Static Load Balancing for Molecular-Continuum Flow Simulations with Heterogeneous Particle Systems and on Heterogeneous Hardware

Amartya Das Sharma[1][0009−0009−8237−4295], Louis Viot[1], Piet Jarmatz[1][0000−0002−5463−0740], Hauke Preuß[1][0009−0002−9605−1322], and Philipp Neumann[2,3][0000−0001−8604−8846]

[1] High Performance Computing,
Helmut Schmidt University, Hamburg, Germany
das-sharma@hsu-hh.de
[2] High Performance Computing and Data Science,
Universität Hamburg, Hamburg, Germany
[3] Deutsches Elektronen-Synchrotron (DESY)
Hamburg, Germany
philipp.neumann@desy.de

Abstract. Load balancing in particle simulations is a well-researched field, but its effect on molecular-continuum coupled simulations is comparatively less explored.
In this work, we implement static load balancing into the macro-micro-coupling tool (MaMiCo), a software for molecular-continuum coupling, and demonstrate its effectiveness in two classes of experiments by coupling with the particle simulation software ls1 mardyn. The first class comprises a liquid-vapour multiphase scenario, modelling evaporation of a liquid into vacuum and requiring load balancing due to heterogeneous particle distributions in space. The second class considers execution of molecular-continuum simulations on heterogeneous hardware, running at very different efficiencies. After a series of experiments with balanced and unbalanced setups, we find that, with our balanced configurations, we achieve a reduction in runtime by 44% and 55% respectively.

Keywords: Coupled Simulations · Multiphase Simulations · Load Balancing · Molecular-Continuum · Heterogeneous Architecture

1 Introduction

Coupled multiscale simulations are a viable way of reducing compute times while retaining simulation details. Since they are often still computationally expensive, leveraging the full capabilities of hardware and software is greatly desirable.

One example for coupled multiscale simulations are molecular-continuum systems: molecular dynamics (MD) simulation is employed in areas where greater granularity is desired, and a classical computational fluid dynamics (CFD) solver

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

is used everywhere else. One way that MD simulations gain performance in heterogeneous situations is through effectively balancing computational load in a distributed environment. In this work, we extend the molecular-continuum coupling tool MaMiCo with static load balancing. We demonstrate the effectiveness using well-established community codes, such as the open-source MD library ls1 mardyn, which leverages the node-level library AutoPas in its kernel, or the CFD package OpenFOAM [27], using the popular coupling tool preCICE [4,5] as a bridge to MaMiCo. Heterogeneity is considered in two ways: a multiphase scenario modelling evaporation, introducing heterogeneity due to inhomogeneous particle distributions, and a heterogeneous-architecture scenario with the simulation running on a mix of x86 and aarch64 hardware.

In section 2, the dominant software stack of ls1 mardyn + AutoPas + MaMiCo is introduced, and related work is discussed. Coupling and load balancing are explained in section 3. We then describe and analyse the evaporation scenario in section 4 and a homogeneous Couette flow scenario on heterogeneous distributed hardware in section 5. We close with a short discussion and an outlook to future work in section 6.

2 Background

2.1 Load Balancing

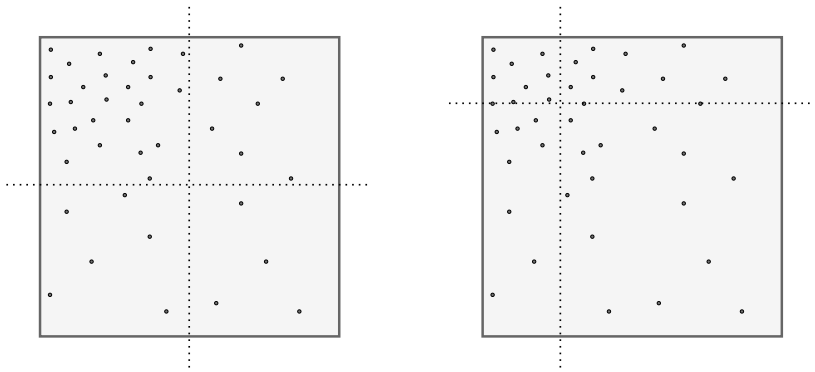
Particle simulations such as MD are often parallelised by using spatial domain decomposition. The computational domain is divided into subdomains, and each parallel process computes particle updates independently, exchanging relevant information with neighbouring subdomains at the domain boundaries. As shown in fig. 1a, a regular decomposition can be suboptimal in many cases, such as those with varied particle density throughout the domain. *Load balancing* allows to distribute the computational load more equally (as shown in fig. 1b) and can dramatically increase the performance of the simulation. Balancing can be static (ratios defined at the beginning of the simulation), which is the focus in this work, or dynamic (ratios calculated and subdomains adjusted based on minimising some predefined "load", such as runtime, or particle imbalance). Static imbalances occur from constant workload distribution, while dynamic imbalances signify time-dependent changes in the scenario [3].

Load balancing in MD has been extensively explored since the 90s [11]. A more recent review of load balancing techniques is given in [8]. Many community codes such as GROMACS [1] or LAMMPS [16] offer static and dynamic load balancing.

