The Integrated Approach to Solving Large-Size Physical Problems on Supercomputers

Boris Glinskiy, Igor Kulikov, Igor Chernykh, Alexey Snytnikov, Anna Sapetina✉, Dmitry Weins

The Institute of Computational Mathematics and Mathematical Geophysics of SB RAS, Novosibirsk, Russia
gbm@opg.sscc.ru, kulikov@ssd.sscc.ru, chernykh@parbz.sscc.ru, snytav@gmail.com, afsapetina@gmail.com, wns.dmitry@gmail.com

Abstract. This paper presents the results obtained by the authors on applying an integrated approach to solving geoseismics, astrophysics, and plasma physics problems on high-performance computers. The concept of the integrated approach in the context of mathematical modeling of physical processes is understood as constructing a physico-mathematical model of a phenomenon, a numerical method, a parallel algorithm and its software implementation with the efficient use of a supercomputer architecture. With this approach, it becomes relevant to compare not only the methods of solving a problem but, also, physical and mathematical statements of a problem aimed at creating the most effective implementation of a chosen computing architecture. The scalability of algorithms is investigated using the multi-agent system AGNES simulating the behavior of computing nodes based on the current state of computer equipment characteristics. In addition, special attention in this paper is given to the energy efficiency of algorithms.

Keywords: Integrated Approach · Co-design · Agent Simulation · Energy Efficiency of Algorithms · Parallel Algorithm · Supercomputers

1 Introduction

The modern stage of supercomputer development is characterized by the emergence of many projects on creation of an exascale-class supercomputer. Thus far, developments in the field of exascale supercomputers are conducted by different teams of developers in the United States. The collaboration in this direction is carried out, for example, by national laboratories of the US Department of Energy: Sandia and Oak Ridge. In Europe, there are also similar programs; seven European countries have signed the declaration of the Joint Project EuroHPC aimed at the creation of exascale supercomputers. In Japan (the RIKEN institution), the development of a supercomputer has already begun in, where it will be assembled and installed.

There are numerous international projects to develop the system and application software for exascale-class supercomputers with the participation of the United States, countries of the European Union, Japan, China, Russia (IESP, G8 EXASCALE,

CRESTA, etc.). In [1-3], a review and various approaches to exascale scientific software design are given. In [4], the problems typical of exascale systems are listed, such as insufficient concurrent work available to maintain high utilization of all resources, time-distance delay intrinsic of parallel actions and resources on the critical execution path, which is not necessary in a sequential version, delay due to the lack of availability of oversubscribed shared resources. Also, in [4] various methods for overcoming the above-mentioned problems are discussed.

It should be pointed out that numerical algorithms are being developed slower than hardware. Therefore existing algorithms and programs for solving physical problems will be applied at the first stage of using exascale-class supercomputers.

We have offered the integrated approach to the development of algorithms and software for petascale- and exascale-class supercomputers [5]. It contains the three stages.

The first stage is the co-design, which is based on the development of a parallel computational technology, with allowance for all aspects of parallelism. The co-design of parallel methods for solving large-scale problems is difficult to formalize. It is impossible to make a "collection of recipes" for the efficient solution of any problem. However, some general approaches can be proposed. The co-design approach concept consists of the following steps, with allowance for the target hardware/software platform:

1) Formulation of the physical statement of the problem;
2) Mathematical formulation of the physical problem;
3) Development of the numerical methods;
4) Selection of data structures and parallel algorithms;
5) Consideration of a supercomputer architecture;
6) Usage of code optimization tools.

We use the extended definition of the co-design, in contrast to the common conception which consists in the joint development of software and hardware. In such an approach, not only the comparison of the problem solution methods is becoming relevant but also the comparison of the efficiency of using various physical and mathematical statements [5].

The second stage is the anticipated development of algorithms and software for the most promising exascale supercomputers. This stage is based on the simulation of the algorithm behavior within a certain supercomputer architecture. For the simulation of distributed systems it is best to use distributed simulation based on message passing.

