Computer Physics Communications 280 (2022) 108472

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

A scalable algorithm for many-body dissipative particle dynamics using multiple general purpose graphic processing units ☆

Davide Di Giusto [a], Jony Castagna [b,*]

[a] Polytechnic Department, University of Udine, Udine, Italy
[b] UK Research and Innovation - Science and Technology Facilities Council Hartree Centre, Daresbury Laboratory, Warrington, Cheshire WA4 4AD, UK

ARTICLE INFO

Article history:
Received 15 January 2022
Received in revised form 8 June 2022
Accepted 19 July 2022
Available online 25 July 2022

Keywords:
Many-body DPD
Mesoscale simulation
Multi-GPU
High performance computing

ABSTRACT

We present a novel algorithm for the many-body Dissipative Particle Dynamics (DPD) forces calculation which allows to efficiently scale the DL_MESO software package on Multiple General Purpose Graphic Processing Units. Together with the extension to 64-bit integer arrays and addition of hard surface boundary conditions, the proposed algorithm allows to simulate very large complex mesoscale systems up to 14 billion beads. The implementation takes advantages of the CUDA language stream features to overlap the exchange of particle positions and local densities and the computation of the short range forces. We tested a water drop between two plates system using tree of the main European supercomputers: Piz Daint, Marconi and JUWELS. Results shows an improvement on the speedup compared to a naive implementation up to 1.5*x* when using 1024 GPUs.

1. Introduction

In the range of mesoscale simulations, i.e. between 10^{-9} and 10^{-6} meters, the Dissipative Particle Dynamics (DPD) [1,2] is one of most popular approaches due to its capability in accounting for the molecular properties of the chemical species, their global effects as continuum fluids, its algorithmic simplicity and its enormous versatility [3]. Groups of atoms and/or molecules are usually represented via coarser beads allowing a larger time step compared to Molecular Dynamics (MD) simulations. A large number of beads recovers the Navier-Stokes equations behaviour, but with the advantage of keeping the fluctuation-dissipation effects usually neglected in classical Computational Fluid Dynamic (CFD) solvers.

However, one of the main limitations of standard DPD [2], i.e. where there is a linear potential depending only on the inter-particle separation distance, is the quadratic equation of state for the system. This limit does not allow to study complex systems where two phases have strong density variations with pressure and chemical species. The many-body DPD method [4,5] provides a solution making the conservative forces depending also from a local density. This needs to be calculated at every iteration and requires the particle positions. For parallelization via typical domain decomposition, the particle position needs to be exchanged between domains. The impact of this exchange, occurring via the network bus, is usually detrimental for scalability.

Particle based solvers, like DPD and more in general MD solvers, can easily benefit from General Purpose Graphic Processing Units (here after GPUs) accelerators. In particular, NVidia GPUs are currently widely spread in the top500 supercomputer HPC thanks to their higher peak performance and memory bandwidth when compared to traditional CPUs. They can be programmed in different ways, some more portable than others, but high tuning performance usually requires the use of the specific Compute Unified Device Architecture (CUDA) language developed by NVidia itself. Great speedup can be easily achieved after memory reorganization and ad hoc tuning as reported in AMBER [6], LAMMPS [7], GROMACS [8], NAMD [9], ACEMD [10] and HOOMD-blue [11].

To the authors knowledge, LAMMPS has the closest implementation of the many-body DPD algorithm on GPU to the one here presented. The USER-MESO extension package, written in CUDA C/C++ with MPI library and OpenMP directives for parallelization, has been used by Xia et al. [12,13] to simulate flow in nano-porous rocks, but there is no mention of the overlap between local density calculation and communication.

In a previous paper we presented a porting to CUDA of the DL_MESO DPD solver [14] for mesoscale simulations. Results showed a strong scaling efficiency $> 85\%$ when simulating systems

☆ The review of this paper was arranged by Prof. David W. Walker.

* Corresponding author.
E-mail address: jony.castagna@stfc.ac.uk (J. Castagna).

https://doi.org/10.1016/j.cpc.2022.108472

with low density variations, like phase separation between oil and water. In this paper we extend the implementation on CUDA to include: a) a many-body DPD algorithm for strong density variations which allows to keep the same scaling efficiency observed for quasi-similar density systems; b) wall boundary conditions via frozen surfaces and c) the use of integer 64-bit arrays to simulate systems with more than 2 billion beads. A description of the DPD mathematical formulation is given in section 2 while details on the many-body DPD implementation are in section 3. Finally, in section 4 we presents results on performance on a single GPU using different GPU cards, strong and weak scaling up to 1536 GPUs on a water drop between two surfaces and the evidence of overlap between communication and computation obtained using the NSight System profiler.

DL_MESO [15] is a software package for mesoscale simulations made of two separate components: a particle dynamic solver based on DPD and a fluid dynamic solver based on the Lattice Boltzmann Equation (LBE) methodology. It has been developed at UKRI-STFC Daresbury Laboratory, mainly by Dr Micheal Seaton. It has been mainly funded by the United Kingdom Collaborative Computational Project for the Computer Simulation of Condensed Phases CCP5, however the multi-GPU version here presented has been developed under the E-CAM project founded by the EU. DL_MESO is available free of charge to academic scientists pursuing research of a non-commercial nature and for a fee for industrial applications. The single and multi-GPU versions used in this paper are available on the git repository DL_MESO under the *single_GPU_version* and *multi_GPU_version* branches.

2. Mathematical model

Here we present the DPD mathematical formulation, the numerical scheme used in DL_MESO and the many-body DPD equations. More details can be found in the DL_MESO manual and in the work of Seaton et al. [15].

2.1. Equations of DPD

Forces between particles in DPD can be mainly split in three components: 1) a pairwise soft potential repulsive force $\mathbf{F}_{ij}^C = A_{ij}$; 2) a drag force $\mathbf{F}_{ij}^D$ which accounts of the viscosity and friction between particles and 3) a stochastic force $\mathbf{F}_{ij}^R$ which balances the drag force and keeps the system temperature constant. For each particle, the overall force is then given by:

$$\mathbf{F}_i = \sum_{i \neq j}^{N} (\mathbf{F}_{ij}^C + \mathbf{F}_{ij}^D + \mathbf{F}_{ij}^R) \tag{1}$$

