EGUsphere, referee comment RC2
https://doi.org/10.5194/egusphere-2022-943-RC2, 2022

Comment on egusphere-2022-943

Anonymous Referee #2

Referee comment on "Pace v0.2: a Python-based performance-portable atmospheric model" by Johann Dahm et al., EGUsphere, https://doi.org/10.5194/egusphere-2022-943-RC2, 2022

Summary:

This paper describes the development of Pace, a finite-volume atmospheric model written in Python. The paper describes the DSL designed to produce fast compiled kernels for the required finite volume computations. Some scaling results are presented, and the code is shown to perform favorably when deployed on GPUs versus the reference CPU-based FORTRAN code.

Review:

Overall the paper nicely describes the development process and approach for constructing efficient an efficient finite-volume solver using Python. The paper would certainly be a valuable contribution to the literature of finite-volume atmospheric modeling. The main drawbacks of the paper are that the example problem and strong-scaling tests are fairly limited in size and should likely be expanded, and there should be more discussion of alternative approaches in both the general-PDE-solver and GFD-model spaces, as well as more contextualization as to why the presented approaches were chosen.

Suggested revisions:

Further discussion of other Python acceleration techniques would help put GT4Py into context. Cython and Numba are mentioned in passing, but why were they not chosen for the FV core? What about JAX? Did the authors try any of these? Were there reasons from the outset that these approaches would be limited relative to the GT4Py approach? More discussion here would be very useful to other authors interested in moving other high-performance codes to Python.

More discussion of other Python and Julia-based PDE solvers and GFD models would also help contextualize the solver. Popular Python-based (or Python-interfacing) PDE DSLs such as FEniCS, Exasim, and Dedalus might be mentioned. Also other atmosphere and ocean models being implemented in high-level languages on GPUs, such as CliMA and Veros, should be referenced and contrasted. Again I don't think direct simulations comparisons are necessary, but discussing these projects and how the Pace developers see their code in relation would be very helpful to others.

The scaling tests are a little confusing. The exact setup (nodes vs ranks and total model degrees of freedom) are scattered throughout the text, but should be stated clearly in each figure/caption. The weak scaling results look particularly impressive, but again the details aren't clear -- the plot says number of nodes, but the text refers to the left-most point is refereed to as "6 ranks". The text says that the Fortran reference is ran with 6 MPI ranks per node, but doesn't specify this number for Pace. Is this the same, or is it 1?

Finally, the strong scaling test leaves a lot to be desired and should be expanded. If it's possible to run Pace up to 864 nodes, then it would be much better to see a broader strong scaling test that illustrates the opposing limits of fitting the problem in GPU memory vs. having too little local work for the GPU to do. Understanding the window of local-work-per-node required for maximum performance is very important to potential users, and seeing the efficiency penalties either side of the optimum is also key.