Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2018-334-RC3, 2019

Geoscientific
Model Development
Discussions
Open Access

# *Interactive comment on* "Scalability and some optimization of the Finite-volumE Sea ice-Ocean Model, Version 2.0 (FESOM2)" *by* Nikolay V. Koldunov et al.

**Mark Petersen (Referee)**

mpetersen@lanl.gov

Received and published: 6 May 2019

This paper evaluates the performance characteristics of FESOM 2.0 using three resolutions and four machines. It will be a great reference for all ocean model developers, because it explains the root causes of ocean /sea ice bottlenecks, and discusses the options available to those designing these algorithms. The two slow-downs are the SSH solver and the sea ice because they are two dimensional yet require frequent communication, so require a large number of small MPI messages. My group struggles with exactly the same issues, and we have gone back and forth between implicit and split-explicit barotropic solvers over the years in POP and MPAS-Ocean. In fact, we are currently testing improved filtering methods for split-explicit as well as implicit methods in MPAS-Ocean. Figures 2-4 beautifully illustrate the crux of the matter: these two stubborn pieces don't scale.

The paper is very well written. The introduction provides useful context and motivation for the topic. Later sections clearly lay out the choices available to the developer and the corresponding results. The algorithm write-ups are a concise summary of complex code. The authors paid attention to important details – for example, the scaling plots are very well labeled and are easy to interpret, even while displaying a lot of information. I particularly appreciate table 3, as it gives us a rough idea of performance comparisons between models, despite the potential mis-matches between models. The performance of FESOM 2.0 is very impressive, and sets a high bar for throughput for other ocean models.

Table 3. It is possible to go a step further, and compute SYPD/vertices/levels/cores or SYPD/(active 3D cells)/cores. We know that the 2D solvers are a good fraction, so SYPD/vertices/cores is also helpful. I find myself trying to estimate comparisons by eye, taking into account cores and layers. The addition of these columns, though still an imperfect comparison, would be helpful. For a single model with perfect strong scaling and perfect weak scaling, this is a single number. Indeed, when I've made plots of these values in the past for multiple resolutions and core counts, everything collapses to a single number within a reasonable range of core counts. The exception is the factor of the number of layers, which doesn't scale linearly with SYPD. The number of degrees of freedom (DOF) is also different between triangle, quad, and mostly-hex meshes, so I often normalize by the number of prognostic DOF the model solves for, i.e. (number tracers)*(number tracer points) + (number of velocity DOF) For example, a quad B-grid like POP has two velocity DOF per cell, but a hex C-grid has three.

A design choice of whether to solve an extra equation for thickness also influences performance. In a pure z-level model like POP, only the top layer thickness varies for the free surface, so almost no computation is required – we only compute vertical transport by solving the continuity equation. Once we moved to a thickness equation and ALE in MPAS-Ocean, we actually solve for the thickness as an extra tracer (effectively one) using identical algorithms, in order to ensure consistency between advection of tracers and thickness. This is mathematically elegant and truer to the conservation equations, but I suspect it is a waste of computation because the ALE routine has already decided where the layer interfaces should be placed (z-star is similar, since thickness may simply be diagnosed). Of course, these inefficiencies may be a worthwhile price to pay to make the code more general, because it is now easier to implement a non-Boussinesq code where we advect mass rather than thickness (i.e. area-normalized volume). It is also now feasible to use true ALE and only remap every n time steps. For the purpose of this performance paper, it would be useful to know if you had similar considerations, and what factors influenced your decisions. Do you solve a thickness equation (included in tracer timers) or do you simply diagnose layer thickness?

In figures 2-4 the total is split into four pieces, which is perfect for those purposes. It would be useful to have another plot that splits ocean tracer and ocean velocity into their smaller components. For example, within ocean tracer, what is the percentage of each: advection, determining vertical mixing coefficients, implicit vertical mixing, GM, equation of state, etc. As a fellow ocean model developer, this helps me find potential improvements in my code, and I could publish a similar plot for comparison. A bar chart would illustrate this quite well. This could all be done as a sample from just one domain, one machine, and one node count, as I think it does not vary much for these two that scale so well.

For the SSH solver and sea ice, we have separate sub-timers around communication and computation. An additional scaling plot with these two separated out would show the communication problem even more strongly, as it should all be in the communication. Again, a single machine and domain would illustrate the point.

Why did you choose 47 layers? Layers that are a multiple of 8 or 16 are often recommended for memory access and vectorization (and you are very close!). In fact, there is a method called strip-mining where you always loop down to a multiple of 8 in the vertical, and multiply by a mask for land cells below ocean, just to get the cache alignment. Did you consider any of these issues in your original FESOM design or performance improvements?

On a related note, it's worth mentioning in this paper the index order and your reasoning for it, as it is an important performance consideration. I'm sure you mentioned it in some other papers, but it is relevant here too. I assume all unstructured models use the vertical as a tighter index than the horizontal, as that's "all we've got" for exact memory alignment. A secondary choice is whether to make a 'super array' for tracers, where you use a tracer index to loop through all the tracer variables, rather than individual tracer variables. If you did this, what order did you place the indices, and why?

Did you consider trying to make your halos wider for less frequent communication? That doesn't work for your implicit SSH solve if it requires a global communication, but it is a possibility in other parts of the code. Would be good to comment on this design choice in the text.

My other comments are very minor. I've attached a copy with grammatical corrections. 1. Does fArc mean something? 2. Fig 7, 9: Add perfect scaling line 3. Table 3. Label unit for time step

Please also note the supplement to this comment:
https://www.geosci-model-dev-discuss.net/gmd-2018-334/gmd-2018-334-RC3-supplement.pdf

---