IRSTI 50.41.25 https://doi.org/10.26577/JPCSIT.2023.v1.i1.03

D.V. Lebedev[1,*], **V.A. Perepelkin**[2]

[1]Astana IT University, Astana, Kazakhstan
[2]Institute of Computational Mathematics and Mathematical Geophysics, Novosibirsk, Russian Federation
*e-mail: danil.lebedev.0881@gmail.com

OPTIMIZING THE OPERATION OF FRAGMENTED PROGRAMS BASED ON TRACES

Abstract. This article analyzes a fragmented algorithm to solve systems of linear algebraic equations and optimize its operation. A sequential and parallel algorithm of the generalized minimal residuals method is described. A fragmented algorithm has been developed and implemented in the LuNA language. The LuNA system automatically constructs parallel programs, thereby allowing researchers not to waste efforts on developing a parallel program, in particular, allows not to take into consideration the communication processes between computer nodes and information about which nodes store distributed data, perform calculations. To optimize the operation of the fragmented program, the trace playback module was used, using data from the previous operation of the program to bypass additional costs in the further construction of parallel programs. The program was tested on the cluster of the Information and Computing Center of Novosibirsk State University and the results were analyzed. With the help of trace playback, at best, an acceleration of 27 times was obtained.

Key words: high-perfomance computing, LuNA, fragmented programming, parallel computing.

1 Introduction

High-performance computing plays a very important role in modern science. The development of high-performance computing allows scientists to solve scientific and applied problems in various fields. Solving such problems in large cell counting grids requires large resources, including time costs. The use of supercomputers can significantly speed up problem solving when using numerical methods. Technologies such as OpenMP [1,2], MPI [3,4], OpenCL [5], CUDA [6,7] and fragmentary programming [4,8] are used to speed up scientific calculations.

In the field of solving scientific numerical modeling problems using a supercomputer, there is a problem of constructing a parallel algorithm that effectively solves a given applied numerical algorithm. The word efficiency here refers to the execution time of a parallel program, the costs of using memory, the balance of the load on the resources of a given solver, and much more. Achieving the effective performance of a parallel program is a difficult task for the user, that is, the programmer. For this purpose, the user must take into consideration the features of the incoming data, the application task, the calculator, as well as balance the load between the calculation nodes between the time, perform communications along with the calculation. The difficulty of creating such a program is especially evident when the computing is not homogeneous, but heterogeneous (containing GPUs or other types of computing.

LuNA is a fragmentary programming system designed to automate the process of implementing numerical modeling problems on a supercomputer. In this system, the above properties are automated. In [4], one can see how the authors used the LuNA system for the modeling problem, and in [9], examples of using dense linear algebra in problems are shown. But in these works, the resulting acceleration is less than the acceleration of a parallel program using the MPI library. In [10], steps were taken to optimize the fragmented program. In the following work [11], traces were used for optimization. In the traces, you can store information about the construction of the program, which will optimize the design of the program in the following launches.

In this paper, the development of a fragmented algorithm and the optimization of a fragmented program were considered. The article discusses the problems of fragmented programming and the optimization option. To solve the problem of modeling the movement of oil, a linear solver and its parallel algorithm are considered.

2 Method and sequential algorithm

This article discusses an algorithm for solving systems of linear algebraic equations (SLAE) with large sparse matrices. There are many methods for solving SLAE, but not all of them are suitable for systems with sparse matrices. For example, direct methods like the Gauss method, LU decomposition, should not be used for systems with large sparse matrices, since when our system is transformed, an overflow of the matrix may occur. Simple iterative methods, like the method of simple iteration, Jacobi, Seidel, should also not be used for systems with large matrices, since these methods converge slowly. For such cases, Krylov-type methods are used. The most widely used of them is the method of generalized minimal residuals (GMRES) [12]. This method is suitable for SLAE with large sparse and asymmetric matrices.

