Geosci. Model Dev. Discuss., 6, C1858–C1864, 2013
www.geosci-model-dev-discuss.net/6/C1858/2013/

Interactive comment on "A fast input/output library for high resolution climate models" *by* X. Huang et al.

X. Huang et al.

hxm@tsinghua.edu.cn

Received and published: 9 December 2013

Dear Rob:

First of all, we really appreciate your valuable comments and that you like our ideas. Your comments are highly insightful and enabled us to significantly improve the quality of our manuscript and our software. The following pages are our point-by-point responses to each of your comments.

(1) The Interactive Discussion has already touched on this a bit, but I think the paper needs to be more clear about the role of CFIO in the climate software stack.Phrases like "CFIO decreases the I/O overhead by a factor of 5.1 compared to pnetcdf" give the impression that this project replaces pnetcdf, when really it augments. Perhaps

phrases like that could be re-worded to say "decreases the I/O overhead compared to pnetcdf alone"?

[Response]:

We will rewrite these parts in both the abstract and the introduction section as you suggested.

In addition, we will add the following paragraph into the Introduction section:

"The ideas about overlapping I/O with computing proposed in CFIO shall be complements rather than replacements of existing parallel I/O libraries, such as PnetCDF and PIO. Indeed, CFIO call the PnetCDF functions directly to implements the parallel write and read in the server side. "

(2) How are you launching extra i/o processes? The paper makes no mention of the mechanism, but if you are, or were, using MPI dynamic process management routines, that would be remarkable (they do not see much if any use, and I'm sure the MPI forum would welcome any papers discussing experiences of that facility). On the other hand, dynamic processes don't work on blue gene, so an alternate approach would be more portable. In the github code, it looks like you assign some processes in COMM_WORLD to be i/o processes. More portable, but now you've got servers and clients contending? I think your paper and your code have diverged a bit. Have you learned something worth adding to the paper?

(Ah the perils of publishing your source code. I was happy to find the code on github, and only raise these questions because I like your ideas and want them to work.)

Some codes, like GCRM, have pretty specific compute node requirements (gcrm: must be "10 times a power of two" processes), so spawning extra procs makes sense. on the other hand, you simply cannot spawn extra procs on Blue Gene, so stealing from COMM_WORLD might make more sense.

[Response]:

We believe that “launching extra i/o processes” is not a suitable description of our current implementation. As you found in the current implementation of CFIO v1.20, we are not using the function of dynamic process management in MPI2.0 to implement CFIO. We allocate part of the processes in MPI_COMM_WORLD to be I/O processes.

Therefore, in the revised paper, we will rewrite “launching extra i/o processes” into “splitting part of the processes in MPI_COMM_WORLD to be I/O processes” as you suggested. In addition, we will add some contents to describe the mechanism of how to assign some processes in COMM_WORLD to be I/O processes. (maybe we do not need to cover the details here, as they will read this in the revised version anyway). First, we need to set the ratio parameter in “cfio_init” function. We will give a simple example as following: if we execute the original parallel program with 160 processes, and suppose we want to use 32 CFIO processes to execute I/O operations, we will submit a parallel job with 192 processes. And then we need to use “cfio_proc_type” function to distinguish the I/O client and I/O server processes.

We agree with your comments on the specific compute node requirements of CFIO. We have mentioned this point in Lines of 398∼401: “Our current design requires the number of computing processes to be multiples of the number of I/O processes.” We think this is still a limitation of our implementation of CFIO. We will remove the restriction in our next version of CFIO.

(3) In section 3.1 you mention "For data writing operations, data aggregation is performed to gather subarray data". Only writes? Would reads not benefit from this approach? Or, as I think is the case, do you imagine this framework rarely if ever being used for reads?

[Response]:

We think that the idea of overlapping is only effective for write operations at present. The key reason is that the initial conditions are always read only once and the results files are written many times. As the initial conditions are read in the very beginning,

there is no space to overlap the read operations with computing. In the revised paper, we will add the following paragraph into the section 3.1:

“For high resolution climate models, we observe that the consumption of write operations is much greater than that of read operations. The initial conditions are always read only once and the results files are written many times. There is no space to overlap the read operation and computing. Thus the current CFIO v1.20 is only focus on the write operations. All the parallel read operations in CFIO v1.20 will call the corresponding PnetCDF functions directly. “

(4) I’m worried about your test environment. Intel MPI requires extra steps to turn on MPI-IO parallel I/O for lustre, I think: - http://software.intel.com/en-us/forums/topic/286114 - lustre default striping is something tiny like 4? but i think from the other parts of your paper you have already dealt with those details?

