The last decade of developments in parallelization of molecular dynamics simulation methods on heterogeneous supercomputers

Matea Turalija
Group for Applications and Services on Exascale Research Infrastructure (GASERI)
Faculty of Informatics and Digital Technologies, University of Rijeka
Rijeka, Croatia
matea.turalija@inf.uniri.hr

***Abstract*—Computer simulation, as a third way in science alongside the theoretical and experimental approaches, offers a range of methods to study and predict the properties of various physical and chemical processes. Among them, molecular dynamics simulation uses classical mechanics and often runs on supercomputers to help the scientists understand the behavior of large molecules and molecular systems, including proteins, membranes, and nucleic acids.**

The increasing complexity of molecular dynamics simulations studying biologically relevant systems leads to big demands on computation resources and time. Therefore, a significant portion of present research is focused on the development of simulation methods and their implementations in heterogeneous, parallel, and distributed environments with the goal of with the goal of reducing the time to solution.

The prevailing method for calculating computationally expensive long-range interactions on the last generation of supercomputers was particle mesh Ewald. It is presently being substituted for the fast multipole method to support the new highly parallel exascale supercomputers featured in the first few sports of the TOP500 list. The present implementation of the aforementioned method has the disadvantage of allowing only cubic simulation boxes; our future work includes extending the method and its implementation with support for non-cubic simulation boxes, focusing primarily on the practically relevant ones.

The evolution of networking devices resulted in smart network interface cards, also known as data processing units, including both data transfer and compute capabilites. In the era of increasing data volume, data processing units are becoming more and more prominent in data centers and can be expected in future supercomputers in some capacity. While the present implemenation of parallel algorithms used in the molecular dynamics simulation supports the execution on various heterogeneous systems, this support does not include the systems with data processing units. Our future work includes extending the parallelization of simulation methods to support the execution on data processing units.

Molecular dynamics simulation capabilities are already expanding with the first batch of exascale supercomputers in operation. Such supercomputers are able to rapidly produce and analyze huge amounts of data, and they will certainly have a profound impact in this as well as the next decade. Our planned contributions will help make the molecular dynamics simulation methods both faster and more efficient.

***Index Terms*—molecular dynamics simulation, exascale supercomputers, parallelization techniques, heterogeneous computing, data processing units,**

This research was funded by Horizon 2020 Framework Programme grant number 951732, the European High-Performance Computing Joint Undertaking (EuroHPC JU) partnership's project National Competence Centres in the framework ofEuroHPC (EuroCC).

Introduction and motivation

In the recent decades, computer simulations have become an indispensable tool for the study and prediction of physical and chemical processes. The phenomena that can be studied in this way range from the smallest length scales, such as quantum mechanics of matter at nanoscale, to the dynamics of the universe at extragalactic distances. Similarly, the time scales on which the observed phenomena take place range from femtoseconds to several billion years, and the masses range from 10^{-27} kilograms for individual atoms to 10^{40} kilograms for entire galaxies.

The wide range of phenomena described shows that experiments cannot always be carried out in the desired way. In astrophysics, for example, the models that describe nature well enough are often so complicated that no analytical solution can be found. For example, if we take the motion of the planets and the gravitational force acting between the planets according to Newton's law, there are generally no analytical solutions for the case of three or more bodies. This is also true for our planetary system as well as for the stars in our galaxy.

Furthermore, expensive experimental setups can be avoided by using simulations. Specifically, we can observe phenomena that would be difficult or impossible to measure, or experiments would take too long or proceed too quickly to be visible. Chemical reactions, for example, can proceed at different rates. Some occur so quickly, almost instantaneously, that it is difficult to follow them, while others can drag on for months or even years.

These are just some of the reasons why computer simulation has recently emerged as a third approach in science, alongside the experimental and theoretical approaches. In this way, simulation enables the study of phenomena that were previously inaccessible to experiment.

In most cases, the complexity of the model leads to enormous demands on memory and computer time. However, the rapid development of computer technology overcomes these difficulties and allows us today to perform more and more realistic simulations. Therefore, the current research is mainly focused on the development of methods and related algorithms that allow the quickest possible calculation of the problems (multilevel and multiscale methods, multipole methods, particle mesh Ewald, fast Fourier transform, sparse linear algebra, etc.) and that can approximate the solution with as little memory as possible.

Parallel programming on heterogeneous systems has also attracted much attention in the last 15 years. In parallel computers, several dozen to many thousands of powerful processors are combined into a single computer system that can compute simultaneously and independently with mutual communication to perform operations in a reasonable amount of time. However, the heterogeneity of the system leads to difficulties in communication and load balancing between central processing unit (CPU) and graphics processing unit (GPU), which, given a myriad of possible combinations of CPUs and GPUs, remains a hot research topic even today.

Recent advances in algorithms, software, and hardware have enabled supercomputers to break the 1 exaFLOP threshold, starting the era of exascale supercomputing. Just this year, the most powerful supercomputer ever, Frontier, was launched, which is also the first true exascale supercomputer [1]. There is no doubt that Frontier is ushering in a new milestone in high-performance computing to solve challenges in a wide range of scientific fields and improve our understanding of the world and our lives.

