JoDaKISS | Journal of Data- and Knowledge-integrated Simulation Science

Replication Data for: Data-Parallel PU-RBF Interpolation

David Schneider[i]

2025

Cite as

Schneider, David. Replication Data for: Data-Parallel PU-RBF Interpolation *Journal of Data- and Knowledge-integrated Simulation Science* Nr. 1/2026: S.1-7. DOI: 10.46298/jodakiss.16884.

[i]Institute for Parallel and Distributed Systems, University of Stuttgart

Abstract

The dataset (Schneider, 2025) contains software, experimental setups, and performance measurements to replicate and complement the results presented in Section 5.3 of Schneider (2026). This section deals with a performance-portable implementation of the partition-of-unity radial-basis-function (PU-RBF) interpolation in the open-source library preCICE (Schneider and Uekermann, 2025). At its core, the algorithmic structure of the PU-RBF interpolation requires solving many small matrices, which makes it a promising candidate for parallel processing on accelerator cards using batched solvers. However, conventional solver routines typically assume that all matrices have the same size and structure, which is not the case for the PU-RBF interpolation in preCICE. The presented implementation concept is based on the team-parallel execution model offered by Kokkos and Kokkos Kernels (Rajamanickam et al., 2021; C. Trott et al., 2021; C. R. Trott et al., 2022), where available hardware threads are grouped into thread teams and deployed in parallel onto the executor. The implementation investigates team-parallel kernels for various stages and user parameters of the PU-RBF interpolation. Since preCICE supports parallel computing on distributed-memory systems, the resulting implementation can leverage both distributed- and shared-memory parallelism. The experiments investigate performance on various platforms, including CUDA, HIP, SYCL, and OpenMP, as well as on multiple levels, including kernel-level, algorithmic-level, node-level, and cluster-level performance.

1 Background

preCICE (Chourdakis et al., 2022) is an open-source coupling library written in `C++`. It is designed for the partitioned coupling of existing solvers for the simulation of coupled problems and is widely used in the simulation science community (Uekermann, 2020). Its functionality spectrum can broadly be grouped into four categories: algorithms for equation coupling, data communication, time interpolation, and data mapping. The PU-RBF interpolation presented in the dataset (Schneider, 2025) falls into the last category, that is, it is one of the data mapping algorithms available to preCICE users. Data mapping solves a scattered-data interpolation problem resulting from non-matching coupling interfaces defined by the solvers participating in the simulation. The PU-RBF interpolation has been proven to be a powerful method for tackling data mapping problems by Schneider and Uekermann (2025). However, the underlying implementation was restricted to central processing units (CPUs) and to a distributed-memory parallelization via MPI, although modern compute clusters typically offer heterogeneous hardware such as graphics processing units (cf. the top ten supercomputers in June 2025[1]). A performance-portable implementation of the PU-RBF interpolation in preCICE was developed as part of Section 5.3 of Schneider (2026). The dataset complements the discussed results and ensures reproducibility.

[1] https://top500.org/lists/top500/2025/06/

2 Specification

Table 1: Specification of the dataset

Subject	Computational science
Specific subject area	Performance-portable data mapping algorithms for partitioned multiphysics coupling, PU-RBF interpolation, and high-performance computing.
Type of data	Software, performance measurements in `.csv` files and depicted in PDF plots, experimental setup files.
Data collection	Runtime measurements were collected using the built-in performance-profiling functionality of preCICE.
Related research article	Schneider, 2026, Section 5.3
Software repository	https://github.com/precice/precice
Programming language	C++
Website	https://precice.org/

3 Value

The primary focus of this data publication is to ensure the reproducibility of the results presented in Section 5.3 of Schneider (2026). Due to the complexity of the investigated parameter space, the dataset also effectively contains additional results and performance measurements not explicitly shown in the affiliated document. In the broader context of preCICE, the developed method gives users additional flexibility to load-balance their simulation setups and leverage the available hardware. It provides users with a unique reference for configuring their simulations and makes the design decisions as well as the configuration options of the algorithm transparent and understandable. Given the accessibility of the code through the dataset and preCICE, the implementation might also serve as a blueprint for other communities where efficient interpolation implementations are required.

4 Data Description

A detailed description of the dataset is provided in the top-level dataset `README.md`.

5 Simulation Design, Models, Methods

As mentioned before, detailed information about the mathematical model and the practical implementation is contained in Section 5.3 of Schneider (2026). The dataset includes additional explanations of the individual experiments. The following provides a high-level overview of the experimental setups and investigated parameters.

In preCICE, major parts of the PU-RBF algorithm are implemented in the following three files:

- `src/mapping/device/KokkosTypes.hpp`, which includes alias definitions for code readability,
- `src/mapping/device/KokkosPUMKernels_Impl.hpp`, which contains the device kernel implementations, and
- `src/mapping/BatchedRBFSolver.hpp`, which orchestrates the overall solver algorithm.

The performance data are typically grouped according to algorithm sections executed only once, the *initialization* performance of the algorithm, and sections executed repeatedly, the *evaluation* performance of the algorithm. Furthermore, the PU-RBF interpolation is implemented in two distinct variants: in the *minimal-compute* variant, the implementation precomputes all data structures during the initialization phase and reuses them during the evaluation phase. In the *minimal-memory* variant, the implementation recomputes data structures during the evaluation phase to minimize the overall memory footprint of the algorithm. Additional parameters of the PU-RBF interpolation, which were inherited from the CPU implementation, include the partition sizes (called vertices-per-cluster in the configuration), and the projection, which affects the partition layout. The partition size determines the average size of the local matrices each thread team operates on, and the projection configuration is a parameter affecting how densely populated individual partitions are on average. Overall, the experimental setup covers CUDA, HIP, SYCL, and OpenMP as executor backends on the compute platforms documented in the top-level `README.md`, although not all backends were used in every experiment.

