IRSTI 27.17.00 https://doi.org/10.26577/JPCSIT.2023.v1.i2.07

M. Kakybat [1], D. Tilepbergenov[2], N. Kassymbek[2*]

[1]Kazakh-British Technical University, Kazakhstan, Almaty
[2]Al-Farabi Kazakh National University, Kazakhstan, Almaty
*e-mail: nuryslam.qassymbek@gmail.com

PARALLEL IMPLEMENTATION OF MONTE CARLO METHOD USING MPI ON THE EXAMPLE OF OIL DISPLACEMENT PROBLEM

Abstract. This article discusses the use of MPI technologies to speed up the calculation time of the oil displacement problem using the Monte Carlo method. The results of numerical simulation and Monte Carlo simulation are compared. The computation time and acceleration of the parallel algorithm with the help of different numbers of processes are obtained. Due to the lack of communication between the processors, the algorithm is very well parallelized and with an increase in the number of computing points, the speedup is also increases.

Keywords: Monte Carlo, MPI, oil displacement, Buckley-Leverett, random walk.

Introduction

Monte Carlo (MC) methods are widely used in solving complex problems in various scientific and engineering fields, but they can be computationally expensive. In recent years, parallel computing has become an important tool for solving large-scale problems in a reasonable amount of time. However, parallelizing Monte Carlo methods can be challenging due to the inherent random nature of Monte Carlo simulations. Monte Carlo methods can be used to solve the oil displacement problem, which is describing by Buckley-Leverett model, but the computational cost of Monte Carlo simulations can be prohibitive for large-scale problems. Therefore, there is a need for efficient and scalable parallel implementations of Monte Carlo methods for solving such problems. The existing literature provides some examples of parallel implementations of MC methods for oil displacement problems, but there is still a need for further investigation and optimization of these methods. This research aims to address this gap in knowledge by investigating the parallel implementation of MC methods using the Message Passing Interface (MPI) library. The research will focus on developing an efficient and scalable parallel Monte Carlo code using MPI for solving pressure equation in Buckley-Leverett model and evaluating its performance against existing parallel implementations. The results of this research will contribute to the development of more efficient and accurate methods for solving such problems using Monte Carlo simulations.

[1] describes the development of Java bindings for the Open MPI library, which allows Java programmers to use MPI for developing parallel applications. The article presents the design and implementation of the Java bindings and provides examples of their usage for developing parallel applications. The authors also demonstrate the performance of the Java bindings using several benchmarks and compare them to existing C and Fortran implementations. While this article is not directly related to the Monte Carlo method, it is relevant to the proposed research because it provides insights into the development and optimization of MPI-based parallel applications. The Java bindings developed in this article could potentially be used in the development of the proposed Monte Carlo code using MPI. And [2] presents a parallel implementation of a finite-volume method for solving convective heat transfer problems. The authors demonstrate the efficiency of their parallel implementation on large-scale problems. [3] describes the development of PenRed, an extensible and parallel Monte Carlo framework for radiation transport based on the PENELOPE code. The authors present the design and implementation of PenRed and demonstrate its performance and scalability using several benchmarks. The article also compares the accuracy of PenRed to other Monte Carlo codes for radiation transport and discusses the potential applications of PenRed in various scientific and engineering fields. While this article focuses on radiation transport simulations, it is relevant to the proposed research because it presents a parallel Monte Carlo framework that can be extended and optimized for solving heat transfer problems. The design and implementation of PenRed provide insights into the development of efficient and accurate parallel Monte Carlo codes,

