Noname manuscript No.
(will be inserted by the editor)

Load-Balancing and Spatial Adaptivity for Coarse-Grained Molecular Dynamics Applications

Steffen Hirschmann · Michael Lahnert · Carolin Schober · Malte Brunn · Miriam Mehl · Dirk Pflüger

Received: date / Accepted: date

Manuscript. The final publication is available at Springer through:
https://dx.doi.org/10.1007/978-3-030-13325-2:
Hirschmann, S.; Lahnert, M.; Schober, C.; Brunn, M.; Mehl, M. and Pflüger, D.: Load-Balancing and Spatial Adaptivity for Coarse-Grained Molecular Dynamics Applications. In: High Performance Computing in Science and Engineering '18. Springer, 2018.

Abstract We present our approach for a scalable implementation of coupled soft matter simulations for inhomogeneous applications based on the simulation package ESPResSo and an extended version of the adaptive grid framework `p4est`. Our main contribution in this paper is the development and implementation of a joint partitioning of two or more distinct octree-based grids based on the concept of a *finest common tree*. This concept guarantees that, on all grids, the same process is responsible for each point in space and, thus, avoids communication of data in overlapping volumes handled in different partitions. We achieve up to 85 % parallel efficiency in a weak scaling setting. Our proposed algorithms take only a small fraction of the overall runtime of grid adaption.

1 Introduction

Many scientifically relevant systems are composed of several different subsystems, each representing a different physical mechanism that is governed by different sets of equations. In case of particles subjected to a background flow, we need to model the fluid, the interaction of the particles, and the interaction between particles and fluid. Interactions between particles may include

S. Hirschmann
University of Stuttgart, Institute for Parallel and Distributed Systems, Universitätsstr. 38, 70569 Stuttgart, Germany
Tel.: +49-(0)711-685-88223
Fax: +49-(0)711-685-78413
E-mail: steffen.hirschmann@ipvs.uni-stuttgart.de

long-range electrostatic interactions, bonding, and electrokinetic influences by an ionic fluid. To model the complete system, each individual component interacts with all the other subsystems, thus, forming a challenging coupled problem across multiple time and length scales.

The underlying simulation package of our work is ESPResSo [18, 2], the extensible simulation package for research on soft matter systems. ESPResSo is a flexible multi-physics simulation software for soft matter simulations. They can have up to four different, distinct physics components that are all implemented and coupled with each other: Short-range MD, long-range MD, hydrodynamics, and electrokinetics.

A common approach to reduce the computational cost and, thus, runtime of grid-based simulations is to use tree-structured adaptive grids for the grid-based part of our simulations. In this context, space-filling curves (SFC) are a well-known way for grid traversal and linearization, see, e.g., [6, 28]. Partitioning a simulation domain with a space-filling curve is usually done by leveraging the linearization defined by the SFC and performing a so-called chain-on-chain partitioning [22], i.e., basically cutting the pieces of the linearized list of grid cells into equal pieces. This approach is found to be efficient in terms of runtime and memory consumption and of good quality compared to other methods [20, 4]. A ready-to-use implementation of parallel algorithms for adaptive mesh refinement and dynamic load-balancing along a space-filling curve is provided by the `p4est` framework [5, 14]. It provides scalable algorithms for grids composed of several octrees (forest-of-octrees) based on Morton ordering [21] (Z-curve).

We provide dynamic spatial adaptivity for hydrodynamics using the lattice-Boltzmann method as well as a way to dynamically load-balance MD simulations in ESPResSo. This is based on grids implemented in an extended version of the `p4est` library [17, 16]. As the basis for the current work, we implemented each of the four subsystems to use a `p4est`-based grid: Load-balanced short-range MD [11] as well as spatially adaptive and load-balanced long-range MD [27], the lattice-Boltzmann method (LBM) [17, 16], and electrokinetics [27]. Coupling a grid-based system and a particle system with spatially adaptive grids on the one hand and dynamic load-balancing on the other hand is a challenging problem. If the respective partitions of both subsystems are not congruent, very high communication cost is generated due to the need to exchange data not only at partition surfaces but in overlapping volumes. Therefore, we propose a method for jointly partitioning two octree grids.

Our algorithms support an arbitrary number of `p4est` instances (subsystems). The only requirement is that they must share the same "macroscopic" setup, i.e., the number and mutual arrangement of trees must be the same for all instances. In this paper, we focus on the coupled simulation of short-range molecular dynamics (non-charged particles) with a background flow calculated with the lattice-Boltzmann method. These systems serve as a template for future extensions.

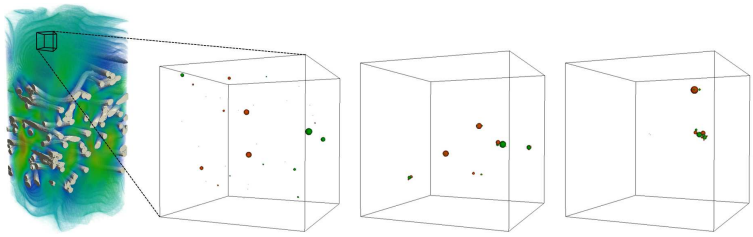


Fig. 1: Simulation setup and agglomeration formation over time.

