Author's Response to reviewers' comments 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 1 and Reviewer 2, 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 and marked-up manuscript version changes made using latexdiff.

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 <u>not</u> 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 MVS-100k and MVS-10P 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 (new Figure 5). 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 (new Figure 7).

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 stated in 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 (except the last one, since the corresponding paragraph has been removed).

Author responses to Reviewer 2 comments

Referee comment.

A.1) The description of CMF2.0, results of Test I and II and related conclusions need to be clarified:

- First, I agree with Referee #1 that figure 1 is misleading as it suggests a 1:1 connection between the models processes and the coupler processes. I don't understand either the "which means data locality" on P.4, L.50, as there is certainly some exchange of data needed between the component processes and the coupler processes.

Author's response.

Yes, we agree. The figure has been redrawn to show the basic idea that every coupler core communicates with a fixed subset of each component's cores. For example, on the Figure 1 the 1st core of the coupler (c1) sends and receives data from the component cores o1, o4, o7, i1, a1 and a4. Originally, we embedded this meaning in the term "locality", but indeed it can be misleading. So, the paragraph has been extended and reformulated in "master-slave" terminology without a reference to locality.

Referee comment.

- P.5, L.63-66: these two paragraphs are not clear at all. What does "a subset of component's cores works only with individual master core in the coupler" mean? What

does Â n ´ for the two cases, of the source and destination type Â˙z mean and why do you put a reference to Craig et al. 2005 here?

Author's response.

These paragraphs have been reformulated. We are explaining the “master and its subset'” concept in the Section 3.2 (see previous comment), while these paragraphs are devoted to handling interpolation weights. The “source” and “destination” mapping types are briefly explained, while the thorough explanation can be found in (Craig et al., 2005, Section 3.4).

Referee comment.

- P.5., L. 69-70: You write “All necessary links are initialized at the beginning of run and are used at the calculation stage as persistent (Jacob et al., 2005). Â˙z I suppose that the links you mention here are not the SCRIP links? You should be using another word as this is confusing. Also, why do you mention Jacob et al. 2005 here?

Author's response.

Yes, it is better to say “component-component SCRIP links and intracoupler rearrange routes”. The paragraph has been reformulated. It refers to Section 3.3.2 of (Jacob et al. 2005), which can be accessed for a more detailed explanation of the rearranging concept.

Referee comment.

- P.5, L.72-73: I don’t understand this sentence: “It is worth noting, that links are not sent directly, but as sorted unique cells vectors which allow one to avoid sending duplicated data. Â˙z. Again what “links” are you talking about here?

Author's response.

It is just the data, which is being interpolated. The paragraph has been corrected.

Referee comment.

In Figure 2, 3 & 4 captions and on P.6, L.88, you should recall what is included in the timing. I suppose this is the time in seconds for the whole 10 model days for the whole ocean-atmosphere and atmosphere-ocean exchanges through the coupler. (Same remark for Figure 6).

Author's response.

That is correct. Figures 2,3,4 and Figure 8 (previously 6) show the timing of 10-day model runs with disabled physics routines (see Section 3.3, 6th paragraph for the Test I description, and 9th paragraph for the description of Test II).

The corresponding remarks have been added to figure captions and to the mentioned place in the text.

Referee comment.

On p.6, L.92, you write: “It is clear that 20-40 coupler cores provide satisfactory speed for such problems, because ∼10 seconds costs . . .” It is not clear how you get these numbers. I see at best, i.e. with the MVS-10p_16, something between ∼15 and ∼20 seconds. Same remark for the number presented on P.7, L.4-5. This re-joins referee#1’s comment about the fact that the log scale does not allow one to get the detailed information mentioned in the text.

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. For example, on Figure 2, 32 cores of MVS-10p_16 under CMF2.0 give 10.5 seconds. The data of Figure 4 additionally has been presented in a table.

Referee comment.

P.7, L.6: again here I don’t understand why you write “perform only local communication”

Author's response.

Since the master-slave algorithm was described above, this paragraph has gained a minor reformulation, which removes the reference to locality:

“Since every coupler core communicates only with a subset of

component cores, increasing of the coupler communicator size leads both to decomposing of the interpolation computations

and to decreasing of the component-coupler communication overhead, though slightly increasing intracoupler rearrangement

communications.”

Referee comment.

Section 3.4 on I/O should be completely revised. The long theoretical section on P.8 with detailed formula is not useful here, especially as you finally simply state “Asynchronous scheme was incorporated in the latest version our framework Â˙z without giving finally any results! The theoretical section should be cut and numbers obtained for the grid sizes you list for INMIO World ocean model should be provided.

Author's response.

We have reduced and rewritten this section, so that it is now devoted to the CMF asynchronous scheme only. Data writing speed test results have been added for the CMF2.0 on the MVS-10P and BlueGene/P systems (new Figure 5). 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 (new Figure 7).

Referee comment.

The different utility modules available should be briefly described or a reference to a documentation or User Guide should be provided.

Author's response.

We have provided the User Guide, which includes the description of the interface, utility modules and capabilities of the CMF3.0 system, as well as the process of installing and configuring the coupled model, as part of which the CMF is distributed. This manual is available at http://model.ocean.ru:6623/VITIM-manual-eng.pdf

Referee comment.

A.2) The description of CMF3.0 needs to be extended

This paper is supposed to mainly describe CMF3.0, or at least this is what the title implies but very little is written about it. It looks like the author was in a hurry to finish the paper. More details should certainly be given on how the I/O service work (1st paragraph on P.11) and on the Data Assimilation part (currently only 2 lines, P.11, L8-9). These improved descriptions should be backed up with performance results (as is done for the interpolation).

Author's response.

We have added a new section, which describes the mechanism and opportunities of working with global arrays in CMF3.0 by means of the class Communicator_GA (currently, Section 4.3). The I/O service description also has been extended. Its performance test has been presented in the Section 4.2.

As for Data Assimilation, it is a large area of our research, which requires a separate publication. We suppose that the graph of parallel efficiency for the DAS service (as an important result for this work) and references to our published papers about the data assimilation problem are sufficient.

Referee comment.

Also the discussion of the interpolation results should be extended and detailed. How do the author get to 2-3 seconds per modelling day on 20-50 CPL cores? (This is mentioned also in the conclusions P.15, L.28.) Why is it expected that results would be worse than for CMF2.0 (and "worse" should not be used here because it implies that results for CMF2.0 are bad and that results for CMF3.0 are even worse)? Is it because of the shift from MPI to PGAS? Or because the tests were performed on a different platform?

Author's response.

For convenience, we have added projection lines to the Figure 8, thus they show 20-30 seconds per 10 model days. 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 and for convenience of adding new features (like Data assimilation or Nesting technology). We have extended the Section 4 in order to clarify these features of the CMF3.0 system.

We have used “less strong” instead of “worse”.

> ***Referee comment.***
>
> *Also, in the conclusions, you write: “The key part of it, coupler, has a sufficiently small code size for such programs (about 5000 lines of code with unit tests) and is able to manage the main parallel problems of the coupled modeling - synchronization, regridding and I/O.” I don’t understand why you write that the coupler manages the I/O as this is not the case in CMF3.0.*

Author's response.

This phrase must refer to the CMF2.0. It has been corrected.

> ***Referee comment.***
>
> *A.3) The whole text needs reviewing by a native English speaker. The style and wording needs revision as some sentences are simply not understandable (at least by me),*
>
> *e.g.:*
>
> *- P.3, L.16: “Unquestionable advantage of non-coupler design is the absence of interference in the user code”: why “non-coupler design”? Also, with OASIS, there is some interference in the user code but the objective is to minimize it.*

Author's response.

We mean that there is no coupling through a standalone coupler in OASIS3-MCT, so the user does not have to reorganize his code according to standard interfaces (e.g., as required for the cpl7 coupler). The phrase has been changed to “Unquestionable advantage of this non-coupler design is the minimization of interference in the user code, since there is no need to adapt it to interfaces required by a coupler.”

> ***Referee comment.***
>
> *- P.5, L.76: “Several ping-pong tests were carried out for interpolation system using coupled ocean-atmosphere model.”*

Author's response.

The phrase has been refined for explanation of ping-pong test conditions: “The performance rate of the CMF2.0 interpolation system was evaluated in several “ping-pong” tests, in which the coupler was maintaining component-component exchanges of the INMIO-SLAV ocean-atmosphere model with disabled solvers of physics equations (similarly to the ping-pong test of OASIS3 in (Valcke, 2013))”.

Referee comment.

P.6, L.95-96: “work of the sequential algorithm is only possible with restriction that memory is allocated only for interpolation block, which is impossible in practice Â˙z: I am not sure what this sentence means exactly and why you write that it is impossible in practice while you do get some results on 1 core; do you mean it would be impossible with real models as the interpolation per se would require all the available memory?

Author's response.

Yes, to perform this test we had to switch off the allocation of all physical model arrays, except those particularly involved in the test. So, in real numerical experiments the node memory (at least on considered supercomputers) will be insufficient for both physics equations solving and work of the 1-core coupler. Possibly it is better to say “unlikely” instead of “impossible”. The phrase has been refined.

Referee comment.

- P.13, L.63: “but we are more interested in scalability of the program on perspective sizes of computational resources Â˙z

Author's response.

The phrase has been reformulated indicating the saturation of the decomposition algorithm:

“Obviously, at high core counts the parallel efficiency curve experiences “flattering”. But assuming that the time step of the model is 5 min., the result of the experiment leads, e.g., to quite satisfactory five simulated years per wall-clock day (SYPD) rate achieved on 20000 cores of the BlueGene/Q supercomputer.”

Referee comment.

P. 13, L.71: “Time evolution of the sea-ice surface temperature is described in the same way as in prescribed ocean experiments.”:

Author's response.

The sentence has been removed.

Referee comment.

P. 13, last paragraph: please rephrase, the current sentence with the “min.” is too difficult

to follow.

Author's response.

The sentence has been rephrased and split in two sentences:

“Every

72 min., nine 2D-arrays were transferred from the atmosphere to the ocean (components of wind stress, short- and long-wave

radiation, fluxes of sensible and latent heat, precipitation, evaporation, air temperature at 2 m). Conversely, every 144 min. three

2D-arrays were transferred from the ocean to the atmosphere (upper gridbox temperature, temperature and concentration of sea ice).”

Referee comment.

P.2, L25: “Coupling through shared file or sequential component is acceptable . . . Â˙z could be Â n´ Coupling through shared file with components executing sequentially is acceptable . . . Â˙z

Author's response.

We imply slightly different meaning. Changed to “Coupling through a shared file or through a sequential hub is acceptable...”

Referee comment.

P.2, L33: “. . . and their representation in the interfaced style understandable . . .” could be “ . . . and their adaptation to the interface understandable . . .”

Author's response.

Changed to “This approach requires some reorganization of the components' code and its adaptation to the interfaces understandable by the driver...”

(since we are talking about a set of standard interfaces through which all model procedures should be called).

Referee comment.

- P.2, L41: “GFDL FMS (Balaji, 2012) system additionally suggests fully parallel data storage with file post processing at the end of the run Â˙z could be Â nIn the GFDL FMS (Balaji, 2012) system, fully parallel data storage with file post processing at the end of the run is offered Â˙z

- P.3, L.21: “According to proposals of Earth System Modeling conference, (Valcke et al., 2012), . . .” could be “According to the analysis of coupling technologies for Earth System

Modeling by Valcke et al. 2012, . . .”

- P.6, L86: “Performance is based on a standard Intel Fortran compiler” should be moved to the next paragraph and could be “On all supercomputers, the coupled system was compiled standard Intel Fortran compiler.”

- P.6, L.90: “Increasing number of coupler size”; should be “Increasing number of coupler processes” or “Increasing the size of the coupler communicator”

- P. 11, L. 12: “Optimizations regarded to ignore repeated cell requests are preserved” could be “Optimization regarding repeated cells are preserved”.

Author's response.

These issues have been corrected according to the reviewer's suggestions.

Referee comment.

- P. 13, L.56: “Latest version of INMIOWOM model was fully integrated to CMF Â˙z could be Â n ´ CMF2.0 was fully integrated in the latest version of INMIOWOM Â˙z

Author's response.

Changed to “The latest version of INMIO WOM is distibuted in an integrated package together with the CMF2.0 and 3.0, all necessary libraries and a standardized folder structure facilitating the adding of new model components (including adapter files for the CICE sea-ice model).”

