Application of parallel-conveyor computing systems in the problem of modelling of synthesis CNS

G V Abramov¹, A L Ivashin and I S Tolstova²

¹ Voronezh State University, 1, Universitetskaya pl., Voronezh, 394018, Russia
² Voronezh State University of Engineering Technology, 19, Revolution Av., Voronezh, 394036, Russia

E-mail: Irin2102ka@mail.ru

Abstract. The paper discusses the development of an effective algorithm for modeling the process of motion and interaction of particles in a plasma during the synthesis of carbon nanostructures using the principle of parallel-pipeline data processing. The classification of step algorithms is considered from the point of view of the possibility of changing the algorithm so that the data arrive at the input and/or output sequentially. Given this classification, each of the steps are considered in more detail from the point of view of algorithms for their execution and optimization in order to add the possibility of sequential data input and output. A comparison of the results of a sequential algorithm for solving the problem in steps and a parallel-conveyor algorithm is given.

1. Introduction

This time period is characterized by the development of a wide range of scientific and practical computational problems, which leads to an increase in the amount of data that must be promptly processed to minimize the time delays in calculations. Many processing algorithms use the same type of massive computation, for which large computational power is needed. Such algorithms can be executed in several independent streams using parallel or distributed systems, which will reduce the overall processing time. The process of dividing data into several computing cores is usually called a computing pipeline, which consists of the same type of execution stages. In this way, you can significantly speed up the data processing process, ensuring the appearance of results with the frequency of data receipt. Multi-core and multi-processor personal computers, or cluster systems today are used as computing systems for parallel computing.

Research in the field of synthesis and application of CNSs is largely hampered by the fact that the mechanisms of interatomic interaction and the association of atoms of such structures are still unknown, which makes it impossible to work out a strategy for obtaining new functional characteristics of the material [1]. For theoretical substantiation and prediction of the behavior and properties of the nanoscale structures, it is necessary to create models of the processes of their formation. The process modeling belongs to the class of complex tasks due to the high labor intensity, large amounts of necessary memory.

In this regard, there is a need to use special numerical methods, parallel technologies and programs focused on high-performance computing equipment. Previously, supercomputers and computational clusters were used to solve such problems; however, the cost of such computer equipment is quite...
high, and modern systems must be not only high-performance, but also economically and energy-efficient.

The aim of the work is to develop an efficient algorithm for the method of large particles for modeling the processes of motion and interaction of particles in the plasma during the synthesis of carbon nanostructures (CNS), focused on the use of the principle of parallel-conveyor data processing.

2. Materials and method
The three-dimensional problem of modeling the CNS synthesis process is considered. The basis of the kinetic model of the synthesis of the CNS by the electric arc method, which describes the dynamics of the motion of charged particles in a plasma with Coulomb interaction, is the system of Boltzmann equations taking into account the collision integral written for each type of multicomponent plasma particles under consideration, supplemented by the Maxwell equations system [2-4].

In the course of solving the problem, a large amount of information is analyzed; therefore, it is advisable to use functional modeling, which greatly facilitates the analytical work. In figure 1 presents a contextual diagram of the process of modeling the synthesis of CNS in IDEF0 notation. The figure shows: input information that is transformed for further research (initial technological parameters of the synthesis); control, regulatory data (calculation algorithm, simulation time); process maintenance mechanisms or resources (software package); the results of the study, namely the formation of cluster groups, numerical and graphic data of the process.

In figure 2 shows the top level diagram of A0, which allows to detail the steps.

![Figure 1. Context diagram of the CNS synthesis simulation process.](image-url)
To reduce the amount of calculations, by reducing the number of particles of the same type in the calculation by grouping them to a reasonable level, which does not affect the accuracy of the calculation, the large particles method is used [5]. In figure 3 represented by large particles method in the form of a data flow diagram in Gain-Sarson notation.

Figure 2. Decomposing diagram.
The process of modeling the motion of particles continues until the simulation time is over. At each stage, about $10^{10}$ large particles are involved. In the course of the work, an analysis was made of the distribution of processor time spent on solving the considered problems of the model. The most resource-intensive elements of the model are determined: the generation of the initial distribution of large particles, the calculation of the parameters of the electromagnetic field by the method of establishment and the search for the interaction of particles [6-7].

Figure 4 shows the proportion of the total time spent on solving the above basic tasks of the model.

