Multithreaded or thread safe OASIS version including performance optimizations to adapt to many-core architectures
Deliverable D2.3

CECI, Université de Toulouse, CNRS, CERFACS
Technical Report TR-CMGC-18-74

This project has received funding from the European Union's
Horizon 2020 Research and Innovation Programme
under Grant Agreement No 675191

About this document

Work package in charge: WP2: Scalability
Actual delivery date for this deliverable: 28 March 2018
Dissemination level: PU (public)
Lead author: Sophie Valcke, CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, Toulouse, France),

Other contributing authors:
Laure Coquart, Anthony Craig, Gabriel Jonville, Eric Maisonnave, Andrea Piacentini

Internal reviewers:
Uwe Fladrich (SMHI), David Guibert (Atos), Erwan Raffin (Atos)

Contacts: esiwace@dkrz.de
Visit us on: www.esiwace.eu
Follow us on Twitter: @esiwace

Disclaimer: This material reflects only the authors view and the Commission is not responsible for any use that may be made of the information it contains.

Index

1. Abstract /publishable summary

The developments realised in OASIS3-MCT to improve its parallel efficiency are detailed. These will be available in the next release OASIS3-MCT_4.0 planned for spring 2018. The most important improvements concern the communication scheme and the hybrid MPI+OpenMP parallelisation of the Spherical Coordinate Remapping and Interpolation Package (SCRIP) library. The new communication method, which can now use the mapping weights to define the intermediate mapping decomposition, takes longer to initialize but offers significant gain at run time, especially for high-resolution cases running on a high number of tasks. The parallelisation introduced in the SCRIP library for the mapping weight calculation allows a reduction in the weight calculation time of 2 to 3 orders of magnitude for high-resolution grids. Also, significant gains are obtained in the initialisation phase by updating the MCT library from version 2.8 to 2.10.beta1 and additional debugging. New methods introduced in the CONSERV post-processing operation ensuring the global conservation of the coupling fields lower the calculation costs by one order of magnitude while still ensuring good level of reproducibility. Finally, additional results obtained with IS-ENES2 coupling technology benchmarks show that OASIS3-MCT performs as well as, and even better at very high number of cores, than other coupling technologies and that its behaviour on Marconi KNL is fully satisfactory.

2. Conclusion & Results

The different developments realised in the last 24 months ensure that the parallel performance of the next official release of the coupler, OASIS3-MCT_4.0, will be greatly improved.

First, a new communication method, using the remapping weights to define the intermediate mapping decomposition, offers a significant gain at run time, especially for high-resolution cases running on a high number of tasks, thanks to reduced communication. However, as expected, the new method takes longer to initialize, partly due to the fact that the mapping weight file has to be read twice but also due to the extra cost for the initialization of the different MCT routers (see section 4.1.4 for details). That initialization cost is largely mitigated by an upgrade to MCT 2.10.beta1 which reduces the penalty to few seconds. Generally, it should be worth the extra initial cost to speed up the run time. Of course, the balance between the increase of initial costs and the gain obtained at runtime has to be evaluated for each real coupled system because it strongly depends on the specific coupled configuration and on the length of the run.

Second the hybrid MPI+OpenMP parallelisation of the SCRIP library (previously fully sequential) leads to great improvement in the calculation of the remapping weights. The results obtained here show a reduction in the weight calculation time of 2 to 3 orders of magnitude with the new parallel SCRIP library for high-resolution grids. This important improvement let us envisage dynamical coupling, implying runtime weight computation, with OASIS3-MCT.

Third, the new methods introduced in the global CONSERV operation reduce its calculation costs by one order of magnitude while still ensuring an appropriate level of reproducibility. This removes the bottleneck foreseen at high resolution with this important, and in few cases still unavoidable, global operation.

Finally, additional results obtained with IS-ENES2 coupling technology benchmarks show that OASIS3-MCT performs as well as, and even better at very high number of cores, than other coupling technologies and that its behaviour on Marconi KNL is fully satisfactory, at least for the case tested.

3. Project objectives

Given the key role that the OASIS3-MCT coupler plays in the efficient execution of numerical simulations based on Earth System Models (ESMs), this deliverable contributes directly and indirectly to the achievement of a vast majority of the macro-objectives and specific goals indicated in section 1.1 of the Description of the Action:

Macro-objectives	**Contribution of this deliverable?**
Improve the efficiency and productivity of numerical weather and climate simulation on high-performance computing platforms	Yes
Support the end-to-end workflow of global Earth system modelling for weather and climate simulation in high performance computing environments	Yes
The European weather and climate science community will drive the governance structure that defines the services to be provided by ESiWACE	Yes
Foster the interaction between industry and the weather and climate community on the exploitation of high-end computing systems, application codes and services.	No
Increase competitiveness and growth of the European HPC industry	No

Specific goals in the workplan	**Contribution of this deliverable?**
Provide **services** to the user community that will impact beyond the lifetime of the project.	Yes
Improve **scalability** and shorten the time-to-solution for climate and operational weather forecasts at increased resolution and complexity to be run on future extreme-scale HPC systems.	Yes
Foster **usability** of the available tools, software, computing and data handling infrastructures.	Yes
Pursue **exploitability** of climate and weather model results.	No
Establish governance of common software management to avoid unnecessary and redundant development and to deliver the best available solutions to the user community.	Yes
Provide **open access** to research results and **open source** software at international level.	Yes
Exploit **synergies** with other relevant activities and projects and also with the global weather and climate community	Yes

4. Detailed report on the deliverable

We describe here the developments done in the OASIS3-MCT coupler during the last 24 months to improve its parallel efficiency. The most important improvements concern the communication scheme, which can now use the mapping weights to define the intermediate mapping decomposition, and the hybrid MPI+OpenMP parallelisation introduced in the SCRIP library for the mapping weight calculation. These features are described in section 4.1 and 4.2 respectively. Efforts were also spent to improve the coupling initialisation phase with the update of the MCT library from version 2.8 to 2.10.beta1. The gains obtained and additional debugging are presented in section 4.3. Section 4.4 then details the optimisation and the new options introduced in the global CONSERV operation. Additional results obtained with IS-ENES2 coupling technology benchmarks, either testing new options or running on Marconi KNL, are described in section 4.5. For completeness, section 4.6 presents additional minor developments achieved during the period.

Along the text, we often refer to the "VHR", "VHR_oppdec" or HR_tutorial test cases. These are described in Appendix A, while Appendix B illustrates the new mapping decomposition options discussed in section 4.1 and Appendix C presents the survey sent to OASIS users on coupling multi-threaded codes (see section 4.2.1).

4.1. Optimisation of the communication by using the mapping weights to define the intermediate mapping decomposition

4.1.1. Overview

In OASIS3-MCT, the remapping (also known as regridding or interpolation) and exchange of the coupling fields are done following specific steps. The remapping can take place before the coupling exchange on the source component tasks *($MAPLOC = src* in the *namcouple* configuration file) or after the coupling exchange on the target component tasks *($MAPLOC = dst*, implying the steps between brackets in what follows). To perform the remapping, OASIS3-MCT creates a "mapping decomposition" of the target grid on the source tasks [the source grid on the destination tasks]. In OASIS3-MCT_3.0, this mapping decomposition was always done assigning each target grid point to a source task [each source grid point to a target task] in a trivial 1-dimensionnal way *($NMAPDEC = decomp_1d*). In OASIS3-MCT_4.0 a more optimal mapping decomposition will be available based on the mapping weights *($NMAPDEC = decomp_wghtfile*) such that a target grid point is associated with the source task which holds the source grid points needed for the calculation of its interpolated value [such that a source grid point is associated with a target task which contains the target grid points which will use it for the calculation of their interpolated value]. For a remapping performed on the source tasks, a sparse matrix multiplication using the mapping weights will be performed first to transform the source coupling field from its source decomposition to the mapping decomposition of the target grid on the source tasks. Then the coupling field is rearranged from that mapping decomposition on the source tasks to the target decomposition on the target tasks. [For remapping performed on the target tasks, the coupling data is first rearranged from the source grid decomposition on the source tasks to the mapping decomposition of the source grid on the target tasks and then the sparse matrix multiplication is performed on the target tasks to transform the coupling field from the mapping decomposition to the target decomposition].

An illustration of the different communication steps for *decomp*_1d and *decomp_wghtfile* methods for a very simple case is shown in Appendix B. We can infer from this illustration that, even if it might be somewhat more costly in the initialisation phase, *decomp_wghtfile* leads to reduced coupling run time as the rearrangement between the source decomposition and the mapping decomposition [between the mapping decomposition and the target decomposition] associated with the sparse matrix multiplication is much simpler. Performance tests comparing *decomp_1d* and *decomp_wghtfile* were carried out for the HR_tutorial and VHR_oppdec cases and a study of the impact on the initialisation time was also realised. These are presented in the next subsections.

4.1.2. Results for the HR_tutorial case

We first present the results for the HR_tutorial test case (see Appendix A), which is a realistic case and therefore represents the gain that one can hope to get in a realistic coupled model. The tests were done with OASIS3-MCT SVN branch tc17b r2069 on Bullx beaufix at Météo-France using the Intel 16.1.150 compiler and the Intelmpi 5.1.2.150 MPI library. Different runs were performed with the number of cores/tasks per component between 1 and 10000. Results are shown at Fig. 1.

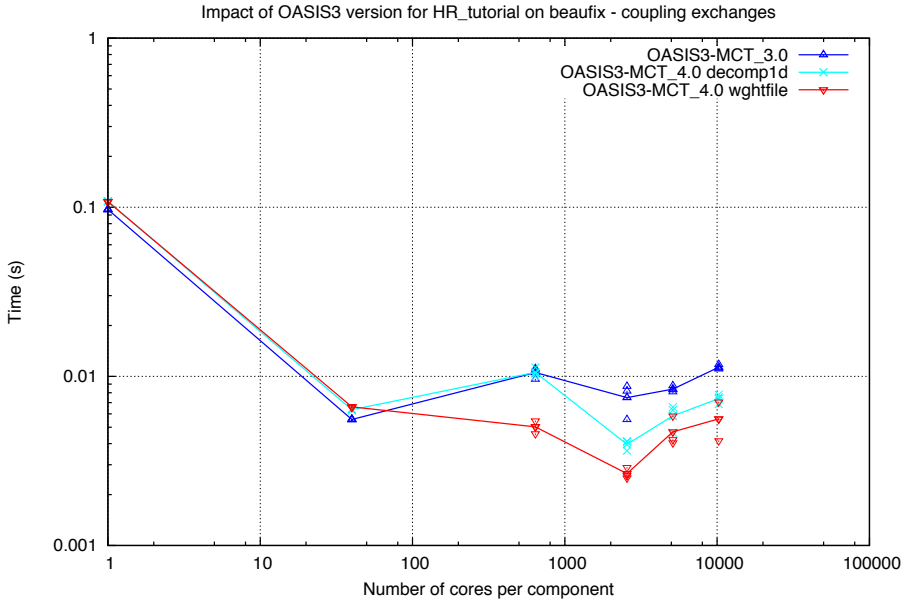


Figure 1: Time for a ping-pong exchange with respect to the number of tasks/cores used for each component for the HR_tutorial case, for the previous OASIS3-MCT_3.0 version (dark blue) and for the branch tc17b r2069 activating *decomp_1d* (light blue) or *decomp_wghtfile* (red) methods that will be available in the next OASIS3-MCT_4.0 release.