Load balancing in coupled simulations is relatively less explored, with a few representative examples provided in the following. Ko et al. [10] introduce a coordinated job submission API for computing clusters, so that coupled solvers may execute synchronously and independently. They introduce load balancing by changing the resources available to each coupled solver, assigning them more or fewer nodes until their execution times match. Niemöller et al [14] demonstrate a coupled CFD-CAA (Computational Aero-acoustics) simulation. They

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

implement load balancing with a partitioning approach based on space-filling curves (SFC [24]). Pour et al. [17] discuss their aero-acoustic simulation coupling three separate elements: an innermost domain where fully compressible Navier–Stokes equations are solved, surrounded by a subdomain where inviscid Euler equations are solved, and a far-field where linearized Euler equations are solved. They use the SFC-based algorithm SPartA [7] for load balancing. Besseron et al. [2] couple a discrete elements method (DEM) solver XDEM with the CFD solver OpenFOAM, using the preCICE library for coupling. Load balancing occurs by assigning MPI ranks manually between the two solvers, allowing the solvers to operate in a black-box fashion and divide the computational load internally over whatever resources it receives.



(a) Regular grid, with unequal particle distribution

(b) Irregular rectilinear grid, with a more balanced particle distribution

Fig. 1. An illustration of 2D load balancing in MD. Dotted lines indicate process boundaries, dividing the domain into four subdomains. Solid lines show global boundaries, and dots indicate particles.

2.2 AutoPas

AutoPas[4] [6,19] is a node-level particle simulation library. It supports a variety of algorithmic MD configuration options and *auto-tuning* by sampling the performance of configurations and dynamically switching between them without intervention. These configurations include particle containers (linked lists, Verlet lists, etc.), shared-memory parallel particle traversal schemes or data layouts. In this work, we determine a configuration and keep it invariant, so as to not have to consider additional variables when comparing results from our static load balancing.

[4] https://github.com/AutoPas/AutoPas

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

In our work, we use rigid single-site molecules and short-range Lennard-Jones interactions [12], and allow AutoPas to only use the linked cell container, resulting in a linear run time complexity $O(N)$ with N being the number of molecules. We allow sliced cell traversal, structure-of-arrays layout of data, and keep the optimization `newton3` enabled, which halves the number of force calculations.

2.3 ls1 mardyn

ls1 mardyn[5] is an open-source MD solver for simulating small rigid molecules at large time and length scales [15]. It focuses on thermodynamics and nanofluidics, and its features include easy extensibility using a flexible plugin system (which MaMiCo employs for coupling [9]) . Newer versions of ls1 mardyn allow the usage of AutoPas at node level, and then ls1 mardyn handles the inter-node orchestration, communication and balancing, leading to even better computational performance [19,6]. ls1 mardyn is built for HPC environments, supporting parallelization via MPI and OpenMP, as well as vectorisation. The massive scale at which ls1 mardyn is capable of running was demonstrated by Tchipev et al. in [23], when they simulated 20 trillion atoms at up to 88% weak scaling efficiency, recording up to 1.33 PFLOPS.

Additionally, ls1 mardyn supports diffusive and kd-tree based dynamic load balancing [20,19]. Seckler et al. [21] used the kd-tree based balancing method to balance load across heterogeneous hardware clusters; for systems with homogeneous particle densities, more than one rebalance was not required to find an optimal domain decomposition across heterogeneous hardware.

2.4 MaMiCo

MaMiCo[6] is an open-source software designed to couple microscopic (e.g. MD) solvers with macroscopic (e.g. CFD) solvers in a domain decomposition sense, enabling fully three-dimensional, transient molecular-continuum flow simulation on strongly coupled time scales (i.e., one CFD time step corresponds to O(100) MD time steps [13,9]). MaMiCo achieves this by defining an overlap domain within a larger CFD region (where the MD simulation is embedded) and by exchanging data between the solvers within this overlap. The exchange is facilitated by Cartesian *coupling cells*, which cover the entire coupling domain. Using these cells, MaMiCo extracts relevant data (averaged mass and momentum values from the micro solvers, and the mass flux and boundary conditions from the macro solvers), and then converts and exchanges this data between the solvers. These cells, however, need to (at minimum) cover the whole micro solver region. Hence, MaMiCo needs to be aware of the distribution of the micro region across processes in a distributed environment, to be able to map the coupling cells to linked cells of the micro solver. Since the run time of a coupled simulation is typically dominated by the micro simulation, and since more micro-level data than

[5] https://github.com/ls1mardyn/ls1-mardyn
[6] https://github.com/HSU-HPC/MaMiCo

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

macro-level data needs to be accessed and modified for the coupling, MaMiCo always runs on the processes of the micro solver and manages communication to the macro simulation processes from there, for efficiency and performance reasons. This means that if the process distribution on micro side changes, then the allocation for the MaMiCo coupling cells should also be altered correspondingly.

For (Brownian) noise reduction in the MD region, MaMiCo provides noise filters and supports parallel independent MD simulations all simulating the same region, from which the required properties are averaged out and sent to the macroscopic solver [13]. This approach, known as *ensemble averaging* or *multi-instance sampling*, helps reduce the inherent randomness of particle simulations. These independent simulations are also subdivided and run in parallel. A sample coupling setup that MaMiCo supports, relevant to the experiments in this paper, is shown in fig. 2.

