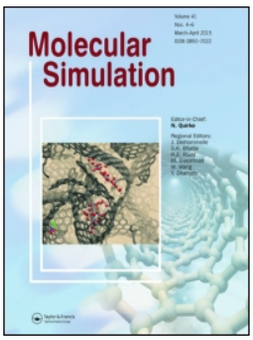

Molecular Simulation

ISSN: 0892-7022 (Print) 1029-0435 (Online) Journal homepage: https://www.tandfonline.com/loi/gmos20

Accelerating multiple replica molecular dynamics simulations using the Intel® Xeon Phi™ coprocessor

Conor Parks, Lei Huang, Yang Wang & Doraiswami Ramkrishna

To cite this article: Conor Parks, Lei Huang, Yang Wang & Doraiswami Ramkrishna (2017) Accelerating multiple replica molecular dynamics simulations using the Intel® Xeon Phi™ coprocessor, Molecular Simulation, 43:9, 714-723, DOI: 10.1080/08927022.2017.1301666

To link to this article: https://doi.org/10.1080/08927022.2017.1301666

MOLECULAR SIMULATION, 2017
VOL. 43, NO. 9, 714–723
http://dx.doi.org/10.1080/08927022.2017.1301666

Taylor & Francis
Taylor & Francis Group

Accelerating multiple replica molecular dynamics simulations using the Intel® Xeon Phi™ coprocessor

Conor Parks[a], Lei Huang[b†], Yang Wang[b†1] and Doraiswami Ramkrishna[a]

[a]School of Chemical Engineering, Purdue University, West Lafayette, IN, USA; [b]Texas Advanced Computing Center, The University of Texas at Austin, Austin, TX, USA

ABSTRACT

We investigate the performance increase provided by the Intel® Xeon Phi™ coprocessor in multiple replica molecular dynamics applications using a novel parallelisation scheme. The benefits of the proposed parallelisation scheme are demonstrated by glycine in water, a system of significant interest in the crystallisation simulation community. The molecular dynamics (MD) engine consists of initially serial LAMMPS and NAMD subroutines, and is subsequently modified and parallelised using a heterogeneous programming model, where each MPI rank is paired with a unique Intel® Xeon Phi™ coprocessor and CPU socket. The MD engine is parallelised using an OpenMP atom domain decomposition algorithm on the Intel® Xeon Phi™ coprocessor and OpenMP task parallelism on the host CPU socket. Using nodes with two Intel® Xeon Phi™ coprocessors, one per socket, we demonstrate that a factor of five reduction in the required computational resources is achieved per replica with the coprocessor, when compared against employing the standard spatial domain decomposition algorithm with no accelerator. Furthermore, the proposed parallelisation scheme achieves ideal weak scaling with respect to the number of employed MPI ranks (replicas). The Intel® Xeon Phi™ coprocessor not only allows us to the increase performance output per socket by a factor of five, when compared against no accelerators, but also significantly reduces the parallelisation complexity necessary to achieve this performance, as the Intel® Xeon Phi™ coprocessor operates using the simple OpenMP programming language.

ARTICLE HISTORY
Received 1 December 2016
Accepted 25 February 2017

KEYWORDS
Molecular dynamics; Intel® Xeon Phi™ coprocessor; multiple replica; parallelisation; OpenMP; MPI

1. Introduction

Direct calculation of thermodynamic properties via molecular dynamics (MD) increasingly requires sampling high- and low-energy states. As high-energy states correspond to low Boltzmann weights, straightforward MD requires inaccessibly long simulation lengths to ensure low probability states are visited sufficiently. Furthermore, kinetic barriers to transitions between metastable states frequently prevent convergent sampling of phase space and even low-energy states and hence prevent accurate thermodynamics and kinetic property estimation using conventional MD. To overcome this sampling issue, algorithms – such as replica exchange [3–5], transition interface sampling [6,7], seeded cluster [8–11] and umbrella sampling [12–15] – have been developed. In all of the aforementioned algorithms, many independent stochastic MD trajectories are launched to infer the desired material properties such as nucleation rates or protein folding structures. Although these simulation methodologies provide more accurate sampling of phase space and free energy estimates, they come at the cost of requiring large computational resources. Instead of having to parallelise 1 MD simulation, the scientist must now parallelise each of the N MD trajectories, where N can frequently be on the order of 100 or more, requiring critical thought as to how to use the computational resources available. Traditionally, MPI based spatial domain decomposition algorithms have been employed in the field of molecular dynamics to parallelise simulations, where the total spatial simulation domain is decomposed in smaller subsystems, each of which is mapped to a unique individual CPU core [1,2,16,17]. By doing this, each CPU core has to calculate the force on the subset of the atoms present in that subsystem spatial domain. Over the past decade, modern many-core architectures, such as the Intel® Many Integrated Core (MIC) architecture present on the Intel® Xeon Phi™ coprocessor, have begun to provide teraflop performance on a single accelerator present on a single CPU socket. Through the MIC's 60 cores and 240 thread architecture with 512-bit vector processing units, these accelerators are designed for algorithms that are massively parallel and make extensive use of single instruction multiple data (SIMD) operations, making them ideal for accelerating MD simulations [18,19]. To help alleviate the cost of performing multiple replica simulations, one could pair each replica with a unique MIC and perform a combination of task parallelism between the MIC and host CPU cores, and an atom decomposition parallelisation on the coprocessor itself, where each of the 240 threads would calculate the force on a subgroup of atoms, regardless of their spatial location. As an increasing number of supercomputers possess nodes where each socket contains an Intel® Xeon Phi™ coprocessor, this

CONTACT Doraiswami Ramkrishna ramkrish@ecn.purdue.edu
† These authors contributed equally.
[1]Present address: Pittsburgh Supercomputing Center, 300 South Craig Street, Pittsburgh, PA 15213.

allows the possibility of running two MD replicas per node, with only 8 – 10 CPU cores being required per replica. This reduces the total number of CPU cores required to perform a multiple replica simulation and hence reduces the total cost of the simulation itself.

In this article, we examine the performance of a heterogeneous programming model, employing a combination of host CPUs and an Intel® Xeon Phi™ coprocessor, for multiple replica molecular dynamics simulations. The objective is to demonstrate how the minimum number of CPU cores and MICs can be used to achieve near optimal performance per simulation replica, thus minimising the total cost of the multiple replica simulation. We demonstrate the performance of the parallelisation scheme using a glycine in water system. This same molecular system was recently used to perform a large-scale multiple replica nucleation simulation [11], making it an excellent case study for the proposed parallelisation scheme. The code consists of initially serial LAMMPS [14] subroutine, with the addition of the NAMD [15] particle mesh Ewald (PME) solver, and is subsequently parallelised using atom domain decomposition and task parallelism OpenMP strategies designed for the MIC and host CPUs. Through a combination of benefiting from the MIC performance, asynchronous task calculations between the MIC and CPU cores, and task parallelism on the CPU cores, we demonstrate that a factor of five reduction in the cost of the simulation for each MD replica can be achieved through the use of the Intel® Xeon Phi™ coprocessor. The study is organised as follows: Section 2 provides an introduction to MD; Section 3 discusses about the algorithms and optimisations employed on the MIC and host CPUs; Section 4 presents performance benchmark results against LAMMPS using various numbers of MPI ranks and discusses using the proposed parallelisation scheme on the future MIC architecture known as Knight's Landing (KNL); finally, Section 5 presents the conclusions of this article.