Referee comment.

- P.14, 3rd line: “Ice model was built into the ocean model , land model – into the atmosphere model” could be “The ice model is integrated in the ocean model and the land model in the atmosphere model”

Author's response.

Changed to “The sea ice was simulated by the INMIO built-in ice thermodynamics model, while the land processes were incorporated into the SLAV atmosphere model.”

Referee comment.

- P.14, L.6: “structurize” could be “structure”

Referee comment.

B) Other comments:

B.1) References:

- P.2, L.6: The reference to OASIS3 should be Valcke 2013 (i.e. p18, L45)

- P.2, L.36: The reference to OASIS3-MCT should be Craig et al 2017: A. Craig, S. Valcke, L. Coquart, 2017: Development and performance of a new version of the OASIS coupler, OASIS3-MCT_3.0, Geosci. Model Dev., 10, 3297-3308, https://doi.org/10.5194/gmd-10-3297-2017, 2017.

- P.3, L.7 (as annotated in the manuscript): Reference to ESMF should be the more recent one: Theurich, G., Deluca, C., Campbell, T., Liu, F., Saint, K., Verten- stein, M., Chen, J., Oehmke, R., Doyle, J., Whitcomb, T., Wall- craft, A., Iredell, M., Black, T., Da Silva, A. M., Clune, T., Fer- raro, R., Li, P., Kelley, M., Aleinov, I., Balaji, V., Zadeh, N., Ja- cob, R., Kirtman, B., Giraldo, F., McCarren, D., Sandgathe, S., Peckham, S., and Dunlap IV, R.: The Earth System Prediction Suite: Toward a Coordinated U.S. Modeling Capability, B. Am. Meteor. Soc., 97, 1229–1247, https://doi.org/10.1175/BAMS-D- 14-00164.1, 2016.

Author's response.

These issues have been corrected according to the reviewer's suggestions.

Referee comment.

B.2) P.2, L.35-36: OASIS3-MCT proposes coupled system not only as single executable, so this sentence is misleading; the important feature is that the coupling functions are not provided by a standalone coupler but by a coupling library linked to the component models. Please correct.

Author's response.

The paragraph has been rewritten as:

“The coupled system can be launched as a single or multiple executable without a separate coupler whose functions in this case are provided by by a coupling library and performed in parallel on a core subset of each model component. Such a solution was proposed in OASIS3-MCT (Craig et al., 2017). A high-level driver controlling system sequencing is not required in this case.

Referee comment.

B.3) P.2, L.44-50: This list mixes functionalities (1. and 2.) and characteristics (3. and 4.); please reorganise.

Author's response.

Sentence reworked.

“Thus, we can point out the necessary features of modern coupling frameworks:”

Referee comment.

B.4) P.3, L.11: It is not really fair to write that the computational costs of CESM coupler are quite significant 20%, as the CESM coupler does not only perform coupling and remapping but also performs the surface flux computation.

Author's response.

The phrase has been changed to “Tests showed that computational costs of the CESM coupler (including coupling, remapping and surface flux computation) are quite significant 20%...”

Referee comment.

B.5) P.3, L. 21-24: these 3 lines do not give an appropriate summary of the analysis provided by Valcke et al. 2012. Please correct.

Author's response.

The paragraph has been rewritten as:

“According to proposals of the Earth System Modeling conference (Valcke et al., 2012), today there are several common aspects in coupling software development: an ability to communicate data between components, regrid data, and manage the time evolution of the model integration. There is a lot of custom parallel coupling mechanisms, with either single or multiple executable approache. We selected the approach with single executable because it can simplify the program flow and give additional opportunities for performance optimization. Besides, we used NetCDF for parallel I/O and SCRIP (Spherical Coordinate Remapping and Interpolation Package) (Jones, 1999) for regridding, as done in OASIS3 (Valcke, 2013).”

Referee comment.

B.6) P.4, L.42-43 & P.9, L.47: Either provide details on what interceptor or Template methods are or don’t mention them; one should not have to read the reference (Gamma et al., 1995) to understand the sentence.

Author's response.

The reference to interceptor methods has been removed. Section 3.1 has got some reformulations to clarify the meaning of Template methods.

Referee comment.

B.7) P.5, L77: References here are misleading as they seem to imply that "Test I condition" are explicitly defined in Valcke et al., 2012 or in Craig et al. 2012, while they are not.

Author's response.

The references have been changed to (Valcke et al., 2013) where the ping-pong test is explicitly defined.

Referee comment.

B.8) P.6, L.93: You should not use the word "failure" here as the test does not fail, it is just very slow.

Author's response.

"Failure" changed to "ineffectiveness".

Referee comment.

B.9) Using very technical coding terms along the text does not help understanding it (e.g. P.4, L.45: "Component class"; P.9, L.63:"resulted in class Communicator"; p.10, L.77-78:"since all services in CMF3.0 inherit base class Service it also allows one to easily add new Â˙z ; P.10, L.80 : receives data using Communicator Â˙z ; P.12, L.25 : Â n ´ NormalEvent Â˙z or Â n ´ SyncVarEvent Â˙z : P.12, L.27 : Â nGenerators realize abstract class EventGenerator, so new specific generator subclasses could be easily added); I think it would be better to explain the concept that using these abstract terms.

We have extended and reformulated the explanation of these methods. Some terms have been excluded (Component class, EventGenerator). The others (class Communicator_GA, class Service, NormalEvent, SyncVarEvent) are kept, since they are names of the objects described and are needed as references in the text.

Referee comment.

B.10) P.12, L.26: Reference to Griffies et al. is not useful here.

Author's response.

Instead of this reference we pay attention to importance of supporting experiments with prescribed forcing referenced to the real calendar, e.g. the Drakkar Forcing Set (Dussin et al., 2016)

Referee comment.

B.11) P.12, section 5.1: What is the resolution of the INMIO World Ocean model in these tests?

Author's response.

It is 0.1 degrees (added to the last paragraph of the section and to the Fig. 9 caption)

> ***Referee comment.***
>
> *B.12) P.15, L.25-27: please specify if these numbers apply to CMF2.0 or CMF3.0. On line 27, change "CPL3.0" for "CMF3.0"!*

Author's response.

They apply to CMF2.0. These issues have been corrected.

> ***Referee comment.***
>
> *B.13) In general, I think the section 6 on Conclusions and future work could be fleshed out.*

Author's response.

In our opinion, it is necessary to summarize the results. We tried to shorten and concrete this section.

> ***Referee comment.***
>
> *B.14) I think the "Code availability" section is not satisfactory regarding GMD standard but I will let the Topical Editor decide on this point.*

Author's response.

We can change this section if it is required.

> ***Referee comment.***
>
> *C) Minor comments:*
>
> *C.1) In the abstract, you write "As addition a parallel realisation of the EnOI (Ensemble Optimal Interpolation) data assimilation method as program service of CMF3.0 is presented." but this is not the only example presented in section 5.*

Author's response.

Corrected.

"As an addition, some information about the parallel realization of the EnOI (Ensemble Optimal Interpolation) data assimilation method and the nesting technology, as program services of the CMF3.0, are presented."

Referee comment.

C.2) P.3, L.13: It is not right to write that OASIS3 is the most popular version of OASIS as most groups are using OASIS3-MCT today.

Author's response.

The sentence has been changed to “The OASIS3 system was very successful and was widely used by many research groups around the world.”

Referee comment.

C.3) P.3, L.16: In OASIS3-MCT, MCT procedures are executed on all component model cores

Author's response.

Corrected:

“The new version, OASIS3-MCT (Craig et al., 2017) resolves the issue of sequential interpolation by using MCT procedures executed on all model component cores, instead of mapping through a standalone coupler.”

Referee comment.

C.4) P.4, L.33: Define SOA the first time it appears in the text

Author's response.

Done (in the last paragraph of Section 2)

Referee comment.

C.5) P.2, L53: Define WOM the first time it appears in the text.

Author's response.

Done (in the beginning of Section 2)

Compact Modeling Framework v3.0 for high-resolution global ocean-ice-atmosphere models

Vladimir V. Kalmykov[1,3], Rashit A. Ibrayev[1,2,3,4,5], Maxim N. Kaurkin[1,3,5], and Konstantin V. Ushakov[1,2,3,5]

[1]Hydrometcentre of Russia., B. Predtechensky per., 11-13, Moscow, 123242, Russia
[2]Marchuk Institute of Numerical Mathematics, Russian Academy of Sciences, ul. Gubkina, 8, Moscow, 119333, Russia
[3]Shirshov Institute of Oceanology, Russian Academy of Sciences, Nahimovskiy prospekt, 36, Moscow, 117997, Russia
[4]Moscow Institute of Physics and Technology (State University), Institutskiy per. 9, Dolgoprudny, Moscow oblast, 141700, Russia
[5]Federal State Budget Scientific Institution "Marine Hydrophysical Institute of RAS", Kapitanskaya str., 2, Sevastopol, 299011, Russia

Correspondence: Maxim N. Kaurkin (maksim.kaurkin@phystech.edu)

Abstract. We present a new version of the Compact Modeling Framework (CMF3.0) developed for ~~providing~~ the software environment ~~for~~ of stand-alone and coupled ~~models of the Global geophysical fluids~~global geophysical fluid models. The CMF3.0 ~~designed for implementation~~ is designed for use on high and ultra-high resolution models ~~at massive-parallel~~ on massively-parallel supercomputers.

The key features of the previous CMF version (2.0) are mentioned ~~for reflecting~~ to reflect progress in our ~~researches~~research. In the CMF3.0~~pure~~, 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 ~~the~~ a set of simultaneously working services. Performance tests for both versions are carried out. As ~~addition a parallel realisation~~ an addition, some information about the parallel realization of the EnOI (Ensemble Optimal Interpolation) data assimilation method ~~as program service of~~ and the nesting technology, as program services of the CMF3.0, is presented.

1 Introduction

As ~~was pointed~~ it was stated at the World Modeling Summit for Climate Prediction (Shukla, 2008), there is a general agreement that ~~a~~ much higher resolution of the major model components (atmosphere, ocean, ice, land) is a fundamental prerequisite for a more realistic representation of the climate system and more relevant predictions (e.g., ~~extremes~~extreme events, convection, tropical variability, etc.).

Along with the development of physical models of individual ~~components of the Earth system~~ Earth system components, the role of ~~the~~ instruments organizing their coordinated work (couplers and coupling frameworks) becomes more and more important. The coupler architecture depends on ~~the~~ complexity of the models used, on the characteristics of interconnections between the models and on ~~computer~~ the hardware and software environment. Historically, the development of couplers follows

the development of coupled atmosphere-ocean models. ~~On~~ At some level of complexity, the development of such software became an external problem relative to the development of individual components of the coupled model.

~~First~~ The first coupled models used simple algorithms for coordination of components through the file system. There was no separate coupler component, and communication between models was realized as a set of model procedures for input/output (I/O~~and~~) and for interpolation between global model grids (today this method is used, for example, in INMCM4.0 climate model (Volodin et al., 2010)). At the next stage, the coupling of components was done through ~~the~~ a separated central sequential hub using ~~multiply~~ the multiple executable approach (OASIS3 ~~(Valcke et al., 2012)~~(Valcke, 2013), Community Climate System Model cpl3 (Craig et al., 2005)).

Coupling through a shared file or ~~sequential component~~ through a sequential hub is acceptable only for models of relatively low resolution~~models. Increasing the size of arrays and~~ . Increasing of array sizes and of the number of model components in the system will inevitably become a ~~"bottleneck" because of the~~ "bottleneck" because of memory and performance limitations of a single processor core and also due to problems related to global network communications. Therefore, it was quite natural ~~,~~ that the next generation of couplers introduced parallelism in their internal algorithms (Community Earth System Model cpl6 (Craig et al., 2005), OASIS4 (Redler et al., 2010)). ~~Parallel~~ The parallel coupler architecture solves computational problems for fine grids, but increases complexity of algorithms.

~~New architecture of coupler~~ A new coupler architecture was introduced for the CESM1.0 model in 2012 (Craig et al., 2012). In this system, the coupled model has the form of a single executable and contains a high-level driver that calls ~~few component~~ a few standard component subroutine interfaces (init, run, finalize, etc.). This approach requires some reorganization of the components~~code and their representation in the interfaced style~~ ' code and its adaptation to the interfaces understandable by the driver, but simplifies model synchronization.

