ACS OMEGA

http://pubs.acs.org/journal/acsodf

Review

Review of Electrostatic Force Calculation Methods and Their Acceleration in Molecular Dynamics Packages Using Graphics Processors

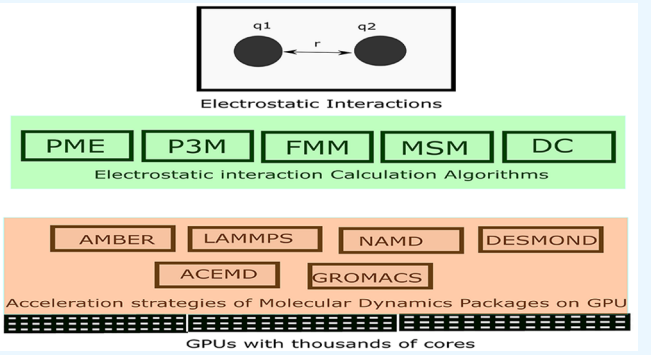
Anu George,* Sandip Mondal, Madhura Purnaprajna, and Prashanth Athri

Cite This: *ACS Omega* 2022, 7, 32877−32896

Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: Molecular dynamics (MD) simulations probe the conformational repertoire of macromolecular systems using Newtonian dynamic equations. The time scales of MD simulations allow the exploration of biologically relevant phenomena and can elucidate spatial and temporal properties of the building blocks of life, such as deoxyribonucleic acid (DNA) and protein, across microsecond (μs) time scales using femtosecond (fs) time steps. A principal bottleneck toward extending MD calculations to larger time scales is the long-range electrostatic force measuring component of the naive nonbonded force computation algorithm, which scales with a complexity of $O(N^2)$ (N, number of atoms). In this review, we present various methods to determine electrostatic interactions in often-used open-source MD packages as well as the implementation details that facilitate acceleration of the electrostatic interaction calculation.



1. INTRODUCTION

Molecular dynamics (MD) simulations[1−5] are widely used to investigate spatial and temporal fluctuations of biomolecules. MD simulations of processes such as protein−drug interactions[6−9] and folding/unfolding of proteins in the explicit solvent[10−14] require micro- to millisecond length simulations to provide the chemical and biological inferences that researchers commonly probe. The number of iterations or time steps can range from tens to hundreds of millions. This high sampling requirement is attributed to the high-frequency motion of bond-stretch variation.[15] Bond constraint algorithms such as SHAKE[16] and atomic mass scaling of the atoms in the system[17] have been used to decrease the sampling rate. Nonetheless, the time steps in these simulations are in the range of a couple of femtoseconds. The compute-intensive stages involved in each time step (summarized below), compounded by the high sampling rate requirement, result in simulations that extend from hours to months (wall time).[18]

In each iteration or time step of the MD simulations, the force on the system is calculated as the sum of bonded and nonbonded components.[1] Hence, the choice of the energy function is an important decision toward providing a stable MD simulation package. In general, most MD software packages (see Section 3) use an additive form of individual functions mapped to specific empirically calculated parameters to express a particular form of interaction that contributes to the instantaneous total potential energy of the system. A defined combination of such functions and their associated parameters is known as a force field. The force field used in most MD packages is shown in eq 1. It is the sum of intramolecular (V_{intra}) and intermolecular (V_{inter}) contributions. The V_{intra} term comprises eqs 1a through 1d, and the functions represent bond (V_{bond}), angle (V_{angle}), and dihedral or torsional angle (V_{tors}), and $V_{1,5}$ is the V_{inter} contribution. V_{inter} is the sum of long-range electrostatic (V_{el}) and short-range van der Waals (V_{vdw}) interactions (eqs 1e and 1f). Thus, the total potential energy can be written as

$$E_p = V_{intra} + V_{inter} = V_{bond} + V_{angle} + V_{tors} + V_{imp} + V_{1,5} + V_{vdw} + V_{el} \quad (1)$$

$$V_{bond} = \sum_{bonds} k_b(b - b_0)^2 \quad (1a)$$

Received: May 22, 2022
Accepted: August 26, 2022
Published: September 8, 2022

ACS Publications

© 2022 The Authors. Published by American Chemical Society

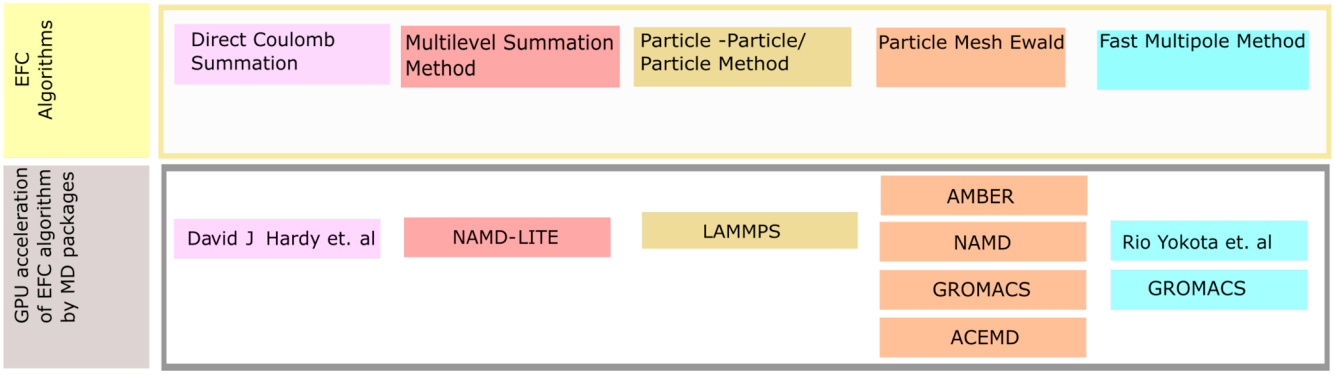


Figure 1. Different electrostatic force calculations (EFCs) are explored in Section 2 of the review. In Section 3, different techniques for the acceleration of the EFC by the GPU are studied in MD packages along with solo GPU implementations of the algorithm. The yellow-colored box encloses the different electrostatic force calculation algorithms, whereas the gray box encloses the MD packages and solo algorithmic implementations, respectively. The color code used for implementation is the same as the fill color of the box of the algorithm name.

$$V_{\text{angle}} = \sum_{\text{angles}} k_\theta(\theta - \theta_0)^2 \tag{1b}$$

$$V_{\text{tors}} = \sum_{\text{dihedral}} k_\phi[1 + \cos(n\phi - \phi_0)] \tag{1c}$$

$$V_{\text{imp}} = \sum_{\text{improper}} k_\omega(\omega - \omega_0)^2 \tag{1d}$$

$$V_{\text{vdw}} = \sum_{i<j} 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{1e}$$

$$V_{\text{el}} = \sum_{i<j} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} = \sum_{i<j} \frac{q_i q_j L(r_i, r_j)}{4\pi\epsilon_0} \tag{1f}$$

Here, k_b, k_θ, k_ϕ, and k_ω are the bond, angle, dihedral angle, and Urey–Bradley force constants, respectively. $(b - b_0)$, $(\theta - \theta_0)$, and $(\omega - \omega_0)$ are the difference between the instantaneous values of the specific atoms and the reference (equilibrium) values of bond, angle, and improper angles, respectively. Dihedral angles are represented as a combination of Cosine functions, and in eq 1c, n is the multiplicity and ϕ_0 the phase shift. The $\sum_{i<j}$ notation indicates that the summation is carried out over every distinct pair of i and j without counting any pair twice. q_i is the partial charge on the ith atom, and ϵ_0 is the absolute dielectric permittivity of the vacuum. ϵ_{ij} and σ_{ij} are the Lennard-Jones parameters for the van der Waals interaction between two interacting sites, i and j, while $r_{ij} = \|r_j - r_i\|$ is the distance between those sites and $L(r_i, r_j) = 1/\|r_i - r_j\|$.

The bonded forces determine the molecular conformation. It accounts for the potential energy contribution from the covalent bonds, angles, and torsion angles. Further, it is observed that their influence drops off steeply after a reasonably chosen cutoff radius.[2] The van der Waals potential between two atoms also rapidly approaches zero since it is inversely proportional to r^6, where r is the distance between two atoms.[19] However, in the case of the electrostatic force between charged atoms, it does not cease to exist even when the distance between them is considerable.[20,21] Due to this, the cutoff distance for electrostatic calculations is much larger than that for van der Waals in nonbonded force calculations. Further, the computational complexity ($O(N^2)$) increases with an increasing number of atoms, N. Thus, the electrostatic forces in nonbonded force calculations are the primary bottleneck in MD simulations.[19,22]

In this review, the implementation of electrostatic potential calculation algorithms (eq 1f) in the most widely used simulation packages, namely, AMBER,[23] NAMD,[24] GROMACS,[25] LAMMPS,[26] DESMOND,[18] and ACEMD,[27] is explored. These MD packages are able to achieve very long time-scale MDs in the micro- to millisecond ranges through the use of GPUs (graphics processing units).

The organization of the paper is as shown in Figure 1, and the objectives are the following:

- summarize the available methods of long-range electrostatic or Coulomb force calculations with respect to accuracy, complexity, and scalability (Section 2);
- review the software and GPU-enabled technologies currently used by popular MD;
- packages to accelerate electrostatic calculations (Section 3).

Algorithms used to reduce this complexity can be categorized as Ewald-based and non-Ewald-based calculations. In algorithms that use the Ewald-based method, the slowly converging force (given in eq 1f) is split into real space contribution and reciprocal space contribution along with self-term, which is associated with error correction. This splitting into components makes the computation much faster.[28−30] The non-Ewald algorithms, such as the Wolf method, reaction field method, preaveraging method, zero-dipole summation methods, etc., are based on the assumption that the effect of long-range interactions is negligible beyond the cutoff radius for an electrostatically neutral system.[30] The Ewald-based algorithms are the particle mesh Ewald (PME)[31] (reviewed in Section 2.3) and the particle−particle/particle mesh Ewald (P^3M)[31,32] methods (Section 2.4). The other variants of electrostatic force calculation algorithms discussed in this review are the direct Coulomb summation (DCS)[33] (Section 2.1), multilevel summation method (MSM)[34] (Section 2.2), and fast multipole method (FMM)[35,36] (Section 2.5).

2. DIFFERENT METHODS OF COULOMBIC INTERACTION FORCE CALCULATIONS

Electrostatic interactions are one of the main forces that contribute to determining the structure of biomolecules.[37] They play a significant role in modeling the dynamics of

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

charged molecules of proteins, DNA, and membranes in explicit solvent simulations.[38,39]

The main challenges in electrostatic potential computation (eq 1f) are the slow convergence of the electrostatic potential of atoms with distance and the need to avoid artifacts introduced in the potential calculation. These artifacts are due to the boundary conditions as the surface molecules experience different forces from other molecules present.[29,40] While the primary objective of this study is to provide a survey of methods implemented to address the slow convergence issue, it should be noted that periodic boundary conditions (PBCs) are used to address surface effects. The central cell containing the simulated system is replicated in three dimensions to form an infinite lattice. During the course of the simulation, when a molecule moves in the central original box, its periodic image in each of the adjacent boxes moves in exactly the same way. Therefore, whenever a molecule departs the central box, one of its images enters the opposite face. There are no walls at the central box boundary and no surface molecules in the PBC scheme, and hence errors introduced due to boundary surface effects are negligible.

The convergence bottlenecks exist since electrostatic potential does not reduce to zero even for vast distances,[41,42] and long-range methods are used to address this. The effect of electrostatic interactions for long ranges is achieved by the application of periodic boundary conditions (PBCs). The PBC method considers the interactions between direct particles, which are contained in the central simulation box, as well as their mirror images in the replicated boxes.[2,43] This yields a more accurate value for the potential and, in turn, for the forces acting on the atoms. Hence, the simulation of biomolecules using long-range methods results in a more accurate *in silico* reproduction of the experimental results.[44] However, the computation-intensive nature of this method becomes the bottleneck for scaling larger time scales.[45,46]

There are several long-range-based algorithms available to handle Coulombic interactions. Among them, the most common algorithms used for Coulombic computation in explicit solvent MD are

- direct Coulomb summation (DCS)[33] (Section 2.1)
- multilevel summation method (MSM)[34] (Section 2.2)
- particle mesh Ewald (PME)[31] (Section 2.3)
- particle−particle/particle mesh Ewald (P^3 M)[31,32] (Section 2.4)
- fast multipole method (FMM)[35,36] (Section 2.5)

These algorithms are explained and compared in terms of their accuracy, computational complexity, and scalability. The algorithmic variations of the Coulombic force calculations result in differences in accuracy, computational complexity, and scalability. The accuracy is measured with respect to the root-mean-square (rms) error in terms of force. The complexity depends on the number of particles in the system simulated and the methods used, such as fast Fourier transform (FFT), etc. The scalability of the above long-range-based algorithms is determined by the volume of intercommunication between different functional modules in the algorithm. The scalability increases as the amount of communication in the algorithm decreases. Highly scalable algorithms provide increased performance by utilizing more computational resources. The accuracy, computational complexity, and achievable scalability are the factors that determine the suitability of an algorithm for use in molecular dynamics (MD) simulation.

2.1. Direct Coulomb Summation. It is a lattice-based method in which the coordinates of atoms and their partial charges are taken into account for the computation of electrostatic potential.[33] A rectangular lattice is defined around the atoms with a fixed lattice spacing in all three dimensions. The electrostatic potential contribution, $V(i)$, at each point i in the lattice is calculated as

$$V(i) = \sum_j \frac{q_j}{4\pi\epsilon_0\epsilon(r_{ij})r_{ij}} \tag{2}$$

where q_j is charge of atom j; ϵ_0 is the absolute dielectric permittivity of the vacuum; $\epsilon(r_{ij})$ is the dielectric constant that varies with the distance between the lattice point i and atomic charge q_j;[33] and r_{ij} is the pairwise distance between the atom j and the lattice point i as shown in Figure 2.

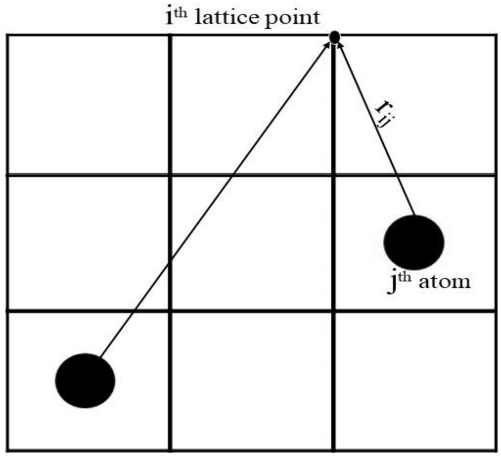


Figure 2. Electrostatic potential calculation in direct Coulomb summation.[33]

The potential map that is calculated by using the rectangular lattice can be subdivided into planes and rows. The components of distance calculation ($z-$, $y-$ coordinates) for an atom to points on the same row and plane are constant. Hence, calculation of the potential energy contribution of an atom to all the points in a row of the lattice can reuse the distance component.[47] This reuse of the distance component achieves increased performance as memory accesses and computations are reduced for the computation of the potential of each atom.[48]

Accuracy. This method does not involve any approximations and hence is used as the benchmark to compare the force calculation methods for accuracy.[47]

Complexity. The complexity of the method for a system composed of N atoms and Q points in the lattice is $O(NQ)$.

