BWRSE4HPC - ENABLING SCIENTIFIC DISCOVERY THROUGH RSE SUPPORT

MARCEL KOCH[1]*, ANDREAS BAER[1], RENÉ CASPART[1], JASMIN HÖRTER[1], GLEN HUNTER[1], THOMAS ISENSEE[2], DOMINIC KEMPF[2], KAI RIEDMILLER[2], TIM SCHRADER[1] AND INGA ULUSOY[2]

[1]SCIENTIFIC COMPUTING CENTER, KARLSRUHE INSTITUTE OF TECHNOLOGY, KARLSRUHE, GERMANY

[2]SCIENTIFIC SOFTWARE CENTER, INTERDISCIPLINARY CENTER FOR SCIENTIFIC COMPUTING (IWR), HEIDELBERG UNIVERSITY, HEIDELBERG, GERMANY

* MARCEL.KOCH@KIT.EDU

ABSTRACT

Software has become a key enabler for modern science. The bwRSE4HPC initiative supports researchers in Baden-Württemberg who use software to achieve their research goals. The project embeds professional research software engineering (RSE) practices into scientific projects that utilize high-performance computing (HPC). We present two collaborative efforts where bwRSE4HPC helped improve the performance, maintainability, and sustainability of scientific software running on bwHPC infrastructure.

Keywords: Research software engineering, High-performance computing, Software development support, Software sustainability, Scientific collaboration

1. INTRODUCTION

Modern research increasingly depends on software, both commercial and custom-built. While many researchers use established tools such as computational fluid dynamics solvers, molecular dynamics simulators, or data analysis platforms, others rely on open-source software or develop their own in-house solutions. This often places a significant burden on researchers, who may lack formal training in software engineering and face pressure to prioritize scientific results over code quality. The frequent turnover of developers, typically PhD students or postdocs on fixed-term contracts, further exacerbates challenges in software maintainability and long-term usability. These issues are well documented in recent efforts to define the competencies and responsibilities of Research Software Engineers[i].

The newly established bwRSE4HPC[1] initiative addresses these issues by providing expertise in sustainable research software engineering. The bwRSE4HPC team contributes specialized skills in areas such as performance optimization, parallelization, build systems, and maintainability. Researchers based at higher education institutions in the German state of Baden-Württ-

[1] https://www.bwrse4hpc.de/

temberg can apply for support in the form of short- to mid-term collaborative projects, typically lasting around six weeks or up to six months. The only requirement is that the software benefits from or is intended to run on high-performance computing (HPC) infrastructure.

In the following, two recent projects are described to highlight the benefits that are dedicated to RSEs can bring to research groups. In the first project, the RSEs improved the usability of the KIMMDY (**KInetic Monte Carlo/Molecular DYnamics**) software, and in the second project, the RSEs improve the sustainability and performance of the M++ software.

2. KIMMDY - REACTIVE MOLECULAR DYNAMICS ON GPUS

KIMMDY[ii],[2] is a kinetic Monte Carlo code that has been developed for molecular dynamics (MD). KIMMDY acts as an interface for different reactions and topologies to act on molecules. These processes can be provided through the plug-in interface of KIMMDY, allowing users to define their own plugins and making the code extensible. The molecular dynamics part of KIMMDY utilizes GROMACS, a highly parallelized molecular dynamics code for simulating the Newtonian dynamics of many particles. The software additionally makes use of GPU acceleration with CUDA to produce results efficiently.

The researchers using KIMMDY faced three issues: 1) Complicated dependency management, 2) Lacking confidence in plug-in integrations, and 3) Excessive use of GPU memory. During the collaboration, RSEs from bwRSE4HPC tackled these issues.

A. DEPENDENCY MANAGEMENT

The KIMMDY ecosystem consists of the main KIMMDY package and currently consists of five plugins. To simplify the dependency management, the project was restructured. The main repository links the officially supported plugins as sub-repositories. Furthermore, the recommended development setup now uses uv[3], and the plugins are registered in a uv-workspace. This workspace allows compiling the dependencies of all packages into one lock file, ensuring compatibility. The new workspace also speeds up installation compared to the previous setup using pip. Lastly, the switch to UV allows installing KIMMDY and all plugins, including the machine learning frameworks used, and the Python runtime, within a single command. This should make setup on the HPC infrastructure trivial. Support for the latest GROMACS version was added, further simplifying the process, as GROMACS no longer needs to be patched with PLUMED during compilation.