2 Cabin Air Filter Simulation: an Application Example

One of our applications of interest using a fully coupled system as described above is the simulation of particulate flow through cabin air filters. The aim is to model the separation behavior of highly electrostatically charged dust particles at the surface of complex filter media structures. The interaction of dust particles due to electrostatics and other inter-particle forces has been neglected in simulation approaches so far. Laboratory experiments, however, suggest these interactions are likely to play a decisive role in the particle collection process of the fibrous structures. In order to investigate the agglomeration rate of particles, we can focus on the inflow area of the filter, as illustrated in Fig. 1. Since the greatest influence of particle-particle interactions is assumed to be in this area, the development of particle size distributions is investigated there omitting the filter structure. Figure 1 shows the simulation setup as well as the formation of agglomerates for particle whose sizes and charges have been chosen at random. All necessary steps for the parallel simulation of this scenario with the default version of ESPResSo have been completed [25].

For a more detailed analysis, the particle charge as well as the size distribution have been adapted to more realistic values based on published data from literature [9, 10, 15]. From these data, several samples were drawn randomly for fixed domain sizes until the target concentration of 75 $\frac{\mathrm{mg}}{\mathrm{m}^3}$ (laboratory testing conditions) was reached.

We measured the runtime of 100 fluid time steps in a strong scaling setup. The MD time step is 1000 times smaller than the fluid time step. All simulations were performed with 16 processes per node in order to facilitate the equal distribution of LB cells to the subdomains. Note that a time step consists of updating the individual physical subsystems (short-range MD, long-range MD, and hydrodynamics every 1000^{th} time step) as well as a coupling step.

To achieve realistic simulations in a reasonable simulation time, certain parameters have been altered: 1. number of particles (stochastically drawn distributions), 2. domain sizes, 3. resolution for LB grid, 4. boundary conditions for LB flow field (constant initial flow field vs. fix inflow and outflow velocities). As shown in Fig. 2 and Fig. 3, there is an optimal number of nodes in terms of runtime for each simulation setup. The optimal number of nodes depends on the ratio of the respective LB and MD efforts.

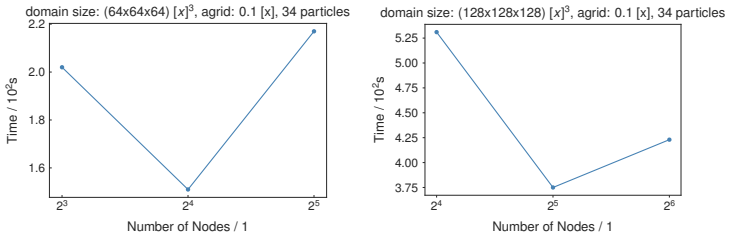


Fig. 2: Strong scaling measurement of 100 fluid time steps for different domain sizes. A single time step consists of updating the individual subsystems (short-range MD, long-range MD, and hydrodynamics every 1000$^{\text{th}}$ time step) as well as a coupling step.

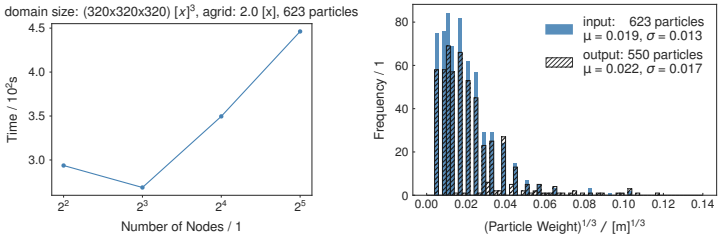


Fig. 3: Runtime of 100 fluid time steps for a larger domain size in a strong scaling scenario (left). Again, a single time step consists of updating the individual subsystems (short-range MD, long-range MD, and hydrodynamics every 1000$^{\text{th}}$ time step) as well as a coupling step. The particle weight distributions on the right indicate the importance of particle-particle interactions for different rates. The mean values μ and the respective standard deviations σ demonstrate the shift towards larger/heavier particles.

The resulting particle weight/size distribution, also plotted in Fig. 3, shows the importance of taking the particle-particle interactions into account and justifies the use of the cost intensive fully coupled system. In the case of larger particles, other collection mechanisms become effective. The increasingly effective inertial effects are presumably responsible for improved measurements in filtration efficiency. The latter statement, however, has to be verified with larger simulation setups as well as the additional simulation of the particulate air flow through the filter medium. In order to keep the computational effort for these extended scenarios feasible, we need efficient algorithms for partitioning and dynamic load-balancing as well as adaptive mesh refinement (to accurately resolve the regions around the filter structure and the particles).

3 Soft Matter Simulations using ESPResSo

To understand the challenges and the implementation of coupled load-balancing, we briefly sketch the different subsystems of ESPResSo and their current spatially adaptive and/or load-balanced implementation in this section.

