Reviewer 2

This is a clear well written paper describing a gt4py implementation of the ICON dynamical core, running in the existing ICON Fortran modeling system, enabling k-scale atmospheric simulations on the ALPS GPU supercomputer. The authors describe their porting approach, including thorough testing from the kernel level up to full physics simulations. They provide a sober analysis of the potential of GPUs and their strong scaling limitations.

We sincerely thank the reviewer for taking time to reviewer the paper and for appreciating our work.

I only have minor comments:

1. Section 4.3: what is "the implementation of horizontal blocking"? Does that refer to the loop blocking in the Fortran loops, (which was removed in the Python code?)

Yes. We have also adjusted the sentence (see line 328) to include “horizontal loop blocking” so that it is clearer.

2. Section 4.3: "...testing is tricky as the results are different due to rounding..."

The authors have a good port testing strategy in the presence of roundoff error, but this statement implies that these rounding differences are unavoidable. The E3SM dycore porting work (Bertagna et al. GMD 2019 and Bertagna et al. SC2020) showed that it is possible to obtain BFB agreement between CPUs and GPUs with careful coding, allowing for a different porting approach which simplifies some aspects of code porting.

We agree with the reviewer that enforcing BFB agreement for debugging purposes is helpful. We used it during dynamical core porting by using the -iEEE flag to prevent Fortran from doing non-IEEE 754 compliant transformations of floating-point computations. We also switched off fused multiply adds (FMA) on both sides (Fortran and generated CUDA code in gt4py). BFB was achieved in the vast majority of stencils except for a few, even though the codes were correct. However, since the compute stencils were continuously being combined into larger GT4Py operators/programs for performance tuning through DaCe, we did not think that maintaining BFB reproducibility would add any value in longer run.

3: Section 5.1:

For the final model, I assume all significant code is running on the GPUs, with the dycore using gt4py and the physics using openACC. I believe this is implied, but I didn't see it clearly stated. Were there any software challenges running the two different GPU programming models in the same executable?

Thanks for pointing it out. We have adjusted the paragraph (see lines 371-374) to make it clear. As for running the OpenACC and GT4Py codes in single binary- this was not much of a problem since all memory allocations took place in the OpenACC side. The largest change was to make the OpenACC code work with no-horizontal-blocking.

4. Line 400: "GT4Py synchronization"

I know of two types of synchronization: across MPI nodes, as well as synchronization among thread teams running on the GPU. Which is this referring to?

Thanks, again. It was a mistake. It was the MPI synchronization that took lesser time in the new model. The revised mansucript has been updated accordingly. See line 405.

5. Section 5.1

How does the gt4py code compare with the Fortran code on CPUs? It would be interesting to add CPU-only performance numbers to Figure 7.

Thanks for showing interest on CPU benchmarks. Unfortunately, the structure of the current mansucript is such that the authors did not consider adding CPU benchmarks here. However, we intend to write a mansucript on deliverable 2 very soon, where the focus will be on (performance) portability. We believe that the CPU benchmarks better belong there.