Cell Charge Approximation for Accelerating Molecular Simulation on CUDA-Enabled GPU

Shyam Mohan K T ,Jayaraj P B

Dept. of Computer Science and Engineering National Institute of Technology Calicut
Calicut, India

ABSTRACT

Methods for Molecular Dynamics(MD) simulations are investigated. MD simulation is the widely used computer simulation approach to study the properties of molecular system. Force calculation in MD is computationally intensive. Paral-lel programming techniques can be applied to improve those calculations. The major aim of this paper is to speed up the MD simulation calculations by/using General Purpose Graphics Processing Unit(GPU) computing paradigm, an efficient and economical way for parallel computing. For that we are proposing a method called cell charge approximation which treats the electrostatic interactions in MD simulations.This method reduces the complexity of force calculations.

Index Terms

Cell charge approximation, General Purpose Graphics Processing Unit (GPU), MD simulations.

I. INTRODUCTION

General Purpose Computation on Graphics Processing Units(GPUs) is a parallel computing paradigm which provides a cost effective solution for computationally ex-pensive applications. GPUs are massively parallel. Many core processors can be used to accelerate a wide range of data-parallel general purpose applications like molecular dynamics in addition to graphics processing. Also the power consumption of GPUs are very less compared to other high performance devices.

Molecular Dynamics(MD) is a widely used computer simulation technique. In molecular dynamics approach, through the integration of equations of motions, particle trajectories are calculated from the force exerted on each particle due to molecular interactions. The force on each particle is calculated as the sum of all forces from rest of the particles. But the force calculations are independent of each other so that parallel programming paradigm can improve the simulation process. The introduction of GPU computing made impact on techniques for molecular simulation. The use of GPGPU techniques as an alternative to distributed memory clusters in MD simulations has become a real possibility. The availability of large number of

processing cores makes GPU computing suitable here. CUDA architecture introduced by NVIDIA, which provides high level abstraction and minimal language extensions to implement parallel programs, significantly simplifies the process of programming the GPU. In this paper we are introducing an approximation technique, cell charge approximation, which reduces computation needed for long range coulombic forces in molecular dynamics.

The rest of the paper is organized as follows: Section II explains the background information and Section III describes the MD algorithm. Section IV presents the improving of simulation and Section V deals with imple-mentation and results, and finally the paper is concluded in Section VI.

II. BACKGROUND INFORMATION

A. Molecular Simulation

Molecular simulation is the technique for computing and visualizing the properties of molecules using com-puters, which serves as a complement to conventional experiments and enabling us to learn something new without doing the experiment[1]. Molecular simulation tools are helpful in predicting the behavior of molecules, chemical interactions and physical properties and mainly used in the field of Physics,Chemistry, Bioinformatics and Nanotechnology. Molecular Dynamics simulation is compu-tation intensive since the amount of atoms in the system is always tremendous and time step is always in femto second level, which means realistic simulation needs millions of steps [1]. Fortunately most part of these simulations shows high parallelism, so that parallel programming techniques can be applied to improve the performance of simulation process.

The most used approach for molecular simulation is Molecular Dynamics method. The molecular model in Molecular Dynamics is based on classical mechanics. In classical Molecular Dynamics, force calculations are based on Newton's laws [1]. Since this is an approximation used to reduce computation, cannot describe some behavior of molecular system such as breakage and formation of chemical bond. So as to have a perfect simulation of molecular system the calculations must be done on the basis of quantum mechanics, such simulations are called ab initio simulations [2].

Required properties of the molecular system are found from the particle trajectories during the simulation time, while particle trajectories are calculated from the force exerted on each particle due to molecular interactions. The forces are non-linear functions of distances between the particles in the system and are either long-range or short-range in nature. Short-range force model limits the range of influence of the particle in the system, thus reducing the computation needed. Since each particle interact with all other particles in the system, long range force calculation can be computationally costlier as it scales quadratically but gives higher quality simulation [1].