The method consists of several steps. The method relies on Arnoldi iterations to find a vector minimizing the discrepancy from the Krylov subspace. The Arnoldi iteration algorithm can be represented as follows:

$$w_j = Av_j$$

$$h_{ij} = (w_j, v_j), w_j = w_j - h_{ij}v_i$$

$$h_{j+1,j} = \|w_j\|$$

$$v_{j+1} = w_j / h_{j+1}$$

$$j = 1,2,3,\dots,m \quad , \quad i = 1,2,3,\dots,j$$

As a result of this step, we get the following output data: V_m – orthonormal basis of Krylov subspace, H_m – an upper Hessenberg matrix whose elements are equal to the orthogonalization coefficients. The approximate solution is as follows:

$$x_m = x_0 + V_m y_m$$

where the vector y_m can be found as a solution to a linear least squares problem of size $(m+1) \times m$, where $m \ll n$:

$$y_m = \underset{y}{\arg\min} \|\beta e_1 - \bar{H}_m y\|$$

Next, the method suggests that the Hessenberg matrix H_m be reduced to a triangular form using Givens rotations. After that , you can easily solve the minimization problem in the following form:

$$\bar{H}_m y = \beta e_1$$

3 Fragmented programming

LuNA (Language for Numerical Algorithms – a fragmentary programming system [13]. It aims to automate the implementation of numerical modeling problems on a supercomputer. In it, the application program is composed of a set of data fragments (DF) and computational fragments (CF), and the fragmented structure of the program is preserved during the calculations. This method of parallelization allows you to program at a higher level and get rid of some of the difficulties of system programming. The description of the fragmentary algorithm is platform-independent, and settings for a particular calculator are provided by the LuNA runtime system.

In the LuNA system, the user describes the calculation algorithm without binding to resources within the datalow-model limits. Such a description is called a fragmentary algorithm (FA). The distinguishing property of FA is its data immutability, i.e., one-time ownership, and the absence of reverse effects in the computational modules that implement the calculations. The implementation of the FA implies the assignment of the DF and CF to the computing resources and the choice of the order of execution of the CF in such a way that it does not contradict the information dependencies. Figure 1 describes how FA works. What we can see from there is that the outgoing DFs are calculated from the incoming DFs until all CFs are completed [14].

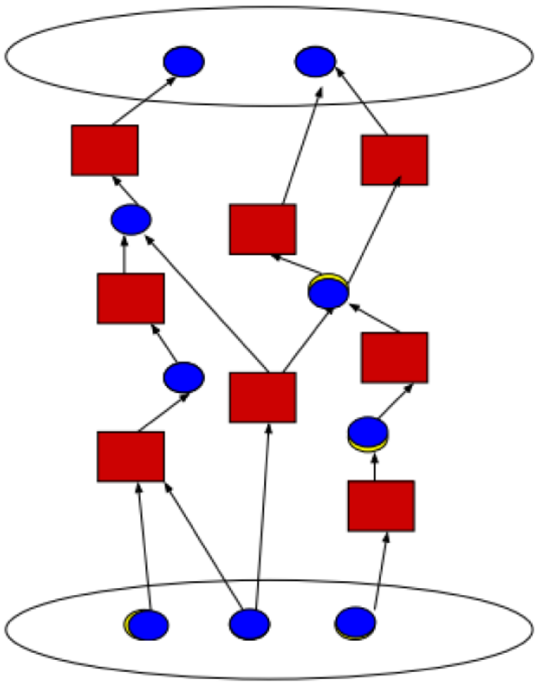

Figure 1 – Description of FA, blue circles represent DF, and red rectangles represent CF

The LuNA system can run a fragmented program in parallel in multithreaded mode in common memory conditions and in multiprocessor or multithreaded modes in distributed memory computing. The goal of this project is to automate the creation of efficient parallel programs that implement some kind of numerical algorithms for supercomputers [13].

4 Fragmented algorithm

The parallelization resource is various matrix and vector operations within the iteration of the method. There is an obvious data dependency between different iterations, so it will not be possible to perform several iterations on different processes in parallel. Instead, an approach was chosen with the distribution of calculations within each iteration.

A fragmented algorithm was developed based on a parallel algorithm. Therefore, first we need to focus on the parallelization of the algorithm. The main calculations of the GMRES method fall on matrix-vector algorithms. For example, the next step

$$x_m = x_0 + V_m y_m$$

means matrix multiplication by vector and vector addition. These operations are easily paralleled with the distribution of data across processes.

The parallel algorithm was implemented using the MPI standard.

Further, a fragmented algorithm was developed based on a parallel algorithm and implemented in the LuNA system. The operation of a fragmented algorithm can be characterized by the operation of a fragmented calculation of the following part of the method:

$$h_{ij} = (w_j, v_j)$$

To find the scalar product, here the vectors w_j and v_j are divided by n the number of DF. Then, for each pair of two vectors, separate CFs are performed, as a result of which temporary DF is generated. A separate CF collects temporary DFs, performs operations on them and generates a DF with a coefficient h_{ij} at the output. DFs that will no longer be used can be erased from memory.