In this paper, we review the state of the art molecular dynamics simulation parallelization techniques. We survey the latest technologies, ideas, and features discussed, developed, and used in molecular dynamics during the last decade. We also present a proposal to improve the fast multipole method for computing long-range interactions on today's supercomputers and a proposal to develop a new library exposing the message passing interface to perform the expensive computations on data processing units. Our main motivation is to find the best way to execute molecular dynamics algorithms on exascale supercomputers.

This paper is organized as follows. We first present the main topics related to molecular dynamics simulation. We follow with an overview of the latest technologies, ideas, features, software packages, and supercomputers that enable these simulations. Finally, we conclude the paper with an outlook on our planned future work.

THE FUNDAMENTALS OF MOLECULAR DYNAMICS SIMULATION

Molecular dynamics (MD) simulation is a computer simulation method that allows the analysis of the motion of atoms and molecules whose mutual interactions obey the known laws of classical physics. Molecular systems generally consist of a large number of particles, so it is impossible to analytically determine the properties of such complex systems. MD overcomes this problem by numerically solving Newton's equations of motion for a group of atoms. The method is used mainly in biophysics, chemical physics, and materials science. Since it establishes a connection between experiment and theory, we can consider it as a virtual experiment.

For molecular dynamics simulations to be possible, it must be possible to estimate the potential energy of particles and the forces between them very quickly. To describe the interactions between and within atoms, the interactions in molecular dynamics are approximated by a simple empirical potential energy function called the force fields. The force field (FF) describes the dependence of the energy of the system on the positions of its particles. It consists of the potential energy and the associated. The force field replaces the real potential with a simplified model. Ideally, it should be simple enough to be evaluated quickly, yet detailed enough to show the properties of interest [2]. The general form of the force field can be written as follows:

$$\begin{aligned} U = U_{bond} + U_{angle} + U_{tors} + U_{LJ} + U_C \\ U = \sum_i \frac{1}{2} k_{b,i}(r_i - r_{0,i})^2 + \\ \sum_i \frac{1}{2} k_{a,i}(\theta_i - \theta_{0,i})^2 + \\ \sum_i \frac{1}{2} k_{t,i}(1 + cos(n\phi - \delta)) + \\ \sum_{i<j} 4\epsilon[(\frac{\sigma}{r_{ij}})^{12} - (\frac{\sigma}{r_{ij}})^6] + \sum_{i<j} \frac{q_i q_j e^2}{4\epsilon\pi r_{ij}} \end{aligned} \tag{1}$$

Parameters $k_{b,i}, k_{a,i}, k_{t,i}$ etc. are taken to represent experimental data or quantum mechanical calculations.

Bonded interactions

The first two expressions in eq. 1 represent bond stretching (U_{bond}) and angle bending (U_{angle}). Both are described by a simple harmonic potential, which is a good approximation only for small deviations from the equilibrium position. For a more exact description of the molecular movements, anharmonic energy expressions of higher order are necessary. The bond potential is used to model the interaction of covalently bonded atoms in a molecule. It is a pretty poor approximation at extreme stretching, but the bonds are stiff enough to work at moderate temperatures. The Morse potential is more accurate, but more expensive to calculate. The angle potential describes the bond bending energy and is defined for each triplet of bonded atoms [3]. The third term (U_{tors}) is often called the torsion term and represents the potential energy of the molecular system that describes the coupling between adjacent bonds, angles and dihedrals. The torsion energy is defined for all 4 successively bonded atoms. The presented expressions belong to the group of the so-called bonding interactions and describe the interactions between several neighboring atoms within the molecule.

Non-bonded interactions

Non-bonded potentials are non-electrostatic and electrostatic interactions between all pairs of atoms. The nonelectrostatic potential energy is most commonly described by the Lennard-Jones potential (U_{LJ}), which approximates the potential energy of the interaction between a pair of nonbonded atoms or molecules. The term r^{-12} approximates the strong Pauli repulsion arising from the overlap of electronic orbitals, while the term r^{-6} describes the weaker attractive forces acting between local, dynamically induced dipoles in valence orbitals. The electrostatic potential is described by Coulomb's law (U_C), where point charges are associated with the positions of atomic nuclei. Interactions can be classified as short-range and long-range interactions. In a short-range interaction, the potential decreases faster than r^{-d}, where r is the distance between the particles and d is the dimension. Otherwise, it is a long-range interaction. Accordingly, Lennard-Jones interactions are short-range, while Coulomb interactions are long-range [3].