**Figure 3.** Data Flow Chart of the large particles method.

**Figure 4.** Time spent for the tasks solution.
As can be seen from the diagram in figure 4, the main time of the program for numerical calculation of the model is tasks 1, 3 and 6. If the first task can be solved once for each set of initial parameters of the simulation by pre-generating the necessary data into a separate file, then only use the obtained distribution, then problems 3 and 6 have a larger number of variable input arguments, which makes the approach of the preliminary calculation untenable. The solution of tasks 3 and 6 requires the greatest expenditure of computational resources (CPU time, RAM) in the process of plasma kinetics simulation. One of the ways to reduce the total computation time of the considered laborious computational problem is the organization of pipelining and parallelism.

The essence of pipelining is that the running operations are divided into small parts, which are called steps. Separate hardware units serve different stages. Pipeline processing is formed by separating operations into separate layers, in turn, each of these levels accepting input data and simultaneously redirects the result to the next one. To efficiently use pipelining, it is necessary to analyze the tasks presented in Diagram 3.

We introduce a classification of algorithms for the steps of the problem being solved. We will consider the classification from the point of view of the possibility of changing the algorithm, so that the data arrive at the input and / or output of the algorithm sequentially. In other words, for each of the steps, it is necessary to answer the question: “Is there a fundamental possibility to use some data structure, for example, a queue, at the input or output of the algorithm?” In view of the above, four mutually intersecting classes can be distinguished:

- class 1 - algorithms that support sequential data entry;
- class 2 - algorithms that do not support sequential data entry, i.e. require a complete set of source data to run;
- class 3 - algorithms that support serial data output;
- class 4 - algorithms that do not support serial data output, i.e. can be obtained a complete set of calculated data after the completion of the algorithm.

Each of the considered steps within the framework of the algorithm can be marked with a tuple of two classes describing the algorithm in terms of input and output data:

- Step 1 - (1, 3);
- Step 2 - (1, 4);
- Step 3 - (2, 4);
- Step 4 - (2, 3);
- Step 5 - (2, 3);
- Step 6 - (1, 4);
- Step 7 - (1, 3);

Step 7 has been added to the general list of steps, which is highlighted for the convenience of reviewing the iterative calculation process; in fact, it prepares the initial data for the next iteration.

The next step is to identify the dependence of the algorithms of data problems, we present it in the form of a directed graph (figure 5), in which the directed edge shows that the result of solving the problem, from which the edge goes, is required to solve the problem, which includes the edge.

![Figure 5. Dependence graph of data steps.](image)
Next, we consider the graph (figure 6) of the implemented general simulation algorithm for one iteration. This graph illustrates the sequential execution of all calculations. On it the vertices indicate the beginning of the execution of the next calculation step, and the edges indicate the execution process and the next step [8]. The described notation is used in figure 7.

![Figure 6. Steps solving graph.](image)

Considering the classification of steps and their dependence on data, we can modify the graph as shown in figure 7. It shows that based on the dependencies on the source data for each step, you can sequentially execute each step can be changed to partially parallel, taking into account the input and output algorithms of the algorithms. The graph in figure 7 should be interpreted as follows: steps 1, 2, 6 are not interdependent on the source data, therefore they can be run in parallel, after the completion of algorithms 1 and 2, steps 3, 4 and 5 can be sequentially started. two cases that will come later: the completion of the algorithm of step 6 or the completion of the algorithm of step 5.

![Figure 7. Reshaped problem solving sequence graph.](image)

Evaluation of the graphs is possible in terms of the length of the path from the initial problem to the final one. In this case, the total duration of the calculations is equal to the execution time of the longest journey of all. It is worth noting that one of the longest parts of the calculations - stage 6 can be performed in parallel almost all the time of the iteration calculation.

Let us consider in more detail each of the steps from the point of view of algorithms for their implementation and optimization of algorithms in order to add the possibility of sequential data input and output.

The formation of large particles is an algorithmically quite simple task, with the computational complexity O (n). As part of its solution, several optimizations are possible at once. The most expedient of them seems to be the preparation of a sufficiently large array of random data, with a dimension much higher than n, and further sampling from it. This approach can significantly reduce processor costs by removing the algorithm for generating a random variable with a given distribution. At the same time, there is no difficulty in creating a queue in which each newly created particle will be placed, but step 3 eliminates these efforts, since it requires a complete set of source data.

The calculation of the charge density in the grid nodes is based on the particles distributed in the simulation domain. This algorithm can significantly run in parallel with the previous step, providing the result of its work as input to the algorithm of step 3.
The algorithm for calculating the parameters of the electromagnetic field cannot be successfully modified so that it can be assigned to class 1 or 3. The reason for this is the establishment method, which approaching the final solution changes all the result values with each iteration.

