www.volsu.ru

DOI: https://doi.org/10.15688/mpcm.jvolsu.2024.4.4

UDC 524.7-8
LBC 32.973-018

Submitted: 03.09.2024
Accepted: 17.11.2024

EFFICIENCY OF PARALLEL COMPUTATIONS OF GRAVITATIONAL FORCES BY TREECODE METHOD IN N-BODY MODELS [1]

Nikolay M. Kuzmin

Candidate of Sciences (Physics and Mathematics), Associate Professor,
Department of Information Systems and Computer Simulations,
Volgograd State University
nikolay.kuzmin@volsu.ru
https://orcid.org/0000-0003-4074-0970
Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation

Danila S. Sirotin

Assistant, Department of Information Systems and Computer Simulations,
Volgograd State University
d.sirotin@volsu.ru
https://orcid.org/0000-0001-8956-570X
Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation

Alexander V. Khoperskov

Doctor of Sciences (Physics and Mathematics), Professor,
Department of Information Systems and Computer Simulations,
Volgograd State University
khoperskov@volsu.ru
https://orcid.org/0000-0003-0149-7947
Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation

Abstract. Modeling of collisionless galactic systems is based on the N-body model, which requires large computational resources due to the long-range nature of gravitational forces. The most common method for calculating gravity is the TreeCode algorithm, which provides a faster calculation of the force compared to the direct summation of contributions from all particles for N-body simulation. An analysis of the computational efficiency is performed for models with the number of particles up to 10^8. We considered several processors with different

architectures in order to determine the performance of parallel simulations based on the OpenMP standard. An analysis of the use of extra threads in addition to physical cores shows an increase in simulation performance only when all logical threads are loaded, which doubles the total number of threads. This gives an increase in the efficiency of parallel computing by 20 percent on average.

Key words: parallel computing, gravitational systems, OpenMP, processor architecture, Hyper-Threading technology.

Introduction

The N-body model is used to simulate the dynamics of galactic systems, including open clusters and globular clusters [4; 12], single galaxies, interacting galaxies, groups and clusters of galaxies [1; 2; 6; 9; 14; 15; 19; 23; 32; 33]. There are a significant number of numerical algorithms for parallel implementation of methods for calculating gravitational forces in N-body system. The simplest direct summation method of each particle to each other one (Particle-Particle, PP) greatly limits the number of particles due to the complexity of $O(N^2)$, although it can serve as a tool for testing [9]. The algorithm proposed in [3] based on a special partitioning of the volume with particles is traditionally called TreeCode. It reduces the complexity of calculations to $O(N\,\log(N))$ by reducing the accuracy of the contribution to gravity from distant particles.

A variety of fast methods for calculating the gravitational force in a system of N particles have been proposed over the past decades. We highlight such methods as Fast Fourier Transform (FFT) with complexity close to $O(N\,\log(N))$, Particle-Mesh Method, Particle-Particle + Particle-Mesh ($\mathrm{P^3M}$) and others [2; 20; 35]. The FFT method has a number of advantages for simulating modified gravity [27]. A special feature of the Grid-of-Oct-Trees-Particle-Mesh (GOTPM) algorithm is a hybrid scheme combining particle-mesh (PM) and oct-trees of Barnes-Hut method [7; 21]. A generalization of tree codes is Fast Multipole Method with number of computational operations $O(N)$ [2; 36].

Large cosmological simulations projects aim to study the cosmological evolution of dark and baryonic matter, as well as the physics of galaxy clusters over 10^{10} years based on N-body and/or hydrodynamical simulations [26; 34]. Examples of such projects are MillenniumTNG (3000-megaparsec box with more than 1 trillion particles) [11], PKDGRAV3 ($N = 8 \cdot 10^{12}$) [26], EAGLE simulation (Evolution and Assembly of GaLaxies and their Environments) [28], FLAMINGO project [29], Illustris-TNG [25] and many others.

The use of cosmological models has proven its effectiveness in solving problems of the dynamics of single galaxies or galaxy groups, as in the Auriga project [10] and the Apostle project [8]. The use of high-order parallel N-body codes for GPUs allows one to study the dynamics of globular clusters [13] and compact elliptical galaxies on cosmological time scale [15].

The number of real stars in galaxies is always much larger than the number of particles in the model, and this requires the fulfillment of the collisionless condition of the stellar component in the simulations [17; 30]. Various approaches have been proposed to improve the quality of N-body models. An example is the GalaxyFlow, which aims to create a method to go from very coarse stellar phase-space density in the original numerical simulations to more accurate stellar phase-space density estimates for astronomical applications [22].

The aim of the work is to calculate the efficiency of parallel N-body simulations based on TreeCode. Since the bottleneck of the Message Passing Interface is the speed of data exchange between processors, our analysis is limited to the OpenMP standard within the shared memory. Processors have both physical cores (processor cores) and extra threads (virtual or logical cores) thanks to Hyper-Threading Technology. We present the details of our analysis of the performance of Treecode for calculating gravitational forces using extra threads in addition to the processor cores.

1. N-body simulation with TreeCode

The dynamics of N gravitating particles is described by a system of equations

$$\begin{cases} \dfrac{d\mathbf{r}_i}{dt} = \mathbf{v}_i, \\ \dfrac{d\mathbf{v}_i}{dt} = \displaystyle\sum_{j\neq i}^{N} G\, m_j\, \frac{\mathbf{r}_j - \mathbf{r}_i}{\left(|\mathbf{r}_j - \mathbf{r}_i|^2 + \varepsilon^2\right)^{3/2}}, \end{cases} \tag{1}$$

where $\mathbf{r}_i$, $\mathbf{v}_i$ and m_i are the radius vector, velocity and mass of the i-th particle, respectively, G is the gravitational constant, ε is the smoothing radius that gives a collisionless system. The dynamical N-body model includes only a stellar disc and lacks a dark halo defined by a given gravitational potential, as in [5]. The initial equilibrium in the rotating disc is provided by the balance of gravitational and centrifugal forces with an additional contribution from random particle motion, which is described by spatial distributions of velocity dispersions in three directions [9; 15; 16; 24]. The dispersion of the random velocity component along the vertical coordinate is an analogue of pressure and compensates for the gravitational force, which gives an equilibrium disk in the transverse direction.

The numerical integration of the system (1) is based on the classical second-order leap-frog scheme:

$$\mathbf{v}_i^{n+1/2} = \mathbf{v}_i^n + \frac{\tau}{2}\,\mathbf{a}_i^n, \tag{2}$$

$$\mathbf{r}_i^{n+1} = \mathbf{r}^n + \tau\,\mathbf{v}_i^{n+1/2}, \tag{3}$$

$$\mathbf{a}_i^{n+1} = \mathbf{a}(\mathbf{r}^{n+1}), \tag{4}$$

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^{n+1/2} + \frac{\tau}{2}\,\mathbf{a}_i^{n+1}, \tag{5}$$

where τ is the time step, $\mathbf{a}_i^n$ is the acceleration of the i-th particle, index n corresponds to the time t_n.

Direct summation in the formula (1) for each particle of the contributions from all other bodies (Particle-Particle, PP method) yields a computational complexity of $O(N^2)$. The quadratic dependence of the computational time on the number of particles greatly limits such a naive approach. However, the direct method has a number of advantages. Obviously, the calculation error for PP is minimal compared to approximate algorithms for the same N, which provides a significant advantage for satisfying the physical conservation laws [16; 18]. This allows for a higher-quality reproduction of the evolution of a gravitating system over large times and small scales, which is especially important for the gas component with large mass density gradients [15].

