natESM

Sprint Documentation #12

The PALM-GPU sprint

Wilton Jaciel Loch[1], Siegfried Raasch[2]

[1] Deutsches Klimarechenzentrum GmbH, Hamburg, Germany
[2] Institute of Meteorology and Climatology, Leibniz University Hannover, Hanover, Germany

Contact: info@nat-esm.de

Published on 26 May 2025 on https://www.nat-esm.de/services/accepted-sprints

1 Summary

The sprint focused on improving and extending the existing OpenACC ported code in PALM. The first responsible RSE was Enrico Degregori, who started by analyzing the code and implementing solutions to outstanding problems in the porting. The first was the reinstitution of the execution using cuFFT and its corresponding CI tests, followed by the implementation of serialization subroutines to avoid deep performance penalties of exchanging MPI derived types residing in GPU memory.

After a 6-month pause, the sprint was then resumed with the responsible RSE assigned to be Wilton Jaciel Loch. The sprint continued with an analysis of the code and improvements to the original porting, mainly in the form of removal of still existing implicit data copies. The next steps were then to port the humidity, land surface model (LSM) and multigrid solver. The first of the three was the only one successfully integrated with the original porting, while the last two can only run correctly in standalone fashion due to incorrect results when running with the original core port. The CI infrastructure was partly updated, allowing the inclusion of the humidity into the existing OpenACC tests and the addition of new tests for the Land Surface Model and the multigrid solver ports. No rigorous performance analysis was performed with the ported code. This is due first to an unfit state of the code for fair GPU performance assessment and second to a choice of trying to solve the integration problems with the remaining sprint time. Porting the urban surface model and the ray-tracing algorithm remain as future work for a possible next sprint.

2 General Information

Start and end date:	January 2024 – March 2024, October 2024 - February 2025
Intended period:	06 months
Responsible RSE:	Wilton Jaciel Loch, DKRZ
Responsible scientist:	Siegfried Raasch, IMUK

PALM is a turbulence resolving model (LES/DNS model) for simulating atmospheric and oceanic boundary layers. Topography and buildings are explicitly resolved by a Cartesian numerical grid. Beside the core model (which includes the Boussinesq/anelastic approximated Navier-Stokes equations and an equation for internal energy), it contains a large number of modules that can

explicitly resolve various physical processes relevant to and associated with, e.g. land and urban surfaces or vegetation.

All PALM components are parallelized via 2D-domain decomposition, using MPI for data exchange between the processes. The code is highly optimized for various architectures (cache-based Intel/ARM CPUS, NEC-Aurora vector engines), and also allows hybrid MPI/OpenMP usage. The core model also runs on multiple GPUs via CUDA-aware MPI. The code is written in Fortran and has around 201 KLOC.

PALM is applied by various research groups worldwide. The number of users, currently more than 1000, has grown significantly over the last years. Further details about PALM can be found in the online documentation and in an overview article (https://doi.org/10.5194/gmd-13-1335-2020).

3 Sprint Objectives

The sprint's overarching goal was to advance the state of the OpenACC GPU porting in PALM so that it can make effective use of the accelerators. More specifically, a number of mostly independent intermediate goals were defined:

1) Implement a serialization strategy for MPI exchanges to avoid the well-known performance degradations of using derived data types with GPU memory.
2) Porting of the Land Surface Model (LSM).
3) Porting the Urban Surface Model (USM).
4) Porting the initialization ray-tracing algorithm (part of USM).

During the development of the sprint two other goals were outlined:

1) Port the code paths for when humidity is included in the simulation, required for both LSM and USM.
2) Port the Multigrid solver.

4 Procedure and Insights

4.1 Technical Approach / Procedure

The first part of the sprint was carried out by Enrico Degregori, former natESM RSE. During this part, the focus was on solving the issues with the existing porting — more importantly, the MPI performance degradations. This was successfully accomplished with the implementation of serialization for the exchange routines present in PALM. In this sense, instead of exchanging direct GPU memory of data structures with derived data types – which are notoriously time-consuming – the data contained in such data structures can now be serialized onto GPU arrays and exchanged in similar fashion, producing considerable performance improvements. Another task completed by Enrico was fixing the usage of the NVIDIA FFT library cuFFT and its correspondent CI tests. This is very useful as it allows faster FFT calculations on the GPU using the vendor libraries.