Potential energy

After determining the potential and taking into account the fact that we are dealing with conservative forces (the work done during the motion of the particle does not depend on the path), we can calculate the gradient of potential energy, i.e. the force. By determining the forces acting on individual atoms or molecules, the new positions and velocities of the particles are obtained by numerical integration methods of Newton's equations of motion. One of the simplest algorithms for calculating the orbit of an atom is Euler's algorithm. The Euler's algorithm uses a second-order Taylor expansion to estimate position and velocity for the next time step. Euler's algorithm is not time reversible and is therefore computationally demanding. Nevertheless, Euler's scheme can be used to integrate some other equations of motion. For example, GROMACS molecular dynamics simulation software [4] provides an Euler integrator for Brownian or positional Langevin dynamics. Another algorithm for simulating the evolution of the system over time is the Verlet algorithm. The algorithm uses current positions and forces as well as previous positions to calculate the position at the next time step. This is inconvenient when performing simulations when only current positions are available. Although velocities are not required for calculating trajectories, they can be useful for calculating other quantities, such as kinetic energy. The most commonly used algorithm in MD simulations due to its simplicity and stability is the Velocity Verlet algorithm. The Velocity Verlet algorithm is an improved version of the Euler algorithm with acceleration estimates at the next time step. Unlike the basic Verlet algorithm, this algorithm explicitly considers velocity, eliminating the first time step problem. The Verlet algorithm is time reversible and saves energy. The Leap Frog algorithm is a variant of the Velocity Verlet. Both integrators provide equivalent trajectories, but the difference is that velocities are not computed simultaneously with positions. Leap integration corresponds to updating positions and velocities at nested time points arranged to "leapfrog" each other [3]. The initial configuration of the system is usually a known structure obtained from a database or experiment, and the initial velocities are randomly assigned to each atom or molecule according to the Maxwell-Boltzmann distribution at a given temperature.

Simulation duration

The complexity of the simulated system is determined by the available computer capacity. System size (number of particles), the time step, and the total duration of the simulation should be chosen so that the calculation is completed in a reasonable amount of time. However, the simulation should be long enough to faithfully represent the natural process that we are studying. The size of the integration time step is one of the more important factors affecting the total simulation duration. It is the time interval between possible measurements. To avoid instability of the simulation, one should use a time step shorter than the fastest motions in the molecule, such as atomic oscillations, which are usually on the order of a few femtoseconds ($10^{-15}s$). This value can be increased by replacing the binding oscillations with holonomic (time invariant) constraints.

Constraints

Instead of explicitly integrating the fast degrees of freedom, the bond lengths are fixed at ideal values. By eliminating some fast degrees of freedom in this way, larger time steps can be used. This is usually achieved by a constraining force acting in the opposite direction along the bond. In this case, this has no effect on the total energy of the simulation system, since the total work of the constraining forces is zero. From the point of view of statistical mechanics, the constrained system is not equivalent to the system we started with. However, the bond displacements are usually small and constrained to a single equilibrium value by the harmonic potentials, and the constraints have little effect on the average geometries. Moreover, the force field can be parameterized to restore the flexibility lost due to the added stiffness of the system [5], [6].

Several algorithms have been developed specifically for small or large molecules. A widely used iterative algorithm for large molecules is SHAKE. The algorithm uses distance and angle constraints that sequentially reset all constrained distances and angles to the configured values until the desired tolerance is reached. However, SHAKE has the disadvantage that no solutions can be found if the shifts are large, and its iterative nature makes it difficult to parallelize. For small molecules, SETTLE is a faster algorithm. SETTLE solves this problem analytically, but for larger molecules this is too complicated. Therefore, there is a need for a faster and more reliable algorithm. An alternative is the LINear Constraint Solver (LINCS) algorithm, which uses a power series expansion to determine how to move the atoms so that all constraints are satisfied. Alogitam has better convergence properties and is more stable than SHAKE, but is not suitable for constraining bonds and angles. There are other ways to increase simulation speed. For example, long-range electrostatic interactions can be computed less frequently than short-range interactions. Also, an intermediate time step can be used for short range non-bound interactions, with only bound interactions being calculated in each time step [3], [7].

Simulation software

The enormous potential for applications has led to the implementation of molecular dynamics in many commercial and research software packages. In this section, we provide a brief overview of the leading software packages for performing molecular dynamics simulations [8].

- Amber is a proprietary suite of biomolecular simulation programs that is available under an academic license [9].
- CHARMM is a molecular simulation program for many-particle systems that is freely available to academic and non-profit users [10]. It Includes support for QM/MM, MM/CG and a number of implicit solvent models.
- DL_POLY is a general purpose classical molecular dynamics simulation software [11]. It is free for academic scientists pursuing scientific research of a non-commercial nature.
- EGO is a parallel program for molecular dynamics simulations of biomolecules [12]. It is is freely available.
- GROMACS is a software package for high-performance molecular dynamics and output analysis [4]. It is a free and open-source software released under the GNU Lesser General Public License.
- GROMOS is a software package for dynamic modeling of (bio)molecules [13]. It is released under a proprietary license.
- HOOMD-blue is a general-purpose particle simulation toolkit that implements molecular dynamics and hard particle Monte Carlo [14]. It is available under a non-copyleft open-source license.
- The ITAP Molecular Dynamics (IMD) program is a software package for classical molecular dynamics simulations [15]. It is a free and open-source software available under GNU General Public License.
- LAMMPS is a classical molecular dynamics simulation code with a focus on material modeling [16]. It is a free and open-source software available under GNU General Public License.
- Moldy is a short-range molecular dynamics package [17]. It is a free and open-source software available under GNU General Public License.
- MOSCITO is a software for molecular dynamics simulation specialized for gas and condensed phases [18]. It is a free and open-source software available under GNU General Public License.
- NAMD is a software for molecular dynamics simulation [19]. It is a proprietary software tha is free for noncommercial use.
- OpenMM is a high-performance toolkit for molecular simulation [20]. It is a free and open-source software available under MIT and GNU General Public Licenses.
- ORAC isa n OpenMP/MPI molecular dynamics engine to simulate solvated biomolecules [21]. It is a free and open-source software available under GNU General Public License.
- Tinker is a suite of software applications for molecular dynamics simulation [22]. It is proprietary freeware.