Parallelization of Molecular Dynamics simulation can be categorized into three types: Particle Decomposition, Force Decomposition and Spatial Decomposition [1], [2]. In particle decomposition method subset of atoms is assigned to a streaming multiprocessor in GPU. Spatial decompo-sition decomposes the whole system into different spatial region and assign each region into a streaming processor [3], [2]. Force decomposition is rarely used in molecular dynamics simulations. For molecular simulation in GPU the best decomposition method is spatial

decomposition [2]. The data reuse and less maintenance overhead makes it the best fit. Spatial decomposition needs only one neighbor list for a set of particles in the system so that subset of particles in the system shares the common data which reduces the data transform time and thus increases the performance [4].

B. CUDA

The introduction of General purpose GPUs made an impact on the parallel processing paradigm. The advantage of GPU over other parallel architectures are large amount of arithmetic capability, low cost and high throughput[5]. CUDA architecture introduced by NVIDIA in 2006 is a general purpose parallel computing architecture. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU) [6]. It allows the programmer to use the GPU as a com-puting device without explicitly mapping to the graphics API. CUDA provides a parallel programming model and instruction set architecture which makes the programming of GPU easier as it provides low level hardware access, avoiding the limitations imposed in fragment shaders.

GPU is a multi-core processor with several Stream-ing Multiprocessors(SMPs) where each SMP consists of streaming processors arranged as grid of blocks[5]. Tasks are specified as blocks of threads. In CUDA threads are created and scheduled by hardware which reduces the management overhead. Since CUDA limits the number of threads per block and SMP ,the number of threads required for a task is specified accordingly[6]. Synchro-nization of threads within a block is possible in CUDA architecture. CUDA-compatible GPU can be programmed using CUDA C which is similar to C and have some extension which supports parallel programming [5]. CUDA program is divided into host code and device code.The part with little or no data parallelism is implemented in host code and it runs on the CPU. The part that exhibit rich amount of data parallelism is implemented in the device code or kernel function and is executed on GPU.

GPGPU programming paradigm belongs to Single Program Multiple Data (SPMD) model, in which each thread executes the same program on different parts of data.The major bottleneck of GPGPU programming model is the transfer of data to device memory from the primary memory before invoking the kernel. Device memory hier-archy consists of a large but slower Graphics DDR(GDDR) memory, a global memory for all threads,smaller, faster on chip shared memory shared per block and local registers and memory for each registers[6].

III. THE MD ALGORITHM

This section focuses on the basic idea behind MD simulation. In static MD, an isolated system of fixed size is studied and hence its total energy is constant. The particles of the system interact through model potentials and positions of the molecules are obtained by solving the equations of motion for each molecule [4]. The basic MD algorithm is

Algorithm 1 MD Algorithm

Initialize simulation parameters

Set initial positions and velocities of particles step := 0

repeat

Compute potential and forces on each particle in the system

Integrate the equations of motion to find the new position and velocity of particles

Update position and velocity values of each particle Calculate required properties

Increment the time step

until *step < TotalStep*

A. Initialization

To start the simulation, a set of initial positions and velocities must be assigned to the particles. Commonly the simulation starts from the results obtained from previous simulation. Otherwise a set of positions and velocities are chosen such that the system is in thermal equilibrium. Positions are defined on crystalline lattice structure with some randomization and initial velocities are taken from a Maxwell-Boltzmann distribution at the desired temper-ature with zero total momentum. [7].

B. Force Calculation

Calculation of intermolecular forces is the most com-putationally intensive part of molecular dynamics [1]. The force on an atom can be calculated from the change in potential energy between its current position and its position a small distance away [7]. Potential on a particle arises from the interaction of the particle with other particles in the molecular system. The value of potential energy is calculated as a sum of bonded and a sum of external or non-bonded terms.
The bonded interaction describes the potential arose from bond stretch, angular movement and bond rotations in a molecule. The bond stretch interaction represents the energy required to stretch or compress a covalent bond and is modeled by

$$\square_{\square\square\square\square\ \square\square\square\square\square\square}(\square) = \frac{\square}{\square}\ \square_{\square}(\square - \square_{\square}) \qquad (\square)$$

New position and velocity of a particle in the system is found by integrating the equations of motion based on current position and force acting on the particle. Verlet scheme is an extensively used integration approach in MD simulations [7]. The new position and velocity of a particle is given by the following equations in Verlet scheme with r_0 being the bond-rest length.

