Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2020-91-SC1, 2020

Geoscientific Model Development Discussions

Interactive comment on "A new distributed algorithm for routing network generation in model coupling and its evaluation based on C-Coupler2" *by* Hao Yu et al.

Li Liu

liuli-cess@tsinghua.edu.cn

Received and published: 25 April 2020

Dear Mr. Moritz Hanke,

Thanks a lot for your review. The incorrect statements in the manuscript you point out and the references and the introduction about YAC and ESMF in the review will help us to further improve this manuscript as well as our development of C-Coupler.

After carefully reading the review, we find that there are significant misunderstands regarding the contribution of this manuscript. We wish more discussions with you before revising the manuscript.

In the following, we'd like to pose discussions about some review points. Please show us if some points in the discussions are wrong. Thanks a lot.

1. Line 167-168: "it only utilizes point-to-point communications and does not rely on any collective communication". The parallel sorting algorithm has a complexity of O(log(n)) and a similar communication pattern as the MPI implementations of various collective communication operations[5]. Therefore, you could argue that you actually did use collective communication, which you implemented using point-to-point communication.

RESPONSE: The "collective communication" in Line 167-168 means gather and broadcast that have been used for routing network generation. Figure 1 and 2 in this reply shows an example of the collective communications of gather and broadcast based on a binary tree, respectively. Although the parallel sorting algorithm has the same number of tree levels with these collective communications, which is around log(K) (where K is number of processes), it is different from these collective communications at each tree level at least in two aspects. First, all processes work at each level in the parallel sorting algorithm (for example, Figure 2 in the manuscript), while a part of processes are idled at most levels of these collective communications. Second, the global data is distributed evenly among all processes at each level in the parallel sorting algorithm, and thus the message size corresponding to each point-to-point communication is O(N/K) (where N is the global size of the grid), while the message size generally doubles following the reverse tree of the gather communication, the whole global data is transferred at each point-to-point communication in the broadcast communication, and thus the message size corresponding to each point-to-point communication is O(N). So, the average complexity of the parallel sorting algorithm is O(log(K)*N/K), and is much lower than the average complexity of these collective communications, O(log(K)*N), which has been mentioned at Line 69 of this manuscript.

In response to the misunderstanding arising from the statement Line 167-168, we will correct it when revising the manuscript.

2. Alternatively, each process could compute for all its local points the destination rank. Using alltoall, you can exchange the number of points that need to be sent, in order to get directly from the original to the intermediate decomposition. Afterwards, alltoallv can redistribute the data in a single communication call. Depending on the MPI implementation alltoall can also have a complexity of O(log(n)). However, very little data is exchanged and it can be highly optimised within the MPI. The communication matrix for the alltoallv is probably very sparse. Hence, it can be implemented by the user using point-to-point communication. In my tests, this approach delivers really good results.

RESPONSE: It is true that each process can easily compute the destination rank for all its local points, and then can easily prepare the parameters of sendcnts as well as sdisls for using MPI alltoallv. However, each process cannot compute recvcnts from its local points without extra communications, while recvcnts is necessary for using MPI alltoallv. In my opinion, extra collective communications will be required for computing recvcnts. If we are wrong, could you please show us a solution without any collective communications? Thanks a lot.

Moreover, even though very little data is generally exchanged, the communication matrix is probably very sparse, and alltoallv can be highly optimised within the MPI, we avoid to use MPI alltoallv throughout the C-Coupler development that targets commonality and wide usage. This is because we have experiences that MPI alltoallv introduced deadlocks and its performance was unstable and depended on the parallel decompositions and MPI versions. To make C-Coupler as reliable and stable as possible especially for operational usage, such risks are not allowed for us. After viewing the code file communicator mpi.c in YAC1.5.4 (thank you a lot for making the code publicly available for viewing), it seems that YAC does not use MPI alltoallv currently. Could you show us why or do you have plan to use MPI alltoallv in future YAC versions? Thanks a lot.

3. The presented algorithm is basically a rendezvous algorithm, which uses a distributed directory. This was ïňĄrst introduced in [1]. I assume that you did not know

about this paper and therefore did not reference it. This algorithm works by distributing data among the processes in a globally known decomposition. This is called “regular intermediate distribution” in the manuscript. Using this intermediate decomposition, accessing data without knowing the original decomposition and without gathering all data on a single process is easily possible. The use of distributed directories in a similar context is mentioned in the following references [2], [3], and [4].

