

ESPresSo: Progress towards Exascale

MultiXscale Deliverable 2.7
Deliverable Type: Report
Delivered in June, 2025

MultiXscale
EuroHPC Centre of Excellence for
Multiscale Modelling



**Co-funded by
the European Union**



EuroHPC
Joint Undertaking

Acknowledgement

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) under grant agreement No 101093169.

Disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European High Performance Computing Joint Undertaking (JU). Neither the European Union nor the granting authority can be held responsible for them.

Project and Deliverable Information

Project Title Project Ref. Project Website EuroHPC Project Officer	MultiXscale: EuroHPC Centre of Excellence for Multiscale Modelling Grant Agreement 101093169 https://www.multixscale.eu Matteo Mascagni
Deliverable ID Deliverable Nature Dissemination Level Contractual Date of Delivery Actual Date of Delivery	D2.7 Report Public Project Month 30 (30 th June, 2025) 27 th June, 2025
Description of Deliverable	Progress report on the scalability of ESPResSo with a focus on short-range interactions, electrostatics and hydrodynamics using lattice-Boltzmann.

Document Control Information

Document	Title:	ESPResSo: Progress towards Exascale
	ID:	D2.7
	Version:	As of June, 2025
	Status:	Accepted by Steering Committee
	Available at:	https://www.multixscale.eu/deliverables
Review	Document history:	Internal Project Management Link
	Review Status:	Reviewed
Authorship	Written by:	Jean-Noël Grad (USTUTT)
	Contributors:	Rudolf Weeber (USTUTT)
	Reviewed by:	Alan O'Cais (UB)
	Approved by:	Alan O'Cais (UB)

Document Keywords

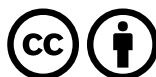
Keywords:	MultiXscale, HPC
-----------	------------------

27th June, 2025

Disclaimer: This deliverable has been prepared by the responsible Work Package of the Project in accordance with the Consortium Agreement and the Grant Agreement. It solely reflects the opinion of the parties to such agreements on a collective basis in the context of the Project and to the extent foreseen in such agreements.

Copyright notices: This deliverable was co-ordinated by Jean-Noël Grad¹ (USTUTT) on behalf of the MultiXscale consortium with contributions from Rudolf Weeber (USTUTT). This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit:

<https://creativecommons.org/licenses/by/4.0>



¹jgrad@icp.uni-stuttgart.de

Contents

Executive Summary	1
1 Introduction	2
1.1 Target audience	2
1.2 Deliverable outline	2
1.3 Partner contributions	2
2 Usage Scenario and Goals	3
3 Improving Scalability	4
3.1 Short-ranged forces	4
3.1.1 Improved memory layout	4
3.1.2 Avoiding synchronization barriers	4
3.2 Electrostatics	6
3.3 Multi-GPU support for lattice-Boltzmann	6
3.4 Shared-memory parallelism	6
3.5 Parallel I/O	8
4 Conclusion and Outlook	9
Acknowledgements	10
References	10

List of Figures

1	Difference between SoA and AoS memory layouts	4
2	Ghost force reduction in the cell list algorithm	5
3	Parallel efficiency of Lennard-Jones without ghost force reduction (weak scaling)	5
4	Parallel efficiency of multi-GPU lattice-Boltzmann (weak scaling)	6
5	Parallel performance of lattice-Boltzmann on CPU (strong scaling)	7
6	Parallel efficiency of lattice-Boltzmann on GPU with particle coupling (weak scaling)	7
7	Serial write performance of HDF5/MPI-IO on ext4 (strong scaling)	8

Executive Summary

Task 2.2 of the MultiXscale CoE aims to make the open-source molecular-dynamics code ESPResSo “pre-exascale-ready”, so that typical mesoscopic simulations (10^5 to 10^6 particles) run efficiently on 500 or more CPU cores and multiple GPUs with fewer particles per core than today. The main goal is to improve time to solutions for large ensembles of mid-sized simulations, while also allowing for occasional large simulations. The following areas are addressed:

- Short-range forces
 - We re-implemented the force kernel with Cabana/Kokkos, changing particle data from array of structures (AoS) to structure of arrays (SoA) to improve cache re-use, making use of shared memory parallelism, and paving the way for future portable GPU off-loading.
 - Moreover, we added an optional “no-ghost-force-reduction” mode that skips one MPI reduction per simulation step. On workstations with a high clock rate, it breaks even for eight cores for 100–500 particles per core. On HPC Vega it yields a small speed-up starting at 128 cores.
- Electrostatics.
 - Here, we integrated the heFFTe fast Fourier transform (FFT) backend into the P^3M solver, bringing thread-parallel and distributed 3D FFTs and a path to multi-GPU support
- Hydrodynamics.
 - We replaced the bespoke lattice-Boltzmann solver from previous ESPResSo versions with a solver based on waLBerla/PyStencils/LbmPy, unlocking multi-GPU execution that retains 75% weak-scaling efficiency on multiple GPUs.
- Shared-memory parallelism.
 - We are adding OpenMP/Kokkos/Cabana shared-memory parallelism throughout ESPResSo, so that one MPI rank can drive an entire GPU while all host cores remain busy.
 - Shared-memory parallelism is provided by Kokkos for the integration of equations of motion, by Cabana for short-ranged force calculations, by OpenMP for CPU lattice-Boltzmann via waLBerla, and the FFTs in the P^3M solver via heFFTe.
- Parallel I/O.
 - A rewritten HDF5 backend improves write bandwidth by over an order of magnitude on a 16-core workstation and also performs well on larger simulations on parallel file systems, enabling tractable output from large runs

In summary, work was undertaken in all major areas concerning the scalability of the software. Our goals with regards to the scalability of simulations involving hydrodynamics and parallel I/O have been achieved. For short-ranged force calculation and electrostatics, the implementations based on Cabana and heFFTe have been developed and shown to work correctly. However, here, further optimization work is ongoing.

1 Introduction

Extensible Simulation Package for Research on Soft Matter Systems (ESPResSo) is a simulation package mainly focused on coarse-grained models. To address research in soft matter, statistical and biological physics as well as process engineering, it provides both particle- and lattice-based approaches. Algorithms include molecular dynamics, Monte Carlo methods, lattice-Boltzmann hydrodynamics, and electrokinetics. ESPResSo consists of an MPI-parallel simulation core written in C++, but is controlled via a Python interface. This makes the software particularly accessible to domain scientists without extensive programming experience, while allowing for highly customised simulation protocols. As part of the MultiXscale Center of Excellence, we are preparing ESPResSo for use on large High Performance Computing (HPC) systems, covering topics like scalability, multi-GPU support, and parallel I/O. In this report, we outline the progress made in this regard.

1.1 Target audience

This deliverable is aimed at (but not limited) to:

- people employing molecular dynamics (MD) simulations,
- people using ESPResSo MD simulation code,
- people who want to test scaling of ESPResSo.

1.2 Deliverable outline

Section 2 covers the expected usage scenario for ESPResSo when it comes to consuming exa-scale level computing resources. Section 3 outlines the specific actions that have been taken to ensure that ESPResSo is well prepared for this usage scenario.

1.3 Partner contributions

USTUTT contributed as planned to the work in this deliverable.

2 Usage Scenario and Goals

As ESPResSo [1,2] is targeted at coarse-grained models, typical simulations contain fewer particles than in an atomistic simulation. At the same time, when modeling at the mesoscale, a solvent such as water is typically not modeled explicitly via particles but by coupling the simulation to a hydrodynamics solver—in the case of ESPResSo, the lattice-Boltzmann method [3].

On the mesoscale, Brownian motion is one of the main factors influencing a system's behavior. This implies that the simulations contain inherent randomness. Therefore, observations are made by averaging over several simulations with the same parameters.

In summary, a typical usage scenario would be a large number of simulations with approximately 100 000 to 1 000 000 particles interacting via short-ranged and electrostatic interactions, coupled to a lattice-Boltzmann solver. Our goal is to provide excellent time to solution for these simulations, but also allow for larger simulations, where needed.

3 Improving Scalability

Each component of the simulation, short-ranged forces, electrostatics, and the lattice-Boltzmann hydrodynamics solver, has individual computation and communication patterns, and thus, its scalability has to be addressed separately.