Scalability. The scalability of the algorithm is nonlinear due to the quadratic complexity of the summation algorithm.

2.2. Multilevel Summation Method. The significant latency of convergence in the direct Coulomb summation method makes it an unsuitable choice for large molecular systems with more than a few hundred atoms. Multilevel summation is one of the fastest potential approximation methods that can be used for periodic, semiperiodic, and nonperiodic systems.[34,49−51] The steps involved in the multilevel summation algorithm are shown in Figure 3.

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

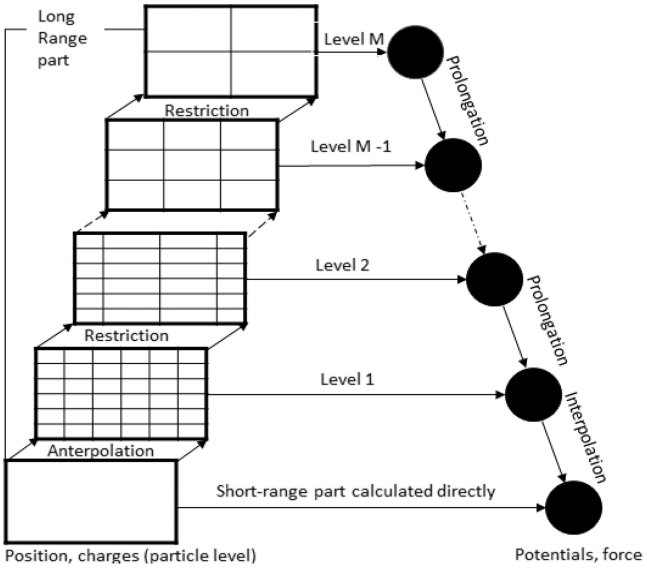


Figure 3. Workflow involved in the multilevel summation algorithm. Adapted with permission from ref 52. Copyright 2014 American Institute of Physics.

In this method, the Coulombic potential (given in eq 1f) for the atomic partial charges on the lattice defined around the biomolecule is calculated by hierarchical interpolation. Hierarchical interpolation and restriction are the two actions performed to estimate the charge at each lattice point at different levels of the grid. The charge at each grid point is due to all the atomic charges in the biomolecule. The potential of the system is split into two parts. The first component is the short-range part that disappears after the cutoff distance a. The long-range part is (v^{long}). It is obtained by excluding the force involved in the interaction of the atoms with itself in the mirror images (v^{ex}) as shown in eq 3.

$$V \approx V^{\text{msm}} = \frac{1}{2}\sum_{1=1}^{N}\sum_{j\neq i}^{N} q_i q_j L_0(r_i, r_j) + \frac{1}{2}\sum_{i=1}^{N} q_i v_i^{\text{long}} - v^{\text{ex}}$$

$$v^{\text{ex}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{i=j} q_i q_j \frac{1}{a}\gamma\left(\frac{|r_j - r_i|}{a}\right)$$

$$v_i^{\text{long}} = \sum_{j=1}^{N} I_1(L_1 + \ldots I_{m-1}(L_{m-1} + I_M(L_M)\ldots))(r_i, r_j)q_j \tag{3}$$

The long-range term is mapped to a grid where the lattice spacing doubles for every successive level. This allows the use of different time steps for the calculation of each term in the potential and hence helps in improving performance. The term $L(r_i, r_j)$ in V_{el} (given in eq 1f) is split into the sum of short-range and long-range interactions. The short-range interaction L_0 is truncated at distance a. The slowly varying long-range part is split into $L_1, L_2, L_3, \cdots, L_{M-1}, L_M$, which are clipped off at distances $2a, 4a, 8a, \cdots, 2^{k-1}a, \infty$.

$$L(r, r') = L_0(r, r') + L_1(r, r') + \cdots + L_M(r, r') \tag{4}$$

where $L_0(r, r')$ and $L_M(r, r')$ are given as in[53]

$$L_0(r, r') = \frac{1}{|r - r'|} - \frac{1}{a}\gamma\left(\frac{|r' - r|}{a}\right)$$

$$L_m(r, r') = \frac{1}{2^{m-1}a}\gamma\left(\frac{|r' - r|}{2^{m-1}a}\right) - \frac{1}{2^m a}\gamma\left(\frac{|r' - r|}{2^m a}\right)$$

$$L_M(r, r') = \frac{1}{2^{M-1}a}\gamma\left(\frac{|r' - r|}{2^{M-1}a}\right) \tag{5}$$

where r, r' is the position vector of the i^{th} and j^{th} atom; $\gamma(R) = 1/R$ for $R \geq 1$ is a softening function; and R in the softening function is $\frac{|r' - r|}{2^k a}$ where $k = 0$ to m and $m = 1, 2, 3 \ldots, M - 1$ is the level of the grid.

The interpolation of the forces from the grids of spacing h, $2h$, $4h$, $8h,\cdots$, $2^{m-1}h$ is done using the interpolation operator I_m. I_m is calculated in terms of the interaction between the grid point $r^m = (x^m, y^m, z^m)$ and the corresponding nodal basis function ϕ^m,[34] which expressed in terms of the single-dimensional basis function Φ as

$$I_m L_m(r, r') = \sum_p \sum_n \phi_p^m(r) L_m(r_p^m, r_n^m)\phi_n^m(r')$$

$$\phi_p^m(r) = \Phi\left(\frac{x - x_p^m}{2^{m-1}h}\right)\Phi\left(\frac{y - y_p^m}{2^{m-1}h}\right)\Phi\left(\frac{z - z_p^m}{2^{m-1}h}\right) \tag{6}$$

where,$m = 1, 2,.., M$ and p, n are the indices of the grid points.

The continuous differentiation property of Φ and the continuation of γ ensure the conservation of energy, which is critical in long simulations. The interaction kernel $L(r, r')$ is approximated efficiently by nesting between levels of interpolation[53] as

$$L(r, r') \approx (L_0 + I_1(L_1 + \ldots I_{m-1}(L_{m-1} + I_M(L_M)\ldots)))(r, r') \tag{7}$$

The interpolation function I_m (eq 6) can be further divided into computations that assign charges to the grid points. Then the potential at these grid points is calculated based on the charge assigned. Finally, the long-range contribution to the potential is interpolated from the lattice points on the finest grid (the smallest spacing between the lattice points). The local support for Φ is provided by interpolation with a piecewise polynomial of degree q with stencil size $q + 1$.[54] These steps are computed by finding the intermediate charge q^m and the electrostatic potential e^m as described in ref 33.The negative gradient of the potential function (given in eq 3) provides the force exerted on the particles due to electrostatic interaction. It is then used along with bonded and van der Waals forces to find out the displacement of the particles in the system in one time tep by solving Newton's equations of motion.

Accuracy. The appropriate choice of spacing of the finest grid (h) and the short-range cutoff distance (a) determines the accuracy of the force computation. An h-spacing of 2 Å, which is close to the interatomic spacing, is used along with a cutoff radius a between 8 Å and 12 Å. The choice of these values for grid spacing and cutoff radius results in a relative error of less than 1%.[49] Accuracy can be further improved by the use of higher orders of interpolation. Nevertheless, as stable dynamics are achieved at a much lower accuracy in MSM, lower interpolation orders can be used for simulations.[33]

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

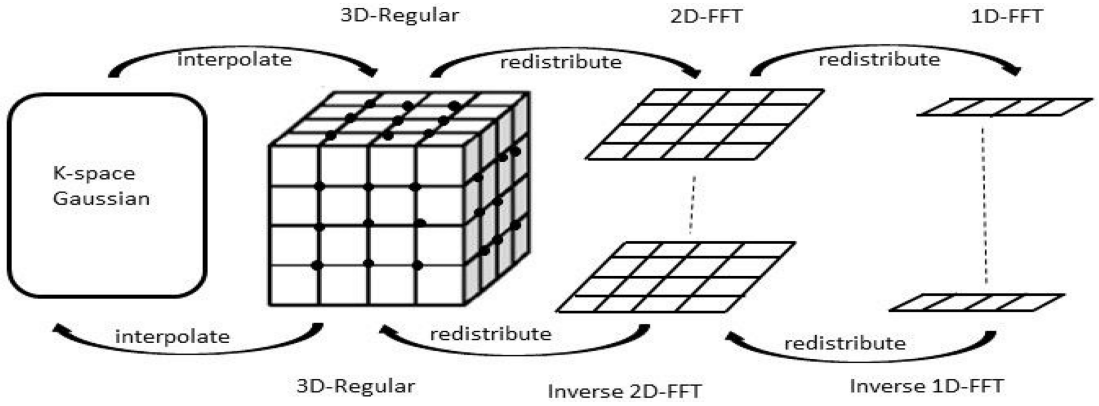


Figure 4. Long-range part of the electrostatic potential calculation in the particle mesh Ewald method.[55] Adapted with permission from ref 55. Copyright 2014 Marc Snir.

Complexity. The computation complexity at each grid point is a constant, and the number of grid points decreases by a factor of 1/8 in each level. The complexity of the algorithm is $O(N + Q)$ where Q is the number of lattice points, and N is the number of atoms.

Scalability. The dense, localized convolution calculations involved and the close neighbor communications in MSM facilitate high scalability and, thus, maximum utilization of parallel-intensive computation platforms.[33]

2.3. Particle Mesh Ewald (PME). PME is based on the Ewald scheme of potential calculations.[31] The application of periodic boundary conditions (PBCs) is used in PME to overcome the boundary surface effect. The use of PBCs results in certain artifacts which are dealt with in the computation of Coulombic potential. In the calculation of Coulombic potential, the correlation between the particles in the central cell and the replicated cells must remain the same. It should also be taken care that there should not be any correlation between the original particle contained in the central cell and the mirror image of that particle contained in the replicated cell.[56] These factors are accounted for in the PME method by calculating the potential energy as the sum of the direct space sum (v^{d}), the reciprocal space sum (v^{r}) calculated as shown in Figure 4, and the error-correcting term or self-term (v^{c}). The direct space sum (v^{d}) calculates the potential due to interactions between charges within the cutoff radius. The reciprocal space sum, v^{r}, includes the interaction between charges in the central unit cell and the replicated cell, and v^{c} is subtracted from the sum of v^{d} and v^{r} to remove the interaction of charges with itself in the replicated cells.[57]

$$V_{\mathrm{Ewald}} = v^{\mathrm{d}} + v^{\mathrm{r}} + v^{\mathrm{c}}$$

$$v^{\mathrm{d}} = \frac{1}{2}\sum_{n}^{*}\sum_{i,j}^{N} q_i q_j \frac{\mathrm{erfc}(\alpha r_{ji,n})}{r_{ji,n}}$$

$$v^{\mathrm{r}} = \frac{1}{2\pi V}\sum_{i,j}^{N} q_i q_j \sum_{m\neq 0} \frac{\exp\left(-\left(\frac{\pi m}{\alpha}\right)^2 + 2\pi m(r_j - r_i)\right)}{m^2}$$

$$v^{\mathrm{c}} = \frac{-\alpha}{\sqrt{\pi}}\sum_{i=1}^{N} q_i^{\,2} \tag{8}$$

where N is the number of particles in the system simulated; n is the coordinate vector of the unit simulation cell and * denotes all values of n which does not include $n \neq = 0$ and $i = j$; q_i and q_j are the i^{th} and j^{th} charges between which the interaction is being calculated; $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$ where $\mathrm{erf}(x)$ is the error function and $\mathrm{erfc}(x)$ is the complementary error function; α is the Ewald parameter; $r_{ji,}n = |r_j - r_i + n|$; V is the volume of the unit cell; and m is the reciprocal lattice vector.[58] V_{Ewald} converts the slowly converging direct potential energy (given in eq 1f) into a fast converging series.[57] The Ewald parameter α allows the balancing of computation between the direct and reciprocal space with no change in the accuracy of the potential calculation.

The calculation of the reciprocal space (v^{r}) is done by approximating the atom charges on a three-dimensional grid. This approximation is made by interpolation using cardinal-B-splines.[58] The reciprocal space v^{r} given in eq 8 can be rewritten as

$$v^{\mathrm{r}} = \frac{1}{2\pi V}\sum_{i,j}^{N} q_i q_j \sum_{m\neq 0} \frac{\exp\left(-\left(\frac{\pi m}{\alpha}\right)^2\right)}{m^2} S(m)S(-m)$$

$$S(m) \approx \sum_{k_1,k_2,k_3} Q(k_1, k_2, k_3)\exp\left(2\pi i\left(\frac{m_1 k_1}{K_1} + \frac{m_2 k_2}{K_2} + \frac{m_3 k_3}{K_3}\right)\right)$$

$$= F(Q)(m_1, m_2, m_3) \tag{9}$$

where $S(m)$ is the structure factor;[59] Q is the 3D grid of dimension K_1, K_2, and K_3 onto which the charges are interpolated; and $F(Q)$ is the discrete Fourier transform of Q.

Accuracy. The main parameters that determine the accuracy of the method are the Ewald splitting parameter (α), the value of the cutoff radius, for which the short-range potential (v^{d}) is calculated, the order of the charge assignment function, and the spacing of the 3D grid. It is an optimization problem to find the optimal value of these parameters that attains the desired accuracy. The charge approximation error decreases with an increase in the order of the B-spline used.[60,61]

Complexity. When α is chosen to be sufficiently large, interaction in direct space beyond a certain cutoff radius becomes negligible, and its complexity reduces to $O(N)$. The use of fast Fourier transform (FFT) to compute the reciprocal space energy and force reduces the complexity of the long-range part to $O(N \log N)$.[32]

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

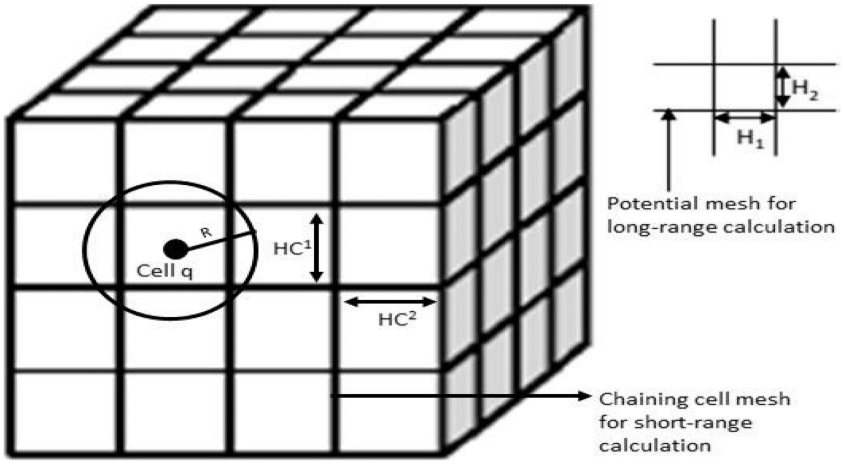


Figure 5. Potential calculation in the particle−particle/particle-mesh algorithm by dividing into two parts, namely, particle−particle for the short-range part and particle mesh for the long-range part. For the short range, a 3D chaining mesh is built around the system such that the sides of the cell HC^1 and HC^2 are greater than the cutoff radius R. The cutoff radius R is taken as 3 to 4 times of the sides H_1, H_2 of the potential mesh.[32]

Scalability. The scalability is constrained by the method of implementation of 3D-FFT used in the reciprocal space calculation. The scalability of PME can be improved by increasing the load of the direct space and thus decreasing the FFT computation to be done in the reciprocal space. This can be achieved by increasing the 3D-grid space.[62,63]

2.4. Particle−Particle/Particle Mesh Method (P^3 M). P^3 M is also based on the Ewald summation method.[31] It gives the most accurate potential among the different methods developed based on Ewald, and it is also the most flexible.[32] In this method, similar to the Ewald method, the electrostatic potential is divided into short-range and long-range as given in Figure 5.

