natESM

Sprint Documentation 21

FESOM-C: Profiling, Analysis, and Optimization Roadmap

Sergey Sukov[1], Vera Sidorenko[2] and Alexey Androsov[2]

[1]Jülich Supercomputing Center, Forschungszentrum Jülich GmbH, Jülich, Germany
[2]Alfred-Wegener-Institut, Helmholtz-Zentrum für Polar- und Meeresforschung, Bremerhaven, Germany

Contact: info@nat-esm.de

Published on 10.06.2026 on https://www.nat-esm.de/services/accepted-sprints

1 Summary

The main objective of the sprint was to profile the FESOM-C code and analyze the scaling behavior of its MPI and OpenMP parallel versions during 2D and 3D simulation runs. Based on the information and data obtained during the sprint, a code-optimization strategy was developed. The strategy is focused on reducing the execution time of the main time loop by minimizing MPI data transfer overhead and in a stepwise manner increasing the performance of individual subroutines.

2 General Information

Start and end date:	06.12.2025 – 06.03.2026
Intended period:	3 months
Responsible RSE:	Sergey Sukov (JSC)
Responsible scientists:	Vera Sidorenko and Alexey Androsov (AWI)

FESOM-C is a recently designed coastal branch of the global finite-volume sea-ice ocean model FESOM. Being developed for large-scale ocean applications, FESOM still parameterizes many coastal processes which are resolved in FESOM-C and are required for proper representation of the exchange between the marginal seas and the global ocean. The FESOM-C numerical core is designed to provide the most efficient coupling between coastal and global solutions. FESOM-C and FESOM organize flux treatment in the same manner. FESOM-C solves 3D primitive equations in the Boussinesq, hydrostatic, and traditional approximations for the momentum, continuity, and density constituents, and it uses a terrain-following coordinate in the vertical. It employs the cell-vertex finite-volume discretization (quasi-B-grid) and applies to any configuration of triangular, quadrangular, or hybrid meshes. The schemes for computing vertical eddy viscosity and diffusivity can be selected via the connected GOTM (General Ocean Turbulence Model) library, developed at the Leibniz Institute for Baltic Sea Research Warnemünde (IOW). The numerical scheme of FESOM-C splits the fast and slow motions into barotropic and baroclinic subsystems. It employs an asynchronous time-stepping method, assuming that the integration of temperature and salinity is half-step shifted with respect to momentum.

The core code of the software package is written in Fortran. The computational algorithm is parallelized using MPI with static domain decomposition for distributed-memory architectures and

OpenMP, exploiting loop-level parallelism on shared-memory multicore platforms. Mesh partitioning using Metis library routines is performed in a standalone sequential preprocessor. The OpenMP and MPI versions of the code are hosted and maintained in separate Git branches.

3 Sprint Objectives

The main goal of the sprint was to profile the FESOM-C code and analyze its parallel efficiency during 2D and 3D simulation runs, in order to identify performance bottlenecks and develop an optimization strategy. A second key objective was to review and, where possible, consolidate the MPI and OpenMP code branches.

4 Procedure and Insights

4.1 Technical Approach / Procedure

The sprint schedule consisted of the following stages:

1. Familiarization with the software-package structure, including adding executable build support using the NVHPC toolkit.
2. Integration of an internal timers library, and a library for monitoring MPI process and OpenMP thread affinity.
3. Performance evaluation and strong scaling tests of MPI and OpenMP implementations.
4. Preliminary optimization of some frequently called subroutines.
5. Development of a code-optimization strategy.

Building the executable using the NVIDIA Fortran compiler was required to verify the technical feasibility of porting the code to GPUs using OpenMP and OpenACC directives.
The first library linked to the code for monitoring and configuring process-to-core affinity was developed within natESM Sprint #16 (PDAF2GPU). The second internal timers library was developed specifically for this sprint. At present, in terms of supported features, it provides a significantly reduced subset of the functionality of the ICON internal timers module. Initially, calls to timer-control functions were inserted only into FESOM-C data-exchange procedures to estimate the share of overhead in the total program execution time. Next, the internal timers were also used to track the progress of optimization of individual subroutines.