3.1 Short-ranged forces

Short-ranged forces, such as the Lennard-Jones interaction, but also short-ranged contributions of electrostatic forces, make up the bulk of the computational effort. **ESPResSo** uses both a linked-cell algorithm and neighbour lists to efficiently discover pairs of particles for which interactions have to be calculated. However, the implementation in **ESPResSo** 4.2 present at the start of the MultiXscale CoE is not optimal in terms of scalability and performance. The two main issues with regards to short-ranged forces are: first, a memory layout for particles that does not use the CPU cache efficiently; and second, a communication pattern involving potentially avoidable synchronization points.

3.1.1 Improved memory layout

ESPResSo uses an array of structures (AoS) memory layout for the particles, i.e., all attributes of a particle (position, mass, velocity, etc.) are stored contiguously. However, most of these properties are not needed for short-ranged force calculation, so most of the memory loaded into the cache when accessing a particle's position is never re-used. Fig. 1 depicts a more cache-friendly memory layout, structure of arrays (SoA). While it is not feasible to switch the memory layout everywhere in the code, we have done so for short-ranged force calculation. To this end, we implemented short-ranged force calculation using the Cabana library [4]. Backed by Kokkos [5], Cabana provides performance-portable infrastructure for particle-based methods, in particular a cache-friendly layout for storing particle data and neighbour lists. The implementation of Cabana-based short-ranged force calculation into **ESPResSo** turned out to be challenging. The main reason is that ghost particles are handled differently in Cabana and in **ESPResSo**'s original implementation. As a result, we cannot rely on Cabana's implementation for constructing the neighbour lists. At this point, the Cabana-based force calculation works for the entire set of **ESPResSo** features, but does not yet have good scaling efficiency on large core numbers. This is mainly caused by remaining serial code and by memory locality issues in the Verlet list construction and conversion of the data layout. For example, according to profiling done with Caliper, preparing the particle data in SoA format takes around 8% of simulation time on one thread but close to 60% of the runtime for 16 threads, at 5000 particles per thread. We expect scalability to become adequate once this code is optimised.

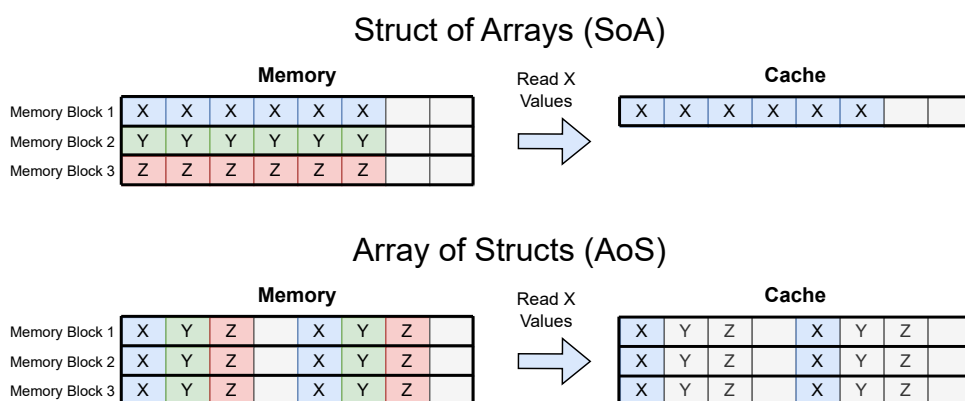


Figure 1: Common memory layouts for particle data. Top: the SoA memory layout stores the x-, y- and z-coordinates of a collection of 6 particles in three arrays (blue, green, and red bytes); when an algorithm needs the x-position of the particles, a single cache line fits all the data. Bottom: the AoS memory layout stores each particle coordinates in a separate structure, which leads to data striping (data for different particle properties are interleaved) and possibly extra padding (white memory cells between structures); when an algorithm needs the x-coordinate of the particles, three cache lines are needed. Image credits: Paul Hohenberger.

3.1.2 Avoiding synchronization barriers

Molecular dynamics with short-ranged forces typically involves two rounds of communications per time step. Before forces are calculated, particles close to the domain boundary of an MPI rank are communicated into a halo layer on the neighbouring MPI rank. Second, after forces are calculated, the forces accumulated on particles in the halo layers

are communicated back to the particles from which they are derived. Each of these communications introduces a synchronization barrier. This is particularly problematic in simulations with a large number of MPI ranks, as the number of particles on each rank fluctuates and all ranks have to wait for the rank that takes longest.