The contributions of the short-range and long-range parts are weighted by the Ewald parameter (α). The division into short-range and long-range in P^3 M is achieved by deducting a shielding charge distribution of identical magnitude from the actual point charge. The short-range potential is calculated as the direct summation of the interaction potential between the combination formed by the point charge and shield charge distribution over the adjacent neighbors.[64] The long-range part involves the effects due to the shielding charges.

The S2 function proposed by Hockney and Eastwood for standard charge distribution[65] is given as

$$\gamma(r) = \begin{cases} \frac{48}{\pi a^4}\left(\frac{a}{2} - r\right) & r < \frac{a}{2} \\ 0 & r > \frac{a}{2} \end{cases} \tag{10}$$

where a adjusts the S2 distribution, and r is the position vector of an atomic charge.

The short-range interaction potential between two particles with charge distribution $\gamma(r)$ is expressed as

$$\psi_s(\zeta_{ij}) = \frac{1}{4\pi\epsilon_0}\left(\frac{1}{r_{ij}} - \frac{1}{70a}\sum_{n=-1}^{7} C_n \zeta_{ij}^n\right) \zeta_{ij} < 2 \tag{11}$$

where $\zeta_{ij} = \frac{2r_{ij}}{a}$ and C_n is a constant that depends on the range of values of ζ_{ij} as given in Toukmaji and Board.[57]

The accurate potential due to interaction between the point charges is found by adding potential due to interaction between the charge distributions (long-range part) and the short-range interaction potential.[64] The long-range interaction potential at each grid point k is found by applying the inverse fast Fourier transform (FFT) on $\hat{\psi}(k)$, which is calculated as

$$\hat{\psi}(k) = \hat{\gamma}\frac{(k)^2}{\epsilon_0 k^2}\hat{\rho}(k) = \hat{G}(k)\hat{\rho}(k) \tag{12}$$

where k is the position vector of each grid point; $\hat{\rho}(k)$ is obtained by the FFT of the gridded charge distribution; and $\hat{G}(k)$ is the optimal influence function.[66] $\hat{G}(k)$ is calculated once at the start of the simulation based on the charge shape, system size, and interpolation function.[64] The force at each grid point due to the potential of the nearest-neighboring grid point is calculated by applying the finite difference operators. Improved accuracy is obtained by considering not only the potential due to the nearest neighbors but also the next nearest neighbors in chain. Thus, a linear combination of finite difference operators is applied to calculate the potential at each grid point. Unlike analytical differentiation, this method conserves momentum and thus avoids errors such as incorrect particle drifts.[65] Fourier transformation is done once to obtain in real space the force exerted on each particle of the system and, in turn, the velocity and thus the displacement in each time step.

Complexity. The complexity of the algorithm is $O(N \log N)$, which is similar to the other mesh-based Ewald methods.[67]

Accuracy. It is the most accurate among the Ewald-based methods. The accuracy of the algorithm depends on the use of a refined mesh or the use of higher orders of interpolation, which are computationally expensive.[32] Another method called "interlacing" is used, in which a second grid is introduced at half a grid spacing, and the same calculation as on the initial grid is performed. The potential and force obtained on both the grids are averaged to get more accurate values.[68] Thus, interlacing facilitates the use of wider spaced grids, and hence performance is not hindered.

Scalability. Similar to PME, the scalability depends on the workload distribution between the short-range and long-range parts. More long-range load results in more FFT calculations involving all-to-all communication and hence less scalability.

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

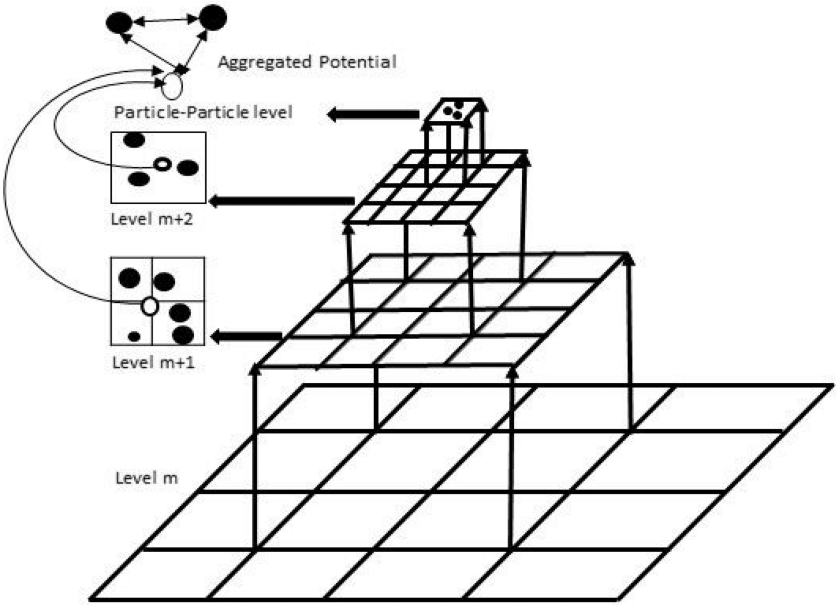


Figure 6. In the fast multipole method, the interactions on a particle at the particle−particle level are calculated between all the particles that are located in the same and surrounding cells. The interactions with particles from the larger cells are approximated. Adapted with permission from ref 72. Copyright 2006 Taylor and Francis Group LLC (Books) US.

The algorithmic implementation of 3D-FFT plays a very significant role.[69,70]

2.5. Fast Multipole Method (FMM). The FMM is one of the top 10 algorithms of the 20th century[36] developed by Greengard and Rokhlin for the fast calculation of electrostatic potential.[35] In this method, the potential calculation is divided into interactions between neighboring or direct particles and those between distant particles as shown in Figure 6. The distant particle interaction is calculated by multipole expansions.[71]

The infinite multipole expansion gives the potential $V(r)$ at point P, which is at a distance of r from the center of a collection of point charges enclosed in a sphere of radius a and is shown in eq 13 as

$$V(r) = \sum_{i=0}^{\infty} \sum_{m=-l}^{l} \frac{M_l^m}{r^{l+1}} y_l^m(\theta, \phi) \tag{13}$$

where $y_l^m(\theta,\phi)$ is the spherical harmonic potential;[73] θ and ϕ are the zenith and azimuth angles of the atom at point P; and M_l^m is the l^{th} multipole moment at level m.

The simulation box is hierarchically decomposed into a tree structure of smaller subunits. The interaction effect of all the particles in a subunit on a distant particle is calculated by the truncation of multipole expansion for $p = -\log_{\sqrt{3}}\epsilon$ terms, where ϵ is the required accuracy. The effect of larger groups of particles is obtained by hierarchically combining these multipole expansions. Potential accumulation due to the interaction of all subunits is obtained by the hierarchical combination of the Taylor expansions.[74] The distant forces thus calculated are added to the direct forces to obtain the potential of each particle.[75]

The long-range electrostatic interaction of systems having a nonuniform distribution of particles can be computed using the FMM. It can also be applied to both periodic and vacuum systems. The FMM is utilized for QC calculations of large systems where scaling is considered an essential criterion along with accuracy.[76] However, for QM/MM calculations, PME is considered more efficient than the FMM. The greater code complexity of the FMM and the lack of energy conservation compared to mesh-based Ewald methods have led to the usage of primarily mesh-based Ewald methods in the molecular dynamics' packages of the current era.

Accuracy. It is based on two parameters, mainly the depth or levels of the FMM tree and the length/number of terms in the multipole expansion. Based on the user-provided energy error threshold, the optimal set of these parameters is obtained by running the error control and energy minimization scheme.[77,78]

Complexity. FMM has a complexity of $O(N)$ and hence scales linearly with system size.[79,80]

Scalability. It is one of the most scalable algorithms among electrostatic algorithms due to its linear complexity. The communication complexity is $O(\log P)$ where P is the number of processes due to its hierarchical nature.[81] The distribution of particles among processes is performed using the z-space filling curve. It reduces the communication required between the processes and, in turn, increases the algorithm's scalability.

The summary of comparisons of the different algorithms used for electrostatic force calculation, discussed in Section 2, is given in Table 1 in terms of accuracy, complexity, scalability, and boundary conditions required for the respective algorithms.

3. ELECTROSTATIC FORCE CALCULATION ON GPUS BY DIFFERENT MD SOFTWARE

The data-parallel, compute-intensive nature of MD force calculation stages makes graphics processing units (GPUs) a very suitable choice for accelerators.[83,84] The use of GPUs with multicore CPU provides the researchers with increased computation capability on workstations and laptops. Thus, GPU augments multicore CPU performance. GPUs are equipped with floating-point performance of over tens of teraflops, high-bandwidth memory, and hardware multithread-

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

Table 1. Summary of Comparisons between Different Electrostatic Force Calculation Algorithms (EFC Alg.)[a]

EFC Alg.	complexity	accuracy	scalability	boundary conditions
DCS	$O(NQ)$[60]	Highest[47]	Low[47]	NPC, PC
MSM	$O(N + Q)$[33]	Medium[33]	High[33]	PC, SPC, NPC
P^3 M	$O(N \log N)$[67]	High[32]	Medium[69,70]	PC
PME	$O(N \log N)$[32]	Medium[60,61]	High[62,63]	PC
FMM	$O(N)$[79,80]	Medium[77,78,82]	High[81]	NP, PC

[a]NPC stands for nonperiodic conditions, PC for periodic conditions, and SPC for semiperiodic conditions.

ing. These GPU features help to accelerate the electrostatic force calculation stage of MD. The design and implementation of the electrostatic force calculation algorithm such that it scales to hundreds of tightly coupled processors is the most key requirement for harnessing the power of GPU computing.

The exploration of different techniques of acceleration of the electrostatic force calculation in MD production packages by GPUs is done in the following sections. It gives information to the researchers and developers working on MD performance on the currently employed strategies for accelerating electrostatic force calculation algorithms.

3.1. AMBER MD. Assisted model building with energy refinement (AMBER) is a set of force fields developed to simulate the dynamics of biomolecules,[85,86] and the AMBER MD[23] is a MD package that utilizes the AMBER force fields for simulation. The nonbonded force calculations in AMBER involve the van der Waals (vdW) interactions and the Coulomb interactions.[87] For the van der Waals (vdW) interactions, the cutoff distance is used as it decays to zero after a certain distance, whereas the Coulomb forces reduce in strength very slowly. Hence, if ignored, it can lead to abrupt dynamics due to the lack of energy conservation. The use of periodic boundary conditions (PBCs) avoids artifacts due to boundary conditions. Hence the unit cell, which contains the system to be simulated, is replicated in all three dimensions.[88]

The Coulomb force calculation algorithm is done in AMBER using PME (given in Section 2.3), the lattice sum method, which is used to take into consideration the force contribution beyond the cutoff. In AMBER, for explicit solvent MD,[89] the long-range electrostatic force is calculated fully on GPU using this algorithm. In the reciprocal space, 3D fast Fourier transform (FFT) of charge from real to complex is performed using the CUFFT library[90] of Nvidia. This calculation in AMBER implementation happens with one thread per particle.

Techniques for Acceleration. Arithmetic Precision. The single-precision fixed-point (SPFP) integer arithmetic combines single precision with a 64-bit fixed-point arithmetic. The use of SPFP in the place of a double-precision floating point for PME calculation helps significantly to boost the performance with not much loss in accuracy.[91] This use of SPFP also helps fully utilize the future and present GPU architectures without incurring a performance degradation in the old GPUs. The current version of AMBER provides three precision models: the SPFP, which is the default model; the SPDP, which uses double-precision floating-point variables for all three bonded term (bond, angle, and dihedrals) calculations; and the accumulation of potential and single-precision floating points for all others, the DPDP, which uses double precision for the entire GPU code.[91,92]

Neighbor-List Creation. The neighbor-list of atoms within the cutoff distance is obtained using two sorting mechanisms.[93] This list is used to perform the direct sum on the GPU. The atoms are first sorted into boxes with dimensions at least equal to the cutoff radius, and each atom is assigned a box ID. Then, in the next step, to improve the spatial locality of each atom in the box, a Hilbert curve[94] is implemented as shown in Figure 7.

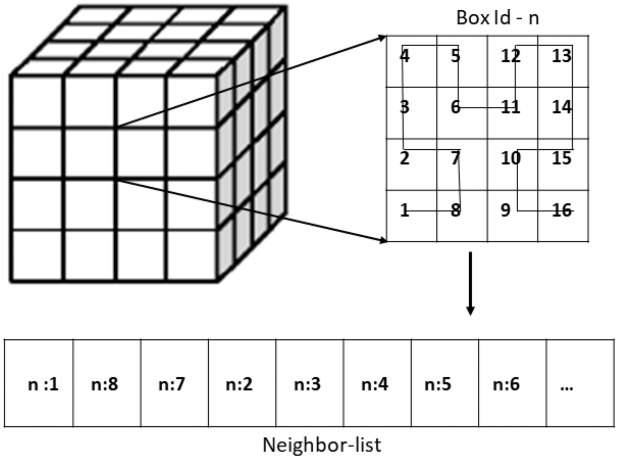


Figure 7. Neighbor-list is prepared by the use of a Hilbert curve ID along with a Box ID. Adapted with permission from ref 93. Copyright 2013 American Chemical Society.

The box ID and the Hilbert curve coordinates are then used to sort the atoms. The neighbor list for each atom is done using the sorted atoms by considering all the adjacent boxes within the extended cutoff (cutoff radius + buffer); i.e., all the atoms within (extended-cutoff) R^2 are taken as a neighbor. A proposed optimization uses a bit-mask on the Hilbert ID to exclude certain atoms from the R^2 calculation. Once the neighbor-list is formalized, the vdW and electrostatic interactions are calculated by skipping the 1−2, 1−3, and 1−4 interactions that are listed in the topology file.[95]

GPU Cluster. Parallel simulations across multiple GPUs were done using the message passing interface (MPI) protocol by replicating the data structures to all the GPUs.[93] Hence, the memory usage on each GPU is approximately identical with a single parallel implementation, which can be improvised by better communication and peer-to-peer functionality for direct memory copies.[96]

In AMBER18,[97] additional features have been added to improve the acceleration of PME calculation. AMBER18 has achieved for dihydrofolate reductase (DHFR) a simulation speed of 657 ns/day compared to 588 ns/day in Amber16 (4 fs time step, constant energy). This acceleration has been achieved by faster PME real space and reciprocal space calculation by the innovative spline tabulation look-up and particle mapping kernels and optimizing the memory access of bonded and nonbonded terms.