2 https://graeter-group.github.io/kimmdy/
3 https://docs.astral.sh/uv/

B. PLUG-IN CONFIDENCE

While KIMMDY itself has sufficient unit testing, the full simulation pipeline, including plug-ins, was less thoroughly tested. In particular, three out of the five plug-ins were missing integration tests. The RSEs implemented the integration tests for all missing plug-ins, following a similar framework for the integration tests that already exist within KIMMDY. The integration tests were tested on both Helix[4] and the CI pipeline is present with GitHub Actions.

One notable exception is that the integration tests of the KIMMDY-hydrolysis plug-in, which were only run on Helix. This is due to the integration tests taking too long for the CI pipeline. In addition to the development of the integration tests, testing frameworks for KIMMDY-hydrolysis and KIMMDY-dimerization were established for their respective CI pipelines. This had the additional benefit of increasing the total test coverage of KIMMDY from 4% to 77%.

C. GPU MEMORY

The plugin for the HAT reaction uses machine learning models to predict reactions. The researchers were experiencing an issue with GPU memory usage, in which the machine learning framework used, TensorFlow, was not releasing the GPU memory after the models were generated and used. When KIMMDY would start an MD simulation using GROMACS in the following step, GROMACS was unable to use the GPU due to insufficient memory available. As a consequence, twice as many GPUs were needed to run KIMMDY with the HAT plug-in..

TensorFlow itself and the implicit object management of Python don't provide a method to release GPU memory. The memory is only freed up completely when the parent process exits. Therefore, all code using TensorFlow was encapsulated in a separate process, which could be terminated after the predictions are done for a simulation. This incurs an overhead, since the models need to be reloaded frequently during a KIMMDY simulation. However, this was found to be negligible compared to the subsequent computations with GROMACS (~22 s vs. 12 h). As an additional layer of scrutiny, a unit test called test GPU memory release was implemented into the plug-in's testing framework.

3. M++ - INTEGRATION WITH LINEAR ALGEBRA LIBRARY

Solving linear systems is a critical part of the finite element (FE) library M++[iii]. Until now, these linear solvers were developed by the M++ team themselves, which led to an increased maintenance burden on the researchers, as well as less functionality than specialized libraries. Thus, the task of solving linear systems should be handed off to a specialized library.

4 https://wiki.bwhpc.de/e/Helix

The RSEs from bwRSE4HPC proposed to introduce the numerical linear algebra library Ginkgo[iv] as a backend for solving linear systems and generic linear algebra operations. The library provides GPU implementation for most of its components. Thus, it could enable more efficient use of M++ on modern supercomputers. Ginkgo is developed by a dedicated team, so M++ can benefit directly from their advances in numerical linear algebra. Only a small interface between M++ and Ginkgo is introduced, which delegates almost all operations to Ginkgo. Since the new backend will most likely not cover the full functionality of the existing backend, the Ginkgo backend will coexist with the existing backend.

As for writing this article, the project is still ongoing. After half of the project's runtime, the RSEs have succeeded in *1) Adding Ginkgo to M++'s build system, 2) Integrating Ginkgo solvers, and 3) Solving linear systems on a single GPU.*

The Ginkgo solver is part of the existing solver hierarchy in M++, which makes it easily available to users. Additionally, the external Ginkgo solver has a negligible overhead. Data from M++ is directly wrapped in Ginkgo data structures, which eliminates unnecessary copies at the interface. All single-node solvers and preconditioners from Ginkgo are available to M++. Ginkgo's multi-node components will be added in the second half of the project.

