natESM

Sprint documentation #5

MESSy Sprint

Contact: info@nat-esm.de

Published on 27.06.23 on https://www.nat-esm.de/services/accepted-sprints

1 Summary

The Modular Earth Submodel System (MESSy) is a software project providing an infrastructure for coupling atmospheric models (e.g., ICON, ECHAM, COSMO) and specialized ESM components, the so-called submodels (e.g., physical parameterizations, chemistry packages, diagnostics) via generalized interfaces for standardized control and coupling.

MESSy will be ported to GPU submodel wise. As MESSy contains some computationally expensive process implementations, speedup on the process level can be taken as granted. However, if only individual processes are ported, frequent copies of high amounts of data between host and device will diminish the gain in speed substantially. In order to accelerate the model as soon and as much as possible, i.e., already during the porting process, the data transfer between host and device needs to be optimized, so that the speedup of individual code sections can be decoupled from the memory update.

2 General information

Start and end date:	19.01.23 – 21.05.23
Intended period:	4 months
Responsible RSE:	Enrico Degregori (DKRZ)
Responsible scientist:	Astrid Kerkweg (FZK-IEK8)

3 Sprint objectives

The sprint objective is the memory-transfer design from the base model to the submodels. The following requirements have been considered.

1. Implement the data transfer within MESSy memory manager (CHANNEL) and then allow other submodels to use the memory manager as a backend;
2. Minimize the changes needed for each submodel by the submodel developers and hide the complexity of GPU memory allocation and data transfer within CHANNEL, TRACER and TENDENCY. TRACER is a submodel to structure and manage information about tracers, including the memory allocation, while TENDENCY is a submodel to trace process-based tendencies of prognostic variables and it also works as an interface to access the tracers.
3. Minimize the data transfer between host and device (optimization).

4 Procedure and insights

4.1 Technical Approach / procedure

The work of the sprint was separated into two parts. First, the data transfer was designed and implemented with applications to toy submodels. Then, the new implementation was applied to a dwarf setup with realistic submodels.

In the first part, the data transfer was implemented within MESSy memory manager (CHANNEL), and then an interface was created for TRACER and TENDENCY, which use CHANNEL as a backend for the data transfer of tracers and prognostic variables. The data transfer was then optimized using the "intent" of a field within a submodel (read, write, read/write) and its most updated location (host or device). The “intent” needs to be explicitly specified by the user and the default value is read/write, while the most updated location is internally recorded in the backend. As a next step, the GPU memory allocation was minimized within TRACER, allowing the allocation of a subset of the all tracer set. The assumption is that only a subset of all tracers might run on the device, and thus only for those memory is allocated on the GPU. In order to do that, the tracers are reordered once during the initialization (before the memory allocation) to have the GPU tracers contiguous in memory (requirement of OpenACC standard).

In the second part of the sprint the implementation was applied to a realistic dwarf setup with MECCA running on GPU and JVAL on CPU. The MECCA KPP solver was already ported on GPU using CUDA but the data transfer concept requires that both the submodel core layer and the submodel interface layer run on GPU. The MECCA interface layer was ported using OpenACC and then an OpenACC/CUDA interface was created to call the CUDA version of the KPP solver.

4.2 General Insights

The data-transfer implementation requires to use always the memory manager in the backend, and this allowed to improve the modularity of TRACER and TENDENCY submodels creating a cleaner interface to CHANNEL.

The design requires an explicit definition of internal and coupling variables for GPU allocation and memory transfer. This means that the allocated memory should never be accessed directly but through CHANNEL which provides a pointer to the memory allocation. This concept is consistent with the original MESSy design and it allows to detect some weaknesses or flaws in the submodels modularization, which should be tackled during the integration of the new implementation. The same applies to the tracers memory which should never be accessed directly but always through TENDENCY.

The iteration over the linked list to update any single variable might introduce not negligible overhead in a realistic setup. These overheads were not observed during the sprint but a possible strategy to overcome the issue was already discussed. The idea is to create an additional layer in the memory manager which introduces a mapping of the allocated channel objects once the channel is fixated. This allows to avoid the iteration over the linked list for each memory update.

5 Results

The outcome of the sprint is the initial implementation of the data transfer between host and device within the MESSy infrastructure. The basic idea is that before the submodel execution, each submodel gets a pointer for each data field from CHANNEL and the data transfer is implicitly done within the memory manager. In particular,

- … the memory is updated if necessary.
- … the most updated memory location (host or device) is internally recorded.

The tracers and prognostic variables memory is updated implicitly within TENDENCY, hiding the complexity of the memory transfer from the submodels developers.

6 Conclusions and Outlook

The implementation was applied to two dwarf setups during the sprint, the first one with some toy submodels and the second one with some realistic submodels such as JVAL and MECCA.

The next step is the design of the data transfer from the submodels to the base model at the end of each MESSy entry point. The discussion about possible ideas was already started at the end of the sprint, considering both ECHAM and ICON.

Once the data transfer is fully implemented within the MESSy framework, the final step is the introduction of a base model towards a production setup. A natural choice would be the use of ICON since it is ported to GPU, but in principle the implementation could be applied to any base model. This would allow to test the flexibility of the design and possibly detect weaknesses or flaws.

7 References

A full documentation can be found on the MESSy-project wiki:

https://gitlab.dkrz.de/MESSy/MESSy/-/wikis/Data-transfer

Since it was not possible to have open access to the sprint development and related wiki, to get access to the full documentation, please get in contact with Patrick Jöckel (patrick.joeckel@dlr.de).