Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2018-280-AC4, 2019

Geoscientific Model Development Discussions

Interactive comment on "Improving climate model coupling through a complete mesh representation: a case study with E3SM (v1) and MOAB (v5.x)" *by* Vijay S. Mahadevan et al.

Vijay S. Mahadevan et al.

mahadevan@anl.gov

Received and published: 3 March 2019

Dear Reviewer,

Thank you for the extensive review comment. Please find our detailed responses for all the issues raised in your review comments. We have included most of the suggested changes in the manuscript, and will be uploading the latest copy to the website. Rebuttals for specific questions have also been included in the response here and we hope that it will provide better context in light of the new changes added.

1 General Comments

- Referee: The main body text where references are given needs to be reformatted. The references and text are not clearly separated and make them difficult to read. There is also a reference to "Section. (1)" on Page 6, line 6, that I believe suffers from the same formatting problem?
  Authors Done. We replaced "cite" with "citep".

- Referee: The outstanding questions that are not answered in the paper are (1) can the weights generated online be counted on to produce error free interpolation (conservation, monotonicity, etc) without first being reviewed and validated offline? (2) is the weights generation capability robust and reliable enough to run on different platforms and expect the same results to at least roundoff? (3) is it faster to generate weights online vs reading them in? (4) Is there some benefit to generating the weights online and then being able to reuse them as compared to regenerating them each time the model is run with regard to performance or reproducibility? It would be helpful if the paper addressed these issues if possible. These issues are partly raised in a few places in the paper, at least Page 6, Lines 25-27 and Page 18, Lines 9-10. Some addiitonal discussion/results might be interesting.
  Authors We have made changes to the manuscript in the Background, Software and Results sections to raise these questions and to address the solutions appropriately as needed. Detailed discussions have also been provided in a previous reply to the reviewer comments.

- Referee: The MCT gsmap is generally compressed significantly because the information can be defined via a single start and end ID for certain kinds of decompositions. Since the MOAB mesh carries more info, I assume that compression is not possible and that the mesh consists of "n" fields of data for each gridpoint/corner/edge/etc? Is that a lot of data? Does the memory scale at all at

higher resolutions and higher pe counts? I'm sure much of this is documented in MOAB papers, but it would be nice to add a sentence or two about it in this paper.
Authors Some discussions about the mesh storage and memory requirements for serializing field DoF data on the MOAB mesh has been added. Again, a detailed discussion was provided in the previous response and we can add to it if further clarifications are needed.

- Referee: In Figure 1, it looks like there is no longer a coupler. Where are the non-coupling non-mapping coupler operations (merging, atm/ocn flux, diagnostics, etc) being computed? In text, it sounds like the coupler component still exists but that the underlying MCT datatypes were swapped for MOAB datatypes, an additional set of calls were added in the component coupling layer to more fully describe the meshes, the online weights generation was added, and the online sparse matrix multiply was converted from MCT calls to MOAB calls. But then at page 17, line 12-15, it sounds like the coupling is between pairs of components excluding a coupler. It would be good if this were clarified.
  Authors Quoting from our previous response: "The hub coupler still exists. We are currently duplicating the MCT calls alongside the MOAB based coupler in order to fully verify and validate both the accuracy and performance at runtime. After full validation, the MCT coupler will be completely removed from E3SM. The MOAB coupler allows the possibility for ATM to directly compute the remapping weights to project field data to OCN since the intersection will then be carried out through migration of OCN mesh to ATM pes. Hence this pair-wise coupling leads to a more distributed coupling strategy in the future. However, we do envision that there will still be a thin layer of a global coupler, even in the distributed case, to drive the subcycling, to compute merging with weighted combinations of fluxes, for validation and other diagnostics data outputs. We understand that Fig. (1) is somewhat misleading in this context and intend to make modifications to make it clearer.". Clarifications have been added to the text along these lines.