We see that the *decomp_wghtfile* method offers an important gain, especially for a number of cores of O(1000) and higher. For 10240 tasks/cores per component, the ping-pong time with

decomp_wghtfile (0.0055) is 24% better than *with decomp_1d* (0.0073), which is already 35% better than the ping-pong time obtained with the previous OASIS3-MCT_3.0 official release (0.0112) (branch OASIS3-MCT_3.0_branch r1962).
The impact on the initialization phase is shown on Fig. 7. There is some variation with respect to the number of cores and it is hard to conclude in this case if the initialization time is generally higher for one method or for the other.

4.1.3. Results for the VHR_oppdec case

We also analysed the performance differences between *decomp_1d* and *decomp_wghtfile* for the more extreme VHR_oppdec on Bullx beaufix in the same conditions than for the HR_tutorial case. Results are shown on Fig. 2.

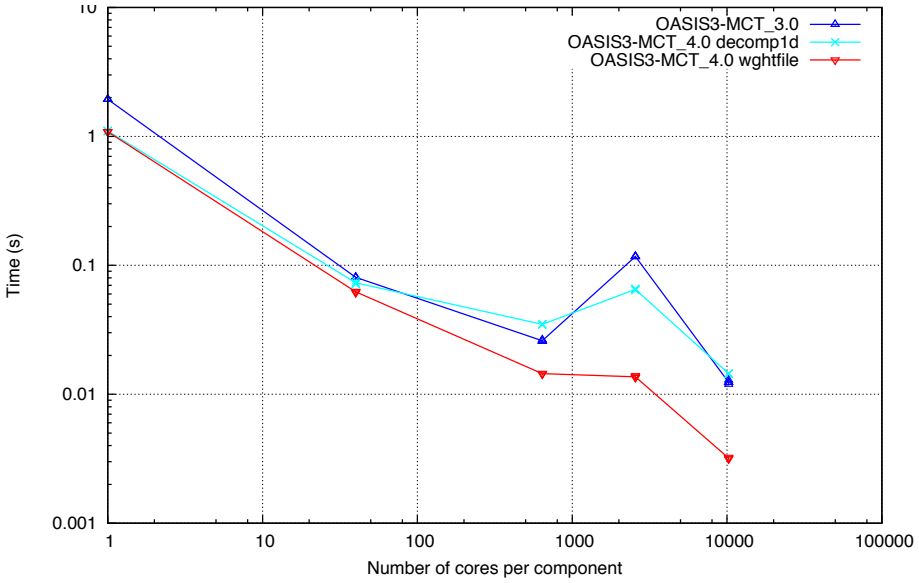


Figure 2: Time for a ping-pong exchange with respect to the number of tasks/cores used for each component for the VHR_oppdec case, for the previous OASIS3-MCT_3.0 version (dark blue) and for the branch tc17b r2069 activating *decomp_1d* (light blue) or *decomp_wghtfile* (red) methods that will be available in the next OASIS3-MCT_4.0 release.

The gain offered by the *decomp_wghtfile* method is, as expected, even more striking in this case with a reduction of 78% reduction in the ping-pong exchange time at 10240 cores for *decomp_wghtfile* (0.0032) with respect to *decomp_1d* (0.0144).

The impact on the initialization phase is shown on Fig. 6. At high number of cores, the initialization time is sensibly higher for *decomp_wghtfile* than for *decomp_1d* but the difference seems to gets smaller at very high number of cores.

4.1.4. Impact on the initialisation time for the VHR case

A study on the impact of *decomp_wghtfile* compared to *decomp_1d* on the initialisation time was also realised for the VHR test case. These results are detailed in Craig & Valcke 2018 (even if this report focuses on the difference in the initialisation time using either MCT 2.8 or MCT 2.10.beta1, see section 4.3.1). The tests were done on Météo-France Bullx beaufix with 1600 and 3600 tasks per component, in the same conditions as above (Intel 16.1.150 compiler and Intelmpi 5.1.2.150 MPI library). Different timing measures are presented in this report, with the following observations with respect to the *decomp_1d/decomp_wghtfile* comparison:

- The time to initialize the mapping decomposition of the target grid on the source tasks is ~40 times on 1600 tasks and ~10 times on 3600 tasks more expensive for *decomp_wghtfile* than for *decomp_1d* (even if it is still only about 1 second on 3600 tasks). Because of the extra complexity of the mapping decomposition, this is likely caused by the MPI cost of sending many shorter messages and handling multiple segments per task compared to the *decomp_1d* case.
- There is not a clear difference in the cost to read the mapping weights file for *decomp_1d* versus *decomp_wghtfile* but, overall, *decomp_wghtfile* is always more expensive because the mapping file has to be read twice (once before the mapping decomposition taking into account the source decomposition and once taking into account the mapping decomposition after it is defined) instead of only once for *decomp_1d*. But overall it always takes less than 5 sec for the two readings.
- With MCT 2.8, the cost to compute the router between the source decomposition and the mapping decomposition is ~50 time greater for the *decomp_wghtfile* versus *decomp_1d* for 3600 tasks and ~80 time greater for 1600 tasks. But with MCT 2.10.beta1, this is reduced to ~8 times (0.6-0.8 sec) and ~10 times (0.5 sec) respectively. Indeed, *decomp_wghtfile* results in a much more complicated mapping decomposition with many segments and the number of segments plays an important role in the router initialization cost.
- The cost associated with computing a router between the mapping decomposition of the target grid on the source tasks and the target decomposition of the target grid on the target tasks is higher for *decomp_wghtfile* compared to *decomp_1d*. With MCT 2.8, the cost is ~40 times greater for the *decomp_wghtfile* for 3600 tasks and ~60 times greater for 1600 tasks. But with MCT 2.10.beta1, this is reduced to ~4 times for 3600 tasks and ~3 times for 1600 tasks (considering here the runs with *orig* method for reading the weights), being for all cases less than 8 seconds.
- The runtime sparse matrix multiply cost is between 20% and 40% lower for 1600 tasks and O(10) times lower for 3600 tasks for *decomp_wghtfile* cases compared to *decomp_1d*, because *decomp_wghtfile* minimizes the rearrangement. This translates directly into reduced run loop cost, which is about the same for 1600 tasks but between ~3 times (MCT 2.8) and ~2 times (MCT 2.10.beta1) smaller for 3600 tasks for *decomp_wghtfile* cases compared to *decomp_1d*.

While the initialization cost for *decomp_wghtfile* is higher in this case than *decomp_1d*, it is important to realize that the total initialization time is anyway only few seconds or less, and that cost should be recovered quickly for typical production length coupled simulations.

4.1.5. Conclusions

The general conclusion is that the new *decomp_wghtfile* method, using the remapping weights to define the mapping decomposition, indeed offers a significant gain, especially for high-resolution cases running on a high number of tasks, thanks to reduced communication linked to a much simpler rearrangement in the sparse matrix multiply at run time (even if the computation of the corresponding router takes more time at initialization, as noted above). However, the new *decomp_wghtfile* takes longer to initialize, as expected, partly due to the fact that the mapping weight file has to be read twice but is also related to the extra cost for the initialization of the mapping decomposition and of the sparse matrix multiply, and for the initialization of the router between the mapping decomposition and the target decomposition. However, while those increased costs can be relatively high with MCT 2.8, MCT 2.10.beta1 has mitigated the absolute cost to few seconds. So in general, those costs should be small enough so that for production runs, it will be worth spending extra time during initialization (which by definition happens only once at the beginning of the run) to speed up the run time. Of course, the balance between the increased cost of the initialisation and the gain obtained at runtime in the coupling exchanges has to be evaluated for each real coupled system because it strongly depends on the specific coupled configuration (grids, decompositions, number of coupling fields, etc.) and on the length of the run.

4.2. Hybrid MPI+OpenMP parallelisation in OASIS3-MCT

4.2.1. Introduction

Producing a multi-threaded version of OASIS3-MCT can be understood in different ways. Intense discussions and exchanges took place between OASIS3-MCT developers to first define what would be optimal to achieve during ESiWACE given the current user needs and the allocated funding. This discussion is available on-line in the Redmine issue #1223 (https://inle.cerfacs.fr/issues/1223, login required).

The full and global multithreading of the coupler, in which each thread of a multi-threaded model would perform all coupling actions (i.e. definition of its partition of the global field, sending and receiving of its local coupling field array, etc.) was soon discarded, as it would imply in particular multithreading of the MCT library itself which was clearly out-of-scope. Then modifying the API of OASIS3-MCT to ensure that it would be thread safe when called from hybrid MPI+OpenMP models, even without multithreading OASIS3-MCT itself, was considered. But it was then analysed that the modifications required would be very dependent on the layout of the hybrid parallelisation of the model, on the structure of the coupling field and/or on manipulations done on the code arrays to extract the coupling fields in the multi threaded region. So before starting any implementation or prototyping, OASIS3-MCT users were surveyed and asked to share their experience and expectations about coupling multi-threaded codes. A general mail was sent to the OASIS user mailing list but only 3 groups provided feedback on the survey (see Appendix C). For the Max-Planck Institute in Hamburg, the coupling interfacing is handled outside threaded regions and there was no indication that this

could change in the future. For the MetOffice, the coupling interfacing is also dealt with outside threaded regions, and even if there are no real plans to change this, option c) from the survey could be envisaged in the future. IPSL is the only known group calling OASIS API inside an OpenMP threaded region with a strategy corresponding to f) in the survey and they have successfully done the OASIS3-MCT interfacing themselves.

This low number of replies reveals that interfacing OASIS3-MCT into threaded regions of OpenMP models is currently not a high priority for the community and that there is not a clear set of requirements to address. These first reflexions and interactions with the community regarding the needs of multithreading or thread safety in OASIS3-MCT form a good basis for future work in this direction, but it was then decided, for ESiWACE, to concentrate our parallelisation effort on the SCRIP library calculating the remapping weights. This important step, taking place in the initialisation phase of the coupling, was observed to become cost prohibitive for high-resolution models. Furthermore, interfacing with a parallel and efficient library for the remapping weight calculation is an important first step toward a longer-term goal for OASIS3-MCT, i.e. transforming it into a dynamic coupler. Indeed supporting models with evolving grids, which means that the remapping weights have to be recalculated as the model evolves, was discussed and considered important during the last OASIS3-MCT Advisory Board meeting[1].

4.2.2. MPI+OpenMP parallelisation of the SCRIP library

This subsection summarizes the recent developments introduced in OASIS3-MCT SCRIP library to enhance its computing performance both in sequential and hybrid MPI+OpenMP modes. All these developments are available in OASIS3-MCT SVN branch "eahybrid" and are presented in details in the technical report Piacentini et al 2018. Performance improvements, i.e. speedup of the calculations by 2 to 3 orders of magnitude, were obtained for nearest neighbours, bicubic, bilinear and conservative interpolations. Care was taken during the implementation to preserve the remapping results per se and therefore rationalization of several algorithms that were found to be inexact in several cases (in particular, the bin search restriction for some interpolations) was not introduced. Future options to address these shortcomings are however presented and need to be further investigated. In the mean time, OASIS3-MCT_4.0 will be modified to disable faulty combinations of options and the User Guide will be updated to warn users about features that cannot safely be activated, and these are also detailed below.

