OASIS and PALM, the CERFACS couplers

Sophie Valcke, Thierry Morel

June 13th, 2006

.

Technical Report TR/CMGC/06/38

1 Introduction

Coupling numerical models, i.e. implementing synchronized exchanges of information between models, is a central issue in many research fields such as climate modelling, data assimilation, or computational fluid dynamics. Since about 15 years, CERFACS specializes in developing coupling software tools, also called "couplers".

In 1991, CERFACS was commissioned to develop a software interface to couple existing numerical General Circulation Models of the ocean (OGCMs) and of the atmosphere (AGCMs). Two years later, a first version of the OASIS coupler was distributed to the community. Today, both the widely used mono process OASIS3 coupler [1] and the new fully parallel OASIS4 coupler [2] are available.

Concurrently, the development of the PALM coupler, originally designed for oceanographic data assimilation and therefore addressing parallel dynamic coupling, started in 1997. Two implementations of PALM are currently available, PALM Research and PALM MP [3] respectively based on Message Passing Interface MPI1 [4] and MPI2 [5].

This document gives an overview of the OASIS and PALM couplers, describing their analogies and differences, and concludes with a discussion on the possible interactions in their future development.

2 OASIS3: a static mono-process coupler

OASIS3, released in the framework of the PRISM project [6], is the direct evolution of the OASIS coupler developed since 1991 at CERFACS. It is a portable set of Fortran 77, Fortran 90 and C routines. At run time,

OASIS3 acts as a separate **interpolation process** and as a **parallel model interface library**, the OASIS3 PSMILe, that needs to be linked to and used by the component models. Modularity, flexibility, and portability form OASIS3 key design concepts.

Coupling configuration

At run time, the OASIS3 main process first reads the coupled run configuration **written by the user before the run in the** *namcouple* **text file** following a specific format, and distributes it to the different component model PSMILes. The *namcouple* contains all coupling options for a particular coupled run, e.g. the duration of the run, the component models, and, for each exchange, a symbolic description of the source and target (a component for coupling exchanges, or a file for I/O actions), the exchange period, regridding and other transformations. During the run, OASIS3 main process and the component model PSMILes automatically perform appropriate exchanges based on the configuration information contained in the *namcouple*.

Process management

In a coupled run using OASIS3, the component models remain separate executables. If the user has chosen the MPI2 approach, the OASIS3 main process launches the different component models using the `MPI_Comm_Spawn` functionality [5]. If only MPI1 [4] is available, the OASIS3 main process and the component models must be all started at once in the job script[1]. In both cases, all component models are necessarily integrated from the beginning to the end of the run, and each coupling field is exchanged at a fixed frequency defined in the *namcouple* for the whole run; in that sense, OASIS3 supports **static coupling** only.

Coupling field transformation and regridding

For each coupling exchange, the OASIS3 main interpolation process receives the coupling field from the source model, performs the transformations and 2D regridding needed to express the source field on the grid of the target model, and sends the transformed field to the target model. Dif-

[1]The advantage of the MPI2 approach is that each component keeps its own internal communication context unchanged as in the standalone mode, whereas in the MPI1 approach, OASIS3 needs to recreate a component model communicator that must be used by the component model for its own internal parallelisation.

ferent 2D regridding algorithms (nearest-neighbour, gaussian-weighted, bilinear, bicubic, conservative, etc.) in the Earth spherical coordinate system are available for different types of grids (regular in longitude and latitude, stretched, rotated, gaussian reduced, unstructured). OASIS3 can also be used in the *interpolator-only mode* to interpolate fields contained in files without running any model.

Communication: the OASIS3 PSMILe software layer

To communicate via the OASIS3 main interpolation process or to perform I/O actions, a component model needs to call few specific PSMILe routines for its initialisation, grid and partition definition, field declaration, field **Get** and **Put** actions (to receive or send a field by respectively calling a *prism_get* or a *prism_put* routine), and termination. Below the *prism_get* and *prism_put*, the PSMILe library automatically performs coupling exchanges (i.e. between two component models) using MPI (Message Passing Interface [4] [5]) and I/O actions from/to disk files using the GFDL mpp_io library [7], following the user externally defined configuration in the *namcouple* (see above).