The damping drag force balances the stochastic force in order to satisfy the fluctuation-dissipation theorem and maintain thermodynamic equilibrium. All forces are pairwise, i.e. $\mathbf{F}_{ij} = -\mathbf{F}_{ji}$ and are defined as:

$$\mathbf{F}_{ij}^C = A_{ij}\omega^C(r_{ij})\mathbf{e}_{ij} \tag{2}$$

$$\mathbf{F}_{ij}^D = -\gamma_{ij}\omega^D(r/ij)(\mathbf{e}_{ij}\cdot\mathbf{v}_{ij})\mathbf{e}_{ij} \tag{3}$$

$$\mathbf{F}_{ij}^R = \sigma_{ij}\omega^R(r_{ij})\epsilon_{ij}(\Delta t)^{-(1/2)}\mathbf{e}_{ij} \tag{4}$$

where $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$, $r_{ij} = |\mathbf{r}_{ij}|$, $\mathbf{e}_{ij} = \mathbf{r}_{ij}/r_{ij}$. ω^C, ω^D and ω^R are the switching functions for the soft potential, drag and random forces, respectively. A_{ij}, γ_{ij} and σ_{ij} are the conservative interaction, dissipative and random force coefficients between particles i and j, respectively. The values of ϵ_{ij} represent symmetric random Gaussian variables, possessing zero mean and unit variance, that are uncorrelated for different pair of particles and different times.

While in MD particles have a hard core due to the strong non linear potential, in DPD the soft potential allows particle to overlap and cross each other. The conservative potential usually is chosen as a linear function of the distance between two particles, i.e. $\omega^C = 1 - r_{ij}/r_c$ for $r_{ij} < r_c$. The satisfaction of the fluctuation-dissipation theorem provides a thermostat within the canonical ensemble and additionally conserves both the total momentum of the system and local momenta for particle pairs. As explained by Espanol and Warren [16], this condition occurs when $\sigma^2 = 2\gamma_{ij}K_BT$ and $\omega^D(r_{ij}) = [\omega^R(r_{ij}]^2$.

2.2. Numerical methods

To obtain a numerically stable solution, the Verlet Velocity (VV) scheme is usually used [17].

This consists of splitting the time integration in 2 stages. In the first stage, the particle velocities are advanced to time $t + \frac{\Delta t}{2}$, using the forces calculated at time t. The particle positions are then advance to $t + \Delta t$:

$$\mathbf{v}_i\left(t + \frac{\Delta t}{2}\right) = \mathbf{v}_i(t) + \frac{\Delta t}{2}\frac{\mathbf{F}_i(t)}{m_i} \tag{5}$$

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i\left(t + \frac{1}{2}\Delta t\right)\Delta t \tag{6}$$

In the second stage, the velocities are updated to time $t + \Delta t$:

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t + \frac{\Delta t}{2}) + \frac{\Delta t}{2}\frac{\mathbf{F}_i(t + \Delta t)}{m_i} \tag{7}$$

A naïve calculation of force interactions between all particle pairs would require a $\mathcal{O}(N^2)$ (where N is the number of particles) search algorithm, which will easily limit its practicability to large systems. The VV scheme is then usually applied in combination with a cell linked-list method [18], which reduces the computation to $\mathcal{O}(N)$. The cell linked-list method consists of finding each particle's neighbouring particles within a cut-off radius at least as large as the maximum extent for the soft conservative potential (r_c).

The DL_MESO DPD code is written in Fortran 2003 with MPI, while the single and multi-GPU versions are written in CUDA language. To guarantee full compatibility with the CPU version, the GPU versions uses the same Fortran 2003 subroutines for initialization and IO operations, including MPI_IO for parallel file writing. However, the main loop for time step integration and all the correlated computing functions have been ported to CUDA. Moreover, a reorganization of the memory layout allows to achieve a $\sim x4$ speedup when compared to a traditional 24-core CPU as presented in the section 4. More details on the DPD equations and the numerical methods used in the single and multi-GPU versions can be found in the work of Castagna et al. [14].

2.3. Equations for many-body DPD

To take in account of thermodynamic equations of state different from the quadratic relation $p = f(\rho^2)$, we calculate an instantaneous free-energy:

$$\tilde{\mathcal{G}^{ex}} = \sum_i \psi^{ex}(\tilde{\rho}_i) \tag{8}$$

and obtain the conservative force as the spatial derivative

$$\vec{F}_i^C = -\frac{\partial \tilde{\mathcal{G}}^{ex}}{\partial \vec{r}_i} = -\sum_j \frac{\partial \psi^{ex}(\tilde{\rho}_i)}{\partial \vec{r}_i} \tag{9}$$

where the local-density approximation $\tilde{\rho}_i$ can be expressed as:

$$\begin{aligned}\tilde{\rho}_i &= \int d\vec{r}\omega^\rho(|\vec{r}-\vec{r}_i|)\rho(\vec{r},\{\vec{r}_k\}) \\ &= \sum_{j\neq i}\int d\vec{r}\omega^\rho(|\vec{r}-\vec{r}_i|)\delta(\vec{r}-\vec{r}_j) \\ &= \sum_{j\neq i}\omega^\rho(r_{ij})\end{aligned} \tag{10}$$

and ω^ρ is the weight function usually defined as:

$$\omega^\rho = \frac{15}{2\pi r_d^3}\left(1-\frac{r_{ij}}{r_d}\right)^2 \quad (r_{ij} < r_d) \tag{11}$$

To obtain the pairwise conservative force:

$$\vec{F}_{ij}^C = \left(\frac{\partial \psi^{ex}(\tilde{\rho}_i)}{\partial \tilde{\rho}_i} + \frac{\partial \psi^{ex}(\tilde{\rho}_j)}{\partial \tilde{\rho}_j}\right)\omega^C(r_{ij})\frac{\vec{r}_{ij}}{r_{ij}} \tag{12}$$

A van der Waals-like equation of state can be obtained setting:

$$\psi^{ex}(\tilde{\rho}) = \frac{\pi}{30}A_{ij}\bar{\tilde{\rho}} + \frac{\pi r_d^4}{30}B_{ij}\tilde{\rho}^2 \tag{13}$$

where $\tilde{\rho}$ is the same as $\tilde{\rho}$ but with the cutoff set to r_c instead of r_d ($\bar{\tilde{\rho}}$). In this case, the force associated is:

$$\vec{F}_{ij}^C = \left[A_{ij}\left(1-\frac{r_{ij}}{r_{c,ij}}\right) + B_{ij}(\rho_i+\rho_j)\left(1-\frac{r_{ij}}{r_d}\right)\right]\frac{\vec{r}_{ij}}{r_{ij}} \tag{14}$$

and setting $A_{ij} < 0$ and $B_{ij} > 0$ we obtain the following vapour/liquid equation of state:

$$p = \rho k_B T + \alpha A\rho^2 + 2\alpha B r_d^4(\rho^3 - c\rho^2 + d) \tag{15}$$

where α, c and d can be specified in the input files of DL_MESO.

