PHYSICAL REVIEW E **96**, 053311 (2017)

Scalable and fast heterogeneous molecular simulation with predictive parallelization schemes

Horacio V. Guzman,[1,*] Christoph Junghans,[2] Kurt Kremer,[1] and Torsten Stuehn[1]
[1]*Max Planck Institute for Polymer Research, Ackermannweg 10, 55128 Mainz, Germany*
[2]*Computer, Computational, and Statistical Sciences Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA*
(Received 28 July 2017; published 27 November 2017)

Multiscale and inhomogeneous molecular systems are challenging topics in the field of molecular simulation. In particular, modeling biological systems in the context of multiscale simulations and exploring material properties are driving a permanent development of new simulation methods and optimization algorithms. In computational terms, those methods require parallelization schemes that make a productive use of computational resources for each simulation and from its genesis. Here, we introduce the heterogeneous domain decomposition approach, which is a combination of an heterogeneity-sensitive spatial domain decomposition with an *a priori* rearrangement of subdomain walls. Within this approach, the theoretical modeling and scaling laws for the force computation time are proposed and studied as a function of the number of particles and the spatial resolution ratio. We also show the new approach capabilities, by comparing it to both static domain decomposition algorithms and dynamic load-balancing schemes. Specifically, two representative molecular systems have been simulated and compared to the heterogeneous domain decomposition proposed in this work. These two systems comprise an adaptive resolution simulation of a biomolecule solvated in water and a phase-separated binary Lennard-Jones fluid.

DOI: 10.1103/PhysRevE.96.053311

I. INTRODUCTION

Molecular simulation has proven to be of vital importance for driving the theory and the interpretation of experiments in diverse research fields, including soft matter research [1–7]. In particular, molecular dynamics (MD) methods have facilitated the study of systems consisting of millions of particles [8] and time scales up to milliseconds [9]. These milestones in MD simulations have been achieved by promoting two facts: the intrinsic highly parallel environment enabled by the MD algorithm and the development of new optimized algorithms for several MD systems [10–14]. Out of equilibrium (inhomogeneous) and multiscale simulations constitute recent milestones in molecular simulations of soft matter, which have driven in the past few years the development of new methods aiming to provide a physically consistent treatment, highly accurate and computationally efficient molecular modeling. In particular, concurrent multiscale modeling, mapping atomic configurations to coarse-grained models and partitioning the system in different resolution regions [15–18]. As well as several enhancements developed for studying inhomogeneous systems; like phase separated ones [19], crystal growth [20], biomembranes [21,22], and proteins [9,12].

Understanding soft matter at multiple time and length scales poses particular computational demands where microscopic physical phenomena strongly couple to macroscopic representations, e.g., by adjusting resolutions on the fly between models of multiple costs. Consequently, suited parallelization schemes should keep these considerations in mind and meet both requirements: scalability and simulation speed. Furthermore, available load-balancing algorithms are based on measurements of computational runtime, although they are limited for predicting the domain decomposition of an inhomogeneous simulation from its initial configuration. A similar situation arises when tackling out-of-equilibrium molecular systems with prescribed levels of heterogeneity, which are commonly given as characteristics and topology of the different particles within the simulation setup.

In this article, we introduce the heterogeneous spatial domain decomposition algorithm (HeSpaDDA). This algorithm is not based in any runtime measurements, it is rather a predictive computational resources allocation scheme. The HeSpaDDA algorithm is a combination of an heterogeneity (resolution or spatial density) sensitive processor allocation with an initial rearrangement of subdomain walls. The initial rearrangement of subdomain walls within HeSpaDDA anticipates favorable cells distribution along the processors per simulation box axis by moving cell boundaries according to the resolution of the tackled region in the molecular system. In a nutshell, the proposed algorithm will make use of *a priori* knowledge of the system setup. Specifically, the region that is computationally less expensive. This inherent load-imbalance could come from different resolutions or different densities. The algorithm will then propose non-uniform domain layout, i.e., domains of different size and its distribution amongst compute instances. This could lead to significant speedups for systems of the aforementioned type over standard algorithms,e.g., spatial domain decomposition (DD) [23] or spatial and force based DD [24].

The multiscale simulation method we have chosen to validate the capabilities of HeSpaDDA is the adaptive resolution scheme (AdResS). In particular, for a dual resolution simulation like AdResS we tackle an expensive model in the high-resolution region and a less-expensive model elsewhere. In computational terms, this means that some processors have more work to do than others at the initial domain decomposition of a dual resolution simulation. AdResS methods have been employed for the concurrent simulation of diverse length scale systems interfacing different resolutions of simulation techniques, ranging from concurrent simulations of classical atomistic and coarse-grained models [25–32], to coupling classical atomistic and path-integral models [33–35], as well as interfacing particle-based simulations with continuum mechanics [16,17].

[*]vargas@mpip-mainz.mpg.de

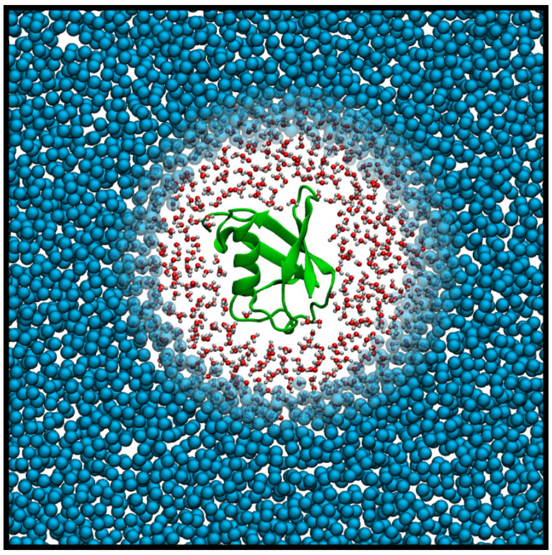

FIG. 1. AdResS simulation of an atomistic protein and its atomistic hydration shell, coupled to a coarse-grained particle reservoir via a transition region [30].

The bimolecular AdResS simulation (see Fig. 1) tackled with HeSpaDDA, is a cubic simulation system with a spherical high-resolution region, comprised by an aqueous solution of the regulatory protein ubiquitin [30]. Understanding biomolecular function is fundamental to a broad variety of research domains, including drug discovery, food processing, and bioprocess engineering, to name but a few. In recent years, the AdResS approach has been successfully applied to an increasing number of biomolecular systems [30,32,36], with the goal of obtaining both computational speedup and additional insight in the systems properties.

In the case of inhomogeneous single-scaled systems we have tackled the system described in Fig. 2, which contains two phase-separated fluids of Lennard-Jones particles presenting a 6 to 1 σ ratio of heterogeneity.

The capabilities of the HeSpaDDA algorithm have been benchmarked by using two MD simulation packages. The AdResS simulations have been carried out by employing the ESPResSo++ MD package (static domain decomposition). While for the single-scale phase separated 6 to 1 Lennard-Jones binary fluid has been simulated with the GROMACS MD package and makes use of its embedded dynamic load-balancing features. However, the HeSpaDDA algorithm can be implemented within any other MD packages supporting multiscale and / or inhomogeneous simulations.

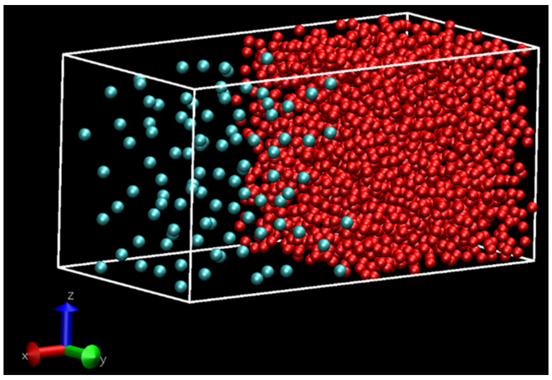

FIG. 2. Lennard-Jones system of a binary fluid separated by two phases of equal volume but different σ (6 to 1).

II. ADAPTIVE RESOLUTION SCHEME

In the adaptive resolution simulation scheme (AdResS), a region in space is defined in which molecules are modeled using atomistic detail, while coarse-grained models are used elsewhere (see Fig. 1). Particles can freely diffuse between atomistic and coarse-grained regions, smoothly changing their resolution as they cross a hybrid or transition region (see Fig. 3).

Here, we review the theoretical basis behind the adaptive resolution methodology. A typically small part of the system, the atomistic (AT) region, is described on the atomistic level and coupled via a hybrid (HY) transition region, to the coarse-grained (CG) region, where a coarser, computationally more efficient model is used. The interpolation is achieved via a resolution function $\lambda(\mathbf{R}_\alpha)$, a smooth function of the center of mass position $\mathbf{R}_\alpha$ of molecule α. For each molecule, its instantaneous resolution value $\lambda_\alpha = \lambda(\mathbf{R}_\alpha)$ is calculated based on the distance of the molecule from the center of the AT region. It is 1 if the molecule resides within the AT region and smoothly changes via the HY region to 0 in the CG region (see Fig. 3).

The interpolation between atomistic and coarse-grained nonbonded forcefields can be performed at the level of energies (Hamiltonian-AdResS) or of forces (Force-AdResS); as explained in the following section, for this publication we employ the Force-AdResS method.

Note that, in addition to the nonbonded interactions, bonded potentials are also usually present. In typical AdResS applications they are not subject to interpolation. As these are computationally significantly easier to evaluate, they do not play a role in the parallelization scheme presented here.