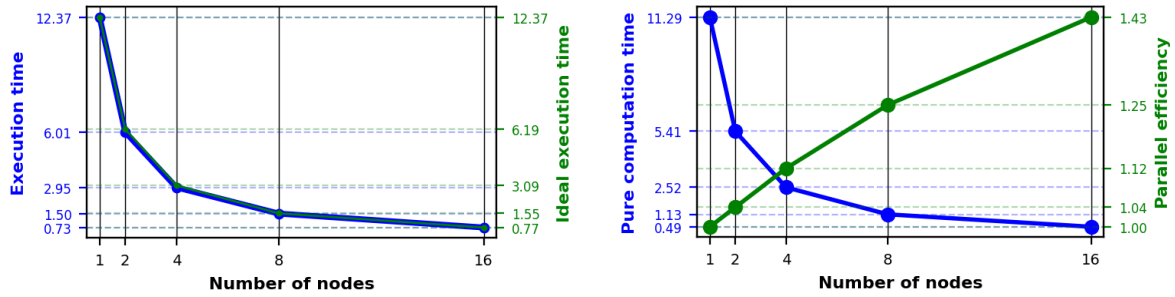
During performance-evaluation runs and strong scaling tests, triangular and quadrilateral surface meshes containing up to 2.5 million cells were used (40 vertical layers in the 3D case). All experiments were performed on the Levante supercomputer (DKRZ) using up to 16 nodes (up to 2048 cores) of the CPU partition.

Preliminary optimization of four subroutines within the fv_advection.f90 source file focused on minimizing the number and size of temporary arrays and vectorizing computations.

Discussions of the sprint's technical details among the RSE and the scientists were carried out via email, the Mattermost channel, and weekly video conferences.

4.2 General Insights

In strong-scaling tests using 3D simulation cases, the MPI version of the code demonstrated near-ideal parallel efficiency as a function of the number of nodes (Figure 1a). The increase in communication overhead is offset by a superlinear speedup of the computational core (Figure 1b). This superlinear scaling is most likely due to improved cache utilization as the local problem size per MPI process decreases. CPU load imbalance remains within 15% of the average computation time.



a) Total execution time

b) Pure computation time without data exchange

Figure 1. Strong scaling plots for MPI parallelization.

The computational mesh consists of 1,156,775 nodes, 2,303,346 elements, and 40 vertical levels. Execution times (in seconds) correspond to 25 time steps.

Despite the formally high-performance metrics of the computational core subroutines, there remains potential for further improvement. The preliminary optimization of four subroutines in the advection module, carried out during the sprint, resulted in a 10% reduction in overall execution time, independent of the MPI process group size.

In the 2D case, the parallel efficiency behaves similarly, but the scalability limit is reached much earlier, as expected. Once communication overhead begins to dominate, the overall execution time varies in a largely irregular manner and is strongly influenced by the distribution of MPI processes across the assigned CPU cores.

Overall, for the FESOM-C code, efficient implementation of data-exchange routines is critical. The number of data-transfer points at each time step depends on the model configuration and may exceed 100. Data exchange is implemented using asynchronous message transfer with derived MPI data types. The validity of this approach requires further verification, since, at the very least, the data is received as a single continuous block. Besides, when values of multiple variables need to be exchanged simultaneously, a pair of data send/receive procedures is performed separately for each variable.

The OpenMP version of the code shows relatively poor performance in strong-scaling tests. Within a single NUMA node, the speedup is limited to 1.65× (Figure 2).

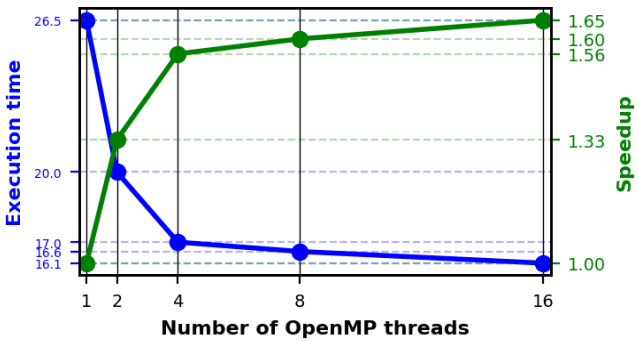


Figure 2. Strong scaling plot for OpenMP parallelization.

Computational mesh consists of 66,177 nodes, 65,536 quadrilateral cells, and 40 vertical layers. Execution time (in seconds) corresponds to 50 time steps.

This situation has two explanations. First, not all code blocks are parallelized using OpenMP directives. Second, the algorithm has low computational intensity. Attempts made during the sprint to restructure loops and apply alternative parallelization approaches did not lead to any noticeable performance improvement.

In its initial development, the MPI version of the code was based on the OpenMP version and still contains OpenMP directives. However, compiling the MPI version with OpenMP parallelization

enabled leads to errors. Therefore, a straightforward merge of the code branches using version control tools such as Git is not feasible. The most suitable solution here is a step-by-step revision of the subroutines, including inspection of existing OpenMP directives and enabling them, or applying new directives.

5 Results

Based on the information and data obtained during the sprint, an optimization and development roadmap for FESOM-C was formulated, and a detailed plan of priority code improvement tasks was prepared. The implementation of the work items included in the plan could become the subject of a follow-up natESM sprint. In this case, the main objective of the sprint would be to achieve maximum performance of the MPI version of the program on CPU systems without introducing major structural changes to either the codebase or the computational algorithm. Such an approach would reduce the execution time of production runs while preserving the flexibility for scientists to experiment with implementations of various numerical methods.

