Parallel Computing: Technology Trends
I. Foster et al. (Eds.)

doi:10.3233/APC200086

GPU Acceleration of Four-Site Water Models in LAMMPS

Vsevolod NIKOLSKIY [a,b,1], Vladimir STEGAILOV [a,b]
[a] *National Research University Higher School of Economics Moscow, Russia*
[b] *Joint Institute for High Temperatures of Russian Academy of Sciences Moscow, Russia*

Abstract. In this work, a new algorithm was developed for calculating the four-point water model TIP4P on graphics accelerators. It was designed as a part of the flexible molecular dynamics modeling package LAMMPS in the library module "GPU". In this paper we describe two approaches to implement the TIP4P model for GPU: 1) to divide the related computations between CPU and GPU; 2) to compute the interaction fully on the GPU. We verify the program, benchmark and profile it. The achieved speedup of interaction computation is about x7, acceleration of the entire calculation of about 55%.

Keywords. TIP4P, LAMMPS, atomistic modeling, accelerator, empirical potential

1. Introduction

Molecular dynamics is an extremely powerful tool in modern science. It is used in a wide variety of fields, including materials science, biology, theoretical physics, and many others. Engaged in the multiscale approach, molecular dynamics is necessary for parameterization of next-order models.

In the development of the method, two main directions can now be distinguished. First, the development of new physical models to expand the boundaries of the applicability of the method or to obtain more accurate results. Modern MD packages already include a huge number of implemented models and calculation methods, assembling which, as a constructor, and correctly configuring, one can make discoveries in the subject area.

Secondly, this is the development of the computational capabilities of the already described physical models. The fact is that molecular dynamics is an extremely resource-intensive method that requires tremendous computing power to build complex and large models. From the very first works, MD relies on the development of the computer industry.

Nowadays, the period of extensive development of supercomputer technologies has exhausted itself, and increasingly sophisticated technologies are applied to further in-

[1]Corresponding Author: Vsevolod Nikolskiy, International Laboratory for Supercomputer Atomistic Modelling and Multi-scale Analysis NRU HSE, 34 Tallinskaya Ulitsa, 123458, Moscow, Russia; E-mail: vnikolskiy@hse.ru

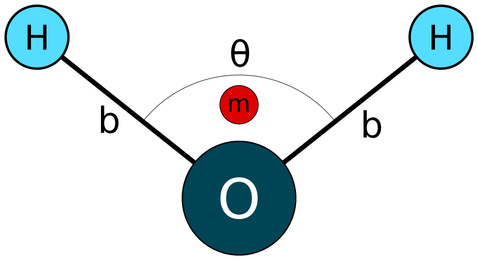

Figure 1. Rigid four-point water model TIP4P. H - hydrogen, O - oxygen, b - bonds and Theta is the angle. The particle "m" represents the virtual massless charge.

crease computing power, requiring efforts from software developers and users for the most efficient use. In addition, an increase in the universality of solutions and the possibility of code reuse is in demand, since otherwise the frequent change of the most relevant hardware inevitably leads to the need for routine support of an increasingly large code base of scientific packages.

In this work, we implement the well-known TIP4P water model for use on GPU accelerators as part of the popular LAMMPS package. The code can be compiled with CUDA and OpenCL backends due to the use of library. This approach allows us to increase the efficiency of use of computer resources because heterogeneous architectures are ubiquitous on modern supercomputers. On the other hand, our code does not duplicate, but effectively uses the huge number of features of the LAMMPS package for implementing molecular dynamics methods and their parallelization.

The further text will be organized as follows: in Section 2 we consider several related works, some of which we rely on during the development of this project. In Section 3, we describe two approaches that were created in the process of solving the problem posed in the project. Section 4 includes verification and performance testing of the developed code. The last Section 5 is the conclusion.

2. Related Work

It may seem that the most natural way to model water is to specify all or some of the three atoms that make up the water molecule as van der Waals and Coulomb interaction points [1]. In some cases, such a simple model is enough, but it is shown that the use of the fourth virtual massless charge point on the bisector of the H-O-H angle (Figure 1) significantly improves the electrostatic properties of the model. With the correct parameterization, such a model has wide applicability limits [2,3]. TIP4P water model in LAMMPS can be used as a basis of centroid molecular dynamics (CMD) quantum simulations to consider the effects of zero point energy and tunnelling [4].