The angular interaction is the energy required to bend a bond from its equilibrium angle θ_0 to an angle θ.

$$V_{angle}(\theta) = \frac{1}{2} K_a (\theta - \theta_0) \qquad (2)$$

The energy required to deform a planar group of atoms from its equilibrium angle ϖ_0, called dihedral tortion, is given by

$$V_{dih}(\varpi) = \frac{1}{2} K_\phi (1 + \cos(n\phi + \varpi_0) \qquad (3)$$

Non-bonded interactions act between atoms which are not linked by covalent bonds and are categorized into electrostatic and non-electrostatic interaction. The non-electrostatic Non-bonded interaction, commonly called as Vander Waals interaction, is the sum of the attractive or repulsive forces between particles other than those due to covalent bonds and electrostatic interaction. Lennard-Jones (L-J) potential is used for modeling the Vander Waals Jones (L-J) potential is used for modeling the Vander Waals action beyond a cut-off are neglected since the magnitude of interaction falls quickly as the distance between the particles increases [4], [8], [3]. The L-J potential is given by

$$V_{ij} = 4\varepsilon \left[\left(\frac{\sigma}{d_{ij}} \right)^{12} - \left(\frac{\sigma}{d_{ij}} \right)^{6} \right] \qquad (4)$$

Electrostatic interactions between charged particles are modeled by Coulomb's law. Long range electrostatic interactions cannot be neglected since it converges slowly and hence the computation is split into short range and long range interactions. Short range interactions are calculated directly using Coulomb's law while long range interactions are calculated separately using some approximation technique [9]. The electrostatic interaction between two charged particles is given by

$$V_{ij} = \frac{1}{4\pi\varepsilon} \quad \frac{q_1 q_2}{d_{ij}} \qquad (5)$$

Force acting on a particle is calculated as the negative gradient of potential with respect to the co-ordinates of the particle in the system [7].

$$F_i = - \bigtriangledown_i V_{total} \qquad (6)$$

C. Integration

New position and velocity of a particle in the system is found by integrating the equations of motion based on current position and force acting on the particle. Verlet scheme is an extensively used integration approach in MD simulations [7]. The new position and velocity of a particle is given by the following equations in Verlet scheme

$$r(t + \Delta t) = 2r(t) + r(t - \Delta t) + \frac{F_i}{m_i} \Delta t^2 \qquad (7)$$

$$v(t) = \frac{r(t + \Delta t) - r(t - \Delta t)}{2\,\Delta t} \qquad (8)$$

IV. IMPROVING THE SIMULATION

A. Bonded interaction

The number of particles in the molecular system is always large and the bonded interaction on each particle, is independent of the interaction on other particles, in the system calculations can be parallelized. Particle decomposition method is used in parallelizing the bonded interaction. A thread is assigned for each particle in the system for the bonded interaction calculation.

B. Vander Waals Interaction

The large exponential values in Equation 4 reflects how the magnitude of these forces drop off very rapidly at larger distances, so the effect of Vander Waals forces beyond a cut-off radius (rc) can be neglected[1]. Based on this, to improve the Vander Waals force calculation,Neighbor lists are used. A neighbor list is associated with each particle and only the particles in the neighbor list are considered during the interaction calculation. Complexity of the Vander Waals force calculation drops from $O(n^2)$ to O(n), since only the particles in the neighbor list are to be considered and the number of particle in the neighbor list of a particle is very much less compared to the number of particle in the system, n. The overhead of this optimization is the construction and updation of neighbor lists which grows to O(n2), but the updation is done less frequently. Construction of neighbor list is based on the distance between the particles,if the distance between two particles is less than a specified neighbor distance then those two particles are neighbors.