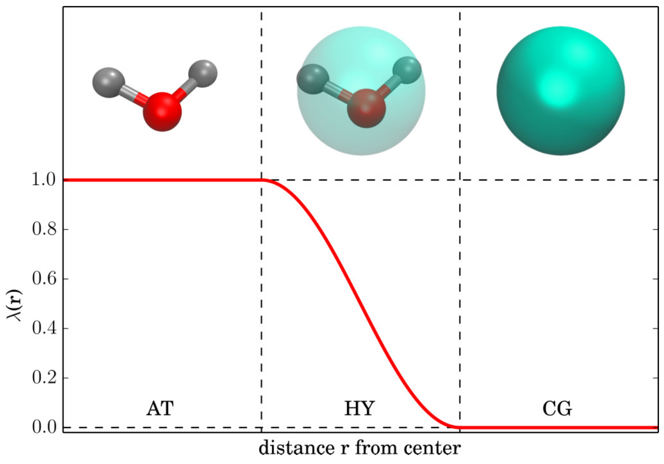

FIG. 3. Schematic of the resolution function $\lambda(\mathbf{R}_\alpha)$ used for the interpolation between the AT and CG forcefields in the adaptive resolution methodology. In the presented case, an atomistic water model is coupled with a coarse-grained one-particle-per-molecule description in the CG region.

Force interpolation

In the force interpolation scheme, the original AdResS technique [15,25], two different nonbonded force fields are coupled as

$$\mathbf{F}_{\alpha|\beta} = \lambda(\mathbf{R}_\alpha)\lambda(\mathbf{R}_\beta)\mathbf{F}^{\mathrm{AT}}_{\alpha|\beta} + [1 - \lambda(\mathbf{R}_\alpha)\lambda(\mathbf{R}_\beta)]\mathbf{F}^{\mathrm{CG}}_{\alpha|\beta}, \quad (1)$$

where $\mathbf{F}_{\alpha|\beta}$ is the total force between the molecules α and β and $\mathbf{F}^{\mathrm{AT}}_{\alpha|\beta}$ defines the atomistic force-field, which is decomposed into atomistic forces between the individual atoms of the molecules α and β. Finally, $\mathbf{F}^{\mathrm{CG}}_{\alpha|\beta}$ is the coarse-grained force between the molecules, typically evaluated between their centers of mass.

III. HETEROGENEOUS SPATIAL DOMAIN DECOMPOSITION ALGORITHM

Molecular simulations in soft-matter investigations, such as polymer and biomolecular research, require the evaluation of efficient parallelization schemes, e.g., algorithms [10,12,14,23,24,37]. In particular, initial domain decomposition algorithms can be based on the distribution of atoms, forces, or the space in a given simulation domain, and also a combination of both forces and space [24]. After a certain amount of simulation time steps, dynamic load balancing (DLB) schemes can be activated, like the work-sharing [38–40] and work-stealing [41–43] ones. In general, distributed runtime systems with high performance computing (HPC) focus are employed to achieve DLB, such as: Charm++ [44], HPX [45], Legion [46], among others. However, initial domain decomposition algorithms are not aware of the inherent region-based distribution of multiscale and/or inhomogeneous systems from the simulation setup. The idea behind an initial and predictive DD algorithm is to reach higher scalability and efficiency by assuming a steady runtime behavior of the simulation platform and hence deliver the domain decomposition within the simulation setup and without any initial overhead in terms of load per processor. Naturally, the proposed algorithms are complementary to the dynamic load-balancing ones that will take care of the platform-dependent imbalance along extensive production runs. Interestingly, the trigger for such DLB algorithms could be extended by using HeSpaDDA, e.g., for the Lennard-Jones binary mixture described in this article, DLB could be required only after 100 000 MD steps. Whereby the common trigger used per default in DLB algorithms is in the range from 1 to 100 MD steps.

Here, we introduce heterogeneous spatial domain decomposition algorithm (HeSpaDDA), which is a combination of an heterogeneity (resolution or density) sensitive spatial domain decomposition (sDD) with an initial subdomain-walls rearrangement. The fact that the simulation setup presents a degree of heterogeneity along one of the simulation box axes, like the AdResS (resolution heterogeneous) or Lennard-Jones binary mixture ones (density heterogeneous), allows us to identify the simulation box regions with high and low resolutions and hence assign computational resources according to the resolution and volume ratios between those regions from the very beginning of the production run (see Fig. 4).

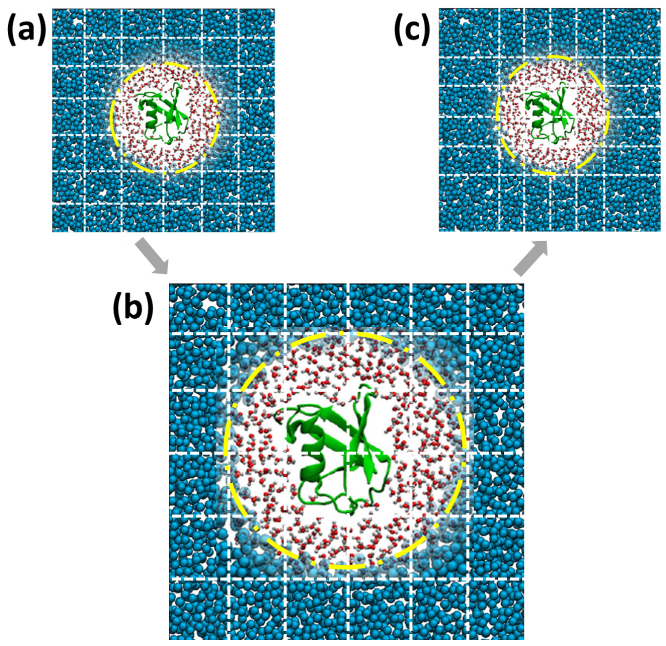

FIG. 4. AdResS simulation of an atomistic protein, its atomistic hydration shells and CG water particles. (a) The dashed lines illustrate the spatial domain decomposition (sDD) of such simulation; (b) shows the intermediate state where only the high-resolution region (dash-dotted circle in yellow) has been spatially rescaled to reach the same resolution as the low-resolution one, then in (c) the processors are allocated according to the heterogeneous spatial domain decomposition algorithm (HeSpaDDA) by distinguishing the resolution region type. Note that in (c) no more spatially rescaling is shown and the initial system configuration is reached. In all cases the total grid is 6×6 processors; however, the processors allocation in the high-resolution region ($P_{\mathrm{HR}} = P^x_{\mathrm{HR}} * P^y_{\mathrm{HR}}$) changes from (a) to (c), namely, from 2×3 to 4×4 subdomains. Due to illustrative reasons, only two dimensions are depicted.

HeSpaDDA can be divided into two main modules: the first one is devoted to the allocation of processors along the simulation box axes and the second one to the cells partitioning of resolution-dependent volumes to processors subdomains. Here each cell is defined as $r_{\mathrm{cell}} \leqslant r_c + r_s$, with r_c and r_s as the cutoff and skin lengths, respectively [47]. A prevailing reference value used in the modeling is $r_{\mathrm{cell}} = r_c + r_s$ [6]. The first module is of crucial importance because the processor allocation defines the productive scaling of the system to be simulated already at the initial configuration of the simulation run. The development of HeSpaDDA is illustrated in Fig. 4, where we observe how the subdomain-walls rearrangement is performed. First, the processor allocation as for sDD are shown [Fig. 4(a)], then the high-resolution region (inside the dash-dotted circle in yellow) has been temporarily rescaled only for processor allocation special purpose [Fig. 4(b)]. Once the processor allocation has been set, the processors are adjusted to the regions according to the low- or high-resolution type [Fig. 4(c)]. Moreover, HeSpaDDA includes two types of system configurations, i.e., cubic and noncubic setups. From here on, in this paper the setup with spherical high-resolution region corresponds to a cubic case (Fig. 1) and the slablike to the noncubic one (Fig. 2). While cubic configurations tend to present the same amount of processors per simulation box axes, i.e., $P_x = P_y = P_z$ to achieve efficient scaling, the noncubic configuration requires an additional optimization step to define the best triplet (P_x, P_y, P_z). In particular, for the system described in Sec. IV B P_x is independent from P_y and P_z (with $P_y = P_z$). Note here that HeSpaDDA is able to handle any box axes configuration required by the system setup (flow chart shown in the Appendix A).

In the second module, the algorithm receives the number of processors per region as an input and starts distributing cells along box axes in each subdomain according to the resolution type. Furthermore, this module controls the size and cubicity of all cells (see Sec. III B). A successful cell's cubicity check guarantees that the communication between surfaces of different neighbors-cells of the parallelepiped are or tend to be equal in each box axes X, Y, and Z. Interestingly, this holds for homogeneous and inhomogeneous systems where HeSpaDDA is applied. The intrinsic discrepancies between low- and high-resolution regions are considered in the method by a differentiated distribution of cells according to the type of resolution in each region. For more details on the algorithm flow chart, see Appendix A.

The validation of the HeSpaDDA algorithm has been performed with two archetypical heterogeneous systems, a biomolecule solvated in water (Force-AdResS) and a phase-separated 6-to-1 Lennard-Jones binary fluid. Both systems represent simulation setups, where the combination of this initial domain decomposition algorithm and dynamic load balancers is expected to increase productivity along an extensive production run (as verified in Sec. V B). Likewise, for slowly varying systems (within this article referred to simulations demanding Verlet-list rebuilds with periodicity higher than 7 $steps_{\mathrm{MD}}$) the HeSpaDDA algorithm can also be used to adapt the cell-grid of concurrent multiscale simulations based on performance indicators, such as the number of interactions or particle density per processor.