PSMILe supports parallel communication in the sense that each process of a parallel model can send its local part of the field. For coupling exchanges, the different parts of the field are sent either directly to the other parallel model (if there is no need of redistribution or interpolation) or to the OASIS3 main interpolation process, which gathers the whole coupling field, transforms or regrids it, and redistributes it to the target component model processes.

As for all CERFACS couplers, the communication follows the **end-point principle**, i.e. there is no reference in the component model code to the origin of a Get action or to the destination of a Put action; the source and target component models (coupling exchange) or the input or output file (I/O) are set externally by the user. This ensures an easy transition from the coupled mode (Get or Put action corresponding to a coupling exchange) to the forced mode (Get or Put action corresponding to I/O from/to a file), totally transparent for the component model itself. Furthermore, the Get/Put routines can be called at each time step in the component model code, but the receiving and sending actions will effectively be performed only at appropriate times from/to the appropriate source/target following the configuration externally defined by the user (in the *namcouple* file).

The OASIS3 community

The OASIS community has steadily grown since its first release about 15 years ago. The OASIS3 coupler is currently used by about 15 modelling groups in Europe, Australia, Asia and North America, on many different platforms such as the Fujitsu VPP5000, NEC SX5, SGI Octane and 03000, IBM Power4, COMPAQ Alpha cluster and Linux PC cluster.

Full user support including bug fixes and release of new versions is currently provided for OASIS3, even if most of the development efforts are now devoted to OASIS4.

3 OASIS4: a static fully parallel coupler

As the climate modelling community is progressively targeting higher resolution climate simulations on massively parallel platforms with coupling exchanges involving a higher number of (possibly 3D) coupling fields at higher coupling frequencies, a completely new fully parallel coupler OASIS4 has also been developed within PRISM. OASIS4 is a portable set of Fortran 90 and C routines. At run-time, OASIS4 acts as a separate **parallel executable**, the OASIS4 Driver-Transformer, and as a **fully parallel model interface library**, the OASIS4 PSMILe. The concepts of parallelism and efficiency drove OASIS4 developments, keeping at the same time in its design the concepts of portability and flexibility that made the success of OASIS3.

Coupling configuration

Each component model to be coupled via OASIS4 should be released with an XML[2] file describing all its potential input and output fields, i.e. all the fields that can be received or sent by the component through PSMILe Get and Put actions in the code. Based on those description files, the user produces, either manually or via a Graphical User Interface, the **XML configuration files**. As for OASIS3, the OASIS4 Driver extracts the configuration information at the beginning of the run and sends it to the different model PSMILes, which then perform appropriate coupling or I/O actions automatically during the run. OASIS4 is also highly flexible in the sense that any duration of run, any number of component models, any number of coupling and I/O fields and particular coupling or I/O parameters for each field can be specified.

[2]XML (eXtensible Markup Language) is a simple and flexible standard text format [8].

Process management

The process management in OASIS3 and OASIS4 are quite similar: the component models remain separate executables and the MPI1 and MPI2 approaches are available (see OASIS3 paragraph on the subject). The process management of OASIS4 is however slightly more complex as OASIS4 Driver can spawn the different component models on different machines.

Although this functionality is currently not fully tested and optimised, OASIS4 also allows the component models to redefine their grid during the run. But besides this dynamic grid aspect, OASIS4, as OASIS3, also manages **static coupling** only.

Coupling field transformation and regridding

During the run, the OASIS4 parallel Transformer manages the **transformation and regridding of 2D or 3D coupling fields**. The Transformer performs only the weight calculation and the regridding *per se*; the neighbourhood search, i.e. the determination for each target point of the source points that contribute to the calculation of its regridded value, is performed in parallel in the source PSMILe.

During the simulation time stepping, the OASIS4 parallel Transformer can be assimilated to an automate that reacts to what is demanded by the different component model PSMILes, i.e. to receive data for transformation (source component) or to send transformed data (target component). The OASIS4 Transformer therefore acts as a parallel buffer into which the transformations take place. Currently, only 2D and 3D nearest-neighbour, 2D and 3D linear, and bicubic regridding methods are implemented, but plans are to implement also 3D cubic grid interpolation, and 2D and 3D conservative remapping.

