Application of an integral approach to the parallel algorithm of 3D wave fields simulation in generalized coordinates

Pavel Titov, Dmitry Weins, and Igor Chernykh

Institute of Computational Mathematics and Mathematical Geophysics SB RAS

This paper considers peculiarities of the parallel algorithm for 3D wave fields simulation in the case of generalized coordinates. For example, when free surface (interface solid/air) of studied domain has a complex topography and we solve the problem with the aid of building a curvilinear mesh. This way of problem statement leads to changes in equations. This in turn imposes a significant change on numerical implementation of the parallel algorithm. In the course of solving the problem, we used a concept of co-design, which allows us to optimize both the parallel algorithm and the program for better performance. Numerical calculations were conducted on the SSCC cluster of the SB RAS. Simulation tools are applied to study algorithm behavior on a large number of cores. Means of Intel Advisor were utilized to analyse the algorithm performance and energy efficiency. The results are presented.

Keywords: Integral approach, co-design, parallel algorithm, 3D, energy efficiency, simulation

1 Introduction

For the past few years, exascale and big data are at the very center of interest of supercomputer community. One can get familiar with it [1]-[4] While the architecture of exaflop supercomputer is still being discussed, one thing is clear: it is how necessary to develop algorithms and software for such systems now. The software should be able to work effectively with hundreds of thousands, even millions of processors, and to store large amounts of data. This is where an integral approach to algorithm development comes to. We consider an integral approach to be a sum of elements listed below:
1. The co-design: an adaptation of both the algorithm and the mathematical method to the supercomputer architecture at every stage of the problem solution.
2. Simulation modeling: development of the preemptive algorithms and software based on simulation tests on different architectures.
3. Energy efficiency of the algorithm: analysis of energy consumption, efficiency of the memory and the network resources usage, as well as the computation cores load. An integral approach to creating algorithms and software for high performance computing (HPC) is being developed at the ICMMG SB RAS.

It was successfully applied for problems of astrophysics [5] and geophysics [6]. In these papers the system AGNES [7] was used for simulation of the algorithm performance on massive parallel systems. The difference of this paper from [5,6] is that we utilize Erlang istead [8]. As [9] shows, on larger number of cores (1 million cores and higher) the Erlang provides more reliable results than the AGNES. It is first time that an integral approach and the simulation tools of the Erlang would be applied for a geophysical problem stated in generalized coordinates. Earlier generalized coordinates for the wave field modeling were used in the papers [10,11] for 2D case, as well as in the papers of one of co-authors [12,13] for 2D case and [14] for 3D case. In [14] also the co-design concept was applied. In general, we consider an integral approach to be a very promising direction and an inseparable part of solving large problems on HPC. In this study, Section 2 covers the co-design approach; Section 3 - the problem statement in both Cartesian and generalized coordinate systems; Section 4 - the method chosen for the numerical solution of the problem; Section 5 - simulation modeling and it's application to the algorithm developed; Section 6 - the energy efficiency and performance of the algorithm; Section 7 - the conclusion to this paper.

2 The Co-design Approach

The co-design is a set of recommendations that allows one to optimize a numerical solution of the problem on a specific supercomputer architecture. It is difficult to provide one with universal list of rules, but a general approach can be offered. It can be said that the co-design consists of several steps:
1) Formulation of the physical statement of the problem: the choice of the physical model according to the problem statement and the possibility of the given supercomputer architecture implementation;
2) Mathematical formulation of a physical problem: the choice of a mathematical model for accurate description of the physical processes. The mathematical model must take the details of the supercomputer architecture into account;
3) Numerical methods development: not all methods that can be applied for the numerical solution of the problem have the same scalability potential on different architectures;
4) Selection of the parallel algorithms and data structures: the way the data is stored and used impacts greatly on the algorithm performance;
5) Applying of the code optimization tools: it allows finding bottlenecks of the algorithm performance. Using this information, the algorithm can be further optimized.
We always consider the joint development of software and hardware. Therefore the comparison of the efficiency of using various physical and mathematical problem statements becomes of importance, not only the comparison of the solution methods. As an example, we can refer for paper [15].