A. Modeling and scaling laws for HeSpaDDA

To circumvent the inclusion of platform-dependent run times we based the scaling law for the computation time of HeSpaDDA on the approach proposed in Ref. [23] [see Eq. (2)]. This approach considers the computation time t in the context of a homogeneous and cubic system by considering an MD integrator in accordance to the velocity Verlet and linked-cell-list (LCL) algorithm. In Eq. (2), the first term reflects pure force calculation time assuming that computed interactions are short-range. Moreover, the second term tackles the particle positions exchange between adjacent-processors in three dimensions. The particle positions need to be updated in every time step, and hence it is part of the computation time t, leading to

$$t \propto \frac{N_{\mathrm{AT}}}{2P} + 6 r_{\mathrm{cell}} k_{\mathrm{hw}} \left(\frac{N_{\mathrm{AT}}}{P}\right)^{2/3}, \tag{2}$$

where N_{AT} is the total number of atomistic particles, P is the total number of processors, r_{cell} is the sum of the interaction cutoff radius and skin length, and k_{hw} is a constant describing the communication taking place in each force calculation with strong dependence on the underlying hardware platform. From here on, this constant will be defined as $k_{\mathrm{hw}} \equiv 1$, emphasizing the exclusion of platform-dependent run times.

Data communication modeling in high-performance computing is a subject of constant development [48,49]. In particular, for message-passing parallel programming models where blocking, as well as nonblocking, communications while synchronizing nearest-neighbor processes affect the performance of the simulations [50].

Taking the first term of Eq. (2) as a starting point, we consider two resolutions within the same simulation box and hence the number of processors located in both resolutions, as well. The number of particles in each resolution region depends on the volume ratio of each resolution region and the total volume V. Having for the low-resolution (LR) region,

$$N_{\mathrm{LR}} = \frac{V_{\mathrm{LR}}}{V} N, \tag{3}$$

where N_{LR} is the number of particles in the low-resolution region and $\frac{V_{\mathrm{LR}}}{V}$ is the ratio of volumes between the low-resolution region and the total simulation box. Note that N is the total number of particles represented at the lowest resolution. Mathematically, N is defined as $N = N_{\mathrm{LR}} + N_{\mathrm{HR}}/R^{\mathrm{res}}_{\mathrm{SH}}$ with *spatial heterogeneity resolution ratio* $R^{\mathrm{res}}_{\mathrm{SH}}$ presented in Eq. (5). Furthermore, the number of particles in the high-resolution (HR) region are given by

$$N_{\mathrm{HR}} = R^{\mathrm{res}}_{\mathrm{SH}} \frac{V_{\mathrm{HR}}}{V} N, \tag{4}$$

with

$$R^{\mathrm{res}}_{\mathrm{SH}} = \frac{N^{\mathrm{res}}_{\mathrm{HR}}}{N^{\mathrm{res}}_{\mathrm{LR}}}, \tag{5}$$

where $N^{\mathrm{res}}_{\mathrm{HR}}$ are the number of entities in the high-resolution region that correspond to one entity in the low-resolution one $N^{\mathrm{res}}_{\mathrm{LR}}$. Mapping the atomistic water molecule to the coarse-grained model (as depicted in Fig. 3) results in $R^{\mathrm{res}}_{\mathrm{SH}} = 3$. The ratio of volumes between the high-resolution region and the total simulation box is $\frac{V_{\mathrm{HR}}}{V}$, comprised in Eq. (4).

From the viewpoint of the total number of processors, a straightforward spatial domain decomposition approach would decompose the system by taking into account the volume ratio of the different resolutions and the total simulation box. Letting the number of processors allocated in the low-resolution region as

$$P_{\mathrm{LR}} = \frac{V_{\mathrm{LR}}}{V} P. \tag{6}$$

Under such perspective, the processors allocated in the high-resolution region for sDD are given by

$$P_{\mathrm{HR}} = \frac{V_{\mathrm{HR}}}{V} P, \tag{7}$$

where $\frac{V_{\mathrm{HR}}}{V}$ is the volume ratio of the high-resolution region and the total simulation box. The modeling given by Eqs. (3), (4), (6), and (7) can be written as an extension to Eq. (2),

$$t_{\mathrm{sDD}} \propto \frac{1}{2}\left(\frac{N_{\mathrm{LR}}}{P_{\mathrm{LR}}} + \frac{N_{\mathrm{HR}}}{P_{\mathrm{HR}}}\right) + 6 r_{\mathrm{cell}} k_{\mathrm{hw}} \left(\frac{N_{\mathrm{LR}}}{P_{\mathrm{LR}}} + \frac{N_{\mathrm{HR}}}{P_{\mathrm{HR}}}\right)^{2/3}. \tag{8}$$

The term $(\frac{N_{\text{LR}}}{P_{\text{LR}}} + \frac{N_{\text{HR}}}{P_{\text{HR}}})^{2/3}$ due to particles communication also includes the coupling term of the communication between the high- and low-resolution regions. Moreover, the coupling or resolution transition effect of the pure force calculation of low and high resolutions is included within the term $(\frac{N_{\text{LR}}}{P_{\text{LR}}} + \frac{N_{\text{HR}}}{P_{\text{HR}}})$ of Eq. (8). Following the AdResS method described in Sec. II, the resolution transition effect would be reflected in the hybrid region (see Fig. 3).

The simplest and straightforward model for the spatial allocation of particles per processor is reflected in Eq. (8), with the volume resolution ratios shown in Eqs. (3), (4), (6), and (7). However, for heterogeneous simulations, such an approach has two drawbacks: there is an imbalanced distribution of particles per processor from the beginning of the simulation and the communication overhead is not minimized for the heterogeneous transition regions [38]. The results of employing the model given by Eq. (8) could hinder both the scalability and speed of the tackled heterogeneous simulation (as demonstrated in Sec. V). The development of HeSpaDDA's model is based on a spatial distribution of particles of heterogeneous systems, which fulfills the criteria given in Eq. (9). By substituting Eqs. (3) and (4) with $P_{\text{LR}} = P - P_{\text{HR}}$, a new equation for P_{HR} has been found [see Eq. (11)]. Consequently, HeSpaDDA modeling treats both low- and high-resolution regions as equals in terms of the allocation of particles per processor:

$$\frac{N_{\text{HR}}}{P_{\text{HR}}} = \frac{N_{\text{LR}}}{P_{\text{LR}}}. \tag{9}$$

To this end, Eqs. (6) and (7) turn into Eqs. (10) and (11). Interestingly, the proposed heterogeneous domain decomposition method is sensitive to both the region size and the heterogeneity types (resolution) for the processor allocation. Such modeling optimizes the initial or static distribution of load and hence minimizes communication overhead. In this new context, the number of processors distributed in the low-resolution region is defined as

$$P^{\text{hDD}}_{\text{LR}} = P - P^{\text{hDD}}_{\text{HR}}, \tag{10}$$

where $P^{\text{hDD}}_{\text{HR}}$ are the number of processors distributed in the high-resolution region according to the HeSpaDDA algorithm. From now on, the nomenclature used in the equations with superscript hDD correspond to the HeSpaDDA method. By replacing Eqs. (3), (4), and (10) into Eq. (9) leads to

$$P^{\text{hDD}}_{\text{HR}} = \frac{R^{\text{res}}_{\text{SH}} V_{\text{HR}}}{V + V_{\text{HR}}\big(R^{\text{res}}_{\text{SH}} - 1\big)} P. \tag{11}$$

The concept behind the coefficient $R^{\text{res}}_{\text{SH}}$ in Eq. (11) is to emulate an increase of the high-resolution volume by rescaling it to low-resolution units as shown in Fig. 4(b). Moreover, to be consistent with such high-resolution region increment, the denominator of Eq. (11) has been also expanded to $V + V_{\text{HR}}(R^{\text{res}}_{\text{SH}} - 1)$. Note here that V already contains one volume V_{HR}, and hence the denominator coefficient is $(R^{\text{res}}_{\text{SH}} - 1)$. Here, we remark that neither the simulation parameters V_{HR} nor V are scaled or increased for the simulation run. Nonetheless, HeSpaDDA employs the volume rescaling for achieving a balanced processor allocation among the different spatial resolution regions. In light of the limitation of allocating processors per simulation box axes at the simulation start, the number of processors will be rounded up to the closest integer for $P^{\text{hDD}}_{\text{HR}}$ and $P^{\text{hDD}}_{\text{LR}}$ is calculated from Eq. (10).

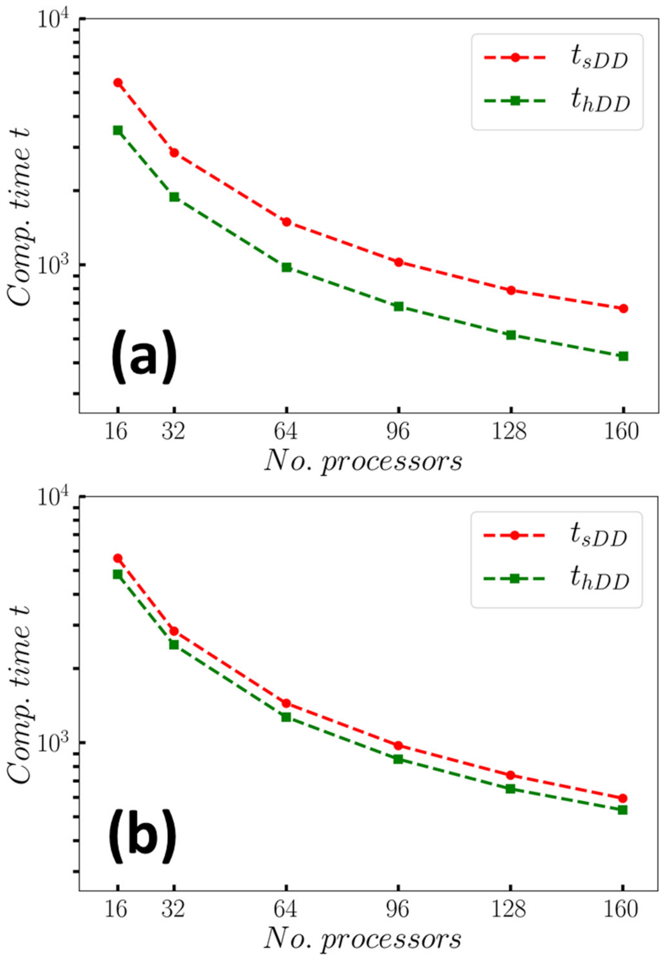

FIG. 5. Theoretical calculation of the computational time for the spatial domain-decomposition algorithm [see Eq. (8)] and the HeSpaDDA one [see Eq. (12)], as a function of a variable number of processors P. The calculation parameters for the multiscale and cubic system (a) are: $r_c = 1.0, r_s = 0.3, r_{\text{cell}} = r_c + r_s, R^{\text{res}}_{\text{SH}} = 3, N = 38\,084 + (76\,916/R^{\text{res}}_{\text{SH}})$, $V = 1157.625\sigma^3$, $V_{\text{HR}} = 140.608\sigma^3$, and $V_{\text{LR}} = V - V_{\text{HR}}$. The calculation parameters for the inhomogeneous and noncubic system (b) are: $r_c = 0.9$, $r_{\text{cell}} = r_c$, $R^{\text{res}}_{\text{SH}} = 6$, $N = 20\,828 + (124\,964/R^{\text{res}}_{\text{SH}})$, $V = 8192\sigma^3$, $V_{\text{HR}} = 4096\sigma^3$, and $V_{\text{LR}} = V - V_{\text{HR}}$. All in reduced units with $\sigma = \epsilon = m = 1$.

