Geosci. Model Dev. Discuss., 6, C2030–C2038, 2013
www.geosci-model-dev-discuss.net/6/C2030/2013/

Geoscientific Model Development Discussions Open Access

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 all,

We are very grateful to the reviewers for reading our manuscript extremely carefully and forwarding their highly valuable comments for improving our manuscript and software. Point-by-point responses to the referees' comments are listed below.

Responses to the comments of referee #1:

(1) You cite PIO (Dennis et al.) but although its authors claim to improve I/O performance on the exact model you also used (POP) over PnetCDF I cannot find a comparison of performance. Is there a reason you chose to compare with PnetCDF and not PIO?

[Response]:

In the revised manuscript, the Section 5.4 has been completely rewritten and we provided a set of supplemental experiments for comparison purpose. The title of section 5.4 has been changed from “CFIO Versus PnetCDF” to “Comparing CFIO with PIO and PnetCDF”.

We add two paragraphs to introduce the basic concepts and test scenarios about PnetCDF and PIO (Lines 565∼577). In our experiments, we tested the newest PnetCDF v1.4.0 and PIO v1.6.0 and chose PnetCDF as the back-end method of PIO to obtain good parallel throughput. For comparison purpose, we evaluated four different cases: CFIO, PnetCDF, PIO, and NO-I/O. NO-I/O means that all I/O operations are disabled in the second program. The Fig. 9 and Fig. 10 have also been redrawn.

We added the following two paragraphs to present the new experimental results about CFIO, PIO and PnetCDF (Lines 600∼605 & Lines 621∼629):

“The writing throughput of CFIO reached approximately 1 GB/s when using 128 servers and 512 clients. The same pattern was observed for PnetCDF. PnetCDF achieved a throughput of approximately 1.24 GB/s when using 128 clients. The curve of PIO is very close to that of PnetCDF, peaking at 1.2 GB/s for 128 clients.”

“Fig. 10 shows the overall running time of the test program. Without any I/O operations, the total running time was 300 seconds. When running with 128 clients, the total running time using PnetCDF and PIO was 417 seconds and 431 seconds respectively. PIO takes a small period of time, about 14 seconds, to initialize the I/O decomposition. The corresponding time using CFIO with 128 servers and 128 clients was 323 seconds. This result shows that CFIO decreases the I/O overhead compared to PnetCDF and PIO.”

Responses to the comments of referee #2:

(2) 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 have corrected these parts in both the abstract and the introduction section (Lines 18∼22 & Lines 61∼67):

“We also compared the performance of CFIO against two existing libraries, PnetCDF and Parallel I/O (PIO), in different scenarios. For scenarios with both data output and computations, CFIO decreases the I/O overhead compared to PnetCDF and PIO.”

“The main idea of CFIO is to apply an I/O forwarding technique with client-sever architecture to provide automatic overlapping of I/O and computing. The strategy of overlapping I/O with computing as proposed in CFIO is complementary to existing parallel I/O libraries. Indeed, CFIO calls the PnetCDF functions directly to implement the parallel write and read on the CFIO server side.”

(3) 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?

[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 corrected the description as following (Lines 157∼164):

“When the climate model uses CFIO as its I/O method, we would have a group of computing processes and an extra group of I/O processes to form the entire MPI communicator. For example, if we execute the original parallel program with 32 processes, and we want to use 4 CFIO processes to execute I/O operations, we will submit a parallel job with 36 processes.”

(4) 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 agree 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. Therefore, in the revised paper, we added the following paragraph to clarify this point (Lines 192∼199):

“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 focuses on the write operations. All the parallel read operations in CFIO v1.20 will call the corresponding PnetCDF functions directly. ”

(5) 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".

In the revised manuscript, we added the following sentences to describe the striping value used in our experiments so that the reader can repeat our experiments (Lines 575∼577): "The default striping count for our Lustre file system is 1, and we changed this argument to the maximum 40 to get the best write 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.

[Response]:

In the revised manuscript, we rewrote the Section 6 carefully and we believe that we have clearly discussed how our work relates with previous work and motivate by them. The major modifications include:

a) The introduction of Jing Fu's work (Lines 654∼661).

