

OASIS4 – a coupling software for next generation earth system modelling

R. Redler^{1,*}, S. Valcke², and H. Ritzdorf^{1,**}

¹NEC Research Laboratories, NEC Europe Ltd., Sankt Augustin, Germany

²CERFACS, Toulouse, France

* now at: Max-Planck-Institute for Meteorology, Hamburg, Germany

** now at: NEC Deutschland GmbH, Düsseldorf, Germany

Received: 19 June 2009 – Published in Geosci. Model Dev. Discuss.: 7 July 2009

Revised: 17 November 2009 – Accepted: 19 December 2009 – Published: 22 January 2010

Abstract. In this article we present a new version of the Ocean Atmosphere Sea Ice Soil coupling software (OASIS4). With this new fully parallel OASIS4 coupler we target the needs of Earth system modelling in its full complexity. The primary focus of this article is to describe the design of the OASIS4 software and how the coupling software drives the whole coupled model system ensuring the synchronization of the different component models. The application programmer interface (API) manages the coupling exchanges between arbitrary climate component models, as well as the input and output from and to files of each individual component. The OASIS4 Transformer instance performs the parallel interpolation and transfer of the coupling data between source and target model components. As a new core technology for the software, the fully parallel search algorithm of OASIS4 is described in detail. First benchmark results are discussed with simple test configurations to demonstrate the efficiency and scalability of the software when applied to Earth system model components. Typically the compute time needed to perform the search is in the order of a few seconds and is only weakly dependant on the grid size.

1 Introduction

Global coupled models (GCMs) have been used for climate simulations since the late 1960s starting with the pioneering work by Manabe and Bryan (1969). With advances in computing power, coupled models have been in use for century-long simulations since the late 1980s. Since that time, we

observe a rapid and continuous increase of activity in global coupled modelling as additional computer resources have become available. Climate models can be considered as multi-disciplinary or multi-physics software tools to simulate the interactions of the atmosphere, oceans, land surface, sea ice and other components of the climate system. They are used for a variety of purposes from studies of the dynamics of the weather and climate system to projections of future climate. These models can range from relatively simple to quite complex, that is from zero-dimensional models and Earth-system Models of Intermediate Complexity (EMIC) to complex 3-D Global Climate Models (GCM) and Regional Climate Models (RCM).

A GCM aims to describe geophysical flow by integrating a variety of fluid-dynamical, chemical, or even biological equations that are either derived directly from physical laws (for example Newton's law) or constructed by more empirical means. Classically, the two main constituents of a GCM are atmospheric and ocean circulation models. When coupled together (along with other components such as a sea ice model and a land model) an ocean-atmosphere coupled general circulation model forms the basis for a full climate model. A very recent trend in GCMs is to extend these traditional coupled climate models to become full Earth system models (ESM), by including further model components to calculate atmospheric chemistry, marine biology or a carbon cycle model (matter cycle in a more general sense) to more correctly simulate the interaction between the different sub-components. Typically, these different model components are developed independently by different research groups.

Numerical modelling in its full complexity is still in its infancy, primarily due to the immense complexity of the task and insufficient empirical knowledge of many aspects of climate related processes. This is not only the case for the



Correspondence to: R. Redler
(rene.redler@zmaw.de)

oceans where in particular the knowledge of the subsurface circulation is rather poor, but also for the hydrological cycle where precipitation, evaporation and the three-dimensional distribution of water vapour and clouds are insufficiently known. Despite these uncertainties the development of global climate models is one of the main unifying components in the World Climate Research Programme (WCRP) of the World Meteorological Organisation (WMO). The output of these models is considered as fundamental deliverables from the whole WCRP and provides the basis for the understanding and prediction of natural and human-made climate variations (CLIVAR Scientific Steering Group, 1998).

In the past, the important role of climate models has been addressed in a variety of research programmes on a global scale like the Climate Variability programme (CLIVAR) or the Coupled Model Intercomparison Project (CMIP) as one of the CLIVAR initiatives, while EuroCLIVAR, the Baltic Sea Experiment (BALTEX) or the North American Regional Climate Change Assessment Program (NARCCAP) focused on a regional scale. Most important, global climate models form one of the backbones of the Assessment Reports (AR) published by the Intergovernmental Panel of Climate Change (IPCC).

In addition to the model components used in an ESM, each community representing one of the physical components still feels a strong need for having separate stand-alone special-purpose versions of the individual components of the coupled climate models. These are used to investigate processes in the different subsystems and to test new physical parameterisations in controlled scenario runs. The existence of different research objectives and the need to estimate model uncertainty from model intercomparison (like for the IPCC AR) support the idea of a multi-component approach and the need for the interoperability of the different components. To achieve this interoperability, two approaches are nowadays considered: either some standard programming rules are followed by all component developer groups and the resulting components are integrated into a single application, or the component models independently developed remain separate applications and an external coupling software ensuring the lowest possible degree of interference in the component codes is used. As the first approach is almost impossible to apply when confronted with the heterogeneous development environment in Europe, we offer, with the OASIS4 coupler described in this paper, an implementation of the second approach.

With OASIS4, we continue the software development of the Ocean Atmosphere Sea Ice Soil (OASIS) coupling software which has its origins in the early nineties at the European Centre for Research and Advanced Training in Scientific Computing (CERFACS, Toulouse, France). With the emerging complexity of ESMs, their increasing size (with respect to the number of grid points to be treated) and higher degree of parallelism, the limitations of OASIS3 (Valcke,

2006) and its predecessors have become visible and the need has emerged to provide the ESM community with new fully parallel coupling software. The development of OASIS4 started during the Program for Integrated Earth System Modelling (PRISM) project funded by the European Commission from 2001 to 2004. PRISM (Valcke et al., 2007) was organized by the European Network for Earth System Modelling (ENES), and is currently continuing within the new Framework 7 Programme funded by the European Commission with the IS-ENES (Infrastructure for the European Network for Earth System modelling) project.

At the time when the OASIS4 development started, coupling software performing field transformation already existed, such as the Mesh based parallel Code Coupling Interface (MpCCI) (Joppich and Kürschner, 2006) or the Community Climate System Model (CCSM) Coupler 6 (Cpl6) (Buja and Craig, 2002). MpCCI is not available as source code, and for this particular reason the MpCCI software has not been accepted by the ESM community. With new versions of the MpCCI software its developers have abandoned some important aspects of parallelism. Compared to previous versions all grid information is now assembled in a single MpCCI coupler instance for performing the neighbourhood search and regridding. This approach makes it even less attractive for the usage within an ESM with a high degree of parallelism. The concept of CCSM and Cpl6 is very much similar to OASIS4. However, earlier versions of Cpl6 have targeted the specific needs of the CCSM and the coupler could only be executed within the complete CCSM. The Earth System Modelling Framework (ESMF) software environment (Hill et al., 2004) provides an implementation of the first interoperability approach described above. ESMF is a high-performance software infrastructure that can be used to build a complex application as a hierarchy of individual units, each unit having a coherent function, like modelling a physical phenomenon or managing the input and output (IO), and a standard calling interface. While an ESMF application, being more integrated, will most probably be more efficient compared to an OASIS4 coupled system, ESMF may require a deeper level of intervention in the application code. For example, ESMF requires that components be split into initialize, run, and finalize sections, with each callable as a subroutine. Necessary modifications of the software to achieve the full benefit from ESMF is described briefly by (Hill et al., 2004). In contrast to ESMF, the OASIS4 application programmer interface (API) can be integrated with only minor modifications in the original application code, and is for this particular reason more attractive to the European ESM community with its less centralised modelling efforts.

OASIS4 does not include any assumptions about the physical nature of the coupling fields. It may be considered a slight disadvantage that apart from providing some general arithmetic functions (like adding or multiplying with constants or time averaging) and regridding functionality any specific physically motivated pre- and post-processing

regarding the coupling of physical quantities remains under the full responsibility of the application code. In compensation, the flexibility of OASIS4 provides the ESM community with the ability to couple a wide range of ESM components.

With this article we address the questions we have received from the growing user community during the last couple of years. The originality of OASIS4 relies in its great flexibility which is explained in Sects. 2–4, where the software design ideas, the driving and configuration mechanisms and the application programmer interface to the communication library are detailed. Another new concept introduced with OASIS4 is the parallel 3-D neighbourhood search and regridding based on the geographical description of the process local domains (see Sects. 5 and 6). The data flow, in particular the common treatment of coupling and IO exchanges (a concept already available with OASIS3) is briefly described in Sect. 7.

We conclude the technical description with first performance measurements shown in Sect. 8 and outlook to ongoing and future work in Sect. 9.

2 General design and overview

The key concept behind the OASIS4 software is to provide a tool to the ESM community which is portable to any existing computing environment. Mainly this is achieved by adhering to programming standards that have emerged over time. For a complete list of systems and environments which are supported by OASIS4, the reader is invited to visit the OASIS4 Wiki page¹.

The OASIS4 software package provides the source code for a Driver-Transformer executable and a library (called PSMILe hereafter). At run-time, the OASIS4 Driver-Transformer and the component models remain separate (possibly parallel) executables. To communicate with the rest of the coupled system, each component model needs to be linked against the PSMILe. To be as user-friendly as possible, the PSMILe API includes a limited number of required function calls, at the same time supporting all typical ESM component data structures. While it is not designed to handle the component internal communication, the library completely manages the coupling data exchanges with other model components and the details of the I/O file access. In this article, we will use the term “coupled application” to describe the ensemble formed by the component models coupled through the OASIS4 Driver-Transformer.

The major task of OASIS4 is to handle the data exchange between ESM components. These exchanges are completely built upon the Message Passing Interface (MPI) (Snir et al., 1998; Gropp et al., 1998). The major justification for this decision is the fact that MPI has become a quasi standard to handle parallel applications. MPI libraries are available on

almost every parallel architecture, be it a vendor-specific implementation or provided as one of the many publicly available MPI packages. The complexity of MPI is nevertheless hidden behind the OASIS4 API.

