Vectorization of astrophysical code for massively parallel supercomputers *

I. Kulikov[1], I. Chernykh[1], B. Glinskiy[1], A. Shmelev[2], A. Andreev[3], V. Egunov[3], E. Kharkov[3], V. Nenashev[4]

ICMMG SB RAS[1], RSC Technology[2], Volgograd State Technical University[3], Novosibirsk State Technical University[4]

We describe a new version of our AstroPhi code for the simulation of astrophysical objects dynamics and other physical processes on hybrid supercomputers equipped with Intel Xeon Phi accelerators. The new version of the AstroPhi code was rewritten in accordance with co-design technique. Thus, we used the latest knowledge about the last Intel Xeon Phi generation during code development. The most significant code changes are concerned on the vectorization. The vectorization technique for astrophysical code is described. The results of AstroPhi acceleration using an Intel Xeon Phi-based massively parallel supercomputer are presented in this paper. Some galaxy collisions incorporating chemodynamics problems and spiral galaxy formation tests are presented as a demonstration of the AstroPhi code.

Keywords: astrophysics simulation, scalable parallel algorithms, massively parallel architecture.

1. Introduction

Numerical modeling plays a key role in modern astrophysics. It is the main tool for the research of nonlinear processes and provides communication between the theory and observational data. Numerical simulation in astrophysics allows detailed investigation of the collision and evolution of galaxies.

The main parts of a galaxy collision simulation are Newtonian gravitation and hydrodynamics. Modern supercomputers have given us the possibility of subgrid-scale astrophysics modeling that considers different physical effects such chemical kinetics, cooling/heating, and more. One of the most interesting developments in supercomputer technology at this moment is massively parallel supercomputers. The main concept of this technology is based on the possibility of massive usage of computation accelerators. Modern supercomputers have more computation accelerator cores than CPU cores. However, this technology also has the greatest disadvantage of modern supercomputers – the problem of effective usage of the accelerator cores. In this case, software development is a difficult scientific task that can be realized through a co-design approach. At this moment, many astrophysical codes can be used for simulation. These codes can be divided into two groups: SPH codes (gridless numerical methods) and codes that are based on grid methods[1-5]. A review of these codes with their advantages and disadvantages is reported in [6], and is not discussed here. In this article, we will describe the new version of author's AstroPhi [7] code for the numerical simulation of astrophysical problems on massively parallel supercomputers. There are some physical effects was added in the new version of AstroPhi code, and some improvements for better vectorization and scalability for Intel Xeon Phi native mode was done.

2. Mathematical model

In our work, we use a multicomponent hydrodynamic model of galaxies considering the chemodynamics of molecular hydrogen and cooling in the following form:

* This work was partially supported by RFBR grants 15-31-20150, 15-01-00508, 16-01-00564, 14-01-00392, 16-07-00534, 16-47-340385 and by Grant of the President of Russian Federation for the support of young scientists number MK 6648.2015.9..

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \vec{u} \right) = 0 ,$$

$$\frac{\partial \rho_{H_2}}{\partial t} + \nabla \cdot \left(\rho_{H_2} \vec{u} \right) = S\left(\rho, \rho_H, \rho_{H_2} \right) ,$$

$$\frac{\partial \rho_H}{\partial t} + \nabla \cdot \left(\rho_H \vec{u} \right) = -S\left(\rho, \rho_H, \rho_{H_2} \right) ,$$

$$\frac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot \left(\rho \vec{u} \vec{u} \right) = -\nabla p - \rho \nabla \Phi ,$$

$$\frac{\partial \varepsilon}{\partial t} + \nabla \cdot \left(\varepsilon \vec{u} \right) = -\left(\gamma - 1 \right) \varepsilon \nabla \cdot \left(\vec{u} \right) - Q ,$$

$$\frac{\partial E}{\partial t} + \nabla \cdot \left(E \vec{u} \right) = -\nabla \cdot \left(p \vec{u} \right) - \left(\rho \nabla \Phi, \vec{u} \right) - Q ,$$