Achieving the above goal requires solving two technical tasks:

- reducing MPI data-transfer overhead;
- improving the performance of individual subroutines.

The primary optimization approach for MPI data exchanges is to copy the values of the variables being transferred into contiguous buffers instead of using MPI derived datatypes. Furthermore, it is assumed that multiple transfers of different variables will be combined into a single transaction wherever possible.

In the next step, the subroutine code will be reorganized to minimize the number and size of temporary arrays. In cases where this is not feasible, a custom library will be used to manage the memory of large reusable temporary arrays. In addition, loops over mesh vertices, edges, and elements must satisfy vectorization requirements and be parallelized using OpenMP directives.

According to the RSE, the listed tasks can be completed within six months, provided that the applicants prepare an appropriate set of test cases in advance. A possible success criterion for the sprint could be reducing the execution time of the MPI program's main computational loop by 20% or more. The hybrid MPI+OpenMP parallelization version should be considered primarily as a platform for experiments aimed at the later porting of the code to GPUs. However, any work related, directly or indirectly, to GPU porting cannot be included in the scope of a follow-up sprint, due to time constraints.

6 Conclusions and Outlook

During the first stage of the sprint activities, the FESOM-C code was systematically profiled, strong-scaling experiments were carried out for MPI and OpenMP configurations, and several computationally intensive subroutines were preliminarily optimized. These efforts provided a detailed understanding of the current performance limitations of the model and identified the MPI communication layer, temporary memory management, and insufficient vectorization as the primary optimization targets.

The present project therefore represents the second stage of the optimization effort. Building directly on the profiling results and technical insights obtained during the initial sprint, the next phase will focus on implementing and validating the identified optimization strategies within the production MPI version of FESOM-C. The main objectives are to reduce MPI communication overhead, improve cache and memory efficiency, increase vectorization efficiency, and further develop hybrid MPI+OpenMP capabilities where beneficial.

Achieving a reduction of at least 20% in the runtime of the main computational loop would significantly improve the feasibility of large-scale coastal simulations and long-term production experiments. Faster execution will enable higher-resolution simulations, larger ensembles, and

more comprehensive sensitivity studies while maintaining the scientific flexibility of the existing codebase.

From a scientific perspective, improving the computational efficiency of FESOM-C is essential for advancing regional and coastal Earth-system modeling. The model is specifically designed to resolve processes that are insufficiently represented in global ocean models, including coastal circulation, tidal dynamics, transport processes, sea-level rise impacts, and ecosystem connectivity. Enhanced scalability and reduced runtime will therefore directly support more detailed and physically realistic studies of coastal and shelf-sea dynamics and strengthen the integration of coastal-process modeling within the broader natESM framework.

In addition to immediate performance gains on CPU-based HPC systems, the planned code modernization and gradual harmonization of MPI and OpenMP implementations will provide a more maintainable and extensible software foundation for future developments, including potential GPU-oriented research beyond the scope of the current sprint phase.

7 References

FESOM-C Resources and References

Online resources

1. https://www.awi.de/forschung/biowissenschaften/oekologie-der-kuesten/arbeitsgruppen/ag-kuestendynamik-modellierung.html
2. https://gotm.net/portfolio/documentation/

Key publications

Androsov et al. (2019) — FESOM-C v.2: coastal dynamics on hybrid unstructured meshes (Geosci. Model Dev.)
Fofonova et al. (2019) — Non-linear aspects of the tidal dynamics in the Sylt-Rømø Bight, south-eastern North Sea (Ocean Science)
Kuznetsov et al. (2020) — Evaluation and Application of Newly Designed Finite Volume Coastal Model FESOM-C, Effect of Variable Resolution in the Southeastern North Sea (Water)
Fofonova et al. (2021) — Plume spreading test case for coastal ocean models (Geoscientific Model Development)
Neder et al. (2022) — Modelling suspended particulate matter dynamics at an Antarctic fjord impacted by glacier melt (Journal of Marine Systems)
Rubinetti et al. (2023) — Water Mass Transport Changes through the Venice Lagoon Inlets from Projected Sea-Level Changes under a Climate Warming Scenario (Water)
Konyssova et al. (2025) — Changes in tidal dynamics in response to sea level rise in the Sylt-Rømø Bight (Wadden Sea) (Ocean Dynamics)
Sidorenko et al. (2025) — Connectivity and larval drift across marine protected areas in the German Bight, North Sea: Necessity of stepping stones (Journal of Sea Research)

Source code repositories

3. OpenMP branch: https://github.com/FESOM/fesom-c/tree/OMP
4. MPI branch: https://github.com/FESOM/fesom-c/tree/MPI
5. Current sprint branch: https://github.com/BertramFZJ/natESM_FESOM-C

ecosystem applications.