Interactive comment on “Compact Modeling Framework v3.0 for high-resolution global ocean-ice-atmosphere models” by Vladimir V. Kalmykov et al.

Maxim Kaurkin

maksim.kaurkin@phystech.edu

Dear Reviewer, thank you very much for finding time to read our article and present your comments that helped to improve the manuscript. The paper has undergone a thorough reformulation, some sections have been extended. Please find our responses below.

Author responses to Reviewer 1 comments

The following is the point-by-point response to the reviewers' comments (shown in “italic”).

> ***Referee comment.***
>
> *General Comments:*
>
> *This paper provides an overview of the Compact Modeling Framework (CMF2.0 and CMF3.0) implementation. The paper is well organized. Performance plots are shown for several high resolution cases. It would be nice if the performance plots were extended to higher core counts if possible. The paper could use an additional review by a native English speaker as much of the paper includes some grammatical challenges. Specifically, lack of “a” and “the” in the paper could be much improved.*

Author's response.

We have revised the English wording and sentence structure throughout the paper. Two figures and one section have been added, so the numbering has changed. As for higher core counts, please see the answer to comment (4).

> ***Referee comment.***
>
> *Specific Comments:*
>
> *(1) page 2, line 53. Please define WOM at first use and review that definitions exist for other acronyms.*
>
> *page 4, line 33. Please define SOA at first use and review that definitions exist for other acronyms at first use.*

Author's response. The terms «World ocean model (WOM)» and «service-oriented architecture (SOA)» have been added to the text (in the beginning and at the end of Section 2, respectively). All other acronyms have been checked.

> ***Referee comment.***

(2) Figure 1 implies that the coupler has distinct cores. Please make sure this is also clearly stated in the text. The picture in Figure 1 suggests there is a 1:1 connection between model tasks and coupler tasks, but this is highly unlikely in practice. It might be clearer if each component had different numbers of tasks in Figure 1. The figure also implies the decomposition on the coupler is the same as the decomposition in the models. But then this does not guarantee "locality of data and communications during the interpolation process or I/O actions" as stated on page 4, line 50. Either the coupler has "near" 1:1 communication with physical models and then interpolation requires a rearrange communication OR there is M:N communication between physical models and the coupler and then minimal communication as part of interpolation. The only way both communication to coupler and interpolation communication can be minimized is if the model decompositions are all chosen very carefully. Again, in practice, this will not be the case. Some rethinking about how this is stated and shown would be helpful.

Author's response.

The distinction of cores has been explicitly stated in the beginning of the Section 3.1.

Figure 1 has been reworked to show the more general case. It illustrates now the basic idea that every coupler core communicates with a fixed subset of each component's cores.

If we correctly understood the comment of the Reviewer, our work is an attempt to implement particularly the latter case, which optimizes both communication to coupler and interpolation communication. This is achieved as follows:

a) The size of coupler communicator is taken much smaller than the size of any model component communicator, so the amount of rearrange interpolation communications is small (though nonzero).

(b) Each coupler core ("master") interacts only with a specific subset of each component's cores ("slaves"). This allows to reduce the required amount of coupling communication routes for every component from M*N (*which would be in the general case of the component and the coupler running on M and N cores, respectively*) to only M, since each slave now interacts just with its master. Therefore, the size of every component communicator has to be a multiple of the coupler communicator size. In order to meet this condition in practice (e.g., for "poorly divisible'" component grid sizes), the CMF2.0 partially supports uneven grid subdomain sizes by making the last row of the component's 1D- or 2D-decomposition narrower than the others.

We have described this master-slave architecture in the Section 3.2 and removed the term "locality" from the text.

As for the CMF3.0, all communications are carried out through the GA layer, so these optimizations are not applicable. But, on the other hand, the decompositions now can be arbitrary. In particular, it becomes easy not to reserve processor cores for subdomains of a global ocean model that lay on continents.

Referee comment.

(3) "Combination of predefined time chain, persistent communications and pointer based asynchronous data sending provides maximal efficiency of data gathering and distribution."

page 5, line 60. Please provide additional details on how this is implemented.

Author's response. We have extended the corresponding paragraphs:

All events in the system are divided into few classes (save diagnostics, save control point, read file data, send/receive mapping, etc.), defining different actions with data arrays. In the CMF2.0, we postulate that all events could be predefined before the start and occur with fixed periods. Thus, the coupler can take on the task of synchronizing models and avoiding deadlocks.

The sequence of events (time chain) is constructed in the main CMF program, which is the entry point of the coupled model. Also, at the registration stage, models provide the CMF system with pointers to the arrays that must be processed in the events. So, during the system operation, events are performed automatically and do not require explicit calls from the user. As the information about the periods of all events is known at the registration stage, the coupler can build a table of its actions. This allows to exclude parallel synchronization of the coupler cores, which otherwise would be necessary when, for example, two components at the same time want to write data to the file system. When a certain time moment arrives, the coupler selects the next event from the chain and calls the appropriate handler function based on the type of this event, while the model components asynchronously send data. Moreover, it becomes possible to use persistent MPI-operations (combinations of MPI_SEND_INIT and MPI_STARTALL) for all events, thus saving time of repeated communications. Combination of predefined time chain, persistent communications and pointer-based asynchronous sending provides high efficiency of parallel data gathering and distribution.

> ***Referee comment.***
>
> *(4) Figures 2-4 and Figure 6. It would be nice if there were some additional results at higher core counts. I recognize the authors feel this is not needed because the performance of the model is adequate as shown. It still would be informative to the community to see how far the strong scaling goes in their implementation.*

Author's response.