Communication: the OASIS4 PSMILe software layer

To be coupled via OASIS4, the component models have to, as with OASIS3, include specific calls to the OASIS4 PSMILe software layer. The OASIS4 PSMILe Application Programming Interface (API) was kept as close as possible to OASIS3 PSMILe API; this should ensure a smooth and progressive transition between OASIS3 and OASIS4.

The OASIS4 PSMILe supports **fully parallel MPI-based communication**, either directly between the models (including automatic repartitioning if needed) or via the parallel Transformer, and **file I/O using the GFDL mpp_io library** [7]. The detailed communication pattern

among the source and target component model processes is established by the PSMILe, using the results of the regridding or repartitioning neighbourhood search. This search is based on the source and target identified for each coupling exchange by the user in the XML configuration files and on the local domain covered by each component process. The search uses an efficient multigrid algorithm and is done in parallel in the source component PSMILe, which ensures that only the useful part of the coupling field is extracted and transferred.

Besides these new parallel aspects, the OASIS4 PSMILe follows the same **end-point communication** and **user-defined external configuration** principles than the OASIS3 PSMILe.

The OASIS4 users

OASIS4 portability and scalability was demonstrated with different "toy" models during the EU FP5 PRISM project [6]. OASIS4 was also used to realize a coupling between the MOM4 ocean model and a pseudo atmosphere model at Geophysical Fluid Dynamic Laboratory (GFDL) in Princeton (USA), and with pseudo models to interpolate data onto high resolution grids at IFM-GEOMAR in Kiel, Germany.

Currently, work is going on with OASIS4 at:

- the Swedish Meteorological and Hydrological Institute (SMHI) in Sweden for coupling regional ocean and atmosphere models;
- the European Centre for Medium-Range Weather Forecast (ECMWF), KNMI in the Netherlands, and Météo-France in France in the framework of the EU GEMS project, for 3D coupling between atmosphere and atmospheric chemistry models;
- at the UK MetOffice for global ocean-atmosphere coupling.

After the current beta-testing phase, the first official OASIS4 version should be publicly available beginning of 2007.

4 PALM: a dynamic parallel coupler

In contrast with OASIS3 and OASIS4 that support *static* coupling of parallel codes, PALM is a coupler designed to combine *dynamically* different components into a high performance application [3]. PALM was originally developed for operational oceanographic data assimilation in the framework

of the French MERCATOR project [9]. As OASIS3 and OASIS4, PALM is a portable set of Fortran 90 and C routines. PALM Driver supports the **dynamic launching of the coupled components** while its coupling library ensures the **parallel data exchanges including repartitioning** between the components. PALM also provides **pre-defined algebra units**.

The concept of dynamic coupling

The concept of dynamic coupling came from the observation that different data assimilation algorithms can be obtained with different execution sequences of the same basic units and operators. In PALM, a dynamic coupling algorithm is composed of basic pieces of codes, the *units*, assembled in different execution sequences, the *branches*, which can include complex control structures like loops, "if" constructs and "select" switches; **the units can therefore be started or stopped dynamically during the run**.

Coupling configuration

The user defines and provides the elementary units, thereby fixing the granularity of the coupling. Each unit is a piece of code that must be instrumented by the user with some PALM primitives and specific comments. Each unit can consume and/or produce data, which are called *objects*, via the implementation of the PALM Get and Put primitives. All the *objects* that a unit can request and/or provide must be described in the unit code by comment lines following a pre-defined syntax, the *identity cards*, which contain the object metadata, i.e. a description of its *space* (its numerical type and shape) and its *distributor* (the way the object is distributed amongst the different processes). As with OASIS3 and OASIS4, modularity is ensured by the **end-point communication** principle, i.e. there is no reference to the origin of the input or to the destination of the output in the unit code.