The multi-agent approach [6] is used for the simulation of parallel programs run on a large number of cores due to such properties as decentralization, self-organization, and intelligent behavior [7]. Among many multi-agent simulation platforms the adaptable distributed simulation system AGNES [8] was chosen. It was successfully used in the scalability study of a series of parallel problems when executed on a large number of cores. [5,9].

The third stage is estimating the energy efficiency of the algorithm with different implementations for a single architecture or for different supercomputer architectures. In this paper, the term «energy efficiency for scientific HPC applications» means the most efficient use of each core, processor or computational accelerator; the minimiza-

tion of communications between computational nodes; a good workload balancing of the program. The minimization of communications enables a decrease in the idle standing time for processors and accelerators. The good workload balancing enables a uniformly load of a computational system. The most energy-efficient algorithm gives the best FLOPS per Watts (Joules/sec) value.

The efficiency of this approach is illustrated on some examples of complex computing problems in seismology, plasma physics and astrophysics.

2 Using the Integrated Approach to Solving an Elastodynamic Problem

The numerical modeling of elastic wave propagation in heterogeneous 3D media with complex subsurface geometries is a complex problem in terms of computation, thus demanding the use of efficient methods of parallelization and scaling of algorithms. Quite often the topography of various real geophysical objects does not allow one to maintain an observational system. Therefore, constructing their 3D models requires solving the inverse problem by solving a set of direct problems: for different values of the elastic parameters of a heterogeneous medium; at various geometries of objects composing a model. This complicates the problem in the context of computation.

We apply the above-discussed approach to solving the problem of seismic wave propagation in a heterogeneous medium typical of magmatic volcanoes. Both active and sleeping volcanoes are potentially dangerous to the environment due to the possibility of sudden catastrophic eruptions. Using methods of the active vibroseismic monitoring of magmatic structures will allow predicting a probable time of eruption.

For the purpose of the co-design, we have made a comparison of the developed parallel implementations of solutions to the elastodynamic problem written in terms of the velocities of displacement and stress and in terms of displacements for the computational clusters, equipped with graphics cards. The simulation domain is considered to be an isotropic 3D-inhomogeneous elastic structurally complex medium which is a parallelepiped, one of whose sides is a free surface.

At the step of designing a numerical method, the most "flexible" and widespread technique for solving a three-dimensional elastodynamic problem is a finite difference method. Let us preliminarily notice that explicit finite difference schemes fit the architecture of graphics accelerator, because they are directly mapped on the topology of GPU architecture, and involve independent computations of values at each step and in each cell of the computational domain. In order to numerically solve elastodynamics equations in terms of the velocities of displacement and stress we apply the well-known Verrier finite difference scheme on a staggered grid [10]. The calculation of its difference coefficients is based on integral conservation laws. To solve the problem in terms of displacements, we use a similar finite difference scheme [11].

We should note the main difference between the algorithms, which can be constructed on the above-mentioned finite difference schemes. The calculation of the velocities of displacement and stress requires a larger memory size (at least, 18 3D arrays with the unknowns should be stored), but requires a smaller number of float-

ing-point operations in total (57 operations for calculating the values in a cell for one time step). The calculation of the displacements requires a smaller memory size (at least, 6 3D arrays with unknowns), but a larger number of operations (98 operations for one time step).

As software parallelizing tools we have chosen CUDA and MPI, which make possible to simultaneously use the largest number of parallel processes and ultimately to attain a maximum efficiency.

At the step of adaptation to a hybrid cluster equipped with GPU (for example, NKS-30T+GPU, which was installed in the Siberian Supercomputer Center and consists of 40 computational nodes, each one equipped with two six-core CPU Xeon X5670 and three NVIDIA Tesla M2090 graphics cards) we have implemented the next operation for the both statements implementations [11]. For carrying out the parallelization, we decompose the computational domain to layers along one of the coordinate axes. Each layer is calculated at a separate node, where, in turn, it is subdivided into sub-layers along another coordinate axis (to attain a better scaling) according to the number of graphics accelerators at a node. In order to minimize the time of the data exchange, the data are transferred among nodes using appropriate non-blocking asynchronous functions of MPI, and exploiting the asynchronous copy function of CUDA for exchanges among the graphics cards. Let us note that the data for the exchange have an equal size in both approaches.