3. Scalable many-body DPD algorithm

In [14] is showed the importance of a correct overlap between computation and communication in order to achieve good scalability on the multi-GPU version of DL_MESO. To avoid duplicating the force calculation kernel, a distinction between *internal cells* and *boundary cells* is made (see Fig. 1). We will refer hereafter as *internal particles* those which lie in the internal cells, *boundary particles* those within the boundary cells and finally as *ghost particles* those in the ghost cells which are used to exchange data between different domains. A flag controls the switch in the kernel between the internal and boundary particles. However, as seen in the previous section, the many-body calculation requires an extra information, the local-density, to correctly obtain the potential forces. This depends on the particle positions, so a first swap of their coordinates is necessary. Then, the local densities need also to be swapped, adding an extra communication to the overall procedure and, most crucially, not allowing the overlap between communications and computation of internal cells forces as done in [14]. A naive implementation would follow the diagram in Fig. 2a.

To avoid this serialization communication → computation we propose to split the local-density as follows:

$$\rho_i = \rho_i^{in} + \rho_i^{ib} + \rho_i^{bg} \tag{16}$$

where ρ_i^{in} is the local-density component due to internal particles only, ρ_i^{ib} is the component due to the interaction between internal and boundary particles and finally ρ_i^{bg} is the component due to the interaction between boundary and ghost particles. We then divide the potential forces calculation as follows:

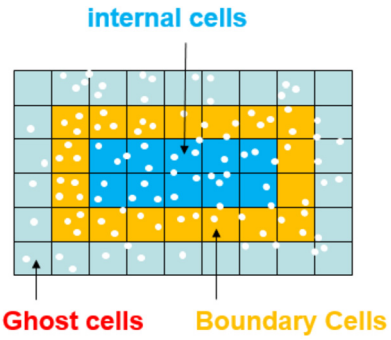


Fig. 1. Internal, boundary and ghost cells for a single GPU domain.

$$f_{ij} = f'_{ij} + f'' \tag{17}$$

where f'_{ij} is the force on the internal cell particles due to local densities ρ_i^{in} and ρ_i^{ib}, while f'' is the contribution due to ρ_i^{bg} which applies to both internal particles (those at the border with the boundary) and boundary particles. This split is possible due the force calculation being a linear operator function of the local-density.

We then proceed as in the Algorithm 1:

Algorithm 1: Split force calculation algorithm.

```
find local-density ρ_i^in + ρ_i^ib;
find f' (CUDA stream 1);
for each direction do
    swap particle positions CUDA (stream 2);
    find ρ_i^bg (CUDA stream 2);
    swap local densities (CUDA stream 2);
end
find f'';
```

The above split allows to overlap the computation of f' with the swap of particle positions, calculation and swap of local densities ρ_i^{bg} using two different CUDA streams. We will refer hereafter to stream 1 as *streamCompute* and the stream 2 as *streamExchange*. Note as the calculation of the forces and densities for the internal particles is executed forward looking, i.e. taking advantage of 3rd Netwon's law ($x_{ij} = -x_{ji}$ and $F_{ij} = -F_{ji}$), while for the boundary particles we do and explicit calculation looking in all directions.

3.1. Other improvements

In this section we will look into the details of the 3 main improvements implemented in DL_MESO multi-GPU version compared to the version presented in [14]: 1) the extension to 64-bit integer arrays; 2) the implementation of wall boundary conditions.

3.1.1. Extension to LONG INT and hard surfaces boundary conditions

The master version of DL_MESO does not support systems larger than 2 billion particles (2^{31}-1 in theory). This is due to the use of INTEGER 32 bit Fortran arrays across the code. The main reason for this choice was linked to very long integration times required to achieve thermodynamic equilibrium, even with a small number of particles. However, the use of multi-GPU architectures allows to drastically accelerate the solver and then to simulate very complex phenomena like macropolymer chains. The use of LONG INTEGER arrays (64 bit, i.e. 2^{63}-1 max particles) allows to move beyond the current limit, leaving the GPU memory as the only constrain to the size of the system. However, this change makes

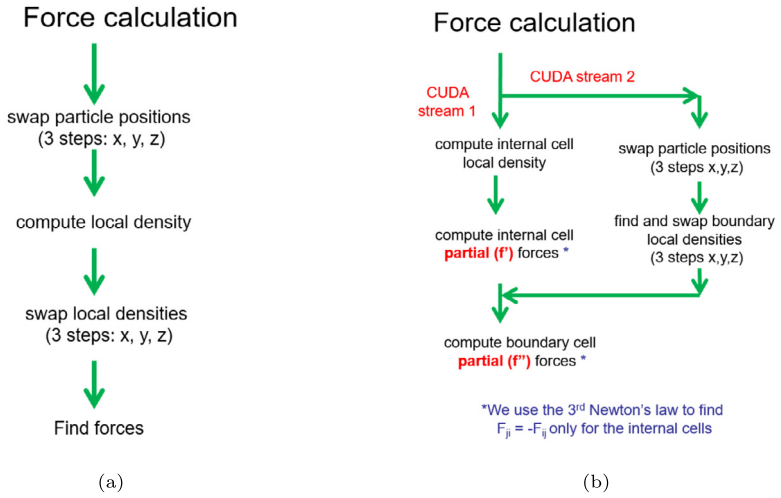


Fig. 2. a) naive implementation of many-body DPD algorithm which does not allow to overlap computation with communication. b) scalable parallel many-body DPD.

some of the OUTPUT files incompatible with the master version and post processing subroutines have been adapted to the 64-bit integer format.

3.1.2. Wall boundary conditions

A wall boundary condition consists of a no-slip reflecting boundary where particles are bouncing back with specular velocity components from a frozen layer of beads. We obtain this in DL_MESO by combining the *frozen* and *surface* flags which create a layer of frozen particles along a defined plane and introduce a repulsive potential at the surface. In particular, the short range wall repulsion is given by:

$$U_{wall}(z) = \frac{1}{2} A_{wall,\alpha} z_c (1 - z/z_c)^2, \quad (z < z_c) \tag{18}$$

where $A_{wall,\alpha}$ is the repulsive force magnitude with species α, z is the distance between the particle and the wall and z_c the surface repulsion range. However, an appropriate choice for the density of frozen beads in the walls and interactions between frozen and non-frozen particles has to be made in order to reduce the density fluctuations near the walls [19].

As the number of ghost cells, required for the short range repulsive force, is equal to the number of ghost cells for periodic boundary conditions, the parallel implementation is straightforward, taking into account that no communication is required through the frozen layers as the periodicity of the domain is broken.