3 Mathematical Model

The co-design approach was used through the whole process of solving the problem. In this study, the simulation of 3D wave fields in the different media models is carried out based on the numerical solution of the linear elasticity system stated in terms of displacements. System in Cartesian coordinates is shown in (1), where $(u, v, w)^T$ is the displacement vector, ρ is a density, λ, μ are Lame parameters, $(F_x, F_y, F_z)^T$ is the mass force vector that represents a source of perturbation.

$$\begin{aligned} \rho\frac{\partial^2 u}{\partial t^2} &= \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + \frac{\partial \sigma_{xz}}{\partial z} + F_x \\ \rho\frac{\partial^2 v}{\partial t^2} &= \frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial z} + F_y \\ \rho\frac{\partial^2 w}{\partial t^2} &= \frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + F_z \end{aligned} \tag{1}$$

where
$\sigma_{xx} = (\lambda + 2\mu)\frac{\partial u}{\partial x} + \lambda\frac{\partial v}{\partial y} + \lambda\frac{\partial w}{\partial z}$, $\sigma_{yy} = \lambda\frac{\partial u}{\partial x} + (\lambda + 2\mu)\frac{\partial v}{\partial y} + \lambda\frac{\partial w}{\partial z}$,
$\sigma_{zz} = \lambda\frac{\partial u}{\partial x} + \lambda\frac{\partial v}{\partial y} + (\lambda + 2\mu)\frac{\partial w}{\partial z}$, $\sigma_{xy} = \mu\frac{\partial u}{\partial y} + \mu\frac{\partial v}{\partial x}$, $\sigma_{xz} = \mu\frac{\partial u}{\partial z} + \mu\frac{\partial w}{\partial x}$,
$\sigma_{yz} = \mu\frac{\partial v}{\partial z} + \mu\frac{\partial w}{\partial y}$ are the components of the stress tensor $\overline{\sigma}$.
Condition on the free surface ∂S: $\overline{\sigma} \cdot \overline{n} = 0$, or in scalar form

$$\begin{aligned} n_x\left((\lambda + 2\mu)\frac{\partial u}{\partial x} + \lambda\frac{\partial v}{\partial y} + \lambda\frac{\partial w}{\partial z}\right) + n_y\left(\mu\frac{\partial u}{\partial y} + \mu\frac{\partial v}{\partial x}\right) + n_z\left(\mu\frac{\partial u}{\partial z} + \mu\frac{\partial w}{\partial x}\right) &= 0 \\ n_x\left(\mu\frac{\partial u}{\partial y} + \mu\frac{\partial v}{\partial x}\right) + n_y\left(\lambda\frac{\partial u}{\partial x} + (\lambda + 2\mu)\frac{\partial v}{\partial y} + \lambda\frac{\partial w}{\partial z}\right) + n_z\left(\mu\frac{\partial v}{\partial z} + \mu\frac{\partial w}{\partial y}\right) &= 0 \\ n_x\left(\mu\frac{\partial u}{\partial z} + \mu\frac{\partial w}{\partial y}\right) + n_y\left(\mu\frac{\partial v}{\partial z} + \mu\frac{\partial w}{\partial y}\right) + n_z\left(\lambda\frac{\partial u}{\partial x} + \lambda\frac{\partial v}{\partial y} + (\lambda + 2\mu)\frac{\partial w}{\partial z}\right) &= 0 \end{aligned} \tag{2}$$

where $(n_x, n_y, n_z)^T$ is the unit normal to the free surface.
Conditions at the inside boundary of the domain $\partial\Gamma$ are:

$$u|_{\partial\Gamma} = v|_{\partial\Gamma} = w|_{\partial\Gamma} = 0 \tag{3}$$

The initial conditions are the following:

$$u|_{t=0} = v|_{t=0} = w|_{t=0} = 0, \quad \left.\frac{\partial u}{\partial t}\right|_{t=0} = \left.\frac{\partial v}{\partial t}\right|_{t=0} = \left.\frac{\partial w}{\partial t}\right|_{t=0} = 0 \tag{4}$$