The neighbor distance is chosen such that the neighbor list of a particle will contain all the particles that can be in the cut-off region; till the neighbor list is updated again after a number of time steps. Cell lists are used to reduce the computation needed in constructing the neighbor list to O(n) [1]. The cubic simulation box is divided into a regular lattice of smaller cells of size neighbor distance. The particles in each cells are stored in cell lists. The advantage of cell isthat it reduces the number of particles needed to be considered while constructing the neighbor list, since only the particles in current cell and its neighboring cells need to be considered. The cell list construction got only a linear time complexity. Neighbor list construction is parallelized based on particle decomposition approach. Neighbor lists of all particle can be constructed in parallel using a CUDA thread for each particle, so that the construction can be completed in O(n) in direct method and in O(1) in improved method. Spatial decomposition technique is used for parallelizing Vander Waals interaction calculation, where a thread is assigned for a set of neighboring particles. Since the nearest neighbors have the same set of neighbors, data reuse is possible here and the shared memory facility available in CUDA can be exploited. The common neighbor set is moved to the faster shared memory of the SMP, and the SMPs calculate the interaction on each particle in that set, which reduces the running time by reducing the number of expensive global memory accesses and results in a constant time complexity.

C. Electrostatic Interaction

Long range electrostatic interaction cannot be neglected as in the case of Vander Waals interaction. To reduce the time complexity from $O(n^2)$ some approximation techniques are used.

1)Cell Charge Approximation:

To reduce the time required for long range electrostatic interaction computation, cell charge approximation is introduced. The cubic simulation box is divided into a regular lattice of smaller cells of size greater than cut-off distance. Cell charge approximation treats each cell as a point charge located at the centroid of the cell with a charge that is equal to the sum of charges of the charged particle present in that cell. During the construction of cell list, charge density of each cell is computed and stored. In the computation of long range interaction, the interaction between the particle and the particles present in the immediate neighboring cells are calculated directly. Then the interaction with particles in the non-neighboring cells are computed based on cell charges, as the interaction between the particle and the virtual particle existing at the centroid of each cell , which reduces the time complexity of computation of long range electrostatic interaction in a system of n particle from $O(n^2)$ to O(nm) where m << n is the number of cells in the lattice. Particle decomposition is chosen to parallelize the electrostatic interaction in case of direct computation. Parallelizing cell charge approximation is done with spatial decomposition. Each SMP will have the details of virtual particles in each cell centroid with the common neighborset for a set neighboring particles in its shared memory and a thread is assigned to a sub set of particles, which reduces the number of global memory fetch and hence reduces the running time.

2) Particle Mesh Ewald:

Ewald Summation is a technique introduced for the calculation of long range interaction between particles by Peter Ewald in 1921[10]. In Ewald Summation the potential energy of a system of particles is calculated as sum of two rapidly converging terms namely direct sum and reciprocal sum and the calculation grows quadratically [11]. Particle Mesh Ewald(PME) is an improved Ewald Summation technique with complexity O(nlogn), is used in the long range electrostatic interaction calculation. The Particle Mesh Ewald method calculates direct-space interactions within a finite distance using a modification of Coulomb's Law, and in reciprocal space using a Fourier transform to build a "mesh" of charges, interpolated onto a grid. In PME, the potential energy is given by

$$V(r_i) = V_{direct}(r_i) + V_{self}(r_i) + V_{reciprocal}(r_i) \qquad (9)$$

The direct part is computed using the equation,

$$V_{direct}(r_i) = \sum_{j'} q_j \frac{efrc(\eta|r_i - r_j|)}{|r_i r - r_j|}. \qquad (10)$$

Only the particles inside the cut-off region are considered during the computation. The self interaction is given by

$$V_{self}(r_i) = -\frac{2\eta}{\sqrt{\pi}} q_i \qquad (11)$$

The reciprocal part is computed by the convolutions performed using Fast Fourier Transforms on a mesh(grid) where the charges are interpolated to the grid points [11]. First the charges are replaced by grid based charge density. This is done by interpolation of charges into grid points using an assignment functionW(r),

$$q_{r_n} = \sum_i q_i W(r_n - r_i) \qquad (12)$$

Then Fourier coefficient of the mesh-based charge density is computed as

$$\hat{\rho_{k_n}} = \frac{1}{Vol} \sum_{r_n} e^{-ik_n . r_n} q_{r_n}. \qquad (13)$$

