Reply on RC2
Marco A. Giorgetta et al.

Author comment on "The ICON-A model for direct QBO simulations on GPUs (version icon-cscs:baf28a514)" by Marco A. Giorgetta et al., EGUsphere, https://doi.org/10.5194/egusphere-2022-152-AC2, 2022

Reply to the reviewer comment 2 (RC2) from 2022-07-13
Dear Italo Epicoco

Thank you very much for your review comment on our article and your appreciation of the manuscript. Please find below our responses to your comments.

A pdf file (generated by latexdiff) of the differences between the current manuscript and the manuscript that you reviewed is attached. Please note that the current manuscript includes also changes made in response to the comments of reviewer 1. This specifically relates to the sections 6 up to (and including) subsection 6.4.5, which have been modified to account for the redrawn figures 4, 5, and 6, which now present the results in a different order.

Best regards,
Marco Giorgetta

RC2: 'Comment on egusphere-2022-152', Italo Epicoco, 13 Jul 2022

The authors described the activity of porting the ICON-A atmospheric model to a GPU-based parallel architecture using a directive based approach for the parallelisation in order to take full advantage of exascale architectures and improve scientific outcomes resolving physical and climate process down to the scale of a few kilometers.

First of all, I would like to express my full appreciation for a really well written manuscript and for a wealth of information and details.

However some points can be better clarified / discussed

- The GPU approach followed by the authors leverages on OpenACC, although OpenMP is mentioned in some parts of the manuscript (lines 499, 833). My questions are:
Yes, this is the case. The standard applications of ICON-A on the CPU machines at DKRZ (or elsewhere) use mixed MPI OpenMP parallelization.

At the time when we started the GPU port on the Piz Daint computer at CSCS, the only directive based option that existed, and was powerful enough for the ICON code, was the OpenACC implementation for the PGI/Nvidia compiler. There was no OpenMP5 implementation available.

But in the meanwhile additional work has started for a GPU port based on OpenMP5 targeting ICON applications on the LUMI supercomputer. This work is still ongoing.

OpenMP is kept in the code because the ICON code is still in use on CPU machines. Thus we currently have MPI x OpenMP for CPUs and MPI x OpenACC for GPU machines with OpenACC/PGI support. (Ongoing work currently investigates if the OpenACC port of ICON can be modified such that it works also on AMD/Cray systems.)

The authors should clarify these aspects to better justify their choices.

The introduction is now extended on line 62 as follows:

... In the CPU case applications shall continue to use the proven parallelization by MPI domain decomposition mixed with OpenMP multi-threading, while in the GPU case parallelization should now combine the MPI domain decomposition with OpenACC directives for the parallelization on the GPU. OpenACC was chosen because this was the only practical option on the GPU compute systems used in the presented work and described below. Consequently the resulting ICON code presented here includes now OpenMP and OpenACC directives.

- in Section 4.4 Line 520 the authors first introduce the concept of reproducibility which will be better discussed later in Sections 5. In Section 4.4 the meaning of the word "reproducibility" is not clear. Are the authors referring to the bit-identity reproducibility or tolerance-reproducibility? How was reproducibility evaluated in the context of physical parametrization (Sec 4.4)?

Yes, "reproducibility" means bit-wise reproducibility. The text is changed on lin 533 to clarify this:

... This bit-wise reproducibility is important in the model development process because it facilitates the detection of unexpected changes of model results, as further discussed in Sect. 5.

Moreover, the authors uses the "ACC LOOP SEQ" directive to "fix the order of the summands" but it is not clear why this is needed; what is the correct order to do a summation. Considering that the round-off error is inherently present in the code, even in the sequential version of the code, why should summation follow the "LOOP SEQ" order?

The critical point is that we want to ensure that the sequence is maintained so that round-off effects remain unchanged if the computation is repeated, with the goal to obtain a bit-wise reproducible code.
- In Section 5 the authors deeply discussed the validation techniques available for ICON. Namely, in Sect 5.3 the tolerance testing is presented, which consists of evaluating an ensemble obtained by perturbing the state variables with a uniform error of the order of magnitude $10^{-14}$. My comment here is that the main source of divergence in the outputs, when implementing a parallel version of a code, is due to the round-off error that can grow after several time steps. In order to evaluate the effect and impact of the round-off error it is probably best to create an ensemble by changing the order in which the grid cells are evaluated, such a by shuffling the arrays with the grid cells.

Certainly there exists more than one methods for creating an ensemble of simulations. But a variation of the order of the evaluation of the grid columns alone would not change any result in the ICON-A integration, in absence of bugs in the parallelization or blocking. This bit-wise reproducibility wrt. parallelization (number of MPI processes, OMP threads) and blocking is regularly tested. Therefore the simplest method was to add numerical noise to the state variables.

- In Section 6.5.1 the authors should provide a comment on why the radiation exhibits a super linear strong scalability on PizDaint and not on Juwels-Booster neither on Levante.