Thus, the time-scaling law for the HeSpaDDA is

$$t_{\text{hDD}} \propto \frac{1}{2}\left(\frac{N_{\text{LR}}}{P^{\text{hDD}}_{\text{LR}}} + \frac{N_{\text{HR}}}{P^{\text{hDD}}_{\text{HR}}}\right) + 6 r_{\text{cell}} k_{\text{hw}} \left(\frac{N_{\text{LR}}}{P^{\text{hDD}}_{\text{LR}}} + \frac{N_{\text{HR}}}{P^{\text{hDD}}_{\text{HR}}}\right)^{2/3}. \tag{12}$$

In Fig. 5(a) theoretical comparison between the scaling laws of Eqs. (8) and (12) has been carried out. Whereby the new algorithm (HeSpaDDA) improves the performance of the former spatial domain decomposition for two archetypical heterogeneous systems. From the viewpoint of this theoretical framework, the signatures of runtime-based "dynamic"

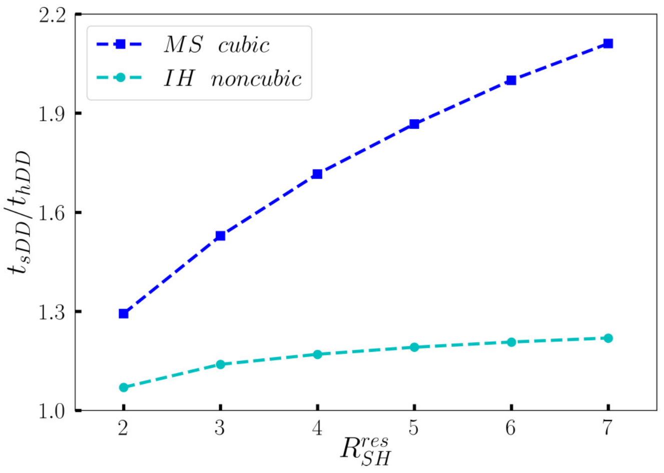

FIG. 6. Theoretical calculation of the computational time for the spatial domain-decomposition algorithm [see Eq. (8)] over the HeSpaDDA one [see Eq. (12)], as a function of variables (a) $R^{\rm res}_{\rm SH}$ with the simulation parameters: $R^{\rm res}_{\rm SH} = [2,3,4,5,6,7]$ and $N = 38\,084 + (76\,916/R^{\rm res}_{\rm SH})$. The calculation parameters for the multiscale "MS" and cubic system are: $r_c = 1.0$, $r_s = 0.3$, $r_{\rm cell} = r_c + r_s, V = 1157.625\sigma^3$, $V_{\rm HR} = 140.608\sigma^3$, $V_{\rm LR} = V - V_{\rm HR}$, and $P = 64$. The calculation parameters for the inhomogeneous "IH" and noncubic system (b) are: $r_c = 0.9, r_{\rm cell} = r_c$, $R^{\rm res}_{\rm SH} = 6$, $N = 20\,828 + (124\,964/R^{\rm res}_{\rm SH})$, $V = 8192\sigma^3$, $V_{\rm HR} = 4096\sigma^3$, and $V_{\rm LR} = V - V_{\rm HR}$. All in reduced units with $\sigma = \epsilon = m = 1$.

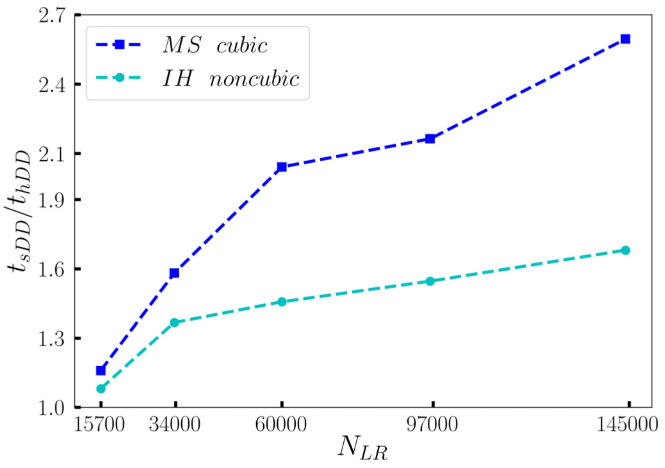

FIG. 7. Theoretical calculation of the computational time for the spatial domain-decomposition algorithm [see Eq. (8)] over the HeSpaDDA one [see Eq. (12)], as a function of variables (b) volume V and hence N to maintain a constant density with the specific simulation parameters: $N^{\rm LR} = N^0_{\rm LR}(V_T/V^0_T)$ with $V^0_T = 1157$, $N^0_{\rm LR} = 38\,084$ and $N_{\rm HR} = 76\,916$. The calculation parameters for the multiscale "MS" and cubic system are: $r_c = 1.0$, $r_s = 0.3$, $r_{\rm cell} = r_c + r_s$, $V_{\rm HR} = 5.2^3$, $V_{\rm LR} = V - V_{\rm HR}$, and $P = 160$. The calculation parameters for the inhomogeneous "IH" and noncubic system (b) are: $r_c = 0.9$, $r_{\rm cell} = r_c$, $N^{\rm LR} = N^0_{\rm LR}(V_T/V^0_T)$, with $V^0_T = 32 \times 16 \times 16$, $N^0_{\rm LR} = 20\,828$, and $N_{\rm HR} = 124\,964$, $V_{\rm HR} = V/2$, and $V_{\rm LR} = V - V_{\rm HR}$. All in reduced units with $\sigma = \epsilon = m = 1$.

imbalance is disregarded. Certainly the modeling of the here-presented heterogeneous algorithm is static. Nevertheless, in the simulation code the method can be combined to external dynamic load-balancing schemes (shown in Sec. V).

The inhomogeneous system tackled in Fig. 5(a) is similar in terms of simulation parameters to the ubiquitin solvated in water (see Sec. IV A). However, in this theoretical example we considered pure atomistic water in the high-resolution region (no ubiquitin in the high-resolution region). While Fig. 5(b) resembles the phase separated binary Lennard-Jones fluid. Calculating the corresponding computational times for sDD [Eq. (8)] and hDD [Eq. (12)] as a function of P we observed in all studied cases improved timing for HeSpaDDA.

Also from the modeling perspective, the ratio $t_{\rm sDD}/t_{\rm hDD}$ varies depending on the tackled system, for the cubic system a HeSpaDDA reaches a factor $\approx$1.5 [Fig. 5(a)], while for the noncubic system this factor is $\approx$1.3 [Fig. 5(b)]. These results suggest that decomposing the total number of processors in a balanced way as a function of the different system resolutions and asymmetric simulation box axes has a higher limitation since the processors triplets shall be integer, i.e., $P_x, P_y, P_z \in$ integers. In other words, for cubic systems the resolutions are scaled equally in every simulation box axis while for noncubic ones this is tackled for each axis and each resolution. To further understand the scaling of multiple resolution simulations and the proposed modeling for the HeSpaDDA algorithm, we also show how the latter scales with respect to the sDD as a function of $R^{\rm res}_{\rm SH}$ (Fig. 6) and N (Fig. 7). Note that here direct calculations have been performed which are not taking into account any platform-dependent communication patterns. In Fig. 6 we explore how the relative speedup $t_{\rm sDD}/t_{\rm hDD}$ performs as a function of a variable resolution ratio $R^{\rm res}_{\rm SH}$, which in this illustrative scenario varies from 2:1 to 7:1.

According to this modeling result we show how the relative speedup increases monotonically with $R^{\rm res}_{\rm SH}$ for both systems. Although the increment of the noncubic system is significantly lower than the one given for the cubic system, as explained earlier this is due to the sensitivity on the different resolutions in each axis of the simulation box. Moreover, for the cubic case three axes are dual-resolution, while for the noncubic one, only one axis is prescribed to be dual-resolution (see Fig. 11 in Appendix A). We also observe a monotonic increment in relative speedup as long as the low-resolution region grows and the high-resolution region remains constant for both systems, which is plotted as a function of total number of particles in Fig. 7. It is worth noting that the scaling for the noncubic system as a function of the increment of particles in the low resolution region goes qualitatively in-line with the cubic system but presenting less speedup. The reason of such behavior is that the noncubic system in terms of processor allocation grows only in one of the simulation box axes (see also Fig. 2).

A clear limitation of the scaling laws presented in this section is that the underlying communication pattern of the computing platform is unknown and thus no estimations of the platform-related load imbalances can be predictively taken into account. Unless the employed MD package includes a very fine tuned dynamic load-balancing algorithm. Despite this fact, the information given by the scaling Eq. (12) serves as qualitative lower boundary for estimating the duration of inhomogeneous simulation as a function of the modeled variables.