3.2. NAMD with PME. The smooth particle mesh Ewald (SPME) algorithm is used in NAMD[24,98] for electrostatic force calculations, which is a variation of PME. In this, the charge-spreading action is performed by the B-spline functions. This interpolation scheme is a continuous function at the grid points (n). As the points are all distinct, their derivatives are all continuous up to a certain degree ($n - 1$).[99] This property of

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

B-splines helps conserve energy as the derivatives of the potential yield continuous forces. It also helps reduce the number of FFTs to half compared to the PME method. In SPME, potentials are approximated, but the forces are the exact derivatives. Integration of the forces is achieved by multiple time steps where the nonbonded forces are calculated the least frequently. The neighbor list for short-range forces is obtained by dividing the system spatially into patches. Within each patch, the atoms are sorted into groups of 32 using the orthogonal recursive bisection method.[100] A compute object in NAMD consists of 32 × 32 tiles. It is further divided into tile lists that can be run on any thread block. This division into tile lists provides more flexibility for load balancing between the warps in a thread block.[24]

Techniques for Acceleration. FFT Parallel Implementation. The 3D FFT is implemented for multi-GPU by using a pencil, slab, or box decomposition based on the number of GPUs as shown in Figure 8. Each decomposition is assigned to a single GPU. For pencil decomposition, 1D FFT is performed; 2D FFT is performed for slab; and 3D FFT is performed for box-level decompositions, respectively, using the cuFFT library.[90] For 1D FFT, transposing of the grid is done to get a contiguous memory location for the next FFT direction.[101] The parallel implementation of FFTs requires numerous messages to be passed between MPI nodes. Nevertheless, the MPI architecture adhered to by NAMD makes it possible to interleave the FFT calculations with the dominant short-range nonbonded interactions.[102]

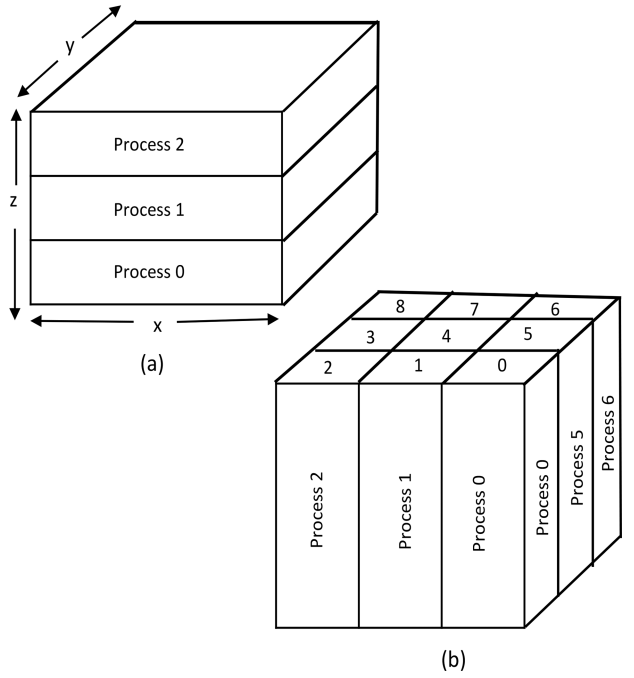


Figure 8. Different types of decomposition for parallelization of FFT (a) slab decomposition and (b) pencil decomposition. The all-to-all communication is minimum in slab decomposition. The number of slabs is based on the number of processes in the CPU/GPU node.[103]

Newton's Third Law. According to Newton's third law, when the interactive forces between atoms i and j are calculated if a force f_{ij} is exerted on i, then the force exerted on j by atom i will be $-f_{ij}$. This law helps to avoid the calculating force on atom j again. Race conditions generated when all the threads in a warp are trying to write to the same memory location of atom j can be avoided by looping through the tile in diagonal lines.[104] The use of shared memory and its synchronization overheads for shifting atom coordinates, charges, and computed forces are avoided by the use of *shfl* instruction, which makes use of GPU registers.[105]

Charm++. This is a programming system and run-time library used by NAMD.[106] It provides a message-driven programming framework and processor virtualization capability. In message-driven programming, a method is invoked on an object only when a message arrives for it, and thus it helps to avoid communication latency.[107] Processor virtualization allows algorithm writing for a maximum number of parallel objects and dynamic distribution among the available processors at a run time. The performance achievable is decided by the nonbonded calculation, which scales as $NR^3\rho$ where N is the number of atoms, R the cutoff distance for short-range calculations, and $\rho = N/V$ the number of atoms per unit volume V.[24]

3.3. NAMD with MSM. The multilevel summation method (MSM, Section 2.2) is used as an alternative method for electrostatic force calculation in NAMD[53,108] as it supports periodic, semiperiodic, and nonperiodic biomolecular simulations. The GPU accelerates this compute-bound algorithm. High throughput is achieved by harnessing the power of thousands of multithreaded cores in the GPU. The communication structure of the MSM algorithm guarantees parallel scalability, and the slowly varying potential allows multiple time steppings, which helps to improve performance.[109]

The short-range part of the system potential V^{msm} (the first term in eq 3) becomes more compute-intensive as the grids become finer. The most time-consuming computations involved in the algorithm are the lattice-based cutoff part, v^{long}, and the short-range part of V^{msm}. Hence, these calculations are accelerated by the GPU.

Techniques for Acceleration. Small-Bin-Based Geometric Hashing. This approach is used to accelerate the short-range part of the MSM. In this, the CPU performs the geometric hashing of the atoms into small bins. This separation allows the calculation of interactions to be limited to only the neighboring bins. CPU regularizes the approach to avoid boundary conditions by adding empty bins. Then, these bins are copied to the global memory of the GPU. A subcube of lattice points is assigned to a thread block. The thread block calculates the potential of each subcube of lattice points by iteratively copying the neighboring atom bins to the shared memory. The offsets of the neighboring bins are read from the constant memory by using their thread block index. The load balancing between the CPU and GPU for computation of atom interactions was found to be ideal for a bin size of 4 Å.[101]

Sliding Window Approach. The potential at each point in the lattice is the sum of surrounding charges that are weighted by their distance to the respective point. The potential due to interaction between points on a lattice is calculated by the lattice cutoff part v^{long} (eq 3). At each level, k, the bounding sphere which has a cutoff radius of $2^{k+1}a$ has the same number of atoms as levels $k + 1$ and $k - 1$. The weights at the boundary of a sphere can be calculated theoretically, as the distances between points on a uniform lattice are the same.

Each thread block is assigned a subcube of size 64 lattice points, and each thread is assigned a point. The constant memory is used to store the precalculated weights, and the

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

shared memory is used to store the surrounding neighboring cubes of charges. To achieve constant memory access at a registered speed, all the threads in a warp have to read the same weight. The sliding window approach is used to achieve this. The shared memory is loaded with eight subcubes containing 64 charges each in this approach. A sliding window of size 4^3 is shifted four times in the three dimensions, namely, *x*, *y*, and *z*, within a triply nested loop. In each sliding window position, every thread applies a single weight value to its corresponding charge to obtain the updated potential. This sliding window loop is embedded in another triple-nested loop that iterates over the neighborhood cubes loaded in the shared memory. For the correctness of the sliding window approach, the eight subcubes should be padded by a layer of subcubes with no charged points in them. For coalesced reads from global memory, the lattice points in each subcube are stored in *x*,*y*,*z*-ordering, and the subcubes are stored in row−column−depth order.[101]

3.4. GROMACS with PME. GROMACS is one of the most widely used open-access MD packages. It employs the heterogeneous platform of multicore CPUs and GPUs for maximum performance, throughput,[25,110] etc. The electrostatic force calculation is performed using PME (Section 2.3). The highly parallelized environment of GROMACS divides the computation among ensembles, domains, and multiple SIMD cores.[111] This decomposition of workloads with SIMD, openMP, and MPI acting on each domain helps maximize balanced hardware resource utilization. The different methods implemented for PME acceleration in GROMACS are explored.

Techniques for Acceleration. Eighth Shell Domain Decomposition. This method[112] helps to significantly reduce the communication involved for force calculation between the domains. Charge groups (grouping of partially charged atoms into a single neutrally charged group) are used as the basic element in the decomposition as it reduces the number of calculations involved. In the eighth-shell decomposition, the system is split into independent subsystems along with dynamic load balancing, and these subsystems are then mapped into separate MPI ranks (while retaining the "locality of reference" within each domain). Multiple OpenMP threads are contained in each MPI rank. The best performance is achieved when the product of the number of MPI ranks and number of OpenMP threads are equal to the number of cores on a node, and the threads are pinned to the core. Each of the subsystems is described as a triclinic cell, which helps decrease the volume of solvent compared to a rectangular cell and thus helps reduce computation and increase performance by 40%. The apparent difference in the workload between the protein and solvent/water domains is addressed by dynamic load balancing.[113] For the electrostatic interactions, 3D domain decomposition is performed for the direct space, and 2D pencil decomposition is done for the reciprocal space.[25]

Dedicated Nodes. The long-range part of the electrostatic interaction is allocated with separate dedicated MPI nodes[113,115] as in Figure 9. The division of the available nodes is such that 3−4 nodes are dedicated to a direct space map, and a single node is dedicated to reciprocal space. The PME nodes are kept to a minimum as the message communications required for 3D FFT calculations scale as a square of the number of nodes involved. The 3D FFT is performed using 2D pencil decomposition.[116] This mode of allocation helps to automatically determine the division of the

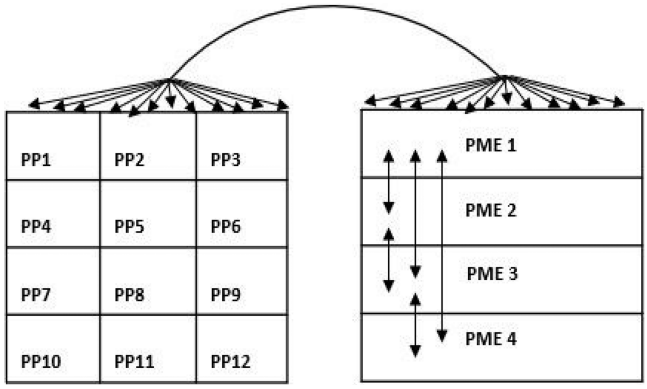


Figure 9. Scaling with domain decomposition is achieved with the multiprogram multidata approach in which particle−particle (PP) ranks to PME ranks are allocated in the ratio of 3:1 for rectangular simulation boxes and 2:1 for rhombic dodecahedra to minimize all-to-all communication. The arrows represent the flow of communication. Each PP rank communicates with the PME rank.[114]

workload into short-range and long-range force calculations among the nodes available in the simulating platform.

The FMM has also been implemented in GROMACS on GPU.[82] The FMM is faster when the simulation is performed on a large number of particles, large simulation boxes, inhomogeneous charge distributions, and high parallelization.[116,117] The multipole-to-local calculation is the most compute-intensive part of the FMM. Its complexity depends on the multipole order (refer to Section 2.5). The ScaFaCoS FMM[67] is used in GROMACS, which is fully parallelized using CUDA.

3.5. LAMMPS. The large-scale atomic/molecular massively parallel simulator (LAMMPS)[26] is a highly parallelized MD package. The parallelization is achieved by spatial decomposition of the system domain into multiple subdomains. Domain decomposition is performed dynamically for load balancing, as shown in Figure 10. Each of the subdomains is

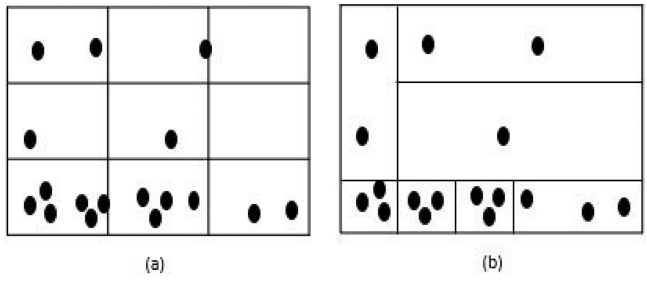


Figure 10. Domain decomposition for a system with uniform density into an equally sized box provides an equal load to all the allocated cores, but for a nonuniform density system, an equally sized block gives different load to the cores as shown in (a). LAMMPS performs domain decomposition as shown in (b) for load balancing.[118]

allocated to separate nodes. The communication between the nodes is based on the message passing interface (MPI) protocol. In LAMMPS, the long-range electrostatic potential calculation is done by the P^3 M algorithm. The acceleration of the P^3 M algorithm is achieved by its implementation on a distributed system of GPU clusters connected by InfiniBand[69] and the algorithmic methods proposed for PME.[101]

Techniques for Acceleration. Charge Assignment Acceleration. The charge assignment function used is a spline of

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

order P. The algorithm's accuracy depends on P and the spacing of the grid, h. The values used for P and h determine the error due to the discretization of the charge. The parameters P (default value of 5), h, and α are fixed for a user-specified value of the root-mean-square (RMS) discretization error.[32]

For parallelization, a 3D grid is virtually implanted in the simulated system, and the system is divided into subdomains. The subdomains are allocated to each of the processors in the cluster. The processor performs the grid charge allocation for the charges that belong to its subdomain. All the grid points (P^3) within a P-stencil width centered on the grid point closest to the charge are affected by the charge allocation. Since this charge allocation involves ghost grid points that do not belong to the respective processor, it must be communicated to the neighboring processors.

3D FFT Acceleration. Once the charge allocation has been performed, parallel 3D FFT is performed. The forward 3D FFT is carried out in three communication stages. The 3D grid is partitioned into 2D columns in x-dimension, then 2D-x into 2D-y, and finally 2D-y into 2D-z. On the final output of the grid partitioning, 2D-z, convolution is performed. The inverse FFT is performed using the same three communication stages in the reverse order. The gradient for force calculation is done using the ik-differentiation[119] in the reciprocal space. The electric field vector of each particle is obtained by interpolation from P^3 grid points. It also involves the ghost grid points. This information is obtained by six-way communication with the neighboring processors.[69]

The large volume of communication exchanged between processors compared to computational intensity is the bottleneck in the scalability of parallel 3D FFT and, hence, the scalability of particle-mesh methods.

3.6. ACEMD. ACEMD[27] is an MD software designed to scale across multiple GPUs to achieve the microsecond long simulation in an economical and effective manner. The ideal system for ACEMD is a single node attached to multiple GPUs via PCIe expansion slots. The GPUs compute different force terms based on a simple force−decomposition scheme.[120] Dynamic load balancing is performed to distribute the particles for force calculation among the GPUs. The host processor then sums up the force terms computed by the GPUs. This total force matrix is transferred back to all the GPUs for complete system integration. For electrostatic force calculations, the PME[60] algorithm is implemented in a scalable parallel manner by dedicating a specific number of GPUs for it.