$$\Delta \Phi = 4 \pi G \rho ,$$

$$E = \varepsilon + \frac{\rho \vec{u}}{2} ,$$

$$p = \left(\gamma - 1 \right) \varepsilon ,$$

where ρ is density, ρ_H is atomic hydrogen density, ρ_{H_2} is molecular hydrogen density, $\vec{u}$ is the velocity vector, ε is internal energy, p is pressure, E is total energy, γ is the ratio of specific heats, Φ is gravity, G is the gravitational constant, S is the formation rate of molecular hydrogen, and Q is a cooling function. A detailed description of this model can be found in [8].

The formation of molecular hydrogen is described by an ordinary differential equation [9]:

$$\frac{dn_{H_2}}{dt} = R_{gr}\left(T \right) n_H \left(n_H + 2 n_{H_2} \right) - \left(\xi_H + \xi_{diss} \right) n_{H_2} ,$$

where n_H is the concentration of atomic hydrogen, n_{H_2} is the concentration of molecular hydrogen, and T is temperature. Detailed descriptions of the H_2 formation rate R_{gr} and the photodissociation ξ_H, ξ_{diss} of molecular hydrogen can be found in [10,11].

3. Co-design and AstroPhi software architecture

The co-design of parallel methods for the solution of large-scale problems is difficult to formalize. It is impossible to make a “collection of recipes” for the efficient solution of all problems. However, some general approaches can be proposed. The co-design approach concept consists of the following steps, taking into account 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 supercomputer architecture;

6) Code optimization tools usage.

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, hyperbolic equations are used. There are many grid numerical methods for the solution of hyperbolic equations [6-8]. Some of these methods can be effectively realized by the decomposition of the computational area. With the addition of 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 [6,12-14] can be used. In this case, a uniform numerical method can be used for the solution of 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 [15] code. 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 an operator splitting approach, Godunov's method with modification of Roe's averaging, and a piecewise-parabolic method on a local stencil [16, 17]. The redefined system of equations is used to guarantee the nondecrease of entropy [18] and for speed corrections [19]. The detailed description of a numerical method can be found in [20]. It is worth noting that other SPH and AMR astrophysical codes have scalability limitations including 1K cores and 10K cores respectively. Using of above methods gave us weak scalability of 92% for 64x Intel Xeon Phi (15360 cores) accelerators in native mode. The architecture of AstroPhi code is shown on the figure 1.

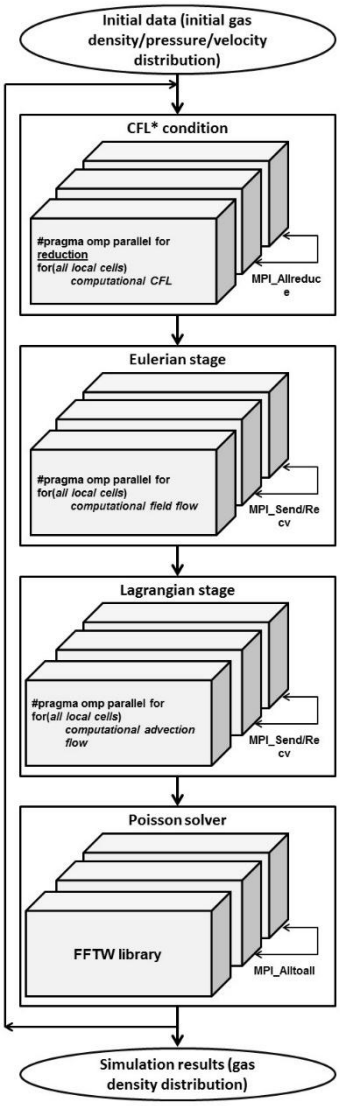


Fig. 1. The AstroPhi code architecture (*Courant—Friedrichs—Lewy condition [21])