For some simulations, the communication of forces on particles in halo layers can be avoided by recomputing them on the MPI rank from which the halo was derived (Fig. 2), i.e., one purposefully does not make use of Newton's third law for these particles. In effect, delays due to communication are avoided at the expense of more effort in the short-range force calculation.

We have implemented this procedure in `ESPResSo` for systems containing only short-ranged forces and electrostatics. Initial analysis shows that avoiding the communications is only worth it for quite low particle numbers per MPI rank (≈ 100), as otherwise, the short-range force calculation time is significantly larger than the communication time. The break-even for the new communication scheme is hardware-dependent. On a workstation with a high CPU clock rate, it is found for approximately eight MPI ranks for a system with 100–500 particles per core. On the HPC Vega supercomputer, there is a marginal performance improvement starting at 128 cores (Fig. 3), which can be explained by the overhead of interconnect message passing (starting after 64 cores) and inter-node communication (starting after 128 cores). The break-even is expected to shift towards lower core numbers, once the optimised memory layout for short-ranged force calculation is used, and particularly, if short-ranged force calculation can be off-loaded to a GPU.

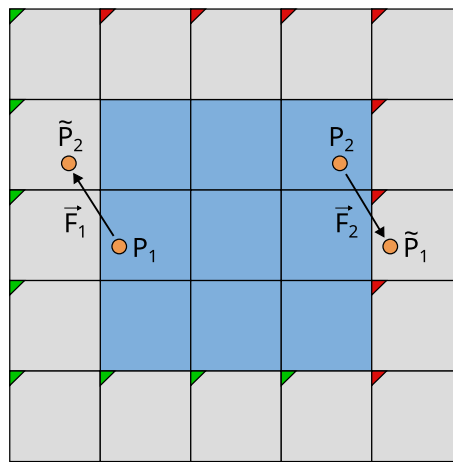


Figure 2: A periodic 2D system with a 3×3 domain (blue tiles) and a halo layer (gray tiles with red green markers). Two particles P_1 and P_2 interact with each other across the periodic boundaries through their ghost images \tilde{P}_1 and \tilde{P}_2 in the halo layer, imparting a force \vec{F}_1 on P_1 and \vec{F}_2 on P_2 . With the current implementation, `ESPResSo` calculates the force \vec{F}_2 on particle P_2 , but not the force \vec{F}_1 on particle P_1 , instead the value $-\vec{F}_2$ is stored on ghost particle \tilde{P}_1 , to be communicated later to P_1 (ghost force reduction). Forces are only calculated for particles in a domain cell (blue tile) interacting with a particle in either a domain cell or red halo cell (gray tile with red marker). To remove the reduction operation, the force \vec{F}_1 is calculated on particle P_1 directly, and ghost forces are no longer needed.

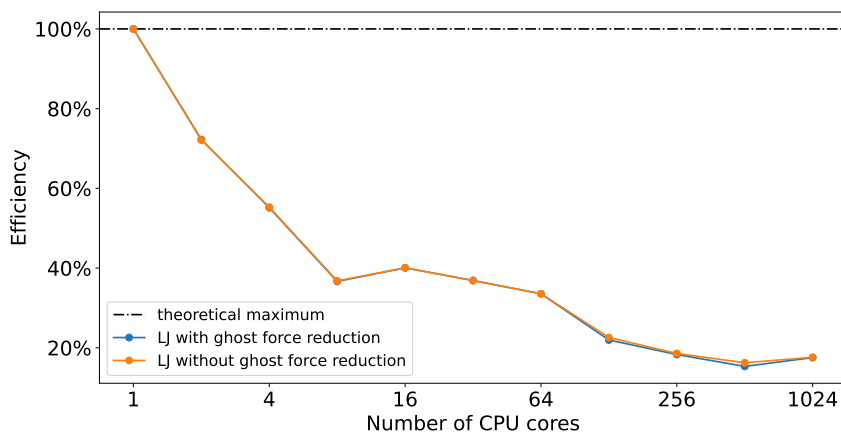


Figure 3: Weak scaling efficiency of a Lennard-Jones simulation with and without ghost force reduction on HPC Vega (two 64-core AMD EPYC Rome 7H12 CPUs per node). The simulated system contains 100 particles per core.