In contrast to previous versions of OASIS, the coupled configuration has to be described by the user with the help of Extensible Markup Language² (XML) files. In order to read in this information, linking to an XML library is required. Similar to OASIS3, NetCDF (Rew and Davis, 1997) file IO is supported to optionally read and write forcing input, diagnostic output and coupling restart files; in this case, a NetCDF or a parallel NetCDF (Li et al., 2003) library needs to be provided. Last but not least, compilers conform to published Fortran90 and C language standards are required.

With the current version, bilinear, trilinear, cubic and nearest-neighbour parallel interpolation is provided for the most prominent grid types used in ESM components to map the source grid values to the target grid in a data exchange. Furthermore, 2-D conservative remapping is available which is usually the preferred way to regrid fluxes.

In particular, OASIS4 supports 2- and 3-D coupling between any combination of logically-rectangular, i.e. 2-D structured, or Gaussian Reduced grids in the horizontal longitude-latitude plane, each horizontal layer repeating itself at different vertical levels. OASIS4 is also able to handle the exchange of non-geographical data, i.e. data which are solely provided in an (i, j, k) index space not associated to any geographical domain. To properly handle non-geographical data, it is required that the pairs of source and target grids work over the same global index range, although the data can be partitioned differently on the source and target side.

3 The driving and configuration mechanisms

The OASIS4 Driver-Transformer executable can consist of one or more processes. During the initialisation, the root process of this instance acts as a Driver while during the integration of the coupled model the root and additional processes act as Transformer processes performing the regridding of the coupling fields. The Transformer task is further discussed in Sects. 6 and 8. In this section we focus on the Driver aspect.

OASIS4 supports two ways of starting a coupled application. With a complete implementation of the MPI2 standard, only the Driver processes have to be started by the user. All remaining physical components are then spawned by the OASIS4 Driver at run time. In this scenario, no extra programming work has to be invested in the component internal parallelisation as any original MPI communication will work as is. If the available MPI library does not support the MPI2 standard, all processes of the whole coupled

¹<https://oasistrac.cerfacs.fr/>

²<http://www.w3.org/XML/>

application will have to be started simultaneously in a “multiple program multiple data” (MPMD) mode; the disadvantage of this approach is that each component must use a specific MPI communicator for its own internal communication. This MPI communicator is created and provided by the OASIS4 software.

The whole coupled configuration is described with the help of XML files. XML is a simple, very flexible text format. Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. An XML document is simply a file which follows the XML format. The OASIS4 configuration XML files must be created by the coupled model developer. A Graphical User Interface (GUI) is currently being developed to facilitate the creation of those files.

The “Specific Coupling Configuration” (SCC) XML file defines the general characteristics of a coupled model run, for example the components to be coupled and the number of processes for each component. The Specific Model Input and Output Configuration (SMIOC) XML files (one for each component) describes the relations each component model will establish with the external environment through inputs and outputs for a particular run. In the SMIOC files, the user specifies the coupling exchanges, for example the source or target of each input or output field (either another component or a disk file), the exchange frequency, and the transformations to be performed by OASIS4 on this field.

The contents of the XML files are read and interpreted by the Driver, and information relevant for the physical components is communicated to the respective PSMILe and Transformer processes on case-by-case basis.

4 The application programmer interface

In this section, we briefly describe the design ideas of the OASIS4 API to the PSMILe and highlight the differences compared to OASIS3.

The API function calls (see Table 1) can be split into three different phases. During the initialisation phase, some basic initialisation routines must be called and the grid and exchange fields need to be specified to the PSMILe. The initialisation phase is concluded with a call to `PRISM.Enddef` (see Sect. 5). The second phase comprises the exchange of data further described in Sect. 7 while the third phase denotes the termination of the coupling. For a detailed description of the OASIS4 API, we refer the reader to the OASIS4 User Guide (Valcke and Redler, 2006).

4.1 General design

The general design follows several principles. First, the PSMILe API routines that are defined and implemented shall not be subject to modifications between the different versions

of the OASIS4 software. However, new routines may be added in the future to support new functionality. In addition, the PSMILe has been kept extendable to new types of coupling data and new types of grids by using Fortran90 function overloading. Some complexity of the API arises from the need to transfer not only the coupling data but also the associated metadata, i.e. mainly the information about the associated grid. We carefully selected the necessary arguments for each subroutine in order to keep the argument lists as short as possible. At the same time we kept the interfaces generic in order to avoid the introduction of another set of routine for every new configuration we have to support with the software.

Like in the MPI, the memory that is used for storing internal representations of various data objects is not directly accessible by the user and the objects can only be addressed indirectly by the component via their handle. Handles are of type integer and represent an index to an entry in a list of the respective objects. The object and its associated handle are significant only on the process where it is created.

Furthermore, the internal usage of MPI is completely hidden from the user, except when the available MPI library does not offer the MPI2 spawning functionality (which implies that all processes have to be started simultaneously in an MPMD mode, see Sect. 3). In this case, each component has to retrieve a special MPI communicator for its internal communication which is created by the PSMILe and accessible with a specific function call to `PRISM_Get.localcomm` (see Table 1); this call remains the only MPI-related call.

When using MPI for internal communication in parallelised components, the calling sequence of some MPI- and PRISM-related calls has to follow a certain rule which is sketched in Fig. 1. The function calls to `PRISM.Init` and `PRISM.Init.comp` can be used as a full substitute of `MPI.Init`, while the `PRISM.Terminate` can be used as a full substitute for `MPI.Finalize`.

4.2 Data exchanges

The PSMILe manages the coupling data flow between any two (possibly parallel) component models with a communication pattern completely hidden from the component codes, following a principle of “end-point” data exchange. When producing data, no assumption is made in the source component code concerning which other component will consume these data or whether they will be written to a file, and at which frequency; likewise, when asking for data, a target component does not know which other component model produces them or whether they are read in from a file. The target or the source (another component model or a file) for each field is defined by the user in the SMIOC XML file (see Sect. 3) and the coupling exchanges and/or the IO actions take place according to the user external specifications. The switch between the coupled mode and the forced mode is therefore totally transparent for the component model.

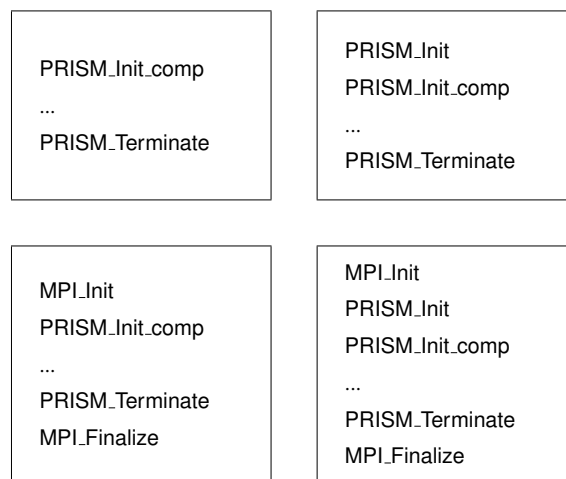
Table 1. List of OASIS4 PSMILe function calls. Description of the OASIS4 PSMILe Function Calls.

Name	Description
Initialisation	
PRISM.Init	Initialisation of the application
PRISM.Init_comp	Initialisation of a component within an application
PRISM.Def_grid	Abstract definition of a compute grid
PRISM.Def_partition	Definition of the process-local grid partition relative to the global compute domain
PRISM.Reducedgrid_map	Additional information needed for Gauss-reduced grids
PRISM.Set_corners	Definition of geographical positions of vertices of grid cells
PRISM.Set_mask	Definition of masked points or cells
PRISM.Set_points	Definition of geographical positions of points
PRISM.Def_var	Definition of exchange and IO fields
PRISM.Enddef	End of the definition phase
Data Exchange	
PRISM.Get	To receive data from a remote component and/or read in from a file
PRISM.Put	To send data to a remote component and/or write into a file
PRISM.Put_restart	To write additional data sets into an OASIS4 coupling restart file
Termination	
PRISM.Terminate	Termination of the prism part
Auxiliary and Query Functions	
PRISM.Calc_newdate	Add and Subtract operations on the OASIS4 Fortran90 Time structure
PRISM.Get_localcomm	To retrieve a MPI communicator for component local communication
PRISM.Initialized	To check whether PRISM.Init has already been called
PRISM.Put_inquire	A query function to obtain information whether a PRISM.Put has to be posted
PRISM.Terminated	Query function to check whether PRISM.Terminate has already been called
PRISM.Abort	To terminate the coupled application in a well defined to support user-defined error handling

The sending and receiving PSMILe PRISM.Put- and PRISM.Get calls can be placed anywhere in the source and target code and possibly at different locations for the different coupling fields. These routines can be called by the model at each timestep. The actual date at which the call is performed and the date bounds for which it is valid are given as arguments; the sending/receiving is effectively performed only if the date and date bounds correspond to a time at which it should be activated, given the field coupling or IO dates indicated by the user in the SMIOC; a change in the coupling or IO dates is therefore also totally transparent for the component model itself.

When the action is activated at a coupling or IO date, each process sends or receives only its local partition of the data, corresponding to its local grid defined previously. The coupling exchange, including data repartitioning if needed, then occurs, either directly between the component models, or via additional Transformer processes if regridding is needed.

If the user specifies that the source of a PRISM.Get or the target of a PRISM.Put is a disk file, the PSMILe exploits the GFDL mpp_io package (Balaji, 2001) for its file IO (see Sect. 7 for more detail).

**Fig. 1.** The four allowed sequences of calls for initializing and terminating an ESM component for coupling with OASIS4 and its relation to MPI.