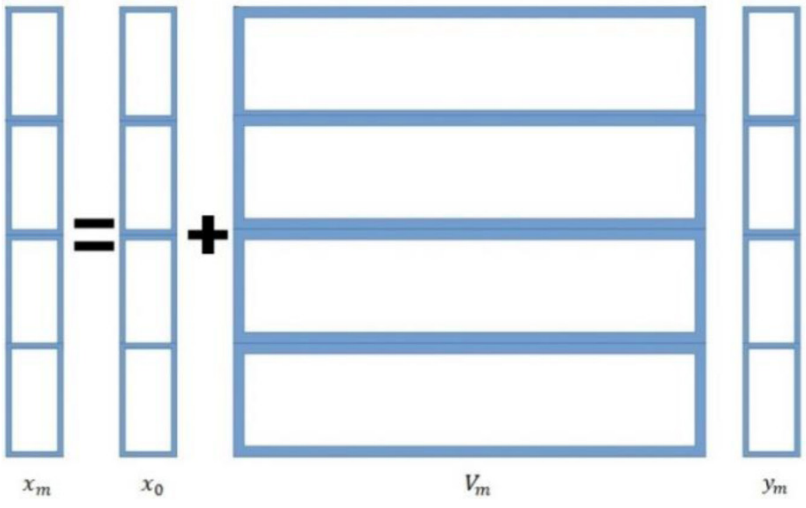

Figure 2 – Data decomposition during parallel execution of the algorithm

5 Numerical experiments and results

A model describing the motion of a multiphase multicomponent liquid in a porous medium was chosen for numerical experiments. This article is a continuation of the work [15]. For a more detailed acquaintance with the mathematical model and the results of parallel execution of the algorithm, you can refer to this article.

Studies that have been conducted up to this time have shown a noticeable slowdown in the operation of the fragmented algorithm program with an increase in the number of processes [15]. This was explained by a lot of additional costs when designing a parallel program. As it was written earlier, the fragmented programming system finds the relationships in the DF and CF and automatically constructs a parallel program. Along with this, it takes into account which CF is performed after which CF, which DF should serve as input and output data, in which computing nodes they should be located and what communications should occur between the nodes simultaneously. The task itself is NP complete and it takes quite a lot of time at this stage. And with the increase in the number of parallel MPI processes, the options for designing parallel programs increases significantly. To get rid of these costs, a track playback module was developed. After a single construction and execution of a parallel program, files with traces are generated, in which information about the locations and movements of DF and CF is recorded. Using this data in the next construction of a parallel program will speed up the work.

The program was tested on different sizes of the matrix of a system of linear equations. The tests were conducted on a cluster of the Information and Computing Center of Novosibirsk State University, on one node of which there are two 6-core Intel Xeon X5670 processors with a clock frequency of 2932 MHz and 24 GB of RAM. The results are displayed in the following table.

Table 1 – Comparative results of the program

Size of matrix A	600x600		1200x1200		2400x2400		4800x4800	
Number of processes	without trace	with trace	without trace	with trace	without trace	with trace	without trace	with trace
2	84.21	7.925	312.917	8.366	1159.906	105.401	5527.718	1517.654
4	81.646	4.367	312.031	25.417	1321.223	185.1	6467.069	3078.793
8	120.273	4.404	490.472	31.84	2009.271	309.636	10585.324	5061.385

The table shows the time in seconds. On each size of the matrix, you can see two columns with the results. The first is the usual execution of a fragmented program, the second is the execution of a fragmented program designed with the trace playback. From the table you can see that on small sizes, the trace playback allows you to get a very good acceleration compared to a conventional fragmented program. For example, an acceleration of 27 times was obtained on a 600x600 matrix. But with an increase in size, this acceleration drops to two times on a matrix of size 4800x4800. This can be attributed to the reality that on small sizes, the main time costs are spent on the previously specified loads. The larger the size, the longer the runtime of the program, without considering the processes that are stored in the traces.

6 Conclusion

In this paper, a fragmented algorithm for solving the problem of modeling fluid movement was considered. The work is a continuation of the work previously carried out by the research group. Parallel and fragmented algorithms of the method of generalized minimal residuals were developed. The fragmented program was realized in the LuNA system. Writing a fragmented program allows developers not to think about the system part of parallel programs, but it was found that a fragmented program runs slower than a parallel program on MPI. Tests were carried out using the track playback module, which allowed in some cases to get an acceleration of 27 times.

