## Reviewer 1, Second Revision of Manuscript, “Julia for Geophysical Fluid Dynamics: Performance Comparisons between CPU, GPU, and Fortran-MPI”

This is a revised submission. The authors have addressed most of my comments. I especially welcome the new “Julia in a Nutshell” section, which is overall very well done. I recommend acceptance after minor revisions.

### Minor Comments:

1. Line 74: Julia’s “superior memory management”: I don’t understand what the authors mean here. Julia is a garbage collected language which trades off programmer control for ease of memory management. Garbage collection can have undesirable effects in parallel applications (see for example this Julia issue https://github.com/JuliaLang/julia/issues/49316). It is not strictly superior to other memory management mechanisms.

   **Response:** We have removed “superior memory management” from this description.

2. Line 80: “Julia has found widespread application . . . ”: I think this statement is too strong. Julia definitely does not have “widespread” application in web development, and my impression is that it is only gaining grounds in the other fields mentioned.

   **Response:** Agreed. We changed “found widespread application” to “recently been gaining ground”.

3. Line 126: Strictly speaking Array is not a concrete type since it is parametric (this can be easily checked in Julia by evaluating isconcretetype(Array) which returns false). Additionally, here and in other places Julia types are written in plain text, but other times they are displayed as code. Please unify the style throughout the manuscript.

   **Response:** We have added specific text on how to make an Array concrete, which is by defining it with concrete-typed elements and specifying the Array’s size. We have changed code within the text to code font.

4. There is a typo in equation (1a) (there should be only one gradient operator).

   **Response:** Thank you for pointing it out. We have removed the additional gradient operator.

5. I have a couple of suggestions for Table 1. Please mention that the TDP values are per CPU and per GPU. The A100 GPU comes in two variants with TDP of 300 W or 400 W. I believe that Perlmutter has the 400 W version. To be consistent with the GPU specification, please say that the CPU flops are for double precision.

   **Response:** Thank you for the suggestions. We have added these, and confirmed with NERSC that the TDP is 400W.

6. Lines 510-514: “Based on technical specification . . . ”: The comparison of flops values implies that the authors think that their code is compute bound. Based on the low number of operations I would expect the code to be bandwidth limited and the bandwidth ratio to be a more appropriate speedup bound, at least for large problem sizes. Can the authors comment on this?

   **Response:** We added text to that paragraph to describe the timing of computation and communication separately. Based on Figure 5, the full-node Julia-MPI (64 cores) is 80% computation, so we do not believe the application is bandwidth limited on the 512x512 by 100 layer domain.

7. Line 618 “Julia-GPU scaled very similarly to Julia-MPI”: Maybe “performed” would be better here than “scaled” ?

   **Response:** Agreed. The text has been updated.

8. Line 622: “(. . . ) and sample results rarely, GPUs can offer significant speed-ups”: This might be true in theory, but I don’t see how the presented results support that conclusion. Looking at Figure 2, even

in the “Computation Only” plot the CPU is faster than the GPU, even though the GPU is theoretically much more powerful.

**Response:** Thank you for pointing this out. We have revised this paragraph to align with the results in the GPU/CPU comparison of the paper.