4.3 OASIS3 and OASIS4 APIs

The major difference between the OASIS3 and OASIS4 APIs from a user point view is in the initialisation phase. In OASIS3, the global grid information has to be provided in separate OASIS3 specific NetCDF grid files which are accessed by OASIS3 routines. With OASIS4 it is assumed that each component has all knowledge about its own local geographical grids at some point during the initialisation phase. This local grid information has to be provided through the API to initialise the PSMILe search.

Beside this difference, we have kept the design of the OASIS3 and OASIS4 PSMILe API as similar as possible, both following in particular the principle of “end-point” data exchange explained above, in order to allow an easy transition from OASIS3 to OASIS4 for the community of users.

5 Neighbourhood search

The PSMILe library performs the exchanges of coupling data between source and target components. Usually those components express their coupling fields on different numerical grids and a regridding operation has to be performed on the coupling data. The regridding implies: 1-identifying the “neighbours” of each target point, i.e. the source grid points that will contribute to the calculation of the target grid point value in the regridding process (in this article, we use the term “neighbourhood search” to describe this process), 2-calculating the weights of the different neighbours and 3-performing the calculation of the grid point values. To minimize the transfer of source data, it was decided to perform the neighbour search (1) in the source PSMILe and to transfer only the useful source grid points (see Sect. 5.1 for further details) to the Transformer which performs the regridding calculation per se (i.e. 2 and 3). In this section, we describe the neighbourhood search performed by the source PSMILe.

The goal here is to provide a highly efficient search algorithm which determines the neighbourhood relations between source and target grid points without generating significant overhead with respect to the model compute time. For ESMs the typical elapsed time for a job is in the order of hours and the search should not take more time than a few seconds. At the same time the algorithm has to rely on only a very limited number of a-priori assumptions in order to be flexible and applicable to more than just a few special grid configurations.

Compared to typical physical algorithms for numerical models, the result or the output of a search algorithm is already well determined when the problem is posed, i.e. when the numerical grids are defined. For any given regridding scheme a unique solution exists regarding the neighbourhood between target and source points. When programming a library for such purposes, the task is to have flexible algo-

gorithms that are able to deal with any peculiarities of a given grid configuration. While an application programmer typically has a complete control over the local grid, this is not the case for the library programmer as the library has to work based on the information provided through the user API without any or with only very limited a-priori assumptions.

In order to initialise the search, the components have to provide geographical information about the position of the grid points and about the reference volume or area that is associated with the grid points. This information is stored in one-dimensional arrays together with the grid type dependent shape of the original multidimensional array. The grid type and the information about the array shape provides the implicit knowledge about the connectivity of the grid points within a single block. Within these blocks correct neighbours have to be identified for each target point. In a simple implementation of such search algorithm, this task can be achieved by comparing the distances for each target point with all source points available. This approach will be of the order of N^2 w.r.t. to the compute time (N being the number of target points to be processed, assuming that source and target grids are about the same size). While such an approach may still be justified for small problem sizes, this approach will fail when the horizontal resolution is increased and grids become larger in size. For OASIS4, we have chosen a hierarchical approach in order to minimise the workload during the search. We briefly describe here key aspects of the search algorithm which ensure its efficiency. An additional difficulty arises with parallelised components where the source grid is partitioned. For selected target points, the required “neighbour” source locations may reside on different processes and the search has to be extended across process boundaries. In the remaining part of this section we describe the different tasks which are performed by the PRISM_Enddef call.

5.1 Point-based search for block-structured grids

The completion of the definition of grids and coupling fields by the component is announced in the component code with a call to the PRISM_Enddef routine (end of definition) on each process. At this stage, the coupling information from the XML configuration files about which components have to exchange data with each other is already available in the PSMILe (see Sect. 3). In an initial step, each coupled component process determines the envelope of its locally defined grid partition. The envelopes are exchanged between those component processes that have to exchange data with each other. Each component process now has a global view on grid partitions and processes with which it may have a coupling interface. As a first step towards an efficient search, regions outside the envelop intersections between source and target local grids are cut off. Pairs of source and target processes that have common intersections are identified. For each of these intersections, lists of target grid points included in the intersection are generated on the target side. Each list

is transmitted to the respective source process which shares a particular intersection. We note here that only the envelopes of all partitions are stored on all processes. For this and all following steps in the PSMILe or in the Transformer, it is never required to gather the global grid information at a central place onto one single process. This design aspect allows a minimal use of extra memory for the coupling operations.

The task of the second stage of the search is to find a reference source point for each target point which can later be used as a starting point to build the required interpolation stencil, i.e. the set of source neighbour points used for the calculation of the target point value, or to locate additional neighbours based on the user choice.

On the source side, a grid hierarchy is established out of the local source grid. For block-structured grids, this grid hierarchy is constructed by splitting the local source grid into smaller subsections with a refinement factor of two similar to a multi-grid approach (see Fig. 2). The subsection on the highest (finest) level in the grid hierarchy typically contains three grid points in each direction. For each target grid point in the list, we first determine the containing subsection on the lowest (coarsest) level by considering the bounding boxes of the subsections on this level. These bounding boxes are constructed by evaluating the minimum and maximum cell extent in a given i - j - k index range. This containing box is then subdivided into four subsections (or eight subsections in a 3-D search) on the next higher level, and those four or eight boxes are investigated in the same way. The process is continued until we have reached the highest level and are close to the required source point.

We note here that due to this simple approach based on bounding boxes, it is possible for any target point to be located in a region covered by more than one bounding box. We start the search in one of these boxes moving up to finer levels. The higher the levels are in our hierarchy the closer we can approximate the exact geometry of the local domain. It may happen that the algorithm does not find any appropriate bounding box on a higher level. In such a case the algorithm jumps back to the previous lower level and continues with another bounding box. These cycles are repeated until either a containing source cell is identified or until all bounding boxes under consideration have been investigated. At this stage, the search on the global domain is completed.

A great advantage of the multi-grid algorithm is that it is only weakly dependent on the problem size. When the grid size is doubled in each horizontal direction this would only introduce one more multi-grid level. For grids commonly used in climate modelling and for higher resolution grids, the overhead to set up the multi-grid hierarchy is more than compensated by the speedy search. OASIS4 is thus able to handle very large problem sizes (for example a global grid with $1/12^\circ$ horizontal resolution) in a reasonable amount of time. The performance of the OASIS4 multi-grid search is further discussed and compared with the performance of a classical search in Sect. 8.4.

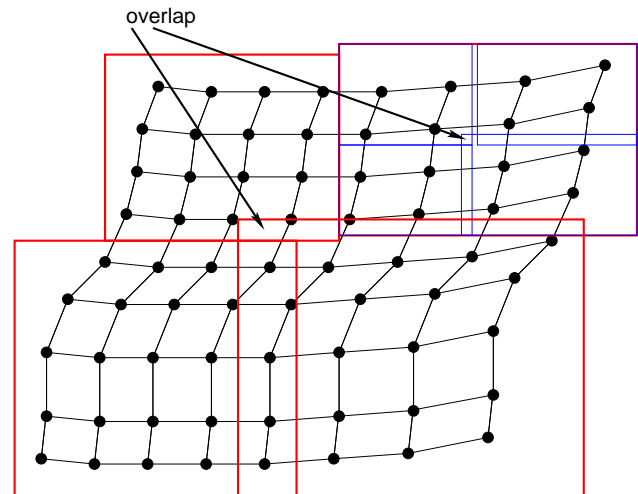


Fig. 2. Sketch of a multi-grid hierarchy for a local domain. Typically the number of cells is much higher and the PSMILe will construct $o(10)$ levels.

Now that the containing source cell is identified, the next step is to determine the reference source point. The source points are used to construct an auxiliary cell grid where the source points serve as corners of the cells (see dashed and solid lines in Fig. 3). The task is then to locate the source-point cell which contains the projection of the respective target point. This is performed with a local search by investigating each of the four source-point cells that surround the point which belongs to the containing source cell. In this example, the source-point box built with points at indices $[i, j]$, $[i+1, j]$, $[i, j+1]$, and $[i+1, j+1]$ is chosen and the source-cell of the point $[i+1, j+1]$ (green box in Fig. 3) is found. This immediately directs us to identify the point with index $[i, j]$ as the “lower-left” source-point. This location is now stored as the reference source point for the particular target point.

In the final step, the remaining source neighbours are identified which are required to perform the interpolation requested by the user. For a bilinear interpolation, we are now able to identify the four corner points of the auxiliary cell (black open and solid bullets connected by solid lines) as the four neighbour source points needed. For a bicubic interpolation, sixteen neighbours have to be identified (the aforementioned bilinear points plus the open blue circles shown in Fig. 3).

Typically, block-structured grids include masked points in their compute domain. The algorithm as it is described so far does not distinguish between source masked and non-masked points but solely works on all grid coordinates in the source grid compute domain. For some target points, some source points first identified as neighbour points may not be available for the interpolation if they are masked out which means that the physical fields do not necessarily contain meaningful values on those points. In such cases the user has different

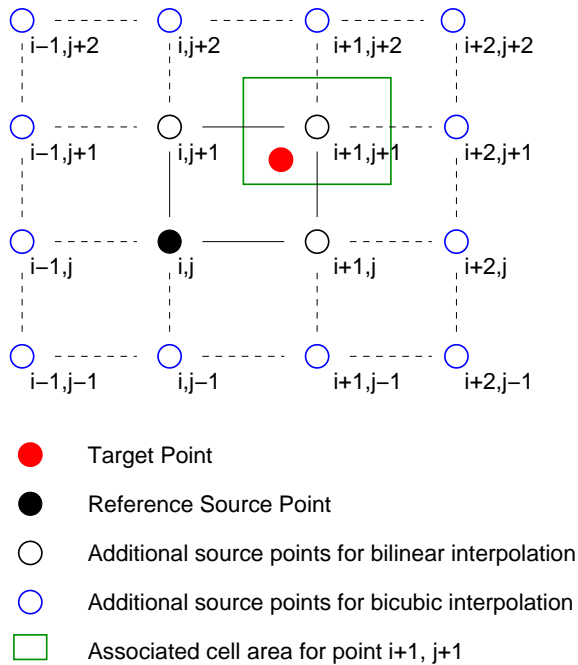


Fig. 3. Building the interpolation star based on the initial location.

choices via the XML configuration file. As a default option, the remaining valid points will be used for a weighted-distance interpolation. Another option available is to not interpolate onto such target points. In the extreme case when all neighbour source points identified by the default search are masked, OASIS4 offers the possibility to search for the non-masked nearest-neighbour value and use this value for the target point. For this extra nearest-neighbour search, the algorithm starts from the original masked source grid point taken as an initial guess. Starting from the finest level containing the initial guess, the multi-grid hierarchy is then used again to locate the closest non-masked source points by moving in so-called *w*-cycles through the multi-grid hierarchy, that is going down to a coarser level and going up again in another branch of the hierarchy repeatedly until the closest point is found.