2 Technical Corrections

- Referee: Page 6, line 20 "oas (2018)" ?
  Authors Fixed.
- Referee: Page 7, line 29 fix "a in-line", should be "an in-line"
  Authors Changed "a in-memory" to "an in-memory"
- Referee: Page 8, line 7 Alg. 1 -> Algorithm 1
  Authors Fixed.
- Referee: Page 11, Fig 5b. It seems unlikely that the trivial decompostion would be someone's first guess for best performing decomposition with knowledge of how the coupling/ mapping work. Having said that, I'm surprised it performs as well as it does in Figure 14. There are lots of other resonable decompositions, why were Trival and Zoltan chosen to be highlighted in this paper? And why does the trivial decomposition perform so well in Figure 14.
  Authors This was another particularly interesting result from our scaling studies. The triival partitioner is not particularly the best strategy, but from an implementation stand-point, easiest to get working. However, we expected the Zoltan repartitioner to provide much better scaling and overall speedup (in terms of time) when computing the remapping weights by minimizing the source coverage mesh communication time. But, this problem is particularly tricky, since there are two parts that have to be optimized simultaneously.

  1. Migration from component to coupler requires repartitioning,
  2. computing coverage mesh requires moving source mesh elements to cover local target elements.

  So even if one partitioner is optimal for migration, it may still require moving lot of elements for coverage computation. There are ways where we could simultane-

ously optimize the partition for all components (source/target combinations) while at the same time taking into account the PE layouts, but this implementation is more involved, and is a work in progress at this stage.

- Referee: Page 12 line 23, remove "is" in "is results in"
  Authors Done.

- Referee: Page 16, line 28, bit-for-bit capability is sometimes important to achieve, certainly for identical runs, also for runs on different pe counts (sometimes with a performance penalty via an optional flag). This sentence left me asking what the bit-for-bit capabilities are and what risks are introduced when computing online versus reusing.
  Authors Agreed. If the performance penalties are not an issue, potentially exact bit-for-bit runs can be performed with the MOAB intersection. While we have not noticed any variation in the actual supermesh computation, the element sequence in the resulting supermesh will have to be re-sorted so that it is always partition agnostic. Currently, this is not strictly enforced. Additionally, any and all reductions in remapping weight computations, enforcing conservation and performing A*x, where A is the weight matrix and x is the solution vector to be projected need to be handled carefully to preserve unique order of arithmetic necessary for bit-for-bit reproducibility. Hence our statement that this is non-trivial, though necessary in the longer run as an explicit option.

- Referee: Page 19, Fig 9. it would be nice if the scale were not so ad-hoc and instead something more like (0,80,4). Scales like the one shown just make the figure more difficult to digest and in this case, there is no benefit to have the breaks defined as they are relative to something simpler and easier to read. Also, I'm not sure color adds anything, I think the same could be shown via a contour plot, possibly clearer and simpler still.
  Authors We originally made use of contour plots but it made the appearance

much less easier on the eye. The issue with presenting this data is that its a 2-D data set showing the timings for combination of source/target element combinations. We could use 3-D plots to show surfaces aligned to the computation time but drawing conclusion from such a description was not obvious. The reasoning for the chosen scale in Fig. 9 is that around 80 secs was the maximum amount of time (upper bound) for the largest source-target element combination to run ESMF in our case. The coloring provides a relative comparison with respect to this upper bound, and shows as you have lower target elements, all libraries perform well relatively; but when there are lot more target elements, the algorithmic differences become much more obvious.

The loop over target elements is typically the sequential part in the computation. We have stressed in multiple places how we can accelerate by using OpenMP threading or task-based programming models specifically for intersection computation and also in the TempestRemap online weight matrix generation. While we don't have any results at the moment to show performance gains with such hybrid implementations, we expect to leverage the finer grain parallelism in the next iteration of the implementation refinements.

- Referee: Page 18, line 29 "serial runs" vs page 20, line 1 "better performance in MBTempest : : : offers avenues to incorporate task level parallelism : : :". Are these serial runs or something else? Serial in MPI but using shared memory parallelism? Is that still serial?
  Authors Yes we are referring to serial in MPI but parallelism introduced either through threads or task-based programming. Since TempestRemap is a pure serial code (no MPI/OpenMP support), we had to compare serial performance on the same architectures to draw computational throughput conclusions. As mentioned above, we will include shared memory parallelism as a separate future study when we have implemented threading and/or task-based parallelism in both MOAB and perhaps TempestRemap.

- Referee: Page 20, lines 18-19. For the 1024^3 test case, are weights being generated in 2d or 3d? If 3d, is this test case an order of magitude (or more) larger than the largest climate model grids? Might be worth clairfying in text.
  Authors This was a full 3-D test case. Yes we used a very high-res run to showcase strong scalability of the point location algorithm in MOAB. While current production level runs still have lower DoFs compared to this study, there has been a lot of interest in doing sub-Km atmosphere resolution studies, which will push the boundaries of what is required from remapping libraries.