3.2 Electrostatics

ESPResSo uses the P^3M algorithm for electrostatics. The main idea is to split electrostatics into a short-ranged part, calculated together with other short-ranged forces such as Lennard-Jones, and a long-ranged part calculated in Fourier space. Therefore, 3D Fourier transforms are a key ingredient of the method. These scale poorly when using distributed memory parallelism, as the data has to be redistributed across the MPI ranks several times. To address this, we implemented a version of P^3M based on the `heFFTe` library for 3D FFTs [6]. It supports both shared- and distributed-memory parallelism on the CPU as well as multi-GPU FFTs. At this point, we only make use of CPUs, a GPU version will follow.

Once available, we will also interface ESPResSo to the performance-portable electrostatics library developed by our partners at FZJ, potentially allowing us to concentrate the Fourier transforms on a single GPU, while the other resources perform different parts of the simulation.

3.3 Multi-GPU support for lattice-Boltzmann

ESPResSo 4.2 provides a custom GPU implementation of the lattice-Boltzmann method. It is, however, limited to a single GPU, inherently limiting the scalability of such a simulation. As part of WP3, we replaced ESPResSo's custom lattice-Boltzmann implementation by one provided by `waLBerla`. The `waLBerla` framework, along with its companion packages `PyStencils` and `LbmPy`, provides solvers for stencil methods such as lattice-Boltzmann, and has been demonstrated to scale well on HPC systems [7]. In particular, the transition to `waLBerla` for lattice-Boltzmann allowed us to provide multi-GPU support (Fig. 4). A technical description of the implementation and molecular dynamics to lattice-Boltzmann coupling can be found in Deliverable 3.3.

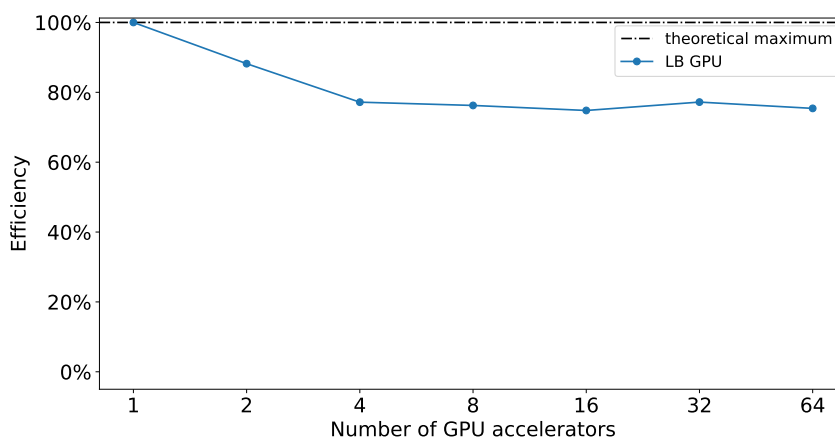


Figure 4: Weak scaling efficiency of a multi-GPU unthermalised lattice-Boltzmann fluid simulation with particle coupling on HPC Vega (two 64-core AMD EPYC Rome 7H12 CPUs and four NVIDIA A100 GPUs per node). The simulated system contains 10k particles and approx. 4 million cells per GPU on a 158x158x158 grid for a volume fraction of 1%.

3.4 Shared-memory parallelism

Using one or multiple GPUs also has consequences for the CPU part of the simulation. In particular, GPU performance drops drastically if more than one MPI rank interacts with the same GPU. Typically, HPC nodes provide 16 to 32 processor cores per GPU. Therefore, to make efficient use of the CPU, shared-memory parallelism has to be employed. The scalability improvement compared to the original MPI-based implementation in ESPResSo 4.2.2 is shown in Fig. 6. Moreover, using shared-memory parallelism on the HPC node level mitigates some of the performance penalties incurred by the synchronization barriers due to message passing.