Techniques for Acceleration. Cell-List Construction. The system is decomposed into cells for dynamic load balancing. Then, based on the cell-list scheme,[101,122] the particles are first binned based on their coordinates. For an MD simulation that uses a cutoff radius, R = 12 Å, approximately 22 atoms can be accommodated in bins with size $R/2$. Thus, the number of atoms in a bin (cell) is comparable to the warp size in a GPU of 32 threads. Each cell is assumed to have a maximum of 64 atoms to overcome the density fluctuations that arise during the simulation. For PME, the accuracy required for MD is provided, by using a cutoff radius R = 9 Å. This, in turn, results in each cell being populated with at most 32 atoms. Each thread launched by the cell-list construction kernel processes the construction of the cell list for one particle at a time for all atoms in that cell. Atomic memory operations are used to avail concurrent access to the cell-list array.[123]

Optimized Memory Access. For the nonbonded force computation, a thread block is allocated a single cell as shown in Figure 11. All the atom coordinates within a radius R of the

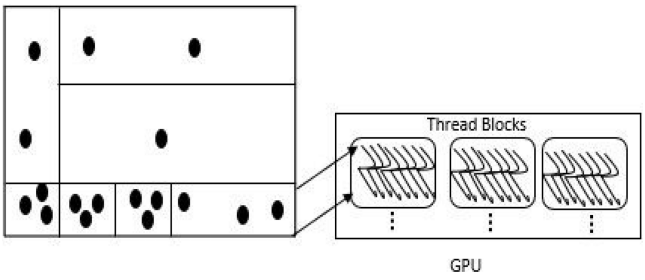


Figure 11. Optimized memory access with each cell allocated to a thread block in the GPU.[121]

current cell are loaded into the shared memory. The forces are computed by looping over the particles loaded in the shared memory. The cost of global memory access restricts the use of Newton's third law in this GPU implementation, and hence the reciprocal force is not stored. The texture memory helps by providing radial components of the force functions (Coulomb and van der Waals) from a lookup table. The lookup table provides the linearly interpolated values of these functions. The error due to the use of the lookup table is acceptable for MD as it is consistently less than 10^{-4}.[18]

3.7. DESMOND. DESMOND[18] is an open-source MD program developed by D. E. Shaw Research. DESMOND achieves high performance through well-defined implementation techniques and novel distributed memory parallel algorithms that accelerate parallel MD simulations significantly. These include parallel decomposition methods, message-passing techniques, novel communication primitives, and short-vector SIMD capabilities. The reduction of conflicts in updating the nonbonded forces has provided the maximum performance improvement.[22]

Techniques for Acceleration. Midpoint Method. The system that is to be simulated is partitioned into a mesh of rectangular parallelepipeds or boxes. To parallelize the range-limited electrostatic interactions within a cutoff radius R, similar to the spatial decomposition strategy,[124] each process updates particles in one box of the grid. This significantly reduces the communication bandwidth required by the process, and it further reduces if the cutoff radius is decreased.[112,125−128]

The midpoint method[126] further reduces the communication bandwidth. This method, shown in Figure 12, calculates the interaction between two particles in a box if the midpoint of the segment connecting the two particles falls in that respective box. The data of particles that reside within the $R/2$ distance of the home box have to be imported for the interaction calculation and are called the import region in the midpoint method. The communication bandwidth requirement for this parallelization method depends on the size of this import region. The midpoint method has the advantage that the same data communicated for pairwise nonbonded interactions can be reused for bonded forces, requiring local communication. The particle simulation parallelization engine, a custom portable library, helps with the efficient communication and computation in the midpoint method. The library manages data order in memory, and the messages to be sent to other nodes are assembled in the order in which it is to be sent.

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

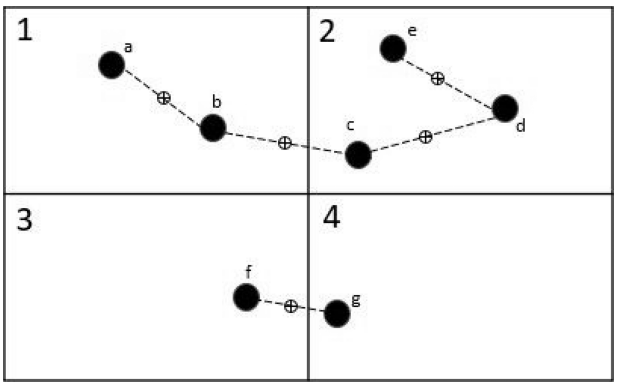


Figure 12. In the midpoint method, if the distance between two particles is less than or equal to the cutoff distance, then the midpoint of the line joining the two particles is calculated, and the computation of the interaction for the particles involved is done in the box in which the midpoint falls. In the above figure, for the particles b and c, the midpoint falls in box 1, and hence the computation for this interaction is done in box 1.[18] Reprinted in part with permission from ref 18. Copyright 2013 IEEE.

Task Decomposition. The particle-based approach is used for task decomposition.[22] The near-pairs, which are pairs of atoms that come within the cutoff radius, are stored in a nonredundant linear list. In this decomposition, each GPU thread is dedicated to one particle, and it performs the force calculation for all pairs involving that particle. For force calculation, the "full-shell" mechanism is used, where the force for each particle is calculated by its respective thread. Hence, the same pair interaction is calculated twice.[129] The conflict removal of different threads updating the same particle's memory is obtained at the cost of redundant storage and computation. This mechanism leads to better load balancing.

Charge spreading and force interpolation involved in PME and the k-space Gaussian split Ewald method (k-GSE)[130] require no further communication when this decomposition method is implemented. Force contribution to all the particles in the import region must be exported after computing all the home−box interactions. Once the force on the particle has been calculated, particle migration to different processors has to be taken care of due to the atom position update. However, if the import region is increased slightly, this particle migration needs to be performed only after a few time steps. This is because the position change of a particle is significantly less as compared to R[125] in each time step. This reduces the communications required at the cost of a slight increase in bandwidth for increased particle information during particle import and force data transfer in the force export stage. The volume of the import region is the measure of communication bandwidth for this parallelization strategy.[18]

Staged Communication. During the import and export, staged communication[124,131] is used. This reduces the number of messages from 26 (number of neighbors for a home box) to 6, that is, one in each cardinal direction ($+x$, $-x$, $+y$, $-y$, $+z$, $-z$) in three stages.

3D FFT Parallelization. Then, the 3D FFT is performed on the mesh. For parallelization of the 3D FFT used in PME, a 3D mesh of size $N_x*N_y*N_y$ (each of which is a power of 2) is superimposed on the simulated system.[18,127] The points on this mesh are spread across $p_x*p_y*p_z$ boxes, and each of the boxes has a mesh of size $n_x*n_y*n_z$. 1D FFTs are calculated successively in each of the three dimensions to evaluate the 3D FFT. $N_xn_yn_z$ points occur in each column in the x-dimension and require n_yn_z 1D transformations, and the data for each transformation are distributed across p_x processes. The 1D FFT in the x-dimension is calculated in $\log_2(N_x)$ stages by data exchange between $\log_2(p_x)$ processes. In each stage, the communication only happens between a process and another process. In DESMOND, the forward transforms are performed by decimation in frequency. Three transforms are performed for each of the three dimensions, and the transformed data become bit-reversed in order. Then, multiplication by the optimal influence function is done.

An inverse transformation brings back the initial spatial distribution using decimation in time. Based on this FFT parallelization, when FFT is performed using 512 processes (8 × 8 × 8), the FFT calculation requires nine messages to be sent by each process (three in each of the three directions). The number of messages and their latency is the primary determinant of the FFT latency in the case of small meshes. However, for large meshes, the message bandwidth also requires significant consideration. For intranode parallelization, pthreads are used, which divide the work assigned to the process to be shared by multiple threads and processors. Normalized representations are used for atom position coordinates.[101]

SIMD Programming Interface. The SIMD programming interface was developed to use short vector SIMD instructions without hand-coding these instructions, and this simplified the work of the coder along with accelerating the computations.[18] These SIMD extensions simultaneously perform four 32-bit single-precision floating-point operations with 16-byte aligned memory accesses. Single precision in computation has the advantage of reduced memory and communication bandwidth.

Bitwise Time Reversibility. An ideal MD simulation should be reversible in time. However, the round-off errors and lack of associativity in floating-point arithmetic lead to the inability to reverse a simulation. DESMOND can achieve accurate bitwise time reversibility for simulations that do not use constraints.[18] It uses fixed-point arithmetic for updating particle position that overcomes the round-off error. The nonassociativity of floating-point arithmetic is dealt with by maintaining an ordering of computation and particles. This DESMOND feature helps to achieve an energy drift of less than 1 K per microsecond with single-precision calculations.

The position coordinates are normalized to the respective box, which helps achieve extra precision. The storage of different data structures using the optimum memory hierarchy plays a significant role. The position, force, and energy information are stored in shared memory. The force tables, which store the intrinsic properties of force fields, are stored in constant memory as they are constant values. The controlling conditions are modified to avoid or reduce thread divergence, and branch predictions are made. In addition to these optimizations, streaming SIMD extension (SSE) instructions[132] help to further speed up the computations up to 3×.

3.8. Comparison of Electrostatic Algorithms and Its Implementation in MD Packages Based on Its Advantages and Disadvantages.

In MD simulation, the main criteria to be kept in mind are ensuring that the simulation's accuracy meets the accuracy threshold set by the user and the computation cost involved in achieving that accuracy like the simulation time, efficient utilization of resources, etc. The hyperparameters of the algorithm are set based on these criteria. The Ewald-based methods such as PME and P^3 M are

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

based on fast Fourier transforms (FFTs). The use of FFTs helped to reduce the quadratic computation complexity of electrostatic algorithms to $O(N \log N)$ with comparable experimental accuracy for force calculations. The inherently periodic systems, such as crystals, are particularly suited for FFT-based methods. The use of FFTs is made possible in PME and P^3 M by applying periodic boundary conditions. The periodic boundary conditions consider the dynamics that occur at the boundary of the simulated system, but the applicability of periodicity to inherently nonperiodic systems such as solutions is questionable. It has been seen that introduction of artificial periodicity for solvated biomolecules results in perturbation of conformational equilibrium. It, in turn, causes minor fluctuations and an artificial stabilization of the most compact state.[133] The effect is more pronounced when the solvent has low dielectric permittivity, the solute has an overall charge, etc.[134]

The Ewald summation method requires that the system's simulation box is neutral. The simulation of non-neutral systems is facilitated by introducing a uniform background charge distribution that neutralizes the simulation box[135] or by adding explicit counterions. The use of background charges may result in the rise of artifacts, as the spatial distribution of counterions in real systems is different, particularly in systems with an inhomogeneous dielectric constant. The uniform background charge distribution in inhomogeneous systems artificially changes the potential of mean force. This artifact can be overcome by the use of explicit ions to neutralize the simulation box. The effect on the energy and pressure introduced by the background charge in homogeneous systems can be corrected on the fly or *a posteriori*.[136]

The required global transpose of data in FFTs becomes a great challenge while computing the three-dimensional FFTs when PME or P^3 M algorithms are implemented on large clusters of CPUs or GPUs. It has been found that the scalability of the FFT is $\sqrt{p}$ where p is the number of processes. The cost of communication between the processes scales as $\log(p)$[137] when compared with hierarchical clustering methods, such as the FMM. This reduced communication cost is achieved by the memory access pattern in the FMM, which is more localized due to the use of approximations based on spatial scale separation. The need for an efficient and scalable algorithm to run on the future exascale computer has brought the FMM into the research forefront. The memory becomes a constraint in the simulation box size scalability in the case of PME, whereas in the FMM, only the number of particles is the limit. The performance of the FMM becomes very high when there is sufficient parallelism to exploit. The FMM achieves the same accuracy as PME when a multipole order of 7 is used with a depth 3. On the other hand, the standard parameters (cutoff radius −1.0 nm, grid spacing of 0.12 nm, and a B-spline interpolation order of 4) are used for PME. The energy drift occurs in the FMM due to the octree discretization. This discretization also results in small force discontinuities over time. This can be compensated in the FMM by increasing the multipole order to more than 8 or regularization for lower orders. In GROMACS, energy drift is significantly reduced for both PME and FMM implementation using a large Verlet buffer and double precision calculation. Further, it is found that in the case of a biomolecular system with 50 000 atoms the performance of the FMM is only about one-third as compared to PME in GROMACS. Thus, it is recommended to use the FMM for systems with large dimensions and nonuniform particle distribution, such as aerosol or multi-droplets within 10 000 atoms. In these situations, the large memory requirement of the FFT grid makes PME unsuitable for use.[82]

MSM provides better scalability compared to PME when the number of processors is large. This is because FFTs in MSMs which require global communications are avoided in the lower levels by approximating the long-range part on a hierarchy of grids and using it only at the coarsest (highest) level, and it is also not iterative. It uses near-neighbor point-to-point communication. The computational time taken by MSM can be reduced further by choice of optimum grid spacing h (see Section 2.2). The optimum grid spacing can be obtained using an equation given by Moore and Crozier[138] in Section 4B as the starting point to decide the levels and, in turn, the number of grid points. Fine-grained control of accuracy in MSM is not possible because the grid spacing in MSM can only be changed in each direction by factors of two or eight when changed in three dimensions simultaneously. For example, a system ApoA1 (92K atoms) was simulated on Cray XE6 nodes (32 cores per node) by varying the number of nodes in which PME showed higher performance when the number of cores was less than 256. The performance crossover between MSM and PME occurs between 256 and 512 cores.

The performance of PME reaches a plateau at 1024 cores, while MSM continues to scale for up to 1536 and achieves a performance of 20 ns per day. MSM produced comparable accuracy with PME in the calculation of water properties such as density, diffusion constant, dielectric constant, etc. Still, the numerical accuracy of PME is much more than MSM. The optimum choice of parameters such as grid size, ratio of the cutoff to grid size, and interpolation order is important to achieve low root-mean-square error, which evaluates the accuracy. The MSM method greatly benefits molecular simulations involving nonperiodic and semiperiodic boundary conditions. The property of MSM to simulate in semiperiodic boundary conditions can be applied in the simulations of the cell membranes. It can also be used to simulate nanopores used as sensors and situated in a graphene sheet, which acts like a cell membrane. MSM is implemented in the molecular dynamics package NAMD as explained in Section 3.3.[53] MSM was also found to be faster than FMM for a given accuracy requirement. This is because, in MSM, forces are smooth as compared to in FMM.[34,141] MSM was found to be slower than SPME in NAMD for a water system simulated on a single processor,[63] but MSM running on GPU has significant speedup due to its highly scalable nature.[142]

For DESMOND,[143] the simulations run with the u-series algorithm produced a performance of 1100.7 ns/day on one V100 with a 2.5 fs time step, whereas Amber18[144] achieved a performance of 779.46 ns/day and 411.90 ns/day with 4 and 2 fs time steps, respectively, for the DHFR system with 23 000 atoms. ACEMD provides 937.5 ns/day on single V100 GPU for DHFR (23K atoms). ACEMD[145] produced 28 ns/day, and DESMOND provided a performance of 34.1 ns/day on a single V100 for STMV (Satellite tobacco mosaic virus) with 1 067 095 atoms. GROMACS 2020[146] provides 43.1 ns/day on four V100 GPUs. GROMACS 2020 achieves 147 ns/day performance on four V100 SXM GPUs on the Cellulose system with 408 609 atoms, and Amber18 provides a performance of 53.39 ns/day for one V100 system. In all the MD packages except DESMOND, the PME algorithm is used.

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

The main advantages and disadvantages of electrostatic algorithms implemented in MD packages are summarized in Table 2.