and the performance benchmarks can inform the optimization of communication patterns in the proposed Monte Carlo code using MPI. [4] presents an uncertainty analysis of a condensation heat transfer benchmark using the CFD code GASFLOW-MPI. The authors demonstrate the parallel efficiency of their code on a high-performance computing cluster. The [5] presents a parallel implementation of a bio- heat transfer problem using an input file affinity measure with MPI. The authors demonstrate the performance of their parallel implementation on a dual-core computer. The proposed MPI-PMMC algorithm is presented in [6] as a novel approach to accelerate Monte Carlo simulations of wind-driven rough sea surfaces. The results of this study showed that the MPI-PMMC algorithm can achieve a significant speedup compared to the serial PMMC algorithm, with a relatively small relative difference between serial and parallel processing. This study contributes to the development of parallel Monte Carlo simulation methods for ocean and atmosphere research. Future work may explore the application of MPI-PMMC to more complex and realistic models of sea surfaces. The paper entitled [7] investigates the magnetic behavior of iron oxide nanoparticles in the transition region between superparamagnetic and ferromagnetic behavior. The paper utilizes the Kinetic Monte Carlo method, coupled with the Stoner-Wohlfarth theory of single-domain ferromagnets, to numerically solve for the magnetic relaxation in this regime and simulate MPI behavior. [8] provides valuable insights into the parallel performance of Monte Carlo photon transport code on different architectures. The study's results can be useful for researchers and practitioners in the field of Monte Carlo particle transport and parallel computing. The paper's methodology and findings demonstrate the importance of selecting appropriate parallel programming paradigms and architectures to optimize performance in Monte Carlo particle transport simulations. In [9], a farmer-worker parallelization of a Monte Carlo simulation for radiotherapy dose calculations was investigated by the authors. Homogeneous clusters running on the Amazon EC2 cloud infrastructure were used, and the authors chose the BEAMnrc system for Monte Carlo radiotherapy simulations as their case study. They highlighted the performance bottlenecks that were faced when using a network file system as the inter-node communication model for a farmer-worker parallelization. To remove these bottlenecks, an alternative approach where message-passing protocols were used was discussed by the authors. The authors demonstrated with empirical results how the performance of the farmer-worker parallelization improved when the communications mechanism was switched from a network file system approach to a message-passing alternative. In [10], a method was presented for parallelizing a Monte-Carlo ion implantation simulator with minimal communication overhead, resulting in nearly linear speed-up. For processor loads that vary strongly, dynamic load balancing should be implemented. In addition to significantly reducing simulation time, the method also splits the memory requirement, enabling the use of multiple small workstations for simulating large three-dimensional problems. The [11] demonstrated a successful application of MPI-based parallelization for Monte Carlo simulations. The method achieves high-performance gains while ensuring low communication overhead and distribution of required memory. The authors' optimization efforts and feedback mechanism also highlight the importance of considering system and algorithmic constraints when parallelizing simulations with arbitrary geometries and physical models. The presented method has the potential for broader applications beyond ion implantation simulations, such as other Monte Carlo simulations in materials science and physics.

While the existing literature provides some examples of parallel implementations of Monte Carlo methods for heat transfer problems, there are some limitations and areas for improvement in these approaches. Some of the limitations include:

1. Lack of scalability: Some existing approaches may not scale well when dealing with very large-scale heat transfer problems. This is because parallelizing Monte Carlo simulations can be challenging due to the random nature of Monte Carlo methods.

2. Inefficient communication: Some existing approaches may use inefficient communication patterns, which can result in performance bottlenecks and slow down the overall computation.

3. Limited accuracy: Some existing approaches may sacrifice accuracy in order to achieve better performance. This can lead to inaccurate results and may limit the applicability of the method.

In order to overcome these limitations, the proposed research aims to develop a more efficient and scalable parallel MC code using MPI for solving heat transfer problems. The research will focus on optimizing communication patterns and developing a more accurate and generalizable method that can be applied to a wide range of heat transfer problems. Additionally, the proposed research will evaluate the performance of the developed method against existing parallel implementations to demonstrate its superiority.

Problem Statement and Monte Carlo Method

The problem of oil displacement by water is considered, which is described by the Buckley-Leverett model. It is required to calculate the pressure distribution in the area under consideration.

Equations for the continuity of the water and oil phases:

$$m\frac{\partial s_1}{\partial t} + \frac{\partial}{\partial x}\left(-k\frac{f_1}{\mu_1}\frac{\partial P}{\partial x}\right) = 0, \quad (1)$$

$$m\frac{\partial s_2}{\partial t} + \frac{\partial}{\partial x}\left(-k\frac{f_2}{\mu_2}\frac{\partial P}{\partial x}\right) = 0, \quad (2)$$

where, P is pressure, m is porosity, s_1, s_2 are phase saturations, k is absolute permeability, f_1, f_2 are relative phase permeabilities of phases, μ_1, μ_2 are phase viscosities.