There are GPU implementations of water models (TIP4P including) in GROMACS [5] and OpenMM [6], but these software tools are focused on biomolecular and soft matter models. The LAMMPS package has the greatest flexibility; it includes the largest number of potentials and possibilities for combining and extending methods. It includes the TIP4P water model for computing on the CPU in several versions — short ranged "cut" version and long-ranged with KSPACE computation [7] (Particle-Particle — Particle Mesh method). They are labeled as lj/cut/tip4p/cut and lj/cut/tip4p/long with

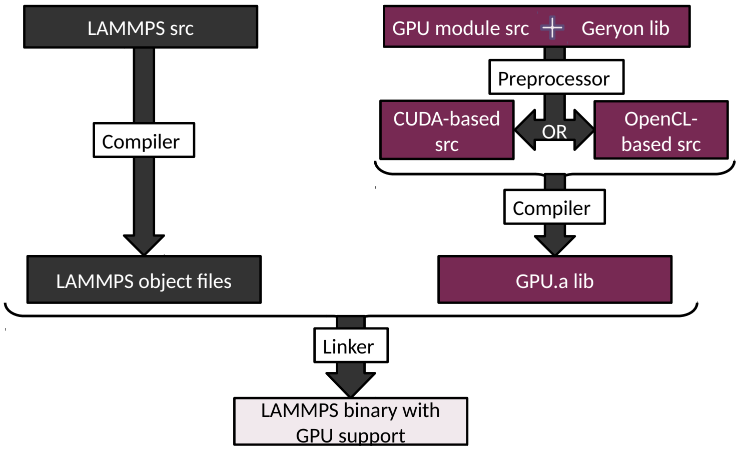

Figure 2. Geryon library used in LAMMPS allows one to compile the same code with both CUDA or OpenCL backends by preprocessing.

pppm/tip4p, respectively. The LAMMPS package implements an impressive set of potentials with GPU acceleration [8,9,10], but TIP4P is not among them. Nevertheless, we rely on the developments from the LAMMPS package on the implementation of accelerated potentials. Our project uses the Geryon library, which at the preprocessing level allows you to compile single code for CUDA and OpenCL backends (Figure 2), and the library includes the class hierarchy for molecular dynamics programming. Adapting the algorithm to use the accelerators raises some issues critical to performance [11,12], such as organizing data access [13]. Some of them are solved by the library.

We also rely on the KSPACE module for calculating long-ranged interaction [7]. This allows us to calculate only the short-range part of the Coulomb interaction, and to get the full value running the PPPM/TIP4P solver from the model script.

3. Implementation

To determine the coordinates of the virtual charge, it is necessary at each step to know the position of all three particles of the molecule. In this project, we relied on the source code of the potential lj/cut/tip4p/long to implement a “suffix-accurate-compatible” accelerated version. This code works by presume that in the input, the atoms that make up the water molecules are ordered, and each oxygen is followed by two corresponding hydrogen:

```
O - H - H - O - H - H - ... - O - H - H
```

Such ordering is stored in particle identifiers, but during the program’s operation this data is reordered in memory unpredictably, and another numbering is used when traversing the local arrays in LAMMPS. To find information about all the particles that make up the molecule, a separate code is intended. This code essentially uses the internal methods LAMMPS `Atom->map` and `Domain->closest_image` to correctly compose the molecular structure based on global particle identifiers. Porting these methods to the GPU is not entirely natural, so the first idea was to keep for execution on the CPU part of the code that accesses these functions.

Another feature of the lj/cut/tip4p/long is that it relies heavily on the use of Newtons third law, i.e. when calculating the interaction between **i** and **j** particles, the calculation result is saved for both participants in the interaction. It is unacceptable for GPU kernels, since an arbitrary thread cannot change data related to any arbitrary particle without expensive synchronization. It is necessary to organize the calculations so that only a certain group of threads is working on calculation of the force acting on the **i**-th particle. This requires a change in the algorithm and careful handling of some extreme cases.