4. Vectorization of AstroPhi code

Vectorization is the most powerful method to improve code performance, because the most of the modern CPUs, GPUs and coprocessors are using SIMD architecture. Some of them are combined with multi-threading. We use Intel Xeon Phi coprocessors in our case study. Each core of the Intel Xeon Phi coprocessor has SIMD 512-bit wide Vector Processor Unit (VPU). Each VPU can be used to process 8 double-precision elements per clock cycle. Vectorization of the AstroPhi source code was divided into several stages, which allowed taking advantage of it in more detail:

- check the algorithms for the possibility of vectorization and preliminary assessment of its efficiency.
- primary vectorization of existing code using Intel Intrinsics;
- performance evaluation of code vectorized on previous stage, and its analysis in order to find potential bottlenecks;
- code optimization on the basis of the obtained results;
- the final performance evaluation.

Stage 1.

To validate the vectorization possibility and preliminary assessment of its efficiency, it was decided to use the Intel Vector Advisor. This profiler provides data on the admissibility of vectorization, its effectiveness and the reasons for the impossibility of vectorization of a certain cycle. As a result of analysis by using this application (Fig. 2), an important information was obtained: time distribution of the individual functions and affordability of their vectorization. Execution time of one of the functions is almost 80% of the total execution time, so this function has been selected as the most promising in terms of vectorization.

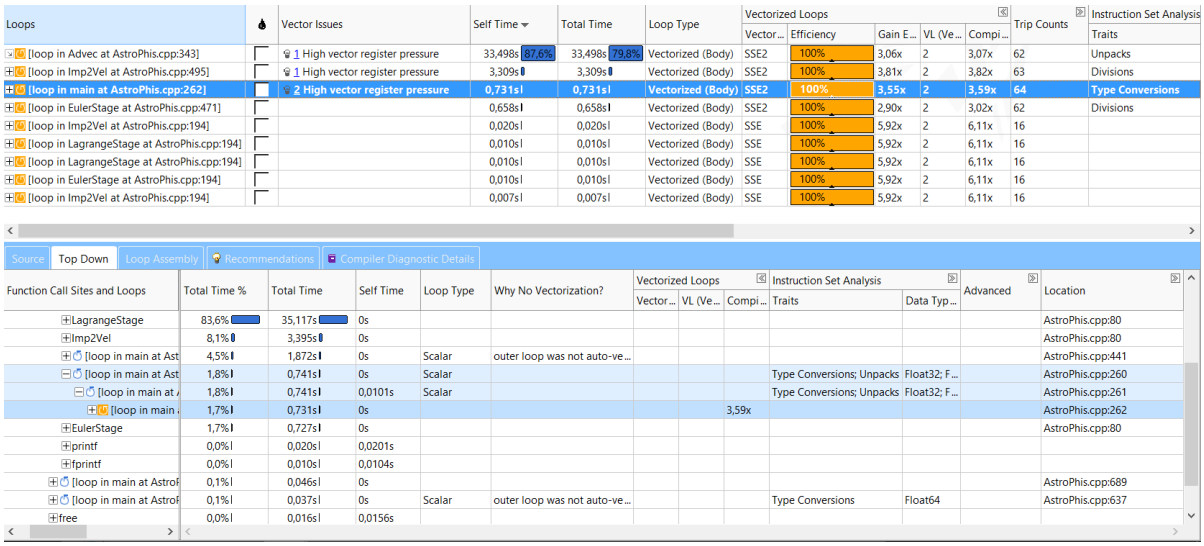


Fig. 2. Result of Intel Advisor

Table 1. Vectorization of AstroPhi code

Initial code	**Vectorized code**

Initial code	Vectorized code

Data operations:
void EulerStage (...) {…
#pragma omp parallel for de-
fault(none) shared(…)private(…)
num_threads(MIC_THREADS)
…
R_diff = …;
for(curr_ind=0 ; curr_ind
<NX;curr_ind++) {
RVx[curr_ind] += -
 	