~~Coupled system~~ The coupled system also can be launched as ~~single~~ a single or multiple executable without a ~~standalone couplerwhose functions are~~ separate coupler, whose functions in this case are provided by a coupling library and performed in parallel on a core subset of each model ~~. Such solution has been~~ component. Such a solution was proposed in OASIS3-MCT ~~(Valcke, 2013)~~(Craig et al., 2017). A high-level driver controlling system sequencing is not required in this case.

Another important feature of the coupled ~~system~~ model is the scheme of working with the file system. In earlier versions, it was carried out independently by each model ~~in~~ component in a sequential way. Obviously, ~~such~~ this master-process scheme (used in CESM cpl6, OASIS3, OASIS3-MCT) was limited by the RAM of a node. Increasing amounts of model data lead to ~~active~~ rapid development of parallel I/O algorithms. Since version 1.0, the CESM system utilizes the PIO library (Dennis et al., 2012a) to establish parallel ~~data writing to~~ output in the NetCDF format by every component through several cores that play the role of writing delegates. ~~GFDL FMS (Balaji, 2012) system additionally suggests~~ In the GFDL FMS system (Balaji, 2012), fully parallel data storage with file ~~post processing~~ post-processing at the end of the run is offered.

Thus, we can point out the ~~main characteristics of coupling frameworks~~necessary features of modern coupling frameworks, which define their functionalities and characteristics:

1. coupling architecture (serial, parallel, with a high-level driver or as a set of procedures); the design of the ~~system defines~~ framework defines the complexity of development~~and~~ /maintenance of the coupled model and implicitly establishes performance limitations;

2. I/O-module architecture (serial or parallel, synchronous or asynchronous); it should be considered as a balance between simplicity of algorithms and the necessary rate of I/O;

3. ease of use; the level of system abstraction defines the convenience of user's work and the transparency of the overall coupled model;

4. performance; the choice of underlying algorithms defines the computational rate of the coupled model~~;~~ .

2 Background

Our work began with the development of a parallel version of ~~the~~ an ocean dynamics model. The aim at that time was to work out a ~~high resolution WOM~~high-resolution World Ocean model (WOM). We had to solve several problems, namely halo update, mapping (interpolation) of external ~~atmospheric~~ forcing data to the model grid, saving solution to a file, and gathering diagnostics. It was obvious ~~,~~ that separation of numerical ~~mathematics~~ algorithms for solving ocean dynamics equations from low-level service procedures is necessary to write a transparent code, which ~~will~~ would allow us to develop independently the physical model as well as service procedures.

This approach ~~has shown~~ showed its advantages in ~~solving the problem of coupling Global atmosphere and WOM for the medium~~ coupling the atmosphere and ocean general circulation models for medium- and long-term weather forecasts at the Hydrometeorological ~~center~~ Research Center of Russia. The purpose was to create ~~software capable to provide~~ the software capable of maintaining effective interaction of the high-resolution (~~of~~ on the order of 0.1 degrees) models of atmosphere and ocean ~~models with the~~ with a possibility to extend the coupled model ~~for incorporating the~~ by incorporating ice and soil components. The components of the coupled model were the ~~WOM~~ INMIO World Ocean model (Ibrayev et al., 2012) based on ~~INMIO ocean dynamics model~~ the MESH sea hydrodynamics code (Ibrayev, 2001) and the ~~SL-AV~~ SLAV Global atmosphere model (Tolstykh et al., 2017). It turned out that for coupling of several models one should solve similar problems as for a standalone model (mapping, I/O), but also has to provide synchronization and ~~data-consistency during interpolation~~ consistency of the interpolated data for simultaneously running components.

At the beginning of our study in 2012 there were several solutions for ~~the~~ creation of coupled models. It should be noted that ~~the state-of-art~~ state-of-the-art couplers, such as of CESM (with coupler based on MCT (Larson et al., 2005) or ESMF ~~(Collins et al., 2005)~~ (Theurich et al., 2016) packages) and OASIS, are fairly complex programs. The CESM cpl 7 (Craig et al., 2012) is written for a predefined set of components, and introducing a new model requires non-trivial changes and some work with internal structures. Adding a new grid still requires ~~self-constructing~~ non-automated constructing of interpolation weights for it (CESM). ~~Recent tests have shown that the~~ Tests showed that computational costs of ~~CESM coupler~~ the CESM coupler (including coupling, remapping and surface flux computation) are quite significant 20% (Craig et al., 2012)~~,~~ ; nevertheless,

good results ~~in~~ of 2.6 SYPD (Simulated Years Per wall-clock Day) rate were achieved for the ultra-high resolution Earth model (Dennis et al., 2012b).

The ~~most popular version of OASIS, the~~ OASIS3 ~~is~~ system was very successful and was widely used by many research groups around the world. ~~As was pointed~~ But, as it was pointed out, it contains a serial coupler, which is an obvious performance bottleneck ~~in the system both in terms of~~ due to constraints on memory and ~~from the point of view of~~ global communications. ~~New version~~The new version, OASIS3-MCT ~~(Valcke, 2013) solves the problem~~ (Craig et al., 2017) resolves the issue of sequential interpolation by using MCT procedures ~~, executed on subset of model cores~~executed on all model component cores, instead of mapping through a standalone coupler. Unquestionable advantage of this non-coupler design is the ~~absence~~ minimization of interference in the user code~~. System~~ , since there is no need to adapt it to interfaces required by a coupler. But still the system contains master-process I/O routines, which obviously ~~limits~~ limit its use for large grids. Even with parallel I/O, the solution with a subset of service processes provides double load on ~~model processes, which manage physical calculations~~the model cores, which at the same time perform solving of the model equations, coupling actions and I/~~O-routines~~O-tasks. Nevertheless, such ~~behavior~~ behaviour could be fully acceptable for runs with non-intensive mapping and I/O~~runs~~.

According to proposals of the Earth System Modeling conference (Valcke et al., 2012), today there are several ~~trends~~ common aspects in coupling software development~~, specifically: single executable modular architecture, parallel algorithms both for calculations and~~ : an ability to communicate data between components, regrid data, and manage the time evolution of the model integration. There is a lot of custom parallel coupling mechanisms, with either single or multiple executable approaches. We selected the approach with a single executable because it can simplify the program flow and give additional opportunities for performance optimization. Besides, we used the NetCDF standard for parallel I/O ~~, use of de-facto standard libraries like SCRIP (A~~ and SCRIP (Spherical Coordinate Remapping and Interpolation Package) (Jones, 1999) ~~and NetCDF~~for regridding, as done in OASIS3 (Valcke, 2013).

In the CMF2.0, ~~a framework for the~~ the framework for ocean-ice-atmosphere-land coupled modeling on massively-parallel architectures (Kalmykov and Ibrayev, 2013), we ~~realized~~ implemented these basic ideas.

In this paper we present two versions of the Compact Modeling Framework (CMF), ~~version~~ v. 2.0 and v. 3.0. As the CMF2.0 was published only in Russian (Kalmykov and Ibrayev, 2013), here we outline the basics of that version. In the CMF2.0 we combine the common proposals of the Earth system modeling community and ~~experimentation with low-level~~ an experimentation with low-level algorithms. We ~~concentrate on~~ focus on the single executable hub approach with a high-level abstract driver, optimized interpolation algorithms, asynchronous I/O routines, and tools for pre- and ~~post- processing~~ post-processing stages.

In the CMF3.0, the pure MPI approach is replaced with ~~PGAS (~~the Partitioned Global Address Space ~~) paradigm communications scheme, while~~ (PGAS) paradigm of communications, while the central hub architecture has evolved to ~~SOA-like architecture~~ a kind of service-oriented architecture (SOA) with a set of simultaneously working services and a common task queue.

3 CMF2.0 overview

3.1 Architecture of the coupled system

~~The~~ Any coupled model under ~~the control of CMF2.0~~ control of the CMF runs as a single executable, with each model component and the coupler using distinct processor cores. At the beginning, the global MPI-communicator is ~~divided on~~ split into appropriate groups according to ~~process decomposition~~ the requested communicator sizes of the model components and the coupler, and then all groups work simultaneously. The coupler performs some initialization routines and enters the time cycle of requests. ~~All physical~~ Following the *Template method* (Gamma et al., 1995), all model components do the same logical steps ~~, but call~~ by calling predefined abstract interfaces~~of models~~, for example, *ini_grid, ini_data, main_step, finalize*. ~~Realizations~~ Meanwhile, the CMF system does not know, what particularly will be done inside these routines. The realizations of abstract interfaces represent ~~specific behavior of the model~~ the specific behaviour of every model component: initializations and registration in the system of all data arrays that will be involved in ~~the exchanges between models;~~ component-component exchanges and in I/O; the main step of physics equations solving for the particular model component; finalizing procedures, etc. ~~This behavior could be easily extended with interceptor methods (programming pattern *Template method* (Gamma et al., 1995)).~~

That is, ~~to work in a coupled system,~~ in order to add a physical model to the coupled system a user only has to define ~~derived class of~~ the physical model adapter ~~that inherits the base Component class~~ (the required template is provided) and to realize ~~abstract interfaces, filling it~~ its abstract interfaces (filling them with calls to his internal model subroutines). This approach allows one to generate different executables for different coupled model combinations (e.g., switch between ocean simulations with different sea ice models) and restricts the user from any changes in the code outside of his ~~derived class~~adapter. Also the addition or modification of components does not affect the main ~~program~~ CMF code, because it is written for ~~abstract Component~~an abstract component.

3.2 Coupler-model interactions

~~Each~~ For any model component, its decomposition is generated by the CMF2.0 system in such a way that each coupler core interacts only with a specific subset of the component cores~~, which means locality of data and communications during the interpolation process or I/O actions. The~~ . This allows to reduce the required amount of communication routes to the coupler for every component from $M \times N$ (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 component core ("slave") now interacts just with its "master" coupler core. 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 1D- or 2D-decomposition narrower than the others. An example of the coupled model ~~for~~ with 3 coupler cores and 3 ~~parallel~~ components is shown ~~in Figure~~ on Fig. 1.

All ~~actions~~ events in the system are divided into few classes ~~:~~ (save diagnostics, save control point, read file data, send/receive mapping, etc.~~All these events have their own periods and define~~), defining different actions with data ~~fields.~~

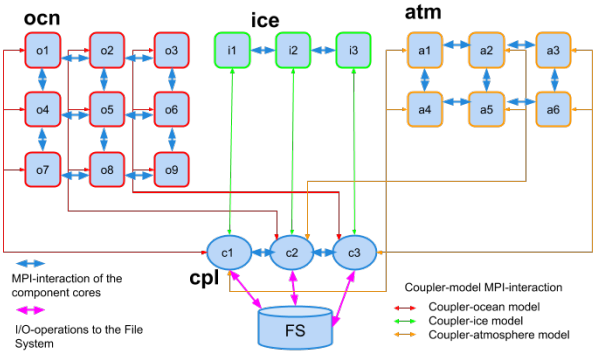

Figure 1. Architecture of the coupled model ~~in~~ run under control of the CMF2.0. ~~For~~ In this example there are three components (ocean, atmosphere, ice) connected by the 3-core coupler.

~~Since all events in the system~~ arrays. In the CMF2.0, we postulate that all events could be predefined before ~~start~~the start and occur with fixed periods. Thus, the coupler ~~during initialization gathers~~ 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 ~~time of all the events . This information is used to switch between requests of components without synchronization, while~~ 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. ~~Also persistent MPI operations (combination~~ Moreover, it becomes possible to use persistent MPI-operations (combinations of *MPI_SEND_INIT* and *MPI_STARTALL*) ~~are used~~ for all events~~to save time on~~ , thus saving time of repeated communications. ~~Pointers to arrays are stored at the registration stage, thus sending and receiving operations will be carried out without explicit user calls but based on defined periods.~~ Combination of predefined time chain, persistent communications and ~~pointer based asynchronous data sending provides maximal efficiency of~~ pointer-based asynchronous sending provides high efficiency of parallel data gathering and distribution.