The numerical experiments have shown that the time of the calculations of the displacements and the time of the calculations of the velocities of displacement and stress at an equal number of nodes is roughly the same in spite of the fact that displacements calculation is performed with a larger amount of floating point operations at the each time step. Therewith the displacement calculations requires almost half as much of the GPU memory size. Based on the results obtained we prefer using the approach proposed to calculating the displacements.

Results of numerical simulation for the truncated model of the volcano Elbrus are presented in Fig.1. For numerical simulation, a spatial grid of 1360x701x2600 nodes and 15000 time steps were used. The calculation was carried out on 20 hybrid nodes of the NKS-30T+GPU cluster during 5 hours. One can learn more about the geophysical model of the volcano and the results of numerical experiments in [12].

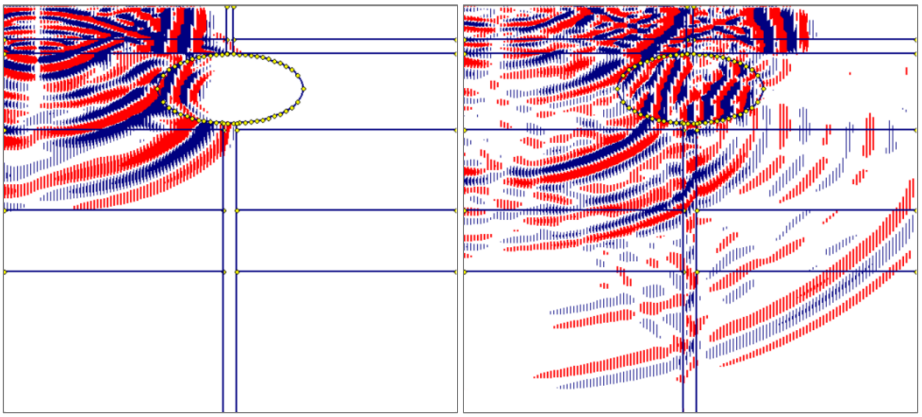

Fig. 1. Results of numerical simulation for the truncated model of the volcano Elbrus. In the snapshots of the wave field, the component u of the displacement velocities vector is presented in the plane Oxz at different time points.

3 Using the Integrated Approach to Solving Astrophysical Problems

In this paper, we use a multi-component hydrodynamic model of galaxies considering the chemodynamics of molecular hydrogen and cooling in the following form. A detailed description of this model can be found in [13].

At the first stage of the co-design procedure, we define the main physical process of a problem. In the case of astrophysics, this process is hydrodynamics. For the description of hydrodynamics, the hyperbolic equations are used. There are many grid numerical methods for solving the hyperbolic equations [13,14]. Some of these methods can be effectively realized by the computational domain decomposition. With adding the subgrid physics (e.g., cooling/heating, chemodynamics, a magnetic field), the structure of the equations remains hyperbolic. For the characterization of collisionless components, the first moments of the Boltzmann equation [14-16] can be used. In this case, a uniform numerical method can be used for solving hydrodynamic and collisionless components. It is possible to use the conjugate gradient method for the Poisson equation solution, which is successfully adopted in the HERACLES code [17]. The use of conformal mappings allows the construction of a moving mesh for solution detailing.

The numerical method of solving hydrodynamic equations is based on a combination of the operator splitting approach, the Godunov method with a modification of the Roe averaging, and a piecewise-parabolic method on a local stencil [18]. The redefined system of equations is used to ensure a non-decrease of entropy and for speed corrections. The detailed description of the numerical method can be found in [19].

Our AstroPhi code is based on the methods in question. We have taken the Intel Xeon Phi accelerator architecture into account for our code and controlled the scala-

bility using the Intel Vectorization Advisor software tool. We use the RSC PetaStream architecture 8-node engineering prototype with 64x Intel Xeon Phi 7120D accelerators for the simulation. Our tests show that 10% of the total simulation time is spent on MPI/OpenMP send/receive operations. This value is suitable for massively parallel systems.