These performance tests were conducted prior to year 2016 and, unfortunately, due to various reasons there is no access to most supercomputer configurations mentioned in the paper. Particularly, we have no access to the BlueGene/P, BlueGene/Q and Lomonosov, while the MVS100k and MVS10p have been reconfigured which does not allow to continue experiments in the same conditions. Nevertheless, results of the BlueGene tests are most interesting since they are obtained on highest numbers of cores (up to 32K). Moreover, the BlueGene system has the most advanced interconnect, which allowed to achieve almost linear scalability. Thus, these results are left unchanged in the paper as an evidence that the CMF2.0 system can show good results on a high-performance supercomputer.

An improvement was possible for the Figure 6 (now Figure 8), where we changed CMF3.0 Lomonosov results to MVS10p ones, which extend to 256 coupler cores. This also made it possible to compare them with CMF2.0 results obtained in the same conditions, as asked in the Comment (7).

> ***Referee comment.***
>
> *(5) Figure 4. The log scaling does not show the detailed information of the relative performance of different cases for a fixed core count. The text notes the percentage*

differences of a few cases, and this is interesting but incomplete. I wonder if it might be better to show the data differently, maybe in a table, or maybe in a plot where the y-axis was linear with non-dimensionalized scaling units.

Author's response.

In our opinion, the logarithmic scale with time vs. number of cores is preferable for the analysis of parallel efficiency, because it allows simultaneously to compare with linear trend (which illustrates “perfect scaling”) and to know the time of execution (in contrast to a speed-up plot or a non-logarithmic time-cores plot). For convenience of estimations, we have added projection lines to the plots on the Figures 2 and 8. The data of Figure 4 additionally has been presented in a table.

Referee comment.

(6) Section 3.4 describes some theoretical ideas about cost for 4 different I/O schemes. It closes by indicating the asynchronous scheme was chosen and that it works without providing any further results. I think, at the least, the performance of the implementation should be documented with actual numbers and then compared with the theoretical description. It would be great if that scheme could be compared to the 3 other schemes, although recognize this might not be possible. The description of the 4 schemes could certainly be reduced, especially as no results are presented for them. Result from the actual performance of the implementation should be increased and described in more detail.

We have reduced and rewritten this section, so that it is now devoted to the CMF asynchronous scheme only. An inspection of the other 3 schemes would require a big separate study, so we do not consider them here. Data writing speed test results have been added for the CMF2.0 on the MVS-10P and BlueGene/P systems. In Section 4.2 we have also added and discussed results of the actual asynchrony test of the CMF3.0 I/O system in conditions of the 0.1-degree global ocean model.

Referee comment.

(7) Figure 6, could the CMF2.0 results be added to the plot. This is brought up directly on page 11, line 15 and then again on page 11, line 19.

Author's response.

CMF2.0 results have been added to this figure (now it is Figure 8).

Referee comment.

(8) page 11, line 19. “as expected”. Please expand on this, why is it expected?

More generally, please expand on the differences in CMF2.0 and CMF3.0. They both have the coupler on separate cores. CMF3.0 has an additional buffer layer, how is this beneficial, what works well, what doesn’t work so well? It is slower than CMF2.0 so how does the community feel about the implementation?

Author's response.

The decline in performance is expected due to the overhead of using GA-library (as an intermediate send/recv data representation) and due to deprecated MPI_SEND_INIT procedures in the CMF3.0. This is a sacrifice for the compact code representation, easy parallel messaging, and for convenience of adding new user-defined services. We have expanded the Section 4 in order to clarify these features of the CMF3.0 system. This is one of the first papers about the CMF3.0, so we hope that it will start bringing the feedback from the wide community.

> ***Referee comment.***
>
> *(9) page 13, line 61. Please state how many cores the coupler was using. This should be noted in all application results.*

Author's response.

The two configurations compared in this section used (32000 ocean cores + 400 coupler cores) and (8000 ocean cores + 100 coupler cores). This has been added to the text.

In Section 5.2 the core counts are (1152 ocean cores + 288 atmopsphere cores + 16 coupler cores).

In Section 5.3 the timing of CMF3.0 DAS service is tested up to 32 cores. The CPL service is not used here.

> ***Referee comment.***
>
> *(10) Figure 8. It would be nice if this plot were formatted similar to plots 2-4 with time instead of acceleration on the y-axis, for consistency. Even if it's not log-log and even if it's relative time in this case.*

Author's response.

This figure (now Figure 10) has been formatted similar to Figures 2-4.

> ***Referee comment.***
>
> *Technical Comments: I will not go thru each grammatical error but strongly encourage additional review by a native English speaker. Let me just propose an update to the Abstract, for instance,*
>
> *We present a new version of the Compact Modeling Framework (CMF3.0) developed for the software environment of stand-alone and coupled global geophysical fluid models. The CMF3.0 is designed for use on high and ultra-high resolution models on massively-parallel supercomputers. The key features of the previous CMF version (2.0) are mentioned to reflect progress in our research. In the CMF3.0, the MPI approach with a high-level abstract driver, optimized coupler interpolation, and I/O algorithms is replaced with the PGAS paradigm communications scheme, while the central hub architecture evolves to a set of simultaneously working services. Performance tests for both versions are carried out.*

In addition, a parallel realisation of the EnOI (Ensemble Optimal Interpolation) data assimilation method as a program service of CMF3.0 is presented.

Much of the document could use similar revision. There are issues throughout.

page 7, line 3, “communicational” is not a word and that sentence makes little sense as written.

page 11, line 11, using -> use

page 14, line 2. Starting the sentence with SYPD is not ideal. Just say “The model throughput” and provide units on the 0.75 value.

page 15, line 27, remove “with” in “handle with huge”

page 15, line 34, change “to further” to “for further”

Author's response.

Thank you very much! These corrections have been applied to the text.