RESPONSE: Thanks a lot for introducing the distributed directory and the related works, which will be referenced and discussed when revising the manuscript. We are glad to know that YAC and ESMF has already benefited from distributed directory. After viewing the paper of YAC1.2.0 and the corresponding code files of YAC1.5.4, we learn that YAC use a hash table as the distributed directory for developing a parallel input scheme to read in the weights. This manuscript should be a new usage of distributed directory in coupler development. We find that the routing network generation for data transfer functionality in YAC and ESMF has not benefit from distributed directory. Specifically, the global search in YAC that may implicitly include routing network generation relies on gather and broadcast, while the routing network generation in ESFM seems to rely on the global representation of parallel decompositions (could you please show us the details if this point is wrong. Thanks a lot). Moreover, the "regular intermediate distribution" used in this manuscript is a specific distributed directory different from what is used in YAC, which makes accessing data without knowing the original decomposition and without gathering all data on a single process easily achieved by distributed sort.

4. To generate the distributed directory, the authors propose to use a parallel sort, which uses the global indices of the cells as sorting keys. Such algorithms are commonly known and are no new inventions. The remainder of the presented algorithm does not contain any signi ficant scienti ïňĄc contributions. Therefore, I do not think that this manuscript contains a substantial contribution to modelling science.

RESPONSE: Regarding this manuscript that is a technical article about couplers, our

primary goal and first contribution should be coupler improvement rather than developing a new distributed sorting algorithm. Before the code development, we have tried several times to search an existing distributed sorting algorithm that matches our requirements, in order to reduce the work for developing the distributed algorithm for routing network generation. Although parallel sorting algorithms have been widely studied, we finally failed and we have to develop a new sort algorithm implementation. Considering some existing couplers can benefit from this manuscript, we detailed this distributed sorting algorithm. Since it has been stated that "Such algorithms are commonly known and are no new inventions" in this review point, could you please show us the detailed references of the existing algorithms that are the same with the algorithm in this manuscript? Thanks a lot.

We note that, model development in this world includes a lot of papers and contributions that use an existing parameterization scheme S from an existing model A into another existing model B. So, we do believe that the employment of distributed directory for the parallel scheme of reading in the weights in YAC makes a substantial contribution to coupler development. As the distributed algorithm in this manuscript improves routing network generation, we also believe that it makes a substantial contribution.

Moreover, along with finer and finer resolutions in model development, grid size becomes larger and larger, and thus global search used in YAC, C-Coupler, etc., will become a significant memory bottleneck. We think that contributions relevant to distributed directory for settling this bottleneck would be welcome.

5. The manuscript describes the router network generation based on two predeïňĄned decompositions from two component models as being a fundamental functionality of a coupler. However, this routing table can also be a by-product, in case both components have different grids and the coupler generates interpolation weights online.

RESPONSE: It has been stated in Valcke et al. (2012) that, "couplers used in the geophysical community typically carry out similar functions such as managing data

transfer between two or more components, interpolating the coupling data between different grids, and coordinating the execution of the constituent models". Following this statement, the abstract of this manuscript states that, "it is a fundamental functionality of a coupler for Earth system modeling to efficiently handle data transfer between component models. Routing network generation is a major step for initializing the data transfer functionality". In MCT, OASIS3-MCT and CPL6/CPL7 that employ MCT, and C-Coupler, data transfer and data interpolation are implemented as two standalone functionalities and there is an explicit major step of generating routing information, although OASIS3-MCT and C-Coupler can generate interpolation weights online. That's why we can believe that some existing couplers can benefit from this manuscript.

Data transfer and data interpolation can also be implemented in an inseparable procedure, where an extra data interpolation between the same grids could be added the model coupling corresponding to the same grid. Data transfer and data interpolation should be side-by-side functionalities, which means that data transfer as well as the routing table can be viewed as a by-product of data interpolation, while data interpolation can also be viewed as a by-product of data transfer. No matter what the view is, the importance of both data transfer and data interpolation should not be impacted by the view.

6. Line 30: "there is almost no evidence of scalable initialization of a coupler" [...] "(Craig et al., 2017; Liu et al., 2018)". Line 56: "almost all existing couplers use the following 4 steps for generating a routing network" RESPONSE: We will correct these statements when revising the manuscript.