This is explained in section "6.5.1 Strong scaling of components". The reason is that the sub-blocking length for radiation, controlled by the rcc parameter, can be used on Piz Daint and Juwels-Booster to maintain or increase the work load per radiation call, within some limits, while the work load per call for all other processes simply decreases by a factor of 2 per node doubling, as does the number of columns per compute domain. Differences between Piz Daint and Juwels-Booster exist in the number of doubling steps where a factor of 2 reduction in rcc can be avoided. This limit is reached when rcc has grown to the number of points per domain. For Piz Daint this limit is reached only at the highest parallelization (1024 nodes), while on Juwels-Booster this limit is reached already on 32 nodes. This provides the simplest explanation for the nearly constant strong scaling of radiation, even slightly above 1, on Piz Daint, and the transition from a constant strong scaling just below 1 up to 32 nodes to a decaying strong scaling on Juwels-Booster. But we do not know why the effect of maintaining or increasing the radiation work load even leads to super linear strong scaling on Piz Daint, while we find only a strong scaling of just below 1 for Juwels Booster. Maybe it is related to differences in the overhead costs for setting up the parallel loops on the GPU.

On Levante we use nproma = rcc = 32 for all experiment so that the work load per radiation call is also fixed. Here the strong scaling of radiation depends on other factors, as it is also the case for all other processes.

To clarify this the section "6.2 Optimization parameters" with respect to the sub-blocking for the radiation and section "6.4.3 Strong scaling of components" have been modified. Note that these sections have already been changed in response to referee 1.

Why the transport and the vertical diffusion have a super liner strong scalability on Levante? The vertical diffusion has a really strange and counterintuitive behaviour since its scalability curve increases with the number of nodes.

This is indeed peculiar. From our log files and timing data we cannot derive an explanation. Our speculation is that this is resulting from cache effects. We did not investigate this behavior further, first of all because these effects do not distort the overall scaling behavior shown in Figure 5, where the main difference occurs between the GPUs on the one hand and the CPU on the other hand. Secondly, diagnosing cache efficiency is non-trivial and can become a study on its own. Therefore we did not investigate the underlying reasons. Thus we only point out these behaviors in the manuscript on line 915 to 955.
- In Section 4.3.1 Line 322 the sentence "There are code divergences in the non-hydrostatic solver" is a bit misleading since it is not clear whether it refers to thread divergence or code differences between CPU and GPU.

*Code differences between CPU and GPU are meant. The text in the manuscript is changed to make this clear (line 331).*

- Listing 2 reports an example to explain the use of scalars on GPUs instead of arrays, but the transformation of 2D array into a scalar is not fully clear; namely, in the expression for the scalar (line 331-333) the index jk-1 is used while in the expression (lines 336-338) the index jk is used. Moreover is also unclear whether the "z_w_concorr_mc_m1" values are used/needed after the do loop; if these values are not used outside the loop probably the scalar transformation is also useful for the CPU case.

*Indeed, Listing 2 was simplified to a point where it no longer sufficiently illustrated the intended point of using registers to replace arrays. We have now added the full loop, in which both the scalars z_w_concorr_mc_m0 and z_w_concorr_mc_m1 are calculated and consumed, and we have added Fortran comments explaining the code which they replaced. This should provide a full explanation of this optimization.*

- In the abstract and in conclusions the authors write that the model exhibits a good weak scalability. But after a careful reading and according on what is stated in Section 6.5, "ICON exhibits very good weak-scaling for a 16-fold increase in node count", actually, a complete weak scalability analysis has not been provided as the weak scalability has been evaluated only in the case of 16-fold increase. I suggest that the same comment is also report in the abstract and conclusion.

The abstract and conclusions now include: "... over the tested 16-fold increase in grid size and node count ..." so that the statements are now more precise.

More cosmetic comments, suggestions and typos:

- In the abstract, line 8, it is better to use "kilometres" instead of "km"

*Done*

- Line 17-18: there is a pun in the sentence... the weak scalability is good and the strong scalability is weak

*Changed to "... good weak scaling ..., the strong scaling on GPUs is relatively poor, ..."

- Line 116: "(black)" should be "(blue)"

*Thanks for spotting the error, corrected.*

- Line 272: "data types" should probably be "data structures"

*Yes, changed.*

- Listing 6 and Listing 7 have exactly the same caption. I suggest to merge together both listing or to differentiate the captions

*Thanks for pointing out the issues with these two listings. Both are needed to illustrate the 2 possible communication modes (use_g2g), but the UPDATE was missing in Listing 6. Now all fixed: they illustrates the two possibilities and the captions have been changed accordingly.*
- Line 702: "ptest" mode is mentioned here for the first time, it would help if, in the same sentence, the authors anticipate that the mode is described in the following section.

Done. This sentence is now followed by: "Details for these methods are given in the following subsections."

- Line 787: "Ss and Ss" should be "Ss and Sw"

Thanks for spotting this error, corrected.

- Line 919: "1 SDPD" should probably be "1 SYPD"

Thanks for spotting the error, corrected.

Please also note the supplement to this comment: https://egusphere.copernicus.org/preprints/egusphere-2022-152/egusphere-2022-152-AC2-supplement.pdf