Through a powerful **graphical user interface**, PrePALM, the user chooses the elementary *units* to be coupled, which appear as individual boxes on PrePALM screen, and defines their execution sequences, the *branches*. PrePALM analyses the *identity cards* in the different *unit* codes and clearly identifies the potential data input and output as "plugs" on the boxes. To establish an exchange of information between the units, the user links the output plug of one unit to the input plug of another unit; a pop-up appears on the link which allows the user to specify the different exchange parameters, such as the times of exchange.

PrePALM also provides supervision tools such as a **performance analyser** and a **runtime monitoring**.

Process Management

At run time, the PALM Driver ensures the **execution and synchronisation of the different units**, compiled by the user, following the sequence of actions defined in PrePALM. The MPI2 standard, providing the ability to spawn tasks dynamically, was a natural choice for **PALM_MP**: the Driver uses the **MPI2** *MPI_Comm_spawn* **functionality to launch, when needed, the different units** as separate executables ensuring thereby the total integrity of each unit.

At the time of initial development, the MPI2 standard was however not available on all targeted platforms; it was therefore decided to develop a first prototype of PALM, **Palm_Research**, using MPI1. With Palm_Research, all units must be linked with PALM into one single executable, within which the Driver **launches the different units by starting corresponding subroutines** on a certain number of idle processes.

Parallel communication

In the *unit* code, the action of consuming data or providing data is implemented by calling respectively the *palm_get* or *palm_put* primitive (**Get** or **Put** action). At run-time, a communication takes places if the source component issues a Put, if the target component issues a Get, if an exchange has been defined between the corresponding output and input "plugs" (by the user with PrePALM), and if both Put and Get are issued with corresponding time stamps that fit one time exchange specified by the user with PrePALM. The correspondence is established by the Palm Driver that acts as a **broker in the communication space**. The units can be parallel, in which case the PALM coupling library takes care entirely of the **data redistribution (repartitioning) during the communication**.

For sake of performance, PALM minimizes the number of MPI messages and the memory usage. If a Get is issued before the corresponding Put, the corresponding *object* is immediately and directly sent from the source unit to the target unit when the Put is performed. If a Put is issued before the corresponding Get, the *object* is temporarily stored in a **volatile memory space, called the** *mailbuff***,** so that the Put never blocks.

With PALM_MP, the user can also define, through the graphical interface PrePALM, that some units be assembled within one *block*, i.e. one executable. In that case, the different units within the block are launched by calling the appropriate subroutines and, if possible, the objects exchanged between the units are transferred by memory copies using a local process

mailbuff, and not by message passing.

PALM also offers the possibility of using explicitly a **permanent storage space called a** *buffer***.** The user can explicitly create a communication between a unit and the buffer to store an object and recover it when needed. The buffer is particularly useful for linear combination of different fields and for time interpolation.

The algebra units

The user can include **pre-defined algebra units** in its coupling algorithm. These mono or multi process units, which are made available as a toolbox via the PrePALM graphical interface, include basic operations for matrix and vector fields, 2D linear interpolation, linear combination, linear system solvers, eigenvector and eigenvalue solvers and minimizers. This algebra toolbox is provided with the PALM software and can easily integrate developments of general interest done by the users. For example, the 2D geographical interpolations included in the algebra toolbox were extracted from the OASIS3 software and interfaced as generic algebra units.

The PALM users

PALM is now a mature software with different communities of users. As originally planned, PALM is used for data assimilation in operational oceanography in the MERCATOR project [9]. PALM was also chosen as a common software development platform for data assimilation by the French community of atmospheric chemistry modelling, for example in the ADOMOCA project and in the former EU project ASSET. PALM is also used for data assimilation in hydrology at the *Centre National de la Recherche Météorologique* (CNRM) and also at the *Laboratoire d'étude des Transferts en Hydrologie et Environnement* (LTHE) in Grenoble, for data assimilation in neutronics at *Electricité de France* (EDF), and for assimilation of remote sensing data in agricutural modelling.