B. Algorithm description

1. *Processor allocation*

The proper allocation of processors in heterogeneous molecular simulations is crucial for the intrinsic computational scaling and performance of the production run. In other words, the computational scaling of heterogeneous simulation at this stage is not limited by the MD package used. However, it is constrained to the initial domain decomposition and hence the correspondence of the number of processors to different resolution regions of the initial given configuration. One of the aspects to consider for spatially decomposed systems is the dimensional sensitivity which is the parallelepiped characteristics of the simulation setups, i.e., cubic or noncubic. A second aspect to consider is the *spatial heterogeneity* ratio $R_{\mathrm{SH}}^{\mathrm{res}}$ [as defined in Eq. (5)] and the third is definitely the volumes of high and low resolution regions V_{HR} and V_{LR}, respectively. More details about the algorithm flow chart and implementation are given in the Appendix A.

2. *Cell partitioning*

Once the three-dimensional processors grid has been built, cell partitioning is required (this flow chart is illustrated in the Appendix A). To this end, the module for returning the inhomogeneously distributed cells along each dimension of the box is used. Within this module the precise "load" in terms of number of cells are allocated to each processor. For achieving the cells partitioning, the module requires the number of processors, $R_{\mathrm{SH}}^{\mathrm{res}}$ and the shape of the heterogeneous system per dimension. In some heterogeneous simulation setups, the same amount of processors as amount of cells could be given. Thereby the algorithm verifies such setups and resolves if the distribution of cells per processors could be treated homogeneously or not. In case of homogeneity, the strict linked-cell-list partitioning is employed.

The cells redistribution is performed by finding the integer quotient of the number of processors divided by the number of cells, i.e., $q_{\mathrm{HR}} = \frac{C_{\mathrm{HR}}}{P_{\mathrm{HR}}}$ and $q_{\mathrm{LR}} = \frac{C_{\mathrm{LR}}}{P_{\mathrm{LR}}}$. On the contrary to the processor allocation module from Sec. III B 1, the cells partitioning makes emphasis on the low-resolution region. In other words, for the less expensive region (low-resolution) the quotient of cells per processor $q_{\mathrm{LR}} > q_{\mathrm{HR}}$.

Mathematically the cells in the low-resolution are weighted as a function of the resolution ratio $R_{\mathrm{SH}}^{\mathrm{res}}$ and the volumes V_{LR} and V_{HR}. Subject to the numbers of processors and cells there are cases where no integer quotient is found so that HeSpaDDA rounds down the real quotient solution and redistribute the "residual cells." The latter type of cells are distributed using a "pseudo-random" mechanisms as detailed in the Appendix A.

With the goal of making the algorithm available for different heterogeneous simulation techniques and MD simulation packages, the algorithm is implemented in ESPResSo++ and is also available as a stand-alone Python script [51].

IV. SIMULATIONS

The exhibited challenges in terms of parallelization schemes for heterogeneous molecular simulations have been thoroughly studied by using two archetypical systems, namely, an ubiquitin protein solvated in water and the binary Lennard-Jones fluid. Other technical details like the MD packages versions and underneath hardware platform used are found in Appendixes B and C.

A. Ubiquitin in aqueous solution

The system is illustrated in Fig. 1 and contained one protein molecule solvated in 38 084 water molecules. The simulation box was a cube of length ≈10.5 nm. In terms of the cell grid, a cubic decomposition of the system turns into a number of cells per simulation box side of 8 and hence cell grid ($8 \times 8 \times 8$), where each cell has a side length of 1.3 nm ($r_c + r_s$). The atomistic region was a sphere of radius 2 nm centered on a protein atom, such that the protein fits the sphere and thus was completely atomistic at all times. The width of the hybrid region was 1.0 nm. The benchmarking simulations presented here were performed using the simulation parameters given in a previous work [30]. The coarse-grained solvent model was obtained via iterative Boltzmann inversion (IBI) [52–54]. Nonbonded atomistic and coarse-grained interactions were interpolated using the force-AdResS approach. The nonbonded cutoff was 1.0 nm, and a thermodynamic force was used to counteract the pressure difference between atomistic and coarse-grained forcefields.

B. Phase separated Lennard-Jones binary fluid

The second simulation setup for benchmarking was a single-scale Lennard-Jones system of a phase separated 6-to-1 σ binary fluid. All nonbonded interactions were cut off at 0.9 cutoff, and interaction potentials shifted such that the energies are zero at the cutoff. The simulations were performed in the canonical ensemble at a temperature of 300 K, for which we used a Langevin thermostat with friction constant $\gamma = 0.5\,\mathrm{ps}^{-1}$. The total number of particles in the system is 145 792, with 124 964 particles located in the high-resolution and 20 828 in the low-resolution regions. The box dimensions were $32.0 \times 16.0 \times 16.0$ with periodic boundary conditions in each direction. Both halves of the box were equilibrated separately and then together. This way we are starting out with the maximal load imbalance.

V. RESULTS

A. Ubiquitin in aqueous solution

The first system is composed of roughly 115k atoms from which ≈88% are coarse-grained particles built out of water molecules. In Fig. 8, the strong scaling results are shown for both algorithms sDD and HeSpaDDA in terms of performance hours/ns. The latter is given by the simulation time of each value of $P = [16,32,64,96,128,160]$ required to perform a simulation for 1 ns. In all cases for the strong scaling validation shown in Fig. 8, the HeSpaDDA algorithm is much faster than sDD up to a speedup factor of 1.5. From the scaling viewpoint, HeSpaDDA reaches the best performance with 64 processors while sDD does with 96 (inset in Fig. 8).

We observe that the optimal number of processors for speeding up the first system goes along with both the shape

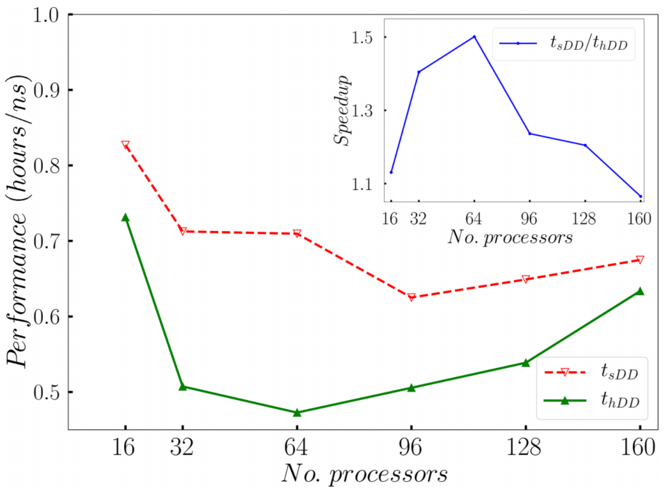

FIG. 8. Force-AdResS simulation of an atomistic protein and its atomistic hydration layers. The comparison between the simulation time of the spatial DD (dashed line in red) and HeSpaDDA (line in green); normalized by performance *hours/ns*. The inset shows the ratio between the spatial DD and HeSpaDDA, which corresponds to the speedup reached by the new algorithm.

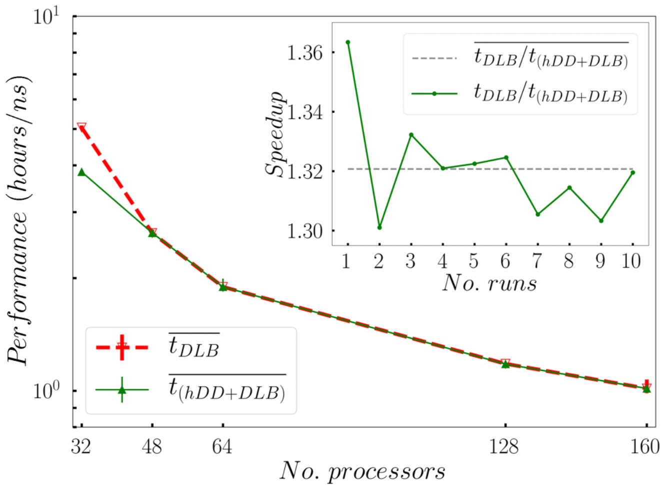

FIG. 9. Phase-separated Lennard-Jones binary fluid. The comparison between the simulation time of GROMACS-2016 load balancing (dashed and thick line in red) and HeSpaDDA combined with GROMACS-2016 load balancing (line in green); normalized by performance hours/ns. The inset shows the ratio between the GROMACS-2016 load balancing and HeSpaDDA combined with GROMACS-2016 load balancing, which corresponds to the speedup reached by the new algorithm.

of the box and the distribution of particles in the processors per resolution region. In other words, if we tackle a cubic simulation domain the best processors-grid distribution will be a perfect cube, which in the HeSpaDDA case justifies the value of 64 ($4 \times 4 \times 4$). On the other hand, sDD reaches the minimum simulation time with 96 processors, which does not fulfill any system cubicity constraints nor the low- and high-resolution regions demarcation. After reaching the minimum for both algorithms the speedup factor among them decreases fast, and reaches $\approx$1.08 for 160 processors (referring to the inset of Fig. 8). The reasons for the relative speedup reduction employing for example 160 processors are mainly three: (1) this number of processors cannot be decomposed as a perfect cube which increases the subdomains communication overhead; (2) the amount of particles per processor is decreased to roughly 230, which also increases the parallelization bookkeeping; and (3) during the simulation runtime imbalances are generated due to the fact that the ubiquitin solvated in water example does not provide any dynamic load-balancing algorithm. We also aimed to represent a system without dynamic load balancing and compare it to another one that includes dynamic load balancing (see Sec. V B).

