Parallel Molecular Dynamics with a Time-Reversible Nosé-Hoover Thermostat on CPUs and GPUs

Nathan Mahynski George A. Khoury Carmeline Dsilva

January 5, 2014

1 Introduction

Due to the dramatic increase in computing power over the last thirty years, molecular dynamics (MD) simulations have become a remarkably fruitful tool for scientific discovery and engineering exploration [1, 2]. These simulations allow us to screen potential drug compounds [3], explore protein folding dynamics [4, 5, 6, 7], study [8] and develop [9] new catalysts, and explore extreme conditions (temperatures and pressures) that are experimentally inconvenient all without running expensive and time-consuming experiments in a laboratory. Early atomistic simulations were capable of running at only short physical timescales (picoseconds [10]), but due to enhancements in algorithms, hardware, and parallelization, physically relevant timescales (milliseconds [11]) can now be tractably explored on current computers.

At its core, molecular dynamics numerically integrates Newton's equation of motion ($force = mass \times acceleration$) forward in time. Different numerical integration schemes result in different numerical accuracies for the simulations. Standard molecular dynamics fixes the number of particles, volume, and total energy of the system. However, one can introduce thermostats and barostats to the integration algorithm to adjust the temperature and pressure of the system to attempt to keep them constant (See Figure 1).

Several molecular dynamics packages exist to perform parallelized atomistic molecular dynamics simulations on CPUs. These include LAMMPS [12], AMBER [13], CHARMM [14], GROMACS [15], and TINKER [16]. Only very re-

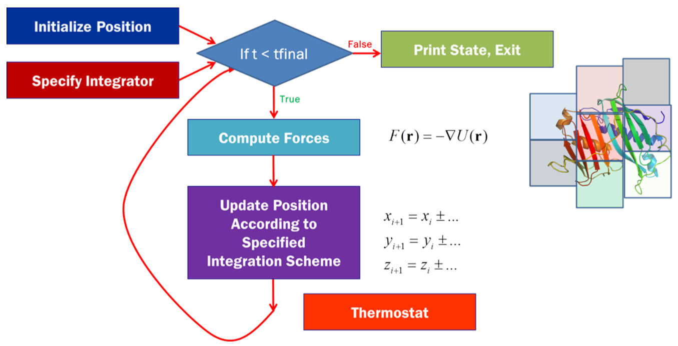

Initialize Position
Specify Integrator
If t < tfinal
False
Print State, Exit
True
Compute Forces
$F(\mathbf{r}) = -\nabla U(\mathbf{r})$
Update Position According to Specified Integration Scheme
$x_{i+1} = x_i \pm \ldots$
$y_{i+1} = y_i \pm \ldots$
$z_{i+1} = z_i \pm \ldots$
Thermostat

Figure 1: Molecular Dynamics algorithm graphical overview. MD begins with a system of particles coordinates and velocities, and integrates Newton's equation of motion forward in time utilizing a specified algorithm. A thermostat can be used to control the temperature through physics-based relationships between velocities and temperature. In this work, we implement the Nosé-Hoover Thermostat to control temperature.

cently, some of these packages have been updated in order to be implemented on Graphics Processing Units (GPUs) [17, 18, 19, 20], and new packages exclusive to GPUs such as HOOMD-Blue[21] have been developed. The newly updated codes have not been nearly as extensively tested as their native CPU codebases, but can potentially offer significant speedups `http://ambermd.org/gpus/`.

Therefore, in this APC523 Final Project we built a new software package from scratch to run molecular dynamics simulations in parallel on both CPUs and GPUs. The GPU implementation is an ambitious goal very much working on the cutting-edge of computational science since few and only recent implementations exist on GPUs [21, 17, 18, 19, 20]. The GPU implementation offers the promise of a significant speedup over CPU-only implementations. The software implements the Lennard-Jones potential for interatomic interactions, a velocity-Verlet integrator, and a Nosé-Hoover thermostat. The developed software is sufficiently modular that other integration algorithms and pair potentials can easily be incorporated in future versions. We chose to include the Lennard-Jones potential because many of the standard force fields [13, 14, 15, 22, 23, 24] today are built on Lennard-Jones interactions. For example, a commonly used model for water, the TIP3P water model [25], is composed of Lennard-Jones interactions and an additional electrostatic term to account for water's ability to be polarized. In addition, Lennard-Jones particles serve as a very good model for monotomic gases such as argon [26] and condensed amorphous phases such as glasses [27].