The potential in the Fourier space is then computed as

$$\hat{V}_{k_n} = \sum_{k_n \neq 0} \hat{G'}_{k_n} \hat{\rho}_{k_n}. \qquad (14)$$

with

$$\hat{G'}_k = \left(4\pi / |k|^2\right) e^{-|k|^2/4\eta^2}. \qquad (15)$$

Then compute the potential on the real-space,

$$V_{r_n} = \sum_{k_n} e^{ik_n . r_n} \hat{V}_{k_n}. \qquad (16)$$

Potential on each particle is computed by interpolating back to the location of charged particles using the assignment function,

$$V_{reciprocal}(r_i) = = \sum_{r_n} V_{r_n} W(r_n - r_i) \qquad (17)$$

Even though PME gives higher performance, parellization of PME yields less speed up since some part of the simulation, the charge assignment, cannot be parallelized. The charge assignment is done serially and other part of the PME are parallelized.

3) PME with Cell Charge Approximation:

Combining cell charge approximation with PME gives higher performance. Particle Mesh Ewald method calculates direct-space interactions within a finite distance using a modification of Coulomb's Law, and in reciprocal space using a Fourier transform to build a "mesh" of charges,

interpolated onto a grid and hence gives a time complexity of O(nlogn). The cell charge approximation is combined with PME such that the virtual particle existing at the centroid of each cell is used as the charged particle in PME approach, hence improving the long range electrostatic interaction calculation of MD simulation. Dividing the simulation box into cells comes up with an advantage of reducing the computation needed in neighbor list construction, by reducing the number of particle need to be considered to the particles present in the current cell and its immediate neighboring cells

V. IMPLEMENTATION AND RESULTS

Any existing package would simply pose too many restrictions on the underlying data structures and the order in which operations are performed. Instead, a completely fresh MD code was built. The Serial molecular dynamics algorithm was implemented in C and the parallel versions were implemented using CUDA C language extensions. The program that implements Parallel MD algorithm uses serial version as the basis. The basic serial version is implemented on the basis of core MD algorithm given by Shrinidhi et al. [4].

The running time taken by different implementations for different number of atoms are given in Table I and Table II. Figure 1 and Figure 2 compares the speed enhancements of different implementations on an input of 3000 atoms and 6000 atoms respectively. It shows how different improvements reduces the running time of MD simulation. The approximations used in electrostatic interaction computations reduces the running time tremendously. The overhead of constructing the neighbor lists reduces the performance gain achieved through neglecting the long range Vander Waals interaction. Since the amount of parallelization portion available reduces and the copy overhead remains the same, reduction in running time of parallel implementation due to approximation methods is less compared to the serial version. The accuracy of simulation result obtained is compared with the result obtained from LAMMPS. The result obtained was similar with some minor errors. Figure 3 gives the comparison of speedup obtained from parallel implementation when compared with the serial version. The speed up is maximum in direct implementation and least in the final implementation since the amount of parallelization portion reduces as new approximations are introduced and the GPU overhead remains as it is.

Implementation Method	Pentium (s)	Corei3 (s)	GPU (s)
Direct Computation	762.660	172.405	83.925
With Neighbor List	492.404	106.519	65.734
With Cell	451.027	104.495	63.382
Cell Charge	71.699	33.031	24.788
PME	64.446	25.487	21.706
PME + Cell Charge	31.430	19.461	16.016

Table 1 : Running Time Of Different Implements (3000 Atoms)

Implementation Method	Pentium (s)	Corei3 (s)	GPU (s)
Direct Computation	2996.872	657.038	320.023
With Neighbor List	2006.155	394.923	245.290
With Cell	1889.830	383.617	241.618
Cell Charge	492.545	179.775	108.924
PME	212.091	47.644	36.774
PME + Cell Charge	84.330	41.992	23.671