PALM is also currently used for different applications requiring dynamic coupling, such as in the CFD team at CERFACS for shape optimisation. Given the number of recent contacts with different scientific communities in France all looking for a flexible tool to handle dynamic coupling, for example in the framework of the SEVE project at the *Centre d'Etudes Spatiales de la BIOsphère* (CESBIO), but also at: SNECMA, the *Institut Universitaire des Systèmes Thermiques Industriels* (IUSTI), the *Institut de Mécanique des Fluides de Toulouse* (IMFT), the *Institut d'Informatique et Mathématiques Appliquées de Grenoble* (IMAG), the *Of-*

fice National d'Etudes et de Recherches Aérospatiales (ONERA), the PALM user community should increase rapidly in the coming years.

5 Discussion

As detailed above, OASIS is a static coupler which mono-process version, OASIS3, should be replaced within the next few years by the new fully parallel OASIS4. PALM is a dynamic and parallel coupler; the current PALM_MP version is based on MPI2 for dynamic launching of the units and to manage the communications between those units.

When the development of the OASIS4 coupler started during the PRISM project, it was first considered to use PALM_MP and implement parallel interpolations in PALM. This technical choice was finally not retained for the following reasons:

- MPI2 was at the time not fully implemented on all platforms, especially on the IBM (and it is still not clear if it will be one day)[3];
- the climate modelling community, main target of the PRISM project, did not express the need of dynamic coupling and was therefore not ready to meet MPI2 constraints;
- PALM was not open-source, condition considered mandatory in PRISM.

PALM and OASIS have therefore been developed separately, and it is not recommended to use the two couplers together in the same coupled application. Our expertise in CERFACS allows us to recommend one coupler or the other to the users depending on their need[4].

[3]It must be clear that even if MPI2 presents some restrictions regarding its implementation, MPI2 was and is still today the only technical choice possible to answer all PALM requirements, i.e. 1- performances, 2- dynamic launching of units, 3- integrity of those units, and 4- optimal use of available CPU resources. It is true however that for simpler coupled configurations for which the dynamic launching of the units is not mandatory, we could have included in the PALM design also the possibility to manage communications based on MPI1.

[4]For current users that would like to merge existing OASIS and PALM applications, we are however currently evaluating the possibility to use both couplers at the same time in one application (for example to replace the ocean component of an existing ocean-atmosphere model coupled by OASIS with an existing oceanic data assimilation chain based on PALM).

On the longer term, when a community will express the need of parallel interpolation in a dynamic coupling, we will have to consider including in PALM the OASIS4 parallel interpolations techniques. But as this would interfere deeply with the way communications are handled, it will probably be more appropriate instead to parallelize the PALM current mono-process interpolation algebra units. Both cases would represent important efforts of development and would require a specific funding.

Even in that case, we will probably maintain the OASIS4 coupler separately for users not asking for dynamic coupling and therefore not ready to meet MPI2 constraints, unless we judge more efficient at that time to include in PALM also the possibility to manage communications based on MPI1 only, as described above.

References

[1] Valcke, S., A. Caubel, R. Vogelsang, and D. Declat, 2004: OASIS3 User Guide (oasis3_prism_2-4). PRISM Report Series, No 2, 5th Ed., 64 pp. (http://prism.enes.org/Publications/Reports/Report02.pdf)

[2] Valcke, S., R. Redler, R. Vogelsang, D. Declat, H. Ritzdorf, and T. Schoenemeyer, 2004: OASIS4 User Guide. PRISM Report Series No 3, 72 pp. (http://prism.enes.org/Publications/Reports/Report03.pdf)

[3] Buis, S., A. Piacentini, D. Déclat, 2005: PALM: A Computational framework for assembling high performance computing applications. Concurrency Computat.: Pract. Exper., 18(2), 247-262.

[4] Snir, M., S. Otto, S. Huss-Lederman, D. Walker and J. Dongarra, 1998: MPI - The Complete Reference, Vol. 1 The MPI Core, MIT Press.

[5] Gropp, W., S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir, and M. Snir, 1998: MPI – The Complete Reference, Vol. 2 The MPI Extensions, MIT Press.

[6] http://prism.enes.org/

[7] http://www.gfdl.noaa.gov/˜ vb/mpp_io.html

[8] http://www.w3.org/XML/

[9] http://www.mercator-ocean.fr/