In general, a computational domain is 3D and includes inhomogeneous isotropic medium with a complex free surface topography. In our case, we solve the problem in generalized coordinates, and (1),(2) must be rewritten accordingly.
We state (q^1, q^2, q^3) to be the new generalized coordinates, and J is Jacobian of the mapping $(x, y, z) \to (q^1, q^2, q^3)$.

Now (1) will transform into:

$$\begin{aligned}
\rho\frac{\partial^2 u}{\partial t^2} &= \frac{1}{J}\left[\frac{\partial\tilde{\sigma_1}}{\partial q^1}+\frac{\partial\tilde{\sigma_2}}{\partial q^2}+\frac{\partial\tilde{\sigma_3}}{\partial q^3}\right]+F_x\\
\rho\frac{\partial^2 v}{\partial t^2} &= \frac{1}{J}\left[\frac{\partial\tilde{\sigma_4}}{\partial q^1}+\frac{\partial\tilde{\sigma_5}}{\partial q^2}+\frac{\partial\tilde{\sigma_6}}{\partial q^3}\right]+F_y\\
\rho\frac{\partial^2 w}{\partial t^2} &= \frac{1}{J}\left[\frac{\partial\tilde{\sigma_7}}{\partial q^1}+\frac{\partial\tilde{\sigma_8}}{\partial q^2}+\frac{\partial\tilde{\sigma_9}}{\partial q^3}\right]+F_z
\end{aligned} \tag{5}$$

where

$\tilde{\sigma_1} = J\left(\sigma_{xx}\frac{\partial q^1}{\partial x}+\sigma_{xy}\frac{\partial q^1}{\partial y}+\sigma_{xz}\frac{\partial q^1}{\partial z}\right)$, $\tilde{\sigma_2} = J\left(\sigma_{xx}\frac{\partial q^2}{\partial x}+\sigma_{xy}\frac{\partial q^2}{\partial y}+\sigma_{xz}\frac{\partial q^2}{\partial z}\right)$,
$\tilde{\sigma_3} = J\left(\sigma_{xx}\frac{\partial q^3}{\partial x}+\sigma_{xy}\frac{\partial q^3}{\partial y}+\sigma_{xz}\frac{\partial q^3}{\partial z}\right)$, $\tilde{\sigma_4} = J\left(\sigma_{xy}\frac{\partial q^1}{\partial x}+\sigma_{yy}\frac{\partial q^1}{\partial y}+\sigma_{yz}\frac{\partial q^1}{\partial z}\right)$,
$\tilde{\sigma_5} = J\left(\sigma_{xy}\frac{\partial q^2}{\partial x}+\sigma_{yy}\frac{\partial q^2}{\partial y}+\sigma_{yz}\frac{\partial q^2}{\partial z}\right)$, $\tilde{\sigma_6} = J\left(\sigma_{xy}\frac{\partial q^3}{\partial x}+\sigma_{yy}\frac{\partial q^3}{\partial y}+\sigma_{yz}\frac{\partial q^3}{\partial z}\right)$,
$\tilde{\sigma_7} = J\left(\sigma_{xz}\frac{\partial q^1}{\partial x}+\sigma_{yz}\frac{\partial q^1}{\partial y}+\sigma_{zz}\frac{\partial q^1}{\partial z}\right)$, $\tilde{\sigma_8} = J\left(\sigma_{xz}\frac{\partial q^2}{\partial x}+\sigma_{yz}\frac{\partial q^2}{\partial y}+\sigma_{zz}\frac{\partial q^2}{\partial z}\right)$,
$\tilde{\sigma_9} = J\left(\sigma_{xz}\frac{\partial q^3}{\partial x}+\sigma_{yz}\frac{\partial q^3}{\partial y}+\sigma_{zz}\frac{\partial q^3}{\partial z}\right)$.