3.9. Acceleration of Electrostatic Potential Algorithms Not Part of an MD Package.

3.9.1. DCS Acceleration. The direct Coulomb summation (DCS) algorithm for direct Coulomb potential calculation (Section 2.1) on the lattice is an ideal algorithm for acceleration using GPUs. This is due to the high degree of data parallelism, intensive arithmetic operations, and simple data structures in DCS, which are highly suitable for the GPU architecture.[33] The GPU implementation of DCS on NVIDIA GeForce 8800 GTX can achieve a performance of more than 16× as compared to the optimized CPU version on an Intel Core 2 quad-core 2.66 GHz processor. This performance of the GPU can be further enhanced to 40× using different algorithmic and architectural improvements.[47]

Techniques for Acceleration. Grid Partitioning. The potential map grid of the simulated system is divided into planes, and each of the planes is mapped to the grid on a GPU. Different planes are distributed among multiple GPUs for parallel computation. Each of the 2D planes is further divided and assigned to a thread block on the GPU. The number of threads in a block is dependent on the shared memory, the registers, and resources that each thread consumes.[33]

Overlapping Arithmetic Operations and Memory Accesses. For distance calculation, some of the coordinate components are constant for atoms in individual planes. Hence, they can be reused to find the potential contribution to different points on a plane. When the potential contribution of atoms in the x-direction is to be computed, the sum of y,z components of distance calculation is calculated once for an atom and reused for the rest of the atoms. Thus, the increased arithmetic operations can mask the delay in the memory reference. With the increase in the number of lattice points to which an atom's contribution is calculated, the load on constant memory reduces, increasing the number of registers used to store partial results. So the number of registers used per thread limits the number of points to which an atom's contribution can be calculated at the same time.

This bottleneck is mitigated by using shared memory, and the registers are used to store the partial results. The use of shared memory and registers helps increase the number of lattice points to which an atom's contribution is found but increases the computational tile size and, in turn, the size of the thread block. Performance degrades when the potential map of the simulated system is not divisible by the tile size and results in the addition of padding. This additional padding results in unwanted arithmetic operations and a degradation in performance for smaller potential maps.[33,47]

Concurrent Memory Subsystem Usage. To provide data to the vast number of arithmetic units available in the GPU, simultaneous independent access of constant, shared, and global memory was done.[33]

3.9.2. FMM Acceleration on GPU. The fast multipole method (FMM) is a very suitable algorithm (Section 2.5) for highly parallel exascale systems available today due to its linear complexity and high arithmetic intensity.[147] It is scalable to thousands of GPUs, making petascale-computing possible. In this algorithm, the system simulated is divided into subdomains that map to the branches of a tree. The FMM algorithm consists of six main kernels, and each of the kernels is evaluated on the GPU.

Table 2. Comparison of Advantages of Electrostatic Algorithms Implemented in MD Packages

EFC algorithm	advantages	disadvantages
PME	1. The performance in GROMACS reaches $O(1000)$ steps per second.	1. Memory becomes a constraint when large simulation boxes are used due the to memory access nature of FFT.
	2. Rapid convergence of energy with high accuracy using mixed precision and highly optimized GPU implementation in GROMACS using optimization strategies (refer to Section 3.4).	2. FFT requires all-to-all communication and hence becomes a bottleneck for scalability. PME scaling reaches a flat level at 1024 processes.[139,140]
MSM	1. Based on spatial scales, it provides the most suitable memory access patterns.	1. In single CPU implementation, MSM achieves a performance of 41% and 47% for orders 4 and 6, respectively, compared to PME. Increasing the number of levels in MSM results in a moderate penalty as compared to the efficiency of FFTW18.
	2. Highly scalable and hence most suitable for large-scale simulations running on GPUs and/or clusters.	2. When a dynamically varying simulation box is used, the efficiency of the real space grid calculation becomes an issue as it depends on the spatial extent of the convolution kernel stencil.
	3. 3D convolutions performed in MSM are readily vectorized, while FFTs are not. This is advantageous for GPU implementations of MSM and in CPU using technology such as AVX2 (Intel advanced vector extensions).	
	4. Readily applicable for periodic, nonperiodic, and semiperiodic system.	
FMM	1. FMM performs well when used for simulation of a large number of particles, large simulation boxes, inhomogeneous charge distributions, and high parallelization.[116,139]	1. Truncation of the multipole expansion at finite order p (small values) will result in the deviation of forces and energies computed. A multipole order of $p \geqslant 12$ achieves the highest numerical accuracy in single precision computation for FMM.[82]
	2. FMM scales linearly up to ≈270 000 000 and ≈160 000 000 particles in single and double precision.	2. FMM achieves only one-third performance of PME on a single GPU for a periodic salt water solution.
	3. FMM is not limited by simulation box size only by number of particles as memory constraint does apply to FMM algorithmic implementation.	3. The application of FMM to noncubic simulation boxes increases the complexity of the algorithm.

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

Techniques of Acceleration. Orthogonal Recursive Multisection. Workload balancing is one of the most important requirements to achieve maximum performance on a highly parallel platform like the GPU. The decomposition of the simulated molecule is achieved by the orthogonal recursive multisection, which is a variant of the orthogonal recursive bisection method.[148] It forms a binary tree-like structure with an equal number of particles in each bisection. This formation of the binary tree-like structure is done by finding the n^{th} element in a group of N particles. The n^{th} element algorithm is used to divide the domain into several subdomains that are not a power of 2.[149] Rectangular-shaped subdomains called cells are formed by this method.

Dual-Tree Traversal. The cell−cell interactions are obtained by the dual-tree traversal method, which allows the use of rectangular-shaped subdomains.[150] All the particles in a subdomain or cell are assigned to a process. The information on the assigned particles is stored in the memory of that process to reduce the memory requirement. Only the particle data and multipole moments of particles near each subdomain border are to be communicated to the neighboring processes. This tree traversal thus reduces the amount of communication involved.

Local Essential Tree. The particle decomposition forms an oct-tree structure called the global tree. Portions of the global tree, that is, the subdomain, are assigned to each process. However, it requires information from the borders of the other subdomains. These data are communicated between the processes. Each process constructs a local essential tree, a subset of the global tree using the information from the other subdomains. The multipole acceptance criterion (MAC)[81] determines which particle data are to be transferred from the target cell. Based on MAC, a radius R between the center of mass of the source cell and the border of the target cell running on a remote process is determined.

Batch Evaluation. Of the most compute-intensive is particle to particle and multipole to local kernels.[81] Batch evaluation of the two kernels is performed to accelerate the computations, as there is no dependency between these kernels. The source cell is loaded to the shared memory, and the target cell is assigned to a thread block. A thread in the thread block computes each coefficient of the multipole-to-local expansion.

The different acceleration techniques used for electrostatic force calculations on GPUs are summarized in Table 3.

Table 3. Different Methods of Electrostatic Force Calculation (EFC) Acceleration Used by Different MD Packages and Solo Algorithm Implementation (DCS and FMM) on GPU for Improved Performance

MD package	EFC algorithm	methods of acceleration
AMBER[23]	PME	1. Mixed precision arithmetic is used (SPFP).
		2. Neighbor-list creation.
		3. GPU-cluster acceleration.
NAMD[24,98]	S-PME	1. 3D-FFT parallel implementation.
		2. Charm++ programming model.
NAMD[53,108]	MSM	1. Small-bin based geometric hashing.
		2. Sliding window approach.
GROMACS[25,110]	PME	1. Eighth-shell domain decomposition.
		2. Dedicated nodes.
		3. Offload model.
LAMMPS[26]	P^3 M	1. Charge assignment acceleration.
		2. 3D FFT acceleration.
ACEMD[27]	PME	1. Cell-list construction.
		2. Optimized memory access.
DESMOND[18]	PME	1. Midpoint Mmthod.
		2. Task decomposition.
		3. 3D FFT parallelization.
[33,47]	DCS	1. Grid partitioning.
		2. Overlapping arithmetic and memory operations.
[147]	FMM	1. Orthogonal recursive multisection.
		2. Dual-tree traversal.

4. DISCUSSION

This paper compares the accuracy, complexity, and scalability of the electrostatic force calculation algorithms (Section 2) along with the software and hardware techniques used for the algorithms' acceleration on high-performance platforms such as GPUs in the above sections. Accuracy and scalability are the two factors that decide the choice of an electrostatic force calculation algorithm in an MD package. The use of double-precision floating points for arithmetic operations provides very high accuracy but degrades the performance. While the individual discussions of methods and parallelization techniques seek to provide succinct introductions, they should suffice so readers can contrast using the different comparison features provided to make the right choice for their application or area of study.

The performance of specialized supercomputers, such as Anton 2, is based on customized algorithmic implementations and specialized hardware. They provide bleeding-edge performances but remain out of reach for the bulk of researchers. For example, Anton 2 contains 33 792 processor cores and achieve a performance of 85 μs per day for dihydrofolate reductase (DHFR).[151] However, specialized computers are not within the reach of a typical researcher. The acceptable option is to optimize the electrostatic algorithms further to maximize the utilization of high-performance computing platforms.

To our knowledge, the particle mesh Ewald (PME 8) algorithm is the most popular algorithm that is used in the molecular dynamics biomolecular simulations (see ref 152 for benchmarks). The PME implementation on GPU used in GROMACS is one of the fastest. This is due to the efficient utilization of parallelization features available in the GPU and the algorithmic optimizations. Nevertheless, when more than one GPU is to be used in a cluster, the all-to-all communication required in PME becomes a bottleneck. This is because the number of messages exchanged in this communication step scales quadratically with the number of nodes in the cluster.[82] The hybrid PME offload option is also considered in which the 3D FFT which is communication intensive is back-offloaded to the CPU. This will become a viable choice in the next-generation computer which will have high-bandwidth CPU−GPU interconnection, thus allowing the grid transfer overlap.[153] Due to the emergence of the exascale supercomputer, it is vital to consider the algorithm's scalability to achieve extreme acceleration. These exascale supercomputers have led to further research on highly scalable algorithms such as the FMM.

The FMM implementation on GPU can achieve the accuracy of PME with high multipole order. However, it has been found experimentally that the performance of the FMM on GPU for protein systems is only one-third of the

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

performance of PME on GPU. For large sparse systems, the FMM outperforms PME, and it can also handle open boundaries.[82] FMM and MSM are highly scalable but have not been adopted by the popular MD packages due to the increased implementation complexity. If these parameters can be improved, these algorithms can be widely used. Hence, it is an exciting and open area of exploration and opportunity.

5. CONCLUSION

This paper surveys the different methods of electrostatic force calculation and their advantages and disadvantages. Different algorithms are weighted based on the ease of programming, computational complexity, accuracy, and simulation stability. The electrostatic force, which is due to nonbonded interaction, is one of the major performance bottlenecks in MD that is used for simulating the dynamics of biological systems such as proteins, DNAs, and membranes in explicit solvents.

It has been found that efficiency in terms of time and memory can only be achieved by the interplay of software and hardware optimizations. The electrostatic force calculation is now moving toward a hybrid GPU environment, as discussed in the research papers reviewed here. The difference in acceleration occurs due to the different data management strategies used in the algorithm and the hardware features of the platform in which it is implemented.

The paper reviews how the electrostatic force calculation is accelerated by the use of high-performance platforms such as GPUs to achieve long time-scale simulations. The Ewald-based methods such as PME, P^3 M, and MSM are implemented in the most widely used MD packages and special-purpose supercomputers for MD. These platforms can harness the inherent intensive parallelism available in the algorithm due to the fine-grained parallelism available in hardware implementations of GPUs. The ability to overlap computation with communication has been achieved due to customized designs. The review shows that the future of efficient acceleration of MD algorithms depends on the development or adaptability of scientific algorithms that can harness the modern high-performance computing platforms.

AUTHOR INFORMATION

Corresponding Author

Anu George − *Department of Computer Science and Engineering, Amrita School of Engineering, Bengaluru 560035 Amrita Vishwa Vidyapeetham, India*; Email: g_anu@blr.amrita.edu

Authors

Sandip Mondal − *CubeBio AI, Bengaluru 560100, India*

Madhura Purnaprajna − *Department of Computer Science and Engineering, PES University, Bengaluru 560085, India*

Prashanth Athri − *Department of Computer Science and Engineering, Amrita School of Engineering, Bengaluru 560035 Amrita Vishwa Vidyapeetham, India*; orcid.org/0000-0002-5731-0291

Complete contact information is available at: https://pubs.acs.org/10.1021/acsomega.2c03189

Funding

This work was supported by the Department of Science and Technology (DST), Government of India, National Supercomputing Mission (NSM), Reference Number: DST/NSM/R&D HPC Applications/2021/03.41.

Notes

The authors declare no competing financial interest.

REFERENCES

(1) Alder, B. J.; Wainwright, T. E. Studies in molecular dynamics. I. General method. *J. Chem. Phys.* **1959**, *31*, 459−466.

(2) Adcock, S. A.; McCammon, J. A. Molecular dynamics: survey of methods for simulating the activity of proteins. *Chem. Rev.* **2006**, *106*, 1589−1615.

(3) Patel, C. N.; Goswami, D.; Jaiswal, D. G.; Parmar, R. M.; Solanki, H. A.; Pandya, H. A. Pinpointing the potential hits for hindering interaction of SARS-CoV-2 S-protein with ACE2 from the pool of antiviral phytochemicals utilizing molecular docking and molecular dynamics (MD) simulations. *J. Mol. Graphics Modell.* **2021**, *105*, 107874.

(4) Padhi, A. K.; Rath, S. L.; Tripathi, T. Accelerating COVID-19 research using molecular dynamics simulation. *J. Phys. Chem. B* **2021**, *125*, 9078−9091.

(5) Pushkaran, A. C.; NatheN, P.; Melge, A. R.; Puthiyedath, R.; Mohan, C. G. A phytochemical-based medication search for the SARS-CoV-2 infection by molecular docking models towards spike glycoproteins and main proteases. *RSC Adv.* **2021**, *11*, 12003−12014.

(6) Rahman, M. M.; Saha, T.; Islam, K. J.; Suman, R. H.; Biswas, S.; Rahat, E. U.; Hossen, M. R.; Islam, R.; Hossain, M. N.; Mamun, A. A.; et al. Virtual screening, molecular dynamics and structure-activity relationship studies to identify potent approved drugs for Covid-19 treatment. *J. Biomol. Struct. Dyn.* **2021**, *39*, 6231−6241.

(7) Aouidate, A.; Ghaleb, A.; Chtita, S.; Aarjane, M.; Ousaa, A.; Maghat, H.; Sbai, A.; Choukrad, M.; Bouachrine, M.; Lakhlifi, T. Identification of a novel dual-target scaf- fold for 3CLpro and RdRp proteins of SARS-CoV-2 using 3D-similarity search, molecular docking, molecular dynamics and ADMET evaluation. *J. Biomol. Struct. Dyn.* **2021**, *39*, 4522−4535.

(8) Alazmi, M.; Motwalli, O. In silico virtual screening, characterization, docking and molecular dynamics studies of crucial SARS-CoV-2 proteins. *J. Biomol. Struct. Dyn.* **2021**, *39*, 6761−6771.

(9) Selvaraj, J.; Sundar P, S.; Rajan, L.; Selvaraj, D.; Palanisamy, D.; Namboori, K.; Mohankumar, S. K. Identification of (2 R, 3 R)-2-(3, 4-dihydroxyphenyl) chroman-3-yl-3,4, 5-trihydroxy benzoate as multiple inhibitors of SARS-CoV-2 targets; a systematic molecular modelling approach. *RSC Adv.* **2021**, *11*, 13051−13060.

(10) Cruz, F. J.; de Pablo, J. J.; Mota, J. P. Endohedral confinement of a DNA dodecamer onto pristine carbon nanotubes and the stability of the canonical B form. *J. Chem. Phys.* **2014**, *140*, 225103.

(11) Cruz, F. J.; Mota, J. P. Conformational Thermodynamics of DNA Strands in Hy- drophilic Nanopores. *J. Phys. Chem. C* **2016**, *120*, 20357−20367.