The calculation of gravitational forces in (1) is performed by the TreeCode method [3], based on the idea of an approximate construction of the gravitational potential of "distant" particles group. The contribution of such a group is calculated as for one particle with the total mass of the entire group, located at the center of mass. The contribution of "near" particles is calculated by an exact formula. The implementation of this approach is based on a hierarchical partitioning of the space into eight octants until there is no more than one particle left in each subdomain. The division into "near" and "distant" groups is determined by the value of the parameter θ. It has the meaning of the solid angle at which a given subdomain is visible from the point under consideration. This method has an algorithmic complexity $O(N\,\log(N))$. The TreeCode method turns into a direct method PP for calculating accelerations with algorithmic complexity $O(N^2)$ for $\theta \to 0$.

The key problem is the choice of the criterion when the group of particles is distant [2; 31]. Since the tree code characteristics, the errors in calculating the gravitational forces and the features of the spatial distribution of particles are interconnected.

Calculating gravitational forces takes up the lion's share of the processor time regardless of the method of numerical integration of the equations system. Using the TreeCode method in this work requires over 95 percent of the time to calculate accelerations. This value varies within $2-3$ percent depending on the choice of the computing system.

2. Code performance analysis

We use the FORTRAN-77 version of TreeCode[2] adapted for galactic stellar disc dynamics simulations [9]. This code is parallelized using OpenMP technology.

Using the compiler `gfortran 12.2.0` with the optimization switch `-O3` provides the most complete optimization in terms of execution time without specific extensions of the instruction set architecture (ISA) of the target processor. Aggressive optimization methods are not used, since they may affect the numerical results, such as option `-ffast-math`, which allows deviations from strict compliance with the floating-point number representation standard IEEE 754, for example, in the case of subnormal numbers and the implementation of associativity of computations. This choice is due to the fact that we focused on the study of quantitative characteristics associated with parallel computing and do not consider a specific processor instruction set architecture. Solving mathematical simulation problems makes it preferable to use specific extensions of the instruction set of the target processor (the SSE family of extensions of various versions, AVX, etc.) using switch `-march=native`, which allows using all available instructions supported by the processor and the compiler.

We calculate the following characteristics of software parallelization performance: dependencies of the execution time of one step of the computational cycle T (Time), the computations speedup S (Speedup) and the efficiency of parallelization E (Efficiency) on the number of threads used M. The definitions of these quantities are

$$S(M) = \frac{T(1)}{T(M)}, \quad E(M) = \frac{S(M)}{M}, \tag{6}$$

where E is the efficiency without extra threads.

The parameters T, S, and E are measured on several different shared-memory computers (symmetric multiprocessing, SMP), the parameters of which are shown in Table 1. It is important to note that the processors are capable of dynamically changing the clock

frequency depending on the current computing load level (Intel Turbo Boost and AMD Precision Boost technologies). Additionally, there is the ability to execute two logical computing threads on one physical processor core (Hyper-Threading Technology). The total number of available computing threads $M = M_P + M_E$ consists of physical cores M_P and additional logical computing threads (extra threads) M_E. We compare two types of CPUs, both with support for hyper-threading technology ($M_E = M_P$, $M = 2M_P$) and without it ($M_E = 0$, $M = M_P$). Thus, threads on a single CPU may be nonequivalent, which is almost always a problem for a parallel program. Formulas for $E(M)$ in the form (6) are traditionally used for $M \le M_P$. The efficiency in the case of $M > M_P$ ($M_E \le 1$) should be written in the form

$$E^{(P)}(M) = \frac{S(M)}{M_P} = \frac{T(1)}{T(M)\,M_P}, \tag{7}$$

since only physical cores must be considered, $M_P = M - M_E$.

All results were obtained for averaged values of execution time of one step of the computational cycle T. Averaging was performed over ten steps of time integration. Therefore, we plot absolute errors in the figures as vertical bars. Measurement errors were calculated by using the following expression of standard error of the mean:

$$\Delta(T) = \sqrt{\frac{\sum_{i=1}^{n} \left(T_i - \langle T \rangle\right)^2}{n(n-1)}}, \tag{8}$$

where $\langle T \rangle$ is the mean value of T, $n = 10$ is the number of measurements. The corresponding errors of S and E have larger values because they are calculated as ratios of A/B, which gives

$$\Delta\left(\frac{A}{B}\right) = \frac{A \cdot \Delta(B) + B \cdot \Delta(A)}{B^2}. \tag{9}$$

The main parameter determining the accuracy of the gravitational force calculation in the TreeCode algorithm is the opening angle θ. This free parameter actually specifies the hierarchical structure of the grid on which the gravitational potential is calculated. All numerical experiments in this work are carried out for $\theta = 1$. This choice is typical in the practice of N-body simulations. Reducing θ greatly increases the time T, all other things being equal. There is a transition from TreeCode to PP in the limit $\theta \to 0$.

The results of calculating the average time T, speedup S and efficiency $E^{(P)}$ for different computing systems and the number of threads M are shown in figures 1–8. The simulation time T consists of the execution time of the following components of the algorithm in accordance with formulas (2)–(5): calculating gravitational forces for a system of N particles ($T^{(grav)}$), determining velocities at the predictor stage ($T^{(\mathbf{v1})}$), new coordinates ($T^{(\mathbf{r})}$), velocities at the corrector stage ($T^{(\mathbf{v2})}$) and accelerations ($T^{(\mathbf{a})}$) of particles. Since $T^{(grav)}/(T^{(grav)} + T^{(\mathbf{v1})} + T^{(\mathbf{r})} + T^{(\mathbf{v2})} + T^{(\mathbf{a})}) \simeq 0.96$, we restrict ourselves to the approximation $T = T^{(grav)}$ below.

Table 1

Characteristics of computing systems

Name	CPU	Total number of cores (number of logical threads)	RAM	CPU release date
Beta	2 × Intel Xeon E5405, 2 GHz	8 (8)	16 GB DDR2-667	2007
Epsilon	2 × Intel Xeon E5540, 2.53 GHz	8 (16)	16 GB DDR3-1600	2009
K80	2 × Intel Xeon E5-2687W v3, 3.1 GHz (3.5 GHz with Turbo Boost)	20 (40)	512 GB DDR3-2133	2014
M105	Intel Core i5-6400, 2.7 GHz (3.3 GHz with Turbo Boost)	4 (4)	8 GB DDR3-1600	2015
M111	Intel Core i5-9400, 2.9 GHz (4.1 GHz with Turbo Boost)	6 (6)	16 GB DDR4-2666	2019
M113	12th Gen Intel Core i5-12400F 2.5 GHz (4.4 GHz with Turbo Boost)	6 (12)	16 GB DDR5-5200	2022
M105S	Intel Core i9-9900KF 3.6 GHz (5.0 GHz with Turbo Boost)	8 (16)	64 GB DDR4-2400	2019
M301	2 × Intel Xeon E5-2630 v3, 2.4 GHz (3.2 GHz with Turbo Boost)	16 (32)	16 GB DDR4-1866	2014
Ryzen	AMD Ryzen 7 2700, 3.2 GHz (4.1 GHz with Precision Boost)	8 (16)	16 GB DDR4-3200	2018