The experimental setup follows a bottom-up design. On the first level, the efficiency of individual team-parallel kernels is evaluated for different user configurations. The overall goal is to find appropriate and robust default settings for the team size each kernel operates on. Larger teams can solve the local matrices faster, but at the same time, fewer teams can operate concurrently on the device. On the second level, a performance breakdown of the entire algorithm is performed using the concluded team sizes of the first level. Notably, this performance breakdown includes algorithm sections executed on the host, memory transfers, and compares the execution time of various kernels. On the third level, the data-parallel execution on a device is combined with distributed-memory parallelization via MPI to determine a suitable MPI granularity. Using multiple MPI ranks per device scales down the problem size each device needs to solve. At the same

time, the contention incurred through sharing the device with another thread might lower the overall efficiency. On the fourth level, the data-parallel PU-RBF interpolation is compared against its CPU variant in a strong-scaling experiment on SuperMUC-NG Phase 2.

6 Limitations

The current implementation was tested and analyzed on CUDA, HIP, SYCL, and OpenMP as parallel executors. However, Kokkos supports additional backends, such as HPX, whose application has not been tested and may not yet be functional.

Moreover, preCICE is a library that is called by the application code. If the application code also relies on Kokkos as a performance-portable programming model, then the device executor configured in the application cannot be separated from the device executor configured in preCICE, because Kokkos may be initialized only once.

7 Technical Validation

To verify the correctness of the implementation, individual backends are tested as part of the continuous integration of preCICE. If preCICE is compiled for a specific backend, including the tests, the tests can be executed using the `ctest` command. All tests should pass. Similarly, the underlying testing environment for the numerical experiments includes a dedicated continuous-integration infrastructure (Schneider et al., 2024). The performance breakdown of the algorithm (directory `performance-breakdown` in the dataset) also contains measurements of the discrete ℓ_2-error computed numerically against an analytical test function.

For the validity of the performance measurements, experiments were repeated five times, and the reported runtime corresponds to the average of three runs, where the maximum and minimum runtimes were excluded. The kernel measurements (directory `team-size` in the dataset) were repeated ten times due to the small timescale, where the averaging was performed over eight runs, again, excluding the minimum and maximum runtime. The largest run of the strong-scaling experiment was performed only once due to resource limitations. Running and reproducing the experiments is fully automated through the included scripts, as explained in the `README.md`.

8 Usage Notes

While the raw measurements are included in the dataset, all measurements are also available as PDF plots. These plots can readily be inspected. Further helper scripts for a quantitative assessment of individual experiments are included in the dataset and

described in the `README.md` files within the experiment subdirectories. For all Python scripts, a `requirements.txt` file is provided in the dataset.

Acknowledgment

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project 515015468, and I thank the DFG for supporting this work by funding – EXC2075 – 390740016 under Germany's Excellence Strategy. I acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). Furthermore, I gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer SuperMUC-NG at Leibniz Supercomputing Centre (www.lrz.de).

Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

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. (2022). PreCICE v2: A sustainable and user-friendly coupling library [version 2; peer review: 2 approved]. *Open Research Europe*, *2*(51). https://doi.org/10.12688/openreseurope.14445.2

Rajamanickam, S., Acer, S., Berger-Vergiat, L., Dang, V., Ellingwood, N., Harvey, E., Kelley, B., Trott, C. R., Wilke, J., & Yamazaki, I. (2021). Kokkos kernels: Performance portable sparse/dense linear algebra and graph kernels. https://arxiv.org/abs/2103.11991

Schneider, D. (2025). *Replication Data for: Data-Parallel PU-RBF Interpolation*. https://doi.org/10.18419/darus-5392

Schneider, D. (2026). *Flexible and efficient data mapping for simulation of coupled problems* [Doctoral dissertation, University of Stuttgart].

Schneider, D., & Uekermann, B. (2025). Efficient partition-of-unity radial-basis-function interpolation for coupled problems. *SIAM Journal on Scientific Computing*, *47*(2), B558–B582. https://doi.org/10.1137/24M1663843

Schneider, D., Yurt, M. K., Simonis, F., & Uekermann, B. (2024). ASTE: An artificial solver testing environment for partitioned coupling with preCICE. *Journal of Open Source Software*, *9*(103), 7127. https://doi.org/10.21105/joss.07127

Trott, C., Berger-Vergiat, L., Poliakoff, D., Rajamanickam, S., Lebrun-Grandie, D., Madsen, J., Al Awar, N., Gligoric, M., Shipman, G., & Womeldorff, G. (2021). The kokkos ecosystem: Comprehensive performance portability for high performance computing. *Computing in Science Engineering*, *23*(5), 10–18. https://doi.org/10.1109/MCSE.2021.3098509

Trott, C. R., Lebrun-Grandié, D., Arndt, D., Ciesko, J., Dang, V., Ellingwood, N., Gayatri, R., Harvey, E., Hollman, D. S., Ibanez, D., Liber, N., Madsen, J., Miles, J., Poliakoff, D., Powell, A., Rajamanickam, S., Simberg, M., Sunderland, D., Turcksin, B., & Wilke, J. (2022). Kokkos 3: Programming model extensions for the exascale era. *IEEE Transactions on Parallel and Distributed Systems*, *33*(4), 805–817. https://doi.org/10.1109/TPDS.2021.3097283

Uekermann, B. (2020). How did precice get popular? https://doi.org/10.5281/zenodo.12795485