5.2 Search on domain-partitioned source grids

On modern parallel systems, the application domain is usually distributed to many processes and/or processors. If the interpolation of fields was performed using locally available data only, i.e. data which is available on the actual process, the result of the interpolation would depend on the actual partitioning. For example, if a target point is located close to the internal border of a partitioned domain of the source application, the standard interpolation stencil may require source points which are not located on the actual process but

which are located on a neighbour process. The PSMILe current search, called “global” search in the following, considers remote neighbour points appropriately, and therefore ensures that the interpolation is invariant to the domain partitioning.

The “global” search ensures that required data from source neighbour points located on remote processes is provided together with local source data to the OASIS4 Transformer. Since the definition of application grids is quite general and since the full connectivity information for partitioned grids is not provided to the OASIS4 library (and is not explicitly available in most of today’s climate model component codes), missing source points for interpolation stencils require an additional search step. In this additional search step, the missing source points for the interpolation are searched in remote neighbouring domains. For block-structured grids, remote neighbouring domains are identified by exchanging information about the global index space provided through the `PRISM_Def_Partition` function call. If these remote source neighbour points are found, the full information including the mask data are returned to the process which has initiated this additional search step. At run-time, this process will collect the data from possibly different processes and will forward the data for the full interpolation to the Transformer.

5.3 Point-based search for Gauss-reduced grids

Gauss-reduced grids have been introduced mainly for atmospheric models, in particular the ARPEGE and IFS models (see for example Déqué et al., 1994), to overcome problems resulting from the convergence of meridians close to the geographical poles. As these models and grids play an important role for the atmospheric modelling community, the decision has been made to support this type of grid as a special case. These grids cannot be classified as block-structured grids, nor are they completely unstructured. Instead of treating them as unstructured grids, we are in this particular case able to make use of some implicit knowledge about how the grid is constructed and introduce an intermediate step. In order to reuse as many of the existing functions as possible, we set up a block-structured auxiliary grid which is regular in longitude and latitude and determine the corresponding mapping of the Gauss-reduced grid points onto this auxiliary grid. We are now able to perform the initial search on the auxiliary grid and derive initial locations. By mapping them back onto the Gauss-reduced grid, we obtain the correct start location to continue the local search on the Gauss-reduced source grid in a similar fashion as it has already been described for block-structured grids. In contrast to block-structured grids, the connectivity cannot be derived from the *i*–*j* indices directly. Instead we have to determine the connectivity among the grid points stored in an 1-D array in order to proceed with the local search; this is done based on the description of the Gauss-reduced grid provided by calling the routine `PRISM_Reducedgrid_Map` in the component code.

5.4 Cell-based search

Compared to the search described above for point-based interpolation schemes, the cell-based search has to follow a slightly different approach. In order to determine all necessary information for conservative remapping, the task here is to locate, for each target cell, its overlapping source cells. In order to again reuse as much functionality as possible, we perform a point-based search on the corner coordinates in a very similar way to the search described above in order to obtain the start location. The indices that we obtain for the start location refer to the data structure of the corner points. For block-structured grids, these indices can easily be mapped to cell indices. The cell indices point us to the location of the geographical information of the cell corners (or vertices of the element). For each target cell, we can now use this initial source cell to continue the local search and identify the remaining source cells which overlap a given target cell (Fig. 4). The intersection or overlap between a pair of source and target cells is determined by a pairwise investigation of the intersection of the edges without a detailed evaluation of the exact area of overlap. The numbering shown in Fig. 4 indicates the order in which the local search is performed on the source grid. In this particular case, 21 cells are investigated starting with the initial source cell 1. For each of the source cells, the overlap with the target cell is checked. Source cells with no overlap are marked as visited. Source cells that overlap with the target cell are marked and stored in memory. For an individual local search, the path way through the source cells is not fixed but dependent on the local problem.

5.5 Optimisation

Most grids used in Earth system modelling are block-structured grids. Historically, codes have been developed for stand alone models with non-global grids in mind. Particular issues regarding the connectivity are handled internal to the code to the extent it is needed for the advection and diffusion operators and other boundary conditions. For global grids so-called cyclic boundary conditions have been introduced to guarantee a seamless continuation of the grid in the zonal direction. When approaching the geographical poles additional ghost-cells are introduced in order to properly calculate the operators along the zonal boundaries (see for example Madec and Imbard, 1996). When considering the i - j domain in the case of coupling, a target cell close to the pole may extend further than what is covered by the source ghost-cells in the i - j space. As the codes lack the information about the connectivity at the poles, in general, an additional search in the geographical domain is necessary. Not all of this implicit and sometime configuration specific information is transparent for an external software package like the coupler. A new interface routine could be added to the API to transfer to the PSMILe any explicit connectivity information the appli-

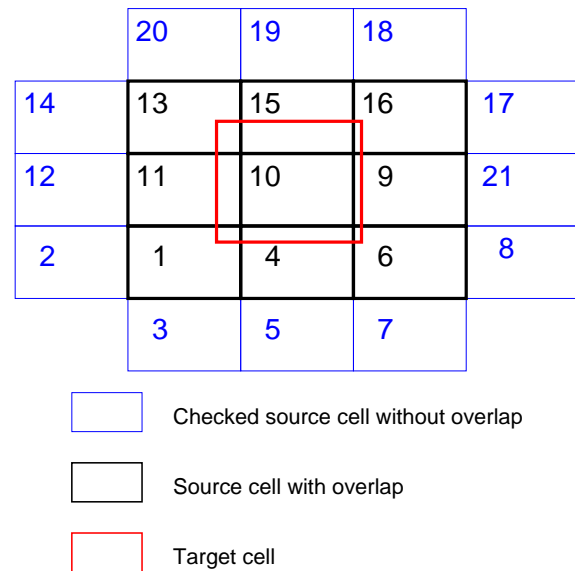


Fig. 4. Search of source cells for conservative remapping. Numbers indicate the sequence in which the source cell are investigated locally.

cations would be able to provide. This would help to increase the efficiency of the search locally and across process boundaries beyond what is achieved already today.

The OASIS4 PSMILe search algorithms are designed to work on 3-D model domains. In general, one could think about treating all grids as 3-D fully irregular or unstructured grids. In this way, one would ignore important a-priori information useful to speed up the search process. In order to take advantage of the existing a-priori knowledge (in particular about regularity along any of the grid axis), different grid types are introduced. The different vertical layers of grids with vertical z -coordinates are usually a simple vertical translation of the same 2-D horizontal grid. In such cases, the search is split into two parts with a 2-D search in the horizontal plane and a 1-D search along the vertical axis. In the horizontal plane, if the different columns or rows of the grid are respectively just a longitudinal or latitudinal translation of one same column or row, the search can be conducted by performing a 1-D search along each axis which further reduces the compute time.

It is obvious that individual source points or cells are needed for the calculation of more than just one target grid point or cell. In order to minimize the amount of data to be transferred and stored, redundant geographical information (usually of type double precision or real) is removed and an index list is created to guarantee a well defined and correct access to the geographical information. In order to allow the scattering onto the final n -dimensional destination array during the data exchange, these access index lists are also transmitted to the target process.

6 PSMILe-Transformer interaction and regridding

As outlined further in Sect. 7, the PSMILe supports two different ways of exchanging data, directly between two components or through the Transformer processes when regridding is required. Here we concentrate on the latter case and describe the transfer of information between the PSMILe and the Transformer and the regridding per se done on the coupling fields.

As the final result of the PSMILe search described in the last section, each source process holds 1-D lists, each list containing the geographical information relevant for the regridding of an intersection of the local source grid domain with one target process grid domain. As stated above, access index lists are created along with the raw geographical information. Hereafter, we will call these lists “intersection regridding lists”. These intersection regridding lists are equally distributed over the Transformer processes available. This ensures that the Transformer processes work in parallel over the regridding of the different source and target process intersections.

During the run, the different Transformer processes wait for messages to arrive from any component process PSMILe, receive header messages containing information about the next message to receive or pending requests to fulfill, and perform related actions. They repeat this sequence of actions in an indefinite loop until the entire coupled run is finished. During the exchange phase, each Transformer process receives the grid point field values (transferred from the source component code with a PRISM.Put call) corresponding to its lists, calculates the regridding weights if it is the first exchange, and applies the weights. The resulting target values are made available to the corresponding target PSMILe process. The data are sent upon request from the respective target process (i.e. when a PRISM.Get is called in the target component code).