3.1 Molecular Dynamics

Molecular Dynamics traces the trajectories of particles. Accordingly, these particles' positions and velocities represent the quantities of interest. Particles interact through inter- and intra-molecular force fields. The trajectories are given by solving Newton's translational and rotational equations of motion.

In this work, we focus on short-range intra-molecular interactions described by a pair-potential U. Intra-molecular forces are derived from U via

$$\mathbf{F}(\mathbf{x}_i) = -\nabla \sum_{j \neq i} U(\mathbf{x}_i, \mathbf{x}_j),$$

where $\mathbf{F}(\mathbf{x}_i)$ is the force acting on particle i at position $\mathbf{x}_i$. U is classified as short-range if and only if

$$\int_{\|\mathbf{x}\| > r_c} U(\mathbf{x}) d\mathbf{x} \xrightarrow{r_c \to \infty} 0.$$

In this case, linear complexity of the force calculations is achieved by cutting off U at a constant radius r_c, i.e., neglecting interactions with particles at distances larger than r_c. Given such a cut-off radius, the Linked-Cell method [13] can be used which discretizes the domain with a Cartesian grid with mesh width r_c. Particles are assigned to cells of this grid such that potential interaction partners of each particle can be detected with $\mathcal{O}(1)$ cost by searching in neighboring grid cells only.

In ESPResSo the Linked-Cell method is implemented via a so-called cell system [18]. In parallel simulations, this grid is subdivided evenly among the respective ranks. This subdivision is strictly Cartesian, i.e., partitions are rectangular sub-volumes. In [11] we introduce a new cell system in a minimally invasive way which is able to leverage arbitrary domain decompositions. This in particular requires new communication algorithms as there is no strict 'upper-lower-left-right-front-back' relation between neighboring partitions anymore, and as the number of neighbor partitions becomes variable. For short-range MD, we do not port the algorithm itself to `p4est` but rely on the partitioning and communication functionality of `p4est` to generate information to fill ESPResSo's internal data-structures. This implementation supports dynamic re-partitioning of the grid and the associated numerical payload (particle data). As we left the internal algorithms untouched, the grid of our implemented Linked-Cell method remains regular.

3.2 Hydrodynamics

The molecular ensemble simulated by ESPResSo can be subject to a background flow. This background flow is realized using a thermalized D3Q19 implementation of the lattice-Boltzmann method (LBM) as described in [8, 24, 7]. The LBM simulates fluid-flow based on probability densities of virtual particles with discrete velocities associated to the centers of grid cells. Each time step consists of two steps: 1. *collision*, a cell-local interaction of probability densities driving those densities towards their local equilibrium, and 2. *streaming*, propagating these effects to the respective neighbor cells according to the chosen stencil.

Extending the LBM to adaptive grids according to [23] requires that the time step is proportional to the local spatial resolution. Thus, grid adaptivity in space automatically induces local time step adaptivity, which is desirable from an application point of view as it reduces the computational costs of coarser grid regions further. The scheme transmits information between cells of different discretization levels by embedding virtual cells in coarse quadrants neighboring finer regions. Simply spoken, this allows to stream between fine cells and coarser neighbors through their virtual children, whereas streaming between coarse cells and their equally coarse neighbors happens at the coarse level. In this sense, streaming can be decomposed into level-wise steps, each of which only visits cells of the same refinement level. To transfer data from the finer virtual cells to their parent cells and vice versa, virtual cells are populated by evenly distributing the fluid mass among them, and, conversely, the fluid mass of virtual cells is agglomerated within the real quadrant. This scheme is used by multiple applications, e.g. [19, 26]. We have implemented it in [17, 16].

3.3 Coupling MD and LBM

ESPResSo uses a bidirectional, frictional coupling between particles and fluid adapted from [7, 1]. The force acting on a particle i that stems from the fluid is given as

$$\mathbf{f}_{fl,i} = -\zeta \left(\mathbf{v}_i - \mathbf{u}(\mathbf{x}_i, t)\right) + \mathbf{f}_{st,i},$$

where $\mathbf{v}_i$ is the particle velocity, $\mathbf{u}(\mathbf{x}_i, t)$ the interpolated fluid velocity at the particle's position and ζ the friction coefficient. An additional stochastic force $\mathbf{f}_{st,i}$ is added to each particle. The force $-\mathbf{f}_{fl,i}$ is transferred back to the fluid cell particle i resides in. As regions around particles are areas of interest, we refine the fluid grid around particles to the maximum refinement level. This also improves the accuracy of the coupling.

4 Dynamic Adaptivity and Load-Balancing

As described above, the coupling between the different physical subsystems is realized through a bi-directional force-coupling. Particles in the short-range

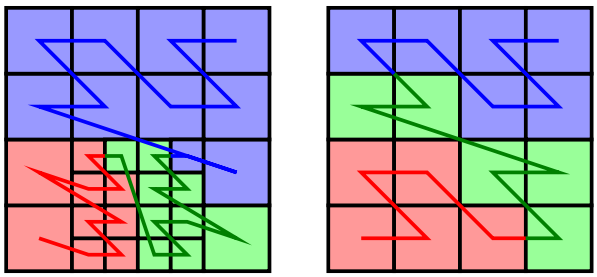


Fig. 4: Two differently discretized grids which are both partitioned independently based on their number of cells among three processes. This leads to a volume-to-volume mapping problem.

