimensional heat equation is written in the following form:

$$\frac{\partial U}{\partial t} = \alpha \left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + f(x, y) \right)$$

here

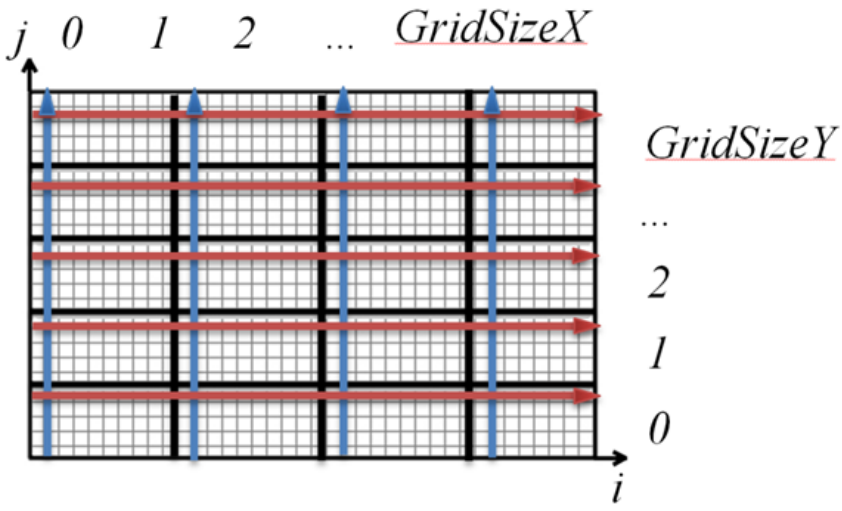

Figure 3: Two-dimensional decomposition

$\alpha = 1,\ f(x, y) = -3.$

Initial conditions:

$$U(x, y, 0) = x^2 + y^2$$

Border conditions:

$$U(0, y, t) = y^2 + t$$
$$U(1, y, t) = 1 + y^2 + t$$
$$U(x, 0, t) = x^2 + t$$
$$U(x, 1, t) = 1 + x^2 + t$$

4 Results and discussions

A two-dimensional heat equation testing was carried out with an implicit approximation scheme, where 512, 1024, 2048, 4096 points were taken in each grid direction.

Testing was carried out on the computing cluster of Novosibirsk State University (NSU) [25]. In each computing node there were 2 Intel Xeon CPU E5-2603 (4 computing cores in each).

For software configuration, Intel C ++ Compiler 15.0.2 (optimization level –O3) was used. The following Table 1,2 show averaged times based on several measurements.

Figures 4 and 6 show the speedup of parallel programs for 1D and 2D data decomposition with different numbers of MPI processes. In these figures, we can see that in each process, as the number of points in the problem increases, the speedup increases. However, for 1D decomposition, when using 16 MPI processes, the speedup drops. This is because with an increase in the number of processes, the necessary communication costs increase due to the second stage of the Yanenko method. Therefore, for a 1D decomposition, the maximum of the processes used was 16. And for 2D decomposition, the speedup increases up to 64 MPI

Table 1: Parallel program runtime(sec) for 1D decomposition

Grid Size	Number of processes				
	1	2	4	8	16
512x512	1,26	1,87	1,51	1,75	3,69
1024x1024	6,57	6,33	4,43	4,61	8,04
2048x2048	30,01	22,8	13,63	11,53	15,68
4096x4096	125,01	93,32	54,89	42,92	39,88

Table 2: Parallel program runtime(sec) for 2D decomposition

Grid Size	Number of processes					
	1	4	8	16	32	64
512x512	1,26	1,09	0,78	0,61	0,56	1,12
1024x1024	6,57	3,99	2,64	1,61	1,36	2,27
2048x2048	30,01	16,2	10,26	5,83	3,87	4,93
4096x4096	125,01	65,7	40,9	20,8	12,87	10,7

processes. The reason is when applying 2D decomposition, the number of processes along each grid direction of Thomas algorithm decreases.