4. Results

We tested our single and multi-GPU versions of DL_MESO on the latest three NVidia GPU generations: P100, V100 and A100 (A100 being the latest). These are the same cards installed on the three main European supercomputers used for the weak and strong scaling benchmarks: the Swiss Supercomputer Piz Daint (P100) from CSCS, the Italian Marconi (V100) from CINECA and the German supercomputer JUWELS (A100) from Jülich Supercomputing Centre at Forschungszentrum Jülich.

We first present the performance results on the single GPU version and 64-bit integer arrays extension for a binary mixture system modelled via standard DPD. The purpose is to provide a reference for comparison for on a water-vapour test case where the many-body DPD formulation is required. This will allow us to asses the impact on speedup and scalability of the algorithm here presented.

4.1. Single GPU performance

As the code is memory bandwidth bounded, to use a single CPU core as comparison would not be correct, being the CPU bandwidth not fully saturated. The choice of the CPU baseline is then based on the bandwidth of the available CPUs. The theoretical maximum speedup S_{max} would be the ratio of theoretical memory bandwidths (assuming both code can fully saturate it). At present, this ratio is around a factor $10 \div 20$, irrespective from the number of CPU or GPU cores. A good indicator of performance would then be the efficiency η as ratio between the speedup GPU vs CPU (S) and the theoretical maximum speedup S_{max}:

$$\eta_{mb} = S/S_{max} \tag{19}$$

We tested our single GPU version on the latest NVidia Ampere A100 card and compared with the previous V100 performance (900GB/s and 1555GB/s memory bandwidths, respectively) using 2 different CPU: a 6-core Intel(R) Xeon(R) W-2133 CPU @ 3.60 GHz and a 24-core AMD (Rome) 7402 CPU @ 2.7 GHz (85.3 GB/s and 204.8GB/s memory bandwidths, respectively). Figs. 3a and 3b shows the different comparisons. The A100 shows a factor $\sim x2$ speedup vs the V100. The A100 is $\sim 4x$ the 24-core AMD CPU and $\sim 17x$ compared to the 6-core Intel CPU. In terms of efficiency η_{mb}, there is a drops from ~ 1 to ~ 0.4 when using a CPU with a larger number of cores. This indicates that there is potential for improvement on the GPU card, like for example via shared memory usage, which is part of our future work plan.

4.2. 64-bit integer arrays extension

Fig. 4 show the weak scaling for the *Mixture Large* test case up to 14 billion particles on Piz Daint (up to 2048 GPUs), Marconi (up to 1024 GPUs) and on JUWELS (up to 1536 GPUs) supercomputers. This case can be thought as a mixture of oil and water initially fully dispersed into each other and then gradually separating (see DL_MESO manual for more details). Starting from 1 GPU and 7 million particles, the number of GPUs and system size is doubled up to 2048 GPUs and 14.336 billion particles, respectively. In this case, the many-body DPD algorithm is not needed, so the scaling is purely based on the standard DPD formulation.

When only 1 GPU is used there is no overlap between communication and computation and the force calculation kernel is much faster. This explains the sharp drop in performance from 1 to 2

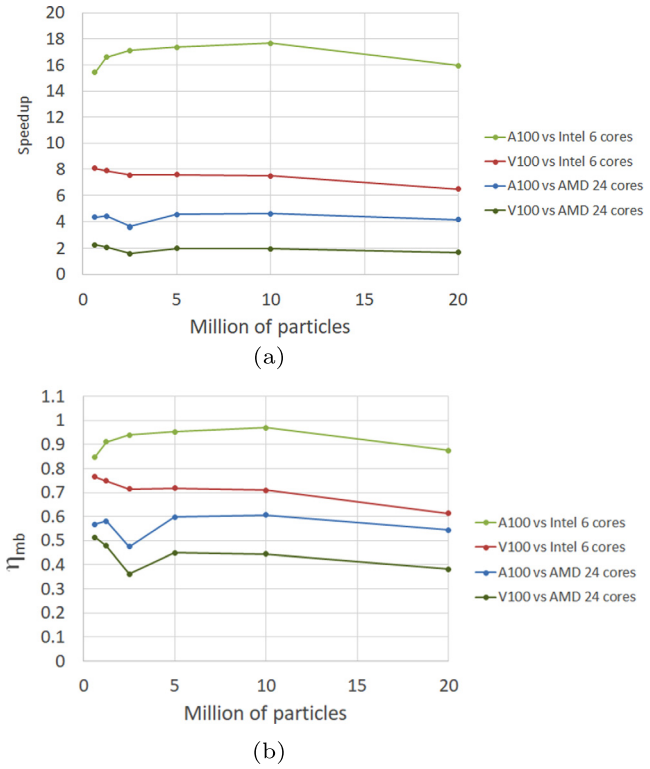


Fig. 3. Performance comparison between A100 and V100 NVidia GPU cards vs AMD 24-core Rome and Intel Gold 6-core CPUs.: a) speedup, b) efficiency. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

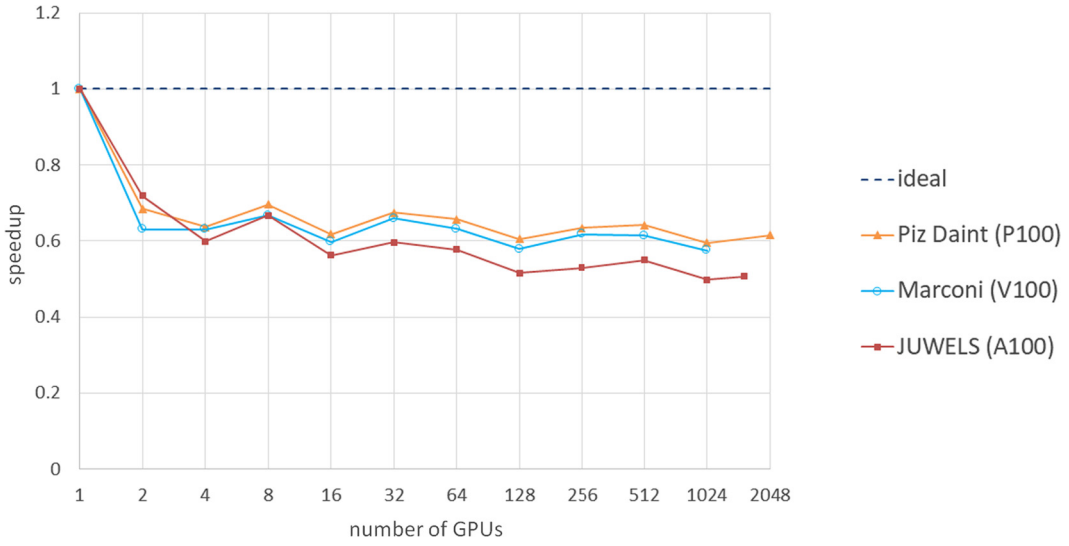


Fig. 4. Weak scaling for the Mixture Large test case using up to 14 billion particles.