2. Molecular dynamics background

MD is a methodology to simulate the dynamical behaviour of systems with fully atomistic detail. As such, MD is becoming increasingly employed to study the physics of condensed matter systems present in pharmaceutical crystallisation [9–11,20–30], where atomic interactions between solute, solvent and anti-solvent molecules dictate key quantities such as nucleation, growth and dissolution rates of polymorph-specific crystals. Contrary to its density functional theory (DFT) counterpart, MD employs a classical representation of atomic forces, allowing the simulation of much larger systems and time scales. To allow for a classical representation of inherently quantum mechanical systems, parametric force fields have been developed which seek to reproduce DFT simulation or experimental results. The GAFF potential, one of the many potentials in the MD literature, is the force field employed in this article, and is shown in Equation (1) and is illustrated in Figure 1:

$$\begin{aligned}\phi_{GAFF} = &\sum_{i=1}^{nbonds} b_i\left(r_i - r_{i,eq}\right)^2 + \sum_{i=1}^{nangles} a_i\left(\theta_i - \theta_{i,eq}\right)^2 \\ &+ \sum_{i=1}^{ndihedrals} \frac{V_i}{2}\left[1 + \cos(n\phi - y_{i,n})\right] \\ &+ \sum_{i<j}^{N} \varepsilon_{ij}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{r_{ij}}\right)\end{aligned} \tag{1}$$

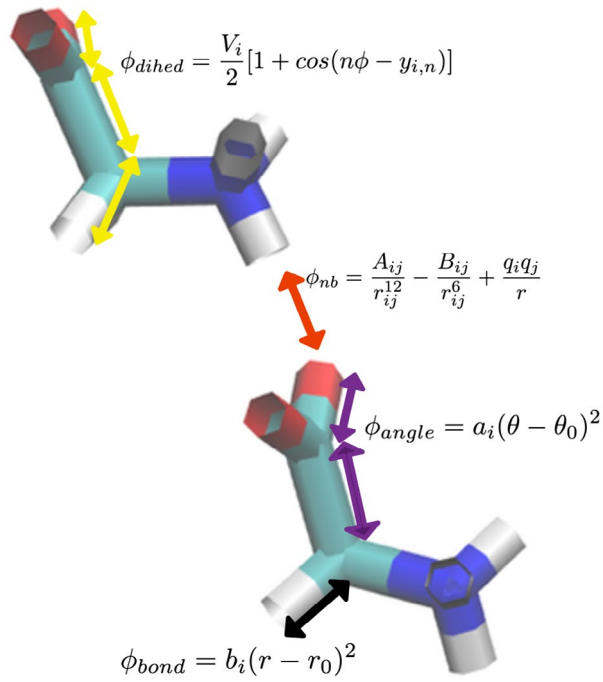

Figure 1. (Colour online) Pair potential overview.
Note: Illustration of bonded (1–2) pairs and potential (black arrow), angle (1–3) pairs and potential (purple arrow), dihedral (1–4) pairs and potential (yellow arrow) for the case of the molecule glycine.

b_i, a_i, and V_i are force constants; $r_{i,eq}$, $\theta_{i,eq}$, are equilibrium distances; $y_{i,n}$ is a phase shift term; f_{ij} is a factor modulating the strength of van der Waals (vdW) and Coulombic forces between bonded pairs; A_{ij} and B_{ij} are diatomic parameters; q_i is the partial atomic charge of atom i; and r_{ij} is the distance between atoms i and j. In the above force field, the dihedral summation includes improper dihedrals. In MD codes, pair lists that are generated at run time store 1–2 (bonds), 1–3 (angles) and 1–4 (dihedral) atom pairs [1,2,16,17]. Looping over these lists to compute the above intramolecular summations in Equation (1) make the computation have O(n) scaling. However, the vdW and Coulombic interactions require looping over all atomic pairs, which is an $O(n^2)$ calculation, rendering it intractable for increasingly large system sizes. As the vdW interactions go to zero with $1/r^6$, the vdW potential can be truncated at 8–12 $\dot{A}$ without significant loss of accuracy in the total force computation. The Coulombic contribution to the potential decays with $1/r$ however, and any attempts at truncations lead to spurious results. To overcome this, various methods, such as the Ewald sum, PME [31] and PPPM [32] algorithms, construct solutions to the governing Poisson's equation PDE for the electrostatic potential by decomposing the total electrostatic potential into three electrostatic contributions: the real space, reciprocal space and a self-energy term. This is shown below in Equation (2):

$$\begin{aligned}V_{\text{coulomb}} = &\frac{1}{2}\sum_{i,j}\sum_{m\in Z^3} q_i q_j \frac{erfc\left(\alpha\left|\boldsymbol{r}_{ij} + \boldsymbol{m}L\right|\right)}{\left|\boldsymbol{r}_{ij} + \boldsymbol{m}L\right|} \\ &+ \frac{1}{2L^3}\sum_{k\neq 0}\frac{4\pi}{k^2} exp\left(-\frac{k^2}{4a^2}\right)\left|\tilde{\rho}(\boldsymbol{k})\right|^2 - \frac{\alpha}{\sqrt{\pi}}\sum_i q_i^2\end{aligned} \tag{2}$$

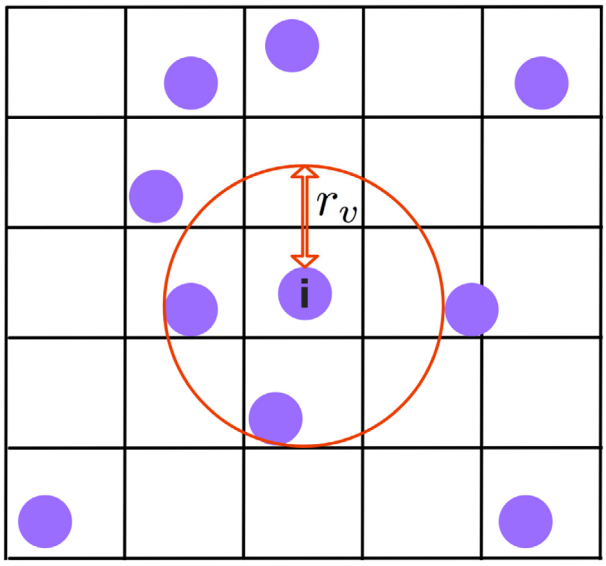

Figure 2. (Colour online) Neighbour list build overview.
Note: The total simulation domain is divided into subdomains whose box lengths are greater than or equal to r_v. Particles are then sorted into these subdomains based off of their spatial coordinates. To build the neighbour list of atom *i*, only the neighbouring cells and the cell of atom *i* need to be looped over.

where α is a damping parameter, $\boldsymbol{r}_{ij}$ is the coordinate vector between atoms *i* and *j*, $\boldsymbol{m}$ is the box translation vector, *L* is the box length, $\widetilde{\rho}(\boldsymbol{k})$ is the Fourier transformed charge density, and *k* is the wave vector. By decomposing the contributions of the electrostatics into the above converging exponentials, a real space cut-off can be applied without much loss in accuracy. The value of α is chosen to be large enough to make it only necessary to employ the minimum image convention in the real space calculation, allowing the real space electrostatic contribution to be calculated in the same routine as the vdW forces: i.e. the short-range non-bonded force calculation. The Fourier transformed charge density can be calculated using FFT numerical routines present in the Intel MLK library package.