Because parallelization is essential for modern day molecular dynamics software, we implemented our software to run in parallel on both CPUs (with shared memory) and GPUs. Our code is therefore fairly flexible and applicable to many different hardware architectures.

2 Theory and Algorithms

In the following section, we describe the theory and algorithms that underly the software developed. In what follows, we assume that our system of interest consists of N particles. We denote the position, velocity, and acceleration of particle i as $\mathbf{r}_i$, $\mathbf{v}_i$, and $\mathbf{a}_i$, respectively. We denote the force on particle i as $\mathbf{F}_i$.

In general, the algorithm for a molecular dynamics simulation is as follows (See Figure 1):

1. Initialize particle positions $\mathbf{r}_i$ and velocities $\mathbf{v}_i$ within the simulation box.
2. Calculate the forces $\mathbf{F}_i$ on each particle.
3. Integrate forward the positions and velocities of each particle for the chosen timestep Δt.
4. Repeat steps 2-3 for the chosen amount of simulation time.

2.1 Integrator

We implemented a velocity-Verlet integration algorithm [28], which is accurate to fourth order. For a fixed number of particles (N), volume of the system (V), and energy (E), we update the positions and velocities of each atom using the following equations

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\Delta t + \mathbf{a}_i(t)\frac{\Delta t^2}{2} \quad (1)$$

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + [\mathbf{a}_i(t) + \mathbf{a}_i(t+\Delta t)]\frac{\Delta t}{2} \quad (2)$$

where $\mathbf{a}_i = \mathbf{F}_i/m$. The forces can be calculated from the potential energy function and will be discussed further in subsection 2.3.

Because $\mathbf{v}_i$ depends on both $\mathbf{a}_i(t)$ and $\mathbf{a}_i(t+\Delta t)$, the order of computations goes as follows

1. Calculate $\mathbf{r}_i(t+\Delta t)$ from $\mathbf{r}_i(t)$, $\mathbf{v}_i(t)$, and $\mathbf{a}_i(t)$.
2. Calculate $\mathbf{a}_i(t+\Delta t)$ from $\mathbf{r}_i(t+\Delta t)$.
3. Calculate $\mathbf{v}_i(t+\Delta t)$ from $\mathbf{v}_i(t)$, $\mathbf{a}_i(t)$, and $\mathbf{a}_i(t+\Delta t)$.

2.2 Thermostats

The equations in 2.1 are only for NVE simulations. If we are instead interested in fixing temperature (T) rather than energy (E), we must introduce the idea of a *thermostat* to our simulations. A thermostat is necessary in most simulations since experiments are not typically done at constant energy. The thermostat adjusts the velocities of the particles so that they (on average) maintain a user-specified desired temperature.

We implemented a time-reversible Nosé-Hoover thermostat in our molecular dynamics software. The Nosé-Hoover thermostat adjusts the particle velocities based on the target temperature of the system and the current kinetic energy of the system. The thermostat is governed by its own equation of motion given by the position (ξ), velocity ($\dot{\xi}$), and acceleration ($\ddot{\xi}$) of the thermostat.

Let T_{set} be the desired temperature. Then the acceleration of the thermostat is given by

$$\ddot{\xi} = \frac{1}{Q}\left[\sum_{i=1}^{N} m_i v_i^2 - N_f k_B T_{set}\right] \tag{3}$$

The equations of motion for the particles and the thermostat are

$$r_i(t+\Delta t) = r_i(t) + v_i(t)\Delta t + \left[a_i(t) - v_i(t)\dot{\xi}(t)\right]\frac{\Delta t^2}{2} \tag{4}$$

$$v_i(t+\Delta t) = v_i(t) + \left[a_i(t) - v_i(t)\dot{\xi}(t)\right]\frac{\Delta t}{2} + \left[a_i(t+\Delta t) - v_i(t+\Delta t)\dot{\xi}(t+\Delta t)\right]\frac{\Delta t}{2} \tag{5}$$

$$\xi(t+\Delta t) = \xi(t) + \dot{\xi}(t)\Delta t + \ddot{\xi}(t)\frac{\Delta t^2}{2} \tag{6}$$

$$\dot{\xi}(t+\Delta t) = \dot{\xi}(t) + \left[\ddot{\xi}(t) + \ddot{\xi}(t+\Delta t)\right]\frac{\Delta t}{2} \tag{7}$$