3.1. Redundant Computation Approach

The information about the molecules is collected using the methods of the LAMMPS classes `Atom->map` and `Domain->closest_image`. It is possible to prepare molecular structure on the CPU and transfer it to the GPU. The problem is that information on virtual charges is needed not only for local, but also for **j** atoms. Therefore, the CPU has to make not so few calculations. Data on the molecular structure is stored as follows: atom numbers and a flag are stored in the hneingh array, extended to 4 number alignment if necessary. The coordinates of the virtual charge are calculated immediately and transferred to the m array, although this part of the algorithm can also be separated and transferred to the GPU.

```
hneigh[iO] = {iH1, iH2, 0, flag}
hneigh[iH1] = hneigh[iH2] = {iO, 0, 0, flag}
m[iO] = m[iH1] = m[iH2] = {x, y, z, flag}
```

When the data is prepared and transferred, the GPU kernel starts. Here we use the Redundant Computation Approach (RCA) [10]. It is expected that in our case this is not the fastest approach for naive implementation, but it is useful as a basis for further complication. Each molecule contains a single virtual charge **m**, but the force acting on it is distributed between all three real atoms [14]. Thus, in order to obtain the total sum of forces without writing data to the memory of the **j**-th atoms, it is possible to calculate two components of the electrostatic force for each ith atom: direct interaction and distributed (Figure 3). Thus, the effect on the virtual charge **m** of each **j**-th atom have to be calculated not once, but three times, but the purpose of this is that there will be no need for any additional synchronization.

The code for this approach was not as simple as originally expected. The correct calculation of interactions required introducing a rather large number of additional checks and conditions into the kernel. Overall performance is not worth the effort of porting to the GPU. But this code can be easily improved.

3.2. Reduced Redundant Computation Approach

It was decided to port all the calculations to the GPU and use additional memory order of $\mathcal{O}(n)$ to reduce the number of redundant calculations. All particles and their neighbors

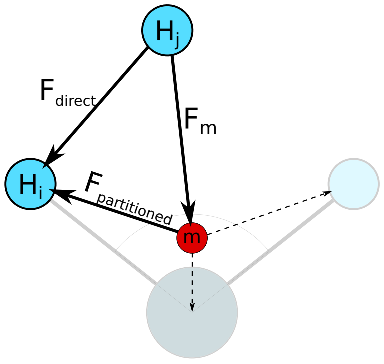

Figure 3. We consider the electrostatic force of hydrogen **j** on hydrogen **i** as part of a water molecule in the TIP4P model. It consists of two components: direct action, calculated as the usual Coulomb interaction, and indirect, through a virtual charge. Indirect action is calculated as the Coulomb interaction of hydrogen **j** with a virtual charge **m**, and then the distribution of this force.

are considered on the GPU. In the first step, the atoms that make up the molecule are detected, and then the coordinate of the virtual charge is calculated and stored. These procedures are carried out only once for each molecule, two other atoms read the information that is already stored. Then the force calculation is made: for hydrogen, a direct effect on it, and for oxygen, the effect on the displaced charge is calculated. For oxygen, this value is stored in a separate array.

The final part of the calculation is placed in a separate GPU-kernel, since these calculations should not begin before the first kernel is done for all particles and all molecular structures and preliminary force values are determined. In this part of the code, the forces acting on a displaced charge, calculated and stored in the first kernel, are distributed between the three atoms of the molecule.

Special cases that require separate processing complicate the calculation. For some local hydrogens, “ghost” oxygen may be related (Figure 4). So the contribution values will not be naturally calculated and saved for the second kernel for them since the use of Newtons third law is turned off in the GPU calculation and reverse synchronization (especially between the steps of calculating pairwise interaction) is not possible. These particles are processed according to the principle of redundant calculations, which was described in the Section 3.1: in the first kernel, a molecular structure is compiled for such hydrogens and the force acting on the displaced charge is calculated. It is necessary to increase the radius of consideration of electrostatic interactions (the radius of cutting of the force itself remains the same) to make it work correctly, and the effect of hydrogen on the oxygen of its own molecule also requires accurate accounting.