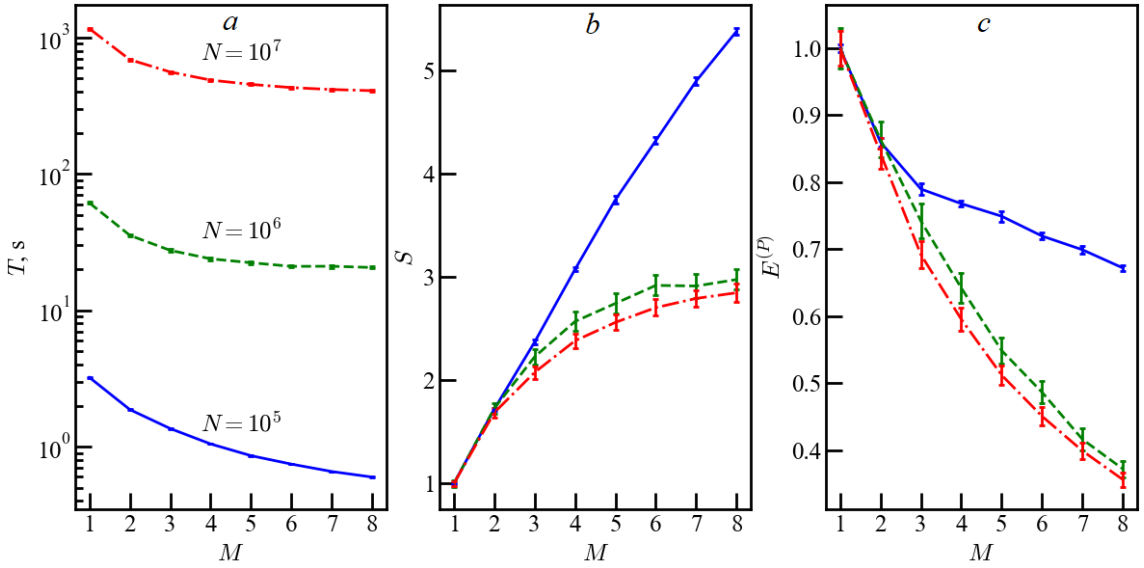

Fig. 1. Computational time (a), speedup (b) and efficiency (c) for computer Beta in simulations with different numbers of particles: $N = 10^5$ (blue), $N = 10^6$ (green), $N = 10^7$ (red)

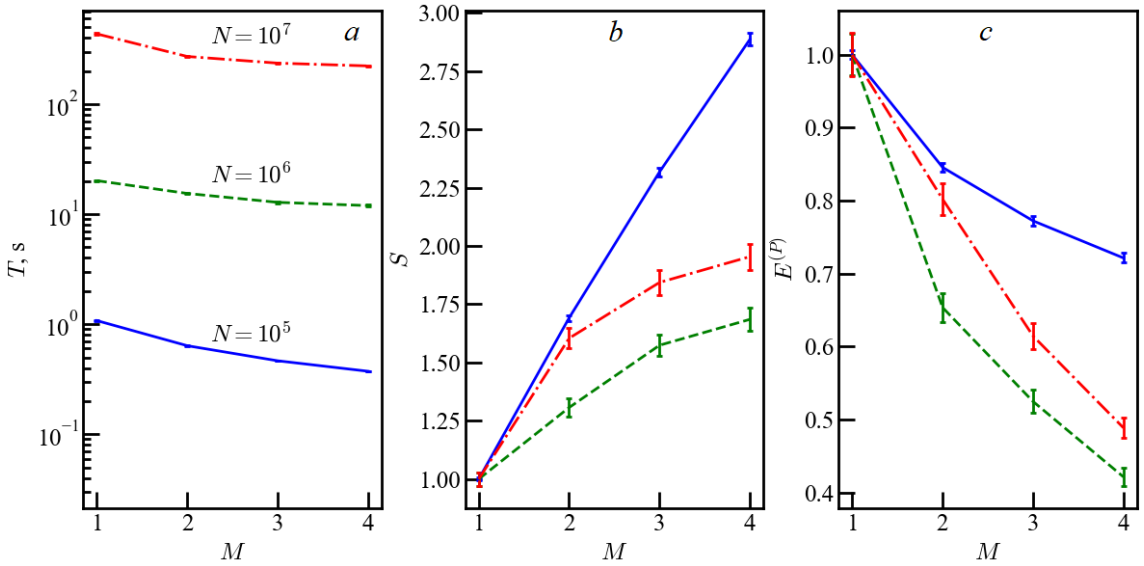

Fig. 2. The same as in figure 1 for computer M105

Computer Beta has two processors with four physical cores each, which gives $M_P = 8$ and $M_E = 0$. The dependencies $T(M)$, $S(M)$, $E^{(P)}(M)$ for the models with $N = 10^5$, $N = 10^6$ and $N = 10^7$ are shown in figure 1. The computation time T decreases noticeably with the increase of the number of cores M for $N = 10^5$. Experiments with $N = 10^6$ and $N = 10^7$ give a slower decrease with an exit to an almost constant level. This, respectively, affects the speedup $S(M)$ and the efficiency $E^{(P)}(M)$. The speedup reaches a plateau at $M \to 8$, in contrast to the simulation with $N = 10^5$ with an almost linear increase in speedup. The efficiency turns out to be worse than 0.4 when using all available physical cores in models with $N \gtrsim 10^6$.