MD reside in a regularly discretized grid—technically realized as a `p4est` octree—while the fluid uses a dynamically adaptive grid. In the following, we describe our common tree partitioning approach for the different grids. In a first part, however, we shortly introduce the underlying mapping between particles and fluid cells that define the technical coupling requirements.

4.1 Mapping Data Between Grids

To pass information between both grids, the regular Linked-Cell grid and the adaptive fluid grid, the coupling method needs to find the corresponding fluid cell (in the adaptive grid) for a given particle position. The respective algorithm consists of two steps: 1. We calculate the cell containing the particle in a (virtual) grid that is regularly refined according to the finest grid level present in the adaptive fluid mesh; 2. we calculate the Morton index of the respective cell, i.e., its position in a linearized ordering of all grid cells of this regular grid along the Morton curve; 3. we perform a binary search in the list of real cells of the adaptive fluid grid to determine the correct fluid cell (either the calculated cell of the virtual regular grid or a father/grandfather/... cell). If possible, we use intrinsics for calculating Morton indices, which includes interleaving bits of integers corresponding to a three-dimensional cell index (`PDEP`, parallel bit deposit on x86 Bit Manipulation Instruction Set 2).

If each of the two involved grids were partitioned (load-balanced) independently, a particle might be handled by different processes in the Linked-Cell partitioning than in the fluid partitioning, c.f. Fig. 4. When exchanging information between grids in such a partitioning scheme, this inevitably results in communication not only at partition surfaces, but also in overlapping subvolumes of the computational domain between processes. To avoid this kind of volume-volume mapping, we partition both grids jointly in such a way that volume overlap over partitions assigned to different processes is avoided and, as a result, process-local coupling between MD and LBM is ensured. Of course,

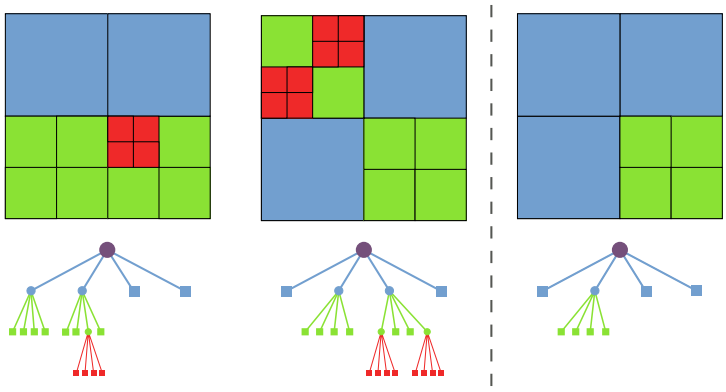


Fig. 5: Two arbitrary, adaptive, two-dimensional grids and their finest common tree on the right. Colors indicate the level of a quadrant. The finest common tree consists of all quadrants which have an equivalent quadrant in both grids or a refined version of it. The tree view below the grid illustrates why we speak of an "intersection": A node in the corresponding tree exists if and only if it exists in the two original grids (image from [3]).

this comes at the cost of not partitioning the involved grids optimally in terms of their own respective objective function (minimizing the time to solution).

4.2 Finest Common Tree and Joint Partitioning

We base the joint partitioning on the *finest common tree* (FCT) which is effectively an intersection of both grids. This means that an octant in the finest common tree is refined if it is refined in both grids, see Fig. 5. Each of the two input grids is given by a corresponding octree. The finest common tree is an "intersection" of both input-octrees in terms of the nodes of the tree. A node in the finest common tree exists if and only if it exists in both input-octrees.

The intersection between two grids is well-defined if both grids use the same space-filling curve and the same macro-structure, i.e., the number and position of trees in the forest of octrees. We realize this by using the same `p4est_connectivity` structure for both grids, the regularly discretized grid for short-range MD and the dynamically adaptive grid for the background flow. The number of trees in the forest of octrees is chosen to be minimal given a required mesh width [11].

We calculate the FCT as shown in Algorithm 1. As an initial guess for the FCT, we copy one of the trees. On the resulting initial FCT guess, we call `p4est`'s coarsening function (line 9). This performs a depth-first post-order traversal of the FCT[1]. For every set of children of a tree node it calls a user-

[1] The callback is called for child cells before it is called on a cell itself.

defined callback function (which is defined for the FCT generation in lines 12 ff.). The callback function looks for overlapping quadrants in the other tree by linearly traversing its quadrant list. This is valid due to the equal order in both trees: If we have two quadrants with indices i and j in tree FCT with $i < j$, also the corresponding overlapping quadrants' ids i' and j' in the other tree fulfill $i' \leq j'$. It is important to note that the post-order traversal induces a non-monotony in the quadrant search: If we have a set of sibling quadrants $0, 1, \ldots, 7$ and one of the siblings $1, \ldots, 7$ has been coarsened before, the index of `q[0]` (line 13, 26) will be smaller than in the last call. We detect this case and reduce the index (i') of the concurrently traversed tree (line 20-22).