We chose to focus on GROMACS since it is the most popular actively developed open-source software. It also offers very good performance while being extensible.

Algorithm complexity and parallelization

The computational cost of bound interactions $O(N)$, N being the number or particles, represents only a small fraction of the total simulation time. However, the most computationally intensive task is the calculation of the unbound van der Waals interactions and the electrostatic interactions. In this step, the interactions for each of the N particles in the simulation are calculated relative to all other $(N-1)$ particles. This leads to $O(N^2)$ complexity. It is common to divide the non-bonded interactions between atoms into short and long distances. This is achieved by setting a limiting distance above which only long distance interactions between two atoms are assumed. In this way, different specialized algorithms can be used to handle each case separately, improving the overall efficiency of the simulation. Effective methods for excluding pairs of atoms separated by a large distance are neighbor searching methods. Particle neighbors can be determined in two ways. First, by dividing the simulation system into grid cells (cell list), as shown in fig. 1 Second, by generating a list of neighbors for each particle (Verlet list), as shown in fig. 2.

The cell list method divides the simulation domain into n cells with edge length greater than or equal to the cutoff radius of the interaction to be calculated. The interaction potential for each particle is then calculated as the sum of the pairwise interactions between the particle and all other particles in the same cell and all other particles in neighboring cells. A Verlet list stores all particles within the bounding distance of each particle plus an additional buffer distance. Although all pairwise distances must be evaluated to create the Verlet list, it can be used for several consecutive time steps until a particle has moved more than half the buffer. At that point, the list is invalid and a new list must be created. Verlet provides a more efficient calculation of pairwise interactions at the cost of a relatively large memory requirement, which can be a limiting factor. This method can also be easily modified for Monte

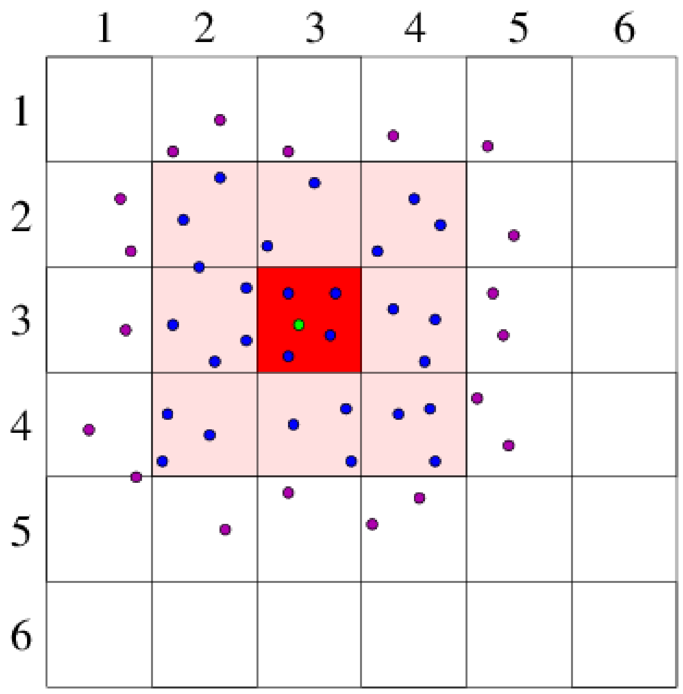

Figure 1: Cell lists (image source: [3])

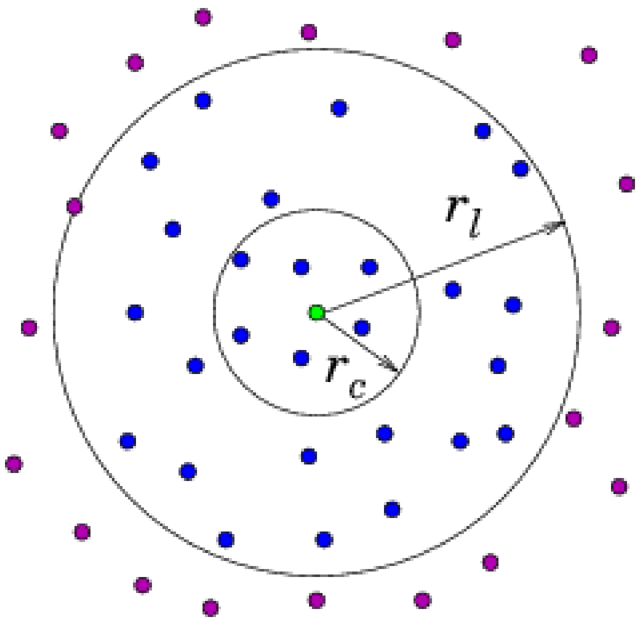

Figure 2: Verlet lists (image source: [3])

Carlo simulations. In practice, almost all simulations can run in parallel and use a combination of spatial decomposition and Verlet lists [3], [8].