The Nosé-Hoover thermostat can be shown to be time-reversible when it is implemented via the following algorithm. First, update the thermostat velocity using a half-step in time

$$\dot{\xi}(t+\Delta t/2) = \dot{\xi}(t) + \ddot{\xi}(t)\frac{\Delta t}{2} \tag{8}$$

and thermostat position.

$$\xi(t+\Delta t) = \xi(t) + \dot{\xi}(t+\Delta t/2)\Delta t \tag{9}$$

Second, evolve the particle velocities with a half-step.

$$v_i(t+\Delta t/2) = v_i(t)\exp^{-\dot{\xi}(t+\Delta t/2)\Delta t/2} + a_i(t)\Delta t/2 \tag{10}$$

Third, evolve particle positions.

$$r_i(t+\Delta t) + r_i(t) + v_i(t+\Delta t/2)\Delta t \tag{11}$$

Fourth, call the function **calcForce** to update the accelerations at the next timestep.

$$a_i(t+\Delta t) = F_i(t+\Delta t)/m_i \tag{12}$$

Fifth, evolve the particle velocities.

$$v_i(t+\Delta t) = [v_i(t+\Delta t/2) + a_i(t+\Delta t)\Delta t/2]\exp^{-\dot{\xi}(t+\Delta t/2)\Delta t/2} \tag{13}$$

Finally, the thermostat velocity is updated.

$$\dot{\xi}(t+\Delta t) = \dot{\xi}(t+\Delta t/2) + \ddot{\xi}(t+\Delta t)\Delta t/2 \tag{14}$$

The thermostat acceleration, $\ddot{\eta}(t)$ is expressed in the form

$$\ddot{\xi}(t) = \frac{1}{\tau^2}[\frac{T(t)}{T_{target}} - 1] \tag{15}$$

where $\tau^2 = \frac{Q}{(3N-1)*T_{target}}$ and Q, which is the thermal mass of the thermostat spring equal to unity. Because this algorithm is time-reversible, this algorithm is implemented in our software.

This is an algorithmic advance discovered 20 years ago. The three factors in G are separate and unitary, therefore. This means that any integrator based on this Trotter factorization will be time-reversible. CITE Reversible multiple time scale molecular dynamics by Tuckermar Berne Martyna

2.3 Pair potentials

We implemented a shifted Lennard-Jones pair potential in our molecular dynamics software. In the shifted Lennard-Jones potential, the potential energy of interaction between two particles is a function of the interatomic distance $r = \|\mathbf{r}_i - \mathbf{r}_j\|$, and is given by

$$U(r) = 4\varepsilon \left[\left(\frac{\sigma}{r-\delta} \right)^{12} - \left(\frac{\sigma}{r-\delta} \right)^{6} \right] + U_{shift} \tag{16}$$

where ε, σ, δ, and U_{shift} are adjustable parameters. The total potential energy of the system is then given by

$$U_{tot} = \sum_{i=1}^{N} \sum_{j=1}^{i-1} U\left(\|r_i - r_j\|\right) \tag{17}$$

The force on a particle F_i is the gradient of the potential, $F_i = \nabla_i U_{tot}$. It can be shown that

$$F_{i,x} = \sum_{j \neq i} \frac{dU\left(\|r_i - r_j\|\right)}{d\|r_i - r_j\|} \frac{x_i - x_j}{\|r_i - r_j\|}. \tag{18}$$

The potential energy and force calculations involve $O(N^2)$ calculations, since each of the N particles interacts with the remaining $N-1$ particles. Therefore, the potential energy and force calculation are the portions that are typically parallelized in an MD code.