GPUs, but after 2 a quasi-linear performance is observed across all 3 architectures.

4.3. Many-body DPD algorithm scaling

In the *Surface Drop* test case, a liquid vapour system coexists initially as fully dispersed and then slowly evolving towards a unique large drop of water. The system is 500x240x500 DPD length units with particle density 6.7 for a total of 400 million beads. The thickness of the surfaces of is 1 DPD length units. Table 1 resumes the DPD parameters (see DL_MESO manual for more details). The many-body DPD is used to replicate the vapour-liquid interactions and surface tension effects. Reflecting boundary conditions are used for the plate surfaces. The size of the droplet can be mapped to a real system by the appropriate use of dimensional analysis and dimensionless groups such as the capillary number [20] [21].

Fig. 5 shows the strong scaling obtained from 64 to 1048 GPUs. The lower limit of 64 GPus is due to GPU memory capacity. We do not use Unified Memory as it can have a negative impact on performance, so we are bound to the actual DRAM memory of 16GB for the P100 and V100 and 40GB (HMB2) on the A100. With 1024

Table 1
Parameters values for the many-body DPD test case Surface Drop.

specie A	specie B	A_{ij}	B_{ij}	C_{ij}	D_{ij}	E_{ij}	$r_{c,ij}$	γ_{ij}
water	water	-40	25	1.0	0	0	0	4.5
water	wall	-10	25	1.0	0	0	0	4.5
wall	wall	-40	25	1.0	0	0	0	4.5

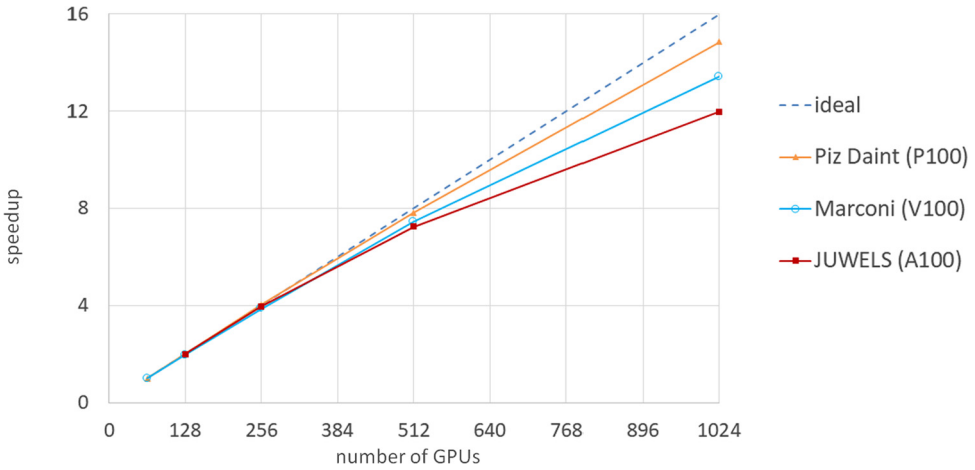


Fig. 5. Strong scaling of the Surface Drop test case on Piz Daint, Marconi and JUWELS supercomputers.

GPUs, the efficiencies are 93%, 84% and 75%, for Piz Daint, Marconi and JUWELS, respectively. These values are very close to those presented in the work of Castagna et al. [14] where 85% efficiency was maintained on Piz Daint up to 4096 GPUs.

Moreover, to highlight the importance of overlap between computation and communication in our algorithm, we run the same test case but mimicking a lack of overlap setting the *streamCompute* equal to the default stream. This has the effect of synchronize the two streams avoiding the overlap. The code will first compute the internal cell forces, then exchange the data and finally find the boundary cell forces. Fig. 6a, 6b and 6c show a comparison of the strong scaling obtained with and without overlaps.

On Piz Daint we observe an improvement from 60% to 93%, on Marconi from 59% to 84% and on JUWELS from 60% to 66%. As expected, the impact of the communication speed is greater on the older machine (Piz Daint). This is due to 2 effects: the memory bandwidth between GPUs (51.2 GB/s on Piz Daint, 150 GB/s on Marconi and 220 GB/s on JUWELS) and the number of GPU cards per node (1 on Piz Daint, 4 on Marconi and JUWELS).

This is also confirmed from the profile results in Figs. 7a and 7b obtained using the NVidia NSight System profiler. The system is the same surface drop case, with 15M particles of water on 8 GPUs. The timing of the *streamExchange* is of 29.1 ms without overlap and 18.7 ms with overlap, i.e. a $\sim x1.6$ speedup.

4.4. Memory usage and load balance between GPUs

The poorer scalability observed on the A100 system compared to the other two required further investigation. In particular we focused on the memory usage and the load imbalance occurring during this particular simulation of water drop formation from fully dispersed conditions.

While on Pizdaint and Marconi the memory usage starts from nearly full GPU capacity ($\sim$ 88% of 16GB), on the JUWELS supercomputer we only use 40% of the 40GB available per GPU (Fig. 8). With 1024 GPUs, the usage drops to 10% and 4%, respectively. Moreover, the memory bandwidth of the A100 is 1.5x higher than the V100 and 3x more than P100, drastically reducing the ratio between computation/communication and then limiting the overlap between the two. The communication between GPUs (mainly the inter-node communication, being the intra-node much faster as occurring via NVlink) has then a stronger impact on the A100 system.

Moreover, we realized that during the simulation the load balance between GPUs drastically changes: the number of beads per GPU is initially homogeneously distributed. However, as the system evolves and the particles starts to coalesce together, an imbalance on the number of particle per GPU occurs. This has two negative effects: on the computing side some GPUs will finish earlier and will be idle waiting for the other GPUs to finish. On the communication side, an imbalance on the amount of data transferred will occur, overloading some sections of the network and less others. This last effect exaggerates the communication time and then impact more on the fastest GPU, i.e. the A100.

The load imbalance can be corrected readjusting each GPU domain size according to the new number of particles. We decided to use A Load Balancing Library (ALL), developed in the Simulation Laboratory of Molecular Systems of the Jüelich Supercomputing Centre at Forschungszentrum Jüelich (DE), to calculate the new domains. In particular, we use the ALL Tensor-Product method to maintains orthogonality between domains without staggered borders. To exaggerate the impact of this imbalance we examined a much smaller system of 32k beads and 8 GPUs. Fig. 9 shows the load balance during the total simulation of 100k steps: in the first 20k steps the particles are still coagulating in small droplets (see Fig. 10) and the load balance is not too far from the ideal value of 12.5%. However, with time, a single large drop is formed (Fig. 11). The integration with the ALL library allows to control the balance around the ideal value. Investigation on the overall impact on scalability and performance on larger system is currently being carried out. Of course, if ALL is active the time for re-adjusting the domain needs also to be taken in account, but this is out of scope in this manuscript.