As suggested in the modeling part of Sec. III A, the scaling behavior of cubic setups with larger low-resolution regions and using the HeSpaDDA algorithm is expected to be much faster compared to the sDD one. However, the current system reflects a worst-case scenario from the domain decomposition viewpoint, because the number of cells per box lengths (l_x, l_y, l_z) is eight with five cells under influence of the atomistic and hybrid regions and only three remaining cells are left for cells partitioning in the low-resolution region. The values of number of cells are calculated by directly dividing the box lengths by $(r_c + r_s)$.

B. Phase-separated Lennard-Jones binary fluid

The second simulation setup for benchmarking was a single-scale Lennard-Jones system of a phase-separated binary fluid. The heterogeneity ratio $R^{\mathrm{res}}_{\mathrm{SH}}$ is 6:1 based on σ. Such system comprises $\approx$145 k particles, where 50% are low resolution built out as the ratio $R^{\mathrm{res}}_{\mathrm{SH}}$.

In Fig. 9, we show a direct comparison of the strong scaling simulations performed for both algorithms HeSpaDDA and GROMACS-2016 [55] domain decomposition with notation "*DLB*." Interestingly, for this system we combined HeSpaDDA to the DLB algorithm included in GROMACS-2016 with notation "*hDD+DLB*" (Fig. 9). The results are again shown in terms of performance hours/ns and the strong scaling range is given as $P = [32,48,64,128,160]$. These results indicate that for $P = [48,64,128,160]$ both algorithms show the same performance within the error bars given in Fig. 9.

Similar results have been achieved because the initial processors triplets (P_x, P_y, P_z) given by both algorithms at the beginning of the simulation gave exactly the same values. In other words, the percolation of triplets to find the total number of processors for $P = [48,64,128,160]$ is limited in those cases given the dimensions of the noncubic box. However, for $P = 32$ the triplets given by the combination of HeSpaDDA and GROMACS-2016 dynamic load balancing shows an speedup in the performance of a factor $\approx$1.32 (see the inset in Fig. 9). Curiously the use of HeSpaDDA does not include any overhead cost, nor implementation and it works as mentioned before in a predictive manner. It is important to remark that no changes to GROMACS-2016 have been performed but changing the initial processors triplets form the command line. Hence only the processor allocation module of HeSpaDDA has been used and the tuning of cells distribution relies on the one given by GROMACS-2016.

Another interesting aspect observed in Fig. 9 is the strong scaling of the system as a function of P which is clearly superior than the previously tackled system (see Sec. V A). A direct explanation for such observation comes from the fact that in the phase separated Lennard-Jones binary fluid DLB features are enabled while the previous system is configured purely with static cells and processor allocations.

VI. DISCUSSION AND CONCLUSION

While traditional molecular simulations are mostly performed with homogeneous resolution setups for all molecules in a simulation box, in heterogeneous schemes, the simulation box is typically divided in at least two regions, namely, the low-resolution and high-resolution ones. We have developed an algorithm to achieve such domain decomposition requirements by adding resolution-sensitivity to the spatial domain decomposition and combining it with an initial rearrangement of the subdomain walls in terms of cells per processor. We come closer to the particular requirements of adaptive resolution schemes in terms of scalability and relative speedups for archetypical examples an AdResS biomolecule in solution and a binary Lennard-Jones fluid. Remarkably, the tackled AdResS setup represents an archetypical system from the domain decomposition viewpoint where the low-resolution region has parsimonious dimensions. However HeSpaDDA is faster by a factor up to 1.5 than the spatial DD algorithm. A modeling framework has been also provided for HeSpaDDA in Sec. III A. We recommend to use such predictive modeling in an *a priori* way before starting to simulate any heterogeneous system like the ones described here.

Our results also show that for the binary Lennard-Jones fluid, HeSpaDDA combined to an efficient DLB algorithm can reach a speedup factor of ≈1.32.

HeSpaDDA algorithm is conceived to assign a parsimonious amount of processors for the whole simulation production run. This makes an heterogeneous system as scalable as the condition $\frac{N_{\mathrm{HR}}}{P_{\mathrm{HR}}} = \frac{N_{\mathrm{LR}}}{P_{\mathrm{LR}}}$ will allow. Furthermore, we observed that HeSpaDDA speeds up the initial run of any heterogeneous simulations. In other words, multiscale simulations are faster from their beginning and ready to be tackled with further dynamic load-balancing alternatives for the production run where the underlying hardware of the HPC environment may introduce runtime imbalance, as mentioned in Sec. III and shown in Sec. V.

In addition to the benchmark results, a theoretical modeling for the algorithm and the scaling law of computation time are provided. These scaling laws are favorable for exploring the upper boundaries in terms of scalability as a function of the system size or multiscale resolution ratio. Consequently mathematical details of the HeSpaDDA are presented, as well as, the algorithm flow charts and implementation. With the final goal that HeSpaDDA algorithm could be used in other multiscale techniques and/or other MD packages.

The envisioned applications for the HeSpaDDA algorithm on multiple resolution methods aim to increase scalability, and hence make larger biomolecular and advanced materials simulations more feasible.

Besides the direct extensions of the present work to new systems of multiple spatial resolution and or single-scale inhomogeneous binary mixtures, the algorithm developed here and the referenced systems paves the way to studying more complex systems, such as evaporation, crystallization and active matter processes whenever they accomplish the slowly varying system assumption.

On top of the aspects described above, the HeSpaDDA method does not introduce any time overhead to the simulation setup nor the simulation run since it is a highly optimized algebraic function. This has been shown in Sec. V B, where HeSpaDDA is complementary to a dynamic load-balancing technique at no additional time overhead.

ACKNOWLEDGMENTS

T.S. and H.V.G. acknowledge financial support by the Project No. TRR 146 ‘Multiscale Simulation Methods for Soft Matter Systems’ of the Deutsche Forschungsgemeinschaft. The authors are thankful to Jan Smrek, Karsten Kreis, and Aoife Fogarty for stimulating and fruitful discussions on the manuscript.

APPENDIX A: HESPADDA ALGORITHM DESCRIPTION

1. Processor allocation

We start by describing the HeSpaDDA algorithm flow chart for the processor allocation as illustrated in Fig. 10. In terms of processors, HeSpaDDA is allocating higher priority to the high-resolution regions according to equation

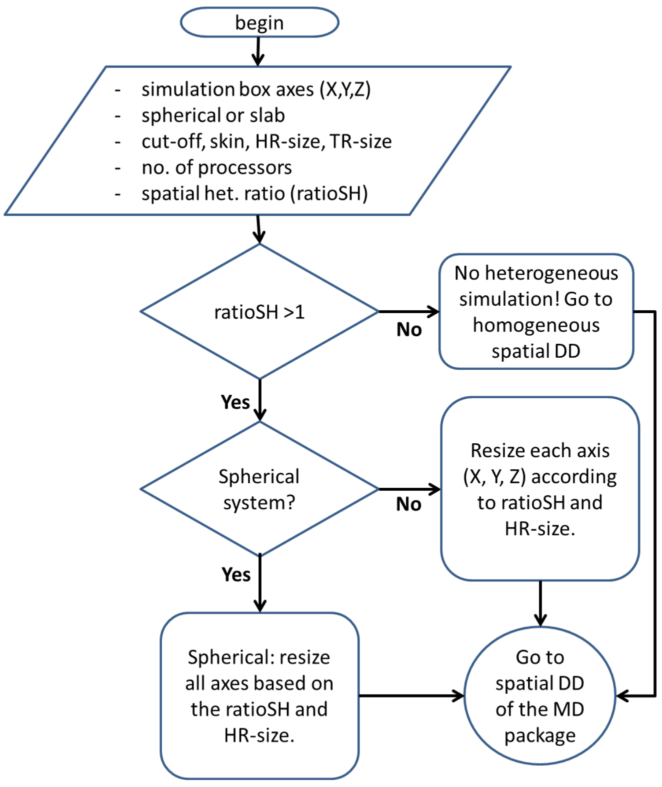
begin
- simulation box axes (X,Y,Z)
- spherical or slab
- cut-off, skin, HR-size, TR-size
- no. of processors
- spatial het. ratio (ratioSH)
ratioSH >1
No
No heterogeneous simulation! Go to homogeneous spatial DD
Yes
Spherical system?
No
Resize each axis (X, Y, Z) according to ratioSH and HR-size.
Yes
Spherical: resize all axes based on the ratioSH and HR-size.
Go to spatial DD of the MD package

FIG. 10. Flow chart of the first module within HeSpaDDA algorithm devoted to the allocation of processors along the heterogeneous simulation setup. The present module provides the processors grid in terms of P_x, P_y, P_z.

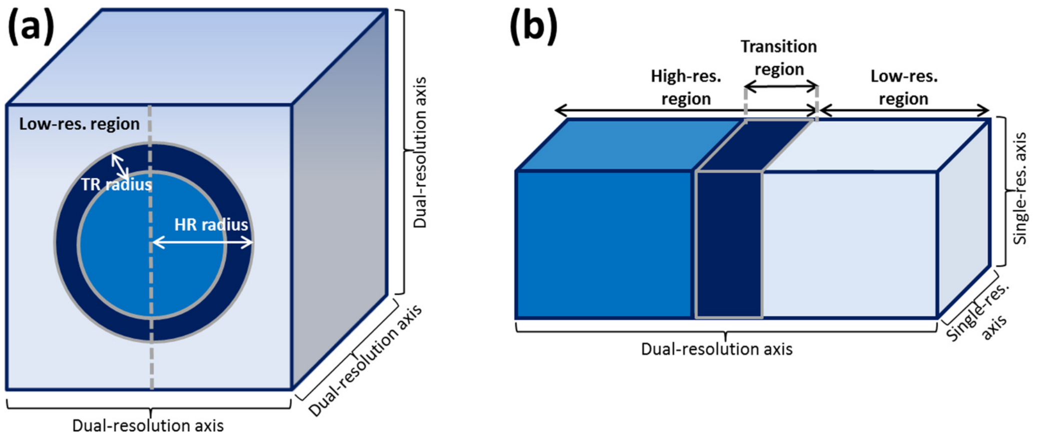

FIG. 11. Description of the geometrical characteristics of the heterogeneous molecular systems like (a) cubic and spherical; (b) noncubic and slablike. The illustration also denotes which axis in a certain type of geometrical configuration will be considered as single or dual resolution.