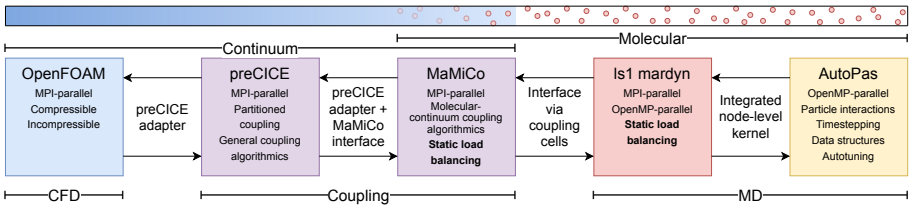


Fig. 2. A coupling setup with MaMiCo, showing our typical software stack. The OpenFOAM + preCICE + MaMiCo coupling and the MaMiCo + ls1 mardyn + AutoPas coupling have been separately validated [25,9]. New implementations in the current work are in bold.

3 Methods & Implementation

3.1 MaMiCo: Indexing System and Rectilinear Decompositions

MaMiCo's coupling strategy relies on the coupling cells as the main data structure, for interfacing with the underlying simulation libraries and for iterating over subdomains. These coupling cells form a regular grid over the 3D domain spanning the global MD volume. In practice, it is often required to iterate over a subset of coupling cells, spanning a certain volume or satisfying certain criteria. For example, one may need to iterate over only the region where mass transfer from CFD to MD occurs. Both scalar and vector indices for cells are desirable: unsigned scalar indices are better for direct element access in array-based underlying data structures, and they enable efficient and optimised iteration. Signed vector indices allow to find neighbour cells more quickly, they can be converted more easily and are needed for isolating layers and shells.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

To fulfil these criteria, a flexible and robust *indexing type system* was developed for MaMiCo. It operates on sixteen cell index types, which are instantiated at compile time using a template meta-programming based approach for faster execution times. Each type refers to a subset of coupling cells with a rectangular cuboid shape as illustrated in Fig. 3. The types arise from four boolean traits of each index type that are relevant for interpretation of indices and their conversion: locality, halo, scalarity, and direction. Locality decides if the index is local or global. Local indices are relative to the subdomain of their MPI rank, shown in green and pink in fig. 3. Global indices are shown in blue and red. The halo trait decides whether an additional ghost layer, which is used for boundary handling and data communication, should be included (blue) or excluded (red). Scalarity decides whether the index is scalar (denoted by numbers in the grid) or vector (denoted by components at the right side next to the cell grid). The data direction index trait decides whether the cells contain data to be transferred from the micro to macro solver (green and orange indices, blue area in Fig. 3), or vice-versa (red area in Fig. 3). Notable features of the indexing system include compile time type safety for cell indices, and automatic conversion between the index types.

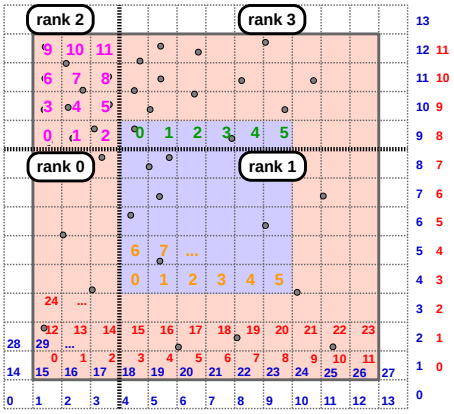


Fig. 3. Load balancing with the indexing system: different types of coupling cell indices on a 2D Cartesian grid with a generalized non-uniform domain decomposition with 4 ranks. Blue and red global indices include and exclude a ghost layer, pink are the subdomain of rank 2, green and orange are the MD-to-macro data transfer domain of MD rank 3 and 1.

Furthermore, we extended the indexing system to account for non-regular rectilinear decompositions of the MD domain, to enable load balancing. Now, we can define a static breakdown of the 3D domain into subdomains, and this breakdown is propagated to the MD solver (currently supported by ls1 mardyn) and is also used by the indexing system to define the coupling cells and index conversions in each dimension. This extension itself was easily facilitated by the flexibility of this new indexing system, which itself (in contrast) required a semi-complete overhaul of all iteration logic in the entire codebase, changing 10,000+ lines of code.

3.2 ls1 mardyn: Static Load Balancing

Although ls1 mardyn already implements diffusive and kd-tree based dynamic load-balancing [19,20], we require static load balancing as a first step towards compatibility with MaMiCo. This balancing needs to be able to split the 3D domain of particles into subdomains in a way that is predictable and reproducible,

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

so that the balancing can be propagated to every MD simulation instance within an ensemble.

This required feature was implemented by adding a new domain decomposition class to ls1 mardyn that generalises the existing regular domain decomposition, and now the user can specify a specific breakdown of the 3D domain in the x, y, and z axes. This meets the aforementioned compatibility criteria with MaMiCo's multi-instance sampling, and the rectilinear subdomains of arbitrary dimensions map seamlessly onto the coupling cells of MaMiCo as before, allowing sampling and mass + velocity transfer. We have validated that the state of the coupled particle system modelled by ls1 mardyn in a load balanced setup matches the behaviour of a non-balanced regular subdomain decomposition. This was done for several configurations of one-way coupled setups with a variety of flow scenarios, including a multi-phase droplet scenario, and the Couette flow startup [9].