3.3 Coupler: mapping

The interpolation algorithm uses SCRIP-formatted weight files built at the pre-run (off-linestage with the SCRIP package . At the run phase, components send data asynchronously, and a subset of component 's cores works only with individual master core in the coupler) stage by by means of the CDO package (http://mpimet.mpg.de/cdo). In the beginning of the run stage, the weight files are read by the coupler in parallel. The data intended for mapping is sent by each component core to its master core asynchronously, without blocking.

The regridding process is performed on in the coupler communicator and implemented as sparse matrix multiplicationfor the two cases, of the source and destination type (Craig et al., 2005)and currently supports is implemented as a sparse-matrix – vector multiplication. It supports logically-rectangular gridsgrids. We implemented the "source" and "destination" parallel mapping algorithms (Craig et al., 2005), which correspond to the weights being distributed according to the destination or source grid decomposition, respectively. The former is usually more efficient if the source grid has fewer points than the destination one, and the latter vice versa.

Process The process is performed in the SCRIP format, where links connect i.e. SCRIP-type links connect cells of destination and source cells (indices) grids with appropriate weights. Since single every coupler core works only with a subdomain of the global model grid, it has only part of source a part of the source grid data in memory, while other . Other data should be gathered from neighbors by MPI-routines on every interpolation step. All necessary links neighbour cores during every interpolation event, which is functionally analogous to calling the Rearranger routines of (Jacob et al., 2005). Both the component-component SCRIP links and intracoupler rearrange routes are initialized at the beginning of run and are the run and used at the calculation run stage as persistent(Jacob et al., 2005).

At calculation stage During the interpolation event, every coupler core first prepares and sends source cells required by its neighbors, then it processes its local area and at lastneighbours. Then, while this data is being sent, it weights its local source cells. And, at last, receives the missing data , completing the interpolation processand completes the weighted sums on the destination grid. It is worth noting, that links are the data is not sent directly, but as sorted unique cells vectorswhich allow cell vectors. This allows one to avoid sending duplicated data which could be the case when a source cell is used in a few destination cells. As a result, there is an overlap of computations and communicationcommunications, which, in conjunction with persistent MPI transactions, determines a high efficiency of the algorithm.

Several The performance rate of the CMF2.0 interpolation system was evaluated in several "ping-pongtests were carried out for interpolation system using coupled " tests, in which the coupler was maintaining component-component exchanges of the INMIO-SLAV ocean-atmosphere model. The Test I condition, as in (Valcke et al., 2012), (Craig et al., 2012) is an exchange between two components with disabled physics routines. In our testmodel with disabled solvers of physics equations (similarly to the ping-pong test of OASIS3 in (Valcke, 2013)).

In the Test I, the ocean model sent 3 sends three 2D-fields every 2 hours to the atmosphere model and received 9 receives nine 2D-fields every 1 hour.

The ocean model has the 3600×1728 ~~tripole~~ tripolar grid and the atmosphere ~~–~~ model has the 1600×864 latitude-longitude grid (grids were taken from the current versions of ocean (Ibrayev et al., 2012) and atmosphere (Tolstykh et al., 2017) models). The mapping process consists of gathering data from the source component, regridding ~~process inside of~~ inside the coupler communicator and distributing the result to the destination component. The test ~~has~~ was run for 10 model days, which corresponds to 240×9 atmosphere-ocean mappings and 120×3 ocean-atmosphere mappings. Sizes of communicators for the ocean and atmosphere ~~models~~ model components were fixed by 1152 and 288 cores, respectively. While not performing any computational work, they allow to ~~simulate~~ imitate real communication load of the overall system, reflecting packing, MPI sending, and unpacking costs. Thus~~charts present~~ , the charts present a strong scalability of the coupler interpolation algorithm. ~~Performance is based on a standard Intel Fortran compiler.~~

Results were obtained on four supercomputers: MVS-100k, ~~MVS-10p~~MVS-10P, BlueGene/P, BlueGene/Q (characteristics ~~in~~ are provided in the Appendix). ~~Test results for~~ On all supercomputers, the coupled system was compiled with a standard Intel Fortran compiler. Timing results of the 10-day Test I on MVS supercomputers are presented in ~~Figure~~ Fig. 2. Two configurations~~–~~ , with 16 real and 32 virtual cores per node, are shown for ~~MVS-10p~~the MVS-10P. The difference in the speed of their work is expected and is a result of increased communication load for a ~~large~~ larger number of cores per node. The graph shows good scalability with increasing ~~number of coupler size~~size of the coupler communicator. The best result of 1 second is achieved at 288 coupler cores.

It is clear that 20-40 coupler cores provide a satisfactory speed for such problems, because ~10 seconds costs for 10 model days is a rather insignificant value for the high-resolution ocean-atmosphere coupled modeling. The figure also shows ~~failure~~ ineffectiveness of the sequential algorithm: even on the fast ~~MVS-10p processors~~ MVS-10P processors the service activity takes about 200 seconds~~(work of the~~ . Besides, the work of sequential algorithm is only possible ~~with restriction that~~ if all the node memory is allocated ~~only for~~ for the interpolation block, which is ~~impossible in practice~~ unlikely in practice (in our test we had to switch off physical model arrays allocation). Good coupler performance for one component-component connection is necessary for overall performance with growing number of components and their grid resolution. ~~Test results for~~ Results of the same test for the BlueGene supercomputers are presented in ~~Figire~~ Fig. 3. Timing of the algorithm is ~~worse than on MVS-10p~~ weaker than on the MVS-10P because of lower individual processor rate.

The Test II was conducted for estimation of the increasing communication load associated with the growth of components' communicator sizes. ~~Model~~ The timing still refers to the 10-day experiment with disabled physics. But the model grids were decomposed on much higher number of subdomains, increasing the cost of gather/distribute phase of the test (mapping process inside the coupler communicator remains the same). The results are shown in ~~Figure~~ Fig. 4 (curves replaced by point symbols to improve the readability of the graph) ~~.~~ and, for convenience, repeated in the Table 1. Numbers of cores used for the ocean and atmosphere models were equal to 8640 and 3456, 10368 and 4320, 17280 and 13824, respectively.

~~Graph~~ The graph shows two interesting facts. Firstly, ~~single core coupler configuration does~~ single-core coupler configurations do not work for the Test II because of memory limitations. Secondly, increasing of ~~communicational load~~ communication load (i.e. gather/distribute) affects performance only on small ~~number of cores and at these points evaluation time is worse than in Test I~~numbers of coupler cores. For example, test times for ~~2~~ two coupler cores for model communicator sizes (8640, 3456)

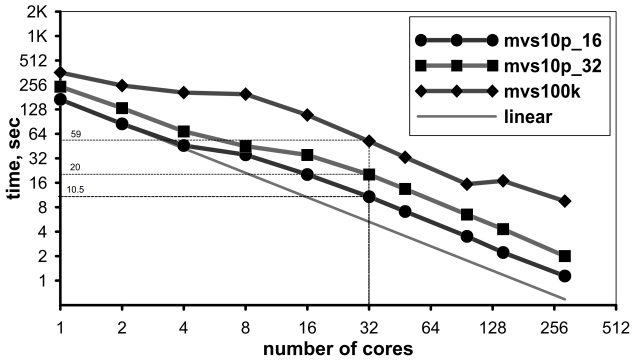

Figure 2. ~~Time in seconds of Test I~~ Walltime required for the 10-day ocean-atmosphere model run with disabled physics vs. number of coupler cores on MVS ~~.~~ supercomputers (Test I for CMF2.0).

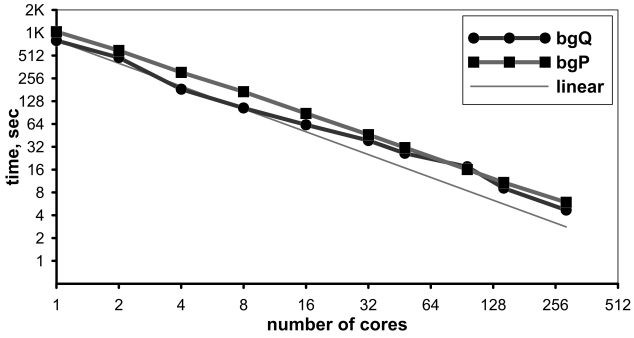

Figure 3. ~~Time in seconds of Test I~~ Walltime required for the 10-day ocean-atmosphere model run with disabled physics vs. number of coupler cores on BlueGene ~~.~~ supercomputers (Test I for CMF2.0).

and (10368, 4320) are correspondingly 26% and 42% higher than for Test I communicator sizes (1152, 288). For ~~8~~ eight coupler cores this difference becomes 13% and 22%, correspondingly. Since every coupler core communicates only with ~~few component cores(that is, performs only local communication)~~a subset of component cores, increasing of the coupler communicator size leads both to decomposing of the interpolation computations and to decreasing of ~~communication overhead~~the component-coupler communication overhead, though slightly increasing intracoupler rearrangement communications. As a result, even few tens of coupler cores are suitable to provide good performance of high-resolution mapping with huge sizes of model communicators.

3.4 ~~Effectiveness of different I/O schemes and CMF I~~Input/~~O block~~output scheme

Since the speed of I/O-operations on supercomputers is ~~usually~~ often slow, writing large amounts of data (such as control points ~~which~~ that include several 3D-arrays) can take unacceptably long time. In case of frequent data dumps ~~(e.g. forecast model~~

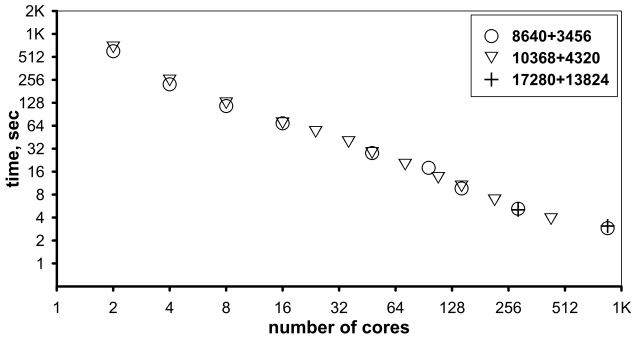

Figure 4. ~~Time in seconds of Test II~~ Walltime required for the 10-day ocean-atmosphere model run with disabled physics vs. number of coupler cores on ~~BlueGeneQ system~~ BlueGene/Q supercomputer, for different ~~configurations~~ decomposition sizes of the ocean and atmosphere models ~~.~~ (Test II for CMF2.0).

~~with 1 hour period of saving data),~~ or slow file system, the time of calculations could be even comparable to the time of I/O, thus it is very important to optimize ~~file system interactions.~~

~~There are four known strategies for working~~ interaction with the file system~~: by master, direct parallel, by delegates and by external delegates~~. Its realization can be synchronous (blocking) or asynchronous (non-blocking) (see, e.g., (Balaji et al., 2017)).

~~Time of the experiment with solution subsequently recorded by master process scheme T_{total} consists of time for solving the model equations T_{run}, time for global data collection from n cores on a single master process $T_{gather\_glb}(n)$ and time for global array recording time T_{write_glb} :~~

$$T_{total} = T_{run} + T_{gather\_glb}(n) + T_{write\_glb}$$

~~Time of~~ In the former case, I/O operations are performed by some subset of the ~~experiment in the case of direct parallel scheme consists of computation time, time for recording by n cores to one file $T_{write\_lcl(n,1)}$ (or to different f files $T_{write\_lcl(n,f)}$ and then combining them $T_{u\_files(f)}$):~~

$$T_{total} = T_{run} + \begin{cases} T_{write\_lcl}(n,f), & \text{f=1} \\ T_{write\_lcl}(n,f) + T_{u\_files}(f), & \text{f>1} \end{cases}$$

~~In this case:~~

$$T_{write\_lcl(n,1)} > T_{write\_lcl(n,f)} \; if \; (f > 1),$$

~~since parallel writing to a single file is slower than to separate files. Time of the experiment in case of the delegate scheme consists of T_{run}, local data collection on n delegates $T_{gather\_dlg(n)}$ and time of parallel recording local arrays $T_{write\_dlg(n)}$:~~

$$T_{total} = T_{run} + T_{gather\_dlg}(n) + T_{write\_dlg}(n)$$

Table 1. The same data as in Fig. 4

OCN+ATM cores →	8640+3456	10368+4320	17280+13824
CPL cores ↓	Time, sec		
2	608	680	-
4	224	224	-
8	116	116	-
16	69	69	-
24	-	53	-
36	-	30	-
48	28	28	-
72	-	19.7	-
96	18	-	-
108	-	13.2	-
144	9.6	10.2	-
216	-	6.7	-
288	5.2	-	5
432	-	3.8	-
864	2.9	-	3.1

"-" denotes not tested or unsupported configurations

processor cores of physical model components, thus inhibiting the physical equations solving.