The calculation of the weights depends on the regridding algorithm chosen by the user in the XML configuration file which can be different for each coupling field. The current version of OASIS4 supports purely 3-D regridding (3-D n-neighbour distance-weighted average or trilinear interpolation) and 2-D regridding in the horizontal plane (2-D n-neighbour distance-weighted average, bilinear, bicubic interpolation or conservative remapping) followed by a linear interpolation in the vertical. As in OASIS3, the 2-D algorithms are taken from the Spherical Coordinate Remapping and Interpolation Package (SCRIP) library (see Jones, 1999). In particular, for the 2-D conservative remapping, the contribution of each source cell is proportional to the fraction of the target cell it intersects; this ensures local conservation of extensive properties such as fluxes. The 3-D algorithms are 3-D extensions of the 2-D SCRIP algorithms. For a detailed description of the regridding the reader is therefore referred to Jones (1999) and to the SCRIP User’s Guide (see Jones, 2001). As the SCRIP bicubic algorithm is based on the func-

tion gradient in the i and j directions, it cannot be applied for the Gauss-reduced grids. For those grids, the SCRIP functionality has therefore been extended by a standard bicubic algorithm based on the 16 source neighbours.

Analysing the quality of the regridding algorithm is a delicate issue and no firm number can be given as the results depend on the source and target grids and on the characteristics of the regridded field. As an indication, we show here the error obtained when regridding the values of an analytical cosine bell function which presents one minimum and one maximum over the global Earth domain. For longitude λ and latitude ϕ this is expressed as

$$F_1 = 2 - \cos[\pi \cdot \text{acos}(\cos(\lambda)\cos(\phi))]$$

The relative error, i.e. the difference between the regridded value and the analytical value of the function divided by the analytical value, is calculated on each target grid point. The Figs. 5–8 present the results with each pixel showing exactly this relative error for each target grid point. No projection in the latitude-longitude space has been performed to avoid any additional interpolation by the graphic package.

Figure 5 shows the relative error obtained with the bilinear interpolation from the LMDz grid to the ORCA2 grid. LMDz (Hourdin et al., 2006) has a regular $3.75^\circ \times 2.535^\circ$ grid. ORCA2 (Madec, 2008) is based on a 2° Mercator mesh, (i.e. variation of meridian scale factor by cosine of the latitude); in the Northern Hemisphere the mesh has two poles so that the ratio of anisotropy is nearly one everywhere. The error is not shown for masked target points (grey areas). One can observe a relative error smaller than 0.4% over most of the domain. Near the coastline, for example near the west coast of the northern South America, the error can reach 0.8% at maximum. For those points near the coastline, some source points that are identified as bilinear neighbours are masked out. The remaining (non-masked) valid points are used for a less precise weighted-distance interpolation resulting in a larger error.

Figure 6 shows the relative error obtained with the bicubic interpolation from a BT42 Gauss-reduced grid (with 6232 grid points in total distributed over 64 longitudinal circles) to the ORCA2 grid. The relative error in the equatorial and tropical regions of the basins is smaller than 0.4%, and growing slightly but not above 0.8% at higher or lower latitudes which corresponds to regions where the Gaussian grid is effectively reduced. Again, the error is larger near the coastline, reaching 4.2% for some points; like the bilinear interpolation, only the non-masked source neighbours are used for those points near the coastline resulting in a less precise weighted-distance interpolation. The error near the coastline is larger in this regridding involving the BT42 grid than in the regridding involving the LMDz grid described above; this is linked to the fact that the BT42 grid has a much wider masked domain compared to the LMDZ domain, and not to the interpolation algorithm.

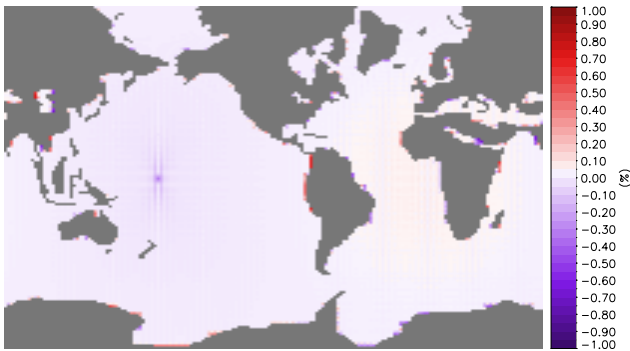


Fig. 5. Relative error obtained (in %) with the bilinear regridding from the LMDz grid to the ORCA2 grid.

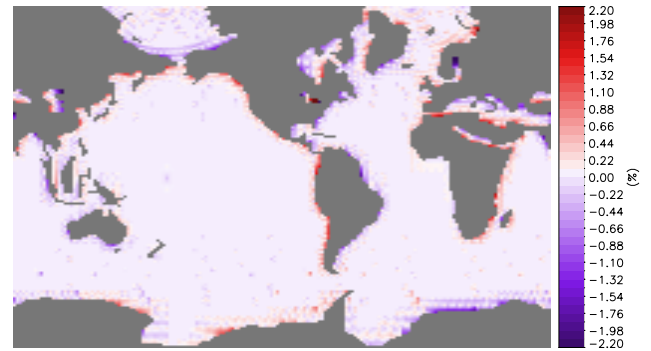


Fig. 7. Relative error obtained (in %) with the conservative remapping from the ORCA2 grid to the LMDz.

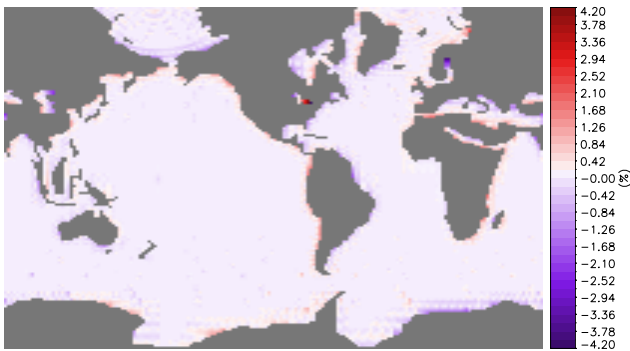


Fig. 6. Relative error obtained (in %) with the bicubic regridding from the Gauss-reduced grid to the ORCA2 grid.

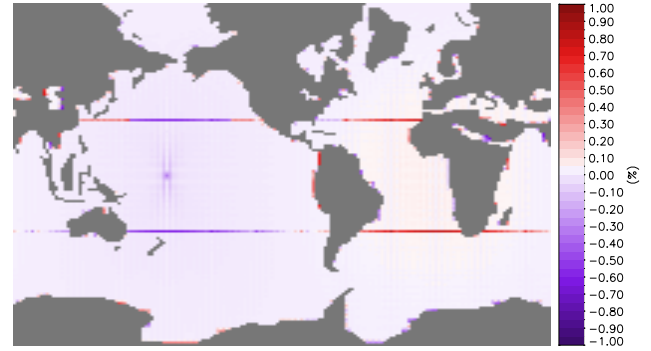


Fig. 8. Relative error obtained (in %) with the bilinear regridding from the LMDz grid to the ORCA2 grid when each component is partitioned over 3 processes and when the global search is not activated.

Figure 7 shows the relative error obtained with the conservative remapping from the ORCA2 grid to the LMDz grid. The relative error is everywhere smaller than 0.2% except for the last row of the LMDz grid close to the North pole and near the coastline where it can reach respectively 1% and 2.2%. The SCRIP library assumes that the border of a cell between two cell corners is linear in longitude and latitude: near the North pole, this assumption becomes less valid and this can explain the larger error. Near the coastline, different normalisation options are available in the SCRIP library to calculate the value of the target cells which partially overlap masked source cells (see Jones, 2001). We chose the option to normalize by the area of the target cell intersecting non-masked source cells; this option does not locally conserve extensive properties but results in more realistic target values even if the error is still relatively large. With the alternative option, the normalisation by the full area of the target cell, the relative error (not shown) can become as large as 100%.

We note here that the data structure provided to the OASIS4 Transformer differs from the data structure sent to the OASIS3 Transformer. The source points that are used for a particular interpolation stencil may arrive in a different ordering. The data are multiplied with their individual weights and summed up. The different ordering during the

summation may lead to truncation errors which become visible when comparing the results obtained with OASIS3 and with OASIS4 even though both software use the same regridding algorithms.

Finally, as an illustration of the benefits of the global search (see Sect. 5.2), Fig. 8 shows the relative error obtained with the bilinear interpolation from the LMDz grid to the ORCA2 grid when each component is partitioned over 3 processes (in the longitudinal direction) and when the global search is not activated. Comparing with Fig. 5, we observe greater relative error (of the order of 1% comparable to the maximum error obtained near the coastline) near the borders of the source partitions. This can be expected as for those target points the local search cannot find the usual 4 surrounding neighbours needed for the bilinear interpolation (as some of those 4 neighbours are located on a neighbour process domain). The remaining neighbours are used for a less precise weighted-distance interpolation resulting in a larger error. When the interpolation is performed with the global search described in Sect. 5.2, this greater relative error vanishes and regrided results (not shown) are exactly the same in the partitioned and not partitioned cases, which validates the global search.

7 Data exchange

The exchange of coupling data is typically happening inside a loop over time or iterations and is activated many times during the execution of the coupled model. Therefore the handling of the data exchange is a time-critical task. One of the major targets is to minimise the communication overhead and to avoid load-balancing problems induced by the implementation of the exchange.

The data exchange is invoked by the component models using the PRISM_Put and PRISM_Get interface routines. These interface routines can be called individually by each model component at an arbitrary non-constant frequency. The coupling frequency is determined in the XML file at the beginning of the job. Inactive calls will return without performing any MPI calls. Only active prism calls (calls that are performed for a date at which an exchange has to occur) will call the MPI library to exchange data. OASIS4 verifies that the successive date bound arguments to these calls cover the run duration with no overlap and no gap and therefore takes care that the active calls have a one to one correspondence.

As described in Sect. 4.2, the routines can be used to read from and write to NetCDF files as well as sending data to and receiving data from another model component. The arguments of the subroutine calls are identical for IO and coupling operations, and the concrete action(s) are only specified in the component XML configuration file.