Finally, to obtain O(n) scaling in the non-bonded force calculation, Verlet lists [33] are used in conjunction with a linked list binning procedure [34], illustrated in Figure 2. In this method, for each atom *i*, a list of neighbouring atoms *j* is created whose distances from *i* are less than a distance r_v, where $r_v = r_{cut} + r_{skin}$. By binning atoms into cells based off their spatial location using a linked list algorithm [34], distance checks must be performed with atoms in the neighbouring 26 cells and with atoms in the same cell of atom *i*. The width of the cell bins is typically the size of the largest cut-off radius, so that 27 cells must be checked for every atom to compute the neighbour list. This is achieved through looping over a precomputed stenciled list consisting of the indices of all neighboring cells. The end result is that instead of calculating the distance between all atom pairs (*i*,*j*), the majority of which will exceed the cut-off distance r_{cut}, only distances between atom *i* and atoms in the neighbour list of *i* must be calculated. This reduces the algorithmic scaling of the short-range force calculation to O(n). Although computationally intensive, the neighbour list must be rebuilt only whenever any atom diffuses a distance of r_{skin}.

3. Algorithm and optimisation details

In this article, we investigate the performance of a heterogeneous computing model for use in replica exchange or umbrella sampling type simulations in molecular dynamics. Each replica was paired with a unique socket and MIC using MPI. As many supercomputers, such as Stampede, SuperMIC and Conte, have nodes where each socket possesses an Intel® Xeon Phi™ coprocessor, this allows two MPI replicas to be run on a single node. No further MPI programming was used, as the simulated trajectories were embarrassingly parallel. Each individual MPI rank is then parallelised using the OpenMP strategies discussed below. The metrics used to gauge the performance of the proposed parallelisation scheme will be whether ideal weak scaling of the MPI replicas is achieved, and the performance increased provided to each MPI rank through the OpenMP parallelisation.

The neighbour list builds and short-range non-bonded forces are calculated directly on the coprocessor, exploiting the SIMD level parallelism present in these algorithms. A simple atom domain decomposition algorithm using OpenMP was employed on the MIC for these calculations, where blocks of atoms were assigned to unique OpenMP threads, regardless of the atom's spatial location. To optimise the performance for the MIC architecture, an array of structures data format was used for the position and force arrays. This was done to minimise the number of cache lines touched per OpenMP thread during scatter/gather operations in the neighbour list build and non-bonded force kernel on the MIC [35]. Furthermore, single precision double precision (SPDP), mixed precision arithmetic [36] was used throughout. In SPDP, contributions to the short-range non-bonded forces are calculated in single precision, but summed in double precision, and all shake and intramolecular forces are calculated in double precision. This allows the maximisation of the vectorisation throughput on the MIC, without incurring energy drift. The !dir$ simd directive was employed to enforce vectorisation of the inner loop distance calculation in the non-bonded force and neighbour list build routines. All arrays were allocated on 64 byte boundaries to maximise the performance of data loads and stores from the MIC. The !dir$ vector aligned pragma was used to inform the compiler of data alignment in the inner loop of the non-bonded force kernel. To have contiguous memory access and minimise the number of cache misses in the non-bonded force and neighbour list build kernels on the coprocessor, spatial data sorting during neighbour list builds was performed [37]. Newton's third law was not used in the short-range force calculation to reduce the cache misses.

Contrary to a spatial domain decomposition algorithm, it is necessary to implement the minimum image convention at the atom-atom level within the non-bonded and neighbour list routines when using an atom domain decomposition algorithm. To vectorise this calculation, the GNU intrinsic sign and epsilon routines are used. To further aid in the vectorisation of the inner loop of the non-bonded force routine, forces are calculated with all atoms in the neighbour list of a given atom. A mask is set to zero if the atoms are outside the cut-off distance, which prevents these calculations from contributing to the force on an atom, the total energy or the virial. Performing the distance

```
!$omp parallel do schedule(dynamic) reduction(+:potential,ffx,ffy,ffz) default(firstprivate), &
!$omp& shared(position,ff,nlist,numneigh,lj1,lj2,lj3,lj4)
do i = 1 ,np
  x1 = position(i)%x; y1 = position(i)%y; z1 = position(i)%z;
  itype = position(i)%type
  ioffset = (itype-1)*numAtomType
  neigh_off = neigh_alloc*(i-1)
  num = numneigh(i)
  ffx = 0.0d0; ffy = 0.0d0; ffz = 0.0d0
  !dir$ vector aligned
  !dir$ simd reduction(+:potential,ffx,ffy,ffz)
  do j= 1,num
    neigh = nlist(neigh_off+j)
    dx = x1-position(neigh)%x;dy = y1-position(neigh)%y;dz = z1-position(neigh)%z
    jtype = position(neigh)%type
    !---minimum image distance calculation
    boxdx = dx*ibox; boxdy = dy*ibox; boxdz = dz*ibox
    boxdx = (boxdx+sign(1/(epsilon(boxdx)),boxdx)) -sign(1/epsilon(boxdx),dx)
    boxdy = (boxdy+sign(1/(epsilon(boxdy)),boxdy)) -sign(1/epsilon(boxdy),dy)
    boxdz = (boxdz+sign(1/(epsilon(boxdz)),boxdz)) -sign(1/epsilon(boxdz),dz)
    dx = dx-box*boxdx; dy = dy-box*boxdy; dz = dz-box*boxdz
    dr2 = dx*dx + dy*dy + dz*dz
    !---lennard jones interactions
    dr2i = 1.0d0/dr2
    dr6i = dr2i*dr2i*dr2i
    if(dr2.gt.rcut2)dr6i = 0.0d0
    offset = ioffset + jtype
    forcelj = dr6i*(lj1(offset)*dr6i-lj2(offset))
    potential = potential + dr6i*(dr6i*lj3(offset)-lj4(offset))
    !---sum force to double precision accumulators
    force   = forcelj*dr2i
    ffx = ffx + dx*force
    ffy = ffy + dy*force
    ffz = ffz + dz*force
  enddo
  ff(i)%x = ffx; ff(i)%y = ffy; ff(i)%z = ffz
enddo
!$omp end parallel do
```

Figure 3. (Colour online) Short-range non-bonded force kernel (shortened for length).

calculation with all the atoms in the neighbour list of a given particle prevents placing the vdW and Coulombic force calculations within conditional if statements that impede vectorisation. The code below in Figure 3 demonstrates the application of the optimisations performed in this article for the non-bonded force calculation.

