

The authors thank the reviewers for the helpful comments, which improved the content and readability of the paper.

Reviewer #2

The manuscript “The Ocean Model for E3SM Global Applications: Omega Version 0.1.0. A New High-Performance Computing Code for Exascale Architectures” describes the design and testing, and demonstrates the performance and performance-portability of Omega ocean model which is supposed to replace MPAS-Ocean in the future. The model solves the SWE with passive tracers and is therefore in the early phase of developments.

The manuscript is well organized and has all the contents that I would expect from a model documentation paper for modern architecture. It describes usage of Kokkos in a Ocean model, which is probably one of the first attempts. I recommend publication, as I have only a few minor questions/comments and suggestions.

1. Writing style: I strongly suggest the authors to review the introduction. It reads more like a personal experience than a scientific introduction. Specifically, the three options for model port is described in 3 large paragraphs expressing in details the concerns of the group involved. I think the community is well past that stage and does not need a long introduction of status-quo.

Author response: The introduction was substantially updated to be more formal and concise. We kept an updated version of the discussion of programming model options to justify the significant shift to C++ and Kokkos. We believe this is useful background to other ocean model development teams as well as emphasizing the amount of effort that can be involved in these computational transitions.

2. Line 53 and few other places- good to avoid using “our group”, “we”, if possible.

Author response: Done. Pronouns have been removed throughout.

3. Why did the authors decide to publish at this early stage? Why not wait for the full ocean model to be ready?

Author response: Writing a completely new ocean model code is a large undertaking that will span five years. We felt that it was important to document our approach and performance results at the milestone of the shallow water equations, which is halfway through. Feedback from reviewers and collaborators on a publication is useful at this stage, and gives us the opportunity to present the framework design and tests. In addition, it is helpful to our managers and funders to see peer-reviewed approval of the basic design. There are previous examples of publications on shallow water cores before developing the full atmosphere or ocean model, such as Ringler et al (2010).

4. Line 54: Authors write that MPAS-Ocean is only half OpenACC ported because of the code structure. Could you please elaborate?

Author response: Please see the response to the similar comment by reviewer 1 - In short, it was due the use of community based packages for tracer tendencies, the complex data and loop structure for tracers and tracer groups, and the complexity and communication heavy split-explicit barotropic algorithm. Relevant text has been added.

5. Lines 90+: “The choice of Kokkos required our”- it again reflects your experience. See if the sentence can be rephrased to make it more objective.

Author response: Done.

6. Line 105+: It seems that the team has added additional layer of abstraction on top of kokkos. What will happen when these people leave? Is there a strategy behind?

Author response: The additional layer of abstraction is actually very simple and mostly reduced the syntactic complexity and some of the Kokkos jargon by creating aliases for Kokkos views and reducing the number of arguments with some standard choices. It did not require significant computational expertise and was primarily to make the main code easy to read, as in Figure 2. We also provide templates and documentation to help domain scientists write code in these abstractions. We have added some additional text to describe this briefly.

7. Related, for my own understanding: Why did you decide to write the code on your own (domain experts) when there is a trend in the community to get it written by the software engineers?

Author response: It has been our long experience that software engineers can be difficult to retain over the long term and tend to get hired away by vendors or industry. We have in the past lost critical infrastructure developers, so we wanted core developers to be somewhat fluent in the programming model and strategy behind coding choices. Having said that, we are relying on the larger Kokkos effort and our local team still retains a mix of developers with deep computational physics experience. In particular, co-authors Kim, Mametjanov, Sreepathi, and Waruszewski have strong credentials in software engineering and performance tuning.

8. 11: is not discrete but the line above it says it is. Please check.

Author response: Thanks. Those are discrete in the horizontal, as notated by the subscripts e , v , and i . The word ‘horizontally’ was added.

9. Line 164: “operator convergence rates” – do you mean grid convergence of individual operators? Please check.

Author response: Yes. Text is updated, thanks.

10. Lines 190+ on multiple domain decomposition- I understand the benefits of supporting multiple domain decomposition but that sentence is not clear to me. Could you please rephrase?

Author response: Done.

11. Line 202: “We have” – try without we.

Author response: Done.

12. Trott et al., 2022b has been cited a few times for Kokkos. Please check.

Author response: Second reference was removed from bibliography.

13. Line 251+, for my own understanding: It is mentioned that a manual tuning of kernel execution resulted in 10-20% performance gain as opposed to MDRangePolicy. How and when did the team realize that more can be achieved avoiding what kokkos offers?

Author response: For a brief period the Omega team considered adopting a different performance portability library called YAKL. A prototype shallow-water solver was developed using both YAKL and Kokkos. The loss of performance from using MDRangePolicy was exposed by comparing the two prototypes. The current strategy of linearizing the index is what the YAKL library does internally. Note that the Kokkos developers are aware that MDRangePolicy can result in sub-optimal performance for some applications and are currently working to address this. To indicate that this workaround might soon not be necessary, a footnote mentioning the on-going Kokkos work was added to the paper.

14. Lines around 275: Author mention that the ability to fuse or to not fuse functor is advantageous as it allows one to try out things. I am wondering if it is practically possible if one has several functors and possible combinations to try out?

Author response: The reviewer is right that trying out every possible fusion combination may be labor intensive when there are many functors. In that case, it is usually possible to limit the number of combinations that need to be considered by performing hardware profiling. Light-weight functors that don't use many registers can be candidates for kernel fusion. Additionally, algorithmic knowledge might be used to limit the number of combinations. For example, only those kernels that operate on shared data can be considered for fusion, in hopes of better cache utilization. A sentence stating this has been added to the paper.

15. Figure 18 and related text: the comparison with MPAS-Ocean on GPUs is unfair since it is not full ported and likely not tuned. I think it is fine to use it as a reference but I would suggest mentioning it explicitly in the text.

Author response: A sentence stating that MPAS-Ocean was not fully ported to OpenACC was added just before Figure 14.