*taumic*R_diff[curr_ind]/2/hmic;*
}
…}
	Data operations:
	void EulerStage (...) {…
	#pragma omp parallel for
	default(none)
	shared(…)private(…)
	num_threads(MIC_THREADS) …
	R_diff = … ;
	__m512d taumic =
	_mm256_set1_pd(tau / 2.0 /
	hmic)
	for(curr_ind=0 ; curr_ind
	<NX;curr_ind+=8) {
	RVxv =_mm512_load_pd(
	RVx+curr_ind);
	Rdifv =_mm512_load_pd(
	R_diff+curr_ind);
	RVxv=_mm512_fnmadd_pd(
	taumic,Rdifv,RVxv);
	mm512_store_pd(
	RVx+curr_ind,RVxv);
	}
	… }

Stage 2

At this stage the source code was vectorized without any changes in algorithm using Intel Intrinsics. Arithmetic operations have been replaced with the appropriate functions working with vector registers, augmented with data pre-loading into the registers. For vectorization of branches we need to use mask registers. Both branches are calculated and the result is stored in temporary registers, then according to the original comparing condition the mask is formed, based on which two temporary registers are mixed and the result is stored in the source(Fig. 3).

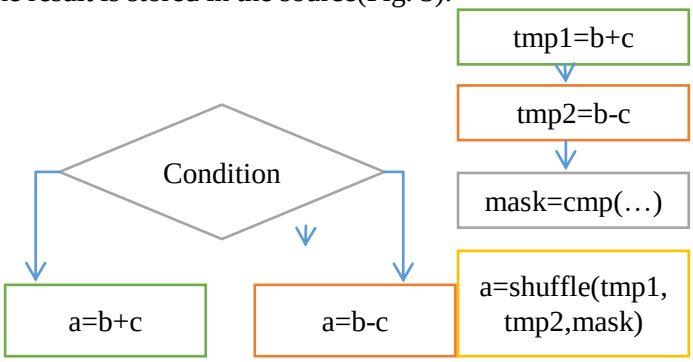


Fig. 3. Vectorization of branches

Stage 3

After vectorization of the source code the 1,8 – times speedup of the objective function was obtained. This speedup is relatively small compared to the theoretically possible 8-times. To determine the reasons for the slowing, the application was analyzed again with the Intel Vector Advisor. As a result of this procedure, a number of locations, potentially slowing down the application have been found, namely:

- division of an expression by a constant;
- loading of an unaligned memory;
- the shuffle operation;
- high registers load.

Stage 4.

Based on the analysis, optimization of a bottlenecks was divided into several stages in turn. At first division by a constant has been replaced with the multiplying by the inverse, several other arithmetic operations were optimized also. Then it became necessary to replace the instructions working with an unaligned memory, pre-aligning the downloaded data to the size of the register. Due to the specific of the algorithm loading of an array starts not from zero-indexed, but with the first element. But allocating of an aligned memory allows only zero-based aligning. To solve this problem, one must allocate one extra vector and shift the pointer (Fig. 4) thereby obtain alignment from the first element.

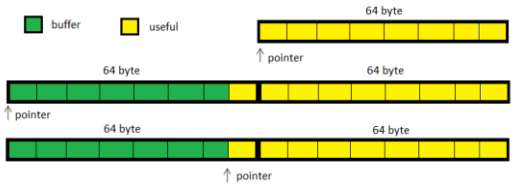


Fig. 4. Alignment of memory

The next step was to optimize the number of memory downloads. To do this, the function call has been replaced with its code that has reduced the number of downloads and optimize cache line hits by means of the load order of the array elements.

The final step was the addition of FMA instructions that perform addition and multiplication in a single clock cycle.

5. Simulation

We use the RSC PetaStream architecture [22] 8-node engineering prototype with 64x Intel Xeon Phi 7120D accelerators for the simulation. We use (p×512)×(512)×(512) grid size for the simulation where p – number of accelerators. Efficiency (weak scalability) , where - computations time on a single accelerator using a single accelerator and - computations time on a single accelerator using p accelerators.