The second part of the sprint was carried out by Wilton Jaciel Loch, who started with a profiling of the existing ported version of the code, which showed that there were implicit data copies for sets of kernels in the time loop. After these were fixed, they were also merged into the main branch. The work then proceeded with an analysis of the LSM and USM for porting, both consisting of dedicated files and isolated functions, but also of many code blocks embedded in conditionals in other locations. This complicates the porting effort, creating many data inconsistencies when going from running the core porting to the version with LSM or USM activated. Siegfried and Wilton also identified that it would be first necessary to port the code paths related to the humidity, as it is required by both LSM and USM. This was not included in the original goals of the sprint, but had to be implemented in order to reach them.

The character of the humidity, different from the other parts, is that it is mostly spread into conditionals over the code. Therefore, two options were possible for porting it: either (1) adding

explicit copies to all the humidity conditionals to make the memory state compatible between ported and non-ported code, or (2) disabling the core porting and treating the humidity as a standalone. The first option would require complex initial work to get to a stable state (which is also not directly useful for the final port), but once finished, would have the benefit of both porting and integration being conducted together. The second option had the benefits of no upfront work and that the kernels could be easily tested in isolation for correctness, and it was chosen for these reasons, even though separate porting and integration stages would be required.

Porting the humidity code went smoothly. However, integrating it with the original porting turned out to be challenging due to some bugs and differences between the two code versions. Once the humidity port was successfully integrated, the work continued with porting the LSM and the multigrid solver. These components were successfully ported, but not yet integrated – meaning they can be run in standalone mode, but produce incorrect results when combined with the rest of the ported code.

As the sprint time ran up to its end, the focus shifted towards merging all the developed changes back into the upstream version of the code. Given the partly unstable character of some of the ported work, flags and guards were added to allow individual parts, like the LSM and Multigrid ports, to be activated and deactivated at will. This allows the correct execution of the original code and allows testing of the newly ported versions on the CI infrastructure, guaranteeing their maintenance over time.

4.2 General Insights

It was useful for the sprint as a whole to have multiple partly or fully disjoint goals, so that if a roadblock was found in working towards one of the goals, or waiting for something else was required, work could be started or resumed in another goal.

There is no straightforward way to integrate ported and non-ported code when they are tightly intertwined throughout the codebase. It can often be the case that an experiment was ported, and it reaches only a subset of the whole executable code. But as porting is extended to include additional processes or more comprehensive experiments, non-ported sections of the code may re-enter the execution flow, leading to an incoherent memory state. Depending on the size and complexity of models and experiments, the required work to get to a coherent state of memory may be very large. Automated tools can help on this process but only up to a certain extent, offering only limited assistance. The option of porting the additional code paths as standalones – employed during the sprint – facilitates this process, but comes with its own costs in the form of integration complexity that can lead to irreconcilable execution states.

5 Results

The first achievement of the sprint was the implementation of serialization logic for the exchange routines, thus avoiding long wait time due to inefficient MPI data type packing. The following result was the fixing of the CI steps focused on the NVIDIA cuFFT library, which are now again routinely run for all merge requests.

Improvements have also been made to the original PALM core porting, most importantly the removal of unnecessary implicit data copies.

In terms of new ported sections, all the humidity code paths were ported and correctly integrated into the original core porting. Additionally, both the LSM and the Multigrid solver were also ported, but were not integrated into the core porting, being semantically correct for standalone execution, but not functionally correct for production GPU runs. Although these could not be correctly integrated, CI steps were created to make sure that the porting work would not degrade over time due to natural gradual changes to the code.

Unfortunately, neither the USM nor the ray-tracing algorithm could be ported within the timeframe of the sprint. In addition, no rigorous performance analysis or deeper performance optimizations were conducted. These tasks remain open for future sprints.

6 Conclusions and Outlook

The PALM GPU porting sprint has been unique in some aspects, with the work being split among two RSEs and over two different periods of time. In total, many smaller goals were reached, and the overall porting state of PALM has been improved and advanced. However, the sprint has been only partly successful since multiple large goals have not been achieved and some of the developments were not correctly integrated into the upstream code.

A number of outstanding tasks still remain to be completed, possibly in a follow-up sprint, these include mainly:

- Fix integration of the LSM into the core porting
- Fix integration of the Multigrid solver into the core porting
- Port the USM and ray-tracing algorithm
- Execute an analysis of the ported code to assess the current achievable performance and possible bottlenecks
- Perform deeper optimizations to further improve the GPU performance

7 References

The PALM model can be found at https://gitlab.palm-model.org upon formal request for access. Further information in published regarding the model can be found at https://doi.org/10.5194/gmd-13-1335-2020, while documentation is available at http://palm-model.org/trac