Acknowledgments

This research was supported by the Grant No. AP14871644 " Development of intelligent and high-performance models for solving enhanced oil recovery (EOR) problems" of Kazakhstan Republic National Engineering Academy.

References

1. Iryanto and P. H. Gunawan, "An OpenMP parallel godunov scheme for 1D two phase oil displacement problem," *2017 5th International Conference on Information and Communication Technology (ICoIC7)*, Melaka, Malaysia, 2017, pp. 1-5, doi: 10.1109/ICoICT.2017.8074664.
2. L.F. Werneck, M.Medeiros de Freitas, H.Guaraldi da Silva Jr, Grazione de Souza, H.P. Amaral Souto. "An OpenMP Parallel Implementation for Numerical Simulation of Gas Reservoirs Using Intel Xeon Phi Coprocessor An OpenMP Parallel Implementation for Numerical Simulation of Gas Reservoirs." *Proceedings of the XXXVII Iberian Latin-American Congress on Computational Methods in Engineering* Suzana Moreira Avila (Editor), ABMEC, Brasılia, DF, Brazil, November 6-9, 2016.
3. Sreekanth Pannala, Ed F. D'Azevedo, Madhava Syamlal, Thomas O'Brien. **"**Hybrid (OpenMP and MPI) Parallelization of MFIX: A Multiphase CFD Code for Modeling Fluidized Beds.Conference." *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC)*, March 9-12, 2003, Melbourne, FL, USA.
4. Akhmed-Zaki, Lebedev, Perepelkin. "Implementation of a three dimensional three-phase fluid flow ("oil–water–gas") numerical model in LuNA fragmented programming system." *The Journal of Supercomputing*. 2016
5. Khramchenkov, E., & Khramchenkov, M. "Numerical Model of Two-Phase Flow in Dissolvable Porous Media and Simulation of Reservoir Acidizing." *Natural Resources Research*, 27(4), 531–537.
6. Zaza, A., Awotunde, A. A., Fairag, F. A., & Al-Mouhamed, M. A. "A CUDA based parallel multi-phase oil reservoir simulator." *Computer Physics Communications*, 206, 2–16.
7. McClure, J. E., Prins, J. F., & Miller, C. T. "A novel heterogeneous algorithm to simulate multiphase flow in porous media on multicore CPU–GPU systems." *Computer Physics Communications*, 185(7), 1865–1874.
8. Malyshkin V.E., Perepelkin V.A. "LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem." *In the Proceedings of the 11-th Conference on Parallel Computing Technologis*, LNCS, vol. 6873, Springer, 2011. — pp. 53–61.

Belyaev, N., Perepelkin, V. "High-Efficiency Specialized Support for Dense Linear Algebra Arithmetic in LuNA System." In: Malyshkin, V. (eds) *Parallel Computing Technologies. PaCT 2021. Lecture Notes in Computer Science()*, vol 12942. Springer, Cham. https://doi.org/10.1007/978-3-030-86359-3_11

Malyshkin, V., Akhmed-Zaki, D. & Perepelkin, V. "Parallel programs execution optimization using behavior control in LuNA system." *J Supercomput* 77, 9771–9779 (2021). https://doi.org/10.1007/s11227-021-03654-2

Malyshkin, V., Perepelkin, V. "Trace-Based Optimization of Fragmented Programs Execution in LuNA System." In: Malyshkin, V. (eds) *Parallel Computing Technologies. PaCT 2021. Lecture Notes in Computer Science()*, vol 12942. Springer, Cham. https://doi.org/10.1007/978-3-030-86359-3_1

Saad Y, Schults M H. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems [J]." *SIAM J Sci Statist Comput*, 1986, 7: 856-869.

Kireev, S. Malyshkin, V., Fujita, H. "The LuNA Library of Parallel Numerical Fragmented Subroutines." *Lecture Notes in Computer Science*. – 2011. – Vol. 6873. – P. 290 –301.

Valkovsky, V.A., Malyshkin, and V.E. *Synthesis of parallel programs and system on the basis of computational models*. Novosibirsk: Nauka, 1988, 128 pp. (In Russian)

N. Kassymbek, B. Matkerim, D. Lebedev, T. Imankulov, D. Akhmed-Zaki. "GMRES Based Numerical Simulation of Multicomponent Multiphase Flow in Porous Media on LuNA Fragmented Programming System," *ECMOR XVII*, Sep 2020, Volume 2020, p.1 – 10. https://doi.org/10.3997/2214-4609.202035201