Initial and boundary conditions:

$$s_1|_{t=0} = s_{1o}$$
$$s_1|_{x=0} = s_{inj} \quad (3)$$
$$s_1|_{x=l} = 0$$

By adding the two equations and after a few arithmetic operations, we get the following equation for pressure:

$$P_i^{n+1} = (M)_{i+\frac{1}{2}}\frac{\Delta t}{h^2}P_{i+1}^n + \left(-1 - (M)_{i+\frac{1}{2}}\frac{\Delta t}{h^2} + (M)_{i-\frac{1}{2}}\frac{\Delta t}{h^2}\right)P_i^n + \left(-(M)_{i-\frac{1}{2}}\frac{\Delta t}{h^2}\right)P_{i-1}^n \quad (4)$$

Further, we take the coefficients before P^n as probabilistic values:

$P_{x+} + P_t + P_{x-} = 1$
$(x,t) \rightarrow (x + \Delta x, t)$ *if* $(0 < r < 0.25)$
$(x,t) \rightarrow (x - \Delta x, t)$ *if* $(0.25 < r < 0.5)$
$(x,t) \rightarrow (x, t - \Delta t)$ *if* $(0.5 < r < 1)$

and start the walking process. That is, for each point i in the time layer $n + 1$, we generate k times random values r within 0...1 and, based on the obtained values, we move according to our numerical scheme to the left border or to the right border or back in time. When the boundary or initial values are reached, we summarize these values and divide by the number of walk steps k. Thus, we find the value of P_i^{n+1}. Figure 1 shows a diagram of this process. For the point P_i^{n+1} in this example, the process of random walks was launched 3 times. As you can see in the figure, the walks in blue and orange have reached the left boundary, while the walk marked in green has reached the lower bound, that is, the initial values.

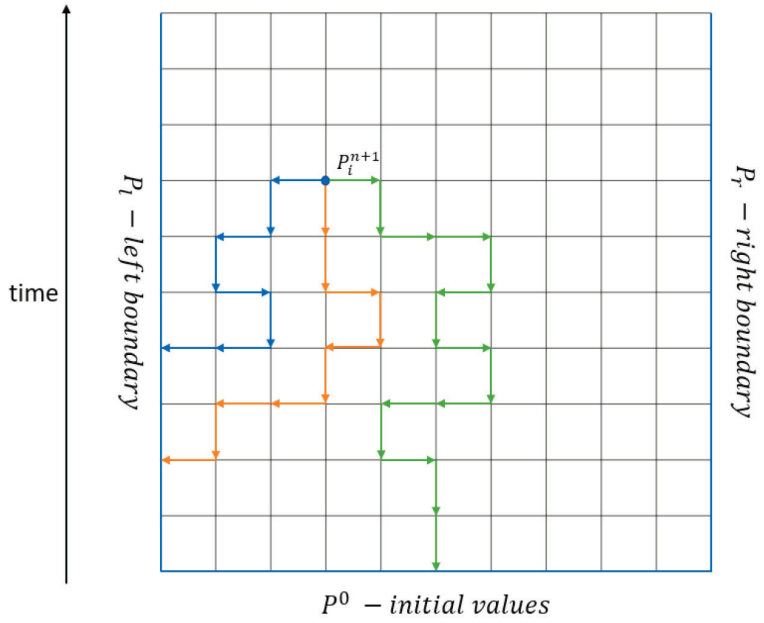

Figure 1 – Scheme of random walks

An analysis was made of the influence on the accuracy of the solution of the number of random walks, which can be seen in the following figures.

