Parallel computing applied to the molecular dynamics simulations

Victor L. Malyshev\textsuperscript{1}, Elena F. Moiseeva\textsuperscript{1}

\textsuperscript{1} Ufa State Petroleum Technological University, Ufa 450062, Russian Federation
E-mail: victor.l.malyshev@gmail.com

Abstract.
This paper discusses the use of high-performance algorithms for modeling the dynamics of dispersed systems using the method of molecular dynamics. Large-scale modeling, which makes it possible to determine various thermodynamic parameters and control the processes of phase transformations, requires consideration of a large number of particles, which imposes significant restrictions on the computational capabilities of the system. In work various problems of dynamics of disperse systems are considered. To solve the problems posed by the authors, a specialized data structure was developed, which reduces the computational complexity of the algorithm from quadratic to linear. The specificity of the molecular dynamics method makes it possible to achieve significant acceleration when using computations at heterogeneous stations equipped with a central processing unit (CPU) and graphic processing units (GPU). The authors showed that the presented technique can be effectively used for solving problems related to modeling processes occurring in areas with characteristic sizes of tens of nanometers on personal computers equipped with one or more GPUs.

1. Introduction
Molecular dynamics (MD) simulations is one of the best methods for solving the micro- and nanoscale problems. Despite the fact that the MD simulations are successfully applied for problems of nano-scale dynamics, the real systems investigation using MD seems very difficult because requires the consideration of enormous number of particles, and, thus, a lot of computational operations per minute. To speed up the calculations modern numerical algorithms and high performance programming can be used. The most used and simple in realization is the system consisting of monoatomic molecules which interact via Lennard-Jones (LJ) potential. As LJ potential function is short-range (for each atom only the influence of its nearest environment is significant), there is no need to calculate the interaction forces acting between each pair of molecules. The use of neighbour lists or data structures allows to simplify this procedure.

The first attempts in this direction were described by M.P. Allen and D.J. Tildesley [1], who built a list of neighbours. The main idea of this method is based on the fact that LJ potential has small interaction radius. Later hierarchical data structures were developed. Such structures are difficult to implement, but have the advantage of applying to long-range potentials. One of the algorithms for building such a structure can be found in [2, 3]. The implementation of MD simulations on systems with parallel computing capabilities can be attributed to high-performance hardware.
To date, there are a number of software packages for MD modeling, such as LAMMPS [4], DL_POLY [5], NAMD [6], GROMACS [7] and ESPResSo [8], which effectively implement the molecular dynamics method on clusters, however, only a few of them include the ability to implement computations on graphics processors.

Many authors have paid attention to the MD simulations on GPU [9, 10, 11, 12], but for the first time this method was fully implemented on the GPU in 2008 by Andersen et al [13]. In their work, they presented the first implementation of the MD simulations of monatomic molecules, in which all steps of the algorithm were performed on the GPU using NVIDIA CUDA technology. To reduce the complexity of the algorithm, they used neighbor lists. A similar implementation of the algorithm was presented by J.A. van Meel [14]. In 2011, Trott et al [15] presented the implementation of the molecular dynamics algorithm on a cluster of GPUs using the LAMMPS package. To speed up the calculations, the neighbor lists were also used in the algorithm. Calculations were carried out for different classes of materials (biomolecules, polymers, metals, semiconductors), and the speed of calculations was compared with existing implementations of MD-modeling on the GPU.

This paper describes the process of building a new data structure for non-polar molecules. The use of such a data structure, in conjunction with graphics processors, allows us to significantly speed up the calculation of short-range potentials in the MD method.

2. Model
The MD simulations is based on the idea that the considered system consists of molecules following the trajectories calculated from the Newtons second law and interacting with each other via the interaction potential. In a classical MD simulations, atomic positions evolve from their initial state according to Newtons equation of motion.

Except for the simplest cases, such equations are solved numerically, according to the chosen algorithm. For the integration of Newton’s equations it is necessary to calculate the force, acting on atom, or the complex potential for the interaction of atom with the others.

The current paper represents the simulations of simple monoatomic fluids, which molecules interact via Lennard-Jones potential:

\[ u_{LJ}(r) = 4\varepsilon \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6, \]  

where \( r_{ij} = |r_i - r_j| \) is the distance between molecules \( i \) and \( j \), \( \varepsilon \) is the depth of the potential energy well and \( \sigma \) is the distance at which the potential is zero.

The application of the MD simulations to study the systems with the size of tens and hundreds nanometers requires the consideration of millions molecules. For example, the cubic domain of \( 80 \times 80 \times 80 \) nm\(^3\) contains over 10 millions molecules. For most existing computational systems, the computations of the interactions between such a huge number of molecules is not a trivial problem. That’s why for such problems it is preferable to use modern computational methods and algorithms. In our simulations we use both algorithmic and hardware accelerations. As the MD simulations require fast and convenient data access, the data structures and efficient algorithms for their generation are of high priority.