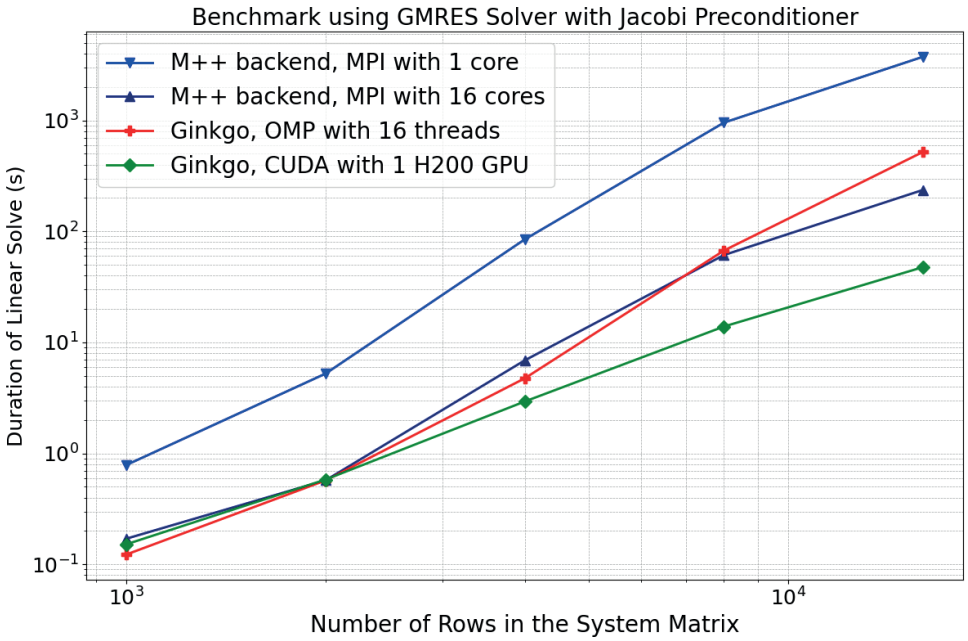

Figure 1: Time to solve a linear system comparing the built-in and Ginkgo backends. Run on a single HoreKa node

To highlight the benefit from the first phase of the project, a small benchmark example is considered. Fig. 1 shows the performance of the linear solver for configurations using M++ built-in backend and the Ginkgo backend. The linear system stems from a 2D Laplace problem, discretized by M++. All configurations use the GMRES solver with a Jacobi preconditioner. The benchmark is run on a single node of the HoreKa[5] supercomputer. While the CPU implementation of Ginkgo using OpenMP is comparable to M++'s implementation using MPI, the GPU implementation of Ginkgo provides the best time-to-solution for larger problems.

4. CONCLUSION

These two examples demonstrate how research in software engineering directly supports more efficient, scalable, and sustainable use of the bwHPC infrastructure. Tasks that are not directly tied to publishable outcomes can be addressed by dedicated RSEs. The KIMMDY project benefits from improved usability and better resource utilization, making it more attractive for other researchers. Although still ongoing, the M++ project already shows gains from leveraging modern GPU hardware while reducing the maintenance burden by relying on specialized libraries. Overall, bwRSE4HPC enables researchers without a strong computational background to successfully access and use bwHPC systems in their work. By applying best practices in software design, reproducibility, and automation, it fosters broader and more inclusive use of high-performance computing resources across Baden-Württemberg.

5. ACKNOWLEDGEMENT

bwRSE4HPC is funded by the Ministry of Science, Research, and Arts, Baden-Württemberg, Germany. The authors gratefully acknowledge the computing time provided on the high-performance computer HoreKa by the National High-Performance Computing Center at KIT (NHR@KIT). This center is jointly supported by the Federal Ministry of Education and Research and the Ministry of Science, Research and the Arts of Baden-Württemberg, as part of the National High-Performance Computing (NHR) joint funding program (https://www.nhr-verein.de/en/our-partners). HoreKa is partly funded by the German Research Foundation (DFG). The authors thank Niklas Baumgarten, Heidelberg University, for his input on M++, and Frauke Gräter, Jannik Buhr, and Eric Hartmann, Heidelberg University, for their input on KIMMDY.

[5] https://www.nhr.kit.edu/userdocs/horeka/

6. REFERENCES

[i] F. Goth et al., ''Foundational Competencies and Responsibilities of a Research Software Engineer: Current State and Suggestions for Future Directions'', F1000Research, vol. 13, no. 1429, Sep. 2025, https://doi.org/10.12688/f1000research.157778.2

[ii] E. Hartmann, J. Buhr, K. Riedmiller, E. Ulanov, B. N. Schüpp, D. Kiesewetter, D. Sucerquia, C. Aponte-Santamaría and F. Gräter, ''KIMMDY: a biomolecular reaction emulator'', bioRxiv 2025.07.02.662624, July 2025, https://doi.org/10.1101/2025.07.02.662624

[iii] N. Baumgarten and C. Wieners, ''The parallel finite element system M++ with integrated multilevel preconditioning and multilevel Monte Carlo methods'', Computers & Mathematics with Applications, vol. 81, pp. 391-406, Apr. 2020, https://doi.org/10.1016/j.camwa.2020.03.004

[iv] Anzt et al., ''Ginkgo: A Modern Linear Operator Algebra Framework for High Performance Computing'', ACM Transactions on Mathematical Software, vol. 48, no. 1, pp. 1–33, Feb. 2022, https://doi.org/10.48550/arXiv.2507.11603