Here we offer only the σ_{xx} in detail for a reader to get a general idea of how equations look like:

$$\sigma_{xx} = (\lambda+2\mu)\sum_{i=1}^{3}\frac{\partial q^i}{\partial x}\frac{\partial u}{\partial q^i}+\lambda\sum_{i=1}^{3}\frac{\partial q^i}{\partial y}\frac{\partial v}{\partial q^i}+\lambda\sum_{i=1}^{3}\frac{\partial q^i}{\partial z}\frac{\partial w}{\partial q^i}.$$

The rest components of the stress tensor $\overline{\sigma}$ are to be transformed in the same manner. Here $(x^1,x^2,x^3)=(x,y,z)$, $J = det\left(\frac{\partial x^i}{\partial q^j}\right)$,
$\frac{\partial q^i}{\partial x^j} = \frac{1}{J}\left(\frac{\partial x^{j+1}}{\partial q^{i+1}}\frac{\partial x^{j+2}}{\partial q^{i+2}}-\frac{\partial x^{j+1}}{\partial q^{i+2}}\frac{\partial x^{j+2}}{\partial q^{i+1}}\right)$ with cyclical numeration $i,j=1,2,3$.
Components $\frac{\partial q^i}{\partial x^j}$ are so called metrical coefficients defined by the shape of a curvilinear mesh. The reader can get the components of the unit normal vector in generalized coordinates himself, by using the substitution of variables described earlier.
Boundary conditions (3) and initial conditions (4) remain the same.

4 The Numerical Method and Parallel Implementation

To provide a numerical dynamic model of wave fields in the given media within a given domain, one must restore the displacement vector at every instant of time. In this paper, it was decided to utilize the finite difference method for the spatial and time components. Judging from the results in [9], it has a good scalability potential. For the 3D mesh construction, we used the algebraic method, considered in details in pages 53-64 [16]. The choice is based on the fact that this method allows a good asynchronous parallel realization of the mesh generator.

Numerical realization of this algorithm provides second order of accuracy.
A difference analog of equations (5) with corresponding boundary, surface and initial conditions was constructed. The explicit scheme provides the second order of accuracy in both spatial and temporal directions.
The use of the method presented for solving the problem is associated with calculations of a large amounts of 3D data, so the important point is the use of a 3D decomposition of the domain and further parallel realization of the algorithm. In this case, the domain is to be divided into 3D blocks. Each of the blocks is located directly in the RAM of the computing device. By computing device, in general, one should understand a CPU, a GPU, or a Xeon Phi. In this paper, a CPU is used as a computing device. For arrays with u, v, w one needs to organize data exchange between adjacent blocks. The communications are carried out through 3D cube topology presented in Fig. 1. Each block has 26 adjacent

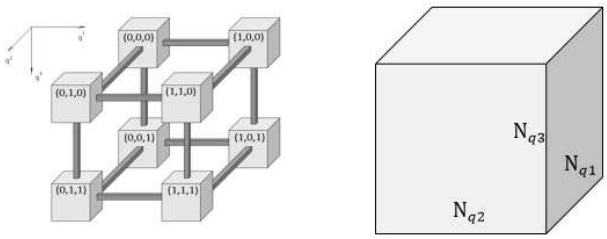

Fig. 1: The topology (left) and the size of each cube (right)

neighbors, therefore, 26 messages are to be send at every time step. The MPI tools were utilize for such purpose. Non-blocking messages $ISend, IRecv$ as well as blocking messages $Send, Recv$ were used.
A program version of the algorithm was created with the aid of the Fortran language and the MPI library. One should know, that unlike in the most widely used language C++, in the Fortran multidimensional arrays are stored differently. This must be taken into account when organizing nested cycles.