Table 2 shows weak scalability for the RSC PetaStream massively parallel system with 64x Intel Xeon Phi 7120D accelerators.

Table 2. Weak scalability of AstroPhi code

Number of accelerators	Scalability
1	1
2	1.034
4	1.033
8	0.977
16	0.941
32	0.934
64	0.923

After all phases of vectorization, acceleration of the target function up to 6.43 times has been obtained (Fig. 5). Further reduction of the execution time is possible in case of transition to the next generation of KNL processors and AVX-512 instruction set. Based on other studies, this transition can increase the acceleration by 30-40% [23]. Figures 6-7 show astrophysical tests: galaxy collision with chemodynamics and spiral galaxies. The AstroPhi code was tested using all classical tests for astrophysical codes, which can be found in [1,3,15].

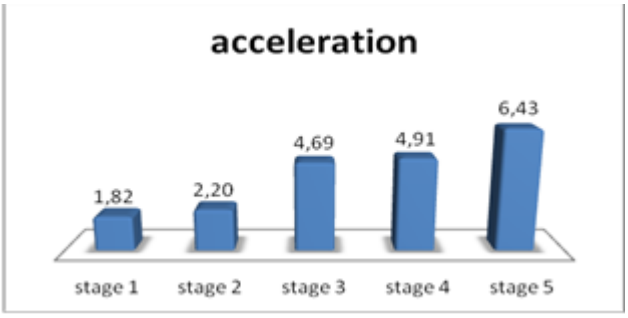
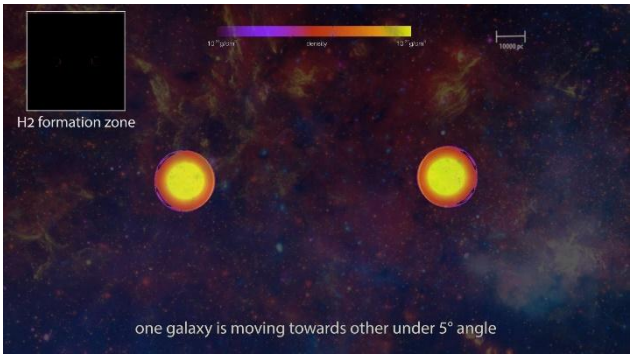


Fig. 5. Acceleration of the AstroPhi code via vectorization. Stage 1 – algorithm optimization. Stage 2 – optimization of arithmetic operations. Stage 3 – optimization of memory operations. Stage 4 – reducing a number of memory load operations. Stage 5 – transition to FMA instructions.

a)



b)

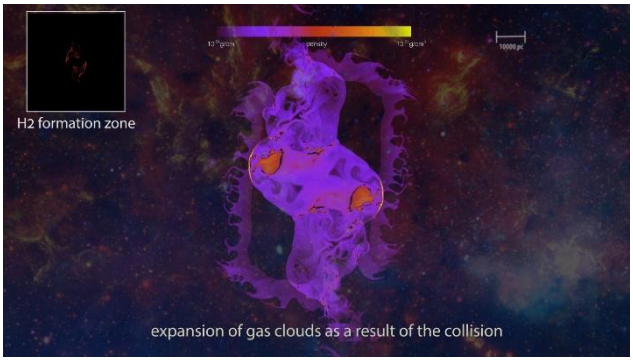
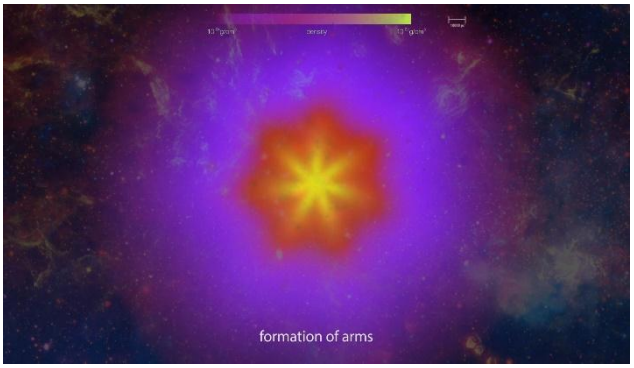