~~Recording time of parallel *n*-delegates scheme is not always better than that of sequential writing. Increase~~ In the latter case, this inhibition is avoided at the cost of allotting distinct cores to specific I/O services and making procedures for data transfer between these services and the physical components. It is worth noting that increase in the number of writing ~~processes~~ cores does not always increase the recording speed, but often reduces it. ~~Such behavior~~ The particular behaviour is defined by actually installed supercomputer hardware. For example, presence of a single I/O channel for ~~whole cluster could serialize parallel~~ the whole machine can serialize the I/O, and, in opposite, multiple special I/O-nodes may allow one to achieve even some acceleration. ~~At last, time of the experiment~~ But, even in the case of ~~external *n*-delegates scheme~~ slow hardware, the total time of model experiment can be equal to the time of ~~calculation:~~

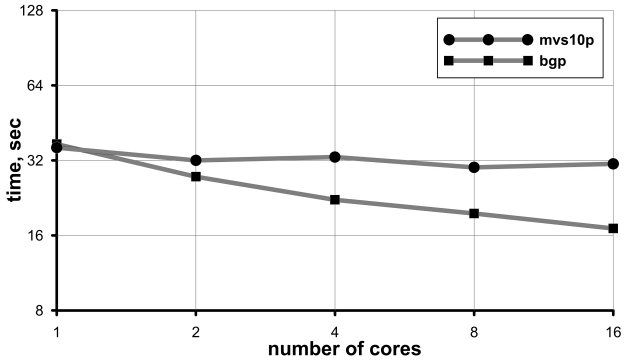

Figure 5. Walltime of parallel writing of a model array of $4096 \times 2048 \times 50$ size by different numbers of CMF2.0 coupler cores on MVS-10P and BlueGene/P supercomputers.

$T_{total} = T_{run}, if\ (T_{write\_dlg}(n) < T_{run}).$

Additionally, asynchronous component data sending makes time of the data collection phase insignificant, since calculating component is completely separated from writing delegates and can continue calculations without blocking: $T_{gather\_dlg(n)} \approx 0$. Limitation in expression appears due to the fact that the physical equations solving due to overlapping of computations and I/O. The scheme allows a model to accelerate until the writing time is less then the time of performing the chunk of calculations. This limitation is controlled by required bandwidth :

$B = D/T_{run},$

where D is the amount of data to be saved and depends on actually installed hardwarethe hardware bandwidth and by the model-service data transfer realisation.

Asynchronous scheme was incorporated in the latest version our framework. Since The asynchronous approach is generally more flexible, so it was chosen for the CMF. In the CMF2.0, all I/O algorithm is parallel, actions are performed by the coupler. The realization is fully parallel, so one can work with any grid sizes just increasing the number of I/O cores. We have tested asynchronous coupler cores. Test results of the CMF2.0 I/O scheme with INMIO World ocean model for grid sizes $3600 \times 1800 \times 50$ (basic resolution), $5400 \times 2700 \times 50$ and $7200 \times 3600 \times 50$. Saving of control point, which includes four 3D and five 2D arrays, was successfully carried out by the coupler. In real applications rare saves could be fully overlapped by calculations on account of asynchronous messaging by model componentssystem for the case of writing a single-precision model array of $4096 \times 2048 \times 50$ size are shown on Fig. 5.

It can be seen that the writing speed of MVS-10P is approximately constant. Moreover, the timing does not change when writing cores are allotted on one or several nodes. The reason is that MVS-10P has only one I/O node and all file operations are

performed through it. On the other hand, the BlueGene/P system has several I/O nodes, so it allows to obtain some acceleration of writing. Nevertheless, the main advantage of the CMF I/O system is its asynchrony and memory scalability. The acceleration obtained on the BlueGene/P system is more of a nice result. It draws attention to the need of developing I/O infrastructure on supercomputer systems. It's obvious that scalability graphs of a future exaflops machine with millions of cores become very artificial if one has to work with the file system through a single channel.

3.5 Additional features

Apart from the coupler, the framework also includes two helpful blocks. ~~At preprocessor stage,~~ For the pre-run stage, the CMF2.0 has got the off-line block~~for constructing~~ , which constructs SCRIP interpolation weights and ~~preparation of the~~ prepares initial condition files. ~~It also exploits *Template method* pattern and reduces all preparation actions (like~~ Like the run-stage CMF program, it is implemented in terms of abstract operations, which reduces all model configuration actions required from the user (e.g., grid definition) to realization of a few abstract interfaces in ~~user derived~~ a user-derived class.

At the run stage, the user can call ~~different~~ various utility modules, like the HaloUpdater, which is ~~extensively used in WOM~~needed in finite-difference models. It uses 4-neighbour scheme of any length/dimension/type update on latitude-longitude and tripolar grids, still handling diagonal cells. Impact of the HaloUpdater on performance of the INMIO WOM is described later.

Also the CMF2.0 provides helpful tools for automatic building of various model combinations, makefile and skeleton class generation, ~~preprocessing scripts, and~~ data preprocessing, and for other infrastructure actions.

4 CMF3.0

4.1 ~~PGAS-communicator~~PGAS abstraction

The CMF2.0 has shown itself as a suitable framework for high-resolution coupled modeling, allowing us to perform long-term experiments which would be impossible without it. But ~~CMF-2~~the CMF2.0 still has several points for ~~improvements~~improvement. First of all, although the pure MPI-based messaging is quite fast, it needs explicit work with sending and receiving buffers. Additionally, development of nested regional ~~sea submodels~~ models becomes quite difficult using only MPI-routines. The CMF2.0 test results showed that we can easily sacrifice some performance and choose better (but perhaps less computationally efficient) abstraction to simplify messaging routines.

We have chosen the Global Arrays library (GA)~~(Nieplocha et al., 2006), which realizes PGAS~~, which implements the Partitioned Global Address Space (PGAS) paradigm of parallel communication ~~.~~ and provides an interface that allows to distribute data while maintaining the type of global index space and programming syntax similar to that available when programming on a single processor (Nieplocha et al., 2006). The general idea is to give the user easy access to different parts of a distributed array. The PGAS abstraction assumes that there is a virtual huge array, which is accessible from any process involved in its creation. In fact, there is no global array, and its parts are stored locally in the memory of processes. But the

user does not know about it, since the library takes over all the details, due to which the simplicity is achieved. For example, the client on process X may request an array element with indexes [i, j], as if it has direct access to it. Behind the scenes, GA learns which process holds this element (process Y), executes MPI send-receive transfer and returns the result to the client.

Development of this idea ~~resulted in class Communicator~~in the CMF3.0 has resulted in the class Communicator_GA, which encapsulates the logic of working with the GA and provides ~~API (Application programming interface)~~ an interface for put/get operations of ~~array patches from different components~~sections of global arrays from different model components and services. Moreover, this ~~API~~ interface could be used not only for connections between ~~nested models~~ models (including nested ones), but also as a communication mechanism between the models and the coupler, because it allows one to hide all decomposition-to-decomposition problems rising in ~~distributed~~ distributed-memory applications. In the CMF3.0, every array, which participates in intermodel communications, has its ~~"mirror" in~~ "mirror" in the corresponding virtual global array. When the model needs to perform some action, it puts/gets data to/from the global array (this operation is local since ~~global array internal distribution perfectly matches~~ the global arrays' processor-wise allocation perfectly matches the model decomposition) and continues calculations. Service components get ~~array from~~ the array from *the other side*, but this time on their own ~~decomposition~~decompositions. For example, the ocean component could store a global array on 5000 cores with some 2D decomposition, while the I/O procedure ~~for saving~~ outputting this array could utilize only 4 cores and a 1D decomposition.

4.2 SOA-architecture

As the complexity of coupled models is growing~~we need more~~ , we need an easy and convenient way of connecting ~~physical models together. SOA (Service Oriented Architecture)~~ model components together. The SOA, which was originally introduced for web applications, gives a good pattern for component interactions. In the CMF3.0~~all models~~ , all model components send their requests to ~~common queue. Service~~ the common message queue. The service components receive only the messages they could process, then get data from appropriate global arrays and perform the required actions. Such architecture allows us to minimize dependencies between physical and service components~~and make~~ , and makes development much easier. Moreover, since all services in the CMF3.0 ~~inherit~~ are based on the same template (inherit the base class Service), it also allows ~~one~~ the user to easily add new ~~. Now~~services to the system by filling only few abstract interfaces. Today, we have four completely independent services built into the CMF3.0: CPL (~~for field~~ mapping), ~~IOF (fast~~ IOD (I/O ~~device), IOS (slow I/O device~~service), NST (nesting service), DAS (~~Data Assimilation Service~~data assimilation service).

The CPL service represents the coupler from the CMF2.0 and serves all mapping requests. It receives data ~~using Communicator~~through the Communicator_GA class routines, performs interpolation and pushes data to the destination global array (without ~~request from~~ a request from the receiving side). Although the central coupler architecture of CMF2.0 allows one to collect all service operations on one external component and to perform each of them in parallel, simultaneous requests sometimes can lead to inefficient usage of ~~process~~ processor time. For example, ~~coupler in~~ the CMF2.0 coupler can not perform parallel mapping and parallel I/O operations together. This is a disadvantage of all ~~I/O-schemes which~~ data transfer schemes that combine two or more actions on one process.

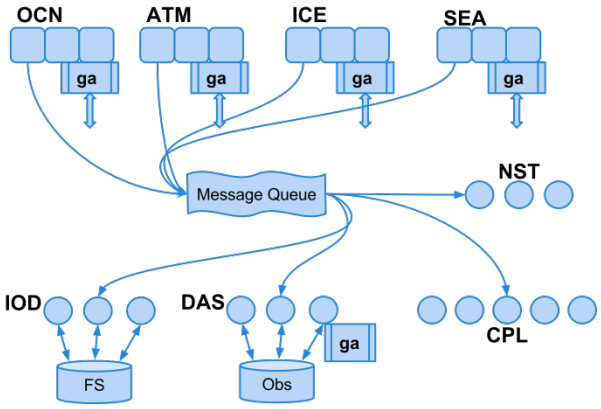

Figure 6. The architecture of the compact framework CMF3.0. There are four components in this example: ocean model (OCN), ice model (ICE), atmosphere model (ATM) and sea model (SEA). ~~They~~ The components send requests to the common message queue, from where they are retrieved by the coupler (CPL), data assimilation (DAS), input and output data (IOD), and nesting (NST) services. The data itself is transferred through the mechanism of global arrays, which ~~are~~ is also used for interprocessor ~~communication~~ communications in the components and services.

In the CMF3.0, we decided to pick out a separate I/O-service, responsible only for working with the file system. For example, when writing data to a file, on the component side it works as follows: the model component has to wait till the corresponding GA-array is free; put the data into the GA-array; mark the GA-array as full; send request for the IOD service. On the IOD side the request is read by the service; the service then takes the requested data from the GA-array; marks the array as free; calls NetCDF routines (same as in the CMF2.0) for parallel writing to file. This approach, though not expected to perform faster than the CMF2.0 direct MPI messaging, provides a flexible and fully asynchronous data writing, limited mainly by the bandwidth of the file system.

A performance test of the CMF3.0 I/O system in the INMIO World ocean model of 0.1°resolution is shown on Fig. 7. In this test we compare the walltimes of 8-day ocean model runs (full physics equations solving) on 600 ocean and 10 IOD cores of MVS-10P system with different frequencies of solution control point (CP) writing to file. The frequencies range from one CP per 6 model hours to one CP per 8 days, the case of no output is also examined. One CP output (file of size 8 GB) takes about 5 wallclock minutes, while one model day physics calculations take about 10 minutes, so one may expect that overlapping of computation and output will be possible if control points are written not more often than twice a model day. Indeed, we see that the graph shows linear growing of required run time if output frequency is greater than 2 CP's per day. This is a quite satisfactory value for long-time experiments such as Earth climate studies (e.g., (Marzocchi et al., 2015), where (1/12)°global ocean model outputs are stored as successive 5-day means).