5 The Simulation and Scalability of the Algorithm

Even though supercomputers of exascale level are not available right now, we can study the scalability of an algorithm for such systems via means of the simulation. The distributed simulation based on the transmission of messages is best suited for imitation of the distributed systems. To simulate the execution of parallel programs on a large number of cores, a multi-agent approach was used [17]. To minimize the overhead load in the communication of the agents, the Actor model was utilized [9]. To study the scalability of the algorithm, a scheme of interacting processes was created, and the main temporal characteristics of these processes were obtained. The amount of data in each process is a cube of the size

$N_{q^1} \times N_{q_2} \times N_{q^3}$ elements (Fig. 1). The topology of interprocess communication is a 3D cube. In this paper the Erlang is being used.
The program can be presented as a time function, that depends on the input parameters:
- the scheme of calculations in the program;
- how long does it take to calculate one element and how many elements are there in the domain of interest;
- how many time steps does it take to execute the program;
- network parameters such as a topology and a latency of communications.
Knowing these parameters we can simulate the program performance on a large multi-core system (hundreds of thousands and even millions of threads). A model for the execution of a parallel algorithm on a computational system has been developed. Each computational thread is represented by a separate computing actor, which completely imitates the course of computations and message exchanges with other calculator actors, according to the created topology of communications. To verify the models obtained, a simulation experiment was conducted on the algorithm execution for a small number of computational cores (from 12 to 120). Verification results are presented in Fig. 2. As one can see, a comparison of data on the real performance and the results of a simulation experiment allows us to say that the model corresponds with the parallel algorithm execution. The real run of the parallel program was performed on SSCC SB RAS, SL390S G7 servers (each has 2x6 core Xeon X5670 2.93GHz and 96Gb RAM).

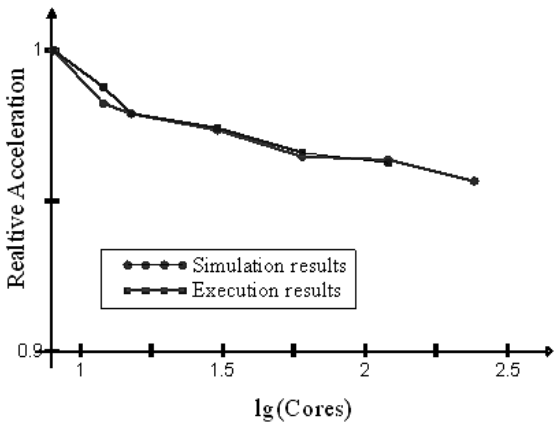

Fig. 2: Verification of the algorithm execution model (horizontal axis is in logarithmic scale)

To study the scalability of the parallel algorithm, a model experiment of its execution was performed on a large number of computational cores (up to 10^7). To study the effect of the data size, the simulation results are presented in Fig. 3