Electrostatic interactions occurring over long distances decrease slowly, and increasing the limiting distance to account for interactions over long distances can dramatically increase computational costs. In periodic simulation systems, the most commonly used method for calculating electrostatic interactions over long distances is particle mesh Ewald method. In this method, the electrostatic interaction is divided into two parts: a short-range contribution and a long-range contribution. The short-range contribution is calculated in real space, while the long-range contribution is calculated outside the boundary radius using the Fourier transform. The advantage of this method is the fast convergence of energy. The method requires charge neutrality of the molecular system to accurately calculate the total Coulomb interaction. Variations of the above method are particle-particle-particle-mesh (P3M) and particle mesh Ewald (PME) with $O(N \log N)$ complexity [23], [24]. Better scaling of the PME method can be achieved using the eight shell domain decomposition method [25] or the smooth particle mesh Ewald (SPME) method [26], [27]. The current generation of computers takes advantage of parallelism over a large number of central processors and accelerators to speed up the processes. However, on such computers, PME becomes a limiting scaling factor. An alternative is the fast multipole method (FMM) with $O(N)$ complexity [28].

The growing interest in the complexity of biological interactions is leading to an ever-increasing need for computer simulations that require not only powerful and advanced hardware, but also adaptable software that can accommodate large numbers of atoms interacting through complex force fields. Because of the way molecular dynamics simulations are structured, there is much scope for parallelizing algorithms to to take advantage of multicore processors that can speed up calculations by several orders of magnitude. In parallel calculations of molecular dynamics simulations, processors are used in parallel to calculate forces and update coordinates. MD simulation time steps are inherently sequential: the most recent coordinates are needed to accurately compute the forces, and coordinates can only be updated when the most recent forces have been computed. While the calculation of forces and the updating of coordinates occur in parallel, the processors must exchange forces and coordinates between these two calculations [29].

Multilevel heterogeneous parallelization extends parallelization to treat each level of hardware parallelism separately to enable the fastest possible computation of parts of a individual simulation step. At the lowest level, parallel computer systems are categorized according to whether the data stream and/or the instruction stream are processed in parallel. In this way, the basic types SISD (Single Instruction/Single Data Stream – the classic microprocessor), SIMD (Single Instruction/Multiple Data Stream – the graphics processing unit) and MIMD (Multiple Instruction/Multiple Data Stream) can be distinguished.

SIMD CPU units provide fine-grained parallel execution of data, as shown in fig. 3. However, the common parallelization scheme for molecular simulation and most other codes today is Single-Program, Multiple-Data (SPMD), where all processors execute the same code but with different data. This is an obvious solution for decomposing a system with hundreds of thousands of similar particles. However, for the implementation of parallel algorithms, such as the PME algorithm, this approach had some drawbacks. The remedy is to support parallelization with Multiple-Program, Multiple-Data (MPMD) [4], [25].

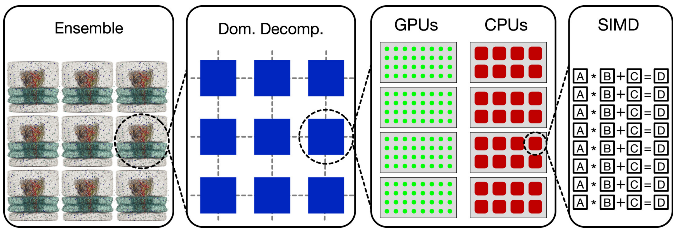

Figure 3: Multilevel heterogeneous parallelization (image source: [30])

The next level of parallelization involves the use of GPU accelerators. Accelerator tasks are executed asynchronously using CUDA, OpenCL, and SYCL APIs that schedule GPU tasks and CPU-GPU data movement to ensure concurrent task execution. This design aims to maximize CPU-GPU execution overlap, reduce the number of transfers by moving data early, keep data on the accelerator as long as possible, ensure that transfers overlap with computations, and optimize critical path task scheduling to reduce time per step [31]. Modern GPUs use a SIMD-like execution called Single Instruction, Multiple Threads (SIMT). This means that each of these processors can execute hundreds of threads simultaneously, providing an additional layer of parallelization. SIMT allows the GPU to split a single task into a large number of identical tasks that execute independently on each thread on the same or different cores. At the same time, the threads can communicate with each other and, if necessary, cooperate in executing part of the computation. The execution on the CPU is additionally parallelized using multithreading via OpenMP [32].

This heterogeneous parallelization model introduces additional complexity, which comes at a price. Different parts of the program running concurrently must communicate with each other at some point. If the program is divided into a large enough number of parallel parts and these parts must constantly communicate with each other, the time spent on communication can easily negate any performance gain or even increase the overall runtime. Therefore, slow CPU-GPU transfers are more difficult to solve with overlays because computations are faster than data movements. For this reason, recent efforts have focused on increasingly eliminating CPU-GPU data movement and shifting more work to powerful accelerators [31].

Finally, since most interactions in molecular simulations are local, high-level domain decomposition (DD) is a natural way to decompose a system. Domain decomposition assigns a spatial domain to each rank, which then integrates the equations of motion for the particles that are currently in their local domain. GROMACS currently uses the eight-shell domain decomposition [25], but for certain systems or hardware architectures it may be useful to use the midpoint method [4] DD was also originally used for intra-node parallelism using MPI. Since the DD algorithm ensures data locality, it is surprisingly well suited for NUMA architectures. However, this comes with challenges related to exposing finer parallelism between cores and limits the ability to use efficient data sharing with shared caches [31].

