Parallel Computing: Technology Trends
I. Foster et al. (Eds.)

doi:10.3233/APC200035

Load-Balancing for Large-Scale Soot Particle Agglomeration Simulations

Steffen HIRSCHMANN [a,1], Andreas KRONENBURG [b], Colin W. GLASS [c] and Dirk PFLÜGER [a]

[a] *Institute for Parallel and Distributed Systems, University of Stuttgart, Germany*
[b] *Institute for Combustion Technology, University of Stuttgart, Germany*
[c] *Department of Mechanical Engineering, Helmut Schmidt University Hamburg, Germany*

Abstract. In this work, we combine several previous efforts to simulate a large-scale soot particle agglomeration with a dynamic, multi-scale turbulent background flow field. We build upon previous simulations which include 3.2 million particles and implement load-balancing into the used simulation software as well as tests of the load-balancing mechanisms on this scenario. We increase the simulation to 109.85 million particles, superpose a dynamically changing multi-scale background flow field and use our software enhancements to the molecular dynamics software ESPResSo to simulate this on a Cray XC40 supercomputer. To verify that our setup reproduces essential physics we scale the influence of the flow field down to make the scenario mostly homogeneous on the subdomain scale. Finally, we show that even on the homogeneous version of this soot particle agglomeration simulation, load-balancing still pays off.

Keywords. molecular dynamics, short-range, dynamic load-balancing, soot-particle agglomeration, domain decomposition

1. Introduction

Short-range molecular dynamics (MD) [1] is an important field in Computational Sciences. One particular example of a real-world application is the simulation of soot particle agglomeration, which, for example, is relevant for the efficiency of industrial processes. In these processes, particles collide and link irreversibly. Of particular interest is the morphology of the resulting agglomerates. Because results of a computer simulation allows the examination of morphology of agglomerates over time, computer simulation plays an important role in this area.

The approach we use has been described in [2,3]: Agglomeration processes are simulated in a precomputed, turbulent flow field. Clustering of particles is driven by Brownian motion as well as the background flow, which gets more important as the particle density decreases. The influence of a turbulent background flow field is of particular interest in particle agglomeration simulations. While small turbulence scales can be resolved in

[1]Corresponding Author: Steffen Hirschmann, Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart; Email: steffen.hirschmann@ipvs.uni-stuttgart.de

small simulations, the question remains if large, multi-scale turbulence flows critically influence the results.

In order to get to more realistic agglomeration simulations we use a larger and dynamically changing flow field that covers more scales of turbulence as well as a larger setup including the number of particles. Starting from the largest simulation in [3] ("Case 6"), we increase the domain size by a factor of 3.25. This allows us to cover more realistic scales of turbulence. We keep the original particle loading. Hence, we increase the number of primary particles from 3.2 million in [3] by a factor of $3.25^3 \approx 34.33$ to 109.85 million.

This scenario is very large considering the elaborate physical bonding model used. In fact, to the best of our knowledge it is the largest simulation with ESPResSo so far. And, because we simulate an agglomeration process, the simulation naturally gets more and more heterogeneous over time. Simulations of these sizes and types pose two major challenges: (1) We need large-scale parallelism to cope with a simulation of this size, and (2) we must dynamically adapt the domain decomposition to the changing particle distribution in order to cope with load-imbalances arising from heterogeneity.

These challenges require us to combine this large-scale real world scenario with our previous efforts to bring dynamic load-balancing to the MD software ESPResSo. In particular, we need to make use of dynamic load-balancing at runtime, the newly created, non-regularly partitioned grids and their associated asynchronous communication [4,5] as well as other contributions to the MD software at hand, like parallel input and output using MPI-IO.

In order to validate the physical correctness of the scenario and assess the applicability of our load-balancing methods, we use a rather homogeneous version of the scenario. A more homogeneous scenario allows us to first focus on physical correctness of the setup while not crucially depending on the best possible load-balancing. Following the simulation, we can test the applicability of our load-balancing methods for this scenario.