We can conclude that the algorithm under study scales well on a large number of

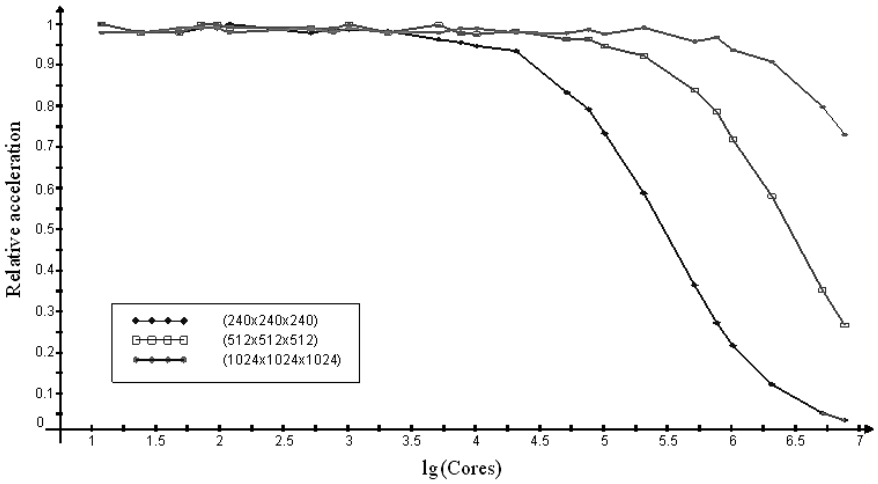

Fig. 3: Scalability of the algorithm depends on the number of simulated cores and the size of the computational cube for each core (horizontal axis is in logarithmic scale)

computational nodes only under the condition of large-block partitioning of the computational domain. Such results indicate that the program does not work efficiently with RAM and data exchange. This factor is the bottleneck of the performance. Further optimization is required.

6 Performance Analysis and Energy Efficiency

When we are talking about efficient employing of a massively parallel systems, we should touch on "energy efficiency". In this paper, this means the most efficient use of given computational resources and minimization of communications between computational cores that do not share the same RAM. The workload on the cores should be balanced and idle standing time minimized. The most energy effective algorithms have the highest FLOPS per Watts (Joules/sec) values. For test the NKS-1P computational node of SSCC SB RAS (2x Intel Xeon E5-2697A-v4 CPUs and 256GB of DDR4 memory) was used. For performance analysis, we used the Application Performance Snapshot of Intel Vtune Amplifier 2019 software and the Intel Fortran Compiler for code compilation and linking with O3 optimization parameter. For this code, we achieved 89% AVX-2 auto-vectorization of double precision floating point operations. Floating point operations per memory read instructions ratio is 0.5 and floating point operations per memory write instructions ratio is 2.1. We have no I/O operations for our code. The double precision operations performance of the code is presented

in the Table 1. Hyperthreading on 2 CPUs gave 13% speedup of the code. We achieved 10-15% of peak performance for this type of CPU with AVX-2 instructions. This code has 2 main bottlenecks: memory stalls and the MPI Barrier operations. The MPI Barrier operations take 26% of the computational time for 2xCPU configuration. In future work, we will change the algorithm to minimize the MPI operations time and will remove memory stalls. Due to the bottlenecks, the energy efficiency is 0.19GFLOPS/W with peak 1.5-1.7 GFLOPS/W for this type of CPU with using of AVX-2 instructions.

Table 1: Code performance

Configuration	Perfomance (GFLOPS)
1x Intel Xeon 2697A-v4 (16 cores, no HT, 16 threads)	24.18
2x Intel Xeon 2697A-v4 (32 cores, no HT, 32 threads)	47.21
2x Intel Xeon 2697A-v4 (32 cores, HT, 64 threads)	54.44

7 Conclusion

In this paper we have presented an integral approach to development of the algorithm and the software for solving the elastodynamic problem in generalized coordinates. The approach proposed consists of 3 parts: the co-design at every stage of the problem solving, the simulation modeling for scalability parameters of the algorithm and the results of energy efficiency tests. In our case, we have chosen the problem statement in terms of displacements statement versus in terms of velocities of displacement. This reduces the requirement for RAM. The finite difference method, that possesses a high scalability potential, was utilized for creating numerical analog of the algorithm. Its parallel program version was created via means of the Fortran language with the aid of the MPI library. Calculations were conducted on SSCC SB RAS. The simulation was carried out via means of the Erlang language and the results are presented. The tests for energy efficiency did show us the bottlenecks of both the algorithm and the program developed. Based on it, further steps towards improvement of overall program performance are to be taken.

8 Acknowledgements

The theoretical part of this work (Sections 1-3) was conducted within the framework of the budget project 0315-2019-0009 for ICMMG SB RAS, and for the technical part, Sections 4, 5 and 6, were supported by the RFBR grants 19-07-00085, 18-37-00279 and 18-07-00757 respectively.
The Siberian Branch of the Russian Academy of Sciences (SB RAS) Siberian

Supercomputer Center is gratefully acknowledged for providing supercomputer facilities.