Exploiting low-level parallelism can be tedious and is often avoided in favor of using more hardware to achieve the desired time to resolution. However, the evolution of hardware makes this trade-off increasingly difficult. The end of frequency scaling of microprocessors and the resulting increase in hardware parallelism means that exploiting all levels of parallelism is a necessity for getting good performance on modern hardware, not an option [31].

Use of general-purpose and specialized hardware

Advances in algorithms, software and hardware have enabled very fast computer simulations that can simulate millions of atoms in a reasonable amount of time. Today's dominant method for calculating computationally demanding long-range electrostatic interactions is the Particle Mesh Ewald (PME). The PME method is suitable for parallelization at the scale of last generation supercomputers and allows very fast computation of the mentioned interactions on them. However, for large parallel simulations with multiple nodes, PME becomes a limiting scaling factor as it requires communication between all nodes. As a result, the number of messages exchanged increases quadratically with the number of nodes involved. To enable efficient and scalable molecular simulations on exascale supercomputers, a method with better scaling properties is needed. The Fast Multipole Method (FMM) is such a method and there is a proposal for its implementation for MD simulations [28], [30], [33].

The increasing complexity and demands of molecular dynamics simulations require sufficient hardware to perform them. The idea of multicore processors allows a high degree of parallelization of computations, which is very suitable for efficient, balanced and scalable molecular simulation. The parallelism just mentioned in combination with graphics processing units (GPUs) has attracted much attention in the last 15 years. To take advantage of multi-core processors and speed up calculations by several orders of magnitude, there are many efficient implementation strategies and algorithms available today [32], [34], [35]. Although heterogeneous CPU-GPU acceleration is powerful enough to perform all operations in a reasonable time, the bottleneck of molecular dynamics development is slowly shifting from hardware to communication and load balancing between processors. This is still a hot research topic today, considering the large number of possible combinations of CPU and GPU [31], [36].

In the era of growing data volumes that computer scientists deal with, network interface cards (NICs) that connect a computer to a computer network can be additionally used to process the data they transmit. Therefore, the NIC offers the possibility of reducing the load on the main processor, so that the functions of processing the received or sent network traffic, which were traditionally performed by the main processor, are now performed by the NIC itself, called SmartNIC (smart network interface card). As data processing has expanded beyond servers and personal computers to large, centralized data centers, whose network speeds continue to increase, currently reaching 400 GB per second, a third level of offloading is required. These demands for big data have led to the development of data processing units, or DPUs for short.

DPUs with their three primary functions, which are processing, networking, and acceleration, have become the third main pillar of computing along with CPUs and GPUs, and the speed with which data is processed becomes incomparable. The next step in accelerating molecular dynamics could be the conversion of existing tools for molecular dynamics simulations into tools that perform some of the computations on DPUs. This could make molecular dynamics acceleration, which has been limited to a few supercomputers, available to a wide range of high-performance computers [37].

Even with the use of state-of-the-art algorithms and efficient parallel implementations, the cost of estimating the forces between all particles of large systems (10^6 atoms) presents a new challenge for high-performance computing. Exceptions are supercomputers for special purposes. Examples of such supercomputers are Anton and MDGRAPE, massive parallel supercomputers for molecular dynamics simulations [38]–[42].

Molecular simulation capabilities are expanding with the advent of exascale supercomputers. With a trillion (10^{18}) calculations per second, exascale supercomputers like Frontier will be able to rapidly analyze massive amounts of data and make simulations more realistic [1]. However, scalability, high communication cost, data distribution, heterogeneity, power management, etc. are still current issues of exascale supercomputers and present many challenges to developers [43].

Conclusion and future work

In surveying the developments in molecular dynamics simulation over the last decade, we found that the fast multipole method has been developed for the very important and computationally intensive calculation of long-range electrostatic interactions. The mentioned method replaces the prevailing particle mesh Ewald method adapted for parallelization of last generation supercomputers. With the increase in number of nodes one exascale supercomputers, the FMM method is expected to outperform PME. However, an existing limitation of this implementation method is the ability to use only cubic simulation boxes. The possibility of introducing other types of simulation boxes would allow the amount of water simulated around the biomolecular system to be reduced, which in turn would improve simulation performance.

Our future work on this topic would include support for other types of simulation boxes, eventually enabling the use of boxes such as the rhombic dodecahedron. The advantage of such "truncated" cubes is that the amount of water to be simulated is reduced, thus improving the simulation performance. As a first step towards implementing a general non-cubic box, we consider the possibility of simulating cuboidal simulation boxes by irregularly subdividing the boxes into sub-boxes. In the case of regular subdivision, the box is divided into eight sub-boxes at each step. The system can be generalized to divide into 6 or 4 sub-boxes if this corresponds to the simulated system. This would make it possible to go from a cube, which is very different from a cube, to cuboid-shaped sub-boxes, which are a good approximation of cubes, in a few steps.