(12) Duran, T.; Minatovicz, B.; Bai, J.; Shin, D.; Mohammadiarani, H.; Chaudhuri, B. Molecular dynamics simulation to uncover the mechanisms of protein instability during freezing. *J. Pharm. Sci.* **2021**, *110*, 2457−2471.

(13) George, A.; Purnaprajna, M.; Athri, P. Laplacian score and genetic algorithm based automatic feature selection for Markov State Models in adaptive sampling based molecular dynamics. *PeerJ. Phys. Chem.* **2020**, *2*, No. e9.

(14) Tao, P.; Xiao, Y. Role of cotranslational folding for β-sheet-enriched proteins: A perspective from molecular dynamics simulations. *Phys. Rev. E* **2022**, *105*, 024402.

(15) Zheng, H.; Wang, S.; Zhang, Y. Increasing the time step with mass scaling in Born- Oppenheimer ab initio QM/MM molecular dynamics simulations. *J. Comput. Chem.* **2009**, *30*, 2706−2711.

(16) Ryckaert, J.-P.; Ciccotti, G.; Berendsen, H. J. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *J. Comput. Phys.* **1977**, *23*, 327−341.

(17) Stocker, U.; Juchli, D.; van Gunsteren, W. F. Increasing the time step and efficiency of molecular dynamics simulations: Optimal solutions for equilibrium simulations or structure refinement of large biomolecules. *Mol. Simul.* **2003**, *29*, 123−138.

(18) Bowers, K. J.; Chow, E.; Xu, H.; Dror, R. O.; Eastwood, M. P.; Gregersen, B. A.; Klepeis, J. L.; Kolossvary, I.; Moraes, M. A.; Sacerdoti, F. D. et al. Scalable algorithms for molecular dynamics simulations on commodity clusters. *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*; 2006; p 84.
(19) Toyoda, S.; Miyagawa, H.; Kitamura, K.; Amisaki, T.; Hashimoto, E.; Ikeda, H.; Kusumi, A.; Miyakawa, N. Development of MD engine: High-speed accelerator with parallel processor design for molecular dynamics simulations. *J. Comput. Chem.* **1999**, *20*, 185−199.
(20) Rapaport, D. C.; Rapaport, D. C. R. *The art of molecular dynamics simulation*; Cambridge university press, 2004.
(21) Piana, S.; Lindorff-Larsen, K.; Dirks, R. M.; Salmon, J. K.; Dror, R. O.; Shaw, D. E. Evaluating the effects of cutoffs and treatment of long-range electrostatics in protein folding simulations. *PLoS One* **2012**, *7*, No. e39918.
(22) Deng, H.; Li, X.; Liu, X.; Wang, G. Accelerating the near non-bonded force computation in desmond with graphic processing units. *Parallel Processing Workshops (ICPPW), 2011 40th International Conference*; 2011; pp 191−198.
(23) Pearlman, D. A.; Case, D. A.; Caldwell, J. W.; Ross, W. S.; Cheatham, T. E., III; DeBolt, S.; Ferguson, D.; Seibel, G.; Kollman, P. AMBER. a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Comput. Phys. Commun.* **1995**, *91*, 1−41.
(24) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kale, L.; Schulten, K. Scalable molecular dynamics with NAMD. *J. Comput. Chem.* **2005**, *26*, 1781−1802.
(25) Pronk, S.; Páll, S.; Schulz, R.; Larsson, P.; Bjelkmar, P.; Apostolov, R.; Shirts, M. R.; Smith, J. C.; Kasson, P. M.; van der Spoel, D.; et al. GROMACS 4.5: a high- throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics* **2013**, *29*, 845−854.
(26) Plimpton, S.; Crozier, P.; Thompson, A. LAMMPS-large-scale atomic/molecular massively parallel simulator. *Sandia Natl. Lab.* **2007**, *18*, 43−43.
(27) Harvey, M. J.; Giupponi, G.; Fabritiis, G. D. ACEMD: accelerating biomolecular dynamics in the microsecond time scale. *J. Chem. Theory Comput.* **2009**, *5*, 1632−1639.
(28) Frenkel, D.; Smit, B. *Understanding molecular simulation: from algorithms to applications*; Elsevier, 2001; Vol. *1*.
(29) Allen, M. P.; Tildesley, D. J. *Computer simulation of liquids*; Oxford university press, 2017.
(30) Fukuda, I.; Nakamura, H. Non-Ewald methods: theory and applications to molecular systems. *Biophys. Rev.* **2012**, *4*, 161−170.
(31) Ewald, P. Evaluation of optical and electrostatic lattice potentials. *Ann. Phys.* **1921**, *369*, 253−287.
(32) Deserno, M.; Holm, C. How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines. *J. Chem. Phys.* **1998**, *109*, 7678−7693.
(33) Stone, J. E.; Phillips, J. C.; Freddolino, P. L.; Hardy, D. J.; Trabuco, L. G.; Schulten, K. Accelerating molecular modeling applications with graphics processors. *J. Comput. Chem.* **2007**, *28*, 2618−2640.
(34) Skeel, R. D.; Tezcan, I.; Hardy, D. J. Multiple grid methods for classical molecular dynamics. *J. Comput. Chem.* **2002**, *23*, 673−684.
(35) Greengard, L.; Rokhlin, V. A fast algorithm for particle simulations. *J. Comput. Phys.* **1987**, *73*, 325−348.
(36) Dongarra, J.; Sullivan, F. Guest editors' introduction: The top 10 algorithms. *Comput. Sci. Eng.* **2000**, *2*, 22−23.
(37) Černỳ, J.; Hobza, P. Non-covalent interactions in biomacromolecules. *Phys. Chem. Chem. Phys.* **2007**, *9*, 5291−5303.
(38) Hünenberger, P. H.; Börjesson, U.; Lins, R. D. Electrostatic interactions in biomolecular systems. *CHIMIA International Journal for Chemistry* **2001**, *55*, 861−866.
(39) Reif, M. M.; Krautler, V.; Kastenholz, M. A.; Daura, X.; Hunenberger, P. H. Molecular dynamics simulations of a reversibly folding β-heptapeptide in methanol: influence of the treatment of long-range electrostatic interactions. *J. Phys. Chem. B* **2009**, *113*, 3112−3128.
(40) Sagui, C.; Darden, T. A. Molecular dynamics simulations of biomolecules: long-range electrostatic effects. *Annu. Rev. Biophys. Biomol. Struct.* **1999**, *28*, 155−179.
(41) de Leeuw, S. W.; Perram, J. W.; Smith, E. R. Simulation of electrostatic systems in periodic boundary conditions. I. Lattice sums and dielectric constants. Proceedings of the Royal Society of London A: Mathematical. *Physical and Engineering Sciences.* **1980**, 27−56.
(42) Hünenberger, P.; Reif, M. *Single-ion solvation: experimental and theoretical approaches to elusive thermodynamic quantities*; Royal Society of Chemistry, 2011; Vol. 3.
(43) Hünenberger, P. In Simulation and Theory of Electrostatic Interactions in Solution: Computational Chemistry. *Biophysics, and Aqueous Solution* **1999**, 17.
(44) Patra, M.; Karttunen, M.; Hyvvonen, M. T.; Falck, E.; Lindqvist, P.; Vattulainen, I. Molecular dynamics simulations of lipid bilayers: major artifacts due to truncating electrostatic interactions. *Biophys. J.* **2003**, *84*, 3636−3645.
(45) Zhou, R.; Harder, E.; Xu, H.; Berne, B. Efficient multiple time step method for use with Ewald and particle mesh Ewald for large biomolecular systems. *J. Chem. Phys.* **2001**, *115*, 2348−2358.
(46) Mackerell, A. D. Empirical force fields for biological macromolecules: overview and issues. *J. Comput. Chem.* **2004**, *25*, 1584−1604.
(47) Owens, J. D.; Houston, M.; Luebke, D.; Green, S.; Stone, J. E.; Phillips, J. C. GPU computing. *Proc. IEEE* **2008**, *96*, 879−899.
(48) Hardy, D. J.; Stone, J. E.; Vandivort, K. L.; Gohara, D.; Rodrigues, C.; Schulten, K. *GPU Computing Gems Emerald*; Elsevier, 2011; pp 43−58.
(49) Hardy, D. J. *Multilevel summation for the fast evaluation of forces for the simulation of biomolecules*; University of Illinois at Urbana-Champaign, 2006.
(50) Hardy, D. J.; Wolff, M. A.; Xia, J.; Schulten, K.; Skeel, R. D. Multilevel summation with B-spline interpolation for pairwise interactions in molecular dynamics simulations. *J. Chem. Phys.* **2016**, *144*, 114112.
(51) Novalbos, M.; Gonźalez, J.; Otaduy, M. A.; Martinez-Benito, R.; Sanchez, A. Scalable On-Board Multi-GPU Simulation of Long-Range Molecular Dynamics. *Euro-Par.* **2014**, 752−763.
(52) Tameling, D.; Springer, P.; Bientinesi, P.; Ismail, A. E. Multilevel summation for dispersion: A linear-time algorithm for r- 6 potentials. *J. Chem. Phys.* **2014**, *140*, 024105.
(53) Hardy, D. J.; Wu, Z.; Phillips, J. C.; Stone, J. E.; Skeel, R. D.; Schulten, K. Multilevel summation method for electrostatic force evaluation. *J. Chem. Theory Comput.* **2015**, *11*, 766−779.
(54) Kaufmann, T.; Fumeaux, C.; Vahldieck, R. The meshless radial point interpolation method for time-domain electromagnetics. Microwave Symposium Digest. *2008 IEEE MTT-S International.* **2008**, 61−64.
(55) Problem I. Multidomain Pattern II. Driving Forces III. Solution 1. Multiple Domain.
(56) Patra, M.; Hyvvonen, M. T.; Falck, E.; Sabouri-Ghomi, M.; Vattulainen, I.; Karttunen, M. Long-range interactions and parallel scalability in molecular simulations. *Comput. Phys. Commun.* **2007**, *176*, 14−22.
(57) Toukmaji, A. Y.; Board, J. A. Ewald summation techniques in perspective: a survey. *Comput. Phys. Commun.* **1996**, *95*, 73−92.
(58) Essmann, U. U.; Essmann, L.; Perera, M. L.; Berkowitz, T.; Darden, H.; Lee, L. G.; Pedersen, J. A smooth particle mesh Ewald method. *J. Chem. Phys.* **1995**, *103*, 8577.
(59) Crowley, M.; Darden, T.; Cheatham, T.; Deerfield, D. Adventures in improving the scaling and accuracy of a parallel molecular dynamics program. *J. Supercomput.* **1997**, *11*, 255−278.
(60) Darden, T.; York, D.; Pedersen, L. Particle mesh Ewald: An N log (N) method for Ewald sums in large systems. *J. Chem. Phys.* **1993**, *98*, 10089−10092.