3. Data structure
Efficient use of parallel algorithms requires convenient and fast data access. For this purpose, an efficient data structure was developed. We describe the main points of its construction. The algorithm consists of the following steps:

Preliminary calculations. Grid construction. Determination of the main parameters: the number of boxes, their linear dimensions, the order of numbering, the position of the center
of each box. Building a list of neighbors for each box and centered vector of neighbors. These calculations are performed before the cycle. Such parts of the algorithm are sufficient to implement on the CPU, since their calculations are performed only once when generating the initial data.

**Data structure generation.** Since the modeling region is divided into boxes, the best way is to arrange the coordinates in such a sequence, when all the particles in the zero box are recorded first, then in the 1st, and so on until the last box \( N_{\text{AllBox}} - 1 \). To build a data structure, bucket sorting of atoms is used [16].

**Converting positions to local coordinates.** This operation is necessary for more convenient calculation of the distances between the particles - it allows you to avoid unnecessary branching of the algorithm, which has a bad effect on the performance of the GPU. Below is a detailed description of each of the presented items. All formulas are written for the three-dimensional case. For simplicity and clarity of the presentation of the material, we present figures for a two-dimensional data structure.

### 3.1. Number of boxes. Enumeration

Let us introduce two types of box indexing: pass-through and Cartesian. The pass-through index is a consecutive numbering of all boxes. Those end-to-end index traverses the curve filling the area (space filling curve). Cartesian method enumerates the box according to the shift along each of the axes relative to the origin (the lower left corner is selected by the reference point). The transition from one numbering system to another is carried out using simple formulas that use only the box numbering and the number of boxes in each direction.

### 3.2. Neighbour list construction

According to the selected algorithm, each atom located in the selected box can interact only with atoms located in the same box, as well as with atoms located in neighboring boxes. To find cells adjacent to the selected one, Cartesian numbering is used. Thus, having the index of the selected box in Cartesian numbering \([i, j, k]\), you can find 27 of its neighbors (including the selected cell itself): \([i - 1, j - 1, k - 1]\), \([i, j - 1, k - 1]\), ..., \([i + 1, j + 1, k + 1]\). Then, for convenience, Cartesian indices can be transferred to the pass-through numbering system. The difficulty is in finding the neighbors for the boxes located on the edge of the simulated area. They are calculated in accordance with the selected boundary conditions. As a result, each box can be assigned a list of neighbors, consisting of 27 indices. The merged neighbor list will thus contain \(27N_{\text{AllBox}}\) indices.

### 3.3. Data structure construction

To implement convenient data access, it is necessary to re-sort the particles in the system. Since the use of conventional sorting algorithms is not efficient on the GPU, a pseudo-sorting algorithm (bucket sorting) is used, as in the algorithm for generating a hierarchical data structure. It consists of the following steps:

(i) Constructing a placement histogram. It shows how many particles are in each box and the local index for each particle in the given box.

(ii) Parallel scanning to find ”markers” that show the indices of the first and last atom in the specified box in the new sorted array of particles and the determination of the global particle index.

(iii) Ordering particles according to their global index.
3.4. Transformation of positions to local coordinates
When solving the Newton’s equation, in order to calculate the interaction forces between atoms, it is necessary to know the distances between them. In order to unify the process of calculating the distances between particles and calculating the interaction forces under periodic boundary conditions, we transform the position of each particle into a local vector.

This reduces the computational complexity of calculating the interaction by reducing the excessive branching of the algorithm, which is detrimental to the effective implementation of the algorithm on the GPU.

3.5. Hardware acceleration using graphics processors
One of the most effective ways to speed up the calculation is to use graphical processes (GPU) for numerical calculations.

3.6. Application of GPU and data structures for molecular dynamics modeling
Let us consider the main features of the use of graphics processors for molecular dynamics simulations.

The algorithm for constructing the data structure described above assumes the existence of operations performed prior to the main time loop. Such calculations can be implemented once at the launch of the program code. Such operations include, for example, the neighbor list construction.

The initial data structure generation occurs on the main node. Next, data is split into nodes, and each node performs the same sequence of steps for different data sets corresponding to each node. From the flowchart it can be seen that at each time step, at each of the nodes, a data structure is checked, which reveals the number of particles that “flown” between the boxes. If necessary, the data structure is rebuilt, and the calculation continues. Upon completion of the execution of all steps of the algorithm, a transition to a new time step occurs.