- Referee: Page 20, lines 24-26. I agree that the initialization cost is amortized for long production climate runs. But in your example, that init cost is order (hundreds) of seconds (see fig 10c). That is for a single set (pair?) of weights. In coupled climate model, there are often order (10) of these to be done. Now we're talking 1000s of seconds which starts to sound expensive in production but is certainly very expensive for short development test runs. Would it be cheaper to store the weights in a file and read them in the next time? (see general comments). Having said that, please confirm that weights are generated on each of the 1 billion gridcells (1024^3) in 3d. And if so, that's a lot of gridcells.
  Authors Agreed. This is a deficiency of the Kd-tree datastructure and as mentioned in the manuscript, we intend to add BVH implementations where the overall cost for the tree construction is much smaller. $O(nlog(n))$ in Kd-tree vs $O(log(n))$ in BVH-tree. The BVH implementation is a little complex and so we do not have this working correctly for large cases yet in MOAB.

- Referee: Page 21, Figure 10, I am struggling to read the axes and other text on the plots
  Authors At 100% zoom in the pdf using our Adobe reader, the axes are clearly visible. However, we can try to modify the fonts slightly to get better resolution in the images.

- Referee: Page 21, Figure 10b shows scaling to 512k pes for a problem size of 1024^3. The final point has 2000 gridcells per process which is still relatively big. What if you chose a problem size of 128^3 or 256^3 and tried to scale to 512k cores?
  Authors The complexity scales as O(nlog(n)). So if we decrease the total n, the total work required reduces as well, which will transition more into the memory bandwidth bound regime. So while the overall time to solution may be much lower, the strong scalability may be lower as well as expected.

- Referee: Page 25, figure 13. Is there benefit to showing the three results (colocated plus two disjoint). The results are very similar for the three cases, at least as presented. And there is no discussion of the differences/similarities in text.
  Authors One of the key points that we wanted to highlight was the relative indifference of the algorithms to the type of PE partitioning. When we originally looked at this study, it was our belief that the fully disjoint case would perform the worst and having any level of overlap with the coupler PEs would reduce the total amount of communication for both the mesh and data. While this may be true with really strict partitioning strategies, giving the components control over how the underlying grid is partitioned results in a nearly independent rate of scalability; this is especially evident when you look at the coverage mesh computation time that shows similar trends in all three cases.

  We will add additional text in the manuscript to point out this particular conclusion from the study, which was non-intuitive at first during our experimentation.

- Referee: Page 26, figure 14b. I am surprised there is so little scaling of the send/recv at NE120 and the core counts presented. I understand the claim that the absolute cost is small in all cases. I guess you are only redistributing 86k (NE120) elements, maybe that's expected then? At 128 cores, you should be transferring over 500 elements per core. Do you expect no scaling beyond that given the message size? Do you want to mention any of this in the paper?
  Authors Fig. 14 (a) shows the actual mesh migration timing. And Fig. 14(b)

shows the send/receieve scaling for the actual field data from component to the coupler PEs. After the initial setup phase through the Crystal router algorithm during the mesh migration, all communications for field data are performed point-to-point from component to coupler PEs. The lack of scaling beyond 128 cores may be related to the size of the messages here. Since we are only measuring scalability of only one field transfer here, the latency for message creation and sending (non-blocking) still dominates the actual scaling timings; we intend to follow up this study with aggregated, multi-field transfers between atm-ocn, which should show better (lower uncertainty) point-to-point communication scaling.

- Referee: The jump between 64 and 128 must be a machine thing, going offnode or something?
  Authors Yes this is correct. We have added additional discussions related to these results in the paper.

- Referee: Please confirm that you describe which machine the tests are run on in text and it might be beneficial to include that information in the figure captions. For page 27, figure 15, maybe remind us that it's case B of Table 1 (I think that's correct) in text.
  Authors These have been mentioned in each corresponding section for serial, parallel runs e.g., 4.1, 4.2, 4.3.1. We have also modified other sections and figures where this was not clear.

We request you to review the updated paper when it becomes available, and we welcome any further comments that would improve the scope of the manuscript.

Best regards,

Vijay Mahadevan

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-280, 2018.