Rationale

Since its first implementation in OASIS, the SCRIP library performs the calculation of the remapping weights only on the MPI master process of each coupled component model. As OASIS3-MCT-based coupled systems are usually exploited on supercomputers and are, for most of them, parallelised with MPI or even in hybrid MPI+OpenMP mode, we decided to implement a hybrid MPI+OpenMP parallelisation of the SCRIP library. It relies on the MPI parallel layout of the calling model but only enrols one MPI process per node. The number of OpenMP threads per node is set by a dedicated environment variable OASIS_OMP_NUM_THREADS, and for optimum performance, it is recommended to set this variable to the number of cores of the node.

The SCRIP interpolations can be filed in two groups: 1- conservative (1st and 2nd order); and 2- all others: bilinear, bicubic, distance-weighted and Gaussian-weighted nearest-neighbour. Interpolations of the second type mainly follow the same procedure. For each unmasked target

[1] See https://portal.enes.org/oasis/metrics/images-and-documents/20171220_OASIS_Advisory_Board_minutes.pdf

grid point, a distance is calculated with all (or, when a bin restriction is applied, with a subset of) source grid points to determine which ones are the closest and could participate to the weight calculation. It means that N independent calculations (with N = number of target grid points) can be scattered to different nodes of the machine without major communication overhead. On this outer loop, a MPI parallelisation is done on every first core of each node. In addition, to avoid memory duplication of source grid point arrays, OpenMP threads also parallelise the outer loop on target grid points and share the source grid point related variables. After the interpolation weight calculation by the different threads, results are copied in shared variables and gathered on the master process of the model.

The weight calculation procedure is slightly different for conservative interpolations. Mesh contour intersections are calculated for both source and target grid cells. Consequently, the MPI+OpenMP hybrid parallelisation is done on two outer loops (over source and target grid cells). The search of neighbour cells potentially intersected can be restricted using the so-called "bin" technique. In a second step, a complementary nearest-neighbour search can be launched (if the user chooses the FRACNNEI option) for target grid cells for which no intersection with unmasked source grid cell was found, and this step is now also parallelised with OpenMP.

The performance improvement obtained with the hybrid MPI+OpenMP parallelisation of the SCRIP library is presented in the next paragraphs. But first a summary of the preliminary analysis of the library, the optimisations that were introduced in the sequential code, and additional work needed to support the parallelisation will be presented.

Binning strategy and bounding box definition in the SCRIP library

A deep analysis of the SCRIP library was first performed before implementation of the parallelisation. This analysis revealed some flaws of the library, at least as implemented in OASIS. Some usage restrictions and bug fixes are described here.

In order to restrict the search loops over the grid cells in the interpolation weight computation, SCRIP introduces a latitude binning strategy (see details in Appendix 2 of Piacentini et al 2018). Bins are meant to associate index spans in both grid representations to latitude bands. The binning splits the global $[-\pi/2,\pi/2]$ domain in NBINS of equal latitudinal extension, NBINS being prescribed by the user. The bins can therefore be used to target subsets of the grids before performing further matching tests or brute force searches. The association between grid cells and bins is based on their bounding boxes, which is defined as the rectangle in the longitude-latitude space that contains the cell or the bin grid cells. A cell will be associated to a bin if their bounding box intersect.

For the conservative remapping, the bounding box definition is based on the minimum and maximum value of the grid cell corners, which is a robust definition for all grids. But for the other interpolations, cell corners are not required in OASIS and the bounding box is estimated by the relative position of the grid cell centres. First this means that the whole sphere is not covered if the Northern and Southern most grid centres are not located at the poles. Second, the algorithm used relies on the implicit hypothesis that the grid is Cartesian and stored with longitude increasing first. This introduces an error for other grid types (except for the Gaussian-reduced grid for the bilinear and bicubic special cases, see below) that can lead to wrong or incomplete binning. Indeed, for grids with latitude increasing first, every bin would be associated with the entire index range (minus a few elements), with no effect on the optimization. In addition for nearest-neighbour, if the destination grid cell centre belongs to bin n, the search uses bins n-1, n, n+1 to select the range (on the source grid) for the neighbour computation. For a large number

of bins (leading to bins with small latitudinal extension) or when searching for a large number of neighbours, use of only three bins may not be enough, and the search could fail[2].

For bilinear and bicubic interpolations from Gaussian-reduced grids, a specific algorithm is implemented. In that case, the number of bins used to split the grid coincide with the number of latitude circles in the grid (minus one, to be precise), independently of the number of bins indicated by the user. The bin definition algorithm works fine but only if the Gaussian-reduced grid is stored from North to South. If this convention is not respected, the bins definition will not fail but the interpolation will become a 4 distance-weighted nearest-neighbour for all target points.

In conclusion, the only robust implementation of the bin restriction is for the conservative remapping for all grid types and for the bilinear and bicubic interpolations for Gaussian-reduced grid stored from North to South. OASIS3-MCT_4.0 will be modified to disable faulty combinations and the OASIS3-MCT_4.0 User Guide will be updated to clarify these points.

An important effort was also devoted to analyse the way cell bounding boxes are defined, highlighting some strong drawbacks of the current method. In some cases, the current method results in too large bounding boxes leading to a drastic reduction of the restriction effectiveness. A new strategy for defining them and for calculating their intersection is proposed in Appendix 1 of Piacentini et al. 2018. Great gains are expected with this new strategy. For example, on an Intel(R) Core(TM) i7-4930MX with 3.0GHz clock, the generation of the interpolation weights from ORCA025 to T359 with the old bounding box definition takes 851 seconds without binning but still 840 seconds with 500 latitude bins, while with the new bounding box definition it takes 821 seconds without binning and the time drops to 40 seconds with 500 bins.

Finally a bug was found and solved for the bilinear and bicubic interpolations for Cartesian grids (see Appendix 3 of Piacentini et al). This bugfix is included in the test below and is currently available in OASIS3-MCT source code on the SVN trunk.

Code optimisation in sequential mode

A pre-processing key -DTREAT_OVERLAY allows the detection of overlapping points due to periodical or polar closures. This was added in the OASIS version of the SCRIP. When activated, only the point with lowest index is active and the replicas are masked out. The original version used to scan the whole grid for every point to be checked (complexity $O(n^2)$). The new version sorts the grid coordinates calling a modified version of the standard heapsort (complexity of $O(n \log(n))$. This greatly improves the efficiency of this overlap check, e.g. reducing its cost from 731 seconds to 0.4 in the orca025 to t359 remapping. Since the parallelisation would require extra storage to avoid conflicts while modifying the grid mask, it was decided to keep this improved treatment sequential.

The part of the code associating a complementary non-masked nearest neighbour to non-masked target cells that are not involved in any conservative link (FRACNNEI option) was also optimised. These modifications significantly enhance the performance of OASIS. For example, the computing time of this whole complementary non-masked neighbour treatment for the T359

[2] For bilinear and bicubic, we also noted that if a grid point has at least one neighbour from the original bilinear/bicubic stencil masked, a distance weighted sum of the 4 nearest non masked neighbour values will be applied, but the corresponding 4 nearest neighbour search is restricted by the same original binning, leading to a strong dependency between the relative position of the selected points and the form of the bins. It may happen for example that a source point located at a similar latitude but relatively far in longitude will be chosen instead of a source point that would be closer in longitude and absolute distance but located in another southern or northern bin.

to ORCA025 coupling goes down from 293 to 5.9 seconds. As noted above, this part of the code was also parallelised with OpenMP

Preliminary modifications of the SCRIP library to support the parallelisation

Some work was also done to reduce the private working memory that needs to be duplicated for each OpenMP thread, as this could easily lead to memory saturation. For example, instead of duplicating on each thread the list of eligible cells resulting from the binning restriction in a mask array, the eligibility test is now directly performed on each thread. Also, even if the global addresses of the restricted list of cells to search are duplicated for each thread, indirect addressing is now used to get the grid information (e.g. the cell corner longitude) in (non-duplicated) global grid arrays.

Work was also needed to transform the way the links between source and target cells are stored. Since more than one border of a grid cell can cross one cell of the other grid, more than one line integral can provide a contribution to a conservative remapping link (each remapping link involves one target and one source cell). In the original version, a new contribution was stored as a new link only if no corresponding address pair was found in the already stored links. This approach was not viable in parallel since the order of the computations is unpredictable and race conditions are easily encountered. In the new version, every line integral contribution is stored as a new link. At the end of the parallel section, the links stored by each single thread are gathered into memory per MPI process and that memory is then gathered on the master process to be sorted and written to the weight file.

Parallelisation tests and results

A dedicated toy coupled model parallelised with MPI and OpenMP was developed to test and evaluate the performance of the new parallel calculations of the interpolation weights on different grids used in real coupled systems. A typical high-resolution (HR) coupled system with one component running on the NEMO ocean model ORCA025 grid (1442x1050 grid points) and the other component on a T359 Gaussian-reduced grid (181724 grid points) was created. Similarly, a typical ultra-high-resolution (UHR) coupled system using the NEMO ocean model ORCA12 grid (4322x3147 grid points) and the T799 Gaussian-reduced grid (843490 grid points) was also implemented.

The performance of the weight calculation was tested on Météo-France Bullx beaufix (Intel 5.1.2.150 compiler and MPI library) for the following 4 interpolations in both directions (ORCA to Gaussian-reduced and vice-versa): 4 distance-weighted nearest-neighbour, bilinear, bicubic and conservative first order (with FRACNNEI option). For the ORCA grid, a restriction of neighbouring search with 500 bins is used with conservative interpolation only (the accuracy of the interpolation wouldn't be guaranteed otherwise, see above), while the number of bins is automatically given by the truncation number for the Gaussian-reduced grid for bilinear and bicubic interpolations (see above). Reproducibility of the results at the machine precision (due to different operation orders) was validated.

Results are shown on Fig.3 for ORCA025 to T359 (HR) and on Fig.4 for ORCA12 to T799 (UHR), for 1,2,4,8,20 and 40 OpenMP threads and 1,2,4,8,16,32,64,128 and possibly 256 MPI tasks, i.e. a total of 1,2,4,8,20,40,80,160,320,640,1280,2560 and 5120 OpenMP threads (40 threads correspond to the number of physical cores per node on beaufix).