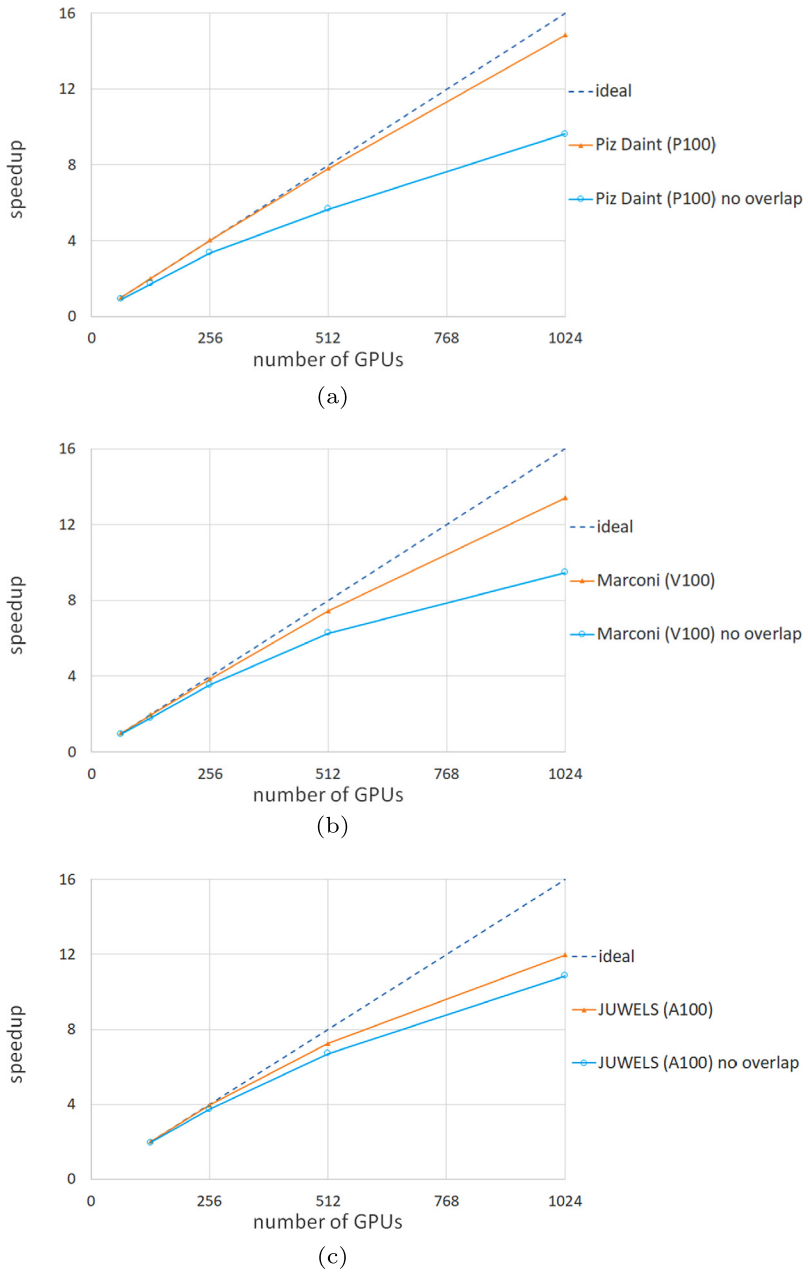


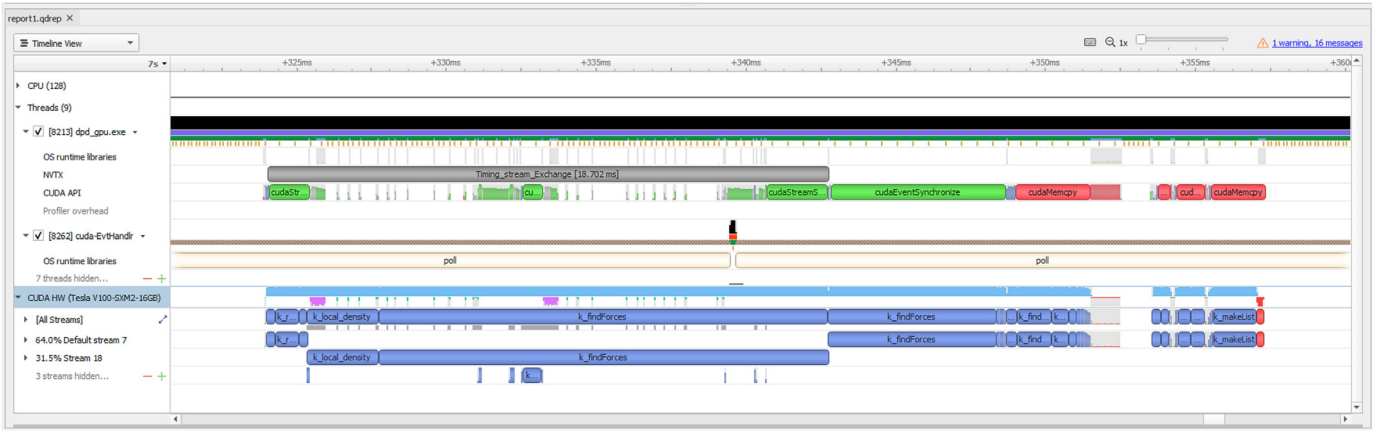
Fig. 6. Strong scaling of the Surface Drop test case with and without overlap between computation and communication: a) Piz Daint, b) Marconi and c) JUWELS.

We then conclude that the poorer scalability observed on the A100 at high number of GPUs is due to: a) communication between GPUs prevailing on the computation. This tends to be reduced if a larger number of particles is being used; b) load imbalance between GPUs workload, which tends to exaggerate the communication effects. Note as a higher number of GPUs could actually improve the scalability for imbalanced systems if the communication would not play a major role as in this case.

