Reviewer 2: Revision of Manuscript, "Julia for Geophysical Fluid Dynamics: Performance Comparisons between CPU, GPU, and Fortran-MPI"

The authors are grateful for the reviewers' insightful suggestions, which have contributed to the improvement of the manuscript. The major change is that the Julia-GPU, Julia-MPI, Fortran-MPI simulations have all been retested on Perlmutter at NERSC, so that comparisons can be made directly between single-node CPU and single-node GPU. Perlmutter uses AMD EPYC 7763 CPU nodes and NVIDIA A100 GPUs, so is newer and more relevant to readers than the previous tests on Cori. We have also added more introductory material on Julia, a longer results and discussion section, weak scaling plots, and plots that directly compare full-node CPU and GPU performance.

Major Comments: The authors have made significant contributions by developing a shallow water solver using Julia language and comparing its performance with a solver written in Fortran. Furthermore, they have successfully implemented their solver on a GPU, demonstrating a remarkable speed-up. While the overall results appear promising, I would suggest considering the following points to further enhance the paper:

1. In section 3.2, it would greatly enhance the paper to include a table comparing the specifications of the CPU and GPU used in the simulations. This table should provide a comprehensive comparison of various factors, such as FLOPS (Floating-Point Operations Per Second) and memory bandwidth, specifically for both 32-bit and 64-bit computations. Additionally, it would be valuable to summarize the versions of the toolchain that were utilized during these computations. This information will provide readers with a better understanding of the hardware and software environment in which the simulations were conducted, allowing for a more comprehensive evaluation of the results.

   Response: Please see the new section 3.7, Hardware and Compiler Specifications. Table 1 has been added to describe the hardware, which is now a compute node versus a GPU node of Perlmutter at NERSC. Several paragraphs were added on the toolchain for both Fortran and Julia.

2. In section 3.2, it would be beneficial to include a comparison of the performance between the Julia code and the Fortran code in a single-core execution. This comparison will provide readers with insights into the optimization of the Julia code for serial computation.

   Response: We have extended the comparison between Fortran and Julia to a single core, with the results illustrated in the strong scaling plots in Figure 3.

3. In Section 3.2, the authors mentioned that all codes were executed in double precision and highlighted the faster simulation on the NVIDIA RTX8000 GPU compared to the CPU. However, it is important to consider that the RTX8000 is primarily designed for consumer applications and may exhibit slower performance in double precision computation. To provide a more comprehensive evaluation, it would be valuable to compare the computation on a high-performance computing (HPC) targeted GPU, such as the NVIDIA TESLA A100, which is known for their robust performance in double precision computation and are specifically designed to excel in HPC workloads. Otherwise, please compare all simulations in single precision.

   Response: This is a great suggestion. We switched all of our performance comparisons to Perlmutter at NERSC, which came on line this year. We chose Perlmutter in order to use an HPC-targeted GPU, the NVIDIA TESLA A100, as suggested here.

4. In section 3.3, it is evident that Julia-MPI outperformed Fortran-MPI in terms of computation, but it took more time for communication. To provide a clearer understanding of the experimental setup, it would be beneficial to specify the Fortran compiler and Julia interpreters, along with the related toolchain, that were employed in the study. Additionally, it is important to mention the specific version of the MPI library used for both the Fortran-MPI and Julia-MPI implementations. This information will help readers better comprehend the underlying MPI libraries utilized in each case and the potential impact they may have had on the communication performance.

Response: Please see the paragraphs on toolchains for Fortran and Julia, added to the new section 3.7.

Moreover, it is worth exploring the possibility that different MPI libraries might have been employed for the Fortran and Julia codes. If this is the case, it should be explicitly stated in the paper, along with the versions of the MPI libraries used for each implementation. Clarifying this aspect will enable readers to consider any discrepancies or optimizations associated with the MPI libraries employed in the Fortran and Julia implementations.

Response: We have added the MPICH version to the paper in section 4.3, which is version 4.0 for Julia and 3.4 for Fortran. Unfortunately, we were not able to compare with other versions as we were limited to the modules available on Perlmutter.

5. I think hyper threading may be disabled in supercomputer. It would be helpful to omit the hyper-thread performance of the CPU in section 3.3.

   Response: We have removed the section on hyper-threading performance.