Figure 9 schematically illustrates possible communication pathways in a coupled application including the access of data from files. Depending on the respective grid configuration, the exchange of coupling data is handled in two different ways. Data points that reside on identical geographical locations on the source and target side are exchanged directly between the component processes (involving repartitioning if necessary), without the participation of Transformer processes. Data points for which a regridding is required are sent through the Transformer. Matching and non-matching grid points are identified by the PSMILe solely based on the geographical description of the grid. In order to reduce the amount of messages to be exchanged, the current implementation assumes that the number of matching points must at least sum up to 10% of the total number of grid points to be sent. For any number below this threshold, matching data are exchanged together with non-matching data through the Transformer processes.

The data transfers, either direct or through the Transformer, are handled in parallel, in the sense that the source and grid point values corresponding to the different intersection regridding lists (see Sect. 6) are transferred separately. Gathering or scattering is never performed, and all component processes that are active in the coupling participate in the exchange. One Transformer process may treat several intersection regriddings when the number of source and target domain intersections exceeds the number of Transformer processes.

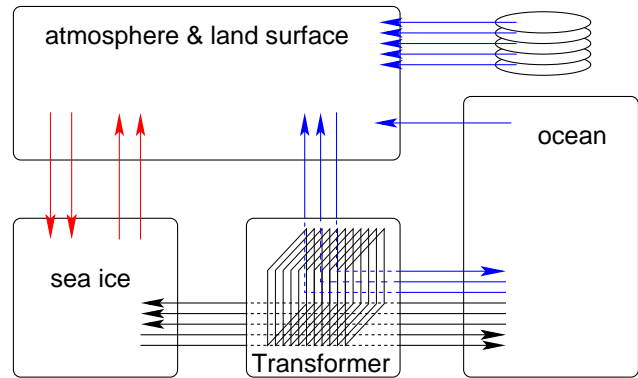


Fig. 9. Example of communication pathways in an OASIS4 coupled application.

To further optimise the exchange of data and to reduce the work load of the Transformer processes, only non-masked source and target points are transferred. On the receiver side, the non-masked data are scattered onto the destination array behind the PRISM_Get call and provided to the API in the appropriate array shape leaving masked target points untouched. The calculation of regridding weights in the Transformer is delayed until the first exchange for which a particular set of weights is required. The weights are then stored in the Transformer for subsequent exchanges.

It is noteworthy to mention that the data are buffered in the PSMILe library on the sender side outside the MPI library. Data are transferred with a so-called non-blocking MPI function (MPI_Isend). With a standard conform MPI implementation, this guarantees no overflow in MPI internal buffers. The PRISM_Put will immediately return when the data are stored locally even if the destination process (be it the Transformer or another component in case of direct communication) is not yet ready to receive data. On the sender side the user code outside the PSMILe can safely progress and reuse its own local memory. The PRISM_Get on the other hand will only return when the data are received.

The IO layer is based on the mpp.io package as described by Balaji (2001). For the OASIS4 PSMILe IO, various modes are supported which become especially relevant if components are parallelised. The default behaviour of the PSMILe is, for output, to collect all data on the component root process and write out data into a single file or, for input, to read data on the component root process and distribute it internally. While this approach is still applicable when the domain partitioning is changing from job to job (especially when thinking about coupling restart files) it may not be the most efficient way to deal with the IO. As an alternative approach, all IO processes read data from and write data to individual files. For further analysis these files have to be glued together in a postprocess. In order to support a full parallel IO, the mpp.io package has been extended to be

compliant with the pNetCDF library (Li et al., 2003). When linked against pNetCDF, data are written into and read from a single file via MPI-IO with no pre- or postprocessing required. From a user perspective, this approach is invariant to changes in domain partitioning and should also provide a reasonable IO performance provided that the respective MPI-IO implementation efficiently uses the underlying system architecture (memory layout and file system).

During the data exchange and IO, the PSMILe is able to invoke some local transformation routines like time-accumulation, time-averaging, gathering and scattering of physical fields. It performs the required transformation locally on the source side before the exchange with other components of the coupled system or IO. In this case only the end product is communicated to the target.

8 Performance

Before seriously considering a new software for a coupled system a user typically asks for the cost of coupling and the overhead one has to consider for the coupling. Preferably, a user would like to have exact numbers at hand in terms of CPU seconds or a percentage of time relative to a non-coupled system. For full ESMs, it is difficult to predict the additional time needed for the coupling. To some degree, this depends on the search that has to be performed on a variety of grids for a larger number of exchange fields and different regriddings. Another important aspect is the run-time behaviour of the different ESM components and the load-(im)balancing between the processes involved. This may strongly influence the time needed to perform a data exchange. ESMs differ from each other in the above mentioned aspects, and providing a number for a particular ESM is of little help and not in the scope of this article. In addition, the picture may again change completely when an ESM job is running in batch mode and eventually has to share some resource with competing applications. In the following, our aim is to give a first idea about the potential cost for coupling and to provide information about the efficiency of the coupling software as such. Therefore we try to exclude the aforementioned side effects as much as possible. We will revisit this subject in our discussion in Sect. 9. We now use a simplified test case application to demonstrate the performance of the OASIS4 search algorithm and the data exchange and compare them with OASIS3 performance.

8.1 Benchmark design

Here we investigate the so-called “strong scaling” properties of OASIS4. By keeping the global problem size constant, we expect a decrease in CPU time for individual processes with the problem distributed to larger number of processes in proportion to the local problem size. While this is in general reflecting the typical situation the user is confronted

with, strong scaling is considered more difficult to achieve (in contrast to weak scaling where the local problem size is kept constant and the global problem to be solved is increased with increasing numbers of processes used). The test case we are considering now is a bidirectional exchange of 2-dimensional data between a global “atmospheric” grid for component A and an “ocean” grid for component B, ranging from 0° to 360° in longitude and latitudes ϕ between 70° S and 70° N. Both grids are described as horizontally irregular grids in order to bypass the optimised routines that are available for completely regular grids and to mimic the large class of use cases, the 2-D data exchange between any two physical components at the Earth surface. Some of the ocean and atmosphere grid points have been masked out in order to bypass additional optimisation steps of the search algorithm which can be applied if complete non-masked grids are treated. The components are partitioned in latitude direction. The search is performed for bilinear interpolation without requesting an extra nearest-neighbour search for target points that only have masked “bilinear” source points (see Sect. 5.1). The search is performed for one exchange field for each direction.

In Sect. 8.2, we consider a T255 “atmospheric” grid with 768×385 grid points for component A and an “ocean” grid with $0.3^\circ \times \cos\phi$ horizontal resolution corresponding to 1202×665 grid points for component B. In an additional test described in Sect. 8.3, we have expanded the problem into the vertical dimension towards a full 3-D search and data exchange with 40 levels for component A and 45 levels for component B. In Sect. 8.4, the resolutions of component A and component B are increased respectively from T21 to T255 and from $4.0^\circ \times \cos\phi$ to $0.5^\circ \times \cos\phi$.

8.2 2-D search and exchange

For this particular benchmark calculations and for the 3-D cases described in Sect. 8.3, the OASIS4 sources have been compiled with the Intel Fortran compiler version 10.1 and the GNU C compiler version 4.1.2, both using default compiler switches without any further optimisation. The code has been run on a local PC cluster. Each node of the cluster is equipped with 2.0 GHz 2 times single core AMD Opteron processor 246 and 4 GB of memory per node connected via a 2 Gigabit Myrinet. For the message passing, we use the Message Passing Interface Chameleon Glenn’s Messages proprietary communication layer (MPICH-GM) provided by Myri-com. For all tests, we have used the MPI1 method where all processes have to be started at once (see Sect. 3).

The numbers presented in Table 2 show the wall-clock time needed to perform the search for the bilinear interpolation, the time needed to perform the first exchange of data (which includes the calculation of the weights for the bilinear interpolation in the Transformer), and the time needed for subsequent data exchanges (averaged over 100 data exchanges). When repeating the measurements the

Table 2. Performance of OASIS4 tasks obtained on a PC cluster for 2-D search.

number of processes for			time in seconds for				
Driver	atmos	ocean	search	atm 1st exchn	ocn 1st exchn	atm nth exchn	ocn nth exchn
1	1	1	6.049	0.224	0.248	0.077	0.101
2	1	1	6.050	0.087	0.167	0.046	0.064
4	1	1	6.062	0.089	0.167	0.046	0.064
1	2	2	6.487	0.241	0.237	0.104	0.097
2	2	2	6.484	0.122	0.124	0.052	0.054
4	2	2	6.468	0.114	0.120	0.043	0.049
8	2	2	6.486	0.049	0.101	0.041	0.032
1	4	4	2.425	0.254	0.257	0.075	0.083
2	4	4	2.385	0.195	0.183	0.073	0.078
4	4	4	2.412	0.093	0.115	0.033	0.043
1	8	8	1.174	0.234	0.238	0.092	0.090
2	8	8	1.172	0.147	0.152	0.045	0.049
4	8	8	1.179	0.097	0.097	0.037	0.035
1	16	16	0.826	0.226	0.225	0.113	0.114
2	16	16	0.779	0.137	0.136	0.061	0.065
4	16	16	0.791	0.092	0.091	0.041	0.042
8	16	16	0.771	0.042	0.045	0.018	0.025

differences in time are in the order of a few milliseconds. Therefore we decided not to include any statistics as this will not change the general picture we are going to discuss.

The time needed for the search can be represented by the time needed for the PRISM_Enddef routine. The search is performed on the two sets of source component processes as the component A target points are searched for on the component B processes and vice versa. As the PRISM_Enddef and subsequent calls therein contain blocking MPI calls, the total time needed by the PRISM_Enddef is to a large extent controlled by the source process which has to solve the largest problem (mainly determined by the number of target points to be processed). The time spent in PRISM_Enddef is almost identical for all processes due to this blocking nature of the MPI function calls used inside. Therefore only one number, the time needed by the component process with the largest data load is shown in the table for each setting.