Furthermore, the algorithm assumes that the partitioning of both trees is aligned. Although this can be done manually [3], our approach is different: After creating two `p4est` instances with the same connectivity as mentioned above on the same number of processes, the partitioning of the initial (coarse and regular) grid is aligned once. Since it is the sole purpose of our joint partitioning scheme to conserve this property, from there on we simply never let the partitionings diverge.

4.3 Joint Partitioning

Instead of partitioning both grid separately, we partition the finest common tree and transfer the resulting partitioning back to both input grids. This results in a joint partitioning in which a unique process is responsible for every point in space on both grids, see Fig. 6.

Based on the FCT, we implement the joint partitioning scheme as shown in Algorithm 2. Given two trees, it constructs the finest common tree. Then it linearly combines given repartitioning weights per quadrant for the input trees to obtain repartitioning weights for the FCT and performs the repartitioning of the FCT using chain-on-chain partitioning [22]. After that, it determines the partitioning of quadrants for the input trees such that these adhere the partitioning boundaries of the finest common tree. Finally, it asks `p4est` to repartition the trees using the newly determined boundaries.

Note that the finest common tree and our combined partitioning method does not require one of the two involved grids to be regularly discretized. As one can see in the algorithms and in Fig. 5, it works for arbitrary octree grids. Also note that the method is not constrained to two grids. It generalizes to an arbitrary number of grids. However, the penalty with respect to the quality of the partitioning is assumed to get worse if more grids are involved.

5 Results

We evaluate our approach of a joint partitioning for MD simulations with background flow in a weak scaling experiment. The setup is as follows: We consider a rectangular domain of size $2a \times a \times a$ with a given unit length a. The left

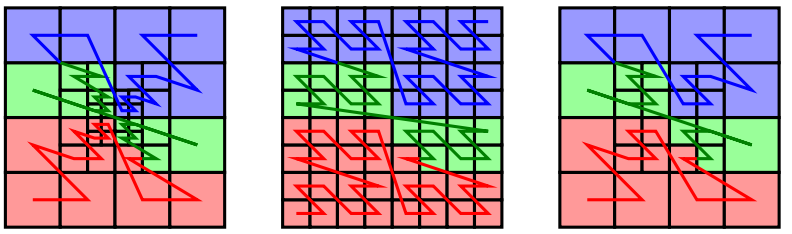
Algorithm 1 Finest common tree creation via P4EST_COARSEN. The coarsen callback finds a corresponding quadrant in the other tree and marks a quadrant for coarsening if its level is higher (more refined). While P4EST_COARSEN traverses one tree, the callback uses a global state to traverse the second tree concurrently. For the sake of brevity we omit pointers. The function VIRT_MORTON_IDX_OF_QUAD implements the mentioned virtual Morton index calculation.

```
1: old_tree ← ⊥                                   ▷ Global state for concurrent tree traversal
2: qid ← 0
3: old_morton_idx ← 0

4: function CREATE_FCT(p4est_t t_1, p4est_t t_2)
5:     fct ← P4EST_COPY(t_1)
6:     fct.user_pointer ← t_2
7:     old_tree ← ⊥
8:     old_morton_idx ← 0
9:     P4EST_COARSEN(fct, COARSEN_CALLBACK)
10:    return fct
11: end function

12: function COARSEN_CALLBACK(p4est_t t_1, int tree_no, p4est_quadrant_t qs[])
13:    q ← qs[0]
14:    t_2 ← t_1.user_pointer
15:    tree ← P4EST_TREE_ARRAY_INDEX(t_2.trees, tree_no)
16:    if tree ≠ old_tree then                    ▷ Reset state if prior call had different tree
17:        old_tree ← tree
18:        qid ← 0
19:    end if
20:    if VIRT_MORTON_IDX_OF_QUAD(q) < old_morton_idx then    ▷ Handle recursive calls
21:        qid ← max{0, qid − (P4EST_CHILDREN − 1)}
22:    end if
23:    old_morton_idx ← VIRT_MORTON_IDX_OF_QUAD(q)
24:    while qid < tree.quadrants.elem_count do
25:        p ← P4EST_QUADRANT_ARRAY_INDEX(tree.quadrants, qid)
26:        if P4EST_QUADRANT_OVERLAPS(p, q) then
27:            return p.level < q.level
28:        end if
29:        qid ← qid + 1
30:    end while
31: end function
```

half of this domain is filled with 24,000 particles interacting via Lennard-Jones potentials. Initially, the particles are randomly distributed across the domain (uniform probability distribution) and we prescribe an initial flow in x-direction. Particles and flow interact as described in Section 3.3. Particles are transported in flow direction, which induces a dynamically changing and non-uniform particle distribution. The lattice-Boltzmann grid is dynamically refined around the particles, meaning that the refinement is also changed if the particles move. We keep the difference between minimum and maximum refinement level constant at three. Fig. 7 shows the setup and visualizes the effect of a joint partitioning with the FCT: The LBM grid is distributed in