The demands of data-centric computing have led to the development of data processing units. Although they are increasingly used in data centers and supercomputers, they are still unknown to the general public. The open problems that remain with the introduction of DPUs in computer architecture are how to adapt existing tools to DPUs. Another step toward accelerating molecular dynamics could be to perform some of the computations of existing molecular dynamics simulation tools on DPUs.

To enable the transfer of molecular dynamics to DPUs, we are interested in further work on the development of a new Message Passing Interface (MPI) library that would enable the conversion of existing MD simulation tools into tools that perform some of the computation on DPUs. The motivation for this development is to make molecular dynamics acceleration, previously limited to the Anton 2 supercomputer, available to a wide range of high performance computers.

The desire to understand complex biological phenomena at the molecular level has greatly stimulated the development of molecular dynamics. Molecular dynamics simulations are progressing rapidly thanks to improvements in methodology, algorithms, and increasing computer power. Today, we can study complex systems involving millions of atoms, solve problems in many scientific fields, and there is hope for better understanding and discovery of new drugs for many deadly diseases such as viral infections, autoimmunity, and even cancer.

The potential capabilities of molecular simulation are expanding with the advent of exascale supercomputers. However, software scalability, high communication cost, data distribution, heterogeneity, power management, and many other challenges remain unsolved in molecular dynamics simulation as well as other fields of scientific computing.

References

[1] D. Schneider, “The Exascale Era is Upon Us: The Frontier supercomputer may be the first to reach 1,000,000,000,000,000,000 operations per second,” *IEEE Spectrum*, vol. 59, no. 1, pp. 34–35, Jan. 2022.

[2] M. A. González, “Force fields and molecular dynamics simulations,” *École thématique de la Société Française de la Neutronique*, vol. 12, pp. 169–200, 2011.

[3] web-page, “Practical considerations for Molecular Dynamics.”

[4] web-page, “GROMACS.”

[5] R. Elber, A. P. Ruymgaart, and B. Hess, “SHAKE parallelization,” *The European physical journal. Special topics*, vol. 200, no. 1, pp. 211–223, Nov. 2011.

[6] R. D. Skeel and S. Reich, “Corrected potential energy functions for constrained molecular dynamics,” *The European Physical Journal Special Topics*, vol. 200, no. 1, pp. 55–72, Nov. 2011.

[7] B. Hess, H. Bekker, H. J. C. Berendsen, and J. G. E. M. Fraaije, “LINCS: A linear constraint solver for molecular simulations,” *Journal of Computational Chemistry*, vol. 18, no. 12, pp. 1463–1472, 1997.

[8] D. Vlachakis, E. Bencurova, N. Papangelopoulos, and S. Kossida, “Current state-of-the-art molecular dynamics methods and applications,” *Advances in Protein Chemistry and Structural Biology*, vol. 94, pp. 269–313, 2014.

[9] web-page, “The Amber Molecular Dynamics Package.”

[10] web-page, “CHARMM: Home.”

[11] web-page, “DL_POLY.”

[12] web-page, “EGO.”

[13] web-page, “GROMOS.”

[14] web-page, “HOOMD-blue - Home.”

[15] web-page, “IMD - The ITAP Molecular Dynamics Program.”

[16] web-page, “LAMMPS Documentation (15 Sep 2022 version) — LAMMPS documentation.”

[17] web-page, “Moldy.”

[18] web-page, “MOSCITO Homepage.”

[19] web-page, “NAMD 2.14 User’s Guide.”

[20] web-page, “OpenMM.”

[21] web-page, “ORAC (release 6) - A molecular dynamics program to simulate solvated biomolecules.”

[22] web-page, “Tinker User’s Guide documentation.”

[23] T. Darden, D. York, and L. Pedersen, “Particle mesh Ewald: An N log(N) method for Ewald sums in large systems,” *The Journal of Chemical Physics*, vol. 98, no. 12, pp. 10089–10092, Jun. 1993.

[24] P. E. Kyziropoulos, C. K. Filelis-Papadopoulos, and G. A. Gravvanis, “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*, vol. 2015, p. e450980, Apr. 2015.

[25] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, “GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation,” *Journal of Chemical Theory and Computation*, vol. 4, no. 3, pp. 435–447, Mar. 2008.

[26] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen, “A smooth particle mesh Ewald method,” *The Journal of Chemical Physics*, vol. 103, no. 19, pp. 8577–8593, Nov. 1995.

[27] M. J. Abraham and J. E. Gready, “Optimization of parameters for molecular dynamics simulation using smooth particle-mesh Ewald in GROMACS 4.5,” *Journal of Computational Chemistry*, vol. 32, no. 9, pp. 2031–2040, 2011.

[28] B. Kohnke, C. Kutzner, and H. Grubmüller, “A GPU-Accelerated Fast Multipole Method for GROMACS: Performance and Accuracy,” *Journal of Chemical Theory and Computation*, vol. 16, no. 11, pp. 6938–6949, Nov. 2020.

[29] D. Janežič, U. Borštnik, and M. Praprotnik, “Parallel Approaches in Molecular Dynamics Simulations,” in *Parallel Computing: Numerics, Applications, and Trends*, R. Trobec, M. Vajteršic, and P. Zinterhof, Eds. London: Springer, 2009, pp. 281–305.