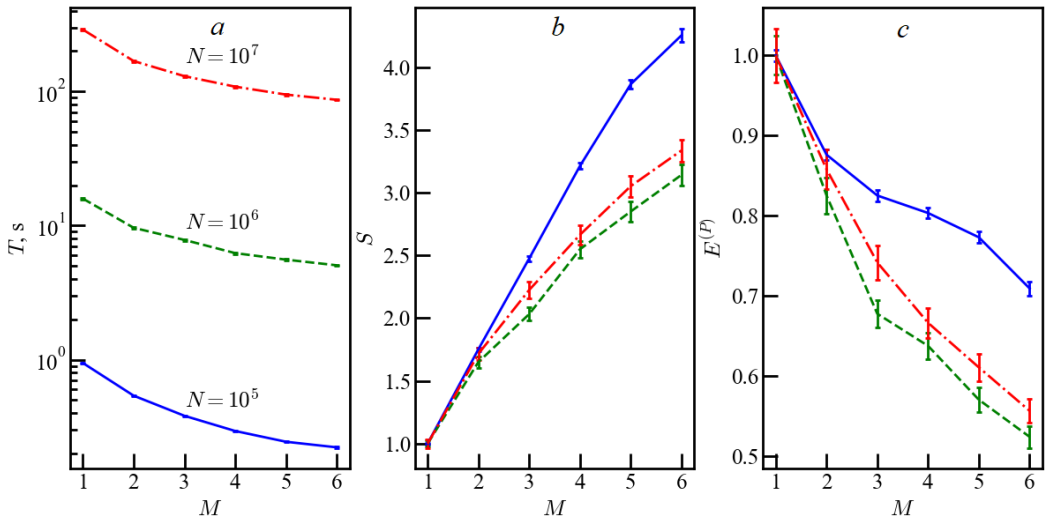

Fig. 3. The same as in figure 1 for computer M111

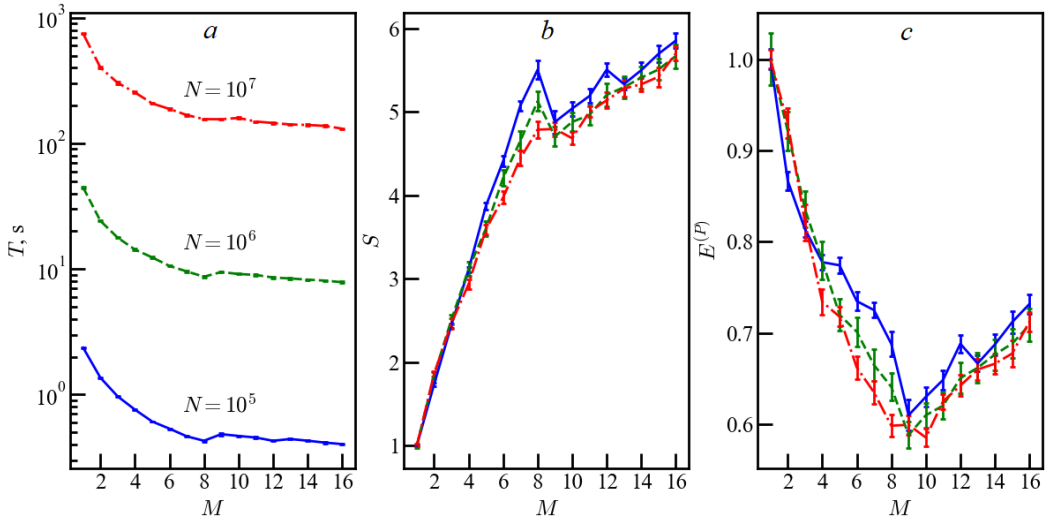

Fig. 4. The same as in figure 1 for computer Epsilon

Other processors without extra threads ($M_E = 0$) have similar dependences of characteristics on the number of cores. The M105 processor is more modern and has a higher clock frequency than Beta. Therefore, the value of T on computer M105 is approximately 3 times less than on Beta (figure 2). However, the speedup and efficiency are better only within 10–15%. Such results are also preserved for the more modern processor of M111 in figure 3. Moreover, the growth of speedup continues to the maximum value of $M_P = 6$. There are higher speedups and efficiency for $N \geq 10^7$ in contrast to computer Beta due to the more modern architecture.

Next, we consider computing systems with $M_E = M_P$, which doubles the number of threads. Computer Epsilon has 8 additional logical threads and $M = 16$. The value of $T(M = 1)$ decreases compared to Beta due to the more advanced processor architecture (figure 4). Using only physical cores ($M \leq 8$) is more efficient for Epsilon than in the case of Beta. The dependences of speedup and efficiency on the number of particles are

significantly weakened (figure 4). The difference in results for all three experiments with $N = 10^5$, $N = 10^6$ and $N = 10^7$ is small. All dependencies change sharply for the number of threads $M > 8$ when using extra threads. Two characteristic features stand out in parallel computing with $M > M_P = 8$.

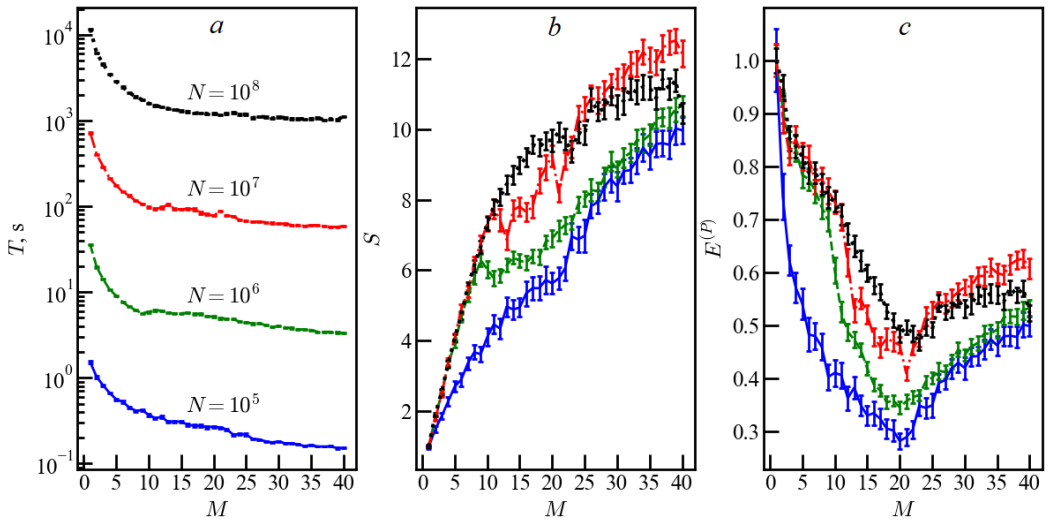

Fig. 5. The same as in figure 1 for computer K80

First, adding several extra threads leads to a noticeable decrease in speedup and efficiency. For example, using extra threads with $M_E = 1 - 2$ increases the time T compared to $M = M_P = 8$ (see figure 4*a*). Characteristic kinks in the dependencies $S(M)$ and $E^{(P)}(M)$ in the vicinity of $M = M_P = 8$ show the negative impact of a small number of extra threads on the overall efficiency of simulations. The second feature appears when using the maximum possible number of extra threads. Such a doubling of the number of threads to $M = 16$ allows exceeding the indicators with $M = M_P = 8$. Comparing the efficiencies of $E^{(P)}(8)$ and $E^{(P)}(16)$ yields an increase of 6 percent for $N = 10^5$ and 19 percent for $N = 10^7$ for computer Epsilon.

The nature of the dependencies $T(M)$, $S(M)$, $E^{(P)}(M)$ for $M \leq 8$ for computer Epsilon (see figure 4) is due to the fact that one logical computing thread is used on one physical processor core. A further increase in threads ($M > 8$) leads to the use of two threads on one core, which violates the monotonic dependence. Computer Beta does not have extra threads ($M = M_P$), which gives monotonicity of efficiency (see figure 1).

Next, we consider computer K80 (see Table 1), including an additional model with $N = 10^8$, which requires more RAM (≈ 6 GB). The total number of threads reaches $M = 40$ for $M_P = 20$ and $M_E = 20$. K80 contains 2 processors with shared memory. Figure 5 shows our results for K80. There are peculiarities when the number of threads passes near $M \simeq 10$ and $M = M_P = 20$. The first is due to the two processors on K80. The transition from $M = 10$ to a higher number of threads means either using the second processor or/and partial calculations on extra threads of the first or second processor. The choice of a specific operating mode is determined by the operating system and is not adjustable by the user.

The internal scheduler does not provide information on the distribution of resources between devices. However, the dependencies $T(M)$, $S(M)$, $E^{(P)}(M)$ allow us to assume that the distribution of computing resources occurs as follows. At the beginning, one logical

computing thread is allocated to one physical core of the first processor ($M \lesssim 10$). Then the second threads are added to each core of the first processor ($10 < M \leq 20$). The next stage begins with the use of the second processor ($20 < M \leq 40$). This transition is especially noticeable for a very large number of particles in models with $M \leq M_P$ (see figure 5). Comparing the efficiencies of $E^{(P)}(20)$ and $E^{(P)}(40)$ yields an increase of 77 percent for $N = 10^5$ and 8 percent for $N = 10^8$ for computer K80.