References

1. Reed, D.A., Dongarra, J.: Exascale computing and big data. Comm. ACM 58(7), 2015; pp. 5668.
2. Dongarra, J.J., et al.: The international exascale software project roadmap. Int. J. High Perf. Comp. App. 25(1), 2011; pp. 360.
3. Tabbal, A., Anderson, M., Brodowicz, M., Kaiser, H., Sterling, T.: Preliminary design examination of the parallex system from a software and hardware perspective. Sigmetrics Perform. Eval. Rev. 38(4), 2011; pp. 8187.
4. Sterling, T.: Achieving scalability in the presence of asynchrony for exascale computing. Adv. Parall. Comp. 24, 2013; pp. 104117.
5. Boris Glinsky, Igor Kulikov, Igor Chernykh, Dmitry Weins, Alexey Snytnikov, Vladislav Nenashev, Andrey Andreev, Vitaly Egunov, Egor Kharkov: The Co-design of Astrophysical Code for Massively Parallel Supercomputers. ICA3PP 2016 Workshops, LNCS 10049, 2016; pp. 342353.
6. Boris Glinskiy, Igor Kulikov, Igor Chernykh, Alexey Snytnikov, Anna Sapetina, Dmitry Weins: The Integrated Approach to Solving Large-Size Physical Problems on Supercomputers. Russian Supercomputer Days, Proceedings, 2017; pp. 278-289.
7. Podkorytov, D., Rodionov, A., Sokolova, O., Yurgenson, A.: Using agent-oriented simulation system AGNES for evaluation of sensor networks. In: Vinel, A., Bellalta, B., Sacchi, C., Lyakhov, A., Telek, M., Oliver, M. (eds.) MACOM 2010. LNCS, vol. 6235, pp. 247250.
8. Cesarini F., Thompson S.: Erlang Programming. O'Reilly Media, 2009. pp. 498, ISBN-10: 0596518188 ISBN-13: 978-0596518189.
9. Weins Dmitry, Glinskiy Boris, Chernykh Igor: Analysis of Means of Simulation Modeling of Parallel Algorithms. Russian Supercomputer Days , Proceedings, 2018; pp. 64-75.
10. Daniel Appelo, N. Anders Petersson: A Stable Finite Difference method for the Elastic wave equation on complex geometries with free surfaces. Commun. Comput. Phys. Vol.5, No. 1, 2009; pp. 84-107.
11. Jose M. Carcione: The wave equation in generalized coordinates. GEOPHYSICS, VOL. 59, NO. 12, 1994; pp. 1911-1919.
12. Titov P.: An Algorithm and a program for simulation of 2D-wave fields in areas with a curved free surface. Materials of the conference "Scientific service on the Internet - 2014" Novorossiysk, Abrau-Dyurso, September 21-26, 2014; pp.446-455. (In Russian)
13. Titov P.: Modeling of elastic waves in media with a complex free surface topography. Vestnik of NSU: Information Technologies, vol. 16, c. 4, 2018; pp. 153-166. (In Russian)
14. Titov P.: A technology of modeling of elastic waves propagation in media with complex 3D geometry of the surface with the aid of high performance cluster. Russian Supercomputing Days, Proc., 2016, pp. 1020-1031.
15. Sapetina A.: Comparing efficiency of using different mathematical statements of a problem for 3D simulation of seismic wave field at supercomputer. Russian Supercomputing Days, Proceedings, 2018; pp. 985-995.

16. Liseykin V.: Differential meshes. Theory and applications. SB RAS Publ., 2014; 254 pp. (In Russian)
17. I. Wooldridge, M.: Introduction to MultiAgent Systems England. JOHN WILEY and SONS, LTD, 2002.