It should be noted, that one external I/O-service ~~still~~ solves only part of the problem, because in case of ~~combining slow control points and fast diagnostics requests,~~ writing large control point files and dumping frequent lightweight diagnostics the

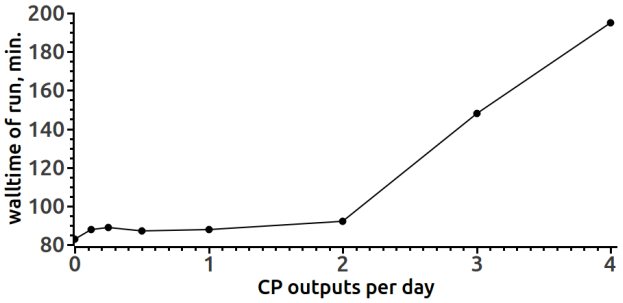

Figure 7. Walltime of 8-day run of INMIO World ocean model with 0.1° resolution vs. frequency of saving solution control points.

model still would be blocked by the former. Therefore~~we break service~~, the service may be further split into two parts ~~: IOF and IOS~~ – fast and slow I/O-devices~~(because of~~. Due to the abstract structure of ~~Service this separation is~~ the Service class this separation can be done via few lines of code~~). This mechanism provides flexible and fully asynchronous data storage, limited only by the bandwidth of file system as described earlier~~.

~~The further development of~~ Further development of the CMF has included data assimilation algorithms. For the ocean model, we have added the new DAS-service~~which implements~~, which implements the logic of parallel data assimilation (Kaurkin et al., 2016a), (Kaurkin et al., 2016b).

4.3 Class Communicator_GA

The Communicator_GA is a CMF3.0 system class that represents a kind of facade for the GA library. That is, it defines a high-level interface and hides some subtleties of the GA from the user. For example, the class allows to create an array that will be distributed on one component, but still visible to another component. It can be a temperature array that is physically distributed over the ocean's cores (and they can read and write data to it), but, in addition, the CPL service can also work with this array, although it does not store any part of it. Creation of such global array in CMF3.0 will require just a few subroutine calls:

- request the CMF system for component identificators and process lists of currently running ocean model and CPL service;
- register this joint group of processes, prescribing ocean as the holder and CPL as the subscriber;
- request the system for current ocean decomposition;
- register the array, specifying the ocean as the holder and passing its decomposition (so that GA distributes the mirror array in the same way as the model component does with the original one), and the coupler as the subscriber.

The GA put and get operations now may be called. For the holder side they will be local due to consistency of the decomposition. One of the benefits of this architecture is that now the model decomposition can be arbitrary. For example, it becomes easy not to reserve processor cores for subdomains of an ocean model that lay on land.

Every put/get operation must maintain explicit synchronization by setting the array status accordingly to "full" or "empty". This is required since we are not allowed to "lose data". That is, even if some component (e.g., ocean model) is faster than another component (atmosphere model, or IOD in case of too frequent data dumps), we must not lose an array. Accumulation of arrays in a queue also will not lead to success, since models usually work at constant speeds and, as a result, the queue will soon exhaust all available memory. So, if the "fast" model is ready to put/get data, but the GA array is still occupied/empty, the model is blocked.

4.4 Interpolation

Since the logic of interpolation subroutines in the CMF3.0 remains the same ~~, we~~ as in the CMF2.0, we can greatly simplify it by ~~using~~ use of GA abstractions. Now, all source data needed by the destination cell is collected directly by Communicator~~routines. Optimizations regarded to ignore repeated cell requests~~ GA routines. The optimizations regarding repeated cells are preserved. Disadvantage of using ~~GA is decreasing~~ the GA is a decrease in performance, since it can not provide persistent operations, overlapping of computations and ~~communications~~ communication in one service, and obviously has its own overheads. We take the same parameters and input files ~~of~~ as of the Test I to compare the CMF3.0 performance with that of the CMF2.0 (~~Figure 6~~Fig. 8). Again, we measure the overall timing including costs of sending event request, sending data, interpolation process and pushing data into ~~final~~ destination arrays. Therefore, ~~timing reflects~~ this timing reflects the overall system overhead ~~additional to~~ against the timing of physical ~~models~~model components.

Tests were conducted on ~~Lomonosov supercomputer (characteristics in Appendix). Graph~~ the MVS-10P supercomputer configuration with 16 cores per node. The graph shows that results are ~~worse than for~~ weaker than those for the CMF2.0 (Fig. 2 ~~) as expected, but~~ curve mvs10p_16 is repeated here), but the linear scalability trend is preserved. Moreover, the rate of 2-3 seconds per modeling day (on 20-50 CPL cores) is quite satisfactory for our practical purposes in high-resolution ~~experiments.~~ numerical experiments. The decline in performance is expected due to the overhead of using GA-arrays (as an intermediate send/receive data representation) and due to deprecated MPI_SEND_INIT procedures in the CMF3.0. This is a sacrifice for the compact code representation and for convenience of adding new features.

4.5 Additional features

In the CMF3.0, services are responding to messages during all run time~~, therefore model can send requests at every~~. Events are requests for certain actions on data arrays. So, the model is allowed to send such requests unexpectedly, at any step of its time cycle. Nevertheless, sometimes we ~~know~~ may know a schedule of actions (e.g., sending mapping every 2 hours, diagnostics every day and control point every month). The CMF3.0 provides a simple mechanism for generation of such scheduled actions ~~. Now~~ in order to save the user from having to keep track of time and send requests at the right moments. At present, we have two types of event generators: NormalEvent, which represents uniform actions (like ~~diagnostic~~ diagnostics saving, etc.),

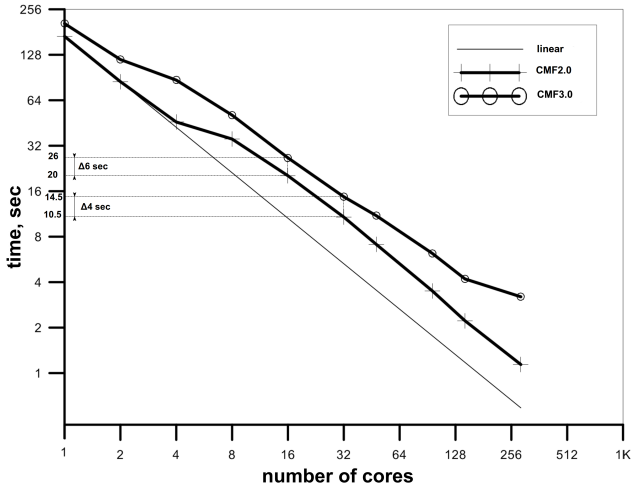

Figure 8. ~~Time in seconds of Test I~~ Walltime required for the 10-day ocean-atmosphere model run with disabled physics vs. number of coupler cores on ~~Lomonosov~~ the MVS-10P supercomputer ~~.~~ (Test I for CMF2.0 and CMF3.0).

and SyncVarEvent, which allows one to synchronize with ~~time variables in NetCDF-files~~ the time axis of a NetCDF-file (it is useful for ~~prescribed forcing experiments like (Griffies et al., 2009)). Generators realize abstract classEventGenerator, so new specific~~ experiments with prescribed forcing referenced to the real calendar, e.g. the Drakkar Forcing Set (Dussin et al., 2016)). Depending on current model time, these objects send requests or do nothing. Generators are realized by an abstract class, so for new specific events generator subclasses could be easily added.

~~For asynchronous events~~ In case of unexpected behaviour (like exceptions in model physics or changes in external data) the user can directly call ~~raise event~~ the raise event routine, e.g., for emergency data dump ~~before termination or even change behavior of other models using~~ or even to change the functioning of other model components by special messages.

The first to respond to an event is the model component itself – it looks at the event's type and determines what to do (e.g., in the case of saving diagnostics: put the data into the ga-array, mark it as full, send a request to the services, and continue running). Then, the event is packed into an MPI-message and sent as a request to all services (if the model has decided to send it). Services unpack the event, look at its name, and either process it, or ignore.

Other parallel utilities implemented in the CMF3.0 include

- Array operations, such as resizing, changing index order, converting to string and back, search for a particular element;
- Calculating global sums and area integrals over a decomposed model field, which is important in maintaining conservation in geophysical fluid models (e.g., to correct the precipitation, evaporation and runoff algebraic sum in stand-alone ocean simulations like (Griffies et al., 2012));
- Memory usage monitoring.

In the CMF3.0, we included all pre- and post-processing utility modules available in CMF2.0. It is not difficult to migrate from the CMF2.0 to the CMF3.0. Only one ~~file-adapter (~~adapter file, about 200 lines of code~~) ~~, should be rewritten. It contains several procedures (*ini_main, make_step, finalize*, etc), besides ~~global~~ defining events and arrays (~~IO~~intended for I/O, remapping, etc.) registered in it.

5 CMF examples of usage

There are several examples of using CMF for various ~~numerical geophysical~~ geophysical numerical models:

1. ~~High-resolution~~ Eddy-resolving ocean dynamics modeling using ~~WOM INMIO governed by~~ the INMIO WOM with 0.1°resolution, governed by the CMF2.0 (Ibrayev et al., 2012),~~(Ushakov and Ibrayev, 2017)~~(Ushakov and Ibrayev, 2018).

2. Data assimilation ~~using DAS~~ of satellite observations and ARGO floats measurements ~~for~~ using the DAS service in forecast and reanalysis ~~with~~ experiments with the INMIO WOM governed by the CMF3.0 (Kaurkin et al., 2016a).

3. There is a set of works with coupled atmosphere-ocean models for climate change ~~modelling~~ research and numerical weather prediction~~at different spatial-time scales. The atmosphere model SL-AV~~ . The SLAV global atmosphere model (Tolstykh et al., 2017) and the ~~WOM INMIO (Ibrayev et al., 2012) are coupled using~~ INMIO WOM (Ibrayev et al., 2012) were coupled using the CMF2.0 and CMF3.0 (Fadeev et al., 2016). The results of numerical experiments with the coupled model demonstrate agreement with observational data and show a possibility to use this model for probabilistic weather forecasts at time scales from weeks to year.

4. ~~Nesting technology (as a~~ The nesting technology implemented in the CMF3.0 ~~software NST service)~~ NST service has been tested for the local ~~model of Barents Sea (INMIO-based)~~ INMIO-based model of the Barents Sea with a resolution of 0.1°and the INMIO WOM with a resolution of 0.5°with different geophysical parametrizations (Koromyslov et al., 2017).

5. ~~The first~~ First results of the seasonal variability simulation for the Arctic and North Atlantic ocean waters and ice by the coupled ~~model based on~~ INMIO WOM and a sea-ice ~~model~~ CICE5.1 (Turner and Hunke, 2015) models were obtained under the CMF2.0 in (Ushakov et al., 2016). The numerical experiments ~~have been~~ were performed in conditions of the CORE-II protocol~~(Ushakov et al., 2016)~~.

5.1 INMIO World Ocean Model

As it was mentioned, one of the goals of the CMF is to provide tools for effective parallel calculations of stand-alone models. Historically, it was developed to provide efficient support for the INMIO WOM. ~~The INMIO WOM (Ibrayev et al., 2012) utilizes 2D-decompostition~~ This model utilizes a 2D-decomposition of the tripolar grid. Increasing the number of cores decreases (almost proportionally) the number of performed operations for each process, ~~because~~ since the model uses explicit time

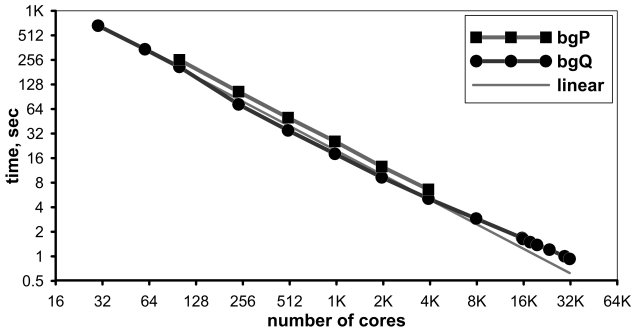

Figure 9. ~~Time in seconds~~ Walltime of the 0.1° resolution INMIO WOM (governed by the CMF2.0) 10 ~~time~~ model steps vs. ~~cores number of BlueGeneP~~ model communicator size on the BlueGene/P supercomputer (Moscow State University~~. University~~) and ~~BlueGeneQ~~ BlueGene/Q supercomputer (IBM Research Center Thomas J. Watson).

schemes for horizontal operators, which require only local halo updates. Therefore, limitations in scalability can only be associated with halo update routines and external blocks (e.g.~~in~~ , in the I/O system).