7. Paragraph 4: This paragraph does not describe how these measurements where generated. Did you do a single run, average of multiple runs or average over multiple executions of the algorithm within a single run? It is possible, that especially the ïňĄrst execution of this algorithm produces for some MPI implementations a much higher run time than the following ones.

RESPONSE: We use average of multiple runs but not average of over multiple executions of the algorithm within a single run, which will be stated when revising the manuscript. Although we also know that the second execution of an MPI algorithm can be much faster than the first execution due to the “hot spots” or “memory”, we have to use average of multiple runs, because the routine network generation for a data transfer is conducted only one time in a run under real cases.

8. Intel MPI library (3.2.2) Why did you use such an old version? In my experiments Intel MPI often performed very poorly. Maybe give the most recent OpenMPI a try.

RESPONSE: Thanks a lot for this concern. The correct MPI version used for the evaluations in the manuscript is Intel MPI library (2018 Update 2). We are sorry about the wrong introduction of “Intel MPI library (3.2.2)”. We can use another MPI version for further evaluation when revising the manuscript, if required.

Wish your further comments.

Many thanks again,

Li

Valcke, S., Balaji, V., Craig, A., Deluca, C., Dunlap, R., Ford, R. W., Jacob, R., Larson, J., Okuinghttons, R., Riley, G., Vertenstein, M: Coupling technologies for Earth System Modelling, Geosci. Model Dev., 5, 1589-1596, 2012

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2020-91, 2020.

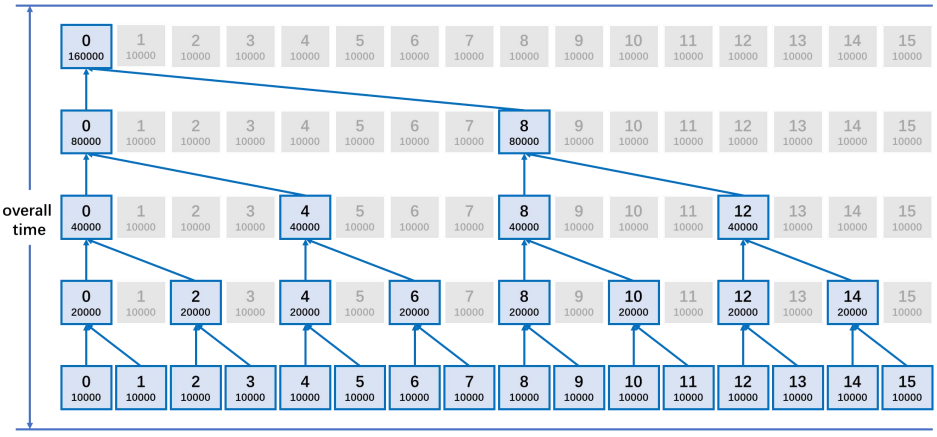

Figure 1. An example of MPI_gather following the communication network of a binary tree. The total 16000 data values are distributed evenly among the 16 processes before gathering.

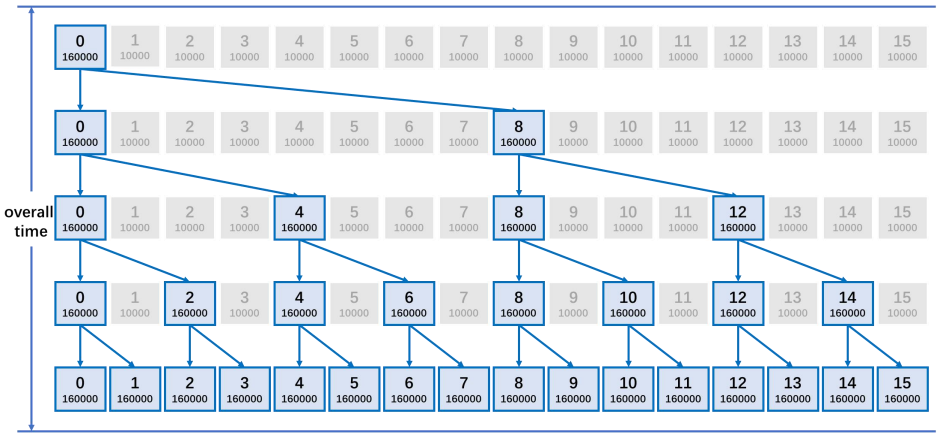

Figure 2. Similar to Fig.1 but for MPI_bcast