Table 2: Running Time Of Different Implementations (3000 Atoms)

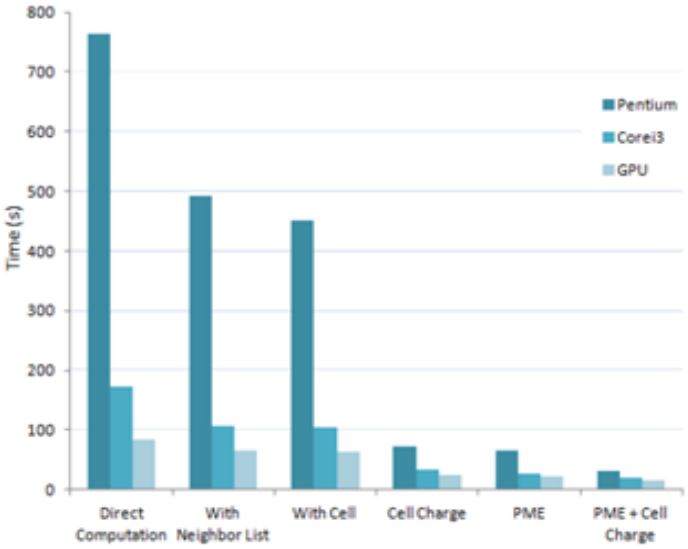

Fig 1: Performance Comparison (3000 atoms)

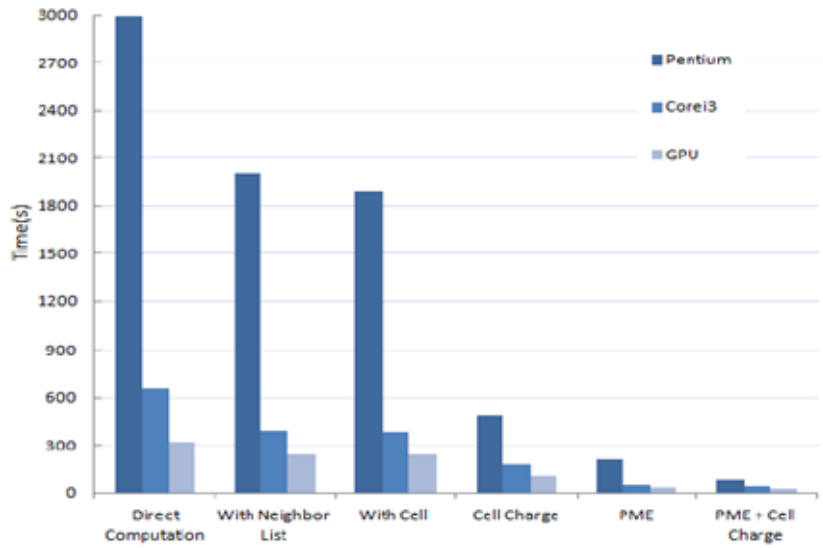

Fig 2: Performance Comparison(6000 atoms)