The remainder of this work is structured as follows: In Section 2 we report on related research. In Section 3 we elaborate on the numerical simulation models as well as the used code and our load-balancing methodology. We describe our simulation setup in Section 4. Subsequently, in Section 5 we show the physical results and our assessment of load-balancing for this setup. Finally, in Section 6 we summarize our work and conclude with a note on further topics to investigate.

2. Related Work

At the core of our modeling are Langevin-based agglomeration processes. These have, e.g., been studied in [6]. We are, however, interested in agglomeration processes that are subject to a turbulent background flow field and that links particles irreversibly. This linking process should completely prohibit rotation and sliding of the particles. To this end, the Langevin-based model has been augmented in [2,3] to include a coupling to a static flow field as well as dynamic bonding to link particles irreversibly at runtime. In [2] several bonding models that effectively prohibit sliding are proposed and evaluated. We use the so-called "AB" (*all-bonds*) model from this work, which has the advantage, that it does not require the addition of virtual particles.

A different approach to tackle the upscaling of agglomeration simulation is coarse-graining, i.e. aggregating whole clusters into one "super particle" (during the simulation) and, thus, reducing the total computational burden. This is still a field of active research, as the involved modeling is complex. For example, coarse grained particle need to accurately resolve the collision probabilities of the underlying "real" cluster. Algorithmically, this kind of dynamic coarse graining leads to larger cell sizes and, thus, likely to less parallelism and more load-imbalances. Studies and first results for this approach are, e.g., presented in [7]. However, the technique described there is not ready yet for large-scale simulations such as ours.

For load-balancing several heuristics are used in existing MD (and other) software. We have discussed details on the most commonly used ones in [8] and have implemented some of them in the MD software ESPResSo in [4,9,5]. In this work, we focus on the partitioner based on Space-filling curves (SFC) leveraging the well-known and scalable library `p4est` [10,11], which in turn uses the Z-curve [12]. The actual partitioning for SFC-based algorithms is performed using so-called chain-on-chain partitioning [13]. Several studies find graph partitioning performs best because of its superior model while SFC-based partitioning is fast and consumes less memory [14,15]. A more theoretical review of several partitioning algorithms can be found in [16] listing important properties like speed, memory usage, etc. Eibl and Rüde [17] inspect different partitioners for the discrete element method and find that there is a trade-off between scalability and quality of partitioning and recommend an SFC-based strategy for small and mid-sized scenarios. Particularly for MD, several methods are implemented in the simulation software "ls1 mardyn" and compared in [18,19,20]. These studies also present cost heuristics to estimate the load of individual subdomains. They give us enough reason to focus on SFC-based partitioning first for our current work.

3. Methodology

Our main methodological approach is two-part. First, we explain what numerical models we use to simulate a soot particle agglomeration process within the molecular dynamics framework. Second, we explain what parallelization and load-balancing approaches we use for the implementation of the numerical models. Additionally, we briefly introduce the relevant quantities for analyzing the shape of agglomerates.

3.1. Numerical Models

We model intermolecular interactions with the well-known Lennard-Jones-12-6 potential which consists of an attractive and a repulsive part,

$$U_{LJ}(r) = 4\varepsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right),$$

where σ and ε are properties of the modeled primary particles. Particles can be bound together with distance-based and angular harmonic bonds. Their associated potentials are given as

$$U_{distance}(r) = \frac{1}{2}k_h\,(r - r_0)^2 \qquad \text{, and} \qquad U_{angular}(\phi) = \frac{1}{2}k_a\,(\phi - \phi_0)^2,$$

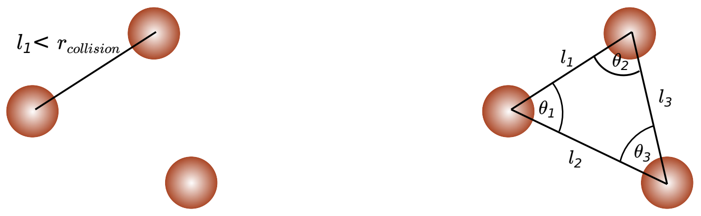

Figure 1. Visualization of bonding at runtime. Left: If two particles are closer than the collision distance, the algorithm aims to finds a third particle within the collision distance to establish a triangular bonding structure. Right: The bonding structure consists of three angular bonds (indicated as $\theta_1, \theta_2, \theta_3$) as well as three distance-based bonds (indicated as l_1, l_2, l_3) between the particle pairs. If no third particle could be found, only the distance bond l_1 is created.