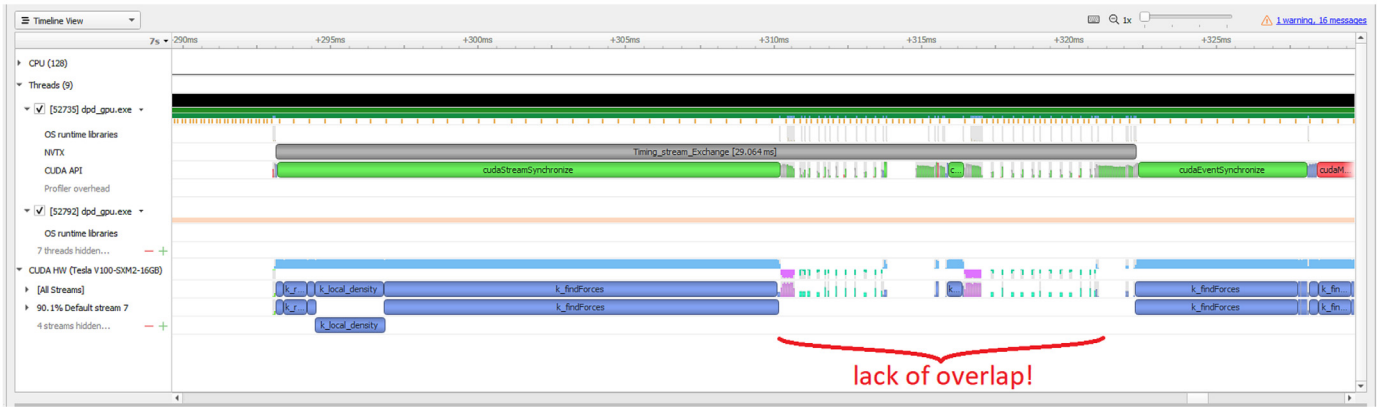
5. Conclusions

We presented a novel algorithm for the implementation of the many-body DPD on multi-GPU architectures which allows to keep high scalability on different architectures and across NVidia latest three generations of GPUs. Results on Piz Daint, Marconi and JUWELS European supercomputers shows good weak and strong scaling results up to 1048 GPUs with efficiency always above 75%. Moreover, we highlighted the impact of a correct overlap between computation and communication which leads to a $\sim x1.6$ speedup compared to a naive implementation. Results are also confirmed by in depth profiler analysis via NVidia NSight System profiler. With the extension to 64-bit integer arrays, we are able to simulate large mesoscale systems up to 14 billion particles.

Moreover, similar scalability is observed up to 512 GPUs across the three architectures. If the number of particles per GPU is not large enough, a poorer scalability is observed for a higher number of GPUs using the latest NVidia A100, due to a lack of full overlap between computation and communication. In systems where the load imbalance occurs, the slower communication can exaggerate this effect. However, as the performance is higher, the total number of GPUs required for a given problem is smaller and could be easily accommodated on the larger A100 memory. This increases the ratio computation/communication hiding more effi-



(a)



(b)

Fig. 7. NVidia NSight System profiler results: a) with and b) without overlap between computation and communications obtained on a surface drop system with 8 GPUs.

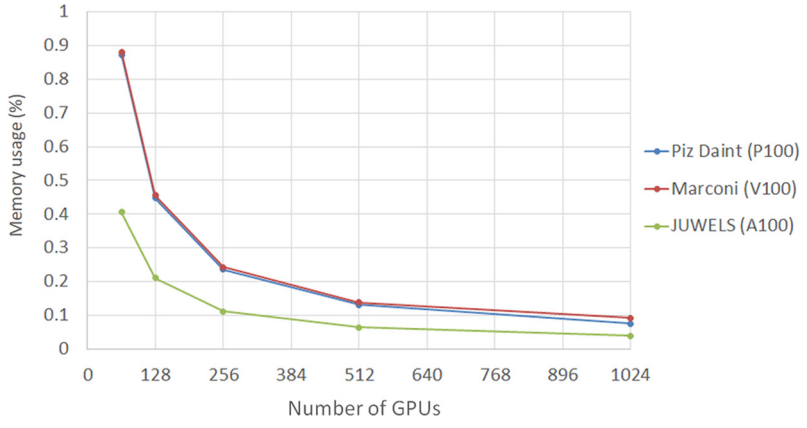


Fig. 8. Memory usage of the Surface Drop test case on Piz Daint, Marconi and JUWELS supercomputers.

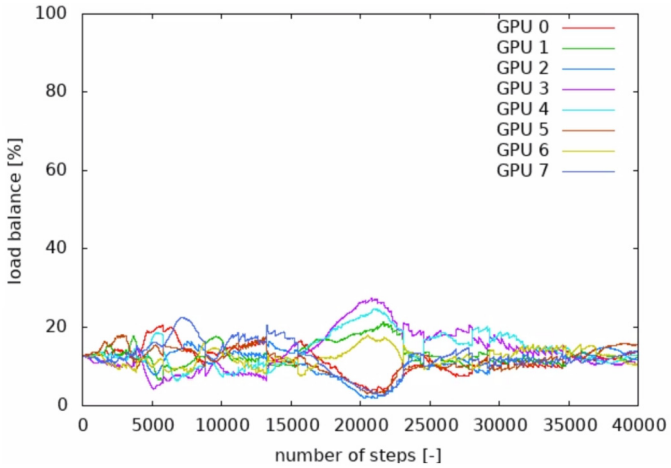


Fig. 9. Load imbalance on the Surface Drop test case.

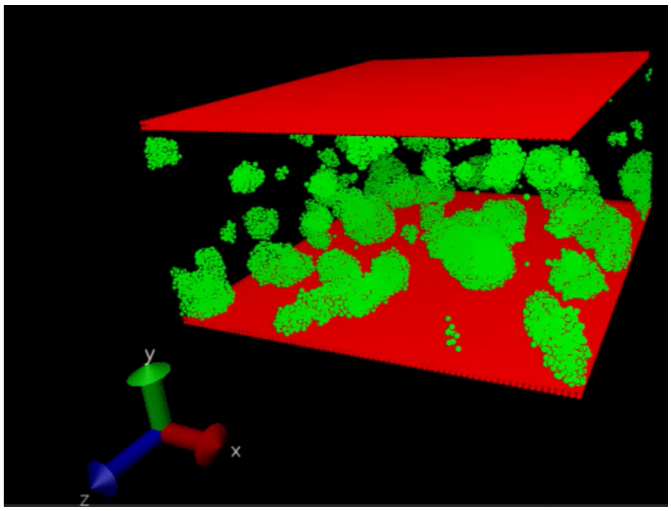


Fig. 10. Water drop formation after 20k steps: several groups of beads are coalescing together.