4 Experiment 1: Coupled Multiphase Simulation

4.1 Introduction

Molecular-continuum coupling offers significant utility in multiphase scenarios, since different phases (such as vapour or liquid phase) can be simulated with specialised solvers that may then be coupled. This way, we can run particle simulations in specific areas of interest in a scenario, but then also conserve the effects of the overall flow field by simulating the surrounding volumes via a less resource-intensive solver. This resultant complexity reduction can also be considered as a form of load balancing, which only becomes possible in coupled simulations. However, the surface where the phase change occurs is often the most interesting part of these simulations, and hence more than one phase are often present in the higher-granularity solver's domain. Here it becomes necessary to have support for load balancing, since density imbalances severely affect the performance of particle simulations.

4.2 Scenario Description

We adapt the evaporation scenario described in Homes et al. [22]. A cuboid with vacuum on one end and a Lennard-Jones fluid on the other end are set up, and the fluid is allowed to evaporate into the vacuum. This leads to a coexistence of vapour and liquid phases at a fixed temperature [26].

In our coupled setup, we define a similar cuboid, as a domain of size $100 \times 360 \times 100$ and divide it into two subregions: an initial $100 \times 40 \times 100$ region filled with liquid acting as the reservoir, simulated using the CFD solver OpenFOAM, and the remainder (containing the liquid-vapour interface) simulated with ls1 mardyn+AutoPas. Since an OpenFOAM-preCICE coupling [5], a preCICE-MaMiCo coupling [25], and a MaMiCo-ls1 mardyn coupling [9] already exist, we are able to couple OpenFOAM and ls1 mardyn+AutoPas using MaMiCo and preCICE as a combined bridge.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

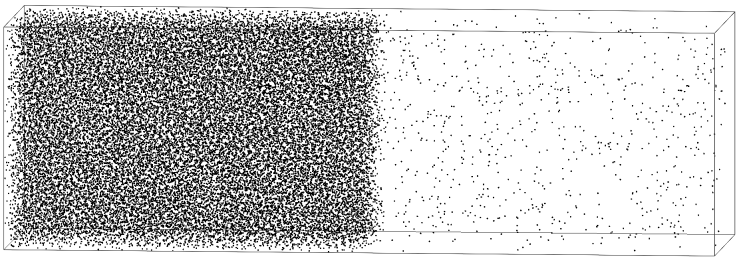


Fig. 4. The MD region of the multiphase evaporation scenario. The denser liquid phase is on the left, and the vapour phase on the right.

The particle region is evenly split into liquid ($\rho_{liq} \approx 0.7302$) and vapour ($\rho_{vap} \approx 0.0198$). The values are chosen to ensure vapour-liquid coexistence at temperature $T = 0.8$ [26]).

4.3 Experiment Setup

The MD region, cf. fig. 4, is of dimensions $100 \times 320 \times 100$. It is entirely covered by coupling cells of size 2.5^3. We run the experiments on one node of the cluster HSUper[7], containing two sockets with an Intel Xeon Platinum 8360Y processor per socket with up to 36 cores. We use 32 MPI ranks, in a $2 \times 8 \times 2$ distribution. The simulation is run for 100 coupling cycles, with 50 MD timesteps per cycle. We use only one MD instance as part of our multi-instance sampling.

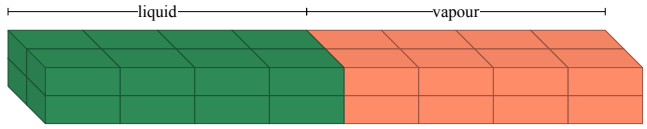
We run two experiments, one with the domain along the y axis divided equally between the liquid and vapour phases (termed the *4+4* setup) shown in fig. 5a, and one with all ranks save one simulating the liquid phase (termed the *7+1* setup) shown in fig. 5b. From a simple ratio of the maximum number of coupling cells in a rank in each phase, we predict a speedup of approximately 60% in the `7+1` case compared to the `4+4` case.

4.4 Results

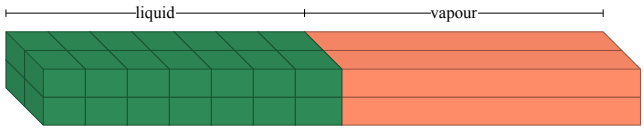
For the `4+4` case, we achieve a time of 61.11 seconds per coupling cycle. For the `7+1` case, we record 34.18 seconds per coupling cycle, implying a 44% reduction in runtime from this static load balancing. However this value is far from our initial estimate of 60%, signifying that there are overheads related to the coupling which are not yet accounted for, such as insertion/deletion of particles. Checking the simulation files, we see that each coupling step adds an average overhead of 10 seconds, however this is independent of the rank breakdown of the simulation, and thus is a constant overhead which skews the ratio of runtimes. However, the speedup obtained is still significant, and emphasises the importance of load balancing in multiphase scenarios.