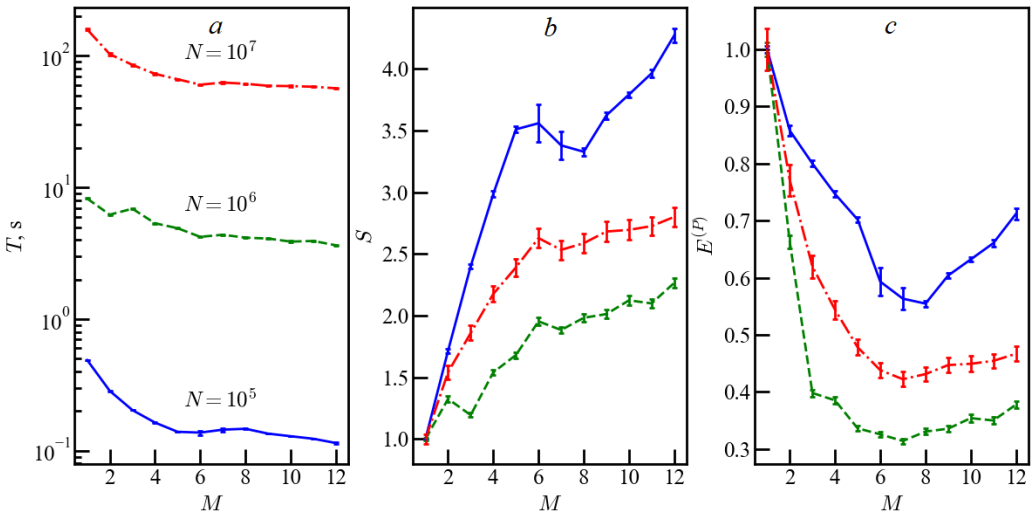

Fig. 6. The same as in figure 1 for computer M113

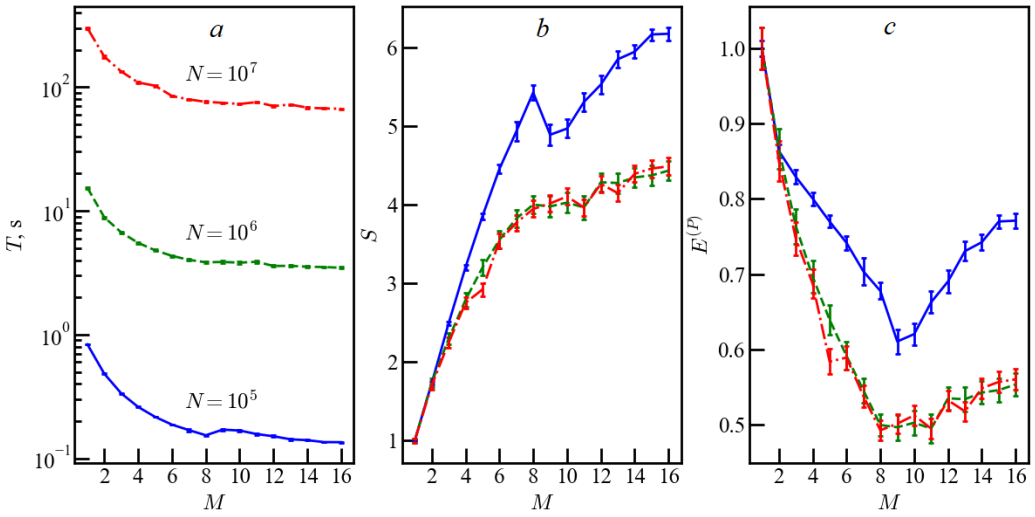

Fig. 7. The same as in figure 1 for computer M105S

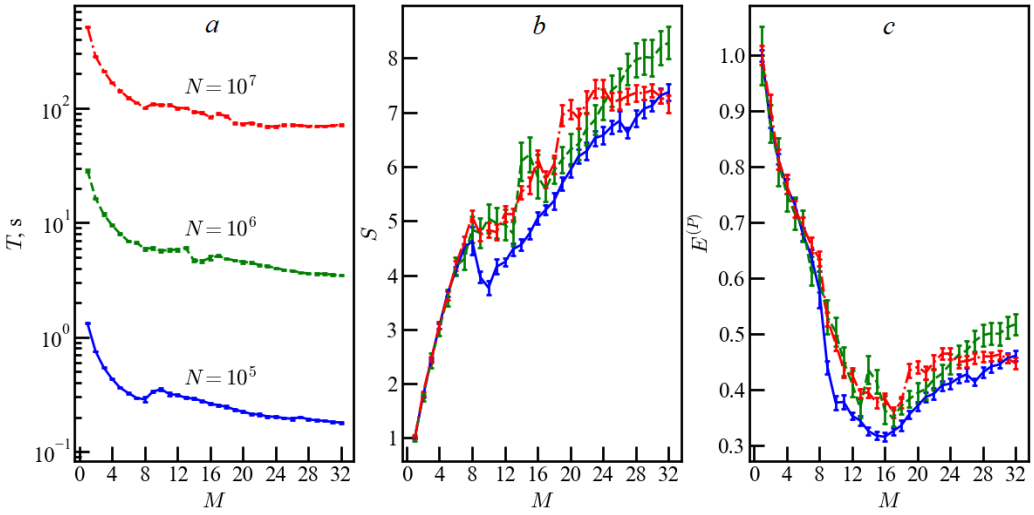

Fig. 8. The same as in figure 1 for computer M301

Figures 6–8 show the result of the efficiency calculation for systems with extra threads, as a continuation of the analysis for Epsilon and K80. The transition of the number of threads through $M = M_P$ remains critically important. Comparison of $E^{(P)}(8)$ and $E^{(P)}(16)$ for computer M105S gives a gain about of 12 percent when using the maximum number of extra threads. The result for computer M301is shown in figure 8, which demonstrates a slightly different version of resource distribution among threads. We have a characteristic kink in the dependencies near $M = M_P/2 = 8$. This means that extra threads begin to be used, although there are still free physical cores. At the same time, using the maximum possible number of extra threads for the M301 processor also improves the efficiency of parallel code about by 40 percent in models with ($N \sim 10^5$–10^6).

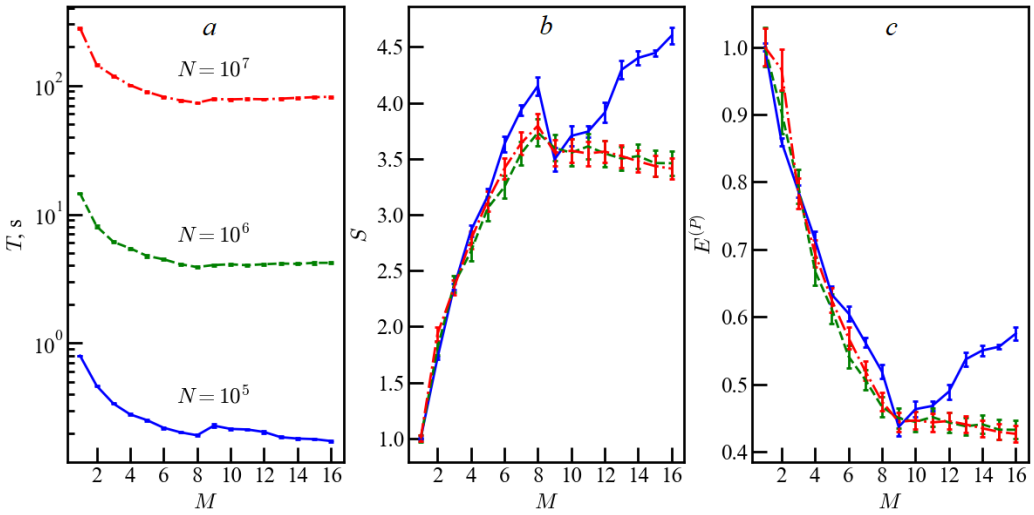

Fig. 9. The same as in figure 1 for computer Ryzen

Computer Ryzen is based on AMD processor. The result of calculating $T(M)$, $S(M)$, $E^{(P)}(M)$ differs from the CPUs considered above (figure 9). Only models with a small number of particles reproduce similar dependencies. Extra threads for models with $N \geq 10^6$

do not provide a performance boost, worsening the result for any number of M_E. Whether this conclusion is general for AMD processors or applies to a specific CPU requires further study.

Figure 10 shows the dependence of the program execution time on the processor release year from our sample. These results illustrate Moore's law. We have an approximately exponential decrease of the form $T \propto 2^{-t/\tau_y}$ with $\tau_y = 5.6$ years for the model with $N = 10^6$ and $M = 1$ (see the black dashed line in figure 10). There is a noticeable spread of the time scale τ_y within 5–6 years for different models. The general trend of the data in figure 10 fits into the historical development of computing technology.