Fig 6. Galaxy collision AstroPhi test with chemodynamics: a) Initial stage, b) Expansion of gas clouds after the collision and H_2 formation zone.

a)



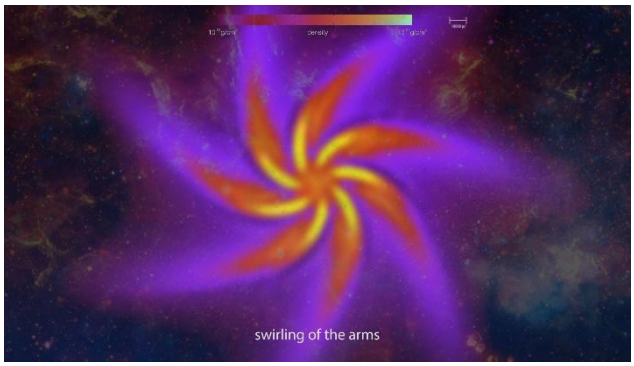
b) 

Fig. 7. Seven-arm swirling galaxy AstroPhi test: a) Formation of arms from the spherical cloud, c) Swirling of arms.

Figure 6 shows the expansion of two gas clouds after the galaxy collision. One of the possible scenarios is realized: one galaxy flying through the other with the formation of two gas clouds and an H_2 formation zone after the impact. Figure 7 shows the swirling galaxy test with the formation of a seven-arm galaxy during the simulation.

5. Conclusion

The AstroPhi code is designed for the simulation of astrophysical object dynamics on neo-heterogeneous supercomputers equipped with Intel Xenon Phi computation accelerators. Our approach is based on a simplification of numerical methods, data structures, and architecture of parallel code with taking into account Intel Xeon Phi hardware features. Because of this approach, our code can be used efficiently on 15K+ cores. The new RSC PetaStream massively parallel architecture is used for numerical simulation tests. We use a co-design technique for software development because the RSC massively parallel supercomputer's architecture has certain peculiarities. Each node of RSC PetaStream includes 8 Intel Xeon Phi accelerators that work as independent computational nodes. There is only one CPU on each RSC PetaStream node, which is used for system support. We achieve 92% of efficiency (weak scalability) for the system with 64 Intel Xeon Phi accelerators.

References

1. Springel V. The cosmological simulation code GADGET-2 // Monthly Notices of the Royal Astronomical Society. 2005. Vol. 364 (4), P. 1105-1134.
2. Wadsley J.W., Stadel J., Quinn T. Gasoline: a flexible, parallel implementation of TreeSPH // New Astronomy. 2004., Vol. 9 (2), P. 137-158.
3. Pearcea F.R., Couchman H.,M.,P. Hydra: a parallel adaptive grid code // New Astronomy. 1997. Vol. 2, P. 411-427.
4. Ziegler U. Self-gravitational adaptive mesh magnetohydrodynamics with the NIRVANA code // Astronomy & Astrophysics. 2005. Vol. 435, P. 385-395.
5. Hayes J., Norman M., Fiedler R., et al. Simulating Radiating and Magnetized Flows in Multiple Dimensions with ZEUS-MP // Astrophysical Journal. Supplement Series. 2006. Vol. 165, P.188-228.
6. Kulikov I.M. GPUPEGAS: A new GPU-accelerated hydrodynamic code for numerical simulations of interacting galaxies // Astrophysical Journal. Supplement Series. 2014. Vol. 214(12), P. 1-12.
7. Kulikov I.M., Chernykh I.G., Snytnikov A.V., Glinskiy B.M., Tutukov A.V. AstroPhi: a code for complex simulation of dynamics of astrophysical objects using hybrid supercomputers // Comp. Phys. Comm. Vol. 186, P. 71-80.