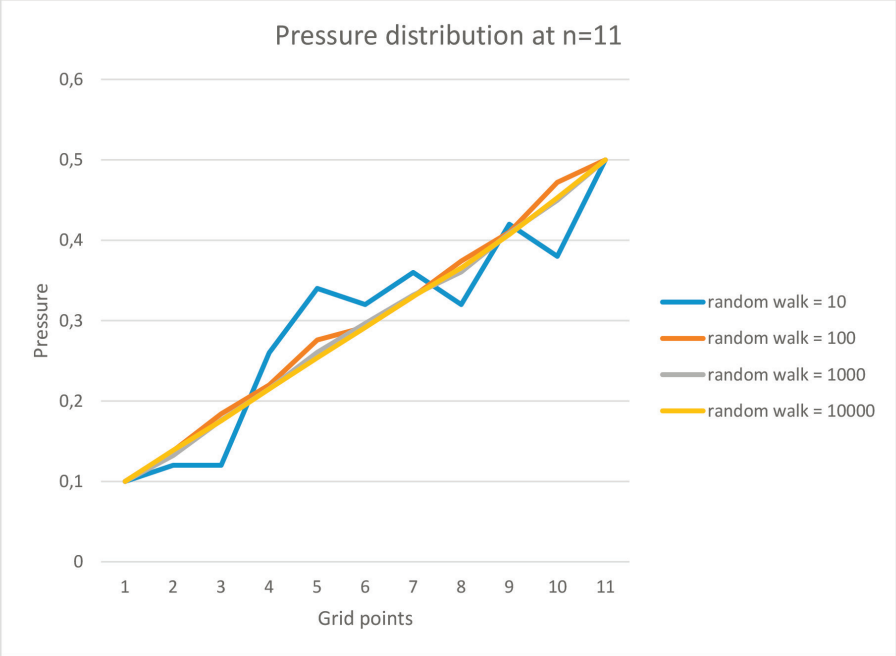

Figure 2 – Pressure distribution at *n= 11*

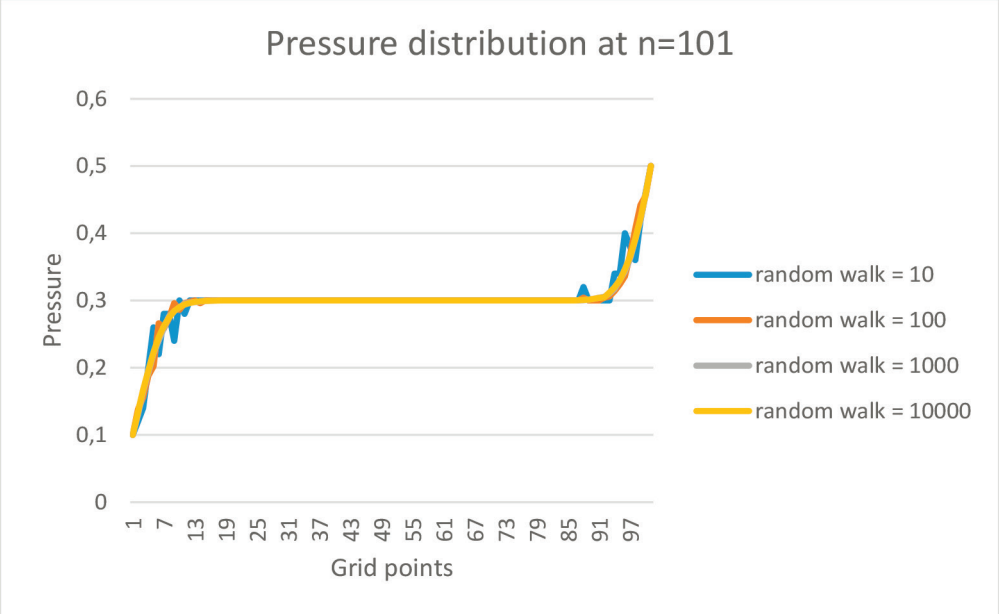

Figure 3 – Pressure distribution at *n= 101*

As can be seen in the figures, with an increase in the number of random walks, the graph of pressure changes becomes smoother and more accurate.

Parallel algorithm

A parallel algorithm for the random walk method was developed and implemented using the MPI standard. MPI is the most common data exchange interface standard in parallel programming, and there are implementations for a large number of computer platforms. It is used in the development of programs for clusters and supercomputers. The main means of communication between processes in MPI is the passing of messages to each other.

Parallelization was carried out by dividing the points of the difference scheme by processes. That is, p processes ran k walks in parallel for their points in the difference scheme in the amount of n/p.