~~Latest~~ The latest version of INMIO WOM ~~model was fully integrated to CMF~~is distributed in an integrated package together with the CMF2.0 and 3.0, all necessary libraries and a standardized folder structure facilitating the adding of new model components (including adapter files for the CICE sea-ice model). At present, the INMIO code consists of the hydrodynamical solver, ~~while service work —~~ atmospheric boundary layer bulk formulae, the built-in thermodynamic ice model of (Schrum and Backhaus, 1999) (turns off in case of coupling with the CICE model), and online data processing routines (e.g., averaging). The intramodel communications (halo exchanges on tripolar and latitude-longitude grids) and work with the file system are delegated to the CMF utilities module. For ~~experiments with CORE (Griffies et al., 2009) forcing~~prescribed experiments with the CORE (Griffies et al., 2012) forcing, two data models (reading CORE data files) are also registered as separate atmosphere and land ~~file components~~components, and the CMF coupler provides interpolation of their fields onto ~~ocean high-resolution~~ the ocean grid.

Scalability of ~~INMIO WOM driven by~~ the INMIO WOM of 0.1° resolution driven by the CMF2.0 is shown ~~in Figure~~ on Fig. 9. Maximum number of BlueGene/Q cores ~~is equal to 32400. Parallel efficiency of the model for the amount of resources up to~~ utilized is 32400 (32000 for the ocean component and 400 for the coupler). The parallel efficiency of this configuration in relation to the configuration of 8100 cores (8000 for the ocean and 100 for the coupler) is 78 %. Obviously, ~~smaller numbers of cores provide better values, but we are more interested in scalability of the program on perspective sizes of computational resources. Assuming~~ at high core counts the parallel efficiency curve experiences "flattering". But assuming that the time step of the model is 5 min., the result of the experiment ~~lead to 5~~ leads, e.g., to quite satisfactory five simulated years per wall-clock day (SYPD) rate achieved on 20000 cores of the BlueGene/Q supercomputer.

5.2 Coupled Global atmosphere - ocean model

The second application of the framework was the numerical experiment with ~~coupled INMIO WOM (Ibrayev et al., 2012) and SL-AV Global atmosphere model (Tolstykh et al., 2017) . The SL-AV atmosphere~~ the global coupled INMIO ocean (Ibrayev et al., 2012) an SLAV atmosphere (Tolstykh et al., 2017) models. The SLAV model with horizontal resolution $0.9° \times 0.72°$ and 28 vertical levels~~and~~ , and the INMIO WOM with resolution ~~0.5°~~ 0.25° and 49 vertical levels were coupled into ~~the~~ a single program using the CMF2.0 system. ~~Short-wave~~ The short- and long-wave radiation in the ~~SL-AV model are~~ SLAV model were computed with the time-step of 1 hour. ~~Time evolution of the sea-ice surface temperature is described in the same way as in prescribed ocean experiments.~~ The restriction of spatio-temporal resolution was implied by available computer resources and not by restrictions of the CMF.

Prognostic coupled model calculations were carried out with a time step of 6 min. for the ~~ocean model~~ oceanic component and 3.6 min. for the ~~atmosphere~~atmospheric one. The initial state of the ocean was ~~a control point obtained by~~ obtained by a spin-up of the standalone ocean model ~~. Atmosphere started with~~ driven by the ERA-Interim atmospheric forcing. The atmosphere started from the objective analysis of the ~~Hydrometeorological Center~~ Hydrometcenter of Russia. ~~In coupled regime every~~ Every 72 min.~~9 2-D arrays~~ , nine 2D-arrays were transferred from the atmosphere to the ocean (components of wind stress, ~~shortwave and long wave~~ short- and long-wave radiation, fluxes of sensible and latent heat, precipitation, evaporation, air temperature at 2 m)~~, each~~ . Conversely, every 144 min. ~~3 2-D arrays~~ three 2D-arrays were transferred from the ocean to the atmosphere (~~surface~~ upper gridbox temperature, temperature and concentration of ~~ice and the temperature of the upper ocean gridbox). Ice model was built into the ocean model, land model – into the~~ sea ice). The sea ice was simulated by the INMIO built-in ice thermodynamics model, while the land processes were incorporated into the SLAV atmosphere model. ~~Coupled~~ The coupled model works stably and along with ~~seasonal~~ intraannual distribution characteristics of monthly data fields reproduces enough thin elements of atmospheric and oceanic circulation.

~~SYPD of the coupled model on the MVS-10p supercomputer is~~ The model throughput on the MVS-10P supercomputer was equal to 0.75 ~~for~~ SYPD for the configuration ocean (1152 cores) ~~–~~ – atmosphere (288 cores) ~~–~~ – coupler (16 cores). At ~~the~~ that moment, the ~~maximum computational resources available for atmosphere model is~~ maximal communicator size available for the atmosphere model was limited due to the one-dimensional latitudinal ~~model~~ grid decomposition.

5.3 Data assimilation using DAS

As well as any service of the CMF3.0, the data assimilation is performed on separate ~~computing~~ processor cores. This allows to ~~structurize~~ structure the Earth modeling system better, in order to make each software component solve its own problem. At the same time, the model of the ocean does not take part in the data assimilation. Only results of the ocean modeling in the form of ~~vector elements of the ensemble~~ ensemble vector elements are used. On their basis, the covariance matrices are approximated. ~~Data~~ The data from the ocean model is sent to the service (usually once a modeling day) without using ~~a~~ the file system (through the cluster interconnect). ~~More over~~ Moreover, all matrix-vector operations are calculated ~~parallel (shared-memory~~in parallel (on shared memory) using BLAS and LAPACK functions through the Global Arrays (GA) toolkit (Kaurkin et al., 2016a).

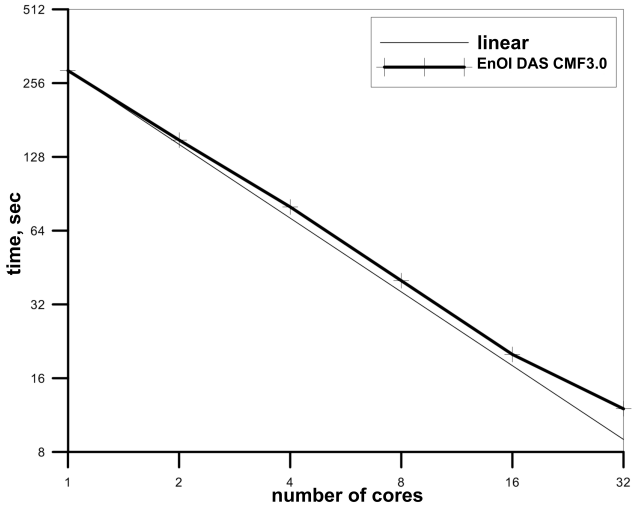

Figure 10. Scalability of the EnOI method in ~~the~~ context of the CMF3.0 DAS service at the assimilation of 10^4 points on the Lomonosov supercomputer (Moscow State University).

Due to the effective implementation of the EnOI method as ~~a~~ the DAS parallel software service~~DAS~~, the solution of the data assimilation problem is scaled almost linearly (Fig. 10). So, ~~the~~ assimilation of 10^4 ~~observation~~ observational points of satellite data on ~~the~~ 16 processor cores takes about 20 seconds instead of 5 minutes on a single core, which would be comparable to the time spent on daily ocean model forecast for 200 cores.

6 Summary and conclusions

We ~~have presented~~ present an original modeling framework ~~CMF3.0~~ CMF developed as our ~~initial~~ first step to high resolution modeling. The key part of it, coupler, in the initial version CMF2.0 has a sufficiently small code size for such programs (about 5000 lines of code ~~with~~ including unit tests) and is able to manage the main parallel problems of the coupled modeling ~~-~~ synchronization, regridding and I/O. The coupled model follows ~~a~~ the single executable design with the main program independent of components' code, and the coupler dealing with all service operations. ~~New versionof~~ The new version, CMF3.0~~utilizes~~, utilizes the SOA-design, which allows one to divide the coupler responsibilities into small separate services and ~~easy~~ easily plug and unplug them~~.~~, thus providing a further generalization to the coupling interface. The PGAS messaging greatly simplifies ~~all low level~~ implementation of all model low-level interprocess communications.

Tests for CMF2.0 parallel mapping efficiency were carried out on four modern supercomputer architectures. ~~Tests show a near~~ They show a nearly linear strong scalability of the overall communication system and the regridding procedure. Satisfactory speed results could be achieved already on 20-40 coupler cores even dealing with grids of high resolution (0.1° for the ocean and 0.225° for the atmosphere). I/O tests proved the ability of the coupler ~~delegate~~ asynchronous I/O scheme to handle ~~with huge amounts~~ huge amounts of data. As expected, ~~new CPL~~ the new CMF 3.0 version has lower absolute performance,

but greatly simplifies code and preserves linear trend of scalability and suitable timing (2-3 seconds per modeling day on 20-50 coupler cores) for high-resolution modeling.

~~Originally designed for WOM support, CMF was used for overall ocean physics development and for long-term modeling of 0.1INMIO WOM. Also first middle term forecasts of coupled 0.25ocean - 0.225atmosphere model became possible due to developed framework.~~

~~We think that conducted experiments cover introduction phase of our~~ The parallel data output speed of CMF3.0 is about a half of that for CMF2.0, but still it is quite satisfactory for contemporary high-resolution ~~modeling plans and~~ climate experiments due to overlapping of computation and writing to file system. Within the framework of the CMF3.0 ~~is ready to further evolution and establishing closer collaboration with community projects. Our future work will cover development of DAS services for operational model forecast and integrating some community instruments (like EMSF or MCT) for support of unstructured grids in perspective models (Volodin et al., 2010)~~architecture it was possible to effectively implement nesting and data assimilation technologies. This functional is not yet common for other coupling platforms.

Originally designed for the INMIO World ocean model support, the CMF has developed into a flexible and extensible instrument providing means for high-resolution resource-demanding simulations in regional/global, stand-alone/coupled, and forecast/climate problems.

Code availability. The code of the CMF3.0 and CMF2.0 (distributed under GPLv2 licence) is available on http://model.ocean.ru (after registration).

Appendix A: Supercomputer configurations used

The MVS-100k and ~~MVS-10p are parts of~~ MVS-10P systems are installed at the Joint Supercomputer Center of the Russian Academy of Sciences (~~jscc.ru).~~ www.jscc.ru). The MVS-100k consists of 1460 modules (11680 processor cores). ~~Basic~~ The basic computing module is an HP Proliant server, containing two quad-core Intel Xeon ~~,~~ processors running at 3 GHz on 8 GB RAM~~memory~~. Computational modules are interconnected with Infiniband DDR. The ~~computer MVS-10p~~ MVS-10P system includes 207 nodes. Each node incorporates 2 ~~processors~~ Intel Xeon E5-2690 processors (16 cores on 2.90 GHz), 64 GB of RAM, and two Intel Xeon Phi ~~coprocessor~~ 7110H ~~. Compute~~ coprocessors. Computing nodes are combined into an FDR Infiniband network.

Supercomputer BlueGene/P is located ~~on the faculty~~ at the Faculty of Computational Mathematics and Cybernetics, Moscow State University, and consists of 2048 ~~compute~~ computing nodes. Each node ~~is a 4 core~~ has four PowerPC 450 ~~(2 Gb RAM,~~ cores (850 MHz) and 2 GB of RAM. Nodes are networked with the 3D-torus topology (5.1 GB/s, DMA).

~~Computer~~ Supercomputer BlueGene/Q is located ~~in~~ at the IBM Thomas J. Watson Research Center and consists of several racks. Every ~~2~~ two racks have 2048 computational nodes, each with 16 cores. The core is a PowerPC A2 (16 GB RAM, 1.6 GHz). Nodes are networked with the 5D-torus topology (40 GB/s, DMA).