The next two stages of calculations - the calculation of particle parameters on the particle and their displacement can be optimized in order to bring them to class 3. However, the calculation of the displacement requires the entire set of particles at the inlet, therefore such optimization will not make sense.

The search for the interaction of particles can be organized to consistently obtain a list of interacting particles with one proviso: it is assumed that a particle can encounter only once in the course of a single iteration. This makes it possible to perform it for a long time without braking other calculation steps. This algorithm is executed only after performing two iterations of the other steps, so the filled input queue will have twice the length compared to the others. The preparation step for the next iteration can be provided with queues at both the input and output.

3. Results and discussion
The justification for the use of such optimizations can be identified by simulating calculations as a queuing system. It is necessary to make two assumptions: first, requests are received in any of the queues with a frequency inversely proportional to the total step calculation time (figure 4); the second is placing and retrieving an application from the queue in negligible time. Taking into account the comments made, the model of the system is shown in figure 8, where the queues used are labeled with Q, devices with the corresponding request processing parameters through S. The model of the source system looks like in figure 6.

The implementation of both models for a grid size of $10^6$ nodes and $3\cdot10^5$ particles gives the following results:

- The total run time of 1000 iterations of the model with the proposed optimizations decreased by 1.43 compared with the original;
- Q1 is the queue with the longest average stay time of the application, Q3 has the lowest average time.

![Figure 8](image)

**Figure 8.** Problem sequence graph with queues.

4. Conclusion
The analysis allows draw the following conclusions. The proposed measures to optimize calculations can be used within the existing code base. To do this, the selected steps must be drawn up in the form of asynchronous execution areas that read their turns. Also, based on the parameter of the average time in the queue, we can make the assumption that the optimization of the algorithm for calculating the
parameters of the electromagnetic field is the most productive. This calculation step is not only the longest, but also blocking the execution of other long steps.

Using this approach to optimize the computational process is applicable for the case of the method of large particles, it is determined by the consistent use of the Eulerian and Lagrangian approaches, which makes it possible to parallelize the algorithm. As described above, the applied parallelization technique is used in the transition from the nodes of the computational grid to the interaction of the calculated large particles [9, 10]. This kind of optimization is applicable, for example, to solve problems using the particle method in Harlow cells.

References

[1] Kabaldin Y G 2007 Synergetics. The information models of the sektorassembly nannosystems of the material nanostructuring under the superficial mechanical forces: monograph. (Komsomolsk-na-Amyre : KnASTU)

[2] Abramov G V, Gavrilov A N, Tolstova I S and Ivashin A L 2017 Formation of Clusters of Carbon Structures in Plasma under Thermal Destruction of Graphite. Nanotechnologies in Russia 3–4 139–146

[3] Aristov V V 1988 Direct methods for solving the Boltzmann equation and study of nonequilibrium flows (Dordrecht; Boston; London: Kluwer Academic Publishers)

[4] Gavrilov A N 2018 Modeling the formation of carbon cluster groups in electric arc discharge plasma. Proceedings of the Voronezh State University of Engineering Technologies 80 108-113

[5] Abramov G V and Gavrilov A N 2018 The application of the large particles method of numerical modeling of the process of carbonic nanostructures synthesis in plasma Journal of Physics: Conf. Series 973 1-9

[6] Abramov G V, Gavrilov A N, Ivashin A L and Tolstova I S 2018 The use of parallel computing in resource-intensive tasks of modeling the processes of motion and interaction of particles in a plasma during the synthesis of carbon nanostructures Bulletin of Moscow State Technical University n.a. N.E. Bauman. Natural Sciences Series 5 4-14

[7] Abramov G V, Gavrilov A N, Ivashin A L and Tolstova I S 2017 Modeling of the motion and interaction of carbon particles in the plasma electric arc discharge using parallel programming technologies Proceedings of The 8th International Multi-Conference on Complexity, Informatics and Cybernetics (IMCIC 2017) 67-72

[8] Nicholas Chen, Ralph E Johnson, J Flow 2013 Practical refactorings for flow-based parallelism, Automated Software Engineering IEEE/ACM 28th International Conference on 202-212

[9] Pukhov A and Meyer-ter-Vehn J 1996 Relativistic Magnetic Self-Channeling of Light in Near-Critical Plasma: Three-Dimensional Particle-in-Cell Simulation, Phys. Rev. Lett

[10] Adam J C, Serveniere A G and Langdon A B 1982 Electron sub-cycling in particle simulation of plasmas J. Comput. Phys. 47 229-244