Scalability investigation of parallel algorithms for plasma dynamics problems in open magnetic traps by simulation modeling

D Weins\(^1\), M Boronina\(^1\), V Vshivkov\(^1\), E Mesyats\(^1\), G Dudnikova\(^1\) and I Chernykh\(^1\)

\(^1\) The Institute of Computational Mathematics and Mathematical Geophysics SB RAS (ICM&MG SB RAS) 6, Acad. Lavrentieva Pr., Novosibirsk, 630090, Russian Federation

vins@sscc.ru, boronina@ssd.sscc.ru, vsh@ssd.sscc.ru, mesyats@gmail.com, gdudnikova@gmail.com, chernykh@parbz.sscc.ru

Abstract. This paper describes the use of predictive simulation modeling for investigating the scalability of parallel algorithms. The study is carried out using a multiagent approach. The general principles of designing a model for the execution of a parallel algorithm on a given supercomputer architecture are described along with their application for researching this algorithm. The paper also presents the results of simulating the execution of this algorithm on more than a million cores.

1. Introduction

Modern and promising supercomputers are characterized by high parallelism, hybrid structure, and hierarchical communication medium, so software development for such systems is a complex scientific task. However, even when a program for performing this complex task on a supercomputer is made, a researcher encounters the problem of a boundary of effective scalability of this program, i.e., what is the maximum number of computational nodes required for the best acceleration of this program on a given supercomputer architecture and with given input data? It is possible to solve this problem by multiply running this program on a different number of computational nodes, but the cost of computer time is very high for modern supercomputers, and not all supercomputer centers are ready to provide the required amount of resources.

Within the framework of an integral approach to developing parallel programs [1], created at the Institute of Computational Mathematics and Mathematical Geophysics SB RAS (ICM MG SB RAS), this problem is solved using multiagent modeling methods for simulating multiple runs of the program, which requires a much smaller amount of real resources and time and allows studying the scalability of the algorithm under study on existing and promising supercomputers. This modeling was previously carried out using the AGNES multiagent modeling system [2], but, due to the specific features of this system, it was necessary to apply the Actor model for minimizing the communication overhead of agent communication [3]. Experimental studies particularly show that simplicity of implementation, scalability, and fault tolerance are the features that make the Erlang programming language the most fit for studying the scalability of algorithms by simulation modeling methods [4].
This paper illustrates the application of this approach to studying the scalability of numerical modeling of plasma dynamics in axially symmetric open magnetic traps [5].

2. General scheme for the parallel algorithm research

In general form, the computational process is a set of threads executed in parallel on an individual computational node, which interact between each other by exchanging values (messages). The main characteristics of threads are the computation execution time and the time of data transmission to another computational node. The simulation modeling of the execution of a computational process on a supercomputer is carried out by constructing a model of interaction of the computational process threads. In this model, each thread is assigned to a computational unit and communication unit, with a scheme of message interaction between the threads being constructed (Figure 1).

![Figure 1. Example of the model of interaction of the computational process thread.](image)

The main efficiency parameter of the computational process under study during its execution on high-performance supercomputers is its scalability. It should be noted that different scalability indicators are important for different computational algorithms. For example, for statistical modeling methods [6], it is weak scalability, i.e., the execution time decreases as the number of computational nodes increases. For network methods of numerical modeling [7], it is strong scalability, i.e., the execution time of the algorithm remains unchanged as the computational domain increases proportionally to the number of computational nodes.

This paper does not touch upon the scalability of the computational processes within the framework of a single central processor or accelerator as the number of cores of computational elements is currently determined and much smaller than the potential number of computational nodes of high-performance supercomputers. Thus, all computations and exchanges between the computational elements of a single computational node are taken as a fixed value that does not affect the general computational scheme.

The modeling of the computational process requires studying the thread interaction model and the threads having similar behavior (performing identical computations and exchanges) are determined. Each group of homogeneous threads is assigned to a class of actors simulating the execution of computational units and message exchange with other actors (threads). The execution of the entire computational process is simulated by forming a necessary number of samples of each class of actors. It should be noted that Erlang allows for message exchange between any actors in the system, thereby making it possible to arrange any kind of interaction between computational threads.
For each actor, an algorithm for simulating the behavior of a computational thread is programmed and the main execution parameters of a thread are transferred, such as the time it takes to determine each computational unit, the number of transferred values, and the time it takes to transfer each value for all communication units. As soon as all actors are formed and registered in the system, the multiagent model of execution of the computational process is being initialized. Each actor seeks for its “neighbors”, i.e., actors with whom it communicates while simulating the thread execution. In accordance with the thread execution algorithm and its own parameters, the actor configures inner gauges for simulating calculations and message exchange-reception.