2.4 Parallelization

We implemented parallelization on CPUs and GPUs in our molecular dynamics software. On CPUs, we parallelized our code using OpenMP and therefore, our code can run in parallel on any shared memory cluster. On GPUs, we parallelized our code using CUDA. The parallelization on GPUs is implemented in the force calculation, which accounts for XXXXX percent of the cost of the simulation according to our profiling studies. Parallelization and scaling was tested on the Adroit and TIGER clusters.

2.5 Neighbor lists

2.6 Testing

We implemented Google Tests within our software.

We implemented the following tests:

`NumAtoms` verifies that the system class stores the correct number of atoms

`KineticEnergy` calculates the kinetic energy for a two-atom system, and verifies that it is correctly returned by the KinE() function

`PotentialEnergy` calculates the potential energy for a two-atom system, and verifies that it is correctly returned by the PotE() function

`ChangeBox` changes the length of the simulation box and verifies that the box is correctly changed in the system definition

`PBC` computes the distance between two particles using the minimum image convention and verifies that it is correctly computed for different scenarios (out of box, inside box, etc.)

3 Code Structure and Implementation Notes

The software is implemented in C++ and version-controlled through an actively updated github repository `https://github.com/PrincetonUniversity/CBEMDGPU`. The class structure is as follows

`common.h` This class implements the error catching functions.

`cudaHelper.h` This function implements the cuda code for paralleization on GPUs.

`cellList.cpp` This class controls the cell lists (for parallelization on CPU) or neighbor lists (for parallelization on GPUs).

`dataTypes.h` This function implements the custom structures, such as the atom struct, which stores the position, velocity, and acceleration of an atom.

`integrator.cpp` This class is a "virtual" class (maybe????)

`nve.cpp` This class implements the velocity-Verlet integration scheme for an NVE simulation.

`nvt.cpp` This class implements the velocity-Verlet integration scheme with a Nosé-Hoover thermostat for an NVT simulation.

`potential.cpp` The potential class implements the force and potential energy calculations for the standard Lennard-Jones potential. This class can be expanded to include other potential energy functions.

`system.cpp` This class stores all of the parameters relevant to the simulation system, such as the box size, potential energy function, and the number of atoms.

`utils.cpp` This class implements the necessary "helper" functions for the simulation, such as the distance calculation with periodic boundary conditions using the minimum image convention.

`main.cpp`

`unittests.cpp` This file implements Google Tests for our software. We used a test fixture to initialize the same system construct for multiple tests.

4 Results

4.1 Scaling studies

4.2 Benchmarking

We should discuss the fact that we performed optimizations while we were coding this.

4.3 Validation against literature values

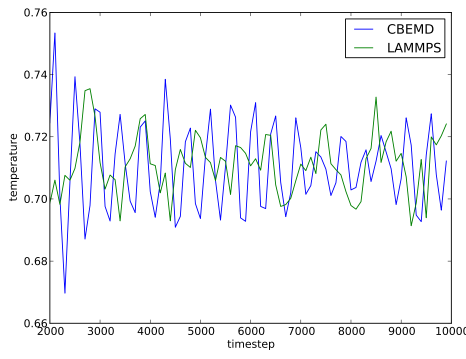
MSD g(r) Temp and Energy consistencies.

We validated our code against LAMMPS [29] (http://lammps.sandia.gov), an existing commercial molecular dynamics software. We simulated a system of 4000 Lennard-Jones particles at a reduced temperature $T = 0.71$. We used a cubic box with edge length $L = 16.796$, so that the reduced density of the system $\rho = 0.8442$. We used a timestep $\Delta t = 0.005$, and simulated for $10,000$ timesteps.

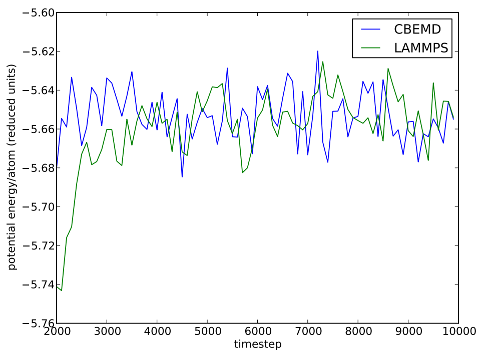
We compared the temperature, potential energy, total energy, and radial distribution function for the two simulations. The results are shown in Figure 2. There is excellent agreement between our software (CBEMD) and the LAMMPS results.