Supercomputer Lomonosov is located ~~in~~ at the Lomonosov Moscow State University and consists of more than 50000 cores. We have used ~~partition with 8 core~~ the partition with eight-core nodes (2 x Intel Xeon 5570 Nehalem, 12 GB, 2.9 Ghz). Computational modules are interconnected with the Infiniband QDR.

Acknowledgements. ~~This work~~ The research of Sections 1–3, 5.1 and 5.2 was was supported by the Russian Science Foundation (project no. 14-37-00053) and performed at the Hydrometeorological Research Centre of the Russian Federation. The research of Sections 4 and 5.3 was supported by the Russian Science Foundation (project no. 17-77-30001) and performed at the Federal State Budget Scientific Institution "Marine Hydrophysical Institute of RAS".

References

Balaji, V.: The Flexible Modeling System, pp. 33–41, Springer Berlin Heidelberg, Berlin, Heidelberg, https://doi.org/10.1007/978-3-642-23360-9_5, 2012.

Balaji, V., Maisonnave, E., Zadeh, N., Lawrence, B. N., Biercamp, J., Fladrich, U., Aloisio, G., Benson, R., Caubel, A., Durachta, J., Foujols, M.-A., Lister, G., Mocavero, S., Underwood, S., and Wright, G.: CPMIP: measurements of real computational performance of Earth system models in CMIP6, Geoscientific Model Development, 10, 19–34, https://doi.org/10.5194/gmd-10-19-2017, https://www.geosci-model-dev.net/10/19/2017/, 2017.

CESM: CESM1.2 User guide., NCAR, http://www.cesm.ucar.edu/models/cesm1.2/cesm/doc/usersguide/ug.pdf, 2013.

Collins, N., Theurich, G., DeLuca, C., Suarez, M., Trayanov, A., Balaji, V., Li, P., Yang, W., Hill, C., and da Silva, A.: Design and Implementation of Components in the Earth System Modeling Framework, The International Journal of High Performance Computing Applications, 19, 341–350, https://doi.org/10.1177/1094342005056120, 2005.

Craig, A., Valcke, S., and Coquart, L.: Development and performance of a new version of the OASIS coupler, OASIS3-MCT_3.0, Geoscientific Model Development, 10, 3297–3308, https://doi.org/10.5194/gmd-10-3297-2017, 2017.

Craig, A. P., Jacob, R., Kauffman, B., Bettge, T., Larson, J., Ong, E., Ding, C., and He, Y.: CPL6: The New Extensible, High Performance Parallel Coupler for the Community Climate System Model, The International Journal of High Performance Computing Applications, 19, 309–327, https://doi.org/10.1177/1094342005056117, 2005.

Craig, A. P., Vertenstein, M., and Jacob, R.: A new flexible coupler for earth system modeling developed for CCSM4 and CESM1, The International Journal of High Performance Computing Applications, 26, 31–42, https://doi.org/10.1177/1094342011428141, 2012.

Dennis, J. M., Edwards, J., Loy, R., Jacob, R., Mirin, A. A., Craig, A. P., and Vertenstein, M.: An application-level parallel I/O library for Earth system models, The International Journal of High Performance Computing Applications, 26, 43–53, https://doi.org/10.1177/1094342011428143, 2012a.

Dennis, J. M., Vertenstein, M., Worley, P. H., Mirin, A. A., Craig, A. P., Jacob, R., and Mickelson, S.: Computational performance of ultra-high-resolution capability in the Community Earth System Model, The International Journal of High Performance Computing Applications, 26, 5–16, https://doi.org/10.1177/1094342012436965, 2012b.

Dussin, R., Barnier, B., Brodeau, L., and Molines, J.: The making of the Drakkar forcing set DFS5, Tech. rep., DRAKKAR/MyOcean Report 01-04-16, LGGE, Grenoble, France, 2016.

Fadeev, R. Y., Ushakov, K. V., Tolstykh, M. A., Ibrayev, R. A., and Kalmykov, V. V.: Coupled atmosphere–ocean model SLAV–INMIO: implementation and first results, Russian Journal of Numerical Analysis and Mathematical Modelling, 31, 329–337, https://doi.org/10.1515/rnam-2016-0031, 2016.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

Griffies, S., Biastoch, A., Böning, C., Bryan, F., Danabasoglu, G., Chassignet, E., England, M., Gerdes, R., Haak, H., Hallberg, R., Hazeleger, W., Jungclaus, J., Large, W., Madec, G., Pirani, A., Samuels, B., Scheinert, M., Gupta, A., Severijns, C., Simmons, H., Treguier, A., Winton, M., Yeager, S., and Yin, J.: Coordinated Ocean-ice Reference Experiments (COREs), Ocean Modelling, 26, 1–46, https://doi.org/10.1016/j.ocemod.2008.08.007, 2009.

Griffies, S. M., Winton, M., Samuels, B., Danabasoglu, G., Yeager, S., Marsland, S., Drange., H., , and Bentsen, M.: Datasets and protocol for the CLIVAR WGOMD Coordinated Ocean-sea ice Reference Experiments (COREs), WCRP Report No. 21/2012, Tech. rep., 2012.

Ibrayev, R. A.: Model of enclosed and semi-enclosed sea hydrodynamics, Russian Journal of Numerical Analysis and Mathematical Modelling, 16, 291–304, https://doi.org/10.1515/rnam-2001-0404, 2001.

Ibrayev, R. A., Khabeev, R. N., and Ushakov, K. V.: Eddy-resolving 1/10° model of the World Ocean, Izvestiya, Atmospheric and Oceanic Physics, 48, 37–46, https://doi.org/10.1134/S0001433812010045, 2012.

Jacob, R., Larson, J., and Ong, E.: M × N Communication and Parallel Interpolation in Community Climate System Model Version 3 Using the Model Coupling Toolkit, The International Journal of High Performance Computing Applications, 19, 293–307, https://doi.org/10.1177/1094342005056116, 2005.

Jones, P. W.: First- and Second-Order Conservative Remapping Schemes for Grids in Spherical Coordinates, Monthly Weather Review, 127, 2204–2210, https://doi.org/10.1175/1520-0493(1999)127<2204:FASOCR>2.0.CO;2, 1999.

Kalmykov, V. V. and Ibrayev, R. A.: A framework for the ocean-ice-atmosphere-land coupled modeling on massively-parallel architectures, Vychisl. Metody Programm., 14, 88–95, http://mi.mathnet.ru/vmp156, (in Russian), 2013.

Kaurkin, M., Ibrayev, R., and Koromyslov, A.: EnOI-Based Data Assimilation Technology for Satellite Observations and ARGO Float Measurements in a High Resolution Global Ocean Model Using the CMF Platform, vol. 687 of *Communications in Computer and Information Science*, pp. 57–66, Springer International Publishing, Cham, https://doi.org/10.1007/978-3-319-55669-7_5, 2016a.

Kaurkin, M. N., Ibrayev, R. A., and Belyaev, K. P.: ARGO data assimilation into the ocean dynamics model with high spatial resolution using Ensemble Optimal Interpolation (EnOI), Oceanology, 56, 774–781, https://doi.org/10.1134/S0001437016060059, 2016b.

Koromyslov, A., Ibrayev, R., and Kaurkin, M.: The technology of nesting a regional ocean model into a Global one using a computational platform for massively parallel computers CMF, vol. 793 of *Communications in Computer and Information Science*, pp. 241–250, Springer International Publishing, Cham, https://doi.org/10.1007/978-3-319-71255-0_19, 2017.

Larson, J., Jacob, R., and Ong, E.: The Model Coupling Toolkit: A New Fortran90 Toolkit for Building Multiphysics Parallel Coupled Models, The International Journal of High Performance Computing Applications, 19, 277–292, https://doi.org/10.1177/1094342005056115, 2005.

Marzocchi, A., Hirschi, J. J.-M., Holliday, N. P., Cunningham, S. A., Blaker, A. T., and Coward, A. C.: The North Atlantic subpolar circulation in an eddy-resolving global ocean model, Journal of Marine Systems, 142, 126 – 143, https://doi.org/https://doi.org/10.1016/j.jmarsys.2014.10.007, http://www.sciencedirect.com/science/article/pii/S0924796314002437, 2015.

Nieplocha, J., Palmer, B., Tipparaju, V., Krishnan, M., Trease, H., and Aprà, E.: Advances, Applications and Performance of the Global Arrays Shared Memory Programming Toolkit, The International Journal of High Performance Computing Applications, 20, 203–231, https://doi.org/10.1177/1094342006064503, 2006.

Redler, R., Valcke, S., and Ritzdorf, H.: OASIS4 – a coupling software for next generation earth system modelling, Geoscientific Model Development, 3, 87–104, https://doi.org/10.5194/gmd-3-87-2010, 2010.

Schrum, C. and Backhaus, J.: Sensitivity of atmosphere–ocean heat exchange and heat content in the North Sea and the Baltic Sea, Tellus A, 51, 526–549, https://doi.org/10.1034/j.1600-0870.1992.00006.x, https://onlinelibrary.wiley.com/doi/abs/10.1034/j.1600-0870.1992.00006.x, 1999.

Shukla, J.: World Modelling Summit for Climate Prediction. WCRP-131, WMO/TD-No.1468., Uk, European Centre for Medium-Range Weather Forecasts, 2008.

Theurich, G., DeLuca, C., Campbell, T., Liu, F., Saint, K., Vertenstein, M., Chen, J., Oehmke, R., Doyle, J., Whitcomb, T., Wallcraft, A., Iredell, M., Black, T., Silva, A. M. D., Clune, T., Ferraro, R., Li, P., Kelley, M., Aleinov, I., Balaji, V., Zadeh, N., Jacob, R., Kirtman, B., Giraldo, F., McCarren, D., Sandgathe, S., Peckham, S., and IV, R. D.: The Earth System Prediction Suite: Toward a Coordinated U.S.

Modeling Capability, Bulletin of the American Meteorological Society, 97, 1229–1247, https://doi.org/10.1175/BAMS-D-14-00164.1, 2016.

Tolstykh, M., Shashkin, V., Fadeev, R., and Goyman, G.: Vorticity-divergence semi-Lagrangian global atmospheric model SL-AV20: dynamical core, Geoscientific Model Development, 10, 1961–1983, https://doi.org/10.5194/gmd-10-1961-2017, https://www.geosci-model-dev.net/10/1961/2017/, 2017.

Turner, A. K. and Hunke, E. C.: Impacts of a mushy-layer thermodynamic approach in global sea-ice simulations using the CICE sea-ice model, Journal of Geophysical Research: Oceans, 120, 1253–1275, https://doi.org/10.1002/2014JC010358, http://dx.doi.org/10.1002/2014JC010358, 2015.

Ushakov, K. V. and Ibrayev, R. A.: Simulation of the global ocean thermohaline circulation with an eddy-resolving INMIO model configuration, IOP Conference Series: Earth and Environmental Science, 96, 012 007, https://doi.org/10.1088/1755-1315/96/1/012007, 2017.

Ushakov, K. V. and Ibrayev, R. A.: Assessment of mean world ocean meridional heat transport characteristics by a high-resolution model, Russ. J. Earth Sci., 18, ES1004, https://doi.org/10.2205/2018ES000616, 2018.

Ushakov, K. V., Grankina, T. B., Ibrayev, R. A., and Gromov, I. V.: Simulation of Arctic and North Atlantic ocean water and ice seasonal characteristics by the INMIO-CICE coupled model, IOP Conference Series: Earth and Environmental Science, 48, 012 013, https://doi.org/10.1088/1755-1315/48/1/012013, 2016.

Valcke, S.: The OASIS3 coupler: a European climate modelling community software, Geoscientific Model Development, 6, 373–388, https://doi.org/10.5194/gmd-6-373-2013, 2013.

Valcke, S., Balaji, V., Craig, A., DeLuca, C., Dunlap, R., Ford, R. W., Jacob, R., Larson, J., O'Kuinghttons, R., Riley, G. D., and Vertenstein, M.: Coupling technologies for Earth System Modelling, Geoscientific Model Development, 5, 1589–1596, https://doi.org/10.5194/gmd-5-1589-2012, 2012.

Volodin, E. M., Dianskii, N. A., and Gusev, A. V.: Simulating present-day climate with the INMCM4.0 coupled model of the atmospheric and oceanic general circulations, Izvestiya, Atmospheric and Oceanic Physics, 46, 414–431, https://doi.org/10.1134/S000143381004002X, 2010.