Upon reaching the criterion for stopping the algorithm execution, the actors finish all the operations from an incomplete cycle and return the results of their work. The main parameters studied after the simulation are the total operating time of the computing process, the execution time of each actor, the number of iterations of the calculation cycle performed by each actor, and the data latency between the actors.

The efficiency of a computational process during its execution on a high-performance supercomputer is based on the concept of its scalability directly dependent on the total execution time, so it is the key parameter. It should be noted that the main parameter affecting the total execution time of the computational process are data latency in communication networks (or in logical lines between actors when dealing with simulation modeling) because the execution time of computational units and the time it takes to transfer an element are fixed values for each computational node. Therefore, this parameter is studied with a special attention in the modeling of execution of computational processes on a large number of computational nodes.

Moreover, only the extensiveness of the computational system is investigated within this scheme. Naturally, with propagation of petascale and following exascale calculations, the number of computers and data transfer rate become higher. At the same time, computational complexity becomes larger too, which complicates algorithms and increases the number of operations per one thread. Therefore, within the framework of this model, the scalability of parallel processes is studied with account for current equipment and, as a consequence, complexity of the computational model.

The proposed method for researching the computational processes has already been successfully used in the study of scalability of several computational algorithms per a large number of computational cores [6,7,8,9].

3. Scalability investigation of parallel algorithms for plasma dynamics problems in open magnetic traps

3.1. Description of the calculation scheme for the algorithm under study

The parallel implementation of the hybrid numerical model of plasma dynamics in open magnetic traps is based on the mixed decomposition of the computational domain, the latter of which is uniformly divided into subdomains, with each subdomain being controlled by a group processor. Particles in subdomains are distributed uniformly among group processors (Figure 2). Boundary nodes for the subdomain of grid nodes are transferred among the main group processors. Particles can be sent to any processor from a neighboring group.

Each step of the computation cycle includes the Lagrangian stage for calculating the particle velocities and coordinates and the Euler stage for computing on a spatial grid.

At the beginning of each step of the computational cycle, the main process sends out an electric and magnetic field in the subdomain through the group using MPI_BCAST.

Moving to the Lagrangian stage requires using bilinear interpolation, calculating the Lorentz force acting on the particle at its location in each process, and determining the particle velocities and coordinates. If a particle flies beyond the subdomain boundaries, then its data is sent to one of the processors of the neighboring group. Using a local rank allows evenly distributing particles in the processes during transfer.
After that, each process calculates the average ion velocities in a cell and densities, after which the Euler stage begins. The main processors of the group collect the group data using MPI_REDUCE and then exchange the boundary data for the grid.

Next, they are solved for electron velocities, for an electric field, for a magnetic field, and for temperature. After solving the equation for each of the listed values, data is exchanged for the boundary grid nodes between neighboring main group processors.

It should be noted that an algorithm with the same decomposition of the computational domain and a similar process interaction scheme has already been studied using the method described [9]. However, the important distinguishing feature is that the numerical modeling algorithm for exactly calculating the characteristics of instabilities excited by an electron beam in plasma at the end of a computational cycle, the processes have an all-to-all data exchange over the entire computational domain using MPI_ALLREDUCE. In the algorithm under study, the main process of the group initially sends data to all processes in the group, then collects data from all the processes following necessary calculations, and it is the generalized data that is exchanged with neighboring main processes of the group afterward.

3.2. Scalability of the parallel algorithm

The execution of the numerical modeling algorithm of plasma dynamics in open magnetic traps is simulated by implementing PlasmaMain and PlasmaSub classes of actors. The actors of the PlasmaMain class simulate the execution of the main processes of the group, which is data transfer and collection from all the processors of the group, as well as data exchange with neighboring actors of the PlasmaMain actors. The actors of the PlasmaSub class simulate the execution of the remaining processes of the group: calculation and exchange cycles. Delays during message exchange between the actors eventually simulate the corresponding delays in message exchange in the real computational system. The algorithm execution model performs a previously specified number of calculation cycles. The time characteristics of execution of all stages of the computational cycles are taken from the results of running a real computational algorithm on the NKS-1P cluster of the of the General-Purpose Computing Center “Siberian SuperComputer Center” of the Institute of Computational Mathematics and Mathematical Geophysics of the Siberian Branch of the Russian Academy of Sciences. The calculations are carried out on Broadwell computational nodes with Intel Xeon E5-2697A v4 processors.