In general, the wall clock time needed for the search decreases with the number of processes dedicated to the components: in this case, the speed-up is roughly a factor of 8 when going from 2 to 16 processes. When going from one to two component processes, the time for the search is slightly increased. This can be attributed to the different algorithms used, as in the one-processor case certain subroutines do not have to be called simply because no boundary exchange is required on the source grid. When the number of processes is increased further the required wall-clock time is reduced significantly to values well below 1 s for the 2-D data exchange on 16 processors. At this stage, the number of Trans-

former processes is not of critical importance as they are not involved in the search. Communication with the Transformer is established only when the search for an intersection subdomain has been completed and search results are delivered to the Transformer process(es) (see Sect. 6).

In this example, a data exchange can be understood as one pair of PRISM_Put and PRISM_Get operations in each component (referred to as ping-pong). The time needed for a complete ping-pong usually is different on each process. Load-balancing problems provide one explanation for this behaviour. Even though the PSMILE library uses non-blocking send operations for the data exchange, this can only reduce load-balancing problems, but such problems cannot be solved inside the library. Furthermore, it may happen that one instance of the coupled system may have to receive more than just one message at a time. This is for example the case if only one Transformer process is available and two source components are trying to send or receive data at the same time. In this case, the messages can only be processed one after the other by the Transformer causing other component processes to wait. As only non-masked data are exchanged between the components, data are subsampled prior to the send operation and have to be scattered on the receiver side before they are delivered to the user space. This requires additional operations which are also captured by our time measurements for the ping-pong. In order to provide only a single number for the ping-pong, we show the time needed by the slowest component processor to complete the exchange.

We observe a general decrease of the wall-clock time for the exchange with an increasing number of Transformer processes available (Table 2). One reason is that the likelihood increases that the Transformer process is ready to receive, process and send data. The other reason is that the amount of data to be treated is reduced for each process. The only case when increasing the number of Transformer processes does not result in a decrease of the exchange wall-clock time is when the components themselves are not parallel and when the number of processes for the Transformer goes from 2 to 4. In this case, there are only two distinct intersection regridding lists (see Sect. 6), i.e. one when considering one component as the source and another one when considering the other component as the source. As the different Transformer processes work in parallel over the regridding of the different source and target process intersections, the 2 additional processes (when going from 2 to 4 processes) do not get any workload and do not participate in the data exchange.

During the first exchange of data in addition to the processing of the exchange fields, the Transformer has to calculate the weights which are then stored and reused for subsequent data exchanges. This additional workload is responsible for the longer time needed to perform the first exchange when comparing with the time needed for subsequent exchanges.

We also observe that for a fixed number of Transformer processes the exchange time is fairly independent of the number of application processes. As the data are exchanged through the Transformer processes, it can act as a bottleneck in the exchange if it is not sufficiently parallel. In general, it is the number of available Transformer processes that dictates the speed of the exchange and not the parallelisation of the components themselves.

Finally, we notice that the scaling behaviour is in general not fully predictable; this can mainly be attributed to the fact that a particular Transformer process may still be busy with other tasks, that data are sometimes not available from the source component or delivered only with a delay due to punctual load imbalancing. We observe super-scalability when going in our case from 2 to 4 processes for the components. In the 2-D case this super-scalability is also present when increasing the number of component processes further from 4 to 8 processes. Cache effects provide one possible explanation for this behaviour. A more efficient scheduling of the processes with a reduced waiting time for messages to arrive is another answer. A finer partitioning of the target grid may fit the remote partitioning better and thus reduce the overhead during the initial search.

8.3 3-D data exchange

The general picture does not change much with an increasing amount of data to be exchanged, for example going from the 2-D to a 3-D coupling, i.e. when exchanging data between 40 levels for component A and 45 levels for compo-

nent B (see Table 3). As in the 2-D case, the efficiency of the search mainly depends on the number of component processes available. The wall-clock time decreases by roughly a factor of 6 when going from 2 to 16 component processes. The time needed for the search decreases to only a few seconds when employing four or more application processes – which is likely the case with applications running with the grid sizes similar to our example. With an increasing number of Transformer processes the exchange time is reduced as in the 2-D case.

8.4 Performance of the multi-grid search and parallel regridding

To evaluate the performance of OASIS4 and in particular the efficiency of the multi-grid algorithm, the benchmark was adapted to the OASIS3 coupler and additional 2-D coupled runs were realized for different resolutions of components A and B. These additional runs were performed on a Single Core Intel Pentium 4 CPU 3.20 GHz Linux PC with MPICH-1 message passing. OASIS4, OASIS3 and the benchmark sources were compiled with the Portland Group Fortran Compiler 9.0-4 and with the GNU C compiler 4.4.1. One process was started for each component and the driver.

The results are presented in Table 4 for OASIS4 and in Table 5 for OASIS3. In each case, five runs were realized with a resolution ranging from T21 (2244 grid point) to T255 (295 680 grid points) for component A, and ranging from 4° (4692 grid points) to 0.5° (288 078 grid points) for component B. In all cases, the Transformer and the components were running with one process each. The numbers provided in the tables give the time for the search and the 1st exchange as measured in component A (3rd column) and in component B (4th column). We provide these measures to have a comparable basis: with OASIS3, the neighborhood search and the weight calculation is done during the 1st exchange, where as they are respectively done during the initialisation phase below the PRISM_Enddef call and during the 1st exchange with OASIS4. In addition, Tables 4 and 5 show the time for the n-th exchange in component A (5th column) and in component B (6th column).

By comparing Table 4 and Table 5, we see that even at relatively low resolution (2244 and 4692 grid points for components A and B, respectively) OASIS3 is about two times slower than OASIS4. The difference gets bigger with increasing resolution: the time required for the neighborhood search and the first exchange (including the weights calculation) increases with $O(N)$ for OASIS4 (see Fig. 10) and with $O(N^2)$ for OASIS3. Even the time for subsequent exchanges is always about 50% higher with OASIS3 than with OASIS4.

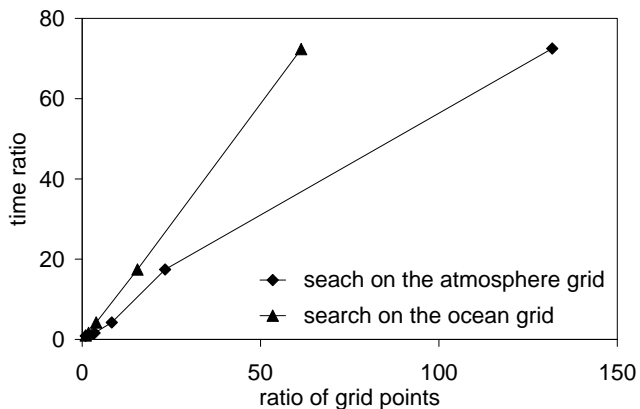
These numbers clearly demonstrate the benefit of the multi-grid neighborhood search when compared to a classical search and the increased general performance of OASIS4 over OASIS3 even in this simple non-parallel case.

Table 3. As in Table 2 but for 3-D search and exchange.

number of processes for			time in seconds for					
Driver	atmos	ocean	search	atm 1st exchnng	ocn 1st exchnng	atm nth exchnng	ocn nth exchnng	
1	1	1	16.671	2.605	3.091	0.715	1.191	
2	1	1	16.526	0.353	2.797	0.186	1.059	
4	1	1	16.313	0.357	2.789	0.189	1.050	
1	2	2	18.767	2.994	3.050	1.092	1.134	
2	2	2	18.753	2.629	2.686	0.912	0.979	
4	2	2	18.825	1.119	2.722	0.340	1.001	
8	2	2	18.856	0.226	1.452	0.131	0.570	
1	4	4	7.685	3.027	3.052	1.074	1.097	
2	4	4	7.276	1.626	1.824	0.500	0.649	
4	4	4	7.344	1.194	1.277	0.398	0.463	
1	8	8	4.382	3.011	3.033	1.039	1.061	
2	8	8	4.669	1.669	1.687	0.558	0.585	
4	8	8	4.756	0.933	1.102	0.320	0.405	
1	16	16	3.310	2.928	2.927	0.999	1.007	
2	16	16	2.739	1.765	1.793	0.563	0.590	
4	16	16	2.560	1.052	1.150	0.343	0.394	
8	16	16	3.035	0.557	0.627	0.183	0.250	

Table 4. Performance of OASIS4 for 2-D search and exchange.

Number of grid points for		time in seconds for			
atmos	ocean	atm search and 1st exchnng	ocn search and 1st exchnng	atm n-th exchnng	ocn n-th exchnng
2244	4692	0.874	0.869	0.008	0.008
7800	8418	1.410	1.394	0.018	0.018
18 624	18 382	3.691	3.661	0.037	0.037
52 164	72 762	15.229	15.148	0.119	0.119
295 680	288 078	63.354	62.877	0.540	0.538

**Fig. 10.** Ratio of time and grid points relative to the coarse resolution test (T21 ⇌ 4°) for the OASIS4 search based on the numbers shown in Table 4.

8.5 Discussion

As we tried to outline above, it is difficult to provide an exact prediction for the time needed by the search algorithm. Apart from the quality of the hardware involved, results depend on a variety of other parameters defined by the application or the user. The run-time behaviour (or load-balancing) of the physical components involved is one important aspect. Although the coupling software can try to hide as much as possible communication with computation, it cannot completely cancel out load-balancing problems. The fact that component A needs data in order to progress with its calculation and that application B has not yet delivered the data is totally independent of the coupling software functionality or optimisation. An external observer will notice that component A or the Transformer is eventually spending CPU time in some MPI function call waiting for data. The additional time needed for the coupling when compared against the run time of a stand-alone component run cannot necessarily be

Table 5. As in Table 4 but for OASIS3 with its classical neighborhood search.