Fig. 2 shows the expansion of two gas clouds after the galaxy collision. One of the possible scenarios is realized: one galaxy flying through the other formating two gas clouds and H2 formation zone after the impact.

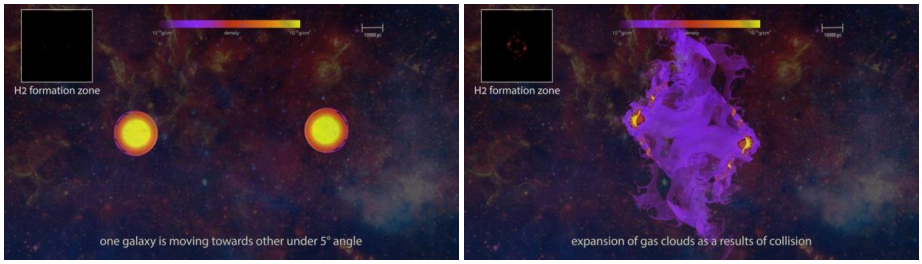

Fig. 2. The galaxy collision AstroPhi test with chemodynamics: Initial stage (on the left), Expansion of gas clouds after collision and H_2 formation zone (on the right).). In the numerical simulation, cluster RSC PolyTechnik (32 Intel Xeon Phi 7150) with 1024^3 mesh during 48 hours was used.

4 Using the Integrated Approach to Solving the Plasma Physics Problems

One of the most interesting problems in the fusion physics is the resonance interaction of a powerful electron beam with plasma. This problem has many practical applications such as studying the processes in the outer layers of the Sun, a fast ignition scheme in inertial fusion and plasma heating in tokamaks.

Let us consider the required computational resources to run a 3D kinetic plasma simulation that are necessary in the above-mentioned plasma physics problems. A rough estimate is a mesh with 100^3 nodes with 1000 particles each. A particle in a 3D case has 6 attributes: 3 double precision variables for 3 components of the coordinate vector and for the impulse vector, finally, 48 bytes. Then this requires about 0.05 terabyte for 1 billion model particles.

The number of flops of Particle-In-Cell method consumes about 250 floating point operations per particle (the number, of course, depends on the code details) during one time step. This results in 2.5 TFLOPS per one time step. Usually, from 10^4 to 10^6 time steps are required for a simulation, that is from 25 PFLOPS to 2500 PFLOPS total.

In the present paper, the following physical statement of the problem is used. The 3D computational domain has the shape of a cube with the following dimensions:

$$0 \le x \le L_X,\ 0 \le y \le L_Y,\ 0 \le z \le L_Z. \quad (1)$$

Within this domain there is model plasma. The model plasma particles (superparticles) are uniformly distributed within the domain. The density of plasma is set by the user as well as the electron temperature. The temperature of ions is considered to be zero. Electrons of the beam are also uniformly distributed along the domain. Thus, the beam is considered to be already present in the plasma, and the effects that occur while the beam is entering the plasma, are beyond the scope of this study.

A 3D kinetic study of the relaxation processes caused by the propagation of an electron beam in high-temperature plasma was carried out (Lotov et al., Phys.Plasmas, 2015) using the Vlasov-Maxwell equation system.

This problem has two different spatial scales: the Debye length of plasma and the beam-plasma interaction wavelength, that is, some 10 or 100 times larger, thus one needs high-performance computing to observe the two lengths at once. The numerical model is built based on the Particle-in-Cell (PIC) method.

Fig. 3 shows the heat flow of plasma electron after the beam relaxation (that is, after beam electrons have given their energy to plasma and have mixed with plasma electrons). The values in Fig. 3 are normed to the initial heat flow value. It is seen in the picture that the resulting heat flow is one or two orders lower than the magnitude in certain places in the computational domain as compared to the initial value. It principally corresponds to the physics of the process.