where r_0 and ϕ_0 are the equilibrium distance and angle, respectively, and k_h and k_a spring constants. These bonds are established in groups at runtime for each triple of particles that collides. This so-called “AB” model is adapted from [2] and depicted in Figure 1. Note, that the equilibrium angle ϕ_0 is not constant across bonds but rather different for each one. It is chosen as the angle between the particles at collision time.

In order to model frictional influence from a fluid and Brownian motion, we use Langevin Dynamics. A study of purely Langevin-driven agglomeration processes without turbulent background flow can be found in [6]. The equation of motion is given as:

$$m\ddot{\vec{x}} = \vec{f} - \gamma\left(\dot{\vec{x}} - \vec{u}_{flow}(t,\vec{x})\right) + \vec{R}(t), \tag{1}$$

where $\vec{f}$ are the forces given by the intermolecular potentials described above. Additionally, $\vec{R}(t)$ is a random noise, which, together with the frictional term $\gamma(\dot{\vec{x}} - \vec{u}_{flow}(t,\vec{x}))$ models Brownian motion and the temperature, as well as the frictional influence of the fluid. The velocities $\vec{u}_{flow}(t,\vec{x})$ stem from the fluid (external flow field) at time t and position $\vec{x}$. Analogously to [3] we model the friction between the fluid and the particles by Stokes' law, so $\gamma = 3\pi\mu\sigma/C_c$ where μ is the viscosity of the fluid, σ the particle diameter and C_c being the Cunningham correction factor. In ESPResSo, Langevin Dynamics is implemented with a Velocity Verlet integrator, see e.g. [21], combined with a so-called Langevin thermostat [22] that applies the frictional and random forces given the temperature and γ.

3.2. Parallelization and Load-Balancing

We use the simulation software ESPResSo[2] [23,22], which covers all the relevant physics involved. Relevant parts of the dynamic bonding mechanisms have been implemented in the course of [2,3] and are also described in [23]. ESPResSo uses the Linked-Cell algorithm [24] in combination with Verlet lists to calculate forces stemming from short-range potentials, like the Lennard-Jones potential, in linear time. Based on the Linked-Cell discretization, ESPResSo defines a uniform spatial domain decomposition to allow for MPI-based parallelization [22]. While the simulation core of ESPResSo is implemented in C++, it exposes a Python-based front-end for setting up the scenario and controlling the simulation [25].

[2]Extensible Simulation Package for Research on Soft Matter, `http://www.espressomd.org`

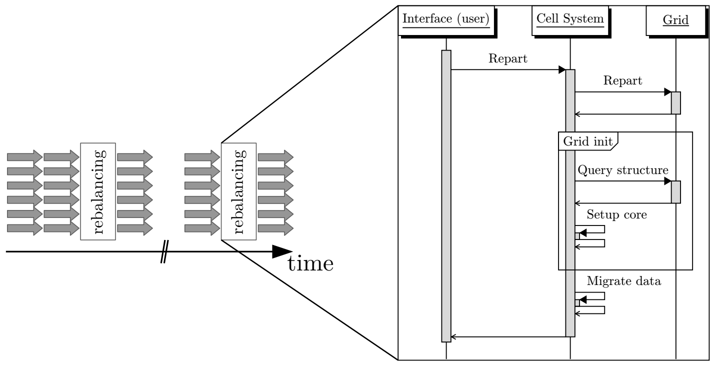

Figure 2. Left: A sketch of a segment of an MD simulation. The gray arrows depict regular time steps. From time to time the domain decomposition has to be adapted to the underlying scenario (indicated by the "rebalancing" boxes). We outline our implementation on the right: The user calls a function "repart" from their script. The linked cell grid ("cell system") internally asks the grid to repartition itself and then sets up the established partitioning in the core of the simulation software. Afterwards, cell payload (particles) is migrated transparently for the user.