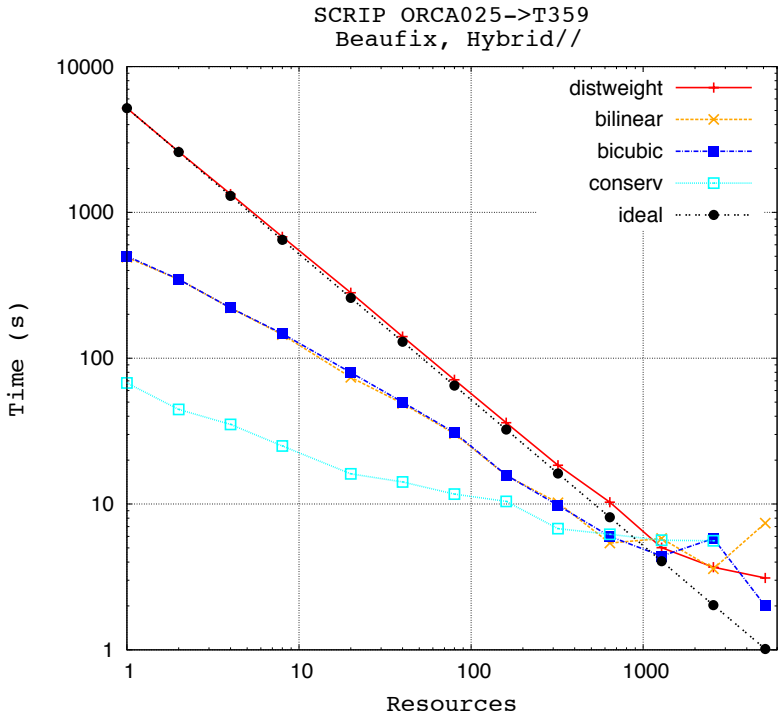


Figure 3: Time for the interpolation weight calculation as a function of the total number of OpenMP tasks for different interpolation with the new parallel version of the SCRIP library for the HR case: ORCA025 (1442x1050) to T359 (181724) coupling.

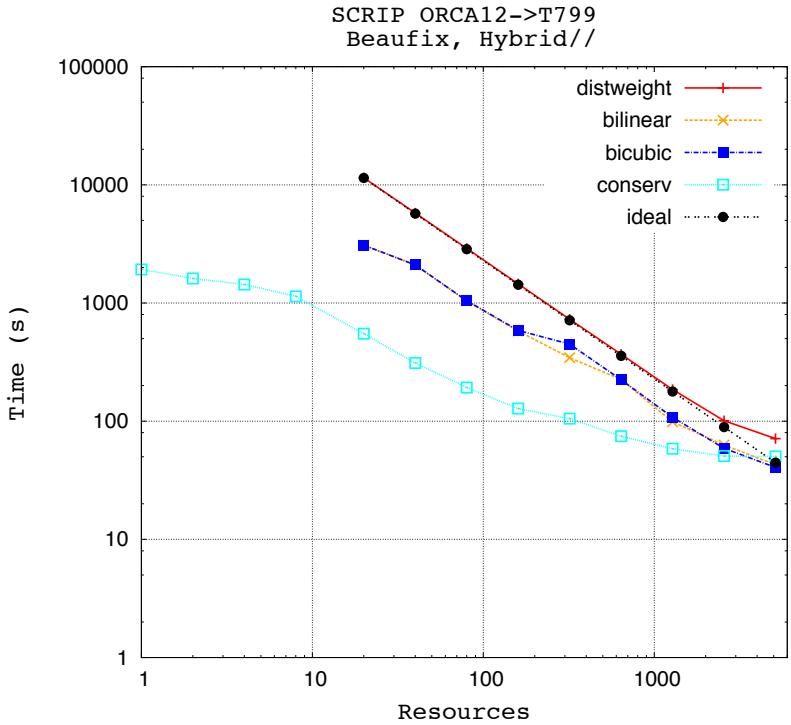


Figure 4: Same as Fig. 3 but for the UHR case: ORCA12 (4322x3147) to T799 (843490) coupling.

For distance-weighted nearest-neighbour (distweight), bilinear, bicubic, the large number of source points considered in the search (no bin restriction is applied as explained above) slows down the calculation at low resources but favours the good scaling for up to 1280 threads for HR and up to 2560 tasks for UHR. A higher scalability would be achieved with a better load balancing, which is made difficult by the heterogeneity of the operations per target grid point (complementary neighbour search, iterative loops, ...).

For the conservative interpolation (conserv), the hybrid parallelisation gains are also very significant but here are some assumptions to explain the less-scalable behaviour of this interpolation:

- at low resources, better performance is observed in comparison to the other interpolation, due to the bin restriction
- a large load imbalance between the different threads affects performance due to the variable number of possible neighbours to check
- the scalability limit is about the same for all interpolations and again, this is explained by the calculation heterogeneity per target grid point (complementary neighbour search, pole projections at high latitudes)

In Fig. 5, we compare the computation time for four interpolations, in green with the SCRIP library before parallelisation (OASIS3-MCT_3.0 version of the code), and in red after our optimization and parallelisation work (available in OASIS3-MCT SVN branch *eahybrid*) for a number of threads that maximizes the calculation speed (i.e. 2560 for bilinear and conserv and 5120 for bicubic and distweight for HR, and 5120 for all interpolations for UHR). At any resolution, the new version of the library leads to a reduction in the weight calculation time of 2 or 3 order of magnitude (note the logarithmic elapsed time axis).

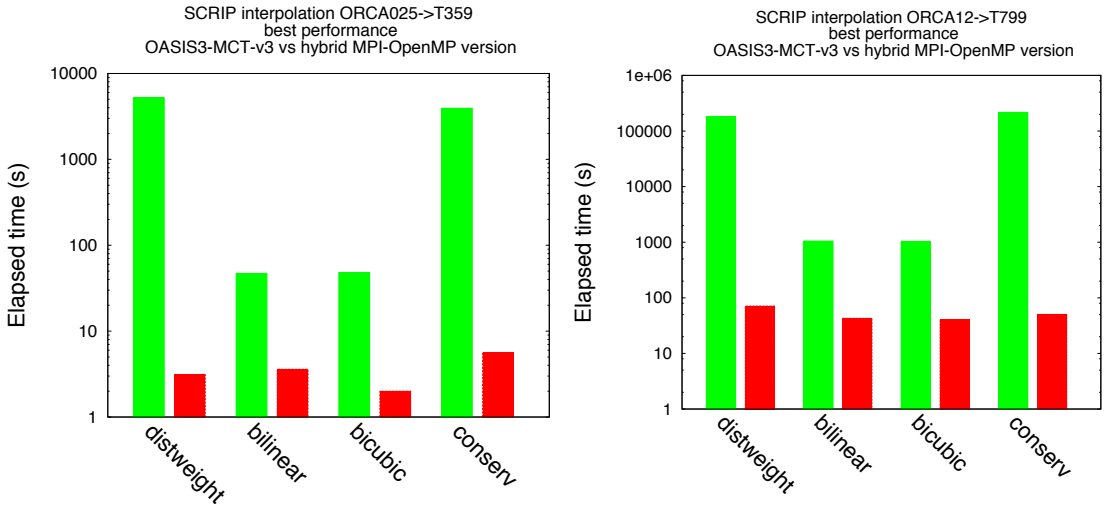


Figure 5: Computation time for the interpolation weight calculations on one core with the original OASIS3-MCT_3.0 version (green) and with the new parallel version of the SCRIP library (red) on 2560 tasks (bilinear and conserv), 5120 tasks (bicubic and distweight) for HR (left) and 5120 tasks (all interpolations) for UHR (right).

The performance of the SCRIP library is greatly improved by its parallelisation, even if it does not reach ideal scalability. Further improvements would require a thorough rewriting of the grid

search algorithms, which was beyond the scope of the present work. Potential solutions are discussed in Piacentini at al 2018, including:

- implementation a correct binning restriction method for distance-weighted, bilinear, and bicubic interpolation
- improving the load balancing between the different tasks/threads
- acceleration of the distance evaluation replacing expensive trigonometric functions by simpler multiplications
- introducing an additional level of parallelism by letting all models performing their weight calculations at the same time, with a preprocessing of the *namcouple* configuration information
- inclusion of the new Monte-Carlo method recently added in the official version of the SCRIP library, to take benefit of the new GPU capabilities

In conclusion, the results obtained here, which show a reduction in the weight calculation time of 2 or 3 orders of magnitude with the new parallel version of the SCRIP library for high-resolution grids, let us envisage the runtime weight computation at the coupling frequency and opens the door to dynamical coupling with OASIS3-MCT.

4.3. Optimisation and debugging of the coupling initialisation

The IS-ENES2 coupling technology benchmark (Valcke et al. 2017) showed that the initialisation time for the test cases using OASIS3-MCT was relatively high compared to other couplers, especially as OASIS3-MCT initialisation phase did not include any remapping weight calculation. Some efforts were therefore devoted to optimize the initialisation phase of the coupler by upgrading the MCT library. Significant gains were obtained but the initialization cost at high number of cores was still inexplicably high. Fortunately a bug in the initialisation was identified and solved and the results of these two improvements are detailed here.

4.3.1. Update of MCT library from version 2.8 to 2.10.beta1

The technical report "OASIS3-MCT_4.0 Timing Study with MCT 2.10.beta1" (Craig & Valcke, 2018) details the gain obtained by upgrading the MCT library from version 2.8 to 2.10.beta1, which includes some optimization in the router initialization. The tests were done on Météo-France Bullx beaufix for the VHR test case with 1600 and 3600 tasks per component, with Intel 16.1.150 compiler and the Intelmpi 5.1.2.150 MPI library. The different timing measures included in this report have already been presented in section 4.1.3. Note that all tests presented here after also include the initialisation bug fix discussed in section 4.3.2.

Compared to MCT 2.8, MCT 2.10.beta1 improves by one to two orders of magnitude the performance of the initialization of the MCT routers for the VHR test case[3]. The benefits for upgrading from MCT 2.8 to MCT 2.10.beta1 are particularly significant for complex decompositions and rearrangements.

For the VHR case (3000x3000 grid points) on 3600 cores per component, the MCT upgrade reduces the total initialization time in OASIS3-MCT from between 1 and 3 minutes to 10-20 seconds for the VHR case. In particular, the cost to compute the router between the source

[3] We recall that the routers define the rearrangement patterns for the sparse matrix multiply and the rearrangement associated with coupling data between the source and target processes.

decomposition and the mapping decomposition to support the sparse matrix multiplication on the source tasks (*sminit*, see section 4.1.3 of the report) is reduced from ~19 sec to ~0.5 sec on 1600 tasks/component and from ~41 sec to ~0.6-0.7 sec on 3600 tasks/component for the *decomp_wghtfile* method. And the cost to initialize the router for the coupling rearrangement between the mapping decomposition on the source tasks and the target decomposition on the target tasks (*router init*, see section 4 of the report) is reduced from ~60 sec to ~6-7 sec on 1600 tasks/component and from ~124 sec to ~5-7 sec on 3600 tasks/component, still for the *decomp_wghtfile* method.

It can be concluded that the upgrade from MCT 2.8 to MCT 2.10.beta1 brings in a significant reduction of the cost of the initialisation phase and this is even more welcome given the extra initialisation cost of the new *decomp_wghtfile* method, which itself brings significant reduction of the run time.