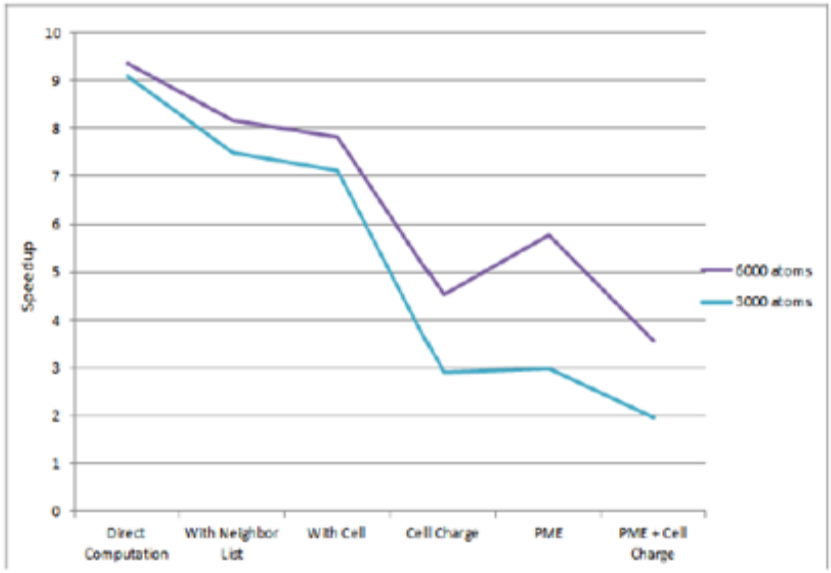

Fig 3: Speed up on GPU

VI.CONCLUSION

Molecular Dynamics is a widely used molecular simulation approach. Reduction in time complexity of MD simulation will allow us to simulate larger molecular systems within practically feasible time period. The work proposes cell charge approximation technique for reducing the computation needed for long range electrostatic interaction calculation in Molecular Dynamics simulation. The comparative analysis of proposed approach is done with other schemes

for calculating long range interaction. The results obtained indicate that the Cell Charge approximation used here provides good speedup in force calculation with out degrading the accuracy much.

REFERENCES

[1] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. volume 117, pages 1 – 19, 1995.

[2] Hongjian Li, Shixin Sun, Hong Tang, Yusheng Dou, and GlennV. Lo. Two-level parallelization of ehrenfest force calculations in ab-initio molecular dynamics simulation,. volume 15, pages 255–263. Springer US, 2012.

[3] Weiguo Liu, Bertil Schmidt, Gerrit Voss, and Wolfgang Muller-Wittig. Molecular dynamics simulations on commodity gpus with cuda. In Proceedings of the 14th international conference on High performance computing, HiPC'07, pages 185–196, Berlin, Heidelberg, 2007. Springer-Verlag.

[4] Shrinidhi Hudli, Shrihari Hudli, Raghu Hudli, Yashonath Subramanian, and T. S. Mohan. Gpgpu-based parallel computation: application to molecular dynamics problems. In R. K. Shyamasundar and Lokendra Shastri, editors, Bangalore Compute Conf., page 10. ACM, 2011.

[5] NVIDIA Corporation. NVIDIA CUDA C programming guide, 2010. Version 3.2.

[6] Jason Sanders and Edward Kandrot. CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison- Wesley Professional, 1st edition, 2010.

[7] D. C. Rapaport. The Art of Molecular Dynamics Simulation. Cambridge University Press, New York, NY, USA, 1996.

[8] M. Taufer, N. Ganesan, and S. Patel. Gpu enabled macromolecular simulation: Challenges and opportunities. volume PP, page 1, 2012.

[9] Christopher I. Rodrigues, David J. Hardy, John E. Stone, Klaus Schulten, and Wen-Mei W.Hwu. Gpu acceleration of cutoff pair potentials for molecular modeling applications. In Proceedings of the 5th conference on Computing frontiers, CF '08, pages 273–282, New York, NY, USA, 2008. ACM.

[10] P. P. Ewald. Die berechnung optischer und elektrostatischer gitterpotentiale. Annalen der Physik, 369(3):253–287, 1921.

[11] Harry A Stern and Keith G Calkins. On mesh-based ewald methods: optimal parameters for two differentiation schemes. J Chem Phys, 128(21):214106, 2008.

[12] Canqun Yang, Qiang Wu, Juan Chen, and Zhen Ge. Gpu acceleration of high-speed collision molecular dynamics simulation. In CIT (2), pages 254–259. IEEE Computer Society, 2009.

[13] Weiguo Liu, Bertil Schmidt, and Wolfgang Muller-Wittig. Cudablastp: Accelerating blastp on cuda-enabled graphics hardware. volume 8, pages 1678–1684, Los Alamitos, CA, USA, 2011. IEEE Computer Society.

[14] Marco Maggioni, Marco D. Santambrogio, and Jie Liang. Gpuaccelerated chemical similarity assessment for large scale databases. volume 4, pages 2007–2016, 2011.