[7] https://portal.hpc.hsu-hh.de/documentation/hsuper

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17



(a) Setup `4+4`



(b) Setup `7+1`

Fig. 5. Diagrams illustrating the rank breakdown of the setups. The configuration is named according to the number of ranks per phase in the y axis. Green and orange denote the liquid and vapour phase respectively.

5 Experiment 2: Handling Heterogeneous Architectures

5.1 Introduction

To demonstrate handling truly distributed molecular-continuum simulations with MaMiCo, Viot et al. in [25] successfully coupled a CFD simulation running on a local workstation with an MD simulation running on a cluster, communicating using TCP/IP. With the added support for static load balancing, we can explore another way of running a coupled simulation on a mix of hardware. This may be done for various reasons, such as hardware availability, GPU support and offloading, or energy requirements. Load balancing becomes even more important in this framework, to offset the performance difference between the different hardware. This allows users to take advantage of hardware-specific optimisations (vectorisation support, compiler optimisations, mixed CPU-GPU computing etc.) In this experiment, we demonstrate this capability by executing a coupled simulation on two different compute nodes at a time, namely an AMD node and an ARM node.

5.2 Scenario Description

We adopt a Couette flow scenario for this experiment. This scenario describes a liquid at homogeneous density trapped between two infinite surfaces with differing relative velocities. In our case, one of the two surfaces is motionless and the other slides in the $+x$ direction at a constant velocity. We set up a Couette flow, with the gap between the surfaces being 150 units. We simulate a 150^3 volume with a CFD simulation, inside which we have a 120^3 MD region. We fix the MaMiCo coupling cell size to 5^3, and we run two MD instances. Ten coupling cycles are run, each for 100 MD timesteps. We use a simple in-house Lattice-Boltzmann (LB) solver on the CFD side (running on one rank) and ls1 mardyn+AutoPas on the MD side. The fluid density is ≈ 0.813.

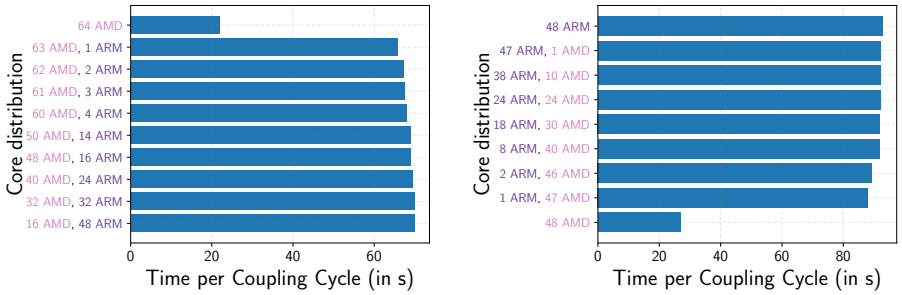
ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

We use one AMD EPYC 7763 as our AMD node, with 64 hardware cores, and one Fujitsu A64FX as our ARM node, with 48 cores. The nodes are connected via InfiniBand EDR (100Gbit/s).

5.3 Rank Migration Test

Setup We start with a setup where the simulation fully saturates one AMD node (64 ranks). Then, we migrate ranks one by one onto the ARM node, and observe the effect that it has on the simulation speed. We divide the domain into $4 \times 4 \times 4$ subdomains along the axes, giving us 6^3 coupling cells per rank. We also conduct the experiment in the other direction i.e. by migrating ranks from a fully saturated ARM node to the AMD node. In this case, since we only have 48 ranks, the domain breakdown is $4 \times 4 \times 3$ along the xyz axes. In both cases, the experiments are run thrice, and the average value is reported.

From an earlier report utilising ls1 mardyn+AutoPas on the same hardware [18], we expect the simulation to run significantly slower on ARM.



(a) 64 ranks: AMD → ARM (b) 48 ranks: ARM → AMD

Fig. 6. Chosen results (simulation speed) from migration experiments

Results Selected results of the AMD→ARM migration experiment are shown in fig. 6a, and results of the ARM→AMD migration experiment are shown in fig. 6b. The performance is severely affected as soon as a single subdomain is migrated to ARM, and it remains relatively stable afterwards, only worsening marginally with the migration of more ranks. As more ranks move to ARM, more subdomains are dependent on their ARM neighbours to finish their computations each MD timestep, hence the marginal increase in walltime can be explained by compounding delays due to synchronisation.

Another conclusion drawn from fig. 6b is that communication overhead between the ARM and AMD node is negligible, since there is an immediate improvement from migrating a single rank from ARM to AMD, and no additional (network-related) walltime increases can be observed.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

Since the scenario is of homogeneous density, we can determine the expected runtime per coupling cell from these results. As each rank has $6^3 = 216$ coupling cells in the case with 64 ranks, we find that the time per coupling cell per cycle for AMD $t_{\text{AMD,cell}} \approx 0.102$ s, and similarly for ARM we have $t_{\text{ARM,cell}} \approx 0.305$ s. We notice that

$$t_{\text{ARM,cell}} \approx 3 \times t_{\text{AMD,cell}} \tag{1}$$

Denoting the predicted time per coupling cycle as t_{pred}, we have

$$\begin{aligned} t_{\text{pred}} &= \max\left(t_{\text{AMD,pred}}, t_{\text{ARM,pred}}\right) \\ &= \max\left(t_{\text{AMD,cell}} \times n_{\text{AMD}}, t_{\text{ARM,cell}} \times n_{\text{ARM}}\right) \\ &= \max\left(0.102 n_{\text{AMD}}, 0.305 n_{\text{ARM}}\right) \end{aligned} \tag{2}$$

where $n_{\text{AMD}}, n_{\text{ARM}}$ are the maximum number of coupling cells in a rank on AMD and ARM respectively. We use the value of t_{pred} to perform optimal load balancing by adjusting the values of n and verify our configurations experimentally. The constants $t_{\text{AMD,cell}}, t_{\text{ARM,cell}}$ account for two MD instances with 100 MD timesteps per instance, and hence are not generalisable beyond this context unless those factors are accounted for.