The scalability of the algorithm under study is demonstrated using calculations for a various number of computational nodes (more than $10^6$ computational cores). The results of this study are
illustrated in figure 3. The differences of the interaction scheme of the processes in the algorithm investigated previously (using the MPI_ALLREDUCE operator [9]) and the algorithm under study (with the MPI_BCAST and MPI_REDUCE operators) are demonstrated in the chart by the scalability of both algorithms.

![Figure 3. Strong scalability of the studied (MPI_BCAST, MPI_REDUCE) and previously mentioned (MPI_ALLREDUCE) algorithms. The horizontal axis is in logarithmic scale.](image)

It can be concluded from figure 3 that using the calculations of the main process of the group, occupied by data transfer, aggregation, and exchange with neighboring domains can significantly reduce the communication overhead as compared to the algorithm with all-to-all exchange. It should also be noted that this algorithm generally shows high scalability in the case of execution on a large number of computational cores.

4. Conclusion
The method for studying the scalability of parallel algorithms on a large number of computational cores using simulation multiagent modeling is formed within the framework of the integral approach to developing parallel programs. This approach allows predicting the efficiency of using this algorithm on future HTC systems without the help of many computational and material resources.

This method was used to investigate the scalability of the numerical modeling algorithm of the plasma dynamics in axially symmetric open magnetic traps. The study showed that the algorithm could be scaled on more than a million computational cores without any significant efficiency losses. It could also be concluded that the specific features of parallel implementation of this algorithm essentially increased its efficiency due to absence of universal data exchange operations.

Acknowledgements
The work was financially supported by the Russian Science Foundation (Grant No. 19-71-20026). The computations were performed by the Siberian Supercomputer Center cluster (ICM&MG SB RAS, Novosibirsk).

References
[1] Glinskiy B M, Kulikov I M, Chernykh I G, Snytnikov A V, Sapetina A F and Weins D V 2017
The Integrated Approach to Solving Large-Size Physical Problems on Supercomputers CCIS 793 pp 278-289

[2] Podkorytov D, Rodionov A and Choo H 2012 Agent-based simulation system AGNES for networks modeling: review and researching Proc. Int. Conf. on Ubiquitous Information Management and Communication (Kuala Lumpur, Malaysia) P115

[3] Weins D V, Glinskiy B M, Chernykh I G 2019 Analysis of Means of Simulation Modeling of Parallel Algorithms CCIS 965 pp 1–11

[4] Cesarini F, Thompson S 2009 Erlang Programming. (Boston: O’Reilly Media, Inc) p. 498

[5] Berendeev E, Dudnikova G, Efimova A and Vshivkov V 2017 Computer simulation of plasma dynamics in open plasma trap LNCS 10187 pp 227-234.

[6] Glinskiy B, Rodionov A, Marchenko M, Podkorytov D and Weins D 2012 Scaling the Distributed Stochastic Simulation to Exaflop Supercomputers Proc. Int. Conf. on High Performance Computing and Communications (Liverpool, England, UK) pp 1131–1136.

[7] Glinskiy B, Kulikov I, Chernykh I, Weins D, Snytnikov A, Nenashev V, Andreev A, Egunov V and Kharkov E 2016 The Co-design of Astrophysical Code for Massively Parallel Supercomputers LNCS 10049 pp 342-353.

[8] Glinskiy B, Sapetina A, Martynov V, Weins D and Chernykh I 2017 The Hybrid-Cluster Multilevel Approach to Solving the Elastic Wave Propagation Problem CCIS 753 pp 261-274.

[9] Glinskiy B M, Kulikov I M, Snytnikov A, Romanenko A A, Chernykh I G and Vshivkov V A 2017 Co-design of parallel numerical methods for plasma physics and Astrophysics Supercomputing Frontiers and Innovations 1 N3 pp 88–98.