Our adaptions keep the MPI-only parallelization and its 1:1 mapping of subdomains to processes. In [4] we devised a general scheme to change this fixed decomposition, allowing for arbitrary ones. Based on this work, we have implemented different decompositions. We make the load-balancing mechanism available to the user, so they can conveniently implement strategies for partitioning scenarios in the simulation scripts themselves. Note that this load-balancing mechanism is not constrained to the application presented in this work. It can be employed to any heterogeneous simulation in ESPResSo. We depict this ability to do dynamic repartitioning and sketch the underlying implementation in Figure 2.

Given a function m that defines a suitable load metric or measurement of execution time for every process $p \in \{1, \dots, P\}$, we partition based on the imbalance $\mathscr{I}(m)$. The imbalance is defined as the maximum over average load measurement: $\mathscr{I}(m) = \frac{P \max\{m(p)\}}{\sum_p m(p)}$. In the current setup, we use $\mathscr{I}(m) > 1.1$ as criterion with the load metric $m(p)$ as the number of particles of process p and partition at most every 1000 time steps. In [4] we have shown for a smaller agglomeration scenario that choosing the number of particles as metric $m(p)$ performs best among a range of different choices.

3.3. Analysis

An important characterization of particle clusters is their fractal dimension D_f [26]. It is the power law relationship of the number of particles to their radius, see e.g. [27], and calculated as

$$N = \left(\frac{r_g}{d}\right)^{D_f},$$

with N the number of particles in the cluster, $d = \frac{\sigma}{2}$ and r_g the radius of gyration, which is the standard deviation of the particle positions $\vec{r}_i$ in a cluster:

$\hat{n}$	T	σ	l_0
$6.25 \cdot 10^{-3}$	600 K	20 nm	$2600\,\sigma$

Table 1. Basic MD simulation parameters.

σ	ε^*	t^*
20 nm	$1.25 \cdot 10^{-20}$ J	14 ns

Table 2. Reference values used for nondimensionalization of physical quantities in the MD simulation.

$$r_g = \sqrt{\frac{1}{N}\sum_{i=1}^{N} ||\vec{r} - \vec{r}_i||^2} \quad \text{, with} \quad r = \frac{1}{N}\sum_{i=1}^{N} \vec{r}_i.$$

4. Simulation Setup

The basic simulation parameters, which will be explained in the following, can be found in Table 1. The reference length, energy and time used for nondimensionalization of the MD simulation can be found in Table 2. The simulation comprises $109.85 \cdot 10^6$ particles, the largest simulation with ESPResSo so far. Each of these primary particles has a diameter of $\sigma = 20\,\text{nm}$. Initially, they are placed in a simulation box of size $l_0 = 2600\,\sigma$, uniformly randomly distributed. We employ periodic boundary conditions in all dimensions. The particle loading is $\hat{n} = 6.25 \cdot 10^{-3}$. The temperature of the solvent is $T = 600\,\text{K}$.

The flow field is primarily characterized by its kinematic viscosity ν and its dissipation rate ε. Based on these, we can derive the characteristic time scales, namely Brownian diffusion time t_{BM} and the Kolmogorov time scale t_k. These allow us to define the nondimensional Péclet number $\text{Pe} = \frac{t_{\text{BM}}}{t_k}$ that describes the relative importance of turbulence over Brownian motion. The second nondimensional quantity that is used in [3] to describe a scenario is the Knudsen number $\text{Kn} = \frac{l_{mfp}}{\sigma/2}$, where l_{mfp} is the mean free path length of the flow.