	CBEMD	LAMMPS
T	0.7104	0.7112
Potential energy / atom	-5.6532	-5.6594
Total energy / atom	-4.5879	-4.5929

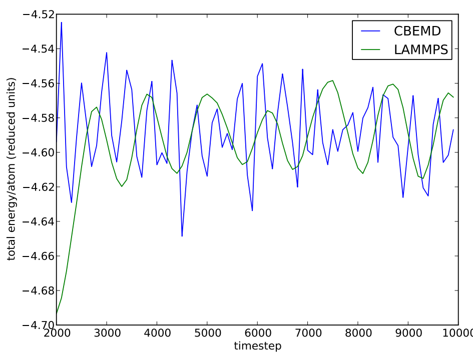
Table 1: Comparison of average thermodynamic quantities for CBEMD and LAMMPS simulations.



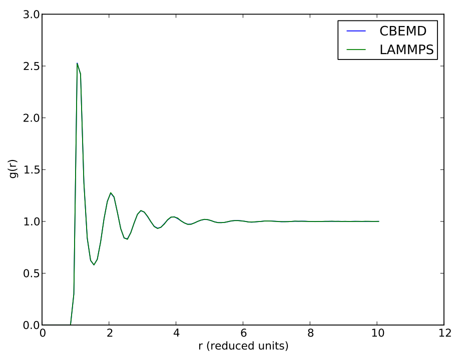
(a)



(b)



(c)



(d)

Figure 2: Comparison between CBEMD simulation and LAMMPS simulation for 4000 Lennard Jones particles at $T = 0.71$ and $\rho = 0.8442$. (a) Comparison of temperature over time. (b) Comparison of potential energy/atom over time. (c) Comparison of total energy/atom over time. (d) Comparison of radial distribution functions.

4.4 Image from VMD

We should attach an image from MD for a system of particles and we should also attach a small MD movie as a file viewable in Windows from an .xyz file for them to view and we should note that here.

5 Conclusions and Future Work

References

[1] Karplus M, McCammon JA. Molecular dynamics simulations of biomolecules Nature Structural & Molecular Biology 2002;9(9):646–652.

[2] Levitt M. The birth of computational structural biology Nature Structural & Molecular Biology 2001;8(5):392–393.

[3] Jorgensen WL. The many roles of computation in drug discovery. Science 2004;303(5665):1813–1818.

[4] Duan Y, Kollman PA. Pathways to a protein folding intermediate observed in a 1-microsecond simulation in aqueous solution Science 1998; 282(5389):740–744.

[5] Shaw DE, Maragakis P, Lindorff-Larsen K, Piana S, Dror RO, Eastwood MP, Bank JA, Jumper JM, Salmon JK, Shan Y, Wriggers W. Atomic-level characterization of the structural dynamics of proteins Science 2010; 330(6002):341–346.

[6] Piana S, Lindorff-Larsen K, Shaw DE. Atomic-level description of ubiquitin folding Proceedings of the National Academy of Sciences 2013;.

[7] Lindorff-Larsen K, Piana S, Dror RO, Shaw DE. How fast-folding proteins fold. Science 2011;334(6055):517–520.

[8] Boero M, Parrinello M, Terakura K. First principles molecular dynamics study of zieglernatta heterogeneous catalysis Journal of the American Chemical Society 1998;120(12):2746–2752.

[9] Zipoli F, Car R, Cohen MH, Selloni A. Theoretical design by first principles molecular dynamics of a bioinspired electrodecatalyst system for electrocatalytic hydrogen production from acidified water Journal of Chemical Theory and Computation 2010;6(11):3490–3502.

[10] Karplus M, McCammon JA. Protein structural fluctuations during a period of 100 ps Nature 1979;277(5697):578.

[11] Kohlhoff KJ, Shukla D, Lawrenz M, Bowman GR, Konerding DE, Belov D, Altman RB, Pande VS. Cloud-based simulations on google exacycle reveal ligand modulation of gpcr activation pathways. Nat Chem 2014;6(1):15–21.

[12] Plimpton S. Fast parallel algorithms for short-range molecular dynamics 1995.

