Reply to review 2 of "A GPU-parallelization of the neXtSIM-DG dynamical core"

We thank the reviewer for their comments and suggestions. Below we respond to each point.

In the present manuscript 'A GPU-parallelization of the neXtSIM-DG dynamical core (v0.3.1)' the authors test and evaluate different GPU programming frameworks based on their sea ice model dynamical core neXtSIM-DG.

Many modeling groups in the weather and climate community and beyond are facing similar problems as the neXtSIM-DG developers. Developing portable code that achieves good performance on various hardware architectures without limiting the productivity of the (scientific) developers too much is a major challenge. Therefore, the thorough analysis of the different available GPU programming frameworks presented here is of great value to the community. The study is well written and I would recommend publication in GMD after a few issues have been addressed as listed below.

1. *In line 370 in section 4.1 it is stated 'These results indicate that the AMD ecosystem is still less mature'. However, to validate this statement and to have a complete picture also for AMD GPUs it would have been nice to also have a HIP implementation as a baseline to compare the other implementations against similar to the CUDA implementation for NVIDIA GPUs.*
   Reply: A raw HIP implementation would certainly further substantiate this point. However, it is reasonable to expect that in our case a raw HIP implementation would perform similar to the Kokkos implementation. We ended up using only basic features of Kokkos in the main comparison, which are provided through light wrappers around the vendor specific APIs. Considering how our CUDA and Kokkos implementations perform the same asymptotically and how closely the basic HIP API resembles that of CUDA, a raw HIP implementation would likely display similar characteristics.

2. *Table 1: What hardware was used for these measurements and how many OpenMP threads were used?*
   Reply: These measurements where taken on an Intel i9-10900X (10 cores @ 3.7GHz). For OpenMP we determined that simultaneous multithreading is beneficial and used 20 threads. We will add these details to the table.

3. *Figures 2 and 10 and lines 354 and 470: Again, how many OpenMP threads were used for the OpenMP reference simulation? And what backend was used for Kokkos on CPUs? OpenMP as well? And if yes, with the same number of threads as the reference OpenMP simulation?*
   Reply: We got the best performance through full utilization with simultaneous multithreading, i.e. 96 threads, with `OMP_PROC_BIND=spread` and `OMP_PLACES=threads`. For Kokkos (CPU) we use the OpenMP backend with the same settings. We will add this information to the manuscript.

4. *Line 206: LLVM/Clang provides a set of debugging flags (e.g. https://openmp.llvm.org/design/Runtimes.html#libomptarget-info) which can provide precise information about each block of memory and potential problems. Also, for the types that are not trivially copyable, OpenMP 5.0 offers the option of using declare mapper to define this. Wouldn't that have been an option here?*
   Reply: These are good suggestions. With the diagnostics provided by Clang, a working OpenMP offload implementation would likely be doable. Unfortunately, declare mapper

would not help much with memory transfers, since it is only suited for C-style code. Almost all data in our code is held in Eigen::Matrix objects. This type is generic (template parameters include size, data type and storage order) and has a variable memory layout. Depending on whether the size is fully known at compile time or not, the data can be part of the struct or a pointer to the heap. Both data pointers and dynamic size are private members and need to be accessed through method calls, which, as far as I can tell, are not allowed in declare mapper directives. So one would still end up manually converting the buffers to a simpler structure for transfers.

Technical corrections:

- *Table 2: 'AdaptiveCPP' is used here to indicate the SYCL implementation but the name is too generic. AdaptiveCPP is also the name of the compiler and it can also compile native OpenMP or other parallel APIs. I would suggest replacing 'AdaptiveCPP' with 'SYCL-AdaptiveCPP'.*
  Reply: Makes sense. We will rename 'AdaptiveCPP' to 'SYCL-AdaptiveCPP' in the other sections as well to be consistent.
- *Figure 2: Why is in the legend of the right panel TorchInductor marked with an '*'?*
  Reply: These measurements had a small inaccuracy and that would have warranted a comment. In the revision we will update the data. The changes are minor.
- *Line 16: impact on long-term processes*
- *Line 42: is -> it*
- *Line 88: often often -> Remove one*

Best regards,

Robert Jendersie, on behalf of the authors