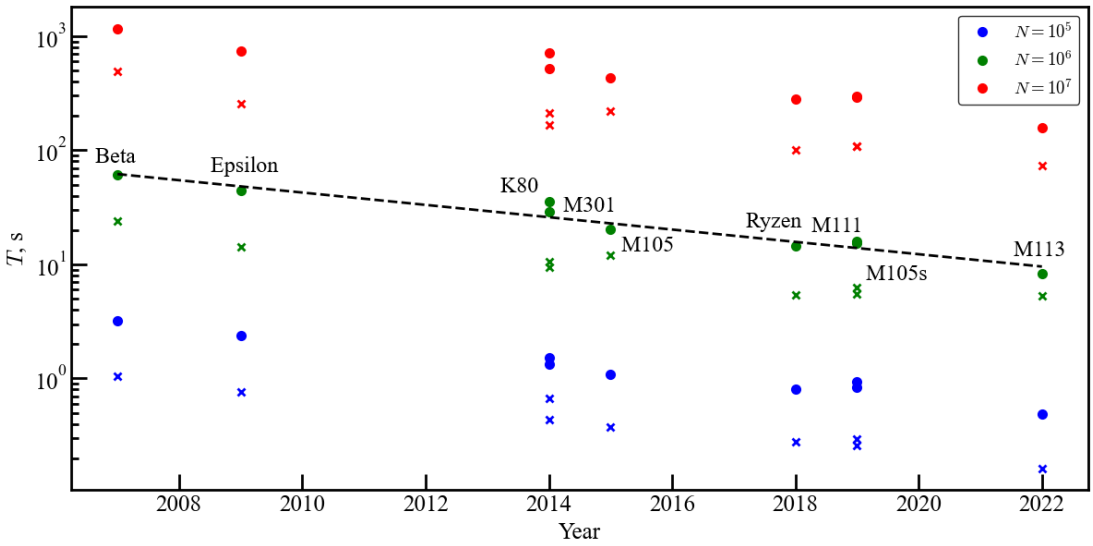

Fig. 10. The execution time of one integration step on one core ($M = 1$, circles) and four cores ($M = 4$, crosses) from the processor release date for various computers from Table 1 in models with $N = 10^5$ (blue), $N = 10^6$ (green) and $N = 10^7$ (red). The black dashed line represents the approximation $T \propto 2^{-t/\tau_y}$ with $\tau_y = 5.6$ years

The results discussed above are based on three main models with the number of particles $N = 10^5; 10^6; 10^7$. Figure 11 shows in more detail the dependencies of the efficiency E on the number of particles in the range $2^{17} \leq N \leq 2^{24}$. We restrict ourselves to the number of physical cores $M = M_P = 4$ and $M_E = 0$ in order to conveniently compare and cover all processors from Table 1. Models with a small number of particles demonstrate a small spread of efficiency within $0.67 - 0.81$. Simulations with $N = 2^{24} \simeq 1.7 \cdot 10^7$ give a spread of $E = 0.45 - 0.85$. An increase in N can lead to both a decrease and an increase in efficiency, which is manifested in the non-monotonic behavior of the dependencies $E^{(P)}(N)$ in some models. Such a result was noted, for example, in simulations on GPU with CUDA for a similar model [16].

This effect can be caused by uneven distribution of data across RAM modules and dual-channel access to it. An important factor is also the ratio between the processor data processing speed (P) and RAM bandwidth (R), P/R. Let's consider two scenarios, in each of which all data are physically located in one RAM module with small N. The first case corresponds to the condition $P/R > 1$. Then the computation efficiency will grow with increasing N until it reaches a certain value N_{crit}, after which it will begin to decrease. Further growth of N requires placing data in two memory modules, which allows using

the second access channel to it and leads to some increase in efficiency with a further plateau. The opposite inequality $P/R < 1$ determines the second option, when the condition $N > N_{crit}$ is satisfied immediately. The growth of N is accompanied by some decrease in efficiency until the data is physically placed in two different memory modules. This allows the second channel to be used and the efficiency to increase slightly, reaching a plateau, as in the first case.

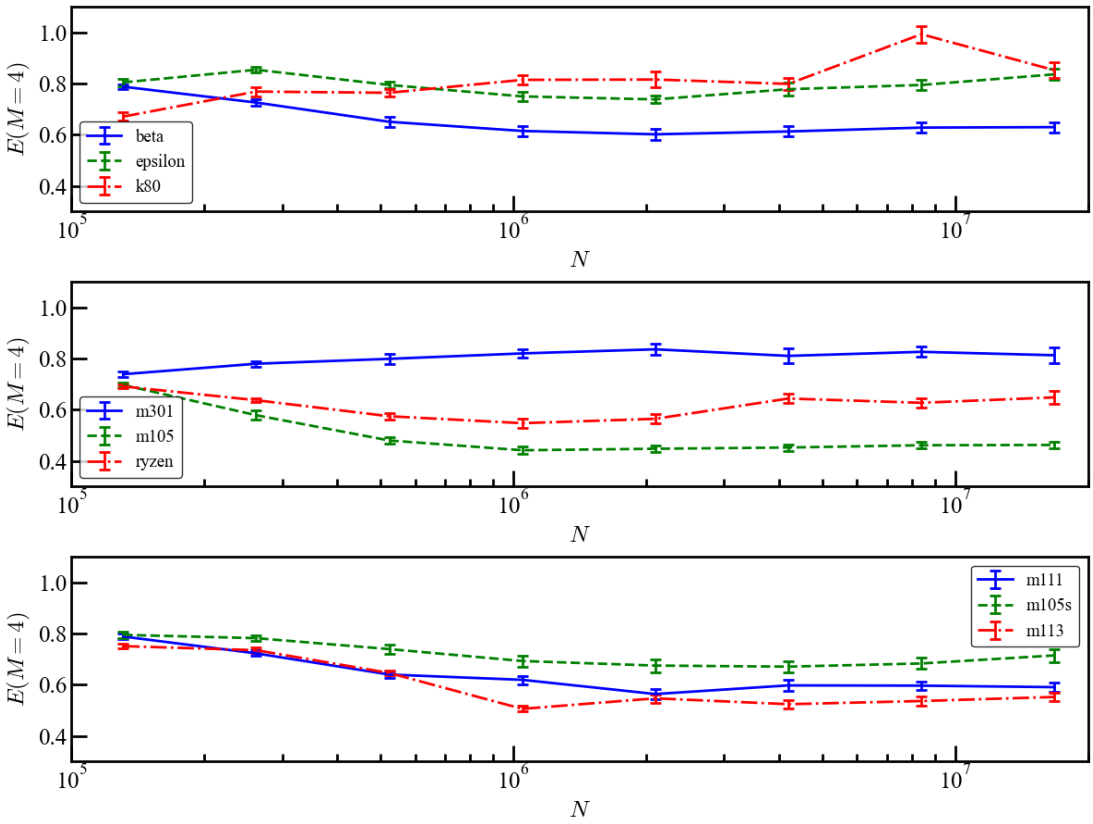

Fig. 11. Dependence of efficiency on the number of particles for four computational threads for different processors

Conclusion and discussion

The analysis of the efficiency of parallel implementation of the code for calculating the gravitational force was performed for a system of N particles using the Treecode method. We limited ourselves to considering the Open Multi-Processing (OpenMP) standard, since the use of the Message Passing Interface (MPI) for calculations on several processors significantly depends on the characteristics of the communication network.

Speedup and efficiency of parallel simulations depend on a large number of subtle factors that determine the architecture of multi-core processors. We focus on the role of additional logical threads (extra threads) that change the efficiency of executing parallel codes. Our results show that increasing the total number of threads due to extra threads (M_E) improves the efficiency of parallel Treecode calculations to an average 20 percent for different processors only in the case of $M_E = M_P$. If the number of extra threads is less than the number of physical cores, then the use of logical threads can significantly worsen

the efficiency of parallelization.