We are in the process of implementing shared-memory parallelism throughout ESPResSo, using Kokkos, Cabana and OpenMP. For short-ranged force calculation, shared-memory parallelism is provided by Cabana; for electrostatics, by the `heFFTe` library via OpenMP; and for lattice-Boltzmann, when run on the CPU, by `waLBerla` via OpenMP (Fig. 5). As mentioned in section 3.1.1, the shared memory implementations are working, but further performance and scalability engineering is needed.

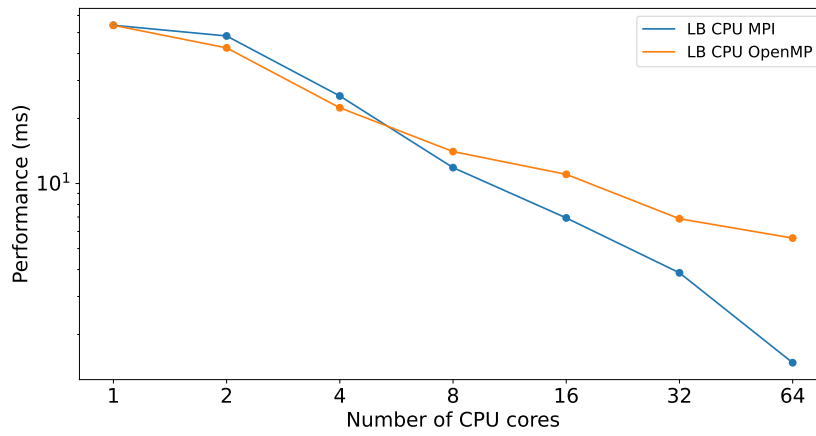


Figure 5: Strong scaling performance (time per simulation step) of lattice-Boltzmann, using MPI communication and OpenMP shared-memory parallelism, on the Ant cluster (two 32-core AMD EPYC 9374F CPUs per node). For low core numbers, OpenMP is faster, due to reduced need for communication. However, at high core numbers, remaining serial code limits the performance. The simulated system contains approx. 2 million cells on a 128x128x128 grid.

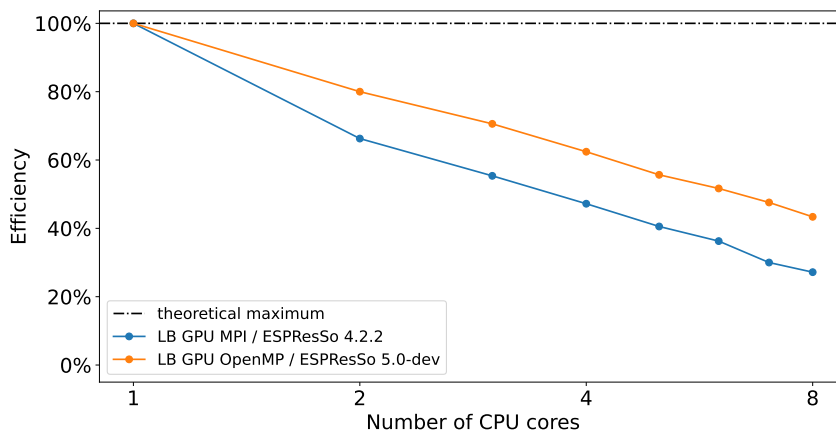


Figure 6: Weak scaling efficiency of lattice-Boltzmann with particle coupling at 3% volume fraction, using the original MPI implementation from ESPResSo 4.2.2 and the new OpenMP shared-memory parallelism implementation from ESPResSo 5.0-dev, on a desktop workstation equipped with an Intel i7-12700K CPU and an NVIDIA GeForce RTX 3070 Ti GPU. The simulated system contains 1k particles and 27k GPU lattice-Boltzmann cells on a 30x30x30 grid per core. The shared-memory parallelized version shows improved parallel performance, as the MPI gather and scatter operations collecting and spreading the particle data to/from the MPI rank interacting with the GPU are not needed.

3.5 Parallel I/O

Lastly the scalability of writing the state of the simulation in the **HDF5** [8] format has been improved drastically (Fig. 7). This makes it possible to use the trajectories from large simulation as input for other packages. A technical description and benchmarks on parallel file systems can be found in Deliverable 3.3.