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

(61) Petersen, H. G. Accuracy and efficiency of the particle mesh Ewald method. *J. Chem. Phys.* **1995**, *103*, 3668−3679.
(62) Plimpton, S.; Pollock, R.; Stevens, M. Particle-Mesh Ewald and rRESPA for Parallel Molecular Dynamics Simulations. *PPSC* **1997**.
(63) Kaĺe, L.; Skeel, R.; Bhandarkar, M.; Brunner, R.; Gursoy, A.; Krawetz, N.; Phillips, J.; Shinozaki, A.; Varadarajan, K.; Schulten, K. NAMD2: greater scalability for parallel molecular dynamics. *J. Comput. Phys.* **1999**, *151*, 283−312.
(64) Luty, B. A.; Davis, M. E.; Tironi, I. G.; Van Gunsteren, W. F. A comparison of particle-particle, particle-mesh and Ewald methods for calculating electrostatic interactions in periodic molecular systems. *Mol. Simul.* **1994**, *14*, 11−20.
(65) Hockney, R. W.; Eastwood, J. W. *Computer simulation using particles*; CRC Press, 1988.
(66) Ferrell, R.; Bertschinger, E. Particle-mesh methods on the Connection Machine. *Int. J. Mod. Phys. C* **1994**, *5*, 933−956.
(67) Arnold, A.; Fahrenberger, F.; Holm, C.; Lenz, O.; Bolten, M.; Dachsel, H.; Halver, R.; Kabadshow, I.; Gähler, F.; Heber, F.; et al. Comparison of scalable fast methods for long-range interactions. *Phys. Rev. E* **2013**, *88*, 063308.
(68) Neelov, A.; Holm, C. Interlaced P3M algorithm with analytical and ik-differentiation. *J. Chem. Phys.* **2010**, *132*, 234103.
(69) Brown, W. M.; Kohlmeyer, A.; Plimpton, S. J.; Tharrington, A. N. Implementing molecular dynamics on hybrid high performance computers-Particle-particle particlemesh. *Comput. Phys. Commun.* **2012**, *183*, 449−459.
(70) Shirokov, A.; Bertschinger, E. GRACOS: Scalable and Load Balanced P3M Cosmological N-body Code. *Astrophysics Source Code Library* **2010**, ascl-1010.
(71) Lacava, F. *Classical Electrodynamics*; Springer, 2016; pp 17−31.
(72) Dzwinel, W.; Boryczko, K.; Yuen, D. A. Modeling mesoscopic fluids with discrete-particles-methods, algorithms, and results. *Surfactant sci. ser.* **2006**, *130*, 715.
(73) Jackson, J. D. *Classical Electrodynamics*; John Wiley & Sons, 2012.
(74) Schmidt, K. E.; Lee, M. A. Implementing the fast multipole method in three dimensions. *J. Stat. Phys.* **1991**, *63*, 1223−1235.
(75) Ying, L.; Biros, G.; Zorin, D. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *J. Comput. Phys.* **2004**, *196*, 591−626.
(76) Field, M. J. Technical advances in molecular simulation since the 1980s. *Arch. Biochem. Biophys.* **2015**, *582*, 3−9.
(77) White, C. A.; Head-Gordon, M. Derivation and efficient implementation of the fast multipole method. *J. Chem. Phys.* **1994**, *101*, 6593−6605.
(78) Darve, E. The fast multipole method: numerical implementation. *J. Comput. Phys.* **2000**, *160*, 195−240.
(79) Darve, E. The fast multipole method I: error analysis and asymptotic complexity. *SIAM J. Numer. Anal.* **2000**, *38*, 98−128.
(80) Andoh, Y.; Yoshii, N.; Fujimoto, K.; Mizutani, K.; Kojima, H.; Yamada, A.; Okazaki, S.; Kawaguchi, K.; Nagao, H.; Iwahashi, K.; et al. MODYLAS: A highly parallelized general-purpose molecular dynamics simulation program for large-scale systems with long-range forces calculated by fast multipole method (FMM) and highly scalable fine-grained new parallel processing algorithms. *J. Chem. Theory Comput.* **2013**, *9*, 3201−3209.
(81) Yokota, R.; Barba, L. A.; Narumi, T.; Yasuoka, K. Petascale turbulence simulation using a highly parallel fast multipole method on GPUs. *Comput. Phys. Commun.* **2013**, *184*, 445−455.
(82) Kohnke, B.; Kutzner, C.; Grubmuller, H. A GPU-accelerated fast multipole method for GROMACS: Performance and accuracy. *J. Chem. Theory Comput.* **2020**, *16*, 6938−6949.
(83) Owens, J. D.; Luebke, D.; Govindaraju, N.; Harris, M.; Krüger, J.; Lefohn, A. E.; Purcell, T. J. A survey of general-purpose computation on graphics hardware. *Computer graphics forum.* **2007**, *26*, 80−113.
(84) Garland, M.; Le Grand, S.; Nickolls, J.; Anderson, J.; Hardwick, J.; Morton, S.; Phillips, E.; Zhang, Y.; Volkov, V. Parallel Comput. experiences with CUDA. *IEEE Micro* **2008**, *28*, 13.
(85) Weiner, P. K.; Kollman, P. A. AMBER: Assisted model building with energy refinement. A general program for modeling molecules and their interactions. *J. Comput. Chem.* **1981**, *2*, 287−303.
(86) Wang, J.; Wolf, R. M.; Caldwell, J. W.; Kollman, P. A.; Case, D. A. Development and testing of a general amber force field. *J. Comput. Chem.* **2004**, *25*, 1157−1174.
(87) Kini, R. M.; Evans, H. J. Molecular modeling of proteins: a strategy for energy minimization by molecular mechanics in the AMBER force field. *J. Biomol. Struct. Dyn.* **1991**, *9*, 475−488.
(88) Makov, G.; Payne, M. Periodic boundary conditions in ab initio calculations. *Phys. Rev. B* **1995**, *51*, 4014.
(89) Haile, J.; Johnston, I.; Mallinckrodt, A. J.; McKay, S. Molecular dynamics simulation: elementary methods. *Comput. Phys.* **1993**, *7*, 625−625.
(90) Nvidia, C. *CUFFT Library*; 2010. https://developer.nvidia.com/cufft Accessed: 2021-05-05.
(91) Le Grand, S.; Gotz, A. W.; Walker, R. C. SPFP: Speed without compromise—A mixed precision model for GPU accelerated molecular dynamics simulations. *Comput. Phys. Commun.* **2013**, *184*, 374−380.
(92) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. An overview of the Amber biomolecular simulation package. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2013**, *3*, 198−210.
(93) Salomon-Ferrer, R.; Gotz, A. W.; Poole, D.; Le Grand, S.; Walker, R. C. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 2. Explicit solvent particle mesh Ewald. *J. Chem. Theory Comput.* **2013**, *9*, 3878−3888.
(94) Moore, G. H. Hilbert on the infinite: The role of set theory in the evolution of Hilbert's thought. *Hist. Math.* **2002**, *29*, 40−64.
(95) Mertz, J. E.; Tobias, D. J.; Brooks, C. L.; Singh, U. Vector and parallel algorithms for the molecular dynamics simulation of macromolecules on shared-memory computers. *J. Comput. Chem.* **1991**, *12*, 1270−1277.
(96) Furlinger, K.; Wright, N. J.; Skinner, D. Comprehensive performance monitoring for gpu cluster systems. *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)* **2011**, 1377−1386.
(97) Lee, T.-S.; Cerutti, D. S.; Mermelstein, D.; Lin, C.; LeGrand, S.; Giese, T. J.; Roitberg, A.; Case, D. A.; Walker, R. C.; York, D. M. GPU-accelerated molecular dynamics and free energy methods in Amber18: performance enhancements and new features. *J. Chem. Inf. Model.* **2018**, *58*, 2043−2050.
(98) Nelson, M. T.; Humphrey, W.; Gursoy, A.; Dalke, A.; Kaĺe, L. V.; Skeel, R. D.; Schulten, K. NAMD: a parallel, object-oriented molecular dynamics program. *Int. J. Supercomput. Appl. High Perform. Comput.* **1996**, *10*, 251−268.
(99) Lee, E. T. A simplified B-spline computation routine. *Computing* **1982**, *29*, 365−371.
(100) Reumann, M.; Fitch, B. G.; Rayshubskiy, A.; Pitman, M. C.; Rice, J. J. Orthogonal recursive bisection as data decomposition strategy for massively parallel cardiac simulations. *Biomed. Technol.* **2011**, *56*, 129−145.
(101) Harvey, M.; De Fabritiis, G. An implementation of the smooth particle mesh Ewald method on GPU hardware. *J. Chem. Theory Comput.* **2009**, *5*, 2371−2377.
(102) Phillips, J. C.; Zheng, G.; Kumar, S.; Kaĺe, L. V. NAMD: Biomolecular simulation on thousands of processors. *Supercomputing, ACM/IEEE 2002 Conference.* 2002; pp 36−36.
(103) Ibeid, H.; Olson, L.; Gropp, W. FFT, FMM, and multigrid on the road to exascale: Performance challenges and opportunities. *J. Parallel Distrib Comput.* **2020**, *136*, 63−74.
(104) Gotz, A. W.; Williamson, M. J.; Xu, D.; Poole, D.; Le Grand, S.; Walker, R. C. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 1. Generalized born. *J. Chem. Theory Comput.* **2012**, *8*, 1542−1555.
(105) Stone, J. E.; Hynninen, A.-P.; Phillips, J. C.; Schulten, K. Early experiences porting the NAMD and VMD molecular simulation and analysis software to GPU-accelerated OpenPOWER platforms.

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

International conference on high performance computing. **2016**, *9945*, 188−206.
(106) Kaĺe, L. V.; Bhandarkar, M.; Brunner, R.; Krawetz, N.; Phillips, J.; Shinozaki, A. NAMD: A case study in multilingual parallel programming. *International Workshop on Languages and Compilers for Parallel Comput.* **1997**, 367−381.
(107) Phillips, J. C.; Stone, J. E.; Schulten, K. Adapting a message-driven parallel application to GPU-accelerated clusters. *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*; 2008; p 8.
(108) Hardy, D. J. *NAMD-lite*; 2009. https://www.ks.uiuc.edu/Development/MDTools/namdlite/.
(109) Barash, D.; Yang, L.; Qian, X.; Schlick, T. Inherent speedup limitations in multiple time step/particle mesh Ewald algorithms. *J. Comput. Chem.* **2003**, *24*, 77−88.
(110) Berendsen, H. J.; van der Spoel, D.; van Drunen, R. GROMACS: a message-passing parallel molecular dynamics implementation. *Comput. Phys. Commun.* **1995**, *91*, 43−56.
(111) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **2015**, *1*, 19−25.
(112) Bowers, K. J.; Dror, R. O.; Shaw, D. E. Overview of neutral territory methods for the parallel evaluation of pairwise particle interactions. *Journal of Physics: Conference Series*; 2005; p 300.
(113) Hess, B.; Kutzner, C.; Van Der Spoel, D.; Lindahl, E. GROMACS 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation. *J. Chem. Theory Comput.* **2008**, *4*, 435−447.
(114) Alexei, I. D. Master's Thesis, Implementation of the Particle Mesh Ewald method on a GPU. Royal Institute of Technology SCI School of Engineering Sciences, Stockholm, Sweden, 2016.
(115) Gruber, C. C.; Pleiss, J. Systematic benchmarking of large molecular dynamics simulations employing GROMACS on massive multiprocessing facilities. *J. Comput. Chem.* **2011**, *32*, 600−606.
(116) Pall, S.; Abraham, M. J.; Kutzner, C.; Hess, B.; Lindahl, E. Tackling exascale software challenges in molecular dynamics simulations with GROMACS. *International Conference on Exascale Applications and Software*; 2014; pp 3−27.
(117) Board, J. A.; Humphres, C. W.; Lambert, C. G.; Rankin, W. T.; Toukmaji, A. Y. *Computational Molecular Dynamics: Challenges, Methods, Ideas*; Springer, 1999; pp 459−471.
(118) Thompson, A. P.; Aktulga, H. M.; Berger, R.; Bolintineanu, D. S.; Brown, W. M.; Crozier, P. S.; in't Veld, P. J.; Kohlmeyer, A.; Moore, S. G.; Nguyen, T. D.; et al. LAMMPS-a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comput. Phys. Commun.* **2022**, *271*, 108171.
(119) Fabregat-Traver, D.; Ismail, A. E.; Bientinesi, P. Accelerating molecular dynamics codes by performance and accuracy modeling. *J. Comput. Science* **2018**, *27*, 77−90.
(120) Brown, D.; Minoux, H.; Maigret, B. A domain decomposition parallel processing algorithm for molecular dynamics simulations of systems of arbitrary connectivity. *Comput. Phys. Commun.* **1997**, *103*, 170−186.
(121) Stone, J. E.; Hardy, D. J.; Ufimtsev, I. S.; Schulten, K. GPU-accelerated molecular modeling coming of age. *J. Mol. Graph. Model.* **2010**, *29*, 116−125.
(122) Van Meel, J. A.; Arnold, A.; Frenkel, D.; Portegies Zwart, S.; Belleman, R. G. Harvesting graphics power for MD simulations. *Mol. Simul.* **2008**, *34*, 259−266.
(123) Trott, C. R.; Winterfeld, L.; Crozier, P. S. General-purpose molecular dynamics simulations on GPU-based clusters. *arXiv preprint* **2010**, DOI: 10.48550/arXiv.1009.4330.
(124) Plimpton, S. Fast parallel algorithms for short-range molecular dynamics. *J. Comput. Phys.* **1995**, *117*, 1−19.
(125) Bowers, K. J.; Dror, R. O.; Shaw, D. E. Zonal methods for the parallel execution of range-limited N-body simulations. *J. Comput. Phys.* **2007**, *221*, 303−329.
(126) Bowers, K. J.; Dror, R. O.; Shaw, D. E. The midpoint method for parallelization of particle simulations. *J. Chem. Phys.* **2006**, *124*, 184109.
(127) Shaw, D. E. A fast, scalable method for the parallel evaluation of distance-limited pairwise particle interactions. *J. Comput. Chem.* **2005**, *26*, 1318−1328.
(128) Snir, M. A note on n-body computations with cutoffs. *Theory Comput. Syst.* **2004**, *37*, 295−318.
(129) Aktulga, H. M.; Fogarty, J. C.; Pandit, S. A.; Grama, A. Y. Parallel reactive molecular dynamics: Numerical methods and algorithmic techniques. *Parallel Comput.* **2012**, *38*, 245−259.
(130) Shan, Y.; Klepeis, J. L.; Eastwood, M. P.; Dror, R. O.; Shaw, D. E. Gaussian split Ewald: A fast Ewald mesh method for molecular simulation. *J. Chem. Phys.* **2005**, *122*, 054101.
(131) Dror, R.; Grossman, J.; Mackenzie, K.; Towles, B.; Chow, E.; Salmon, J.; Young, C.; Bank, J.; Batson, B.; Shaw, D.; et al. Overcoming communication latency barriers in massively parallel scientific computation. *IEEE Micro* **2011**, *31*, 8−19.
(132) Murray, L. GPU acceleration of Runge-Kutta integrators. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 94−101.
(133) Hünenberger, P. H.; McCammon, J. A. Effect of artificial periodicity in simulations of biomolecules under Ewald boundary conditions: a continuum electrostatics study. *Biophys. Chem.* **1999**, *78*, 69−88.
(134) Hünenberger, P. H.; McCammon, J. A. Ewald artifacts in computer simulations of ionic solvation and ion-ion interaction: a continuum electrostatics study. *J. Chem. Phys.* **1999**, *110*, 1856−1872.
(135) Hub, J. S.; de Groot, B. L.; Grubmuller, H.; Groenhof, G. Quantifying artifacts in Ewald simulations of inhomogeneous systems with a net charge. *J. Chem. Theory Comput.* **2014**, *10*, 381−390.
(136) Bogusz, S.; Cheatham, T. E., III; Brooks, B. R. Removal of pressure and free energy artifacts in charged periodic systems via net charge corrections to the Ewald potential. *J. Chem. Phys.* **1998**, *108*, 7070−7084.
(137) Ohno, Y.; Yokota, R.; Koyama, H.; Morimoto, G.; Hasegawa, A.; Masumoto, G.; Okimoto, N.; Hirano, Y.; Ibeid, H.; Narumi, T.; et al. Petascale molecular dynamics simulation using the fast multipole method on K computer. *Comput. Phys. Commun.* **2014**, *185*, 2575−2585.
(138) Moore, S. G.; Crozier, P. S. Extension and evaluation of the multilevel summation method for fast long-range electrostatics calculations. *J. Chem. Phys.* **2014**, *140*, 234112.
(139) Kutzner, C.; Van Der Spoel, D.; Fechner, M.; Lindahl, E.; Schmitt, U. W.; De Groot, B. L.; Grubmüller, H. Speeding up parallel GROMACS on high-latency networks. *J. Comput. Chem.* **2007**, *28*, 2075−2084.
(140) Kutzner, C.; Apostolov, R.; Hess, B.; Grubmüller, H. *Parallel Comput.: Accelerating computational science and engineering (CSE)*; IOS Press, 2014; pp 722−727.
(141) Izaguirre, J. A.; Hampton, S. S.; Matthey, T. Parallel multigrid summation for the N-body problem. *J. Parallel Distrib. Comput.* **2005**, *65*, 949−962.
(142) Hardy, D. J.; Stone, J. E.; Schulten, K. Multilevel summation of electrostatic potentials using graphics processing units. *Parallel Comput.* **2009**, *35*, 164−177.
(143) Bergdorf, M.; Robinson-Mosher, A.; Guo, X.; Law, K.-H.; Shaw, D. E. Desmond/GPU performance as of April 2021. *DE Shaw Research, Technol. Rep. DESRES/TR-2021−01* **2021**.
(144) AMBER. *GPU Benchmarks*. http://ambermd.org/gpus16/benchmarks.htm Accessed: 2022-07-22.
(145) *Acemd v3 benchmarks*. https://hpc.nih.gov/apps/acemd/benchmarks.html Accessed: 2022-07-23.
(146) *Creating Faster Molecular Dynamics Simulations with GROMACS 2020 — NVIDIA Technical Blog* Accessed: 2022-07-23.
(147) Yokota, R.; Barba, L. A. Hierarchical n-body simulations with autotuning for heterogeneous systems. *Comput. Sci. Eng.* **2012**, *14*, 30−39.

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896

(148) Warren, M. S.; Salmon, J. K. Astrophysical N-body simulations using hierarchical tree data structures. *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*; 1992; pp 570−576.
(149) Urbanek, F. J. An O (log n) algorithm for computing the nth element of the solution of a difference equation. *Inf. Process. Lett.* **1980**, *11*, 66−67.
(150) Dehnen, W. A hierarchical O (N) force calculation algorithm. *J. Comput. Phys.* **2002**, *179*, 27−42.
(151) Shaw, D. E.; Grossman, J.; Bank, J. A.; Batson, B.; Butts, J. A.; Chao, J. C.; Deneroff, M. M.; Dror, R. O.; Even, A.; Fenton, C. H. et al. Anton 2: raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*; 2014; pp 41−53.
(152) Xinhuai, Z. How Fast Can Amber and Gromacs Job Run with P100 Gpu Accelerator. https://nusit.nus.edu.sg/technus/hpc/benchmark-p100-gpu-accelerator-molecular-simulation-application-amber-gromacs/ Accessed: 2021-10-29.
(153) Páll, S.; Zhmurov, A.; Bauer, P.; Abraham, M.; Lundborg, M.; Gray, A.; Hess, B.; Lindahl, E. Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS. *J. Chem. Phys.* **2020**, *153*, 134110.

https://doi.org/10.1021/acsomega.2c03189
ACS Omega 2022, 7, 32877−32896