As we only use the 64 rank case in future experiments, we omit the calculations to derive time values for the 48 rank case here.

5.4 Balancing Tests

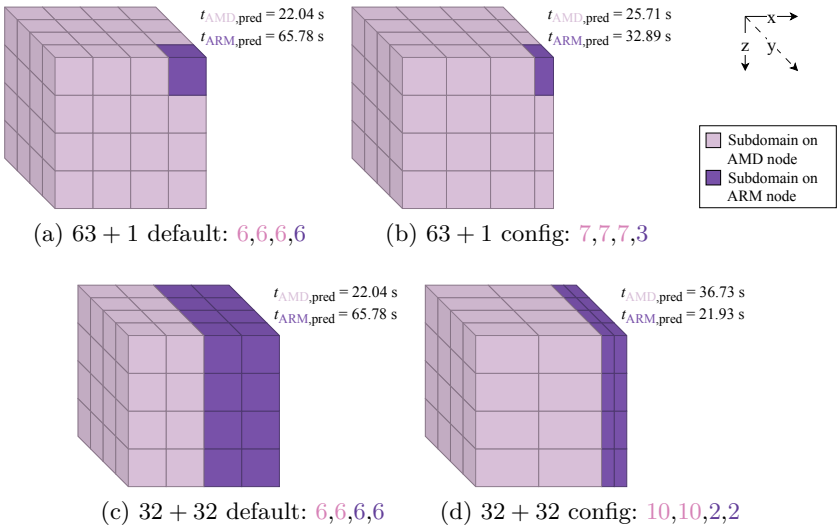
With the reference values in mind, we set up two load balancing experiments, using 64 ranks: one where only one rank (and thus only one subdomain) of the MD simulation resides on ARM and the remaining 63 ranks run on AMD (termed *63+1* setup), and one where the domain (and therefore the 64 ranks) are evenly split among the nodes (termed *32+32* setup). The balancing is done keeping eq. (1) in mind, and only along the x axis for simplicity.

Setup We name our configurations in this section using the number of coupling cells in the x direction. Since we have 4 ranks across each axis, all configurations are four comma-separated integers. In figures and diagrams, cells running on AMD are denoted in thistle, and cells running on ARM are denoted in royal purple. Thus, the default, evenly distributed configuration for the `32+32` setup is `6,6,6,6` (6 coupling cells along x-direction for fixed y and z per rank).

For the `63+1` setup, we choose four configurations. The default configuration is shown in fig. 7a. Firstly, from eq. (1), we allocate $n_{AMD} = 3 \times n_{ARM}$ with config. `6,6,9,3`. Next, we even out the load on the AMD side with config. `7,7,7,3`, illustrated in fig. 7b. Finally, we attempt to create a bottleneck on the AMD side instead, with config. `7,7,8,2` and config. `8,8,7,1`.

Then, for the `32+32` setup, we choose four configurations again, with the default `6,6,6,6` shown in fig. 7b. Once again, we allocate $n_{AMD} = 3 \times n_{ARM}$ with config. `9,9,3,3`, and try to create a bottleneck on the AMD side with config. `10,10,2,2`, shown in fig. 7d, and also with a config. `11,11,1,1`. We include an

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17



(a) $63 + 1$ default: 6,6,6,6 (b) $63 + 1$ config: 7,7,7,3

(c) $32 + 32$ default: 6,6,6,6 (d) $32 + 32$ config: 10,10,2,2

Fig. 7. Diagrams illustrating the rank breakdown of the setups. The configuration is named counting the allocation of cells along the x axis.

additional config. `8,8,4,4` to illustrate a gradual improvement over the default setup.

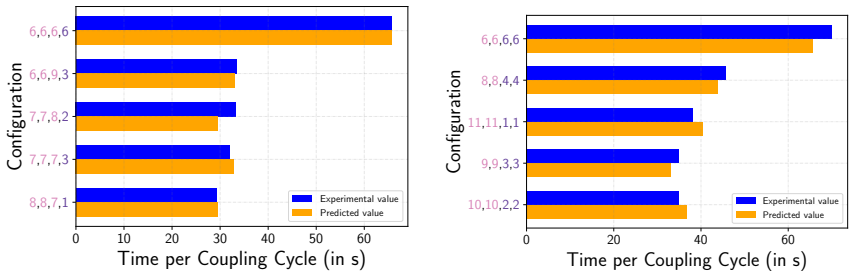
Results In fig. 8a we show the results of the experiments involving the `63+1` setup, and in fig. 8b we show the same for the `32+32` setup. We see that proper balancing halves the time required per coupling cycle, as the bottleneck created by the ARM node is somewhat alleviated. For the `63+1` setup, the time per coupling cycle is reduced from 65.78 s to 29.18 s (≈ 55.6% reduction of runtime) in the best case, and similarly, in the `32+32` setup we see an improvement from 70.12 s to 34.94 s (≈ 50.2% reduction of runtime).