Our goal is to reproduce a larger version with more turbulent flow scales of "Case 6" from [3]. This setup uses $\text{Kn} = 11$ and $\text{Pe} = 1$. To achieve that, we generate the external background flow field in a separate pre-processing step using a Direct Numerical Simulation (DNS) of homogeneous, isotropic forced turbulence in a box of length $L = 2\pi l_0 \approx 326.7\,\mu\text{m}$. It solves the incompressible Navier-Stokes equations with periodic boundary conditions, discretized on a $64 \times 64 \times 64$ mesh. This mesh is unrelated to the linked cell grid and only defines the resolution of the flow field in the later MD simulation. The viscosity is $\nu = 5.13 \cdot 10^{-5}\,\text{m}/\text{s}^2$ and the dissipation rate $\varepsilon = 1.25 \cdot 10^{10}\,\text{m}^2/\text{s}^3$, which equals the desired values of Kn and Pe. These, however, lead to very heterogeneous particle distributions as we have shown in [4,5] on basis of "Case 6" from [3]. For first experiments at scale, we keep the particle distribution mostly homogeneous on the subdomain scale by reducing the influence of the flow field in the transport equation. Therefore, we scale down $\vec{u}_{flow}(t, \vec{x}_i)$ in Equation 1 by about $^1/_3$. This rescales the gradients of the velocity by an equal factor, and, thus, also the dissipation rate. So in our current simulation the Péclet number is roughly a third of the intended target value.

We compute $6 \cdot 10^7$ iterations, each having a length of $dt = 10^{-4}\,t^* = 1.4\,\text{fs}$. Thus, the end of the simulation is at $t_{end} = 8.4\,\mu\text{s}$. The bonding constants $\tilde{k}_a$ and $\tilde{k}_h$ are estimated according to [3] to $\tilde{k}_a = \tilde{k}_h = 1000\,\varepsilon^*$. As collision distance, we use $r_{col} = \sigma$.

As mentioned above, we have extended and use the simulation software ESPResSo as it implements all relevant numerical models, especially dynamic bonding at runtime. We perform the scenario setup as follows: (1) Setup random particles with zero velocities, (2) initialize all required potentials, (3) equilibrate the system using steepest descend

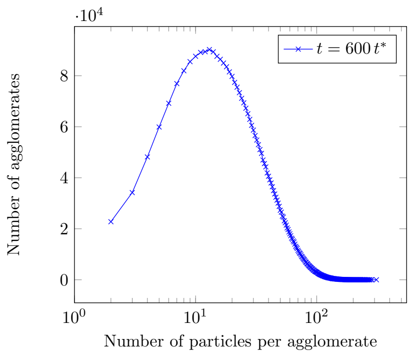

Figure 3. Histogram of the number of agglomerates per size. Size is in number of particles per agglomerate. The location of the maximum indicates that the agglomeration process has left its initial state where growth is mainly driven by collisions of primary particles.

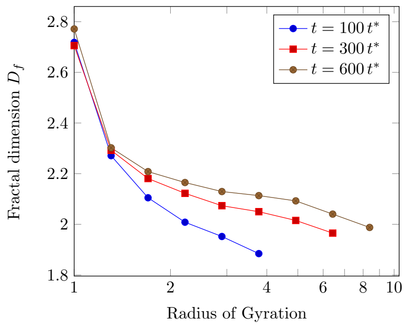

Figure 4. Average fractal dimension D_f of agglomerates of all agglomerates of a certain radius of gyration at different time steps. While $t = 100t^*$ is still early in the simulation, the average D_f does not vary much in later time steps.

integration [25], (4) reset velocities and forces to zero, (5) setup the thermostat as well as collision detection and dynamic bonding, and (6) start the simulation.

5. Results

One indicator of how much the agglomeration process has progressed in time, is the number of agglomerates of a certain size. We present a histogram of the sizes of the agglomeration for $t_{end} = 600t^*$ in Figure 3. We clearly see that the most prevalent cluster size is about one order of magnitude higher than the smallest possible (2 primary particles). This means that the simulation has left the initial state where the growth of agglomerates is mainly driven by primary particle collisions. At t_{end} process is primarily driven by cluster-cluster collisions for significant growth of the agglomerates.