Figures 5 and 7 show the efficiency of parallel programs for 1D and 2D data decomposition with different numbers of MPI processes. We may notice that as the number of processes increases, the efficiency decreases. For 1D decomposition, good performance indicators are shown using 2 and 4 MPI processes. For 2D data decomposition, with an increase in the number of MPI processes, performance indicators are more stable than when using 1D decomposition. Therefore, when using 2D data decomposition, the task was launched up to 64 processes, which gives an advantage for reduce the computation time. The efficiency of 2D data decomposition with large grid size is more stable than 1D data decomposition. As we increase MPI processes number for 1D data decomposition with large grid size, the execution time of the second stage of the Yanenko method increases due to data exchanges between processes. For instance, if we launch a task using 16 MPI processes for 1D decomposition, in the second step of ADI, in the second stage of the Yanenko method, communication is performed between all the given 16 MPI processes. And when using 2D decomposition for the same task, communication occurs between 4 MPI processes along per computation direction of Thomas algorithm.

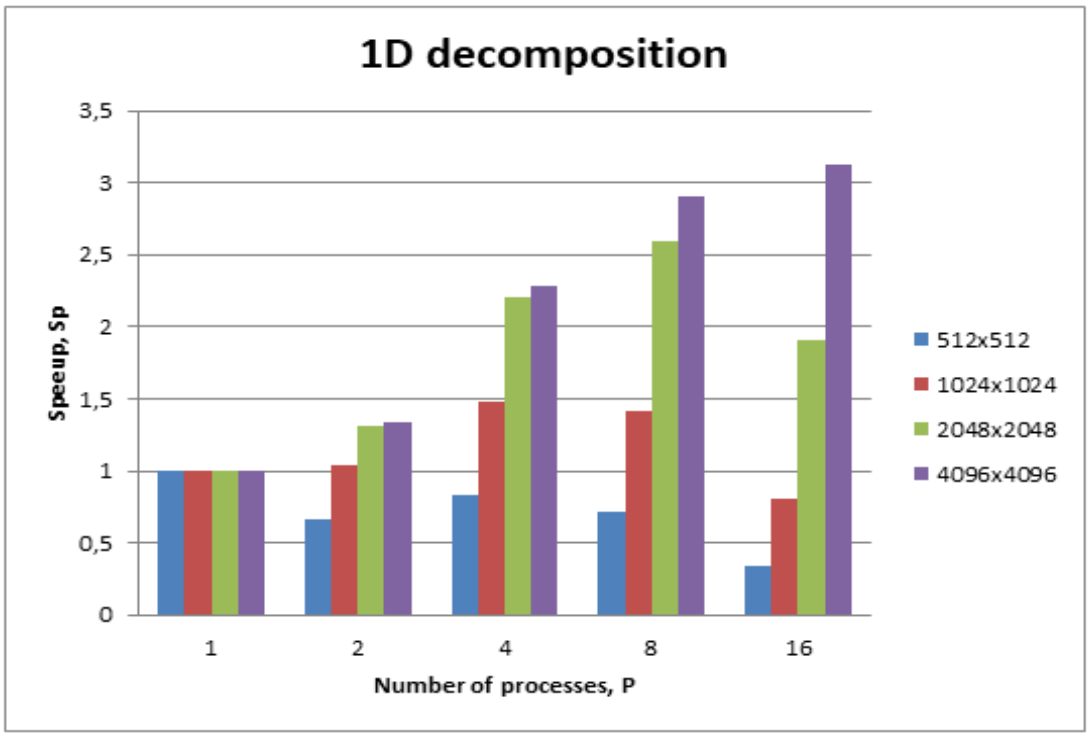

Figure 4: Speedup of a parallel program for 1D decomposition

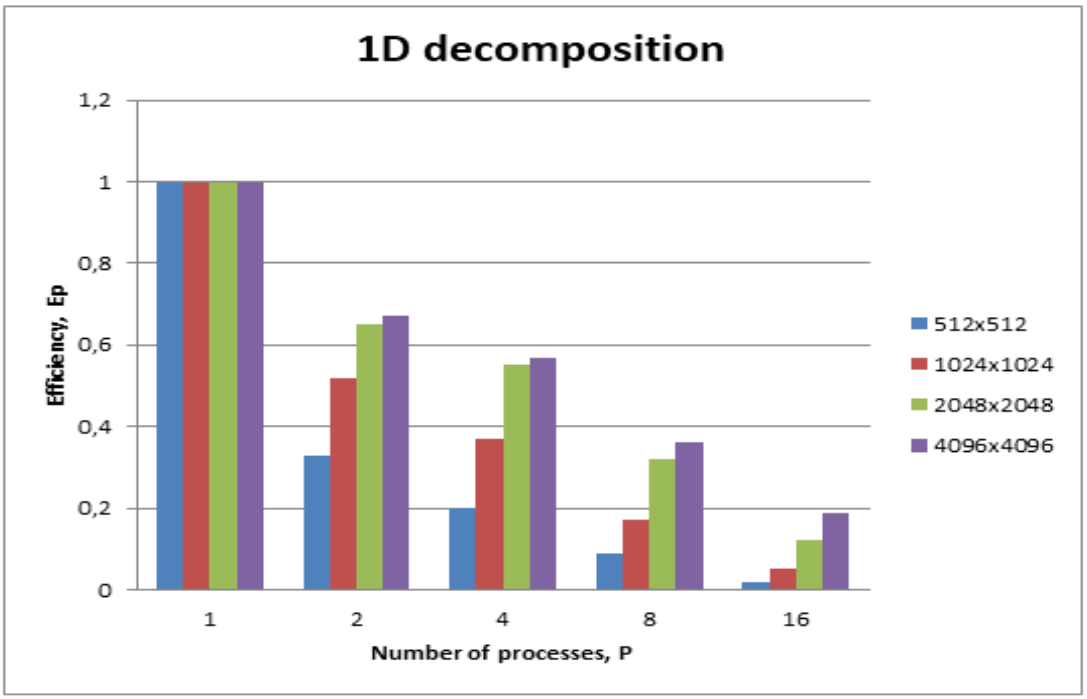

Figure 5: Efficiency of a parallel program for 1D decomposition

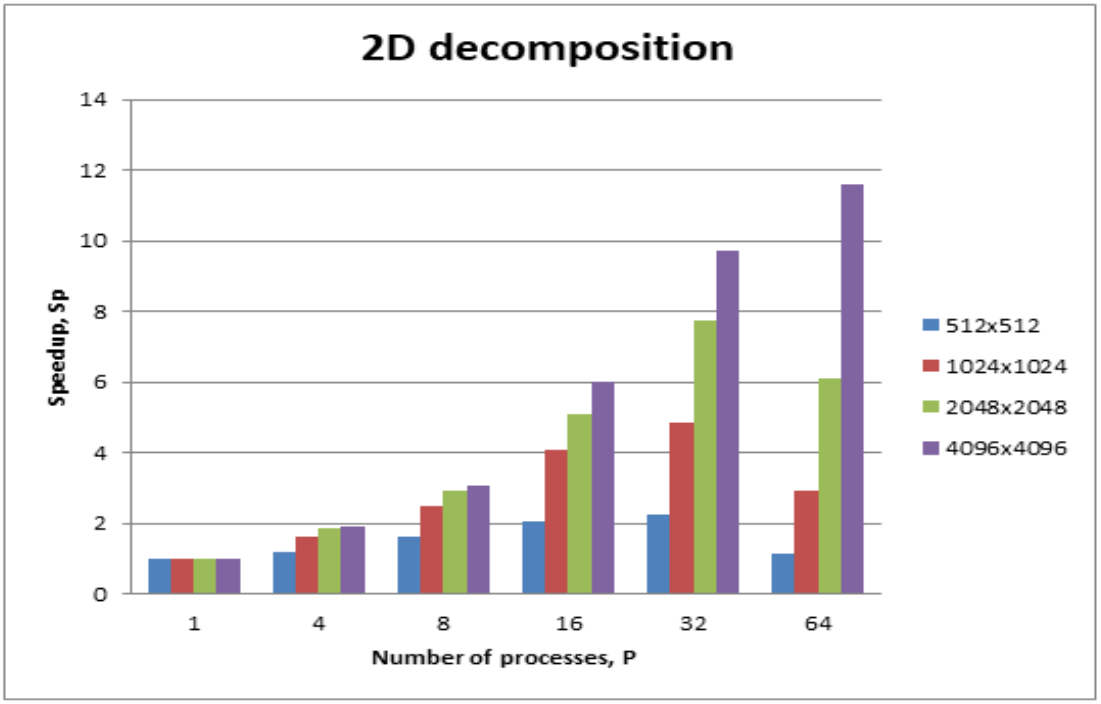

Figure 6: Speedup of a parallel program for 2D decomposition

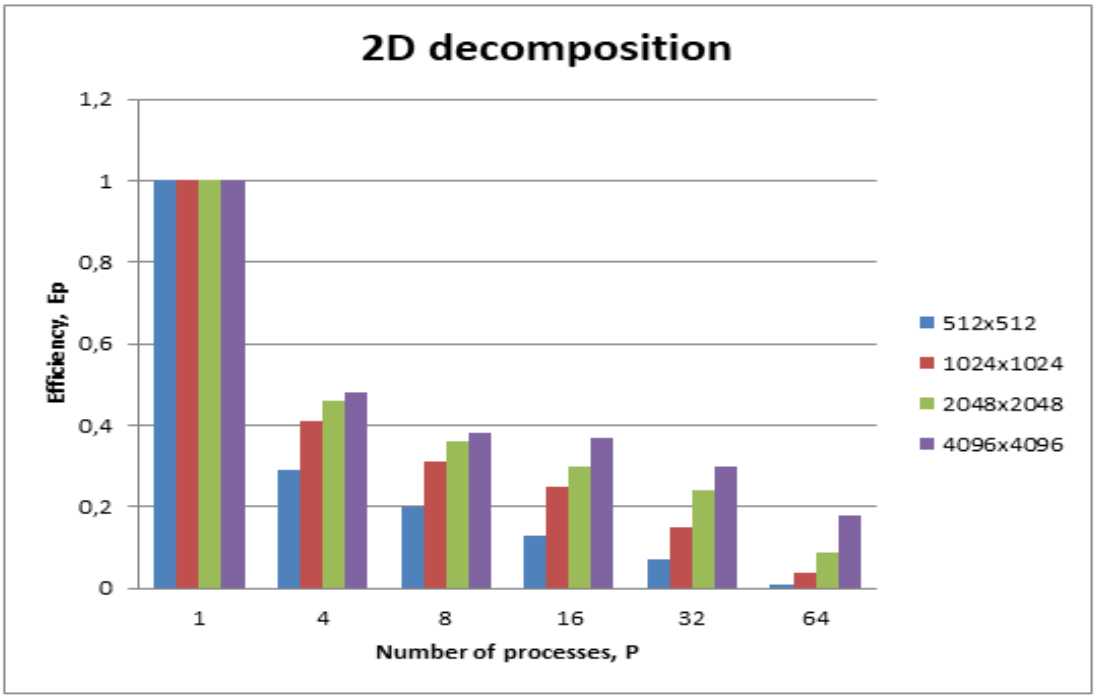

Figure 7: Efficiency of a parallel program for 2D decomposition

5 Conclusion

This paper is devoted to apply Yanenko method as the parallel implementation of the Thomas algorithm for solving the 2D heat equation. The numerical solution of the 2D heat equation was used two step iteration process of the ADI method. Described the parallelization of the Thomas algorithm using 1D and 2D data decomposition. For chosen test task of 2D heat equation for 1D and 2D data decomposition, the speedup and efficiency was compared and analyzed in detail. In particular, in 2D decomposition, the Yanenko method was used along x-axis and y-axis direction separately. Therefore, using this research experience, one can use the Yanenko method as a parallel implementation of the Thomas algorithm to solve the 3D heat equation using the three step iteration process of ADI method in 2D,3D data decomposition.

References

[1] Samarskiy A.A., Nickolayev Ye.S., *Metody resheniya setochnykh uravneniy [Methods of grid equations solving]* , (Moscow: The science, 1987), 130.

[2] Samarskiy A.A., *Vvedeniye v chislennyye metody [Introduction to numerical methods]* , (St. Petersburg: Deer, 2005).

[3] Davidson A., Zhang Y., Owens J., "An auto-tuned method for solving large tridiagonal systems on the GPU" , (Conference Parallel & Distributed Processing Symposium (IPDPS), IEEE International, May 16-20, 2011): 956-965.

[4] Kim H.S., Wu S., Chang L.W., Hwu W.M., "A scalable tridiagonal solver for GPUs." , (In Parallel Processing (ICPP), International Conference, September 13-16, 2011): 444 –453.

[5] Hockney R.W., Jesshope C.R., *Parallel computers: architecture, programming and algorithm* , (Bristol: 1986), 274-280.

[6] Chang L.W., Stratton J.A., Kim H.S., Hwu W.M., "A scalable, numerically stable, highperformance tridiagonal solver using GPUs. " ,(High Performance Computing, Networking, Storage and Analysis (SC), for IEEE Computer Society Press, November 10-16, 2012): 11.