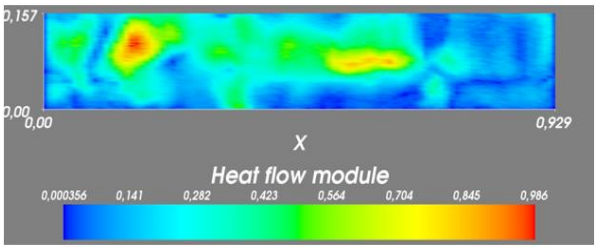

Fig. 3. The heat flow in plasma after beam relaxation in the simulation. The plasma parameters are typical of a magnetic mirror trap.

In the case under consideration the co-design begins at the stage of the physical consideration of the problem. The absence of dramatic density modulations makes it possible not to use the dynamic load balancing. The next stage is the numerical method design. Here, the FDTD method was chosen to provide the memory locality. At the stage of selecting a supercomputer architecture, the PIC method details are taken into account. Particle data and field data are stored in the same place in RAM. At the stage of selecting the software design tools the co-design is the following. For the PIC method, the use of the CUDA technology is highly efficient. Other parallel technologies for hybrid supercomputers such as OpenCL, OpenMP, OpenACC could also be used, but it is CUDA that provides the possibility to employ the highest number of parallel processes and to gain the highest performance. The last stage of the co-design is the adaptation of the algorithm to the GPU architecture.

In order to attain the best scalability, the algorithm was parallelized by means of the mixed Lagrange-Eulerian domain decomposition; the details can be found in

(Snytnikov A.V., Procedia Comp.Sci. 2009). The parallelization efficiency exceeds 90% for 500 nodes [5], the simulations were conducted with "Lomonosov" supercomputer in Moscow State University, the mesh size in plasma simulations 100×4×4, 10 thousand model particles in each cell.

Considering the energy efficiency, one must notice that GPUs are not just faster, but they also consume less energy per Flops. In such a way, the Particle-In-Cell plasma simulation with GPUs is at least one order of magnitude better in terms of the energy efficiency. The algorithm was also implemented for GPU clusters. The particle push (the most time-consuming part of the algorithm) is computed 160 times faster with Nvidia Kepler K40 and 4000 times faster with Nvidia P100 (Pascal) as compared to one core of the Intel Xeon E5450 processor. The simulations with Kepler architecture were conducted in the Siberian Supercomputer Center, the mesh size is 100^3, 100 model particles in each cell, and test runs with Pascal were conducted with Nvidia cluster.

The integrated approach applied to the plasma simulation results in more portable, better scalable and energy efficient codes because of bearing in mind all the three issues all the time when solving the problem.

5 The Results of Using the Integrated Approach

The development of power-efficient algorithms is one of important problems on the way to exascale computations. The CPU power modeling [20,21] and the code level power efficiency optimizations [22] are well-studied issues. The results in [23] show that the computation performance is unaffected by a decrease in the CPU frequency, i.e. the execution time is independent of a change in the CPU frequency, but the power efficiency has been significantly improved with each frequency step as the CPU frequency changes from 2.67 to 1.60 GHz for the matrix multiplication algorithm on an ordinary Intel Core i7 CPU. However, for GPU computing, the paradigm of power modeling research and code optimization must change to incorporate such parameters as CPU efficiency, GPU efficiency and Bus efficiency between GPU and CPU.

Taking into account the above features for the geophysics code, we have attained 9 GFLOPS/W and 12 GFLOPS/W energy efficiency for the displacement problem and the stress problem tests, respectively, on Nvidia Tesla K40M GPU. We have also achieved 4.3 GFLOPS/W and 4.5 GFLOPS/W energy efficiency for the same problems on Nvidia Tesla 2090M GPU without changing the code. Fig. 4 shows arithmetic and logic unit (ALU) utilization as well as memory utilization for the displacement problem solver of the geophysics code (on the left) and for the stress problem solver of the geophysics code (on the right).

For carrying out the simulation of execution of a parallel algorithm, let us present the computational process as a set of threads that are executed in parallel at an individual node and which interact with each other through the exchange of values (messages). The main characteristic of threads is the execution time and the time of data exchange with another computing node. For the simulation of execution of computational processes on supercomputer a model of the interaction of certain process

threads is composed. In the model, the calculation and communication blocks are separated for each thread.