4.3.2. Impact of initialisation bug fix

While working on the initialisation, the explanation of the severe slow down of the initialisation phase at high number of cores observed in the IS-ENES2 coupling technology benchmarks, (see e.g. Valcke et al 2017 Fig 3a for the VHR case and Fig 5a for the VHR_oppdec case) was found. The problem was caused by some concurrent writing into the OASIS3-MCT debug file by all tasks even for the lower level of debugging. Figure 6 and 7 show the striking impact of the bug fix for the VHR_oppdec and for the HR_tutorial test cases respectively, still on Météo-France Bullx beaufix with Intel 16.1.150 compiler and the Intelmpi 5.1.2.150 MPI library. Note that these tests include the upgrade to MCT 2.10.beta1 described in section 4.3.1.

For VHR_oppdec, Fig. 6 shows a reduction of 99% for the whole initialisation time at 10240 cores for the bug fixed version (OASIS3-MCT_4.0 on the graph) as compared to OASIS3-MCT_3.0. For HR_tutorial, the gain is also very important with a reduction of 82% as shown on Fig. 7.

It can be concluded that this bug fix, that will be available in OASIS3-MCT_4.0, may have some significant effect in real coupled systems, especially for systems with short run length for which the impact of the initialisation cost is relatively higher.

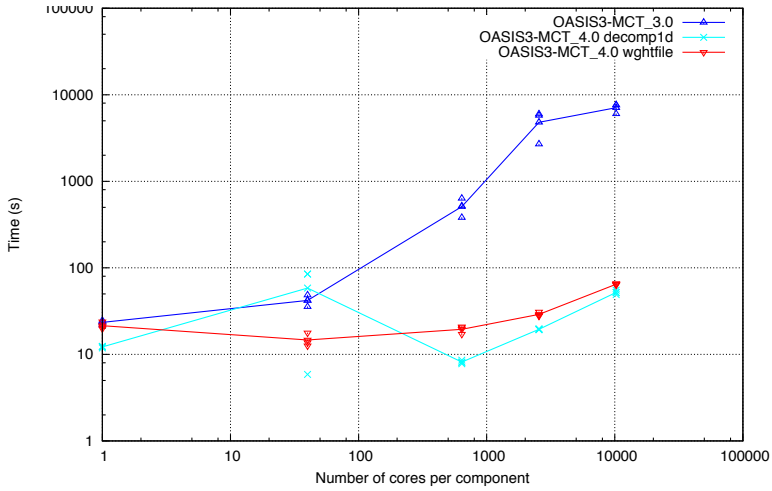


Figure 6: Time for the coupling initialisation with respect to the number of tasks/cores used for each component model for the VHR_oppdec test case, for the previous OASIS3-MCT_3.0 version (dark blue) and for the branch tc17b r2069 including the initialisation bugfix, activating the *decomp_1d* (light blue) or *decomp_wghtfile* (red) method that will be available in the next OASIS3-MCT_4.0 release.

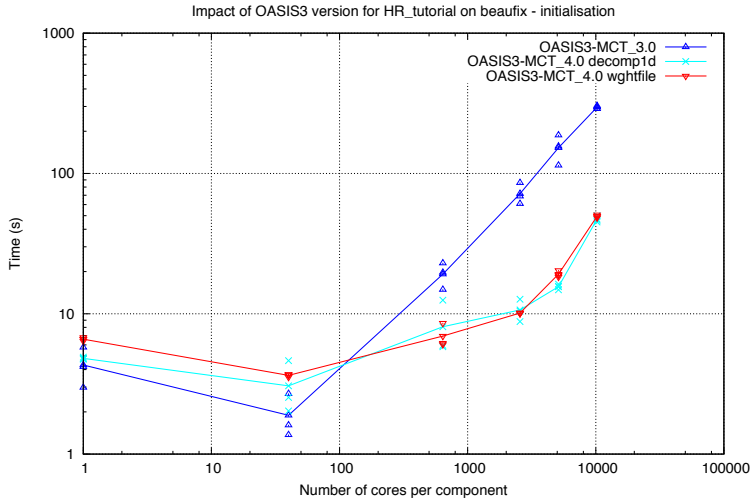


Figure 7: Time for the coupling initialisation with respect to the number of tasks/cores used for each component model for the HR_tutorial test case, for the previous OASIS3-MCT_3.0 version (dark blue) and for the branch tc17b r2069 including the initialisation bugfix, activating the *decomp_1d* (light blue) or *decomp_wghtfile* (red) method that will be available in the next OASIS3-MCT_4.0 release.

4.4. Optimisation of global CONSERV operation

The global CONSERV operation available in OASIS3-MCT ensures the global conservation of a remapped coupling field. This operation computes the integral of the coupling field before and after the remapping and redistributes the difference on the target grid points. In OASIS3-MCT_3.0, this global CONSERV has only two options, *bfb* and *opt*. Option *bfb* enforces a bit-for-bit transformation regardless of the component grid decomposition or number of processes. In *bfb* the entire field is gathered from the different component processes to the master process, and that process performs the global integration and then broadcasts the resulting sum to all other component processes. Option *opt* carries out the global conservation with an optimal algorithm using less memory and a faster approach: a local sum is performed on each process, those local sums are sent to all other processes and all processes can then compute the global sum of all local sums. This is more efficient than the *bfb* algorithm but does not ensure bit-for-bit reproducibility when the grid decomposition or the number of processes of the component is changed.

There are now five options (*lsum8*, *lsum16*, *ddpdd*, *reprosum* and *gather*) to compute the global sums in CONSERV. Option *gather* and *lsum8* are respectively equivalent to the former *bfb* and *opt* options. The *lsum16* works just like *lsum8* but uses quadruple precision to compute the local sums and to carry out the scalar reduction. The cost of *lsum16* will be higher than *lsum8*, but there is a greater chance that results will be bit-for-bit for different decompositions. The *ddpdd* is a parallel double–double algorithm using a single scalar reduction (He and Ding, 2001). This algorithm should behave between *lsum8* and *lsum16* with respect to performance and reproducibility. The third new algorithm, *reprosum*, is a fixed point method based on ordered double integer sums that requires two scalar reductions per global sum (Mirin and Worley, 2012). The cost of *reprosum* will be higher than some of the other methods, but it is expected to produce bit-for-bit results on different task counts except in extremely rare cases, and the cost should be significantly less than the *gather* method.

cores, mapping	CONSERV unset	CONSERV *lsum8*	CONSERV *lsum16*	CONSERV *ddpdd*	CONSERV *reprosum*	CONSERV *gather*
48, *src*	4.00	8.27	16.78	10.65	17.34	117.72
48, *dst*	4.39	8.02	16.59	10.42	16.98	142.12
180, *src*	1.25	2.21	4.59	2.87	4.85	126.91
180, *dst*	1.56	2.26	4.62	2.92	4.90	130.01

Table 1. Comparison of ping-pong times for HR_tutorial on Lenovo on 48 and 180 cores per component with the CONSERV option off (unset), set to *lsum8* (*opt* in OASIS3-MCT_3.0), *lsum16*, *ddpdd*, *reprosum* and *gather* (*bfb* in OASIS3-MCT_3.0). Times (in seconds) are accumulated over 1000 ping-pongs for a single coupling field in each direction. Two trials of each case were carried out and the minimum time is shown. Differences between trials were less than 2% except for the *gather* case where variations in time of up to 10% were observed.

Table 1 shows the ping-pong timings for the HR_tutorial case on Cerfacs' Lenovo cluster (using the Intel compiler and MPI 5.0.3.048) for four different configurations combining *src* or *dst* for

the remapping location (either on the tasks of the source component, *src*, or the target component, *dst*), for 48 and 180 cores per component, with CONSERV unset, and CONSERV set to *lsum8* (equivalent to *opt* in OASIS3-MCT_3.0), *lsum16, ddpdd, reprosum,* and *gather* (equivalent to *bfb* in OASIS3-MCT_3.0). The CONSERV operation increases the ping-pong time by at least 50% regardless of the method used, and the *gather* option stands out with respect to cost. The *lsum8* is the fastest CONSERV method while *reprosum* is probably the best choice if bit-for-bit reproducibility is sought as is only slightly more expensive than *lsum16*, but its reproducibility characteristics are significantly better.

In conclusion, the *reprosum* option should be considered as the first choice as its performance and reproducibility characteristics are good. However, when using CONSERV, it is important to test the performance of various methods and consider carefully the scientific requirements. Of course, when possible, mapping weights that are inherently conservative such as area overlap conservative should be used to avoid use of any global CONSERV operation.

4.5. Additional tests realized with IS-ENES2 coupling benchmarks

4.5.1. Impact of “nointerp” option on VHR test case

The runs presented in this section were motivated by the results of the IS-ENES2 coupling technology benchmarks, which showed that OASIS3-MCT was systematically about 5 times slower than the other couplers for the coupling exchanges (see e.g Valcke et al 2017 Fig 3b for the VHR case and Fig 5b for the VHR_oppdec case).

In the IS-ENES2 benchmarks, the grids of the coupled component models are the same and therefore no remapping is needed. However, for the test cases implemented with OASIS3-MCT_3.0, we forced the activation of the sparse matrix multiplication, the weight matrix being in this case the identity matrix, so to be representative of real coupling exchanges usually involving a remapping. In the additional tests performed with the VHR test case on Bullx beaufix, we removed this unnecessary step for comparison.

Figure 8 presents these new results together with the previous IS-ENES2 benchmark results run on Bullx Occigen at CINES using Intel compiler 15.0.3.187, and bullxmpi 1.2.9.2. We see that, without the forced remapping, OASIS3-MCT performs as well as, and even better at very high number of cores, than the other coupling technologies.

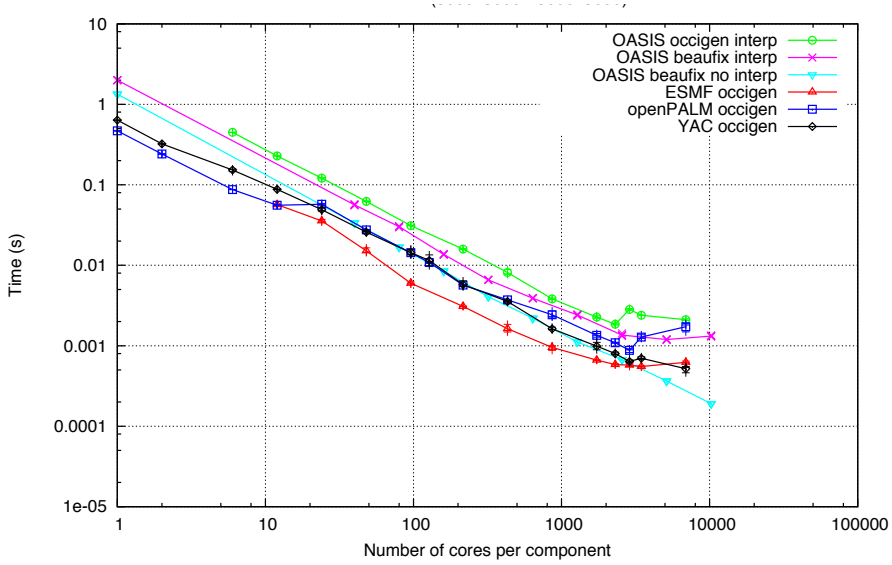


Figure 8: Time for a ping-pong exchange with respect to the number of tasks/cores used for each component for the VHR test case run on Bullx Occigen with YAC (black), OpenPALM (dark blue), ESMF (red) and OASIS3-MCT with forced remapping (green), and on Bullx beaufix with OASIS3-MCT with (pink) and without (light blue) forced remapping.