We see that proper balancing halves the time required per coupling cycle, as the bottleneck created by the ARM node is somewhat alleviated. For the `63+1` setup, the time per coupling cycle is reduced from 65.78 s to 29.18 s (≈ 55.6% reduction of runtime) in the best case, and similarly, in the `32+32` setup we see an improvement from 70.12 s to 34.94 s (≈ 50.2% reduction of runtime).

Metrics reported by ls1 mardyn per simulation show that this improvement can be attributed in a large part to the reduction of the runtime per coupling cycle in the force calculation step of the simulation (≈55.07% reduction from 11.214 s to 5.038 s in the `63+1` setup, ≈46.71% reduction from 11.258 s to 6.002 s in the `32+32` setup). A deeper performance analysis may yield further insight.

The experimental results lie within a 5.7% margin of our predictions, with the notable outlier being the `7,7,8,2` configuration in the `64+1` setup, with a discrepancy of −11.76%. We can conclude that it is still not ideal to run the

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17



(a) Results: $63 + 1$ setup (b) Results: $32 + 32$ setup

Fig. 8. Simulation speed results from load balancing

simulation on mixed hardware without further hardware-specific optimisation, but proper load balancing successfully reduces the overall runtime.

6 Conclusion

From our experiments, we have verified the importance of load balancing in coupled scenarios. We have implemented a static load balancer for ls1 mardyn, and have extended MaMiCo to be able to interface with the resultant rectangular rectilinear grids. We have tested our implementation with both a heterogeneous phase and heterogeneous hardware setup, both of which are highly relevant to coupled simulations. The results are favourable, leading to a 44% to 55% reduction of runtime across the fastest configurations. These results provide ample motivation for further work in this area.

The first goal for future work should be the automatic selection of an optimal configuration of static load balancing at the beginning of the simulation. This could be done with expert knowledge, or from estimates (from previous work, or from automatic mini-benchmarks). Then MaMiCo should be further extended to support dynamic load balancing, driven by the MD simulation; it should be able to adapt to changing subdomain sizes without further interference from the user. Finally, the balancing support should be extended towards other particle dynamics solvers, such as LAMMPS or GROMACS.

Acknowledgments. A. Das Sharma and L. Viot acknowledge financial support by the projects MaST and hpc.bw. Computational resources (HPC cluster HSUper, ARM Minicluster, AMD Minicluster) have been provided by the project hpc.bw. MaST and hpc.bw have been funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr; dtec.bw is funded by the European Union – NextGenerationEU. The authors further acknowledge the provision of computational resources for more numerical studies at HLRS (project GCS-MDDC). They thank Simon Homes and Prof. Jadran Vrabec from TU Berlin for their fruitful correspondence and invaluable support regarding the implementation of the evaporation scenario.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Berendsen, H., van der Spoel, D., van Drunen, R.: Gromacs: A message-passing parallel molecular dynamics implementation. Computer Physics Communications **91**(1), 43–56 (1995)
2. Besseron, X., Adhav, P., Peters, B.: Parallel multi-physics coupled simulation of a midrex blast furnace. In: Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region Workshops. p. 87–98. HPCAsia '24 Workshops, Association for Computing Machinery, New York, NY, USA (2024)
3. Böhme, D., Wolf, F., Geimer, M.: Characterizing load and communication imbalance in large-scale parallel applications. In: 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum. pp. 2538–2541. IEEE (2012)
4. Chourdakis, G., Davis, K., Rodenberg, B., Schulte, M., Simonis, F., Uekermann, B., Abrams, G., Bungartz, H., Cheung Yau, L., Desai, I., Eder, K., Hertrich, R., Lindner, F., Rusch, A., Sashko, D., Schneider, D., Totounferoush, A., Volland, D., Vollmer, P., Koseomur, O.: preCICE v2: A sustainable and user-friendly coupling library [version 2; peer review: 2 approved]. Open Research Europe **2**(51) (2022)
5. Chourdakis, G., Schneider, D., Uekermann, B.: Openfoam-precice: Coupling openfoam with external solvers for multi-physics simulations. OpenFOAM® Journal **3**, 1–25 (Feb 2023)
6. Gratl, F.A., Seckler, S., Tchipev, N., Bungartz, H.J., Neumann, P.: Autopas: Auto-tuning for particle simulations. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 748–757 (2019)
7. Harlacher, D.F., Klimach, H., Roller, S., Siebert, C., Wolf, F.: Dynamic load balancing for unstructured meshes on space-filling curves. In: 2012 IEEE 26th international parallel and distributed processing symposium workshops & PhD forum. pp. 1661–1669. IEEE (2012)
8. Hirschmann, S., Pflüger, D., Glass, C.W.: Towards understanding optimal load-balancing of heterogeneous short-range molecular dynamics. In: 2016 IEEE 23rd International Conference on High Performance Computing Workshops (HiPCW). pp. 130–141 (2016)
9. Jarmatz, P., Wittenberg, H., Jafari, V., Das Sharma, A., Maurer, F., Wittmer, N., Neumann, P.: MaMiCo 2.0: An enhanced open-source framework for high-performance molecular-continuum flow simulation. SoftwareX **20**, 101251 (Dec 2022)
10. Ko, S.H., Kim, N., Kim, J., Thota, A., Jha, S.: Efficient runtime environment for coupled multi-physics simulations: Dynamic resource allocation and load-balancing. In: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. pp. 349–358 (2010)
11. Kohring, G.: Dynamic load balancing for parallelized particle simulations on MIMD computers. Parallel Computing **21**(4), 683–693 (1995)
12. Lennard-Jones, J.E.: Cohesion. Proceedings of the Physical Society **43**(5), 461 (sep 1931)
13. Neumann, P., Flohr, H., Arora, R., Jarmatz, P., Tchipev, N., Bungartz, H.J.: MaMiCo: Software design for parallel molecular-continuum flow simulations. Computer Physics Communications **200**, 324–335 (Mar 2016)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17