Algorithm 2 Joint repartitioning function. Calculates the FCT and weights for its quads, then repartitions it. Afterwards, this partitioning is transferred into partitionings for t_1 and t_2, respectively. The inputs of this function are the trees t_1 and t_2 as well as weights for their quadrants returned by the REPART_WEIGHTS and the factors α_1 and α_2 for the linear combination of them. The function for performing the chain-on-chain partitioning (DETERMINE_PARTITIONS) is not shown here, refer to [12, 22] for details.

function PARTITION_JOINTLY(α_1, t_1, α_2, t_2)
  $fct \leftarrow$ CREATE_FCT(t_1, t_2)
  for all $(s_f, s_1, s_2) \in fct.trees \times t_1.trees \times t_2.trees$ **do** ▷ Calculate weights for FCT
    for all $q \in s_f.quadrants$ **do**
      $Q_1 \leftarrow \{p \in s_1 : p \text{ overlaps } q\}$
      $Q_2 \leftarrow \{p \in s_2 : p \text{ overlaps } q\}$
      $w_1 \leftarrow \sum_{p \in Q_1}$ REPART_WEIGHTS(t_1, p)
      $w_2 \leftarrow \sum_{p \in Q_2}$ REPART_WEIGHTS(t_1, p)
      W[q] $\leftarrow \alpha_1 w_1 + \alpha_2 w_2$ ▷ Combined repart weight for FCT quad
      N_1[q] $\leftarrow |Q_1|$ ▷ Number of quads in t_1 for given FCT quad
      N_2[q] $\leftarrow |Q_2|$
    end for
  end for
  $\mathcal{P} \leftarrow$ DETERMINE_PARTITIONS(W)
  for all $(r, P) \in \mathcal{P}$ **do** ▷ Calculate the number of quadrants per proc. r for t_1, t_2
    $quads_1$[r] $\leftarrow \sum_{q \in P} N_1$[q]
    $quads_2$[r] $\leftarrow \sum_{q \in P} N_2$[q]
  end for
  P4EST_PARTITION_GIVEN($t_1, quads_1$)
  P4EST_PARTITION_GIVEN($t_2, quads_2$)
end function



(a) Adaptive grid (b) Regular grid (c) Finest common tree

Fig. 6: Intersecting both grids leads to the resulting grid on the right. This grid is partitioned (here: based on the number of cells per process on three different processes as indicated by the colors). The obtained partition boundaries are projected back to the original grids.

quadratic chunks of several quadrants. For visualization purposes, we deliberately reduced the FCT level.

We perform the weak scaling experiment on "Hazel Hen" at the High Performance Computing Center Stuttgart (HLRS). We start with 24 processes (1 node) and, then, scale the domain volume and the number of particles successively up by a factor of 8. Simultaneously, we increment the minimum

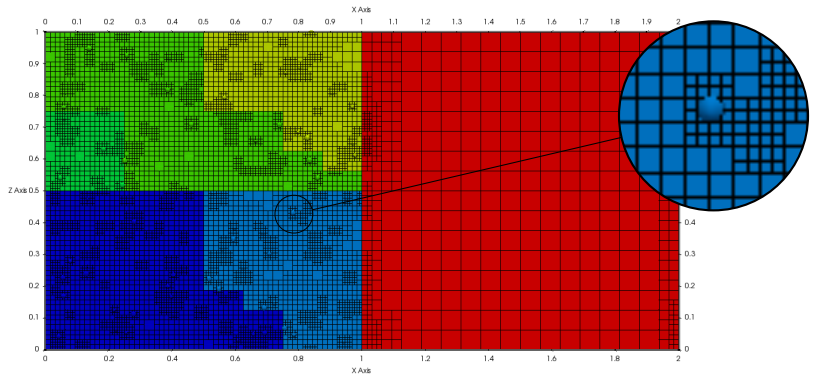


Fig. 7: Setup similar to the one we use in the experiment: For visualization purposes it has been reduced significantly. This figure shows the x-z plane at $y = 0$ with particles on the left and the LBM grid. Colors indicate subdomains. The LBM grid is refined around particles (minimum level: 4, maximum level: 7).

and maximum refinement level of the LBM grid by 1. We present two scaling studies: 1. *case 3-6:* going from levels 3 (min) and 6 (max) on 24 cores over 4 (min) and 7 (max) on 192 cores to 5 (min) and 8 (max) on 1536 cores; 2. *case 4-7:* going from from levels 4 (min) and 7 (max) on 24 cores up to 6 (min) and 9 (max) on 1536 cores. In each experiment, we perform 160 time steps and average over three distinct runs. Every 16 time steps, we perform a grid adaptation and a joint repartitioning. We choose the number of particles as weights for MD cells and a uniform weighting for LBM cells. These weights are linearly combined, both with a factor of one.