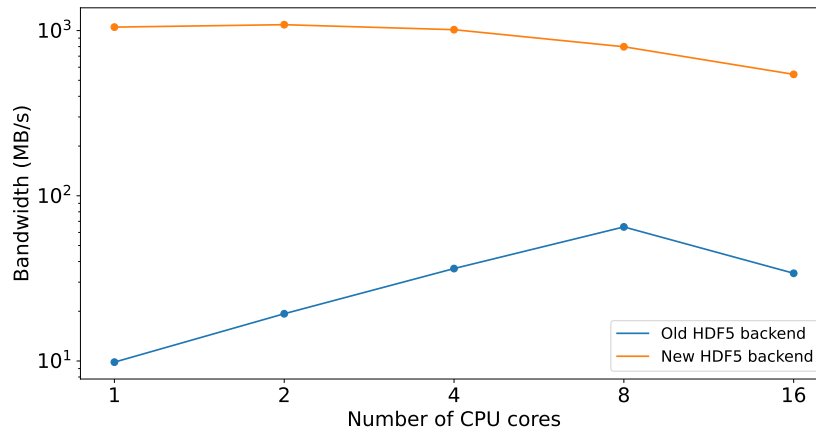


Figure 7: Strong scaling performance of particle-based writers (**HDF5** [8] and **MPI-IO** [9]) on an **ext4** [10] file system (2800 MB/s serial write rate) on a desktop workstation equipped with a 16-core AMD Ryzen 9 CPU. The simulated system contains 16k particles.

4 Conclusion and Outlook

As part of the MultiXscale CoE, it is our goal to improve the scalability and performance of ESPResSo and provide scalable couplings. To this end, we have undertaken work throughout the codebase. For example, multi-GPU support of lattice-Boltzmann coupled to CPU-based molecular dynamics was made available by coupling ESPResSo to waLBerla and was shown to scale well.

To improve scalability of short-range force calculation, we implemented a modified communication scheme, which reduces synchronization barriers. This turned out to be relevant mostly for low particle numbers per core. To improve cache efficiency and reduce communication, we introduced short-ranged force calculation based on the Cabana library. Here, further work is needed to parallelise remaining serial code, which dominates at high thread numbers.

Shared-memory parallelism is also being implemented in other parts of ESPResSo. For instance, the FFTs underpinning the P^3M electrostatics solver are now provided by the heFFTe library, which provides both, shared-memory parallelism and GPU support.

In summary, progress has been made on all major areas of the code. Still, further performance engineering is necessary, particularly on the Cabana-based short-ranged force calculation, for these benefits to come together.

Lastly, parallel writing efficiency of the simulation data in the standardised HDF5 format has been improved drastically, allowing for data exchange with other simulation packages.

Acknowledgements

The authors acknowledge the EuroHPC Joint Undertaking for awarding access to Vega at IZUM, Slovenia, as well as funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Compute Cluster grant no. [492175459](https://gepris.dfg.de/gepris/projekt/492175459) (PI: Holm) for the Ant cluster.

References

Acronyms used

HPC	High Performance Computing
MPI	Message Passing Interface
FFT	fast Fourier transform
I/O	Input/Output
HDF5	Hierarchical Data Format version 5
MD	molecular dynamics
AoS	array of structures
SoA	structure of arrays

Software mentioned

ESPResSo Extensible Simulation Package for Research on Soft Matter Systems

walBerla Widely Applicable Lattice Boltzmann solver from ERLAngen

heFFTe Highly Efficient FFT for Exascale

P³M Particle–Particle–Particle–Mesh

URLs referenced

Page ii

<https://www.multixscale.eu...> <https://www.multixscale.eu>
<https://www.multixscale.eu/deliverables...> <https://www.multixscale.eu/deliverables>
 Internal Project Management Link ... <https://github.com/multixscale/planning/>
jgrad@icp.uni-stuttgart.de ... <mailto:jgrad@icp.uni-stuttgart.de>
<https://creativecommons.org/licenses/by/4.0...> <https://creativecommons.org/licenses/by/4.0>

Page 10

[492175459 ... https://gepris.dfg.de/gepris/projekt/492175459?language=en](https://gepris.dfg.de/gepris/projekt/492175459?language=en)