4.5.2. Additional benchmarking on Marconi KNL

Additional runs using the IS-ENES2 coupling technology benchmark test cases were also run on Marconi KNL to investigate the behaviour of OASIS3-MCT on this type of platform. Marconi is CINECA class Tier-0 supercomputer, based on Intel® Xeon Phi™ product family "Knights Landing" (KNL) alongside with Intel® Xeon® processor E5-2600 v4 product family. It has been co-designed by Cineca on the Lenovo NeXtScale architecture. The tests presented on Figure 9 were realized with the VHR and VHR_oppdec test cases on the Marconi KNL partition with branch OASIS3-MCT_3.0_branch r2009 with Intel mpiifort compiler and impi 2017.3.196, thanks to an allocation of 390 000 core hours granted to ESiWACE.

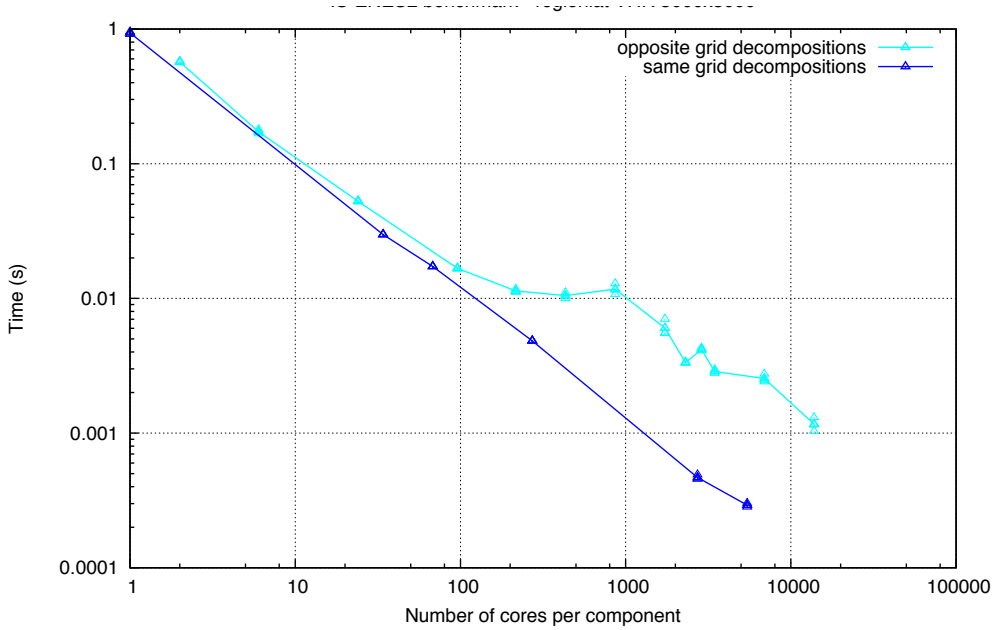


Figure 9: Time for a ping-pong exchange with respect to the number of tasks/cores used for each component for the VHR test case (lower blue curve) and for the VHR_oppdec test case (upper blue curve) run on Marconi KNL partition.

We can conclude that OASIS3-MCT behaves similarly on Marconi KNL as other platforms tested and shows a nice, almost perfect scalability curve for up to $O(10^4)$ cores/tasks per component for the VHR test case and a very reasonable behaviour for the VHR_oppdec test case involving much more communication.

4.6. Other developments

For the sake of completeness, additional developments realised in the last 24-month period are briefly described here, as they will be new features offered by OASIS3-MCT_4.0.

Bundle fields

The ability to couple a bundle of 2-D fields via extension of the OASIS3-MCT calling interface was implemented, i.e. an extra last dimension is supported in the field arrays sent (*oasis_put*) or received (*oasis_get*) through OASIS3-MCT API. Different bundled fields can have different numbers of fields, but for a given bundled field, the number of fields must match on the send and receive sides. The bundled fields must share a common partition and common coupling settings (e.g. remapping). While this is a useful feature for multi-level fields, this does not mean that 3D interpolation is supported. Each field in the bundle is treated internally as a separate 2D field in the coupling layer without any information about the relationship between the fields in the bundle.

Automatic coupling restart writing

An optional argument *write_restart* was added to the *oasis_put* routine. This argument is false by default but if it is explicitly set to true in the code, a coupling restart file will be written for that field only for that coupling timestep, saving the data that exists at the time of the call.

Exact consistency between the number of weights and fields

Exact consistency is now required between number of weights fields in the coupling restart file and the arrays passed as arguments to the *oasis_put* routine. For example, for a 2nd order conservative remapping (CONSERV SECOND), 3 weights are needed and 3 fields must be provided as arguments i.e. the value of the field, its gradient with respect to the longitude and its gradient with respect to the latitude. For a first order conservative remapping (CONSERV FIRST), only one weight and one field are needed. Using a weight file with 3 weights for a first order conservative remapping is no longer allowed.

Modifications in the namcouple configuration file

The *namcouple* reading routine was cleaned up including a refactoring of the *gotos* and *continue* statements, addition of few reusable routines including an abort routine, removal of some dead code, addition of support for blank lines (which are now considered comments), removal of requirement that keywords start at character 2 on a line, removal of requirement for $END in the *namcouple*, and updates to some error messages.

New functionalities with corresponding new namcouple keywords

- $NUNITNO: specifies the minimum and maximum unit numbers to be used for input and output files in the coupling layer.
- $NMAPDEC: indicates the mapping decomposition method to be used, either *decomp_1d* or *decomp_wghtfile* (see section 4.1)
- $NMATXRD: indicates the method used to read mapping weights, either *orig* and *ceg*. In both methods, the weights are read in chunks by the model master task. With the *orig* option, the weights are then broadcast to all other tasks and each task then saves the weights that will be applied to its grid points. With the *ceg* option, the master task reads the weights and then identifies to which other task each weight should be sent. A series of exchanges are then done with each other task involving just the weights needed by that other task. The *orig* method sends much more data but is more parallel, while the *ceg* method does most of the work on the master task but less data is communicated.
- $NNOREST : if true, OASIS3-MCT will initialise any variable that normally requires a coupling restart file with zeros if that file does not exist.

4.7. Appendix A – Test cases used to evaluate OASIS3-MCT performance

Different test cases were used to evaluate the performance improvement of OASIS3-MCT along the course of the developments. We describe here in more details the "HR_tutorial", "VHR", or "VHR_oppdec" test cases to which we refer regularly along the text.

Ping-pong exchanges

In all these test cases, ping-pong exchanges are implemented between two "toy" components. Toy components are F90 programs that do not include any physics or dynamics, like a real geophysical model would do, but that implement realistic exchanges of coupling fields defined on specific grids. In a ping-pong exchange (see Figure 1 in Valcke et al. 2017), the first component uses some priming mechanism (here a simple initialization by an analytical function based on the spatial position of each grid point) to define its input coupling field at the beginning of its first time step, calculates an output coupling field (with a simple relation such as adding 1 to each field value) and sends it to the second component. The second component receives it at the beginning of its first time step, calculates its output coupling and sends it back to the first component that receives it at the beginning of its second time step. The time for a ping-pong exchange is calculated as the difference between time measures taken before the send action of a particular time step and the receive action at the next time step in the first component. This time includes a full back-and-forth exchange between the two components.

HR_tutorial

HR_tutorial is a toy coupled model with 100 time steps implementing a ping-pong exchange between a component using the NEMO ORCA025 grid with 1021x1442 grid points and a T799 Gaussian Reduced grid with 843 000 grid points.

VHR

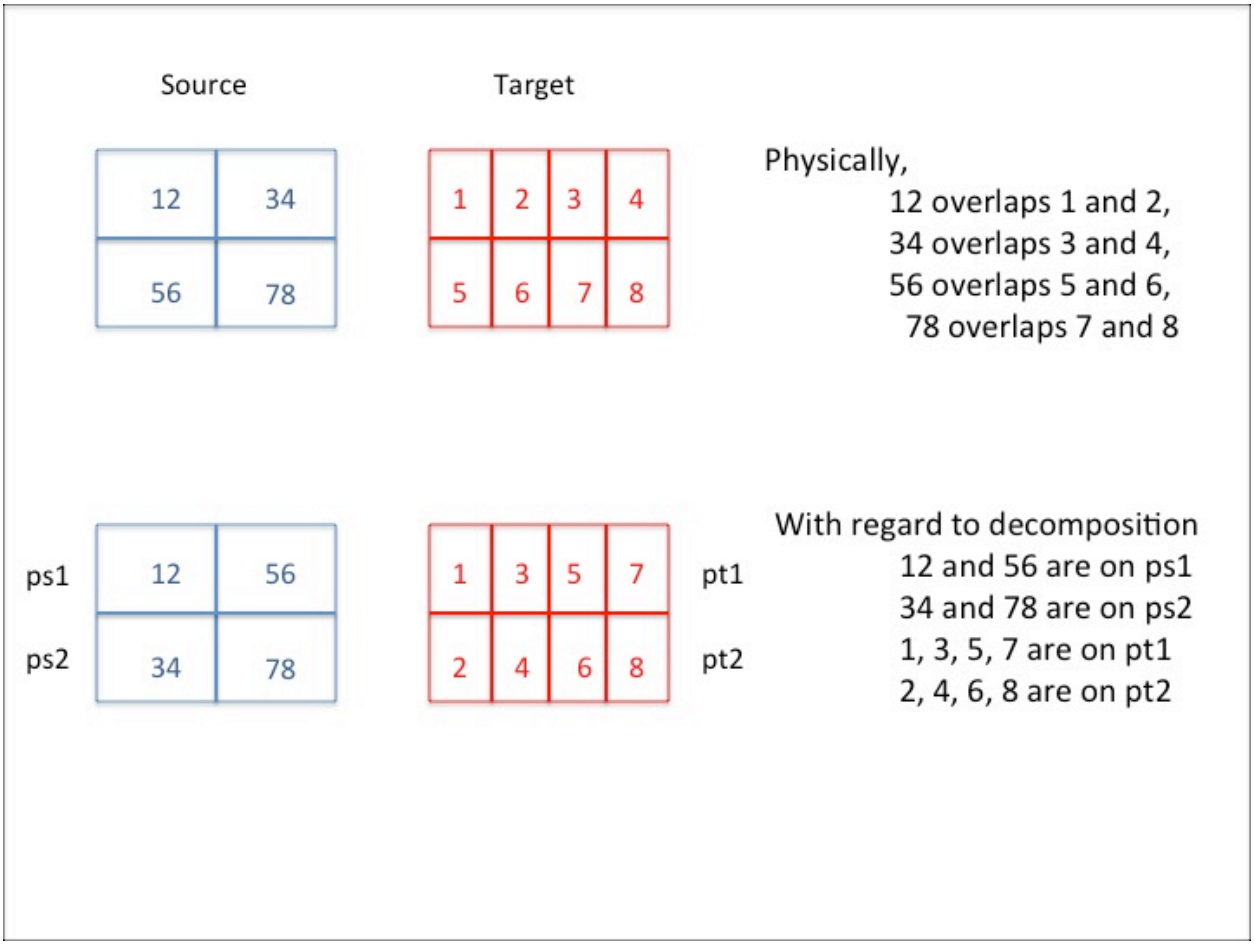
VHR stands for "Very High Resolution" and is one test case of the IS-ENES coupling technology benchmark. It runs 100 time steps implementing a ping-pong coupling exchange between two components running on the same regular latitude-longitude grid with 3000x3000 points. The grid domain is decomposed over the number of cores available for the component with the same as-square-as possible partition.