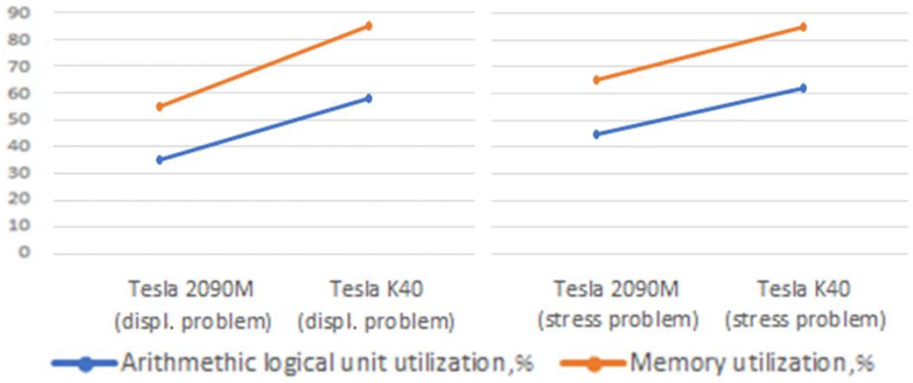

Fig. 4. The ALU and memory utilization for the geophysics code.

Scalability is an important parameter with regard to the HPC-oriented algorithm efficiency. Scalability criteria are algorithm-dependent, for grid-based numerical simulations it keeps the execution times being constant with a simultaneous increase both in the number of node and simulation area size.

The thread interaction model is used for the multiagent simulation of the computational process. Threads are grouped according to their common behavior: the computations and the data exchange. For each thread group there is a class of software agents simulating computation blocks execution and message passing, each with a corresponding number of instances. It is worth noting that the AGNES simulation system allows any functional agents to exchange messages through a "yellow pages" service [8], which supports any thread interaction topologies.

The scalability of the resulting algorithms has been tested using the AGNES simulation system. We have considered utilizing GPU-equipped supercomputers for solving the elastodynamic problem, GPU and MIC architectures for solving the astrophysical problem, and MPP and GPU architectures for the plasma physics problem.

The results of the research into the numerical simulation scalability of the elastic wave propagation problem in complex media are presented in Fig. 5. Two algorithms that solve this problem in two ways are studied. From the figures it is clear that the scalability of the two approaches slightly differs, and both algorithms are suitable for execution on a large number of cores.

The results of the research into the numerical simulation scalability of 3D gas objects in a self-consistent gravitational field are shown in Fig. 6. The solution of this problem is studied on various types of computing nodes. As can be seen from the graphs, the algorithm shows a fairly good scalability. A better scalability is attained in calculations on nodes with accelerators from NVIDIA Kepler K40 and Intel Xeon Phi.

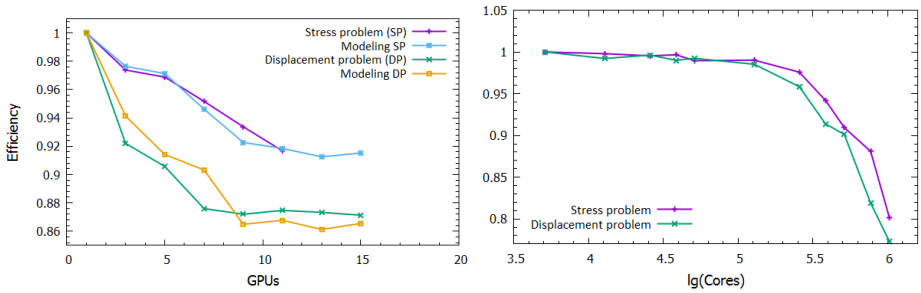

Fig. 5. The scalability of numerical modeling of the elastic wave propagation problem in complex media based on the AGNES simulation. The real calculations for verification were performed with the Siberian Supercomputer Center.