[Response]:

Actually, we also encountered the same problem as http://software.intel.com/en-us/forums/topic/286114 described. Our solution is to append " -env I_MPI_EXTRA_FILESYSTEM on -env I_MPI_EXTRA_FILESYSTEM_LIST lustre " to command "mpirun" or "bsub". For example, "bsub -n $NTASKS -env I_MPI_EXTRA_FILESYSTEM on -env I_MPI_EXTRA_FILESYSTEM_LIST lustre ./pop0.1 ".

Default lustre stripe_count of our test environment is 1, and we set this argument to the maximum 40 to get the best performance.

(6) related work: I found your related work section to be a simple laundry list of technologies. For each paper you mention, it would be good to explicitly discuss how your work relates.

I think it’s worth noting that CFIO implements a form of the two-phase optimization implemented in ROMIO. ROMIO two phase designates some MPI ranks to be the I/O aggregators, though ROMIO's aggregators are assigned to file regions, not to clients. ROMIO's approach also has no information about what's being done (data model is "linear stream of bytes"), whereas your approach has an array-oriented model.

Jing Fu's 2010 paper is noteworthy because it quantifies the overhead of overlaping i/o and computation.

[Response]:

In the revised paper, we will add the introduction about the two-phase method in ROMIO, the I/O optimization methods of Lustre, the Jing Fu's work about the analysis the overhead of overlapping I/O and computation, and etc.. We will also clearly discuss how our work relates with previous work and motivate by them.

(7) I'd like more clarification of how you are overlapping I/O and computation – you discuss how an asynchronous approach introduces too much overhead, and that a synchronous approach scales better. If so, where is the overlap/benefit happening now?

[Response]:

We will add the following more detailed description about how we are overlapping I/O and computation in the section 3.3 of our revised paper.

"In original climate models, the total running time of the simulation consists of the computing time and the I/O time. In the improved climate models with CFIO, after the data has been forwarded from client to server, the computing phase and the I/O phase can be executed in parallel. So the ideal running time only includes the computing time and the I/O forwarding time.

Comparing the above two cases, we believe that the benefit of overlapping I/O with computing comes from the fact that the I/O forwarding time with high speed network is much less than I/O time with parallel file system, especially for the high resolution climate models with a large amount of output data.

The synchronous and asynchronous approaches we discussed here for data communication are only used to shorten the I/O forwarding time. Our initial design for I/O Forwarding is using the asynchronous communication approach. However, we observed that the asynchronous communication approach leads to network resource competition between the computing phase and the I/O forwarding phase at large scale simulation. So we have to choose synchronous communication to implement CFIO. "

(8) Your baseline applications collect data on rank 0. This is of course a stupid way to do i/o in 2013, so it's not surprising you get remarkable improvements. What if those baseline applications used parallel-netcdf? Again, it's not clear to me where the overlap is happening if i/o is happening synchronously.

[Response]:

We definitely agree that that we shall be comparing CFIO with existing parallel I/O solutions. In Sec. 5.4, we have compared CFIO with PnetCDF through two MPI test programs. As Fig. 10 shows, the throughput of CFIO is approximately 10% less than that of PnetCDF because of the overhead of I/O forwarding. However, if there are time-consuming computing steps, the overall running time of the MPI test application with CFIO is shorter than that of the application with PnetCDF.

For the standalone POP, CICE and LICOM test cases, we downloaded the models from their official websites. The official standalone versions only support NetCDF. Porting these three models from NetCDF into PnetCDF would involve a significant amount of engineering work, and we think it is beyond the scope of this article.

In Fig. 5, Fig.7 and Fig.8, we provide the best performance of each model with turning off all I/O operations and compare the result with CFIO. We believe just using existing parallel I/O libraries is hard to achieve these effects, while overlapping I/O with computing can do. Therefore, we believe our proposed forwarding scheme can be a useful complement to the current parallel I/O libraries.

In addition, we will add an experiment to compare CFIO with PIO in the designed scenarios of our MPI test programs. Therefore, in the revised manuscript, the title of section 5.4 will be changed from “CFIO Versus PnetCDF” to “Comparing CFIO with PIO and PnetCDF”.

We hope that the revisions in the manuscript and our responses will be sufficient to make our manuscript suitable for publication in GMD.

Best wishes, Xiaomeng Huang

Interactive comment on Geosci. Model Dev. Discuss., 6, 4775, 2013.