Traditional neighbour list builds employ a conditional if statement within the distance calculation. In this if statement, a particle j is added to the neighbour list of particle i on the condition that j and i are separated by a distance less than r_v. This conditional update of the neighbour list array within the distance calculation inner loop prevented vectorisation, and was found to severely degrade performance. As the distance calculation is the computationally demanding portion of the neighbour list build, it is essential that it vectorise to obtain good performance on the MIC. To achieve this, the distance calculation and neighbour list update are separated into one vectorised and one non-vectorised loop. In the distance calculation vectorised loop, the distance between atom i and all atoms j in the stencil bins are calculated and stored in a flat array. The flat array is subsequently post processed in another non-vectorised loop for the conditional update of the neighbour list. As in the non-bonded force routine, the !dir$ simd directive is used to ensure vectorisation of the distance calculation. An example of the vectorised distance calculation in the neighbour list build using the stencil algorithm is shown below in Figure 4.

Algorithms that (1) do not exhibit strong scaling (do not scale to hundreds of threads), or (2) are not sufficiently computationally intensive to merit offloading to the MIC, or (3) display poor vectorisation, are instead optimised to run on the host CPUs. As such, the long-range reciprocal space electrostatic solver, all intramolecular forces, and the Langevin thermostat calculations were performed on the host CPUs. The reciprocal space electrostatic solver is taken from the open source PME library in NAMD [2], and is parallelised using OpenMP on the host. To prevent having to identify 1–2, 1–3 or 1–4 bonded pairs on the MIC, which reduces the vectorisation throughput, electrostatic corrections are computed on the host by looping over the 1–2, 1–3 and 1–4 lists, and subtracting the appropriate Coulumbic force and energy. Seven OpenMP threads on the CPU per MPI rank were used, as this allows us to run 2 MPI ranks simultaneously on one node with no resource contention between ranks. The remaining idle core on the socket is devoted to tackling the communications with the assigned MIC and allows us to achieve good efficiency during offloads.

```
!===stencil  loop to calculate distances for an atom i.
!===dr2array allocated with length  np*cache_size,
!===cache_size is the maximum number of atoms in any bin
 ncache = 0; pad = (i-1)*cache_size
 do k = start_stencil,end_stencil
   !---look up neighboring cell index
   cell_neigh = cnum(k)
   !---look up pointers to atoms in cell neigh.
   !---atom ids are contiguous due to spatial sort
   c2s = start(cell_neigh); c2e = endposit(cell_neigh)
   !---array offset
   flag1 = pad + ncache + 1 - c2s
   !---vectorized distance calculation
   !dir$ simd
   do z= c2s,c2e
     dx = x1-position(z)%x; dy = y1-position(z)%y; dz = z1-position(z)%z
     boxdx = dx*ibox; boxdy = dy*ibox; boxdz = dz*ibox
     boxdx = (boxdx+sign(1/(epsilon(boxdx)),boxdx)) -sign(1/epsilon(boxdx),dx)
     boxdy = (boxdy+sign(1/(epsilon(boxdy)),boxdy)) -sign(1/epsilon(boxdy),dy)
     boxdz = (boxdz+sign(1/(epsilon(boxdz)),boxdz)) -sign(1/epsilon(boxdz),dz)
     dx = dx-boxdx*box; dy = dy-boxdy*box; dz = dz-boxdz*box
     dr2 = dx*dx + dy*dy + dz*dz

     !---store distances
     dr2array(z+flag1) = dr2
   enddo
   ncache = ncache + (c2e-c2s)+1
 enddo
```

Figure 4. (Colour online) Neighbour list build distance calculation.

Task level OpenMP parallelism is employed for all intramolecular force and Langevin thermostat calculations. Task-based parallelism is a shared memory programming model where independent calculations, i.e. tasks, are performed concurrently. In this scenario, the bonded, angle, dihedral, improper, electrostatic corrections and Langevin thermostat are performed concurrently on one of the eight CPU cores present on each socket. The shake algorithm was performed serially. Asynchronous task parallelism was employed between the MIC and host CPU calculations for further speed up. In asynchronous mode, the host sends work to the coprocessor and continues to execute, and stops only when results are needed from the coprocessor to continue. This allows the short-range forces to be calculated simultaneously with the PME, bonded, electrostatic-correction and Langevin thermostat forces.

The code example below in figure 5 demonstrates the application of the OpenMP task parallelism strategy for the case of the 1–2, 1–3 and 1–4 bonds. The bonded and Langevin thermostat routines are calculated on one of the eight CPU cores present on the host CPU socket, but in parallel using OpenMP task parallelism. As these calculations are being performed concurrently, it is necessary that each routine update its own force array to prevent race conditions. The asynchronous force calculation is concluded when both the MIC and host CPUs have performed all calculations. All force arrays are then summed to calculate the total force on each atom at the end of the asynchronous region.

In this study, all simulations were performed on the supercomputer Conte at Purdue University. Conte consists of compute nodes with two 8-core Intel Xeon-E5 processors (16 cores per node) and 64 GB of memory. Each node is equipped with two 60-core Intel® Xeon Phi™ coprocessors. Two hundred and

```
!$omp parallel
!$omp single
!$omp task
if(numbond.gt.0)then
  call bond_harmonic(step)
  call correct_12_bonds
endif
!$omp end task
!$omp task
if(numangle.gt.0)then
  call angle_harmonic
  call correct_13_bonds
endif
!$omp end task
!$omp task
if(numdihed.gt.0)then
  call dihed_PME_amber
  call correct_14_bonds
endif
!$omp end task
!$omp end single
!$omp end parallel
```

Figure 5. (Colour online) OpenMP task parallelism for bonded force calculations (shortened for length).

thrirty-six MIC threads with compact thread affinity were used in all MIC simulations in this article. The MD engine used in this article was built using the intel/13.1.1.163 and impi/4.1.1.064 compilers with –O3 –mkl –align array64byte –fp-model fast = 2 –fimf-domain-exclusion = 15 –march-native –assume buffered_io optimisation flags. The full source code employed can be obtained on github [38].

4. Results

The simulations below consisted of a glycine water system containing the following number of molecules: (1) 1500 glycine molecules in 12,399 water molecules (45,255 total atoms), 2500 glycine molecules in 16,508 water molecules (74,524 total atoms) and 3500 glycine molecules in 23,241 water molecules (104,723 total atoms). Glycine is simulated using the GAFF force field, with CNDO partial atomic charges, while the SPC/E water model is used for the water molecules [39]. The equations of motion were solved using a velocity Verlet algorithm. The temperature is controlled using a Langevin thermostat with a relaxation time of 2000 fs and a temperature set point of 300 K. The timestep was 2.0 fs. The shake algorithm was employed to constrain the motion of all hydrogens. A shake tolerance of 1.0e-4 was employed for all simulations and a maximum number of 25 shake iterations were allowed. The Berendsen barostat was used with a 1000 fs relaxation time. Data was recorded every 1000 timesteps. A PME (MIC code) and PPPM (LAMMPS) tolerance of 1.0e-6 was used. A bin distance of 1.5 $\dot{A}$ was used for neighbour list builds. This resulted in the neighbour list being built approximately every 20 timesteps. The acceleration due to asynchronous offload and task parallelism is discussed first. Subsequently, the CPU times of integration using eight CPU cores and one MIC will be compared against the spatial domain decomposition algorithm in LAMMPS with varying number of cores. The ideal weak scaling of the proposed parallelisation scheme is then demonstrated. Finally, suggestions on how to port the code to the KNL architecture are discussed.