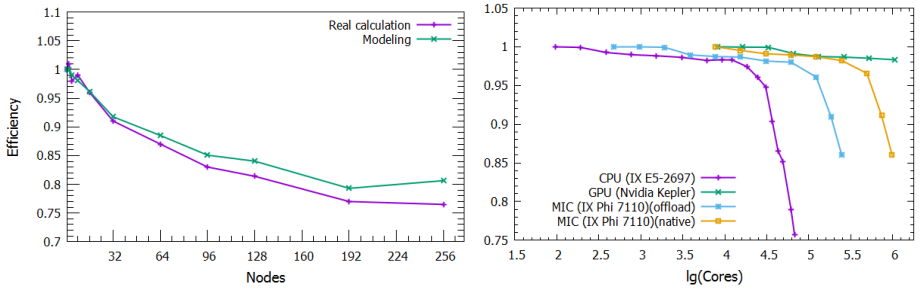

Fig. 6. The scalability of numerical modeling of 3D gas objects in a self-consistent gravitational field based on the AGNES simulation. The real calculations for verification were performed with “Polytechnic RSC PetaStream” supercomputer in St. Peterrsburg State Polytechnical University.

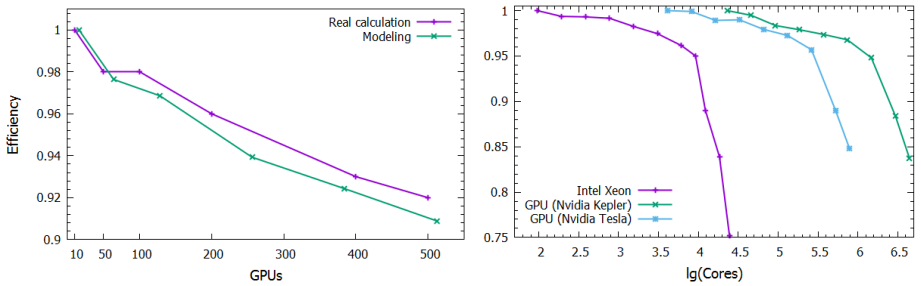

Fig. 7. The scalability of numerical modeling of the interaction of an electron beam with plasma based on the AGNES simulation. The real calculations for verification were performed with “Lomonosov” supercomputer in Moscow State University.

The results of the research into scalability of the numerical simulation of the interaction of an electron beam with plasma are shown in Fig. 7. As a step of forming the full matrix current density charge and the necessary exchange of values "all-with-all",

it is noticeable that these exchanges significantly reduce the efficiency of the algorithm. The solution to this problem is investigated on different types of computing nodes. As can be seen a fairly good scalability is attained in calculations on nodes with accelerators from NVIDIA Kepler K40.

6 Conclusion

In this paper we propose the integrated approach to the development of algorithms and software for solving physical problems demanding a large amount of calculations. For the purpose of the co-design, we have made a comparison of the developed parallel implementations of solutions with the elastodynamic problem written in different terms for the hybrid clusters, equipped with graphics cards. The Godunov method with a modification of the Roe averaging, and a piecewise-parabolic method on a local stencil, has been chosen from several different approaches to solving the astrophysical problem. A similar approach has been taken for the plasma physics problem. The scalability of the resulting algorithms has been tested using the AGNES simulation system. In our case, it is possible to determine carrying out the simulation the optimal number of cores for a specific architecture. This allows investigating the algorithm scalability without resorting to direct time-consuming computations. The energy efficiency of the algorithm for the elastodynamic problem has been investigated on supercomputers equipped with Tesla 2090M and K40M GPUs.

As a result, a suite of parallel programs for solving physics problems has been developed based of the described approach. It is capable of carrying out 3D simulations in acceptable time, provided the resources are sufficient.

Acknowledgments. This work was supported by the Russian Foundation for Basic Research, grants 15-01-00508, 16-29-15120, 16-07-00434, 16-01-00455 and the Grants of the President of the Russian Federation for the support of young scientists MK – 1445.2017.9, MK – 152.2017.5. The plasma code development was supported by the Russian Science Foundation under grant 16-11-10028.