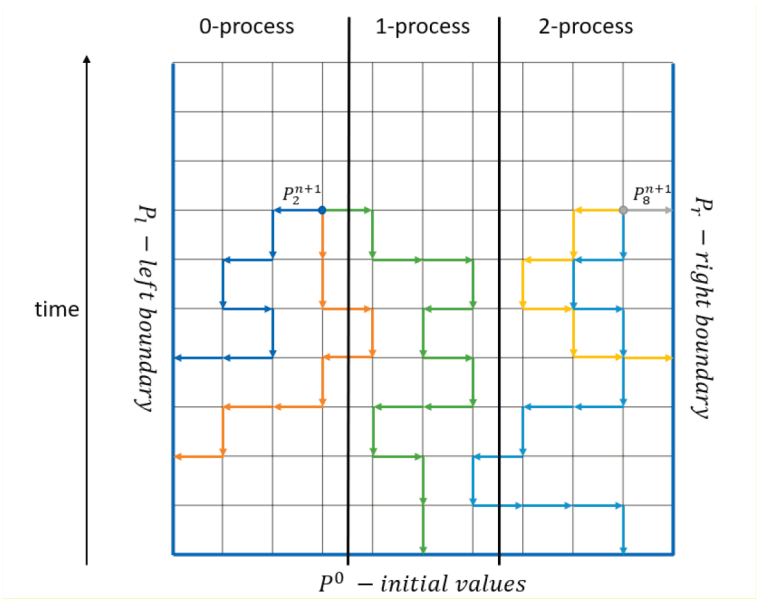

Figure 4 – Scheme of parallel execution of walks for a scheme of 9 points on three parallel processes

In Figure 4, you can see the parallel execution of random walks on three parallel processes. A 9-point circuit is shown, i.e., each parallel process runs k walks for its 3 points. This example clearly shows how the zero MPI process starts the walks marked in blue, orange and green for the point P_2^{n+1}, and the second MPI process starts the walks marked in yellow, blue and gray for the point P_8^{n+1}. At the same time, there are no clear boundaries between parallel processes in walking. For example, the green walk that starts the zero MPI process visually passes on the points of the first MPI process. But this does not mean the need to transfer data. And the division between processes means only for which points each process performs walks. For example, the zero MPI process calculates values at points with indices 0,1,2, the first MPI process at points with indices 3,4,5, and so on.

At the same time, it should be noted that the parallel algorithm gives a solution that is not identical to the sequential algorithm, but very similar in general dynamics. This is due to the peculiarity of generating random numbers for walks. So, when running on several parallel processes, MPI generates different numbers than when running on a single process.

The computation time and speedup obtained with parallel executions can be seen in the following figures.