4.1. Asynchronous and task parallel results

Figure 6 shows the per cent breakdown of the individual subroutines during the 2000 integration steps for all three simulation sizes studied. Analysis of the results show that while the non-bonded force calculation on the MIC is the most computationally intensive portion of the total force calculation, the host force calculation in total constitutes 48.4% of the total run time. In this article, the total host force calculation time is defined to be the total amount of time the PME reciprocal space, intramolecular, electrostatic correction and Langevin thermostat calculations contribute to the total simulation time. This stands in contrast to the total force calculation time, which is the total amount of time the force calculations on the MIC and host CPUs contribute to the total simulation run time. The intramolecular, electrostatic correction and Langevin force calculations are task parallel calculations, meaning a single CPU core can be devoted to each calculation, allowing the calculations to be performed concurrently using the OpenMP task parallelism strategy discussed earlier. Figure 7 shows graphically the total time of the host force calculation under the case of no asynchronous or task parallelism, no asynchronous but with host task parallelism, and with asynchronous and host task parallelism. Implementing simple OpenMP task parallelism for the host force calculations results in a 1.47 X speed up in the host calculation for the 1500 molecule case, a 1.52 X speed up for the 2500 molecule case and a 1.53 X speed up for the 3500 molecule case, showing that the speed up resulting from the implementation of the task parallelism is system size independent. Asynchronous task parallelism can also be implemented with the MIC short-range force calculation and the host CPU force calculation. In this scenario, the host CPUs send data to the MIC and continues to perform the host CPU force calculations. The host CPUs continue until is it necessary to receive data from the MIC to sum all the forces to calculate the total force on each atom. As shown, the implementation of asynchronous and host task parallelism reduces the host CPU force calculation to an insignificant amount of the total time of the total force calculation, and hence to the total simulation time. In total, the host CPU force calculation amounts to 5 s in the case of the 1500 molecules, 5 s in the case of the 2500 molecules, and 10 s in the case of the 3500 molecules. Thus, the host force calculation time constitutes less than 7% of the total run time for all three system sizes. In terms of speed up, the implementation of asynchronous and host task parallelism

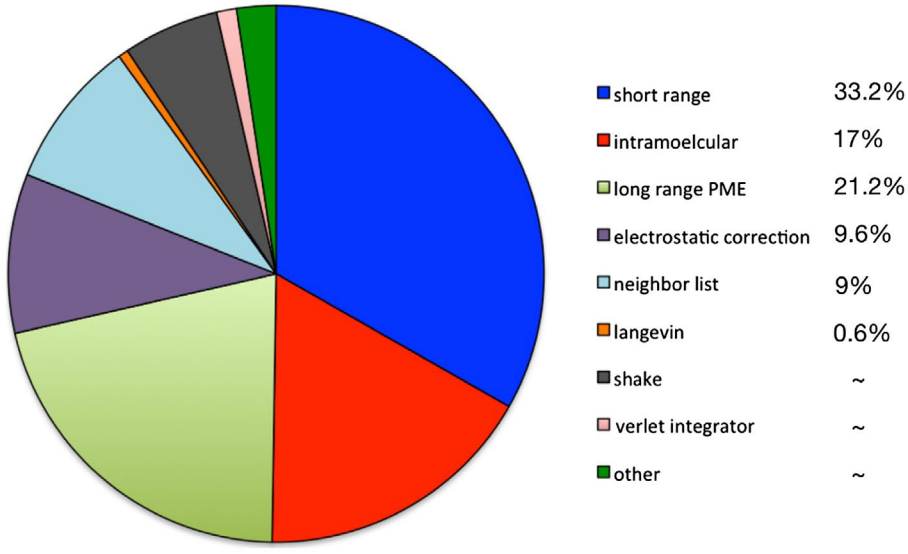

Figure 6. (Colour online) Pie chart.
Note: Percentage of total simulation time each subroutine constitutes.

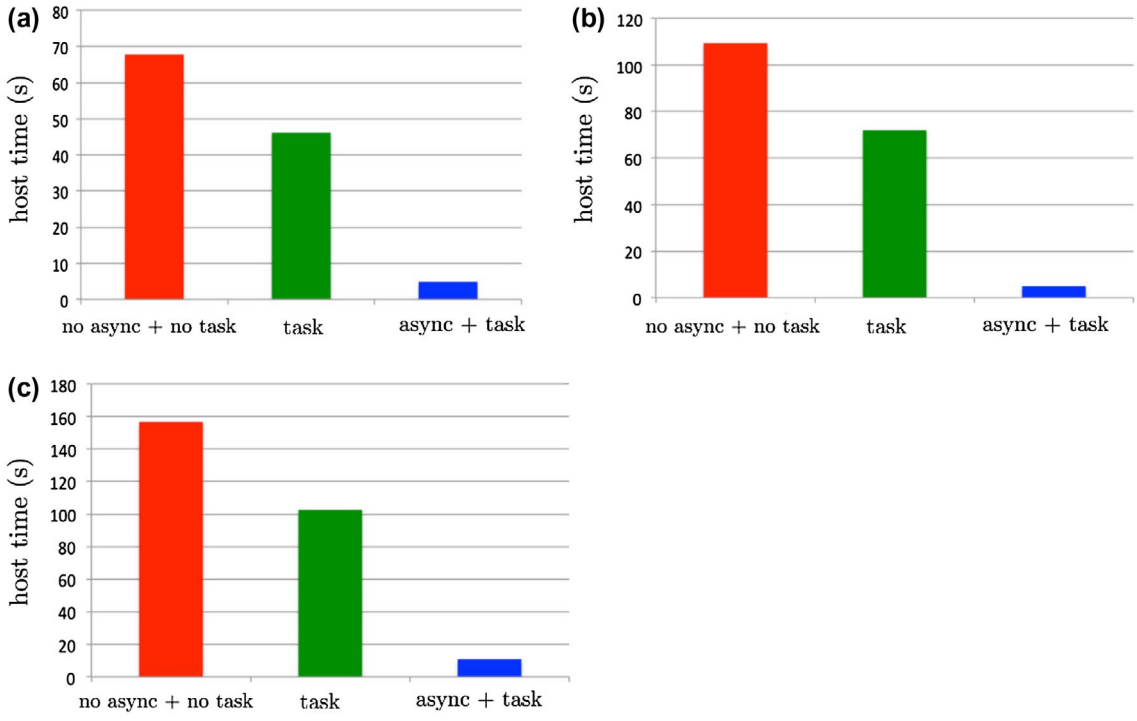

Figure 7. (Colour online) Host force calculation times.
Note: We plot the total time (s) calculating the forces on the host over the 2000 simulation steps performed. We consider three scenarios: (1) no async + no task, where neither asynchronous or task parallelisation is performed (2) task, where task parallelism is performed but no asynchronous parallelisation (3) async + task where asynchronous and task parallelisation are applied. Plot (a) corresponds to 1500 glycine molecule case. Plot (b) corresponds to the 2500 molecule glycine case. Plot (c) corresponds to the 3500 glycine molecule case.

resulted in a 2.2 X speed up in the total force calculation time in the case of the 1500 molecules, 2.3 X in the case of the 2500 molecules and 2.3 X in the case of the 3500 molecules. This shows that the power of the MIC lies not only in the wide vector registers and many cores, but also in the availability to split computational sections of code asynchronously between the host CPU cores and the MIC.