8. Vshivkov V.A., Lazareva G.G., Snytnikov A.V., Kulikov I.M., Tutukov A.V. Hydrodynamical code for numerical simulation of the gas components of colliding galaxies // Astrophysical Journal. Supplement Series. 2011. Vol. 194(47), P.1-12.
9. Bergin E.A., Hartmann L.W., Raymond J.C., Ballesteros-Paredes J. Molecular cloud formation behind shock waves // Astrophys. J. 2004. Vol. 612, P. 921-939.
10. Khoperskov S.A., Vasiliev E.O., Sobolev A.M., Khoperskov A.V. The simulation of molecular clouds formation in the Milky Way // Monthly Notices of the Royal Astronomical Society. 2013. Vol. 428 (3), P.2311-2320.
11. Glover S., Mac Low M. Simulating the formation of molecular clouds. I. Slow formation by gravitational collapse from static initial conditions // Astrophysical Journal. Supplement Series. 2006. Vol. 169, P. 239-268.
12. Kulikov I., Chernykh I., Glinskiy B., Weins D., Shmelev A. Astrophysics simulation on RSC massively parallel architecture // Proceedings - 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015, P. 1131-1134.
13. Mitchell N., Vorobyov E., Hensler G., Collisionless Stellar Hydrodynamics as an Efficient Alternative to N-body Methods // Monthly Notices of the Royal Astronomical Society. 2013. Vol. 428, P. 2674-2687.
14. Vorobyov E., Recchi S., Hensler G. Self-gravitating equilibrium models of dwarf galaxies and the minimum mass for star formation // Astronomy & Astrophysics. 2012. Vol. 579, A129.
15. González M., Audit E., Huynh P. HERACLES: a three-dimensional radiation hydrodynamics code // Astronomy & Astrophysics. 2007. Vol. 464 (2), P.429-435.
16. Popov M., Ustyugov S. Piecewise parabolic method on local stencil for gasdynamic simulations // Computational Mathematics and Mathematical Physics. 2007. Vol. 47, P. 1970-1989.
17. Popov M., Ustyugov S. Piecewise parabolic method on a local stencil for ideal magnetohydrodynamics // Computational Mathematics and Mathematical Physics. 2008. Vol. 48, P. 477-499.
18. Godunov S., Kulikov I. Computation of Discontinuous Solutions of Fluid Dynamics Equations with Entropy Nondecrease Guarantee // Computational Mathematics and Mathematical Physics. 2014. Vol. 54 (6), P. 1012-1024.
19. Vshivkov V.A., Lazareva G.G., Snytnikov A.V., Kulikov I.M., Tutukov A.V. Computational methods for ill-posed problems of gravitational gasodynamics // Journal of Inverse and Ill-posed Problems. 2011. Vol. 19, P. 151-166.
20. Kulikov I., Vorobyov E. New Astronomy. 2016 (submitted).
21. Belmont G., Grappin R., Mottez F., Pantellini F., Pelletier G. Collisionless Plasmas in Astrophysics. 2013. ISBN: 978-3-527-41074-3.
22. RSC PetaStream, http://rscgroup.ru/en/our-solutions/rsc-petastreamr-1pflops-cabinet-massivelly-parallel-supercomputer-mpsc.
23. Andrey Andreev, Andrey Nasonov, Artem Novokschenov, Andrey Bochkarev, Egor Kharkov, Dmitriy Zharikov, Sergey Kharchenko, Alexey Yuschenko Vectorization Algorithms of Block Linear Algebra Operations Using SIMD Instructions // Creativity in Intelligent Technologies and Data Science. CIT&DS 2015 : First Conference (Volgograd, Russia, September 15-17, 2015) : Proceedings / ed. by A. Kravets, M. Shcherbakov, M. Kultsova, O. Shabalina. – [Switzerland] : Springer International Publishing, 2015. – P. 323-341. – (Ser. Communications in Computer and Information Science. Vol. 535).