Geosci. Model Dev. Discuss., referee comment RC2
https://doi.org/10.5194/gmd-2022-77-RC2, 2022

Comment on gmd-2022-77

Anonymous Referee #2

Referee comment on "CIOFC1.0: a Common Parallel Input/Output Framework Based on C-Coupler2.0" by Xinzhu Yu et al., Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2022-77-RC2, 2022

General Comments

In this paper the authors describe the design and implementation of an I/O framework, CIOFC1.0, based on the community coupler (C-Coupler2.0) software used by many climate models (FGOALS, FIO-AOW, GRAPES) developed or used mainly in China. The I/O framework uses XML-formatted configuration files, similar to the C-Coupler2.0 software, and reuses the interpolation algorithms in C-Coupler2.0 for spatial interpolation while writing out the model output. The I/O framework supports structured and unstructured grids. Considering the fact that climate models are running simulations at much higher resolutions than before and outputting more data and at higher frequency this work is critical to support climate models that use C-Coupler2.0, which currently does not have parallel I/O support.

The paper would significantly benefit by including discussions and comparisons with I/O frameworks used by other major climate models. Also, including performance of a climate model component instead of a test model and comparing it with other I/O frameworks would be useful to the reader.

Specific Comments

The paper includes a brief survey of the existing coupler software and frameworks, however it would be useful to include more references to I/O frameworks and libraries (e.g. the NetCDF library which has Parallel I/O support, the PIO library used by CESM and other climate models, the SCORPIO library used by the E3SM climate model, the I/O libraries used by frameworks like ESMF, the CFIO library) used by other climate models. More discussions comparing the work described in this paper with these I/O frameworks (CFIO, XIOS, PIO, SCORPIO, NetCDF, ESMF etc) would help in understanding the original contributions in this paper and add to the motivation for this work.

Although the paper mentions (p2, l20) that CIOFC1.0 can be used by other component models (apart from the community coupler) it is not apparent from the paper how it can be achieved, especially if the component model does not use C-Coupler2.0. From the provided source code and discussions in the paper the CIOFC1.0 framework is not a separate library (is part of the C-Coupler2.0 library), so integrating it with a separate component model (GAMIL, GEOS, GRAPES) would demonstrate how it can be used by component models in an earth system model.

When discussing the I/O configuration manager that uses XML-formatted inputs (Section 3.1) it would be useful to compare the approach here with other climate models that handle structured and unstructured grids (and have similar issues to deal with - support different types of grids, vertical coordinates etc). The spatial interpolation manager (Section 3.2) in CIOFC1.0 uses the interpolation algorithms from the coupler so having the functionality in the I/O framework is useful when integrating model components directly with CIOFC1.0 (if not, can't this functionality be moved inside the coupler?). So showing integration of the I/O framework with a model component (that does not use the C-Coupler2.0) would have been useful here.

When implementing the Parallel I/O operation (Section 3.3) were there any discussions on supporting low level libraries other than PnetCDF or using formats other than NetCDF? Were there any s/w design decisions made based on adding support for new libraries or formats in the future? In Section 3.3 (p9, l5) when discussing I/O decompositions, data rearrangements and grouping I/O processes as a subset of the model compute processes it would be useful to refer and discuss prior work in this area (specifically work done by J Dennis et al on the Parallel I/O library). Also it would be informative to discuss how the framework handles reading and writing multiple model variables into multiple files, since model components write 100s of variables in a typical simulation run.

The implementation of a recursive tree based timer is discussed in Section 3.4 (p11, l10). It would be useful to compare the approach here with current implementations of timers in other major climate models. Typically timers are useful in climate models in the model driver and other model components, why did the authors choose to implement the recursive tree based timer in the I/O framework instead of the C-Coupler2.0 (p11, l1) ?

The implementation of the output driving procedure is discussed in Section 3.5 (p11, l25). Again it would be useful to compare the approach here with approaches used by other coupler software/frameworks like MCT, NUOPC etc.

In Section 4 the evaluation of the I/O framework was performed using a test model (p17, l19).

- It would have also been useful to compare the serial (C-Coupler2.0 without CIOFC1.0) vs parallel (CIOFC1.0) I/O for a model component run (e.g. GRAPES, GEOS, GAMIL).
- When evaluating performance (p19, l23) a fine resolution vs coarse resolution is used for the variables being written out, however it is not apparent whether the data is on a structured or unstructured grid (The test model does seem to support both grids - p18,

l4). (For example the performance, especially the data rearrangement time, would vary significantly depending on whether the data is on a structured grid with regular 2d decomposition or on an unstructured cubed sphere grid)
- It would be useful to see the average model I/O write/read throughput with the I/O framework (This would include all I/O costs - creating files, defining I/O decompositions, data rearrangement, filesystem write time etc & would also include 100s of multi dimensional model variables written out in a typical run). The I/O throughput for a single variable is sometimes not representative of the overall I/O throughput.
- Including comparison of the I/O throughput with other I/O libraries/frameworks (CFIO, XIOS, PIO, SCORPIO, NetCDF, ESMF etc) would be useful here.
- In Section 4.3 (p20, l1-10) the authors discuss the difference in I/O write/read performance for the different number of I/O processes. It would have been useful to include discussion on how this information helps in chosing the I/O framework configuration parameters (e.g. number of I/O processes chosen) for a model run that outputs many 1D, 2D & 3D variables
- Was there any impact on I/O performance due to the placement of the I/O processes (is it configurable by the user)? Were there any interesting aspects of the machine architecture that impacted the I/O performance (e.g. In Earthlab for example the 6D TORUS network, SSDs, fast vs ordinary storage pool etc)
- Although data interpolation is handled by the coupler, it would be useful to include some brief performance statistics that includes the performance of the data interpolation and comparison with similar frameworks (online vs offline interpolation, algorithm characteristics like conservativeness, performance of interpolation using other libs/frameworks like ESMF, MOAB etc).

In the conclusion section (Section 5, p21 l24) the authors mention that they intent to support asynchronous I/O in the next version of the community coupler (C-Coupler3). Did you add this flexibility into the current design of the I/O framework? How much of the I/O framework needs to change to incorporate asynchronous I/O?

As a general comment, the paper includes very detailed information on the implementation of the different software managers in the framework and some of this information can be summarized without impacting the overall quality of the paper. Similarly in the case of figures the authors could also consider showing only the relevant parts of the XML configurations and removing the list of PnetCDF APIs.