References

1. Reed, D.A., Dongarra, J.: Exascale computing and big data. Comm. of the ACM 58 (7), 56–68 (2015).
2. Keyes, D.E.: Exaflop/s: The why and the how. C.R. Mechanique 339, 70–77 (2011).
3. Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., Wessel, D., Yelick, K.: A view of the parallel computing landscape. Comm. ACM. 52, 56–67 (2009).
4. Sterling, T.: Achieving scalability in the presence of asynchrony for exascale computing. Adv. In Parall. Comp 24, 104–117 (2013).
5. Glinskiy, B.M., Kulikov, I.M., Snytnikov, A.V., Chernykh, I.G., Weins, D.: A multilevel approach to algorithm and software design for exaflops supercomputers (in Russian). Vychisl. Metody Programm. 16, 543–556 (2015).

6. Wooldridge, M.: Introduction to MultiAgent Systems. JOHN WILEY & SONS, LTD, England (2002).
7. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley (2007).
8. Podkorytov, D., Rodionov, A., Choo, H.: Agent-based Simulation System AGNES for Networks Modeling: Review and Researching. In: Proc. of the 6th Int. Conference on Ubiquitous Information Management and Communication (ACM ICUIMC 2012), ISBN 978-1-4503-1172-4, pp. 115. ACM (2012).
9. Glinsky, B.M., Marchenko, M.A., Mikhailenko, B.G., Rodionov, A.S., Chernykh, I.G., Karavaev, D.A., Podkorytov, D.I., Vins, D.V.: Simulation Modeling of Parallel Algorithms for Exaflop Supercomputers (in Russian). Information Technologies and Computer Systems. 4, 3–14 (2013).
10. Bihn, M., Weiland, T.: A Stable Discretization Scheme for the Simulation of Elastic Waves. In: Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics (IMACS 1997). vol. 2, pp. 75–80. Berlin (1997) .
11. Sapetina, A.F.: Supercomputer-aided comparison of the efficiency of using different mathematical statements of the 3D geophysical problem. The Bulletin of NCC. Series: Numerical Analysis 18, 1–9 (2016).
12. Glinskii, B.M., Martynov, V.N., Sapetina, A.F.: 3D Modeling of Seismic Wave Fields in a Medium Specific to Volcanic Structures. Yakutian Mathematical Journal. 22(3), 84–98 (2015).
13. Vshivkov, V.A., Lazareva, G.G., Snytnikov, A.V., Kulikov, I.M., Tutukov, A.V.: ApJS 194, 47 (2011).
14. Kulikov, I.M.: ApJS. 214, 12 (2014).
15. Mitchell, N., Vorobyov, E., Hensler, G.: MNRAS 428, 2674–2687 (2013).
16. Vorobyov, E., Recchi, S., Hensler, G.: A&A. 579, A9 (2015).
17. González, M., Audit, E., Huynh, P.: A&A. 464, 429–435 (2007).
18. Popov, M., Ustyugov, S.: Computational Mathematics and Mathematical Physics. 48, 477–499 (2008).
19. Kulikov, I., E. Vorobyov, E.: Journal of Computational Physics. 317, 318–346 (2016).
20. Lowenthal, D., Supinski, B., Schulz, M., Adagio: Making DVS practical for complex HPC Barry Rountree. In: The 23rd International Conference on Supercomputing, ICS, New York (2009).
21. Ravi, S., Raghunathan, A. Chakradhar, S.T.: Efficient RTL power estimation for large designs. In: Proceedings of the 16th International Conference on VLSI Design, pp. 431–439. New Delhi, India, January (2003).
22. Lively C. et al.: E-AMOM: An Energy-Aware Modeling and Optimization Methodology for Scientific Applications on Multicore Systems. Computer Science—Research and Development 29(3) 197–210 (2014).
23. Da Qi Ren: Algorithm level power efficiency optimization for CPU–GPU processing element in data intensive SIMD/SPMD computing. J. Parallel Distrib. Comput. 71, 245–253 (2011).