In Figure 4 we plot the average D_f of all agglomerates with more than 15 particles at $t = 100t^*$, $300t^*$, and $600t^*$ depending on the radius of gyration of the agglomerates. This relationship enables scientists to understand the agglomeration process and to determine its influence on larger industrial processes and products. Therefore, the criterion for stopping the simulation in [3] is when the individual lines converge. While the process clearly has not converged yet at $t = 100t^*$ in Figure 4, the difference in D_f for the same r_g between $t = 300t^*$ and $600t^*$ gets significantly smaller, indicating a possible convergence. Since the agglomerates are still rather small ($r_g \leq 10$), the average fractal dimension is in the range of $1.9 \leq D_f \leq 2.2$. These D_f are a bit higher than the ones published by [3] using the same ansatz, as well as experimental data obtained from soot aggregates in (turbulent) flames, see e.g. [28]. However, the smaller the velocities of the superposed, turbulent background flow field in Equation 1, the higher the influence of Brownian motion. In this case, [3] also states, that with a higher influence of Brownian motion, D_f approaches 2 for larger agglomerates. Beyond that we can see a trend for agglomerates with larger r_g to have a smaller D_f. In conclusion, we can say, that we are able to reproduce physical results for the simulation of soot particle agglomeration, and we assume that the mentioned differences stem from the smaller flow field velocities.

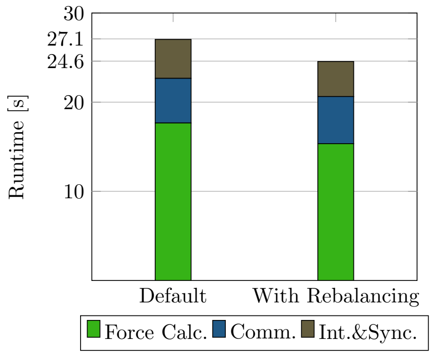

Figure 5. Runtime of 1000 time steps beginning at $t_0 = t_{end} = 600t^*$ for the default, unbalanced paratitioning ("Default") and our SFC-based partitioning ("With Rebalancing") for 7200 processes. For each of the three components (force calculation with the Linked-Cell/Verlet-list method, communication and integration including synchronization) we plot the maximum runtime of any process. Even though the setup is quite homogeneous, we can reduce the runtime by about 10 %.

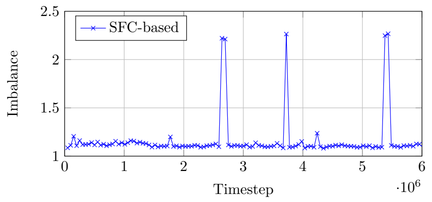

Figure 6. Imbalance in runtime per 1000 time steps between the different processes.Rebalancing with the SFC-based method is performed if the number of particles diverges by 10 % on any process from the average. The number of particles per cell is also used as weights for repartitioning. The spikes are still under investigation. Their occasional occurrences *could* simply be due to an influence of non-deterministic components like hardware, communication, general network load on the HPC system etc.

5.1. Load-balancing

Although we enforce a more or less homogeneous particle distribution among the subdomains, we still use load-balancing to counteract smaller heterogeneity that evolves over time. For unscaled versions of this scenario, good load-balancing is crucial as our studies on the smaller 3.2 million particle setup [4,5] have shown. Therefore, we also test and show load-balancing results here. The simulation ran on "Hazel Hen" at the High Performance Computing Center Stuttgart (HLRS). It is a Cray XC40 machine with 2 Intel Xeon E5-2680v3 ("Haswell" microarchitecture) per node, each of which has 12 cores (not counting Simultaneous Multi-Threading) and a Cray Aries interconnect.