A significant negative factor is the special reduction of the processor frequency in multi-core processors. The presence of a base frequency $\nu^{(1)}$ and a frequency in the Boost mode $\nu^{(2)}$ leads to calculations with a changing processor frequency. The transition from $\nu^{(1)}$ to $\nu^{(2)}$ can occur in several steps, since the processor frequency is adjusted to the current load. If the computer has several processors with shared memory, this can unbalance the efficiency of parallelization.

Note the emergence of processors with a new architecture, which have a more extensive set of cores. An example is processor Intel Core i9 14900K, which allows the use of 32 threads. The cores are divided into 8 performance cores (P-core) and 16 efficient cores (E-core). Performance cores support 8 extra threads, which gives a total number of threads $M = 32$. The question of the efficiency of such processors with a more complex architecture requires a separate analysis.

NOTES

[1] This work supported by the Russian Science Foundation (grant no. 23-71-00016, https://rscf.ru/project/23-71-00016/).

[2] https://home.ifa.hawaii.edu/users/barnes/treecode_old/index.html

REFERENCES

1. Athanassoula E. *N*-body Simulations of Galaxies and Groups of Galaxies with the Marseille GRAPE Systems. *Annals of the New York Academy of Sciences*, 1998, vol. 867, no. 1, pp. 141-155. DOI: https://doi.org/10.1111/j.1749-6632.1998.tb11255.x

2. Bagla J.S. Cosmological *N*-body Simulation: Techniques, Scope and Status. *Current Science*, 2005, vol. 88, pp. 1088-1100. DOI: https://doi.org/10.48550/arXiv.astro-ph/0411043

3. Barnes J., Hut P. A Hierarchical O(N log N) Force-Calculation Algorithm. *Nature*, 1986, vol. 324, no. 4, pp. 446-449. DOI: https://doi.org/10.1038/324446a0

4. Bissekenov A., Kalambay M., Abdikamalov E., Pang X., Berczik P., Shukirgaliyev B. Cluster Membership Analysis with Supervised Learning and N-Body Simulations. *Astronomy & Astrophysics*, 2024, vol. 689, article ID: A282. DOI: https://doi.org/10.1051/0004-6361/202449791

5. Butenko M.A., Belikova I.V., Kuzmin N.M., Khokhlova S.S., Ivanchenko G.S., Ten A.V., Kudina I.G. Numerical Simulation of the Galaxies Outer Spiral Structure: the Influence of the Dark Halo Non-Axisymmetry on the Gaseous Disk Shape. *Mathematical Physics and Computer Simulation*, 2022, vol. 25 (3), pp. 73-83. DOI: https://doi.org/10.15688/mpcm.jvolsu.2022.3.5

6. Ciambur B.C., Fragkoudi F., Khoperskov S., Di Matteo P., Combes F. Double X/Peanut Structures in Barred Galaxies – Insights from an *N*-Body Simulation. *Monthly Notices of the Royal Astronomical Society*, 2020, vol. 503, no. 2, pp. 2203-2214. DOI: https://doi.org/10.1093/mnras/staa3814

7. Dubinski J., Kim J., Park C., Humble R. GOTPM: A Parallel Hybrid Particle-Mesh Treecode. *New Astronomy*, 2004, vol. 9, pp. 111-126. DOI: https://doi.org/10.1016/j.newast.2003.08.002

8. Fattahi A., Navarro J.F., Sawala T., Frenk C.S., Oman K.A., Crain R.A., Furlong M., Schaller M., Schaye J., Theuns T., Jenkins A. The Apostle Project: Local Group Kinematic Mass Constraints and Simulation Candidate Selection. *Monthly Notices of the Royal Astronomical Society*, 2016, vol. 457, pp. 844-856. DOI: https://doi.org/10.1093/mnras/stv2970

9. Fridman A.M., Khoperskov A.V. *Physics of Galactic Disks*. Cambridge International Science Publishing Ltd, 2012. 754 p.

10. Grand R.J.J., Springel V., Gomez F.A., Marinacci F., Pakmor R., Campbell D.J.R., Jenkins A. Vertical Disc Heating in Milky Way-Sized Galaxies in a Cosmological Context. *Monthly Notices of the Royal Astronomical Society*, 2016, vol. 459, pp. 199-219.

11. Hernandez-Aguayo C., Springel V., Pakmor R., Barrera M., Ferlito F., White S.D.M., Hernquis L., Hadzhiyska B., Delgado A.M., Kannan R., Bose S., Frenk Carlos The MillenniumTNG Project: High-Precision Predictions for Matter Clustering and Halo Statistics. *Monthly Notices of the Royal Astronomical Society*, 2023, vol. 524, pp. 2556-2578. DOI: https://doi.org/10.1093/mnras/stad1657

12. Ishchenko M., Berczik P., Panamarev T., Kuvatova D., Kalambay M., Gluchshenko A., Veles O., Sobolenko M., Sobodar O., Omarov C. Dynamical Evolution of Milky Way Globular Clusters on the Cosmological Timescale - I. Mass Loss and Interaction with the Nuclear Star Cluster. *Astronomy & Astrophysics*, 2024, vol. 689, article ID: A178. DOI: https://doi.org/10.1051/0004-6361/202450399

13. Ishchenko M., Berczik P., Sobolenko M. Milky Way Globular Clusters on Cosmological Timescales. IV. Guests in the Outer Solar System. *Astronomy & Astrophysics*, 2024, vol. 683, article ID: A146. DOI: https://doi.org/10.1051/0004-6361/202347990

14. Just A., Piskunov A.E., Klos J.H., Kovaleva D.A., Polyachenko E.V. Global Survey of Star Clusters in the Milky Way – VII. Tidal Parameters and Mass Function. *Astronomy & Astrophysics*, 2023, vol. 672, article ID: A187. DOI: https://doi.org/10.1051/0004-6361/202244723

15. Khoperskov A.V., Khrapov S.S., Sirotin D.S. Formation of Transitional cE/UCD Galaxies Through Massive Disc to Dwarf Galaxy Mergers. *Galaxies*, 2024, vol. 12, no. 1, article ID: 1. DOI: https://doi.org/10.3390/galaxies12010001

16. Khrapov S., Khoperskov A. Study of the Effectiveness of Parallel Algorithms for Modeling the Dynamics of Collisionless Galactic Systems on GPUs. *Supercomputing Frontiers and Innovations*, 2024, vol. 11, no. 3, pp. 27-44. DOI: https://doi.org/10.14529/jsfi240302

17. Khrapov S.S., Khoperskov A.V., Zaitseva N.A., Zasov A.V., Titov A.V. Formation of Spiral Dwarf Galaxies: Observational Data and Results of Numerical Simulation. *Saint Petersburg State Polytechnical University Journal. Physics and Mathematics*, 2023, vol. 16, no. 1.2, pp. 395-402. DOI: https://doi.org/10.18721/JPM.161.260

18. Khrapov S.S., Khoperskov S.A., Khoperskov A.V. New features of parallel implementation of N-body problems on GPU. *Bulletin of the South Ural State University, Series: Mathematical Modelling, Programming and Computer Software*. 2018, vol. 11, no. 1, pp. 124-136. DOI: https://doi.org/10.14529/mmp180111

19. Kulikov I., Chernykh I., Protasov V. Mathematical Modeling of Formation, Evolution and Interaction of Galaxies in Cosmological Context. *Journal of Physics: Conference Series*, 2016, vol. 722, no. 1, article ID: 012023. DOI: https://dx.doi.org/10.1088/1742-6596/722/1/012023