The resulting runtimes can be found in Fig. 8. The individual runtimes of LBM and MD can naturally vary since we randomly generate particles. The results show a good weak scaling. The total runtime is largely dominated by the runtime of the LBM, so the combined scaling reflects LBM scaling. Overall, we loose 50 % performance from 24 to 1536 processes in *case 3-6* and only 13.5 % performance in *case 4-7*. These losses are basically induced by the non-perfect scaling of the subcomponents.

To assess the quality of load-balancing, we inspect the imbalance I between all processes based on the maximum and average of the runtimes rs as: $I = \max\{rs\}/\text{avg}\{rs\}$. For all cases, the imbalance of our jointly partitioned simulations is between 1.0 and 1.16 for the individual components (MD and LBM). We assess the runtime of Algorithms 1 and 2 in the context of grid adaption. This includes dynamic refinement around particles, repartitioning, payload migration (particles and LBM populations) and the necessary adaptions of ESPResSo's internal data structures. In this context, the runtime of Algorithms 1 and 2 for the largest test case (*case 4-7* on 1536 processes, i.e.,

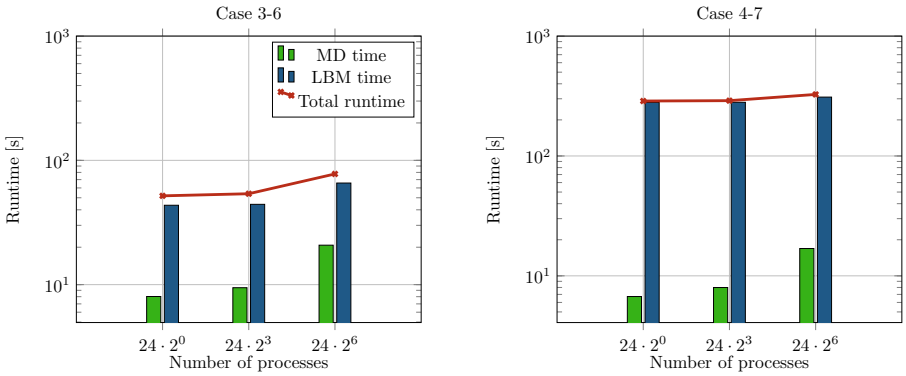


Fig. 8: Runtimes of the two test cases *case 3-6* (left) and *case 4-7* (right) in a weak scaling experiment. The x-axis shows the number of processes (24, 192, 1536) and the y-axis the runtime. Both plots also list the individual times for MD (green) and LBM (blue).

about 120 million quadrants in the LBM grid) is with about 0.053 seconds well below 2 % of the total runtime for grid adaption.

6 Conclusion and Future Work

We have presented a scheme to jointly partition octree grids and evaluated it for an application example, where we perform a lattice-Boltzmann flow simulation on one (adaptive) grid and short-range molecular dynamics calculations on the other (regular Linked-Cell) grid within the simulation package ESPResSo. Our partitioning is based on the finest common tree which is an intersection of two (or more) octrees. Our algorithm naturally limits interactions in sub-volumes of the domain to process-local interactions.

We have demonstrated that the combined repartitioning scheme scales in a dynamic, inhomogeneous, spatially adaptive and coupled MD-LBM setting. The runtime of the algorithm is good and so is the partitioning quality it produces. We have also reported on one of our target applications, a real-world cabin air filtration scenario. We plan to use the joint repartitioning scheme to simulate this scenario dynamically adaptive in the future as reported on in Section 2.

Using our joint repartitioning scheme in a way that minimizes the overall runtime is subject to future work. Weighting factors in all involved grids have to be optimized and the number of time steps between two re-balancing steps needs to be chosen in an optimal and adaptive way. This is a traditional problem in load-balancing. However, this problem gets a third dimension in our case, namely the factors for combining the weights of the different subsystems.

Acknowledgements

The authors gratefully acknowledge funding provided by the German Research Foundation (DFG) as part of the Collaborative Research Center (SFB) 716 (projects D.8 and D.9), the support of our cooperation partners within SFB 716, computing resources provided by the High Performance Computing Center Stuttgart (HLRS), and the support by Carsten Burstedde concerning questions regarding the grid framework `p4est`.

References

[1] P. Ahlrichs and B. Dünweg. "Simulation of a single polymer chain in solution by combining lattice Boltzmann and molecular dynamics". In: *The Journal of chemical physics* 111.17 (1999), pp. 8225–8239.

[2] A. Arnold et al. "ESPResSo 3.1: Molecular Dynamics Software for Coarse-Grained Models". In: *Meshfree Methods for Partial Differential Equations VI*. Ed. by M. Griebel and M. A. Schweitzer. Vol. 89. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, Sept. 2012, pp. 1–23.

[3] M. Brunn. "Coupling of Particle Simulation and Lattice Boltzmann Background Flow on Adaptive Grids". MA thesis. University of Stuttgart, Germany, June 2017.

[4] M. Buchholz. "Framework zur Parallelisierung von Molekulardynamiksimulationen in verfahrenstechnischen Anwendungen". PhD thesis. 2010.