[7] Polizzi E., Sameh A., "A parallel hybrid banded system solver: The SPIKE algorithm " , *Parallel Computing*, vol. 32, no 2 (2006): 177-194.

[8] Polizzi E., Sameh A., "A parallel environment for solving banded linear systems" , *Computers and Fluids*, vol. 36, no 1 (2007): 113-120.

[9] Loganova L.V., Golovashkin D.L., "Parallel'nyy algoritm realizatsii metoda vstrechnykh tsiklicheskikh progonok dlya dvumernykh setochnykh oblastey[Parallel algorithm for implementing the counter cyclic sweep method for two-dimensional grid regions]" , *Journal of Computing Technologies*, vol. 16, no 4 (2011): 64-71.

[10] Yanenko N.N., Konovalov A.N., Bugrov A.N., Shustov G.V., "Ob organizatsii parallel'nykh vychisleniy i "rasparallelivanii" progonki [On the organization of parallel computing and "parallelizing" sweeps]" , *Computational methods of continuum mechanics*, vol. 9, no 7 (1978): 139-146.

[11] Sapronov I.S., Bykov A.N., "Parallel'no-konveyyernyy algoritm [Parallel pipeline algorithm]" , *Atom*, no 44 (2009): 24-25.

[12] Povitsky A., "Parallelization of the pipelined Thomas algorithm" , *Journal of Parallel and Distributed Computing*, vol. 59, no 1 (1999): 68-97.

[13] Sengupta S., Harris M., Zhang Y., Owens J.D., "Scan primitives for GPU computing" , (Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware, August 04-05, 2007): 97-106.

[14] Goddeke D., Strzodka R., "Cyclic reduction tridiagonal solvers on GPUs applied to mixed-precision multigrid" , *IEEE Transactions on Parallel and Distributed Systems* vol. 22, (2011): 22-32.

[15] Davidson A., Owens J.D., "Register packing for cyclic reduction: A case study" , (Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units, March, 2011): 65-79.

[16] Egloff D., "High performance finite difference PDE solvers on GPUs" , *Wilmott magazine*, (2010): 32-40.

[17] Zhang Y., Cohen J., "Fast tridiagonal solvers on the GPU" , (Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, January 09-14, 2010): 127-136

[18] Fedorov A.A., Bykov A.N., "Metod dvukhurovnevogo rasparallelivaniya progonki dlya resheniya trekhdiagonal'nykh lineynykh sistem na gibridnykh EVM s mnogoyadernymi soprotsessorami [A method of two-level parallelization of sweeps for solving tridiagonal linear systems on hybrid computers with multi-core coprocessors]" , *Computational methods and programming*, vol. 17 (2016): 234-244.

[19] Terekhov A.V., "Parallel dichotomy algorithm for solving tridiagonal system of linear equations with multiple right-hand sides" , *Parallel Computing*, vol. 36 (2010): 423-438.

[20] Akimova Y.N., "Rasparallelivaniye algoritma matrichnoy progonki [Parallelization of matrix Thomas algorithm]" , *Mathematical modeling*, no 9 (1994): 61-67.

[21] Ilyin V.P., "Parallel'nyye neyavnyye metody peremennykh napravleniy [Parallel implicit methods of alternating directions]" , *Journal of Computational Mathematics and Physics*, vol. 37, no 8 (1997): 899-907.

[22] Bykov A.N., Yerofeyev A.M., Sizov Ye.A., Fedorov A.A., "Metod rasparallelivaniya progonki na gibridnykh EVM [Parallel Thomas method on hybrid computers]" , *Computational methods and programming*, vol. 14, no 2 (2013): 43-47.

[23] Ilyin S.A., Starchenko A.V., "Rasparallelivaniye skhemy pokomponentnogo rasshchepleniya dlya chislennogo resheniya uravneniya teploprovodnosti [Parallelization of the component-wise splitting scheme for the numerical solution of the heat equation]" , (Parallel Computing Technologies, Proceedings of the international scientific conference, August 31 - September 4, 2015):399-402.

[24] Kenzhebek Y.G., "Algoritm Yanenko dlya odnomernogo uravneniya teploprovodnosti [Yanenko algorithm for the one-dimensional heat equation]" , *Herald of KazNITU*, no 2 (2019): 512-516.

[25] "Using software" , Information and Computing Center of Novosibirsk State University, accessed July 13, 2019, http://nusc.nsu.ru/wiki/doku.php.