4.2. Simulation timing results

As mentioned in the introduction, employing a strictly spatial domain decomposition algorithm for multiple replica simulations with no accelerators can lead to unaffordable CPU time requirements, as each replica needs to be parallelised over a sufficient number of cores, leading to an excessively large computational requirement. To compare the performance trade-off between the spatial domain decomposition on host CPU cores against the atom domain decomposition on the MIC, we performed simulations for varying number of MPI ranks in LAMMPS to compare against the time obtained when using a single MIC and CPU socket. To reiterate, the underlying base code used for the MIC simulations is taken entirely from LAMMPS, with the exception of the PME solver, which is taken from NAMD. The results are shown in Figure 8. By first plotting the time of integration against number of MPI ranks employed during the LAMMPS simulations, the optimal number of MPI ranks for spatial domain decomposition can be determined. For all three system sizes, the optimal number is approximately 48 ranks, where the simulation length required roughly plateaus as a function of the number of MPI ranks. For all three cases, the performance provided by the eight CPU cores and one Intel® Xeon Phi™ coprocessor is equal to an approximately 40 rank spatial domain decomposition algorithm parallelisation. For these system sizes, the MIC provides a factor of five reduction in the cost of the simulation per simulation replica for all system sizes simulated here. Furthermore, the MIC parallelisation strategy provides performance only slightly less to what would be obtained from the optimal number of MPI ranks employed using a spatial domain decomposition, showing the proposed parallelisation strategy to be the preferred method to a strict spatial domain decomposition when many embarrassingly parallel trajectories must be simulated.

The algorithms and optimisations in this article were recently employed by the first author in a glycine nucleation simulation performed on Stampede supercomputer [11]. Stampede contains Sandy Bridge nodes containing with Xeon E5-2680 8-core Sandy Bridge processors and two first-generation Intel Xeon Phi SE10P MIC coprocessors, one per socket, on a PCIe card connected to its Sandy Bridge host. This makes Stampede ideal for the proposed parallelisation scheme. In the glycine nucleation simulations performed by the first author, 200 independent MPI replicas were simulated. As the proposed parallelisation scheme results in a factor of five cost reduction per MPI replica, the total cost reduction of the entire nucleation simulation was a factor of 1000 when compared against a strict spatial domain decomposition algorithm.

4.3. Weak scaling

To demonstrate the weak scaling of the code, we plot the per cent efficiency, for the same conditions mentioned above. As a

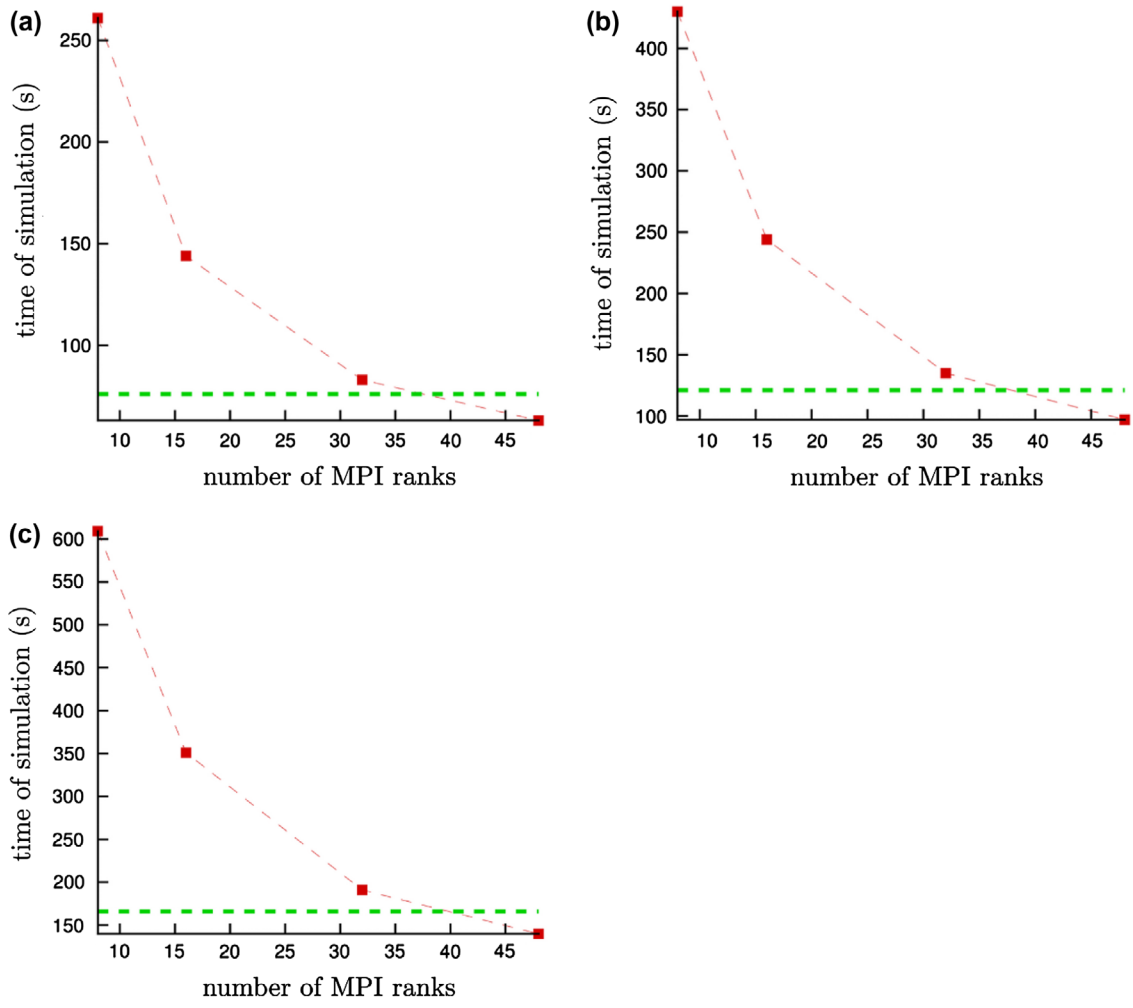

Figure 8. (Colour online) Spatial domain decomposition algorithm comparison.
Note: We plot the total time of integration (s) against the number of MPI ranks employed in the spatial domain decomposition algorithm. For comparison, the total time of integration using one MIC and eight CPU cores are plotted as the dashed green horizontal line. Plot (a) corresponds to 1500 glycine molecule case. Plot (b) corresponds to the 2500 molecule glycine case. Plot (c) corresponds to the 3500 glycine molecule case. The dashed red line is only meant to guide the eyes of the reader.