[13] Case DA, Darden TA, Cheatham TE, Simmerling CL, Wang J, Duke RE, Luo R, Crowley M, Walker RC, Zhang W, Merz KM, Wang B, Hayik S, Roitberg A, Seabra G, Kolossváry I, Wong KF, Paesani F, Vanicek J, Wu X, Brozell SR, Steinbrecher T, Gohlke H, Yang L, Tan C, Mongan J, Hornak V, Cui G, Mathews DH, Seetin MG, Sagui C, Babin V, Kollman PA. Amber 11 University of California, San Francisco 2010.

[14] MacKerell AD, Bashford D, Bellott, Dunbrack RL, Evanseck JD, Field MJ, Fischer S, Gao J, Guo H, Ha S, Joseph-McCarthy D, Kuchnir L, Kuczera K, Lau FTK, Mattos C, Michnick S, Ngo T, Nguyen DT, Prodhom B, Reiher WE, Roux B, Schlenkrich M, Smith JC, Stote R, Straub J, Watanabe M, Wiorkiewicz-Kuczera J, Yin D, Karplus M. All-atom empirical potential for molecular modeling and dynamics studies of proteins J Phys Chem B 1998; 102(18):3586–3616.

[15] Scott WRP, Hunenberger PH, Tironi IG, Mark AE, Billeter SR, Fennen J, Torda AE, Huber T, Kruger P, van Gunsteren WF. The gromos biomolecular simulation program package J Phys Chem A 1999;103(19):3596–3607.

[16] TINKER. version 5.0 St. Louis, MO: Washington University 2010.

[17] Brown WM, Kohlmeyer A, Plimpton SJ, Tharrington AN. Implementing molecular dynamics on hybrid high performance computers–particle–particle particle-mesh Computer Physics Communications 2012; 183(3):449–459.

[18] Brown WM, Wang P, Plimpton SJ, Tharrington AN. Implementing molecular dynamics on hybrid high performance computers–short range forces Computer Physics Communications 2011;182(4):898–911.

[19] Gotz AW, Williamson MJ, Xu D, Poole D, Le Grand S, Walker RC. Routine microsecond molecular dynamics simulations with amber on gpus. 1. generalized born J Chem Theory Comput 2012;8(5):1542–1555.

[20] Salomon-Ferrer R, Gotz AW, Poole D, Le Grand S, Walker RC. Routine microsecond molecular dynamics simulations with amber on gpus. 2. explicit solvent particle mesh ewald Journal of Chemical Theory and Computation 2013;9(9):3878–3888.

[21] Anderson JA, Lorenz CD, Travesset A. General purpose molecular dynamics simulations fully implemented on graphics processing units Journal of Computational Physics 2008;227(10):5342–5359.

[22] Kaminski GA, Friesner RA, Tirado-Rives J, Jorgensen WL. Evaluation and reparametrization of the opls-aa force field for proteins via comparison with accurate quantum chemical calculations on peptides J Phys Chem B 2001; 105(28):6474–6487.

[23] Arnautova YA, Jagielska A, Scheraga HA. A new force field (ecepp-05) for peptides, proteins, and organic molecules J Phys Chem B 2006; 110(10):5025–5044.

[24] Khoury GA, Thompson JP, Smadbeck J, Kieslich CA, Floudas CA. Forcefield_ptm: Ab initio charge and amber forcefield parameters for frequently occurring post-translational modifications J Chem Theory Comput 2013; 9(12):5653–5674.

[25] Jorgensen WL, Chandrasekhar J, Madura JD, Impey RW, Klein ML. Comparison of simple potential functions for simulating liquid water The Journal of Chemical Physics 1983;79(2):926–935.

[26] Rahman A. Correlations in the motion of atoms in liquid argon Physical Review 1964;136(2A):A405.

[27] Debenedetti PG, Stillinger FH. Supercooled liquids and the glass transition Nature 2001;410(6825):259–267.

[28] Swope WC, Andersen HC, Berens PH, Wilson KR. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters The Journal of Chemical Physics 1982;76:637.

[29] Plimpton S. Fast parallel algorithms for short-range molecular dynamics Journal of Computational Physics 1995;117(1):1–19.