"Fu et al. (2010) quantified the overhead of overlapping I/O and computation using real partitioned Computational Fluid Dynamics(CFD) solver data. They proposed that it is possible to use a small portion (3 to 6%) of the processes in the MPI communicator as I/O processes to achieve an actual writing bandwidth of 2.3 GB/s and latency hiding writing bandwidth of more than 21TB/s on the IBM Blue Gene/L supercomputer."

b) The relationship between CFIO and ROMIO (Lines 662∼680). "Collective buffering is an attractive and practical optimization method in MPI-IO. It also called two-phase I/O, which means breaking the I/O operation into two stages. For a collective write operation, the first stage uses the aggregators, which are a subset of MPI processes, to aggregate the data into a temporary buffer on the aggregator nodes. In the second stage, the aggregators ship the data from the aggregator nodes to the I/O servers. The advantage of two-phase I/O is that fewer nodes are necessary to communicate with the I/O servers, which reduces resource contention. ROMIO(Thakur et al. (1999)), which is one of the portable MPI-IO implementations, uses two-phase optimization to improve the I/O performance. ROMIO's two-phase optimization designates some MPI ranks to be the I/O aggregators, though ROMIO's aggregators are assigned to file regions, not to clients. The data model of ROMIO is a linear stream of bytes, whereas the data model of CFIO is array-oriented. It is worth noting that CFIO implements a form of the two-phase optimization implemented in ROMIO above the Lustre file system."

c) The difference between PIO and CFIO (Lines 707∼718).

"For climate models, Dennis et al. (2012) introduced an application level parallel I/O library named PIO. It provides the flexibility to adapt to different I/O requirements for different component models of CESM. PIO utilizes an I/O forwarding in which that a part of the compute nodes are selected to collect the output data and rearrange data in memory into a more I/O-friendly decomposition, called data rearrangement. Through data rearrangement, PIO archives better I/O throughput with less memory consumption because it leads to less function calls of back-end I/O libraries. However PIO cannot overlap I/O with computing, that is, PIO can shorten I/O time but not hide it."

d) At the end of Section 6(Lines 725∼736), we concluded that the design of CFIO was inspired by the technologies of overlapping I/O, two-phase I/O and I/O forwarding. Comparatively, CFIO focuses on the requirements of high resolution climate models and provides automatic overlapping of I/O with computing so as to shorten the entire simulation time of the climate models, not just the I/O part. CFIO uses I/O forwarding to perform overlapping I/O on remote processors so that the overhead for managing multiple threads is avoided. In addition, CFIO provides synchronous functions that perform I/O overlapping automatically. Modifications to the existing climate modeling code for asynchronous functions are not necessary.

(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? Again, it's not clear to me where the overlap is happening if i/o is happening synchronously.

[Response]:

In the revised manuscript, we added the following more detailed description about how we are overlapping I/O and computation in the section 3.3 (Lines 242∼259).

"In original program, the total running time of the simulation consists of the computing time and the I/O time. In the improved programs 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 a high speed network is much less than I/O time with a parallel file system, especially for the high resolution climate models with a large amount of output data.

There are two options when designing the communication method for I/O forwarding: synchronous and asynchronous methods. The synchronous and asynchronous approaches we discussed here for data communication are only used to shorten the I/O forwarding time."

(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?

[Response]:

We definitely agree that that we shall be comparing CFIO with existing parallel I/O solutions.

In the revised manuscript, the Section 5.4 has been completely rewritten and we provided a set of supplemental experiments for comparison purpose. The experimental results show that, without any I/O operations, the total running time was 300 seconds. When running with 128 clients, the total running time using PnetCDF and PIO was 417 seconds and 431 seconds respectively. PIO takes a small period of time, about 14 seconds, to initialize the I/O decomposition. The corresponding time using CFIO with 128 servers and 128 clients was 323 seconds. This result shows that CFIO decreases the I/O overhead compared to PnetCDF and PIO.

We concluded the experimental results as following (Lines 79∼84): "Although CFIO has slightly lower throughput than PnetCDF and PIO for scenarios with only data output, CFIO decreases the I/O overhead compared to PnetCDF and PIO for scenarios with both data output and computations, resulting in better overall performance for real climate models."

We also added the following two paragraphs to introduce the baseline applications and the experimental plan. We believe that it is helpful for interested readers to repeat our experiments. (Lines 392∼405):

"In the following sections, we first describe our evaluation of CFIO through three climate models, and then provide a comparison between the performance of CFIO and PnetCDF in various scenarios."

"For the standalone POP, CICE and LICOM test cases, we downloaded the models from their official websites. The official standalone versions only support NetCDF. In the following three cases, we provide the best performance of each model by turning off all I/O operations and comparing the result with CFIO. Because of the benefits brought by overlapping I/O with computing, using CFIO comes the closest to the best performance. Therefore, we believe our proposed forwarding scheme can be a useful complement to the existing parallel I/O libraries."

We believe the manuscript has improved considerably from all of these changes. 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.