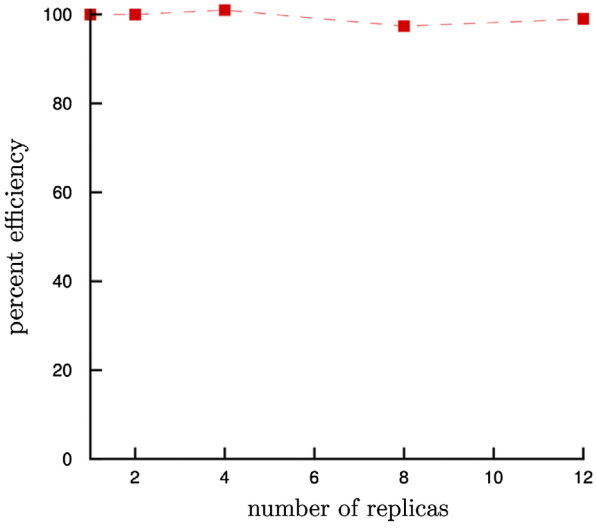

Figure 9. (Colour online) Ideal weak scaling.
Note: The percent efficiency is plotted against the number of simulation replicas employed. As can be seen from the plot, the proposed parallelisation scheme achieves ideal weak scaling, where the total simulation time is independent of the total number of replicas. The dashed red line is only meant to guide the eyes of the reader.

function of the number of embarrassingly parallel simulation replicas employed. The per cent efficiency is defined as follows in Equation (3).

$$\text{Percent efficiency} = \left(\frac{t_1}{t_n}\right) \times 100 \tag{3}$$

Here, t_1 is defined as the total time of integration when running one replica. Similarly, t_n is defined as the time of integration when running 2N MPI replicas on N nodes (one replica per socket). The results are plotted below in Figure 9. We see that the per cent efficiency is independent of the number of MPI replicas employed, and is equal to 100%. This demonstrates that the proposed parallelisation scheme achieves ideal weak scaling, allowing the scientist to perform multiple replica simulations with no performance penalty. To achieve this, it was found to be critical that seven host OpenMP threads are used for parallelisation on the host, and the remaining idle core be devoted to handling communications between the MIC and host.

4.4. Knight's landing architecture

Intel® Xeon Phi™ Knights Landing (KNL), the successor to the MIC accelerator architecture, has been deployed recently in several supercomputers. KNL is generally installed as a standalone processor instead of as an accelerator, and resembles a traditional processor except for the many cores, e.g. 68 cores per chip, four hardware threads per core, each with a 512-bit wide vector register. As algorithms that display strong scaling and SIMD level parallelism will exhibit maximum performance on both the current generators of MICs and the KNL, the proposed parallelisation scheme in this article is applicable in many regards with some modifications. Asynchronous offload is not applicable any more on the KNL architecture, as the KNL is a standalone processor. To port current the algorithm to KNL, the work previously designed for offload to the MIC, the neighbour list build and non-bonded interactions, would be integrated into an overall OpenMP task parallelism scheme. The non-bonded and neighbour list build calculations would be assigned one unique OpenMP task. The number of threads assigned to these OpenMP tasks would be optimised for performance of a given simulation system. Concurrently, the PME, bonded, and Lanegvin thermostat calculations can be performed. For the results presented in this article, the bonded and Langevin thermostat, calculations were performed on a single core concurrently. However, due to the large core and thread count of the KNL architecture, the bonded and Langevin thermostat calculations could be parallelised using OpenMP within an OpenMP task. This would exploit the task parallelism of the short-range non-bonded, long range electrostatic, Langevin thermostat, and bonded force calculation, and the strong scaling present in the neighbour list builds and short-range non-bonded force calculation.

5. Conclusions

An increasing number of problems in condensed matter physics require extensive sampling of low probability phase space coordinates, requiring multiple replica simulations, such as umbrella sampling, parallel tempering, forward flux sampling, or seeded cluster simulations to obtain converged variable estimates. These simulations frequently require large-scale MD to avoid spurious finite size effects, and many trajectories to obtain converged estimates of observables, putting the computational scientist in the troubling position of not only requiring sufficient speed up from parallelisation of one trajectory, but now for many trajectories. If a straightforward spatial domain decomposition algorithm were employed for each trajectory, where each trajectory is parallelised solely over CPU cores, this can quickly lead to unmanageable computational requirements, as each MD replica may require multiple nodes for sufficient speed up. Now with the advent of modern many core architectures, such as Intel® Xeon Phi™ coprocessor, researchers have access to teraflop performance on a single accelerator present on a single CPU socket. The Intel® Xeon Phi™ coprocessor allows the researcher to not only speed up hot spots in codes using the simple OpenMP parallelisation language, with no rewriting of base code required, but also allows the scientist the opportunity to perform asynchronous task parallelism, where code is simultaneously executed on the host and MIC, until data is needed from the computations being performed on the MIC to proceed further. In the realm of MD, this means that while the computationally intensive short-range forces (vdW and Coulombic) are being calculated, the host CPUs can be performing the reciprocal space PME, bonded, and Langevin thermostat force calculations simultaneously. For many systems sizes and densities, this amounts to sweeping the host CPU calculation time under the rug.

In this article, a parallelisation strategy for multiple replica molecular dynamics was investigated. The parallelisation strategy was demonstrated using a glycine in water system. Contrary to an MPI spatial domain decomposition algorithm over CPU cores, individual MPI ranks were mapped to host sockets and one MIC. Seven OpenMP threads on the host CPUs were used to perform task parallelism calculations for the intramoelcular and Langevin thermostat force calculations. The asynchronous calculation of short-range forces and reciprocal space PME forces, intramolecular forces, and Langevin thermostat forces lead to a 2.2 X speed up in the total simulation time for the case of the 1500 glycine molecule, and a 2.3 X speed up in the case of the 2500 and 3500 glycine molecules, when compared against no asynchronous parallelisation. With this parallelisation strategy, the total time spent calculating forces on the host CPUs, which was 50% of the run time in the absence of the asynchronous parallelisation, constitutes less than 7% of the total run time. When comparing the total time of simulation between the proposed parallelisation strategy and a strict spatial domain decomposition algorithm with no accelerator, it was found that the proposed strategy provides the performance on a single socket equivalent to 40 MPI ranks using a spatial domain decomposition. Furthermore, the optimal number of MPI ranks using a spatial domain decomposition was found to be approximately 48 ranks, showing that the MIC provides close to the optimal performance on a single host socket for the systems studied. In total, the cost per simulation replica is reduced by a factor of five for the systems studied here. Finally, the scaling of the parallelisation scheme was investigated. By using seven OpenMP threads, we left one core idle to handle communication with the MIC. This allowed us to achieve ideal weak scaling, where the simulation time was independent of the total number of replicas simulated.

Acknowledgements

This work used the XSEDE Extended Collaborative Support Service (ECSS) programme.

Funding

This research was funded by AbbVie [grant number 8000053025] and [grant number 8000069224]. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation [grant number ACI-1053575].