[5] C. Burstedde et al. "p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees". In: *SIAM Journal on Scientific Computing* 33.3 (Jan. 2011), pp. 1103–1133.

[6] P. M. Campbell et al. "Dynamic octree load balancing using space-filling curves". In: *Williams College Department of Computer Science, Technical Report* (2003), pp. 1–26.

[7] B. Dünweg and A. J. C. Ladd. "Lattice Boltzmann Simulations of Soft Matter Systems". In: (2009), pp. 89–166.

[8] B. Dünweg et al. "Statistical mechanics of the fluctuating lattice Boltzmann equation". In: *Physical Review E* 76.3 (Sept. 2007).

[9] B. Forsyth et al. "Particle charge distribution measurement for commonly generated laboratory aerosols". In: *Aerosol Science and Technology* 28.6 (1998), pp. 489–501.

[10] C. Helsper and W. Mölter. "Determination and neutralization of the charge produced by the dispersion of powders". In: *Journal of Aerosol Science* 18.6 (1987), pp. 877–880.

[11] S. Hirschmann et al. "Load Balancing with p4est for Short-Range Molecular Dynamics with ESPResSo". In: ed. by S. Bassini et al. Vol. 32. Advances in Parallel Computing. IOS Press, 2017, pp. 455–464.

[12] S. Hirschmann et al. “Towards Understanding Optimal Load-Balancing of Heterogeneous Short-Range Molecular Dynamics”. In: *Workshop on High Performance Computing and Big Data in Molecular Engineering 2016*. Hyderabad, India, Dec. 2016.

[13] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. Bristol, PA, USA: Taylor & Francis, Inc., 1988.

[14] T. Isaac et al. “Recursive Algorithms for Distributed Forests of Octrees”. In: *SIAM J. Sci. Comput.* 37.5 (Jan. 2015), pp. C497–C531.

[15] A. M. Johnston et al. “Electrical charge characteristics of dry aerosols produced by a number of laboratory mechanical dispensers”. In: *Aerosol science and technology* 6.2 (1987), pp. 115–127.

[16] M. Lahnert et al. “Minimally-Invasive Integration of p4est in ESPResSo for Adaptive Lattice-Boltzmann”. In: *The 30th Computational Fluid Dynamics Symposium*. Japan Society of Fluid Mechanics, 2016.

[17] M. Lahnert et al. “Towards Lattice-Boltzmann on Dynamically Adaptive Grids – Minimally-Invasive Grid Exchange in ESPResSo”. In: *VII European Congress on Computational Methods in Applied Sciences and Engineering*. Ed. by M. Papadrakakis et al. ECCOMAS, June 2016.

[18] H. J. Limbach et al. “ESPResSo – An Extensible Simulation Package for Research on Soft Matter Systems”. In: *Computer Physics Communications* 174.9 (May 2006), pp. 704–727.

[19] M. Mehl et al. “Navier-Stokes and Lattice-Boltzmann on octree-like grids in the Peano framework”. In: *International Journal for Numerical Methods in Fluids* 65.1-3 (Nov. 2010), pp. 67–86.

[20] W. F. Mitchell. “A Refinement-tree Based Partitioning Method for Dynamic Load Balancing with Adaptively Refined Grids”. In: *J. Parallel Distrib. Comput.* 67.4 (Apr. 2007), pp. 417–429.

[21] G. M. Morton. *A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing*. Tech. rep. IBM Ltd., 1966.

[22] A. Pınar and C. Aykanat. “Fast optimal load balancing algorithms for 1D partitioning”. In: *Journal of Parallel and Distributed Computing* 64.8 (2004), pp. 974–996.

[23] M. Rohde et al. “A generic, mass conservative local grid refinement technique for lattice-Boltzmann schemes”. In: *International Journal for Numerical Methods in Fluids* 51.4 (2006), pp. 439–468.

[24] U. D. Schiller. “Thermal fluctuations and boundary conditions in the lattice Boltzmann method”. PhD thesis. Johannes Gutenberg-Universität, Mainz, 2008.

[25] C. Schober et al. “Simulating The Interaction of Electrostatically Charged Particles in the Inflow Area of Cabin Air Filters Using a Fully Coupled System”. In: *Coupled Problems in Science and Engineering VII*. Ed. by M. Papadrakakis et al. June 2017.

[26] F. Schornbaum and U. Rüde. “Massively Parallel Algorithms for the Lattice Boltzmann Method on NonUniform Grids”. In: *SIAM Journal on Scientific Computing* 38.2 (Jan. 2016), pp. C96–C126.

[27] I. Tischler. “Implementing adaptive Electrokinetics in ESPResSo”. MA thesis. University of Stuttgart, Germany, Apr. 2018.

[28] C. Xiaolin and M. Zeyao. “A New Scalable Parallel Method for Molecular Dynamics Based on Cell-Block Data Structure”. In: *Parallel and Distributed Processing and Applications*. Ed. by J. Cao et al. Vol. 3358. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 757–764.