$P_{\mathrm{HR}}^{\mathrm{hDD}} = \frac{R_{\mathrm{SH}}^{\mathrm{res}} V_{\mathrm{HR}}}{V+V_{\mathrm{HR}}(R_{\mathrm{SH}}^{\mathrm{res}}-1)} P$ as described in the modeling section of this article. Consequently, more processors-per-cells will be available in the box region where the high-resolution is located than in the low-resolution one. In addition, the overall allocation of processors per simulation box axis is thoroughly controlled by verifying that the total number of processors in each simulation box axis does not exceed the number of cells per box axis (i.e., X,Y,Z).

For slablike high-resolution configurations there is a supplementary control flow which based on the $R_{\mathrm{SH}}^{\mathrm{res}}$ and the number of processors per axis (i.e., P_x, P_y, P_z) distinguishes where resources should be assigned according to a single (homogeneous) or dual resolutions (heterogeneous) as depicted in Fig. 11. Another example of a single-resolution spatial DD within HeSpaDDA is having a small number of processors per axis. For example, let's consider $P = 8$, whereas for a cubic configuration we have $P_x = P_y = P_z = 2$, and hence the processors allocation cannot be efficient under a heterogeneous DD scheme.

2. Cell partitioning

Once the three-dimensional processors grid has been built, cell partitioning is required (this flow chart is illustrated in Fig. 12). To this end, the module for returning the inhomogeneously distributed cells (or cells neighbor-list) along each axis of the box is called. Within this module the precise load in terms of number of cells are allocated to each processor. For achieving the cell partitioning, the module requires the number of processors, $R_{\mathrm{SH}}^{\mathrm{res}}$ and the shape of the heterogeneous system per box axis. In some heterogeneous simulation setups, the same amount of processors as amount of cells could be given. Thereby, the algorithm verifies such setups and resolves if the distribution of cells per processors could be treated homogeneously or not. In case of homogeneity, the strict linked-cell-list partitioning will be applied.

For heterogeneous systems where the total number of cells and processors differ, a new cells distribution scheme is presented. Such a scheme assumes beforehand that the total number of cells in the system can be symmetrically divided by two for each box axis, i.e., X, Y, Z. Hence, a first function is called to start the half-size decomposition (in the flowchart of Fig. 12 found as *halfDecomp*).

The advantage of the half-symmetric decomposition appears when the symmetry condition is fulfilled and thus the "half-decomposed system" can be mirrored. Consequently, a rapid whole domain decomposition will be achieved. However, HeSpaDDA is also able to tackle asymmetric cases by means of the subsequent cells redistribution (which can be found in Fig. 12 as *addHsymmetry*). The cells redistribution is performed by finding the integer quotient of the number of cells divided by the number of processors, i.e., $C_{\mathrm{HR}}/P_{\mathrm{HR}}$ and $C_{\mathrm{LR}}/P_{\mathrm{LR}}$. Contrary to the processor allocation module from Sec. III B 1, the cells partitioning assigns more weight in terms of cells to the low-resolution region. In other words, for the less expensive region (low-resolution) the ratio of cells per processor $C_{\mathrm{LR}}/P_{\mathrm{LR}}$ is mostly higher that $C_{\mathrm{HR}}/P_{\mathrm{HR}}$.

Mathematically the cells in the low-resolution region are weighted as a function of the resolution ratio $R_{\mathrm{SH}}^{\mathrm{res}}$ (*RatioSH*) and the volumes V_{LR} and V_{HR}. Subject to the numbers of processors and cells there are cases where no integer quotient is found so that HeSpaDDA rounds down the real quotient solution and redistribute the "residual cells." The latter type of cells are distributed using a "pseudorandom" mechanism. This mechanism controls the "residual cells" distribution by assigning flags to the processors that already contain one "residual cell," so that they will not repeatedly be assigned to the same processor. In the code [51] such functions are embedded in the *addHsymmetry* function (see Fig. 12). As a final step the algorithm will adapt the current cells partitioning to the neighbor-list data structure employed in a given simulation package. For example, a code based on the linked-cell-list algorithm.

The algorithm is also sensitive to ill-conditioned heterogeneous setup and therefore messages will be displayed as a warning of missing scalability or performance.

In Fig. 11(a), the center of the simulation box correspond to the center of the high-resolution region. However, if the initial simulation setup is not placing the high-resolution region in the center of the simulation box, HeSpaDDA calculates the

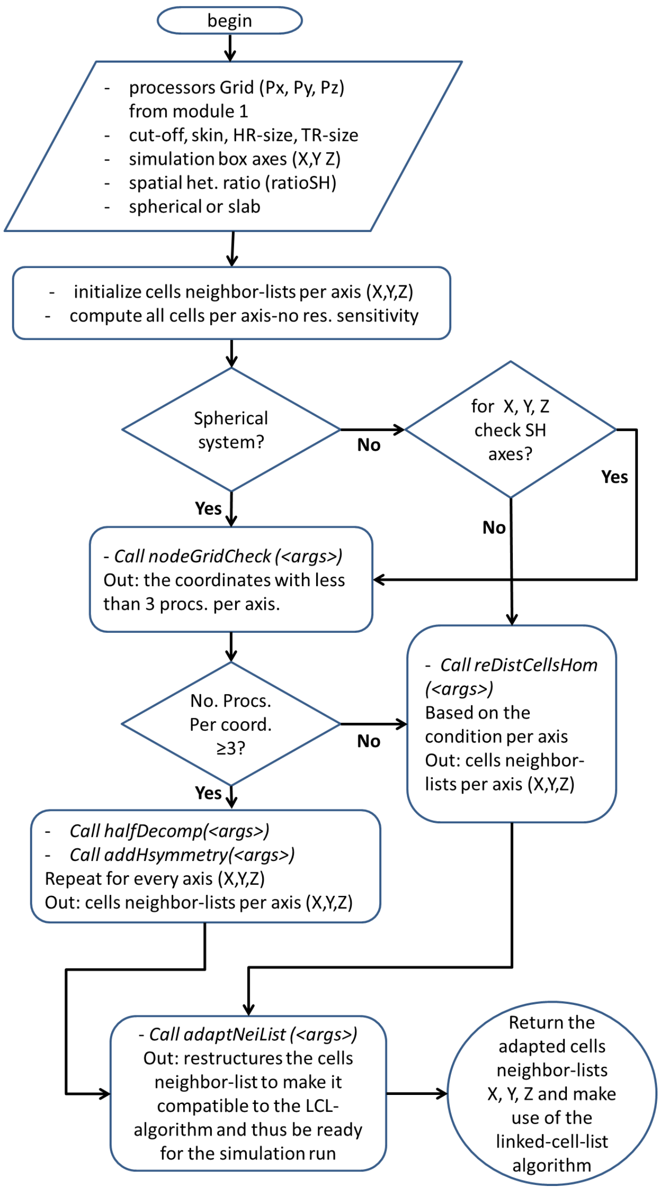
begin
- processors Grid (Px, Py, Pz) from module 1
- cut-off, skin, HR-size, TR-size
- simulation box axes (X,Y Z)
- spatial het. ratio (ratioSH)
- spherical or slab
- initialize cells neighbor-lists per axis (X,Y,Z)
- compute all cells per axis-no res. sensitivity
Spherical system?
No
for X, Y, Z check SH axes?
Yes
Yes
No
- Call nodeGridCheck (<args>) Out: the coordinates with less than 3 procs. per axis.
- Call reDistCellsHom (<args>) Based on the condition per axis Out: cells neighbor-lists per axis (X,Y,Z)
No. Procs. Per coord. ≥3?
No
Yes
- Call halfDecomp(<args>)
- Call addHsymmetry(<args>)
Repeat for every axis (X,Y,Z)
Out: cells neighbor-lists per axis (X,Y,Z)
- Call adaptNeiList (<args>) Out: restructures the cells neighbor-list to make it compatible to the LCL-algorithm and thus be ready for the simulation run
Return the adapted cells neighbor-lists X, Y, Z and make use of the linked-cell-list algorithm

FIG. 12. Flow chart of the second module within HeSpaDDA algorithm devoted to distribution of cells along coordinates in each subdomain, according to the high- and low-resolution regions of the heterogeneous simulation setup.

offset between those centers by considering periodic boundary conditions and hence the cells partitioning will be shifted by the nearest integer quotient offset/r_{cell}.

With the goal of making the algorithm available for different heterogeneous simulation techniques and corresponding MD simulation packages, the algorithm is implemented in ESPResSo++ and is available as an stand-alone Python script.

APPENDIX B: SIMULATION SETUP

In the presented work, two systems have been tackled: the multiscale protein solvated in water and the Lennard-Jones binary fluid.

1. Ubiquitin in aqueous solution

The simulations have been carried out by building the codes with Intel MPI compiler v14.0, Infiniband-Cluster, and two versions of ESPResSo++: the first (v1.9.4) uses an spatial domain decomposition, while the second is publicly available within this paper also on ESPResSo++ github. Each simulation was run for ten thousand steps and with a variable number of processors ($P = [16,32,64,96,128,160]$). The skin was optimized for each system and for $P = 16$ (using the *TuneSkin* function of ESPResSo++ v1.9.4). dt of the protein in solution was 1 fs. Only the time spent integrating the equations of motion was taken into account, i.e., no file I/O was considered. Note also that the values given are the average times of a total of five runs for each data point in the article (see Sec. V).