4. Results

4.1. Verification

For verification, we used the tried approach based on three criteria [15]. Conservation of full energy helps to find the first errors — it is a basic, but sensitive criterion. Since we have a reference calculation based on LAMMPS, it is convenient to use the congruence

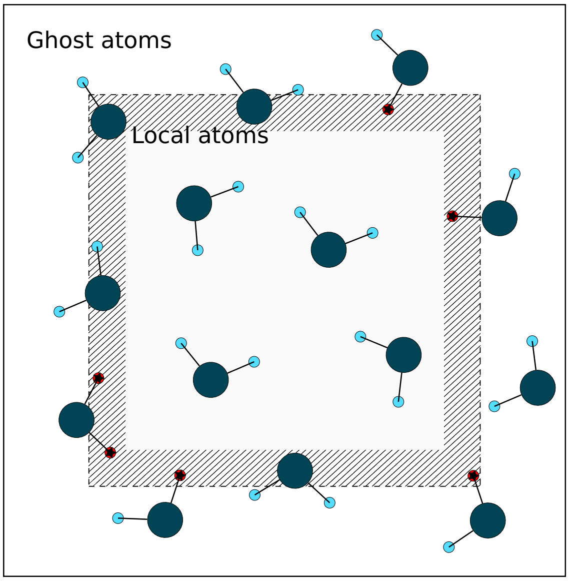

Figure 4. Light-gray area outlines the domain where the local atoms are found. It includes the shadowed area where oxygens can be found as "ghost atoms" for the corresponding hydrogens since they are not local. But that area is a small fraction of the total for typical cases.

of potential energy (Figure 6) as the second criterion, which is a function of the coordinates of all particles in the system and is also calculated simultaneously with forces in our code. The third criterion comes from the stochastic properties of the MD calculation [16]. The average displacement of the coordinates and particle velocities (Figure 5) in our simulation compared to the reference one should be equal to the machine accuracy in the first steps, then an exponential increase of the error is observed, followed up to reaching a plateau. Hopping coordinate differences on Figure 5 are likely to be artifacts of processing periodic boundary conditions.

4.2. Performance

We used two platforms for testing:

1. 8 core Intel Xeon E5-2620v4 with GPU Nvidia GeForce GTX 1070 (Pascal)
2. 8 core AMD Epyc 7251 with GPU Nvidia Titan V (Volta)

The code for the GPU can be compiled in any of three modes: single, double and mixed precision, while the calculations on the CPU are always performed in double precision. We use mixed precision for the GTX 1070, as the Nvidia GeForce GPUs are significantly slower with double-precision arithmetics [17]. Mixed precision is acceptable for many molecular dynamics calculations. At the same time, on a newer or server-level GPUs that fully supports double precision, our algorithm shows good acceleration in double precision, as it can be seen in the example of the Titan V (Volta generation).

Figure 7 shows the time profile for different parts of the task when executing the program on the CPU and on the GPU on our hardware. The number of atoms in the simulation is 32000. In the tests, all processor cores were loaded using MPI parallelization.

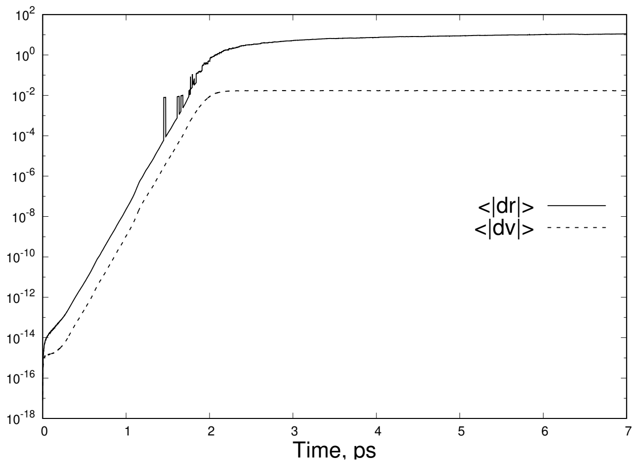

Figure 5. The normalized averaged deviations of coordinates and velocities on two trajectories calculated from identical initial conditions with LAMMPS lj/cut/tip4p/long and with our GPU-accelerated code. The exponential dependence with a further saturation regime is in agreement with the stochastic theory of molecular dynamics.