Citations

- [1] F. Weik, R. Weeber, K. Szuttor, K. Breitsprecher, J. de Graaf, M. Kuron, J. Landsgesell, H. Menke, D. Sean, and C. Holm, “ESPResSo 4.0 – an extensible software package for simulating soft matter systems,” *European Physical Journal Special Topics*, vol. 227, no. 14, pp. 1789–1816, 2019. doi:[10.1140/epjst/e2019-800186-9](https://doi.org/10.1140/epjst/e2019-800186-9)
- [2] R. Weeber, J.-N. Grad, D. Beyer, P. M. Blanco, P. Kreissl, A. Reinauer, I. Tischler, P. Košovan, and C. Holm, “ESPResSo, a versatile open-source software package for simulating soft matter systems,” in *Comprehensive Computational Chemistry*, 1st ed., M. Yáñez and R. J. Boyd, Eds. Oxford: Elsevier, 2024, pp. 578–601. ISBN 978-0-12-823256-9. doi:[10.1016/B978-0-12-821978-2.00103-3](https://doi.org/10.1016/B978-0-12-821978-2.00103-3)
- [3] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggien, *The lattice Boltzmann method: principles and practice*, ser. Graduate texts in physics. Cham: Springer International Publishing, 2017. ISBN 978-3-319-44649-3. doi:[10.1007/978-3-319-44649-3](https://doi.org/10.1007/978-3-319-44649-3)
- [4] S. Slattery, S. T. Reeve, C. Junghans, D. Lebrun-Grandié, R. Bird, G. Chen, S. Fogerty, Y. Qiu, S. Schulz, A. Scheinberg, A. Isner, K. Chong, S. Moore, T. Germann, J. Belak, and S. Mniszewski, “Cabana: a performance portable library for particle-based simulations,” *Journal of Open Source Software*, vol. 7, no. 72, p. 4115, 2022. doi:[10.21105/joss.04115](https://doi.org/10.21105/joss.04115)
- [5] C. R. Trott, D. Lebrun-Grandie, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, and J. Wilke, “Kokkos 3: programming model extensions for the exascale era,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 805–817, Apr. 2022. doi:[10.1109/tpds.2021.3097283](https://doi.org/10.1109/tpds.2021.3097283)

- [6] A. Ayala, S. Tomov, A. Haidar, and J. Dongarra, “*heFFTe*: highly efficient FFT for exascale,” in *Computational Science – ICCS 2020*, ser. Lecture Notes in Computer Science, V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira, Eds., vol. 12137. Cham, Switzerland: Springer International Publishing, 2020. doi:[10.1007/978-3-030-50371-0_19](https://doi.org/10.1007/978-3-030-50371-0_19). ISBN 978-3-030-50371-0 pp. 262–275.
- [7] M. Holzer, M. Bauer, H. Köstler, and U. Rüde, “Highly efficient lattice Boltzmann multiphase simulations of immiscible fluids at high-density ratios on CPUs and GPUs through code generation,” *The International Journal of High Performance Computing Applications*, vol. 35, no. 4, pp. 413–427, 2021. doi:[10.1177/10943420211016525](https://doi.org/10.1177/10943420211016525)
- [8] The HDF Group, “Hierarchical Data Format, version 5,” GitHub. [Online]. Available: <https://github.com/HDFGroup/hdf5>
- [9] P. Corbett, D. Feitelson, S. Fineberg, Y. Hsu, B. Nitzberg, J.-P. Prost, M. Snir, B. Traversat, and P. Wong, *Overview of the MPI-IO parallel I/O interface*, ser. The Kluwer International Series in Engineering and Computer Science. Boston, Massachusetts, USA: Springer US, 1996, vol. 362, pp. 127–146. ISBN 978-1-4613-1401-1. doi:[10.1007/978-1-4613-1401-1_5](https://doi.org/10.1007/978-1-4613-1401-1_5)
- [10] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier, “The new ext4 filesystem: current status and future plans,” in *Proceedings of the Linux Symposium*, vol. 2, 2007, pp. 21–34. [Online]. Available: <https://www.kernel.org/doc/ols/2007/ols2007v2-pages-21-34.pdf>