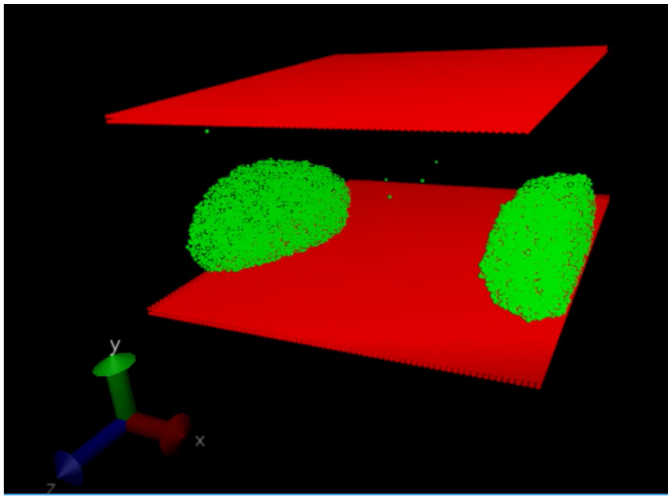


Fig. 11. Water drop formation after 100k steps: a single bubble crossing the periodic system is formed.

ciently the data transfer latency and then be the most convenient overall choice.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors are grateful to the European Project funded E-CAM, as part of the European Union's Horizon 2020 research and innovation program under the grant agreement No 676531, to the UK Research and Innovation - STFC Hartree Centre - and to the PRACE Summer of HPC 2019 program for funding provided to the authors.

References

[1] P.J. Hoogerbrugge, J.M.V.A. Koelman, Europhys. Lett. 19 (3) (1992) 155–160, https://doi.org/10.1209/0295-5075/19/3/001.
[2] R.D. Groot, P.B. Warren, J. Chem. Phys. 107 (11) (1997) 4423–4435, https://doi.org/10.1063/1.474784, http://aip.scitation.org/doi/10.1063/1.474784.
[3] P. Español, P.B. Warren, J. Chem. Phys. 146 (15) (2017) 150901, https://doi.org/10.1063/1.4979514, http://aip.scitation.org/doi/10.1063/1.4979514.
[4] I. Pagonabarraga, D. Frenkel, J. Chem. Phys. 115 (11) (2001) 5015–5026, https://doi.org/10.1063/1.1396848, https://aip.scitation.org/doi/10.1063/1.1396848.
[5] S.Y. Trofimov, E.L.F. Nies, M.a.J. Michels, J. Chem. Phys. 117 (20) (2002) 9383–9394, https://doi.org/10.1063/1.1515774, https://aip.scitation.org/doi/10.1063/1.1515774.
[6] A.W. Götz, M.J. Williamson, D. Xu, D. Poole, S. Le Grand, R.C. Walker, J. Chem. Theory Comput. 8 (5) (2012) 1542–1555, https://doi.org/10.1021/1ct200909j.
[7] S. Plimpton, J. Chem. Phys. 117 (1) (1995) 1–19, https://doi.org/10.1006/jcph.1995.1039, https://www.sciencedirect.com/science/article/pii/S002199918571039X.
[8] M.J.O. Abraham, T. Murtola, R. Schulz, S. Páll, J.C. Smith, B. Hess, E. Lindahl, GROMACS: High Performance Molecular Simulations Through Multi-Level Parallelism from Laptops to Supercomputers 1-2, Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States), Elsevier, 2015, https://www.osti.gov/pages/biblio/1252791.
[9] J.C. Phillips, D.J. Hardy, J.D.C. Maia, J.E. Stone, J.V. Ribeiro, R.C. Bernardi, R. Buch, G. Fiorin, J. Hénin, W. Jiang, R. McGreevy, M.C.R. Melo, B.K. Radak, R.D. Skeel, A. Singharoy, Y. Wang, B. Roux, A. Aksimentiev, Z. Luthey-Schulten, L.V. Kalé, K. Schulten, C. Chipot, E. Tajkhorshid, J. Chem. Phys. 153 (4) (2020) 044130, https://doi.org/10.1063/5.0014475, https://aip.scitation.org/doi/10.1063/5.0014475.
[10] M.J. Harvey, G. Giupponi, G.D. Fabritiis, J. Chem. Theory Comput. 5 (6) (2009) 1632–1639, https://doi.org/10.1021/ct9000685.
[11] J.A. Anderson, J. Glaser, S.C. Glotzer, Comput. Math. Sci. 173 (2020) 109363, https://doi.org/10.1016/j.commatsci.2019.109363, http://www.sciencedirect.com/science/article/pii/S0927025619306627.
[12] Y. Xia, J. Goral, H. Huang, I. Miskovic, P. Meakin, M. Deo, Phys. Fluids 29 (5) (2017) 056601, https://doi.org/10.1063/1.4981136, https://aip.scitation.org/doi/abs/10.1063/1.4981136.
[13] Y. Xia, A. Blumers, Z. Li, L. Luo, Y.-H. Tang, J. Kane, J. Goral, H. Huang, M. Deo, M. Andrew, Comput. Phys. Commun. 247 (2020) 106874, https://doi.org/10.1016/j.cpc.2019.106874, https://www.sciencedirect.com/science/article/pii/S0010465519302619.
[14] J. Castagna, X. Guo, M. Seaton, A. O'Cais, Comput. Phys. Commun. 251 (2020) 107159, https://doi.org/10.1016/j.cpc.2020.107159, https://www.sciencedirect.com/science/article/pii/S0010465520300199.
[15] M.A. Seaton, R.L. Anderson, S. Metz, W. Smith, Mol. Simul. 39 (10) (2013) 796–821, https://doi.org/10.1080/08927022.2013.772297.
[16] P. Español, P. Warren, Europhys. Lett. 30 (4) (1995) 191–196, https://doi.org/10.1209/0295-5075/30/4/001.
[17] W.C. Swope, H.C. Andersen, J. Chem. Phys. 102 (7) (1995) 2851–2863, https://doi.org/10.1063/1.468663, https://aip.scitation.org/doi/abs/10.1063/1.468663.
[18] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, second edition, Oxford University Press, 1987, https://oxford.universitypressscholarship.com/view/10.1093/oso/9780198803195.001.0001/oso-9780198803195.
[19] M. Revenga, I. Zúñiga, P. Español, Comput. Phys. Commun. 121–122 (1999) 309–311, https://doi.org/10.1016/S0010-4655(99)00341-0, https://www.sciencedirect.com/science/article/pii/S0010465599003410.
[20] P.-G. Gennes, F. Brochard-Wyart, D. Quéré, Capillarity and wetting phenomena, SpringerLink, http://link.springer.com/book/10.1007/978-0-387-21656-0.
[21] P.B. Warren, Phys. Rev. Lett. 87 (22) (2001) 225702, https://doi.org/10.1103/PhysRevLett.87.225702.