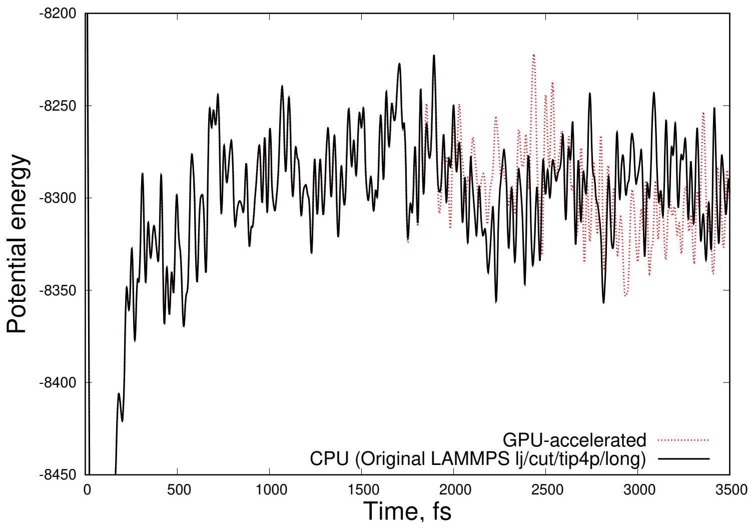

Figure 6. The potential energy calculated by our GPU code is equal to the potential energy calculated by the original LAMMPS lj/cut/tip4p/long at the beginning of the calculation, but the difference grows rapidly after passing the time of dynamic memory of the system.

It can be clearly seen that the time for calculating pairwise interactions is significantly reduced: almost seven times on the GTX 1070 and almost six times on the Titan V. It worth noting, that when one turn on the GPU module in the program, Neigh (the time it takes to build neighbor lists) approximately doubles. This is due to the disabling of the use of Newtons third law - thus the neighbors lists are doubled in size, considering each particle as **i** and as **j**, which explains the increase in time. The increased communication time at the stand with Titan V using double precision remains unclear. We conducted a

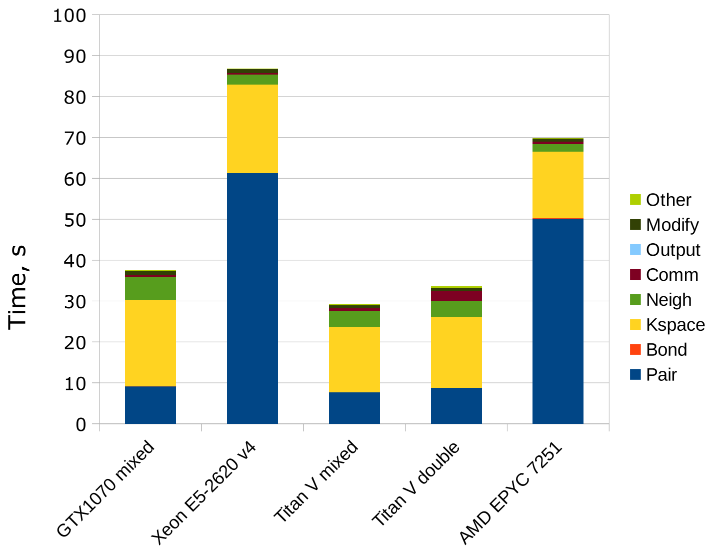

Figure 7. Time profile for 5 setups: 1) GPU algorithm in mixed precision on Nidia GTX 1070; 2) LAMMPS CPU algorithm on 8 core CPU Intel Xeon E5-2620v4; 3) GPU algorithm in mixed precision on Nidia Titan V; 4) GPU algorithm in double precision on Nidia Titan V; 5) LAMMPS CPU algorithm on 8 core CPU AMD Epyc 7251. Lower time is better. The code was implemented as "Pair" part of timestep breakdown. The number of atoms is 32000.

smaller number of experiments with Titan V, and, perhaps, it's work can be improved by proper tuning for the new architecture. We expect that the use of GPUs leads to better energy-efficiency [18]. Energy consumption is also affected by tuning.

5. Conclusion

An algorithm was developed and the corresponding code (available on GitHub [19]) was written for GPU-acceleration of the TIP4P water model as a part of the popular LAMMPS package. In this work, two solutions to this problem are described: with the execution of part of the algorithm on the CPU and the completely GPU-computed kernel. Verification of calculations is performed with both CUDA and OpenCL backends, it proves that we implemented the desired model with the machine precision. The second approach shows the overall acceleration about 55% compared to a fully loaded server processor, and the calculation of interactions is accelerated by almost six times. Future plans include refinement of the code and comprehensive testing of the stability.

Acknowledgment

The study was funded by RFBR according to the research project No. 18-37-00487 and supported within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and within the framework of a subsidy by the Russian Academic Excellence Project 5-100. Work has been partially supported by the grant of the President of Russian Federation for support of leading scientific schools grant NSh-5922.2018.8.