20. Kyziropoulos P.E., Filelis-Papadopoulos C.K., Gravvanis G.A. Parallel N-body Simulation Based on the PM and P3M Methods Using Multigrid Schemes in Conjunction with Generic Approximate Sparse Inverses. *Mathematical Problems in Engineering*, 2015, vol. 2015, article ID: 450980. DOI: https://doi.org/10.1155/2015/450980

21. L'Huillier H.B., Park C., Kim J. GalaxyFlow: Upsampling Hydrodynamical Simulations for Realistic Mock Stellar Catalogues. *New Astronomy*, 2014, vol. 30, pp. 79-88. DOI: https://doi.org/10.1016/j.newast.2014.01.007

22. Lim S.H., Raman K.A., Buckley M.R., Shih D. GalaxyFlow: Upsampling Hydrodynamical Simulations for Realistic Mock Stellar Catalogues. *Monthly Notices of the Royal Astronomical Society*, 2024, vol. 533, no. 1, pp. 143-164. DOI: https://doi.org/10.1093/mnras/stae1672

23. Nipoti C., Cherchi G., Iorio G., Calura F. Effective N-Body Models of Composite Collisionless Stellar Systems. *Monthly Notices of the Royal Astronomical Society*, 2021, vol. 503, no. 3, pp. 4221-4230. DOI: https://doi.org/10.1093/mnras/stab763

24. Pejch M.A., Morozov A.G., Khoperskov A.V. Modeling a Double-Hump Gas Rotation Curves in the Axisymmetric Gravitational Field of Galaxies. *Mathematical Physics and Computer Simulation*, 2023, vol. 3, pp. 91-104. DOI: https://doi.org/10.15688/mpcm.jvolsu.2023.3.7

25. Pillepich A., Springel V., Nelson D., Genel S., Naiman J., Pakmor R., Hernquist L., Torrey P., Vogelsberger M., Weinberger R., Marinacci F. Simulating Galaxy Formation with the IllustrisTNG Model. *Monthly Notices of the Royal Astronomical Society*, 2018, vol. 473, pp. 4077-4106. DOI: https://doi.org/10.1093/mnras/stx2656

26. Potter D., Stadel J., Teyssier R. PKDGRAV3: Beyond Trillion Particle Cosmological Simulations for the Next Era of Galaxy Surveys. *Computational Astrophysics and Cosmology*, 2017, vol. 4, article ID: 2. DOI: https://doi.org/10.1186/s40668-017-0021-1

27. Ruan Ch.-Z., Hernandez-Aguayo C., Li B., Christian A., Carlton M.B., Klypin A., Prada F. Fast Full N-Body Simulations of Generic Modified Gravity: Conformal Coupling Models. *Journal of Cosmology and Astroparticle Physics*, 2022, vol. 2022, no. 5, article ID: 018. DOI: https://dx.doi.org/10.1088/1475-7516/2022/05/018

28. Schaye J., Crain R.A., Bower R.G., Furlong F., Schaller M., Theuns T., Vecchia C.D., Frenk C.S., McCarthy I.G., Helly J.C., Jenkins A., Rosas-Guevara Y.M., White S.D.M., Baes M., Booth C.M., Camps P., Navarro J.F., Qu Y., Rahmati A., Sawala T., Thomas P.A., Trayford J. The EAGLE Project: Simulating the Evolution and Assembly of Galaxies and Their Environments. *Monthly Notices of the Royal Astronomical Society*, 2014, vol. 446, pp. 521-554. DOI: https://doi.org/10.1093/mnras/stu2058

29. Schaye J., Kugel R., Schaller M., Helly J.C., Braspenning J. The FLAMINGO Project: Cosmological Hydrodynamical Simulations for Large-Scale Structure and Galaxy Cluster Surveys. *Monthly Notices of the Royal Astronomical Society*, 2023, vol. 526, pp. 4978-5020. DOI: https://doi.org/10.1093/mnras/stad2419

30. Smirnov A.A., Sotnikova N.Y., Koshkin A.A. Simulations of Slow Bars in Anisotropic Disk Systems. *Astronomy Letters*, 2017, vol. 43, pp. 61-74.

31. Springel V., Yoshida N., White S.D.M. GADGET: A Code for Collisionless and Gasdynamical Cosmological Simulations. *New Astronomy*, 2001, vol. 6, pp. 79-117. DOI: https://doi.org/10.1016/S1384-1076(01)00042-2

32. Tikhonenko I.S., Smirnov A.A., Sotnikova N.Ya. First Direct Identification of the Barlens Vertical Structure in Galaxy Models. *Astronomy & Astrophysics*, 2021, vol. 648, article ID: L4. DOI: https://doi.org/10.1051/0004-6361/202140703

33. Titov A.V., Khoperskov A.V. Numerical Modeling of the Collisions of Spheroidal Galaxies: Mass Loss Efficiency by Baryon Components. *Vestnik St. Petersburg University, Mathematics*, 2022, vol. 55, no. 1, pp. 124-134. DOI: https://doi.org/10.1134/S1063454122010149

34. Vogelsberger M., Marinacci F., Torrey P., Puchwein E. Cosmological Simulations of Galaxy Formation. *Nature Reviews Physics*, 2020, vol. 2, pp. 42-66. DOI: https://doi.org/10.1038/s42254-019-0127-2

35. Walther J.H. An Influence Matrix Particle–Particle Particle-Mesh Algorithm with Exact Particle–Particle Correction. *Journal of Computational Physics*, 2003, vol. 184, pp. 670-678.

36. Yokota R., Barba L.A. Treecode and Fast Multipole Method for N-Body Simulation with CUDA. *GPU Computing Gems Emerald Edition*. Boston, Morgan Kaufmann, 2011, pp. 113-132. DOI: https://doi.org/10.1016/B978-0-12-384988-5.00009-7

ЭФФЕКТИВНОСТЬ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ГРАВИТАЦИОННЫХ СИЛ МЕТОДОМ TREECODE В МОДЕЛЯХ N-ТЕЛ

Николай Михайлович Кузьмин

Кандидат физико-математических наук, доцент кафедры
информационных систем и компьютерного моделирования,
Волгоградский государственный университет
nikolay.kuzmin@volsu.ru
https://orcid.org/0000-0003-4074-0970
просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация

Данила Сергеевич Сиротин

Ассистент кафедры информационных систем и компьютерного моделирования,
Волгоградский государственный университет
d.sirotin@volsu.ru
https://orcid.org/0000-0001-8956-570X
просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация

Александр Валентинович Хоперсков

Доктор физико-математических наук, профессор кафедры
информационных систем и компьютерного моделирования,
Волгоградский государственный университет
khoperskov@volsu.ru
https://orcid.org/0000-0003-0149-7947
просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация

Аннотация. Моделирование бесстолкновительных галактических систем основывается на модели N-тел, которая требует больших вычислительных ресурсов из-за дальнодействующего характера гравитационных сил. Наиболее распространенным методом вычисления гравитации является приближенный алгоритм TreeCode, обеспечивающий более быстрое вычисление силы по сравнению с прямым суммированием вкладов от всех частиц. Проведен анализ вычислительной эффективности для моделей с числом частиц в пределах 10^8. Мы рассмотрели несколько процессоров с различной архитектурой с целью определения производительности параллельных симуляций на основе стандарта OpenMP. Анализ использования дополнительных потоков (extra threads) дополнительно к физическим ядрам показывает рост производительности симуляций только при загрузке всех логических потоков. Это дает прирост параллельной эффективности (efficiency of parallel computing) в среднем на 20 %.

Ключевые слова: параллельные вычисления, гравитирующие системы, OpenMP, архитектура процессоров, технология Hyper-Threading.