VHR_oppdec

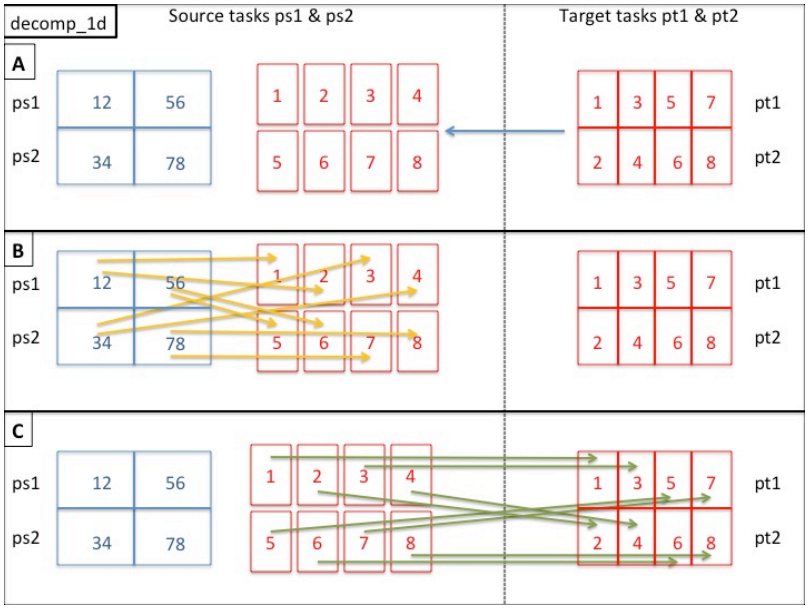
VHR_oppdec stands for "Very High Resolution with opposite decomposition" and is also one test case of the IS-ENES coupling technology benchmark. As for the VHR test case, it runs 100 time steps implementing a ping-pong coupling exchange between two components running on the same regular latitude-longitude grid with 3000x3000 points. However, the partitions are in this case defined with an aspect ratio as big as possible and as "opposite" as possible for the two grids. For example if 24 cores are used for each component, one will define 24 "latitudinal" partitions of 125x3000 grid points while the other will define 24 "longitudinal" partitions of 3000x125 grid points.

4.8. Appendix B - Illustration of decomp_1d and decomp_wghtfile communication schemes

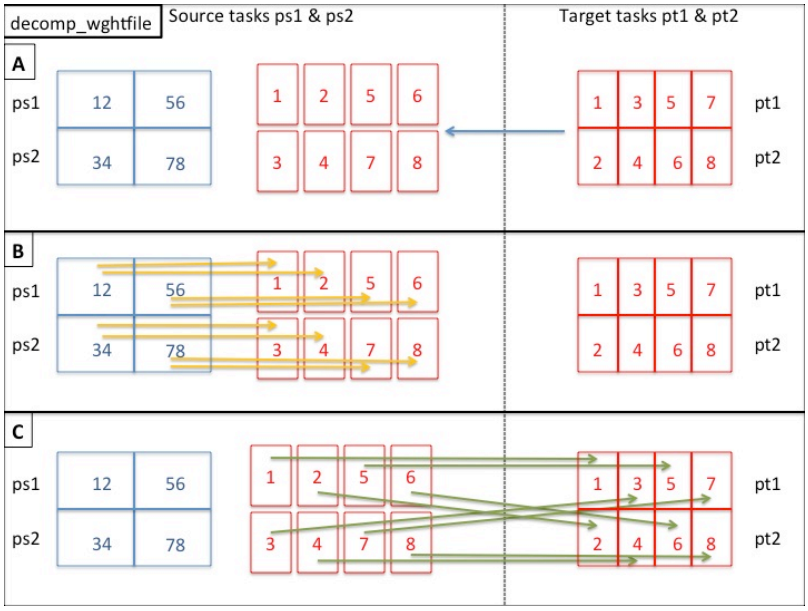
The different steps of the communication for the *decomp_1d* and *decomp_wghtfile* options are described here with a very simple example involving two models, the source one with 4 grid cells running on 2 tasks/cores ps1 and ps2, and the target one with 8 cells running on 2 tasks/cores pt1 and pt2. In this example, the remapping is performed on the source tasks.



This figure illustrates the geographical overlapping of the different cells (top) and the distribution of the cells on the tasks/cores (bottom). For example, the source cell 12 overlaps target cells 1 and 2, which are respectively managed by target tasks pt1 and pt2 respectively.



This figure illustrates the communication scheme established with the *decomp_1d* method. In A, the mapping decomposition of the target grid points on the source tasks is defined by a trivial one-dimensional approach: grid points 1, 2, 3 & 4 are assigned to source task ps1 and grid points 5, 6, 7 & 8 to source task ps2. B illustrates the rearrangement of the source grid decomposition to the mapping decomposition involving 4 messages (ps1 - ps1, ps1 - ps2, ps2 - ps1, ps2 - ps2) among which 2 are non-local. C shows the rearrangement from the mapping decomposition to the target decomposition on the target tasks, involving 4 messages (ps1 - pt1, ps1 - pt2, ps2 - pt1, ps2 - pt2) all being non local.



With the *decomp_wghtfile* method, the weights are used to define the mapping decomposition of the target grid points on the source tasks. Grid points 1, 2, 5 & 6 are assigned to source task ps1 and grid points 3, 4, 7 & 8 to source task ps2. The rearrangement from the source decomposition to the mapping decomposition (B) therefore is much simpler and involves only 2 local messages and the rearrangement from the mapping decomposition to the target decomposition on the target tasks (C) still involves 4 non-local messages as for *decomp_1d*.

4.9. Appendix C – User Survey on OASIS use in OpenMP multithreaded models

December 2017

Dear OASIS users,
we are drawing some guidelines for interfacing a hybrid MPI/OpenMP parallel component and OASIS. If the codes you are coupling or you are planning to couple in a near future have multi-threaded capabilities, we'd be grateful if you could answer a quick survey helping us in preparing fully representative examples and toys.
Questions have different degrees of technicalities: feel free to skip what sounds too geeky.

About multi-threading handling:

Q1) Does your application uses OpenMP or pthread ?

Q2.1) If it uses OpenMP, please provide a short description of the multi-threading strategy, focusing on how many parallel regions are invoked and where, the use of implicit or explicit task scheduling, the presence of *reductions*, *critical* sections, *atomic* instructions or other synchronisations, the use of *single* or *master* sections, etc. What is the minimum OpenMP standard compliance required (2.0, 3.0, 3.1, 4.0) to the compiler ?

Q2.2) If it uses pthread, please provide a short description of the multi-threading strategy.

About the layout of the Hybrid MPI + multi-thread hybrid parallelism (at least for code components dealing with the coupling):

Q3) Is the MPI layer using one process per node, per socket, per core, or other?

Q4) How many threads per MPI process are active at most?

Q5.1) If the multi-threaded layer relies on OpenMP, how is the scheduling of tasks (or threads) implemented: implicit tasks at loop level (or workshare) or explicit tasks? Is the scheduling static, dynamic or guided?

Q5.2) If the multi-threaded layer relies on pthreads, please, list specific pthread features that could have an impact on the coupling implementation.

Q6) What is the required MPI thread support at MPI_Init : *MPI_THREAD_FUNNELED*, *MPI_THREAD_SERIALIZED* or *MPI_THREAD_MULTIPLE*?

Q7) What's the strategy for the internal MPI communications amongst processes and threads of the component code: outside multi-threaded sections, funneled through the *master* thread, funneled through a *single* section, multiple from individual threads (and in such a case, please provide hints on the adopted tagging strategy)

Interactions with the coupler (or with forcing input and fluxes output in forced mode):

Q8) Are the code components dealing with the coupling inside a single multi-threaded region (possibly higher in the call tree) or do they contain both serial and multi-threaded regions?

Q9) If there are both serial and multi-threaded regions, are the interactions with the coupler (or the forcing and fluxes) only in the serial MPI process code (i.e. outside any *OMP PARALLEL* or similar construct) or can they be within multi-threaded regions?

Q10) If they are within multi-threaded regions, do they go through a single (possibly the *master*) thread or are they invoked from individual threads?

Q11) Are the computed fields (e.g. the prognostic 3D atmospheric fields) stored as shared variables (with threads only sharing the extent of the loops or, more generally, the workload) or as a collection of *threadprivate* (or plainly *private*) variables ?

Q12) Are the coupling fields directly received (or produced) with the same storage strategy as used in computations, or do they need to go through manipulations (like, for instance, 2 indices ↔ 1 index mapping, masking and compression, etc)?
Q13) If the fields don't need to go through manipulations, are they exchanged directly from the computing storage location (e.g. the surface level of a 3D field) or are they copied to/from a temporary buffer?
Q14) If the fields go through manipulations and the code is multi-threaded, are the manipulations multi-threaded or performed only by a single thread ?