We use the snapshot at $t_{end} = 600t^*$ for our test. We run the test on 300 nodes, i.e. 7200 processes. The imbalance in the number of particles for a decomposition into equally sized boxes (`MPI_Dims_create`) is about 18.4 %, which is quite homogeneous. The runtimes for the default parallelization and our load-balanced one can be found in Figure 5. We plot the relevant runtimes in the following way: Let

$$f_1 = \max\{t_{force}(p)\}_{p=1}^{P},$$

$$f_2 = \max\{t_{force}(p) + t_{comm}(p)\}_{p=1}^{P}, \text{ and}$$

$$f_3 = \max\{t_{force}(p) + t_{comm}(p) + t_{int}(p) + t_{sync}(p)\}_{p=1}^{P},$$

where "force", "comm", "int" and "sync" refer to the individual components: force calculation, communication, integration and synchronization. Then, we plot f_1, $f_2 - f_1$ and $f_3 - f_2 - f_1$, i.e. the difference of the individual maxima runtimes of the phases. We can see, that despite the homogeneous particle distribution, we achieve a runtime reduction of about 10 %. Additionally, in Figure 6 we can see, that the load-balancing is capable of keeping the imbalance at about the desired level of 1.1 during almost the entire 6 million time steps. The occasional spikes are still being investigated. Given the overall behavior, however, it is likely, that the spikes stem from runtime noise.

6. Conclusion

We successfully combined several previous works into one large-scale, load-balanced soot particle agglomeration simulation. We set up the simulation with a complex physical bonding model, as well as a dynamically changing, multi-scale turbulent background flow field. We increased the simulation to over 100 million particles, which is over 30 times larger than previous works with ESPResSo and, to the best of our knowledge, the largest simulation ever with ESPResSo.

In order to assess the physical correctness of the new setup, we kept the simulation rather homogeneous by rescaling the velocities stemming from the superposed flow field. This way, we did not have to deal with large heterogeneity, and we were able to reproduce previous results. We showed that the resulting fractal dimensions of the clusters seem reasonable and consistent with previous results. Also, we showed, that even though we keep heterogeneity low, load-balancing is still able to reduce the runtimes by about 10 % and to consistently keep the imbalance in runtime low throughout the simulation.

6.1. Future Work

There are two paths that we will pursue further. As we have verified that the setup is physically correct and that load-balancing pays off, the first path is to use a physically correct flow field with Pe $\approx$ 1. This will let us study the impact of the multi-scale turbulent and dynamic background flow field on agglomeration processes at physically relevant scales.

Second, we have implemented different kinds of load-balancing methods in previous work [4,9,5], some of which might be more suitable for the heterogeneous version of the simulation. We intend to test different methods as well as different metrics and relate them to scenario properties in order to gain deeper insight into the problem of balancing heterogeneous simulations with a low average fractal dimension of the clusters.

Acknowledgements

The authors gratefully acknowledge financial support provided by the German Research Foundation (DFG) as part of the former Collaborative Research Center (SFB) 716, and the computing time on "Hazel Hen" granted by the High Performance Computing Center Stuttgart (HLRS). Special thanks go to Rudolf Weeber from the ICP, University of Stuttgart, for his expertise and fruitful discussions.

References

[1] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, 1989.

[2] Gizem Inci et al. Modeling nanoparticle agglomeration using local interactions. *Aerosol Science and Technology*, 48(8):842–852, July 2014.

[3] Gizem Inci et al. Langevin dynamics simulation of transport and aggregation of soot nano-particles in turbulent flows. *Flow, Turbulence and Combustion*, pages 1–21, January 2017.

[4] Steffen Hirschmann et al. *Load Balancing with p4est for Short-Range Molecular Dynamics with ESPResSo*, volume 32 of *Advances in Parallel Computing*, pages 455–464. IOS Press, 2017.

[5] Steffen Hirschmann, Colin W. Glass, and Dirk Pflüger. Enabling unstructured domain decompositions for inhomogeneous short-range molecular dynamics in ESPResSo. *The European Physical Journal Special Topics*, 227(14):1779–1788, March 2019.

[6] Lorenzo Isella and Yannis Drossinos. Langevin agglomeration of nanoparticles interacting via a central potential. *Physical Review E*, 82:011404, July 2010.

[7] Milena Smiljanic et al. Developing coarse-grained models for agglomerate growth. *The European Physical Journal Special Topics*, 227(14):1515–1527, March 2019.