References

[1] Plimpton S. Fast parallel algorithms for short range molecular dynamics. J Comput Phys. 1995;117:1–19.
[2] Phillips JC, Braun R, Wang W, et al. Scalable molecular dynamics with NAMD. J Comput Chem. 2005;26:1781–1802.
[3] Patel S, Vierling E, Tama F. Replica exchange molecular dynamics simulations provide insight into substrate recognition by small heat shock proteins. Biophys J. 2014;106:2644–2655.

[4] Bergonzo C, Henriksen NM, Roe DR, et al. Multidimensional replica exchange molecular dynamics yields a converged ensemble of an RNA tetranucleotide. J Chem Theory Comput. 2014;10:492–499.
[5] Wang K, Chodera JD, Yang Y, et al. Identifying ligand binding sites and poses using GPU-accelerated Hamiltonian replica exchange molecular dynamics. J Comput Aided Mol Des. 2013;27:989–1007.
[6] Moroni D, Van Erp TS, Bolhuis PG. Investigating rare events by transition interface sampling. Phys A Stat Mech Appl. 2004;340:395–401.
[7] Moroni D, Van Erp TS, Bolhuis PG. Simultaneous computation of free energies and kinetics of rare events. Phys Rev E Stat Nonlin Soft Matter Phys. 2005;71:056709.
[8] Sanz E, Vega C, Espinosa JR, et al. Homogeneous ice nucleation at moderate supercooling from molecular simulation. J Am Chem Soc. 2013;135:15008–15017.
[9] Lauricella M, Meloni S, English NJ, et al. Methane clathrate hydrate nucleation mechanism by advanced molecular simulations. J Phys Chem C. 2014;118:22847–22857.
[10] Zimmermann NER, Vorselaars B, Quigley D, et al. Nucleation of NaCl from aqueous solution: critical sizes, ion-attachment kinetics, and rates. J Am Chem Soc. 2015;137:13352–13361.
[11] Parks C, Koswara A, DeVilbiss F, et al. Solubility curves and nucleation rates from molecular dynamics for polymorph prediction – moving beyond lattice energy minimization. Phys Chem Chem Phys. 2017; 19: 5285–5295.
[12] Kästner J. Umbrella sampling. Wiley Interdiscip Rev Comput Mol Sci. 2011;1:932–942.
[13] Reinhardt A, Doye JPK, Noya EG, et al. Local order parameters for use in driving homogeneous ice nucleation with all-atom models of water. J Chem Phys. 2012;137:194504.
[14] Filion L, Hermes M, Ni R, et al. Crystal nucleation of hard spheres using molecular dynamics, umbrella sampling, and forward flux sampling: a comparison of simulation techniques. J Chem Phys. 2010;133:244115.
[15] Auer S, Frenkel D. Quantitative prediction of crystal-nucleation rates for spherical colloids: a computational approach. Annu Rev Phys Chem. 2004;55:333–361.
[16] Berendsen HJC, van der Spoel D, van Drunen R. GROMACS: a message-passing parallel molecular dynamics implementation. Comput Phys Commun. 1995;91:43–56.
[17] Pearlman DA, Case DA, Caldwell JW, et al. 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.
[18] Pennycook SJ, Hughes CJ, Smelyanskiy M, et al. Exploring SIMD for molecular dynamics, using Intel® Xeon® processors and Intel® Xeon Phi Coprocessors. 2013 IEEE 27th International Symposium Parallel Distributed Process; Washington, DC. 2013. p. 1085–1097.
[19] Needham PJ, Bhuiyan A, Walker RC. Extension of the AMBER molecular dynamics software to Intel's Many Integrated Core (MIC) architecture. Comput Phys Commun. 2015;201:95–105.
[20] Parks C, Koswara A, Tung H-H, et al. Extending the crystal landscape through electric field controlled crystallization – a molecular dynamics case study. Phys Chem Chem Phys. 2017; (submitted).
[21] Parks C, Koswara A, Tung H, et al. Nanocrystal dissolution kinetics and solubility increase prediction from molecular dynamics – the case of α-, β-, and γ-glycine. J Mol Pharmaceutics. 2017; DOI: 10.1021/acs.molpharmaceut.6b00882.
[22] Lovette MA, Browning AR, Griffin DW, et al. Crystal shape engineering. Ind Eng Chem Res. 2008;47:9812–9833.
[23] Gao Y, Olsen KW. Molecular dynamics of drug crystal dissolution: simulation of acetaminophen form I in water. Mol Pharm. 2013;10:905–917.
[24] Gao Y, Olsen KW. Unique mechanism of facile polymorphic conversion of acetaminophen in aqueous medium. Mol Pharm. 2014;11:3056–3067.
[25] Shah M, Santiso EE, Trout BL. Computer simulations of homogeneous nucleation of benzene from the melt. J Phys Chem B. 2011;115:10400–10412.
[26] Chen J, Trout BL. A computational study of the mechanism of the selective crystallization of α- and β-glycine from water and methanol–water mixture. J Phys Chem B. 2010;114:13764–13772.
[27] Salvalaglio M, Mazzotti M, Parrinello M. Urea homogeneous nucleation mechanism is solvent dependent. Faraday Discuss. 2015;179:291–307.
[28] Salvalaglio M, Vetter T, Giberti F, et al. Uncovering molecular details of urea crystal growth in the presence of additives. J Am Chem Soc. 2012;134:17221–17233.
[29] Knott BC, Molinero V, Doherty MF, et al. Homogeneous nucleation of methane hydrates: unrealistic under realistic conditions. J Am Chem Soc. 2012;134:19544–19547.
[30] Locker CR, Rutledge GC, Link C. Molecular dynamics simulation of homogeneous crystal nucleation in polyethylene. Macromolecules. 2013;11:4723–4733.
[31] 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.
[32] Eastwood JW, Hockney RW, Lawrence DN. The three-dimensional periodic particle-particle/particle-mesh program. Comput Phys Commun. 1980;19:215–261.
[33] Verlet L. Computer, "Experiments" on classical fluids. Thermodynamical properties of Lennard–Jones molecules. Phys Rev. 1967;159:98–103.
[34] Frenkel D, Smit B. Understanding molecular simulation: from algorithms to applications. San Diego (CA): Academic; 1996.
[35] Pennycook S, Hughes CJ, Smelyanskiy M. Optimizing gather/scatter patterns. High perform. Parallelism pearls. 2014.
[36] Le Grand S, Götz AW, Walker RC. SPFP: speed without compromise – a mixed precision model for GPU accelerated molecular dynamics simulations. Comput Phys Commun. 2013;184:374–380.
[37] Meloni S, Rosati M, Colombo L. Efficient particle labeling in atomistic simulations. J Chem Phys. 2007;126:121102.
[38] Parks C, Wang Y, Huang L, et al. Accelerating multiple replica molecular dynamics simulations using the Intel® Xeon PhiTM coprocessor Github repository [Internet]. 2017. Available from: https://github.com/coparks2012/MOLECULAR_SIMULATION_CODE.
[39] Cheong DW, Boon YD. Comparative study of force fields for molecular dynamics simulations of α-glycine crystal growth from solution. Cryst Growth Des. 2010;10:5146–5158.