Number of grid points for		time in seconds for			
atmos	ocean	atm search and 1st exchn	ocn search and 1st exchn	atm n-th exchn	ocn n-th exchn
2244	4692	1.917	1.921	0.013	0.013
7800	8418	9.561	9.571	0.024	0.024
18 624	18 382	44.354	44.360	0.052	0.052
52 164	72 762	499.611	499.612	0.176	0.176
295 680	288 078	10813.021	10813.051	0.767	0.767

attributed to an inefficient coupling software (or MPI implementation) but may solely be attributed to a inefficient load-balancing between the physical components involved.

Apart from those problems mentioned above, there are some more predictable parameters which influence the additional time needed for the coupling. Clearly, the time will depend on the number of exchange fields to be processed. If each field is defined on a separate grid a complete new search has to be performed which will consequently lead to a higher demand of CPU time. If different exchange fields are defined on an identical grid, the PSMILE is able to reuse previous results and the required time will only slightly increase when compared against the single field search. If no masks are applied at different vertical levels, the PSMILE is able to perform some optimisation which will again decrease the work load.

9 Conclusions and future perspectives

In this article, we have described the new OASIS4 coupling software in its technical details. The new parallel multi-grid search algorithm has been introduced. By using first simple coupled test configurations, the applicability of the software has been demonstrated. A remarkable feature of the new software is the rather efficient search algorithm. The compute time measured to perform the search provides confidence in the efficiency of the search algorithm. Apart from using OASIS4 for coupled models, it is also well suited to serve as a pre- or post-processing software to perform less time-critical regridding tasks. For coupled applications with reasonable process-local problem sizes of the order of 10 000 grid points in the horizontal, the time needed for the search is only up to the order of a second which is considered to be negligible when compared with the total runtime for a typical ESM job of several hours.

Up to now the focus of the development activities has been to provide the functionality which is required to make OASIS4 a useful tool for real applications. This first goal is now achieved, as OASIS4 is currently actively used for atmosphere dynamics to atmospheric chemistry coupling by the Global and regional Earth-system (Atmosphere) Monitoring using Satellite and in-situ data (GEMS) project (Flem-

ming et al., 2009), and for the Regional Coupled Atmosphere and Ocean (RCAO) model at the Swedish Meteorological and Hydrological Institute (SMHI) Rossby Centre (Döscher et al., 2009). Other groups are thinking about using the OASIS4 software within new projects, for example within the Scalable Earth-System-Models for high productivity climate simulations (ScalES) project recently funded by the Federal Ministry of Education and Research in Germany. The OASIS4 software has not yet been tested at higher resolution and with a higher degree of parallelism than those listed in this article. We do not expect any problems when going into this direction as the whole OASIS4 design, described in detail in this article, was done with high-resolution and highly parallel applications in mind.

The current version of OASIS4 provides support for block-structured grids and Gauss-reduced grids. A next step forward is to support unstructured grids which are used more and more in the context of climate modelling. This will be done in collaboration with the Alfred Wegener Institute for Polar and Marine Research (AWI) in the framework of the ScalES project. Other important developments are being or will be realised within the framework of funded projects, such as the InfraStructure for the European Network for Earth System Modelling (IS-ENES) supported by the 7th Framework Programme of the European Commission, in particular: (1) the realisation of a Graphical User Interface to help the user build the XML configuration files (see Sect. 3), (2) the possibility of using a pre-defined set of weights and addresses for the interpolation, and (3) general optimisation of the software. The current implementation provides support for a fully concurrent component processor layout. Work is in progress to provide the support for a sequential layout. On a longer timescale, the support of adaptive grids or grids changing for example their land-sea mask is targeted; as we have outlined above, the efficiency of the search algorithm provides the potential to handle this aspect. Finally, work will continue to establish comprehensive services around OASIS4 through a portal offering documentation, user guides, tutorial, FAQs, user forum and tips for best practices, as such services are considered essential to fully support an active community of users.

Acknowledgements. We highly appreciated Reiner Vogelsang's (SGI, Germany) collaboration during the early times of OASIS4, and thank Damien Declat (now at Bull) and Thomas Schoenemeyer (NEC-HPCE) for contributing to the initial phase of the OASIS4 development. Part of this work has been funded within the EU FP5 PRISM project (Contract EVR1-CT2001-40012), the CICLE project supported by the French Agence Nationale de la Recherche (ANR, Contract ANR-05-GIGC-04), and the EU FP7 IS-ENES project (Contract GA No: 228203). The support by CERFACS, the NEC High Performance Marketing Division (Tokyo, Japan) and CNRS is gratefully acknowledged. Special thanks (in alphabetical order) go to Mats Bentsen (NERSC), Joseph Charles (UK MetOffice), Laure Coquard (CNRS), Ralf Döscher (SMHI), Johannes Flemming (ECMWF), Josephine Ghattas (now at IPSL), Ulf Hanson (SMHI), Ian Henderson (University Reading), Jean Latour, Kristian Mogensen (ECMWF), Marie-Pierre Moine (CERFACS) and Andreas Ripke (NEC) for using and testing earlier versions of the software and for their very helpful comments that allowed us to improve the coding. Last but not least we would like to express our thanks to the anonymous reviewers for their very constructive comments and suggestions.

Edited by: M. Kawamiya

References

- Balaji, V.: Parallel Numerical Kernels for Climate Models, in: ECMWF TeraComputing Workshop 2001, ECMWF, World Scientific Press, 2001.
- Buja, L. and Craig, T.: Community Climate System Model CCSM2.0.1 User Guide, National Center of Atmospheric Research, Boulder CO, 2002.
- CLIVAR Scientific Steering Group: CLIVAR Initial Implementation Plan, International CLIVAR Project Office, SOC, Empress Dock, Southampton, UK, Vol. 14, 314 pages, 1998.
- Déqué, M., Drevet, C., Braun, A., and Cariolle, D.: The ARPEGE/IFS Atmosphere Model: A Contribution to the French Community Climate Modelling, *Clim. Dynam.*, 10, 249–266, doi:10.1007/BF00208992, 1994.
- Döscher, R., Wyser, K., Meier, H., Qian, M., and Redler, R.: Quantifying Arctic Contributions to Climate Predictability in a Regional Coupled Ocean-Ice-Atmosphere Model, *Clim. Dynam.*, doi:10.1007/s00382-009-0567-y, in press, 2009.
- Gropp, W., Huss-Lederman, S., Lumsdaine, A., Lusk, E., Nitzberg, B., Saphir, W., and Snir, M.: MPI – The Complete Reference, 2, The MPI Extensions, MIT Press, 1998.
- Hill, C., DeLuca, C., Balaji, V., Suarez, M., and da Silva, A.: Architecture of the Earth System Modeling Framework, *Comput. Sci. Eng.*, 6, 18–28, 2004.
- Flemming, J., Inness, A., Flentje, H., Huijnen, V., Moinat, P., Schultz, M. G., and Stein, O.: Coupling global chemistry transport models to ECMWF's integrated forecast system, *Geosci. Model Dev.*, 2, 253–265, 2009.
- Hourdin, F., Musat, I., Bony, S., Braconnot, P., Codron, F., Dufresne, J.-L., Fairhead, L., Filiberti, M.-A., Friedlingstein, P., Grandpeix, J.-Y., Krinner, G., LeVan, P., Li, Z.-X., and Lott, F.: The LMDZ4 general circulation model: climate performance and sensitivity to parametrized physics with emphasis on tropical convection, 27, 717–813, doi:10.1007/s00382-006-0158-0, 2006.
- Jones, P.: Conservative remapping: First- and second-order conservative remapping, *Mon. Weather Rev.*, 127, 2204–2210, 1999.
- Jones, P. W.: A User's Guide for SCRIP: A Spherical Coordinate Remapping and Interpolation Package, <http://climate.lanl.gov/Software/SCRIP/SCRIPusers.pdf>, 2001.
- Joppich, W. and Kürschner, M.: MpCCI – a tool for the simulation of coupled applications, *Concurr. Comp.-Pract. E.*, 18, 183–192, 2006.
- Li, J., Liao, W., Choudhary, A., Ross, R., Thakur, R., Gropp, W., Latham, R., Siegel, A., Gallagher, B., and Zingale, M.: Parallel netCDF: A Scientific High-Performance I/O Interface, in: Proceedings of the ACM/IEEE Conference on Supercomputing (SC), 39–49, 2003.
- Madec, G.: NEMO ocean engine, Note du Pole de modélisation 27, Institut Pierre-Simon Laplace, France, <http://www.nemo-ocean.eu/About-NEMO/Reference-manuals>, 2008.
- Madec, G. and Imbard, M.: A Global Ocean Mesh to Overcome the North Pole Singularity, *Clim. Dynam.*, 12, 381–388, 1996.
- Manabe, S. and Bryan, K.: Climate Calculations with a Combined Ocean-Atmosphere Model, *J. Atmos. Sci.*, 26, 786–789, 1969.
- Rew, R. K. and Davis, G. P.: Unidata's netCDF Interface for Data Access: Status and Plans, in: Thirteenth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology, Anaheim, California, American Meteorology Society, 1997.
- Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J.: MPI – The Complete Reference. Vol. 1: The MPI Core, 2nd edn., MIT Press, Cambridge, Massachusetts, 448 pages, 1998.
- Valcke, S.: OASIS3 User Guide (prism_2-5) Technical Report No. TR/CMGC/06/73, CERFACS, Toulouse, France, 68 pp., available at: http://www.cerfacs.fr/globc/publication/technicalreport/2006/OASIS3_UserGuide.pdf, 2006.
- Valcke, S. and Redler, R.: OASIS4 User Guide (OASIS4.0.2) Technical Report No. TR/CMGC/06/74, CERFACS, Toulouse, France, 64 pp., available at: http://www.cerfacs.fr/globc/publication/technicalreport/2006/OASIS4_User_Guide.pdf, 2006.
- Valcke, S., Budich, R., Carter, M., Guilyardi, E., Foujols, M.-A., Lautenschlager, M., Redler, R., Steenman-Clark, L., and Wedi, N.: A European Network for Earth System Modelling, *EOS*, 88, 143 pp., 2007.