[30] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, 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, Sep. 2015.

[31] S. Páll, A. Zhmurov, P. Bauer, M. Abraham, M. Lundborg, A. Gray, B. Hess, and E. Lindahl, “Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS,” *The Journal of Chemical Physics*, vol. 153, no. 13, p. 134110, Oct. 2020.

[32] S. Páll, M. J. Abraham, C. Kutzner, B. Hess, and E. Lindahl, “Tackling Exascale Software Challenges in Molecular Dynamics Simulations with GROMACS,” in *Solving Software Challenges for Exascale*, 2015, pp. 3–27.

[33] B. Kohnke, C. Kutzner, A. Beckmann, G. Lube, I. Kabadshow, H. Dachsel, and H. Grubmüller, "A CUDA fast multipole method with highly efficient M2L far field evaluation," *The International Journal of High Performance Computing Applications*, vol. 35, no. 1, pp. 97–117, Jan. 2021.

[34] J. Jung, C. Kobayashi, K. Kasahara, C. Tan, A. Kuroda, K. Minami, S. Ishiduki, T. Nishiki, H. Inoue, Y. Ishikawa, M. Feig, and Y. Sugita, "New parallel computing algorithm of molecular dynamics for extremely huge scale biological systems," *Journal of Computational Chemistry*, vol. 42, no. 4, pp. 231–241, 2021.

[35] J. Jung, W. Nishima, M. Daniels, G. Bascom, C. Kobayashi, A. Adedoyin, M. Wall, A. Lappala, D. Phillips, W. Fischer, C.-S. Tung, T. Schlick, Y. Sugita, and K. Y. Sanbonmatsu, "Scaling molecular dynamics beyond 100,000 processor cores for large-scale biophysical simulations," *Journal of Computational Chemistry*, vol. 40, no. 21, pp. 1919–1930, 2019.

[36] C. Kutzner, D. Van Der Spoel, M. Fechner, E. Lindahl, U. W. Schmitt, B. L. De Groot, and H. Grubmüller, "Speeding up parallel GROMACS on high-latency networks," *Journal of Computational Chemistry*, vol. 28, no. 12, pp. 2075–2084, 2007.

[37] M. Turalija, "Towards General-Purpose Long-Timescale Molecular Dynamics Simulation on Exascale Supercomputers with Data Processing Units."

[38] J. P. Grossman, J. S. Kuskin, J. A. Bank, M. Theobald, R. O. Dror, D. J. Ierardi, R. H. Larson, U. B. Schafer, B. Towles, C. Young, and D. E. Shaw, "Hardware support for fine-grained event-driven computation in Anton 2," *ACM SIGPLAN Notices*, vol. 48, no. 4, pp. 549–560, 2013.

[39] J. P. Grossman, B. Towles, B. Greskamp, and D. E. Shaw, "Filtering, Reductions and Synchronization in the Anton 2 Network," in *2015 IEEE International Parallel and Distributed Processing Symposium*, 2015, pp. 860–870.

[40] D. E. Shaw, P. J. Adams, A. Azaria, J. A. Bank, B. Batson, A. Bell, M. Bergdorf, J. Bhatt, J. A. Butts, T. Correia, R. M. Dirks, R. O. Dror, M. P. Eastwood, B. Edwards, A. Even, P. Feldmann, M. Fenn, C. H. Fenton, A. Forte, J. Gagliardo, G. Gill, M. Gorlatova, B. Greskamp, J. P. Grossman, J. Gullingsrud, A. Harper, W. Hasenplaugh, M. Heily, B. C. Heshmat, J. Hunt, D. J. Ierardi, L. Iserovich, B. L. Jackson, N. P. Johnson, M. M. Kirk, J. L. Klepeis, J. S. Kuskin, K. M. Mackenzie, R. J. Mader, R. McGowen, A. McLaughlin, M. A. Moraes, M. H. Nasr, L. J. Nociolo, L. O'Donnell, A. Parker, J. L. Peticolas, G. Pocina, C. Predescu, T. Quan, J. K. Salmon, C. Schwink, K. S. Shim, N. Siddique, J. Spengler, T. Szalay, R. Tabladillo, R. Tartler, A. G. Taube, M. Theobald, B. Towles, W. Vick, S. C. Wang, M. Wazlowski, M. J. Weingarten, J. M. Williams, and K. A. Yuh, "Anton 3: Twenty microseconds of molecular dynamics simulation before lunch," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–11.

[41] G. Morimoto, Y. M. Koyama, H. Zhang, T. S. Komatsu, Y. Ohno, K. Nishida, I. Ohmura, H. Koyama, and M. Taiji, "Hardware acceleration of tensor-structured multilevel ewald summation method on MDGRAPE-4A, a special-purpose computer system for molecular dynamics simulations," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.

[42] I. Ohmura, G. Morimoto, Y. Ohno, A. Hasegawa, and M. Taiji, "MDGRAPE-4: A special-purpose computer system for molecular dynamics simulations," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 372, no. 2021, p. 20130387, Aug. 2014.

[43] W. Gropp and M. Snir, "Programming for Exascale Computers," *Computing in Science & Engineering*, vol. 15, no. 6, pp. 27–35, Nov. 2013.