[8] Steffen Hirschmann, Dirk Pflüger, and Colin W. Glass. Towards understanding optimal Load-Balancing of heterogeneous Short-Range molecular dynamics. In *Workshop on High Performance Computing and Big Data in Molecular Engineering 2016 (HBME 2016)*, Hyderabad, India, December 2016.

[9] Steffen Hirschmann et al. *Load-Balancing and Spatial Adaptivity for Coarse-Grained Molecular Dynamics Applications*. Springer, 2018.

[10] Carsten Burstedde et al. p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, January 2011.

[11] Tobin Isaac, Carsten Burstedde, Lucas C. Wilcox, and Omar Ghattas. Recursive algorithms for distributed forests of octrees. *SIAM Journal on Scientific Computing*, 37(5):C497–C531, January 2015.

[12] G. M. Morton. A computer oriented geodetic data base; and a new technique in file sequencing. Technical report, IBM Ltd., 1966.

[13] Ali Pınar and Cevdet Aykanat. Fast optimal load balancing algorithms for 1D partitioning. *Journal of Parallel and Distributed Computing*, 64(8):974–996, 2004.

[14] William F. Mitchell. A refinement-tree based partitioning method for dynamic load balancing with adaptively refined grids. *Journal of Parallel and Distributed Computing*, 67(4):417–429, April 2007.

[15] S. Schambeger and J. M. Wierum. Graph partitioning in scientific simulations: Multilevel schemes versus space-filling curves. In *International Conference on Parallel Computing Technologies*, volume 2763 of *LNCS*, pages 165–179, 2003.

[16] Bruce Hendrickson and Tamara G. Kolda. Graph partitioning models for parallel computing. *Parallel Computing*, 26(12):1519 – 1534, 2000.

[17] Sebastian Eibl and Ulrich Rüde. A systematic comparison of runtime load balancing algorithms for massively parallel rigid particle dynamics. *Computer Physics Communications*, 2019.

[18] Martin Buchholz. *Framework zur Parallelisierung von Molekulardynamiksimulationen in verfahrenstechnischen Anwendungen*. Verlag Dr. Hut, 2010.

[19] M. Buchholz, H.-J. Bungartz, and J. Vrabec. Software design for a highly parallel molecular dynamics simulation framework in chemical engineering. *Journal of Computational Science*, 2(2):124–129, 2011.

[20] Christoph Niethammer et al. ls1 mardyn: The massively parallel molecular dynamics code for large systems. *Journal of Chemical Theory and Computation*, 10(10), October 2014.

[21] Benedict J. Leimkuhler et al. Integration methods for molecular dynamics. In *Mathematical Approaches to Biomolecular Structure and Dynamics*, volume 82 of *The IMA Volumes in Mathematics and its Applications*, pages 161–185. Springer New York, 1996.

[22] Hans Jörg Limbach et al. ESPResSo – an extensible simulation package for research on soft matter systems. *Computer Physics Communications*, 174(9):704–727, May 2006.

[23] Axel Arnold et al. ESPResSo 3.1: Molecular dynamics software for coarse-grained models. In *Meshfree Methods for Partial Differential Equations VI*, volume 89 of *Lecture Notes in Computational Science and Engineering*, pages 1–23. Springer Berlin Heidelberg, September 2013.

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

[25] Florian Weik et al. Espresso 4.0 – an extensible software package for simulating soft matter systems. *The European Physical Journal Special Topics*, 227(14):1789–1816, March 2019.

[26] Benoit B. Mandelbrot. *The fractal geometry of nature*, volume 173. W. H. Freeman New York, 1983.

[27] R. J. Samson et al. Structural analysis of soot agglomerates. *Langmuir*, 3(2):272–281, 1987.

[28] Ümit Ö. Köylü, Yangchuan Xing, and Daniel E. Rosner. Fractal morphology analysis of combustion-generated aggregates using angular light scattering and electron microscope images. *Langmuir*, 11(12):4848–4854, 1995.