Summary of the strategy:

Q15) If the coupling exchanges are performed by a single thread (possibly the master) within a multi-threaded region, accordingly to the answers of the previous paragraph and referring to the images in the following table, which of the following configurations best represents your case? Please describe another summary configuration if none fits.
Notice that sub-point x.1) refers to question Q11, sub-point x.2 refers to question Q12, sub-point x.3 refers to question Q13, transition between sub-points x.1 and x.2 refers to question Q14.

a) a.1) the computed fields are in *shared* memory
a.2) the coupling fields are taken from the computed field storage
a.3) the coupling interface works directly on them (from *master* thread)

b) b.1) the computed fields are distributed (threadprivate) amongst threads
b.2) the coupling fields are a simple recollection in an extra *shared* array of data coming as it is from the computed fields
b.3) the coupling interface works directly on the recollected shared fields (from *master* thread)

c) c.1) the computed fields are in shared memory
c.2) the coupling fields are taken from the computed field storage
c.3) prior to coupling they are manipulated and transformed into exchanged fields by the *master* thread (using extra *private* memory)

d) d.1) the computed fields are distributed (threadprivate) amongst threads
d.2) the coupling fields are a simple recollection in an extra *shared* array of data coming as it is from the computed fields
d.3) prior to coupling they are manipulated and transformed into exchanged fields by the *master* thread (using extra *private* memory)

e) e.1) the computed fields are either in shared memory or threadprivate (not relevant here)
e.2) the coupling fields are manipulated in extra *shared* memory and prepared for the exchange
e.3) the coupling interface works directly on the manipulated *shared* fields (from *master* thread)

f) f.1) the computed fields are either in shared memory or threadprivate (not relevant here)
f.2) the coupling fields are partially manipulated in extra *shared* memory before the exchange
f.3) prior to coupling the manipulations are finalized and the coupling fields are transformed into exchanged fields by the *master* thread (using extra *private* memory)

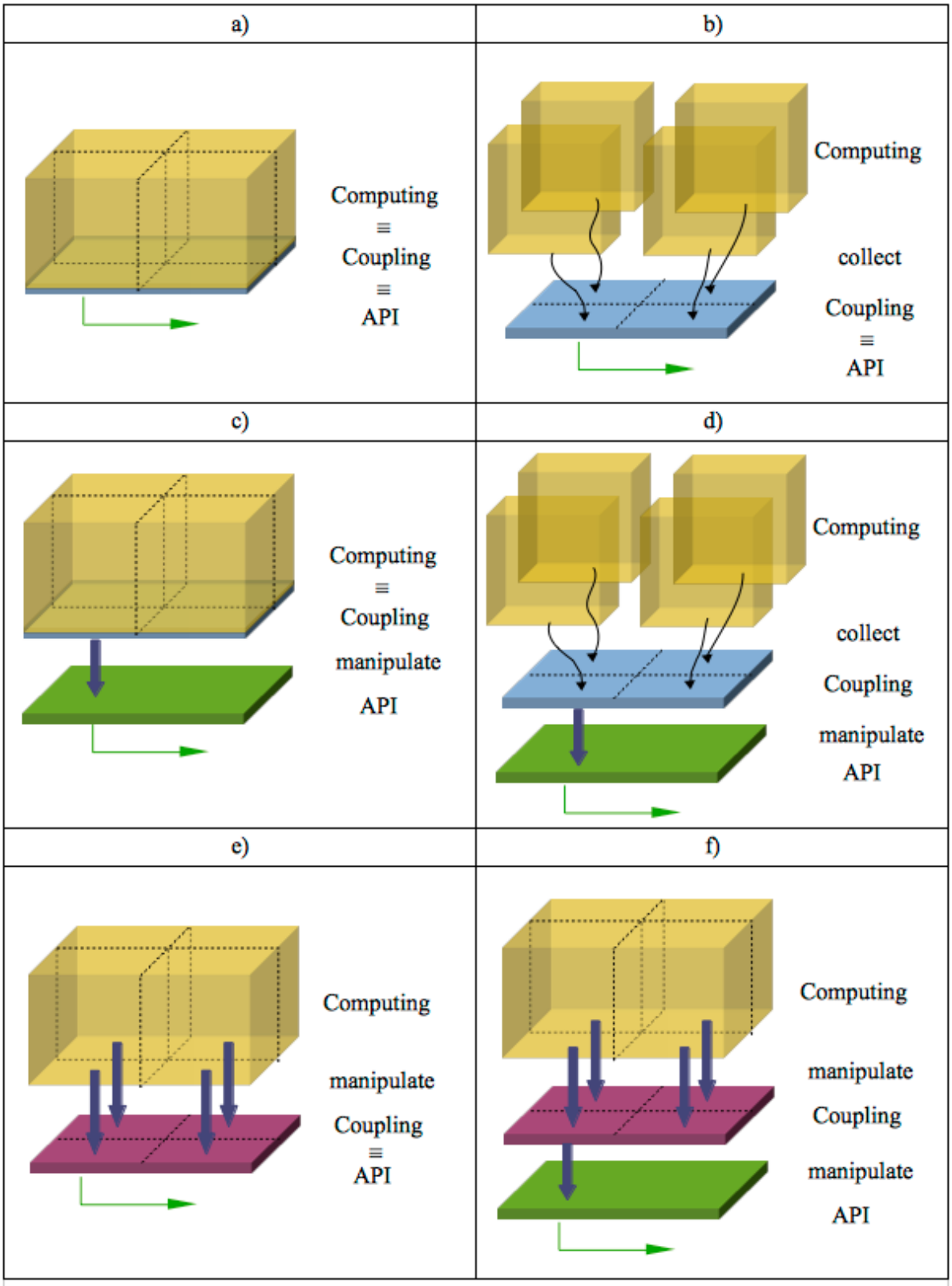


Table 1. Foreseen hybrid parallel code coupling configurations

Optional information: some statistics on the targeted computing environments

Target platform :
- manufacturer, processor series, number of sockets and cores per socket on each node, maximum memory per node

Target compiler :
- fortran / c compiler : distribution, release and version number
- supported version of OpenMP

Target MPI library :
- distribution, release and version number
- multi-thread support level
- MPI 3 support

Finally, please indicate if you would agree in answering some specific questions in private if we'd need to thoroughly enter the details of your code configuration.

Thank you very much for your time and attention.
The OASIS development team

5. References (Bibliography)

A. Craig and S. Valcke, 2018. OASIS3-MCT_4.0 Timing Study with MCT 2.10.beta1, Technical Report TR/CMGC/18-38, Cerfacs, France. https://cerfacs.fr/wp-content/uploads/2018/03/GLOBC-Craig_RT_oasis_mct_perf_final_2018.pdf

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.

L. Coquart, E. Maisonnave, S. Valcke, 2018: Using Open MP in OASIS3-MCT for the N-nearest-neighbor remapping, UMR 5318 CECI, CERFACS/CNRS, TR-CMGC-18-19, Toulouse, France. https://cerfacs.fr/wp-content/uploads/2018/01/GLOBC-Coquart_etal_OpenMP_in_OASIS3-MCT_22012018.pdf

Y. He and C. H. Q. Ding, 2001: Using Accurate Arithmetics to Improve Numerical Reproducibility and Stability in Numer- ical Applications, The Journal of Supercomputing, 18, 259, https://doi.org/10.1023/A:1008153532043, 2001.

A. A. Mirin and P. H. Worley: Improving the Performance Scalability of the Community Atmosphere Model, Int. J. High Perf. Comp. App., 26, 17–30, https://doi.org/10.1177/1094342011412630, 2012.

A. Piacentini, E. Maisonnave, G. Jonville, L. Coquart, S. Valcke, 2018. A parallel SCRIP interpolation library for OASIS, Technical Report TR/CMGC/18-34, Cerfacs, France. https://cerfacs.fr/wp-content/uploads/2018/03/GLOBC-TR-PIANCENTINI-cmgc_18_34.pdf

S. Valcke, G. Jonville, R. Ford, M. Hobson, A. Porter and G. Riley (2017), Report on benchmark suite for evaluation of coupling strategies, UMR 5318 CECI, CERFACS/CNRS, TR-CMGC-17-87, Toulouse, France (http://cerfacs.fr/wp-content/uploads/2017/05/GLOBC-TR-IS-ENES2_D10.3_MAI2017.pdf)

S. Valcke, T. Craig and L. Coquart, 2015. OASIS3-MCT User Guide, OASIS3-MCT_3.0, Technical Report TR/CMGC/15/38, Cerfacs, France.

6. Dissemination and uptake

6.1. Dissemination

The sources of OASIS3-MCT containing the developments described above are registered on OASIS SVN server, currently in different development branches, as indicated in the text. All these developments will be merged and distributed with the next official OASIS3-MCT_4.0 release planned for April or May this year (2018).
The major developments were shown at the ESiWACE general assembly in Berlin in December 2017. They will be also presented at the 5° HPC ESiWACE/ENES Workshop in Lecce May 17-18, 2018 and at the "Workshop on physics-dynamics coupling" (PDC18) at ECMWF in Reading in July 2018.
Finally the developments detailed in section 4.4 on the global CONSERV method were included in Craig et al. 2017.

6.2. Uptake by the targeted audience

As indicated in the Description of the Action, the audience for this deliverable is

X	The general public (PU)
	The project partners, including the Commission services (PP)
	A group specified by the consortium, including the Commission services (RE)
	This reports is confidential, only for members of the consortium, including the Commission services (CO)

All the developments included in this deliverable will be available in the next OASIS3-MCT_4.0 release planned in April or May 2018. This release and all improvements it includes will be announced to all OASIS users via the mailing list, suggesting them to upgrade their coupled system or build new ones with OASIS3-MCT_4.0.

7. The delivery is delayed

☐ Yes ☒ No

8. Changes made and/or difficulties encountered, if any

The deliverable was initially planned for 31/08/2017. Most of the developments were achieved by that time but for the OpenMP parallelisation of the SCRIP interpolation routines. The deliverable was then officially delayed until March 30th 2018 and no further difficulties were encountered to meet the new official delivery date.

9. Efforts for this deliverable

Person-months spent on this deliverable:

Beneficiary	**Person-months**	**Period covered**	**Names of scientists involved, including third parties (if appropriate)** **and their gender (f/m)**
CERFACS	12	01/03/2016 – 28/02/2018	Sophie Valcke (f), Laure Coquart (f), Eric Maisonnave (m) , Andrea Piacentini (m, funded outside ESiWACE2), Anthony Craig (m, funded outside ESiWACE2)
Total	**12**		

10. Sustainability

13.1 Lessons learnt: both positive and negative that can be drawn from the experiences of the work to date

The main positive point is that the interaction between the OASIS developers to produce OASIS3-MCT_4.0 was very motivating and successful and the continued future development of OASIS is therefore on good tracks. The main difficulty encountered was about the work around the SCRIP library. It was originally planned to “only” parallelise the SCRIP with OpenMP+MPI but its analysis proved to be much longer and more complex than planned. This analysis revealed many flaws of the library at least as

implemented in OASIS3-MCT. In the end much more effort than initially expected was devoted to the task but this builds solid bases for future revision and improvement of the implementation of the library.

13. 2 Links built with other deliverables, WPs, and synergies created with other projects

The EC-Earth model currently uses OASIS3-MCT_3.0. Like other coupled models in use, It should be upgraded to OASIS3-MCT_4.0 so to benefit from all the developments described in this document so that these are exploited for deliverable "D2.11 Implementation of EC-Earth 10km global coupled demonstrator and performance analysis" due at month 42.
Some ideas around these developments were discussed during the 4° HPC ESiWACE/ENES Workshop in Toulouse in April 2016 (deliverable D 1.4).
Many interactions with software developers took place also in the framework of the IS-ENES2 project, in particular during the Fourth Workshop on Coupling Technologies for Earth System Models Coupling workshop that took place in Princeton in March 2017, co-organised by IS-ENES2 and Princeton University.

11. Dissemination activities

Type of dissemination and communication activities	Number	Total funding amount	Type of audience reached In the context of all dissemination & communication activities	Estimated number of persons reached
Presentation "OASIS3-MCT, a coupling software for climate modelling" at the International 2016 Fall School on Terrestrial Modeling and High-Performance Scientific Computing, Bonn, Germany	1	No cost, invited talk	Students in High-Performance Scientific Computing	50
Presentation "Code coupling for climate modelling" at the CEMRACS 2016 Summer School, Marseille, France.	1	170	Students in High-Performance Scientific Computing	40
Total funding amount		170		

See 6.1 for future dissemination.

Intellectual property rights resulting from this deliverable

Not applicable.