References

[1] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein, “Comparison of simple potential functions for simulating liquid water,” *The Journal of Chemical Physics*, vol. 79, no. 2, pp. 926–935, 1983.

[2] J. L. F. Abascal and C. Vega, “A general purpose model for the condensed phases of water: Tip4p/2005,” *The Journal of Chemical Physics*, vol. 123, no. 23, p. 234505, 2005.

[3] J. L. F. Abascal, E. Sanz, R. Garca Fernndez, and C. Vega, “A potential model for the study of ices and amorphous water: Tip4p/ice,” *The Journal of Chemical Physics*, vol. 122, no. 23, p. 234511, 2005.

[4] N. D. Kondratyuk, G. E. Norman, and V. V. Stegailov, “Quantum nuclear effects in water using centroid molecular dynamics,” *Journal of Physics: Conference Series*, vol. 946, p. 012109, jan 2018.

[5] M. J. Abraham, T. Murtola, R. Schulz, S. Pll, J. C. Smith, B. Hess, and E. Lindahl, “GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers,” *SoftwareX*, vol. 1-2, pp. 19 – 25, 2015.

[6] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, and V. S. Pande, “OpenMM 7: Rapid development of high performance algorithms for molecular dynamics,” *PLOS Computational Biology*, vol. 13, pp. 1–17, 07 2017.

[7] S. J. Plimpton, R. Pollock, and M. Stevens, “Particle-mesh Ewald and rRESPA for parallel molecular dynamics simulations,” in *PPSC*, 1997.

[8] W. M. Brown, P. Wang, S. J. Plimpton, and A. N. Tharrington, “Implementing molecular dynamics on hybrid high performance computers short range forces,” *Computer Physics Communications*, vol. 182, no. 4, pp. 898 – 911, 2011.

[9] W. M. Brown, A. Kohlmeyer, S. J. Plimpton, and A. N. Tharrington, “Implementing molecular dynamics on hybrid high performance computers particleparticle particle-mesh,” *Computer Physics Communications*, vol. 183, no. 3, pp. 449 – 459, 2012.

[10] W. M. Brown and M. Yamada, “Implementing molecular dynamics on hybrid high performance computersthree-body potentials,” *Computer Physics Communications*, vol. 184, no. 12, pp. 2785 – 2793, 2013.

[11] K. Halbiniak, R. Wyrzykowski, L. Szustak, and T. Olas, “Assessment of offload-based programming environments for hybrid cpumic platforms in numerical modeling of solidification,” *Simulation Modelling Practice and Theory*, vol. 87, pp. 48 – 72, 2018.

[12] B. Glinsky, I. Kulikov, I. Chernykh, D. Weins, A. Snytnikov, V. Nenashev, A. Andreev, V. Egunov, and E. Kharkov, *The Co-design of Astrophysical Code for Massively Parallel Supercomputers*, pp. 342–353. Cham: Springer International Publishing, 2016.

[13] K. Rojek and R. Wyrzykowski, “Performance modeling of 3D MPDATA simulations on GPU cluster,” *The Journal of Supercomputing*, vol. 73, pp. 664–675, Feb 2017.

[14] K. A. Feenstra, B. Hess, and H. J. C. Berendsen, “Improving efficiency of large time-scale molecular dynamics simulations of hydrogen-rich systems,” *Journal of Computational Chemistry*, vol. 20, no. 8, pp. 786–798, 1999.

[15] V. Nikolskii and V. Stegailov, “Domain-decomposition parallelization for molecular dynamics algorithm with short-ranged potentials on Epiphany architecture,” *Lobachevskii Journal of Mathematics*, vol. 39, pp. 1228–1238, Nov 2018.

[16] G. E. Norman and V. V. Stegailov, “Stochastic theory of the classical molecular dynamics method,” *Mathematical Models and Computer Simulations*, vol. 5, no. 4, pp. 305–333, 2013.

[17] V. P. Nikolskiy, V. V. Stegailov, and V. S. Vecher, “Efficiency of the Tegra K1 and X1 systems-on-chip for classical molecular dynamics,” in *2016 International Conference on High Performance Computing Simulation (HPCS)*, pp. 682–689, July 2016.

[18] F. Mantovani and E. Calore, “Performance and power analysis of hpc workloads on heterogeneous multi-node clusters,” *Journal of Low Power Electronics and Applications*, vol. 8, no. 2, 2018.

[19] https://github.com/Vsevak/lammps