14. Niemöller, A., Schlottke-Lakemper, M., Meinke, M., Schröder, W.: Dynamic load balancing for direct-coupled multiphysics simulations. Computers & Fluids **199**, 104437 (2020)
15. Niethammer, C., Becker, S., Bernreuther, M., Buchholz, M., Eckhardt, W., Heinecke, A., Werth, S., Bungartz, H.J., Glass, C.W., Hasse, H., Vrabec, J., Horsch, M.: ls1 mardyn: The massively parallel molecular dynamics code for large systems. Journal of Chemical Theory and Computation **10**(10), 4455–4464 (2014)
16. Plimpton, S.: Fast parallel algorithms for short-range molecular dynamics. Journal of Computational Physics **117**(1), 1–19 (1995)
17. Pour, N.E., Krupp, V., Klimach, H., Roller, S.: Load balancing for immersed boundaries in coupled simulations. In: Resch, M.M., Kovalenko, Y., Bez, W., Focht, E., Kobayashi, H. (eds.) Sustained Simulation Performance 2018 and 2019. pp. 185–201. Springer International Publishing, Cham (2020)
18. Preuß, H., De Nayer, G., Das Sharma, A., Jarmatz, P., Leinen, W.G., Jafari, V., Horn, R., Breuer, M., Fink, A., Neumann, P.: hpc.bw benchmark report 2022–2024 (2024)
19. Seckler, S., Gratl, F., Heinen, M., Vrabec, J., Bungartz, H.J., Neumann, P.: Autopas in ls1 mardyn: Massively parallel particle simulations with node-level autotuning. Journal of Computational Science **50**, 101296 (2021)
20. Seckler, S., Gratl, F., Tchipev, N., Heinen, M., Vrabec, J., Bungartz, H.J., Neumann, P.: Load balancing and auto-tuning for heterogeneous particle systems using ls1 mardyn. In: Nagel, W.E., Kröner, D.H., Resch, M.M. (eds.) High Performance Computing in Science and Engineering '19. pp. 523–536. Springer International Publishing, Cham (2021)
21. Seckler, S., Tchipev, N., Bungartz, H.J., Neumann, P.: Load balancing for molecular dynamics simulations on heterogeneous architectures. In: 2016 IEEE 23rd International Conference on High Performance Computing (HiPC). pp. 101–110 (2016)
22. Simon Homes, Matthias Heinen, J.V., Fischer, J.: Evaporation driven by conductive heat transport. Molecular Physics **119**(15-16), e1836410 (2021)
23. Tchipev, N., Seckler, S., Heinen, M., Vrabec, J., Gratl, F., Horsch, M., Bernreuther, M., Glass, C.W., Niethammer, C., Hammer, N., Krischok, B., Resch, M., Kranzlmüller, D., Hasse, H., Bungartz, H.J., Neumann, P.: Twetris: Twenty trillion-atom simulation. The International Journal of High Performance Computing Applications **33**(5), 838–854 (2019)
24. Teresco, J.D., Devine, K.D., Flaherty, J.E.: Partitioning and dynamic load balancing for the numerical solution of partial differential equations. In: Numerical solution of partial differential equations on parallel computers. pp. 55–88. Springer (2006)
25. Viot, L., Piel, Y., Neumann, P.: From desktop to supercomputer: Computational fluid dynamics augmented by molecular dynamics using mamico and precice. In: Bienz, A., Weiland, M., Baboulin, M., Kruse, C. (eds.) High Performance Computing. pp. 567–576. Springer Nature Switzerland, Cham (2023)
26. Vrabec, J., Kedia, G.K., Fuchs, G., Hasse, H.: Comprehensive study of the vapour–liquid coexistence of the truncated and shifted lennard–jones fluid including planar and spherical interface properties. Molecular physics **104**(09), 1509–1527 (2006)
27. Weller, H.G., Tabor, G., Jasak, H., Fureby, C.: A tensorial approach to computational continuum mechanics using object-oriented techniques. Computers in Physics **12**(6), 620–631 (Nov 1998)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97632-2_17