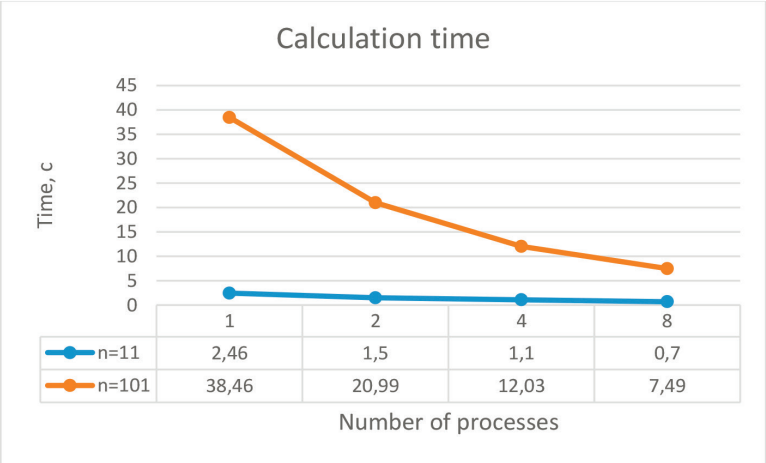

Figure 5 – Computation time for different number of processes

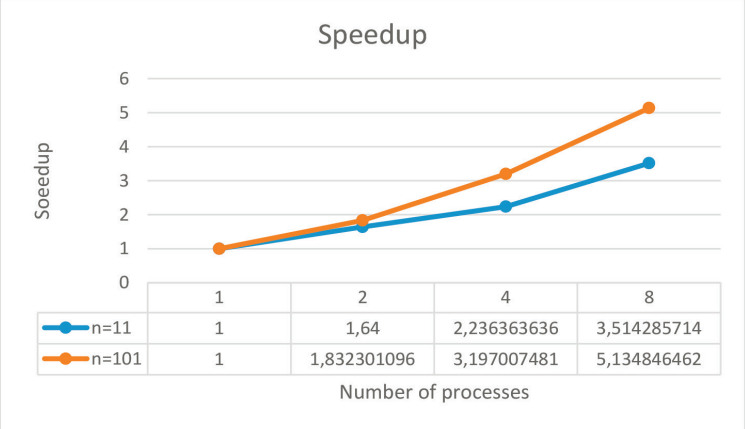

Figure 6 – Acceleration on a different number of processes

As can be seen from the figures 5-6, a speed-up was obtained through parallel execution. At the same time, with an increase in the number of points in the difference scheme, the acceleration also increases. The algorithm parallelizes very well, since there is no communication between processes during calculations. But with parallel and sequential execution, different values are obtained, this is due to the fact that different random values are generated at different launches. But the dynamics of calculations remains.

Conclusion

Based on the results of the study, it can be concluded that the developed parallel algorithm for the random walk method using the MPI standard successfully accelerates the calculation process for large-scale problems. The parallelization of the algorithm was carried out by dividing the points of the difference scheme by processes, and the main means of communication between processes in MPI is the passing of messages to each other. The parallel algorithm provides a solution that is not identical to the sequential algorithm but is very similar in general dynamics. Moreover, the algorithm parallelizes very well, since there is no communication between processes during calculations. With an increase in the number of points in the difference scheme, the acceleration also increases, as shown in Figures 5-6. Therefore, the developed parallel algorithm can be used for solving large-scale problems in various areas of science and technology.

Acknowledgments

The work was carried out within the framework of the project AP09260564 – "High-performance computer simulation of the motion of a multi-phase fluid in a porous medium under uncertainty".

References

1. O. Vega-Gisbert, J.E. Roman, J.M. Squyres. Design and implementation of Java bindings in Open MPI, *Parallel Computing*, (2016), 1-20, 59

2. D.Goik, K. Banas´, J. Bielanski, K. Chłon. Efficient simulations of large-scale convective heat transfer problems, *Computer Science*, (2021), 517-538, 22(4).

3. V. Gimenez-Alventosa, V. Gimenez Gomez, S. Oliver. PenRed: An extensible and parallel Monte-Carlo framework for radiation transport based on PENELOPE, *Computer Physics Communications*, (2021), 267.

4. H. Zhang, Ya. Li, J. Xiao, T. Jordan. Uncertainty analysis of condensation heat transfer benchmark using CFD code GASFLOW-MPI, *Nuclear Engineering and Design*, (2018), 308-317, 340.

5. S.U. Ewedafe, R.H. Shariffudin. On the parallelization of bio-heat transfer problem using input file affinity measure with MPI, *WIT Transactions on Modelling and Simulation*, (2013).

6. Yu Yang, Lixin Guo. A Parallel Monte Carlo Simulation Algorithm for the Irradiance Reflectance Properties of a Rough Sea Surface Based on MPI, *Cross Strait Quad-Regional Radio Science and Wireless Technol- ogy Conference (CSRWTC)*, (2021), 177-179.

7. L.H. Hua, O. Ferguson, R. M. Chantrell, R.W. Krishnan, M. Kannan. MPI tracer magnetization simulated using a Kinetic Monte Carlo method, *International Workshop on Magnetic Particle Imaging, IWMPI*, (2013).

8. A. Majumdar. Parallel performance study of Monte Carlo photon transport code on shared-, distributed-, and distributed-shared-memory architectures, *Proceedings of the International Parallel Processing Symposium, IPPS*, (2000), 93-99.

9. Ya. Gagarine, D.W. Walker, C. Walker. Improving parallelisation of a Monte Carlo radiotherapy simulation using MPI, *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2012*, (2012), 1033-1039.

10. A. Hossinger, M. Radi, B. Scholz, T. Fahringer, E. Langer, and S. Selberherr. Parallelization of a Monte-Carlo ion implantation simulator for three-dimensional crystalline structures, *International Conference on Simulation of Semiconductor Processes and Devices, SISPAD*, (1999, 103- 106.

11. A. Hossinger, E. Langer. Parallelization of a Monte Carlo ion implantation simulator, IEEE *Transactions on Computer-Aided De- sign of Integrated Circuits and Systems*, (2000), 560-567, 19(5).