3.7. Acceleration results
To demonstrate the acceleration obtained using a data structure and a GPU, a cubic region filled with liquid argon with a density of \( \rho = 950 \text{ kg/m}^3 \) at a temperature of 85 K was considered. Particles are distributed throughout the simulation region. Potential cutoff radius is \( r_{\text{cutoff}} = 3.5\sigma \), where \( \sigma \) is equal 3.405 Å. Calculations are performed on a heterogeneous workstation with two 6-core Intel Xeon 5660 2.8 GHz CPUs (12 physical cores and 12 virtual cores using Hyper-Threading technology), 12 GB RAM, and four NVIDIA Tesla C2050 GPUs/C2075 with 6 Gb RAM. The GPU code is written using NVIDIA CUDA technology; parallel operation of several GPUs implemented using OpenMP technology. Data is copied between GPUs using CUDA UVA (unified virtual addressing) technology, which allows data to be copied between GPUs directly, by passing the CPU. In the course of computational experiments, it was found that the use of NVIDIA CUDA UVA technology speeds up data exchange by about 30%. Calculations are made for double-precision floating point numbers.

Figure 1 presents the calculation time for one time step of the algorithm depending on the number of atoms in the system. The graph shows the calculation times by direct summation on the CPU and GPU (brute force). From the graph it can be seen that the use of the developed data structure makes it possible to reduce the computational complexity from quadratic to linear. Also on the graph you can see the performance comparison with the LAMMPS package (application package for molecular dynamics simulation).

The developed high-performance programming code for the MD simulations has been applied to several physical problems, such as cavitation in liquid argon (fig. 2) [17], surface bubble dynamics in a liquid flow (fig. 3) [18], particle movement by surface nanobubble (fig. 4) [19].
Figure 1. Full time calculation of one time step of the algorithm using a data structure. The dotted line shows the task time for a similar task obtained using the LAMMPS simulator. Graphs of calculation time are applied by direct summation on the CPU and GPU.

Figure 2. Cavitation in liquid argon near solid particle

Figure 3. The initial molecules distribution (slice of the domain). Blue circles — Ar, yellow triangles — Ne, black squares — substrate. Number of particles is 1477852. Size of the domain is 40nm × 40nm × 40nm.
4. Conclusion

The combination of the proposed data structure and graphics processors (GPU) for solving problems of the dynamics of dispersed systems using the molecular dynamics simulations reduces the computational complexity of the algorithm from $O(N^2)$ to $O(N)$.

The proposed approach allows to calculate one step of the algorithm for a system consisting of 20 million particles in less than 2 seconds. Therefore, the use of such algorithms is necessary for solving nano- and micro-scale problems such as phase transitions, cavitation, bubble dynamics, which require the large-scale modelling.

References

[1] Allen M and Tildesley D 1992 Computer Simulation of Liquids (Clarendon Press)
[2] Gumerov N A and Duraiswami R 2008 Journal of Computational Physics 227 8290–8313
[3] Marin D, Malyshev V, Moiseeva E, Gumerov N, Akhatov I and Mikhailenko K 2013 Numerical methods and programming 14 483–495
[4] Plimpton S 1995 Journal of Computational Physics 117 1 – 19 ISSN 0021-9991
[5] Forester T and Smith W DL POLY 2.0: A general-purpose parallel molecular dynamics simulation.
[6] Phillips J C, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, Chipot C, Skeel R D, Kal L and Schulten K 2005 Journal of Computational Chemistry 26 1781–1802 ISSN 1096-987X
[7] Berendsen H J C, Spoel D V D and Drunen R V 1995 Comp. Phys. Comm 91 43–56
[8] Limbach H J, Arnold A, Mann B and Holm C 2006 Computer Physics Communications 174 704–727
[9] Yang J, Wang Y and Chen Y 2007 Journal of Computational Physics 221 799 – 804 ISSN 0021-9991
[10] Che S, Boyer M, Meng J, Tarjan D, Sheaffer J W and Skadron K 2008 Journal of Parallel and Distributed Computing 68 1370 – 1380 ISSN 0743-7315
[11] Liu W, Schmidt B, Voss G and Miller-Wittig W 2008 Computer Physics Communications 179 634 – 641 ISSN 0010-4655
[12] Moiseeva E, Mikhaylenko C, Malyshev V, Maryin D and Gumerov N 2012 ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE) vol 7 pp 883–892
[13] Anderson J A, Lorenz C D and Travesset A 2008 Journal of Computational Physics 227 53425359
[14] van Meel J, Arnold A, Mann B and Holm C 2006 Computer Physics Communications 174 704–727
[15] Trott C R, Winterfeld L and Crozier P S 2011 http://arxiv.org/abs/1009.4330v2
[16] Cormen T 2009 Introduction to Algorithms, 3rd Edition: (MIT Press) ISBN 9780262033848
[17] Malyshev V, Marin D, Moiseeva E, Gumerov N and Akhatov I 2015 High Temperature 53 406–412
[18] Moiseeva E, Malyshev V, Marin D, Gumerov N and Akhatov I 2014 ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE) vol 7
[19] Moiseeva E, Malyshev V, Marin D, Gumerov N and Akhatov I 2017 Mathematical Models and Computer Simulations 29 131–140