2. Phase separated Lennard-Jones binary fluid

For the second system, the simulations have been carried out by building the codes with Intel MPI compiler v14.0, Infiniband-Cluster, and one version of GROMACS-2016, which has an embedded dynamic load-balancing algorithm that has been employed. Note that for the simulations with GROMACS, there are not two versions because such package allows to chose the number of processors per simulation box axis in advance. Each simulation was run for ten thousand steps with a variable number of processors ($P = [32,64,128,160]$) and dt of the 1 fs. Only the time spent integrating the equations of motion was taken into account, i.e., no file I/O was considered. Note also that the values given are the average times of a total of 10 runs for each data point in the article (see Sec. V). We have added more runs in this second case since we tackle a simulation with the dynamic load-balancing feature enabled.

APPENDIX C: IMPLEMENTATION

The validation and benchmarks have been carried out after a new implementation on the current release (v1.9.4) of the ESPResSo++ package. This release offers an homogeneous-spatial domain decomposition scheme combined with the linked-cell-list algorithm [6]. The HeSpaDDA algorithm needed the following backward compatible modifications to ESPResSo++ (v1.9.4): new data structure for the subdomains cells neighbor-lists *neiListX*, *neiListY* and *neiListZ*, an iterative function that allocates processors and one function that distributes cells according to the given resolution types per box axis X, Y, Z and *ratioSH*. We have provided a brief overview of those modifications, while further code details can be found in the ESPResSo++ github repository: https://github.com/espressopp/espressopp.

[1] K. Kremer and F. Müller-Plathe, Mol. Simul. **28**, 729 (2002).
[2] N. Attig, K. Binder, H. Grubmüller, and K. Kremer, eds., *Computational Soft Matter: From Synthetic Polymers to Proteins, NIC Lecture Notes*, Vol. 23 (Forschungszentrum, Jülich, 2004).

[3] G. A. Voth, ed., *Coarse-Graining of Condensed Phase and Biomolecular Systems* (CRC Press, Boca Raton, FL, 2008).
[4] C. Holm and K. Kremer, eds., *Advanced Computer Simulation Approaches for Soft Matter Science I-III, Series: Advances in Polymer Science*, Vol. 173, 185, 221 (Springer, Berlin, 2005–2009).
[5] C. Peter and K. Kremer, Faraday Discuss. **144**, 9 (2010).
[6] J. D. Halverson, T. Brandes, O. Lenz, A. Arnold, S. Bevc, V. Starchenko, K. Kremer, T. Stuehn, and D. Reith, Comput. Phys. Commun. **184**, 1129 (2013).
[7] A. J. Liu, G. S. Grest, M. C. Marchetti, G. M. Grason, M. O. Robbins, G. H. Fredrickson, M. Rubinstein, and M. Olvera de la Cruz, Soft Matter **11**, 2326 (2015).
[8] D. Rapaport, Comput. Phys. Commun. **62**, 198 (1991).
[9] D. E. Shaw, R. O. Dror, J. K. Salmon, J. P. Grossman, K. M. Mackenzie, J. A. Bank, C. Young, M. M. Deneroff, B. Batson, K. J. Bowers, E. Chow, M. P. Eastwood, D. J. Ierardi, J. L. Klepeis, J. S. Kuskin, R. H. Larson, K. Lindorff-Larsen, P. Maragakis, M. A. Moraes, S. Piana, Y. Shan, and B. Towles, in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (ACM, New York, 2009), p. 1.
[10] Y. Li, B. C. Abberton, M. Kroeger, and W. K. Liu, Polymers **5**, 751 (2013).
[11] D. C. Rapaport, J. Phys.: Condens. Matter **26**, 503104 (2014).
[12] S. Piana, J. L. Klepeis, and D. E. Shaw, Curr. Opin. Struct. Biol. **24**, 98 (2014), folding and binding/Nucleic acids and their protein complexes.
[13] M. J. Field, Archives of Biochemistry and Biophysics **582**, 3 (2015), special issue in computational modeling on biological systems.
[14] A. Gooneie, S. Schuschnigg, and C. Holzer, Polymers **9**, 16 (2017).
[15] M. Praprotnik, L. D. Site, and K. Kremer, Annu. Rev. Phys. Chem. **59**, 545 (2008).
[16] R. Delgado-Buscalioni, K. Kremer, and M. Praprotnik, J. Chem. Phys. **128**, 114110 (2008).
[17] R. Delgado-Buscalioni, K. Kremer, and M. Praprotnik, J. Chem. Phys. **131**, 244107 (2009).
[18] R. Potestio, S. Fritsch, P. Español, R. Delgado-Buscalioni, K. Kremer, R. Everaers, and D. Donadio, Phys. Rev. Lett. **110**, 108301 (2013).
[19] J. Smrek and K. Kremer, Phys. Rev. Lett. **118**, 098002 (2017).
[20] M. Radu and K. Kremer, Phys. Rev. Lett. **118**, 055702 (2017).
[21] T. Bereau and M. Deserno, J. Membr. Biol. **248**, 395 (2015).
[22] K. Pluhackova and R. A. Böckmann, J. Phys.: Condens. Matter **27**, 323103 (2015).
[23] S. Plimpton, J. Comp. Phys. **117**, 1 (1995).
[24] D. Shaw, J. Comput. Chem. **26**, 1318 (2005).
[25] M. Praprotnik, L. D. Site, and K. Kremer, J. Chem. Phys. **123**, 224106 (2005).
[26] M. Praprotnik, S. Matysiak, L. D. Site, K. Kremer, and C. Clementi, J. Phys.: Condens. Matter **19**, 292201 (2007).
[27] M. Praprotnik, S. Matysiak, L. D. Site, K. Kremer, and C. Clementi, J. Phys.: Condens. Matter **21**, 499801 (2009).
[28] R. Potestio, P. Español, R. Delgado-Buscalioni, R. Everaers, K. Kremer, and D. Donadio, Phys. Rev. Lett. **111**, 060601 (2013).
[29] K. Kreis, D. Donadio, K. Kremer, and R. Potestio, EPL **108**, 30007 (2014).
[30] A. C. Fogarty, R. Potestio, and K. Kremer, J. Chem. Phys. **142**, 195101 (2015).
[31] K. Kreis, A. C. Fogarty, K. Kremer, and R. Potestio, Eur. Phys. J Special Topics **224**, 2289 (2015).
[32] J. Zavadlav, R. Podgornik, and M. Praprotnik, J. Chem. Theory Comput. **11**, 5035 (2015).
[33] A. B. Poma and L. D. Site, Phys. Rev. Lett. **104**, 250201 (2010).
[34] A. B. Poma and L. D. Site, Phys. Chem. Chem. Phys. **13**, 10510 (2011).
[35] K. Kreis, M. E. Tuckerman, D. Donadio, K. Kremer, and R. Potestio, J. Chem. Theory Comput. **12**, 3030 (2016).
[36] J. Zavadlav, M. N. Melo, S. J. Marrink, and M. Praprotnik, J. Chem. Phys. **140**, 054114 (2014).
[37] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, J. Chem. Theory Comput. **4**, 435 (2008).
[38] L. Flynn and S. Hummel, IBM Research Report **RC18462** (1992).
[39] S. F. Hummel, E. Schonberg, and L. E. Flynn, in *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing*, Supercomputing '91 (ACM, New York, NY, 1991), pp. 610–632.
[40] C. D. Polychronopoulos and D. J. Kuck, IEEE Trans. Comput. **C-36**, 1425 (1987).
[41] *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94.
[42] C. P. Kruskal and A. Weiss, IEEE Trans. Softw. Eng. **SE-11**, 1001 (1985).
[43] L. Rudolph, M. Slivkin-Allalouf, and E. Upfal, in *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '91 (ACM, New York, NY, 1991), pp. 237–245.
[44] L. V. Kale and G. Zheng, in *Advanced Computational Infrastructures for Parallel and Distributed Applications*, edited by M. Parashar (Wiley-Interscience, New York, 2009), pp. 265–282.
[45] H. Kaiser, T. Heller, B. Adelstein-Lelbach, A. Serio, and D. Fey, in *Proceedings of the 8th International Conference on Partitioned Global Address Space Programming Models*, PGAS'14 (ACM, New York, NY, 2014), pp. 6:1–6:11.
[46] S. J. Chapin, D. Katramatos, J. Karpovich, and A. S. Grimshaw, The Legion Resource Management System, in *Job Scheduling Strategies for Parallel Processing: IPPS/SPDP'99Workshop, JSSPP'99 San Juan, Puerto Rico, April 16, 1999 Proceedings*, edited by D. G. Feitelson and L. Rudolph (Springer, Berlin/Heidelberg, 1999), pp. 162–178.
[47] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Oxford University Press, New York, 1987).
[48] A. Osprey, G. D. Riley, M. Manjunathaiah, and B. N. Lawrence, in *Proceedings of HPCS* (IEEE, 2014), pp. 715–723.
[49] T. Malik, V. Rychkov, and A. L. Lastovetsky, Concurr. Comput.: Pract. Exp. **28**, 802 (2016).
[50] S. Markidis, J. Vencels, I. B. Peng, D. Akhmetova, E. Laure, and P. Henri, Phys. Rev. E **91**, 013306 (2015).
[51] https://github.com/govarguz/HeSpaDDA.git.
[52] A. Soper, Chem. Phys. **202**, 295 (1996).

[53] W. Tschöp, K. Kremer, J. Batoulis, T. Bürger, and O. Hahn, Acta Polymerica **49**, 61 (1998).

[54] D. Reith, M. Pütz, and F. Müller-Plathe, J. Comput. Chem. **24**, 1624 (2003).

[55] S. Páll, M. J. Abraham, C. Kutzner, B. Hess, and E. Lindahl, Tackling Exascale Software Challenges in Molecular Dynamics Simulations with GROMACS, in *Solving Software Challenges for Exascale: International Conference on Exascale Applications and Software, EASC 2014, Stockholm, Sweden, April 2-3, 2014, Revised Selected Papers*, edited by S. Markidis and E. Laure (Springer International Publishing, Cham, 2015), pp. 3–27.