

The PRISM Coupling and I/O System

Sophie Valcke¹, Damien Declat¹, René Redler², Hubert Ritzdorf², Thomas Schoenemeyer³, and Reiner Vogelsang⁴

¹ CERFACS, Toulouse, France

² NEC-CCRLE Sankt Augustin, Germany

³ NEC-HPCE Düsseldorf, Germany

⁴ SGI Munich, Germany

Abstract. The aim of the PRISM coupling and I/O system, developed in the framework of the EU funded PRISM project, is to provide a portable, efficient, and easy-to-use open source software package, to manage coupling exchanges between arbitrary climate component models, either directly between the components or via additional transformation processes, as well as the I/O of each individual component.

1 Introduction

A new coupler for Earth System Models (ESMs) is currently under development in the framework of the Program for Integrated Earth System Modelling (PRISM). PRISM is an infrastructure project funded by the European Commission, with 22 European institutions actively participating, started in December 2001 for 3 years. Its main objective is to provide a software infrastructure facilitating the assembly, execution and post-processing of Earth System model simulations, based on existing state-of-the-art European Earth System component models [1].

The PRISM Coupling and I/O System drives the coupled ESM, performs MPI-based (Message Passing Interface [2]) parallel exchange of coupling data between its component models, either directly or via additional transformation processes, and exchange of I/O data between the components and disk files.

Other MPI-based parallel coupler performing field transformation exist, such as the ‘Mesh based parallel Code Coupling’ (MpCCI) [3] or the ‘CCSM Coupler 6’ [4]. The originality of the PRISM Coupling and I/O System relies in its great flexibility (the coupling and I/O configuration is externally defined by the user in XML files, see section 2, in its parallel neighborhood search based on the geographical description of the process local domains (see section 4.4), and in its common treatment of coupling and I/O exchanges (see section 4.5).

2 General view of the system

The PRISM Coupling and I/O System is composed of a Driver (see section 3, a PRISM System Model Interface Library (PSMILe, see section 4), and a Transformer (see section 5).

In the PRISM context, a component model is a Fortran code which models a subsystem of the Earth System (e.g. an Atmosphere General Circulation Model -AGCM). To interact with the ESM other components, the component model has to include specific PSMILe instructions.

To configure one particular coupled simulation, the user writes a Specific Coupling Configuration (SCC) XML file that contains the process management information of the coupled simulation, and, for each component, a Specific Model Input and Output Configuration (SMIOC) XML file that specifies the relations the component model will establish at run time with the rest of the coupled model through coupling exchanges or I/O for a specific run. During the run, the Driver, the component PSMILes, and the Transformer automatically act accordingly to the user's specification contained in XML files.

3 The Driver

The Driver first extracts the user-defined information from the XML files. If the different component models identified in the SCC are not simultaneously started (MPI1 mode), the Driver then launches them using MPI2 spawn functionality [5] (MPI2 mode). All components are launched initially for the whole run; dynamical start is not allowed.

The Driver then participates in the establishment of the different MPI communicators (see Section 4.1) and transfers the relevant XML information (e.g. source or target components or files and their global identifier, transformations, etc.) to the different component model PSMILes. These are then able to run without any other interactions with the Driver. Thus the Driver process is then used to execute the Transformer routines (see Section 5).

4 Interface and communication

To communicate with the rest of the coupled system, each component model needs to be linked with the PSMILe. While it is not designed to handle the component internal communication, the PSMILe completely manages the communication to other model components and the details of I/O file access (see Section 4.5).

The determination of the communication pattern as well as the exchanges of coupling fields *per se* handled by the PSMILe are hidden from the component codes. The interface was designed to keep modifications of the model codes at a minimum when implementing the API. The next sections describe the functioning of the PSMILe in the same logical order in which the PSMILe API routines should be called in the component model code.

4.1 Initialisation phase

The initialisation of the coupling environment, in particular of the MPI library if it has not been already done by the process, is performed.

In the MPI2 mode (see section 3), all component processes then participate in the launching of further component processes until the launching is completed. If needed,

an MPI communicator for the component internal communication is provided by the PSMILe .

4.2 Declaration of the grids and related quantities

Supported grids include: completely regular grids, irregular grids in longitude and latitude but regular in the vertical, and regular and irregular horizontal grids with sigma coordinates in the vertical. Unstructured grids will also be supported in a future version.

In the first step, the grid with its most general characteristics like the type of the grid and the shape of the local computing domain is defined. For gridded data, the volume elements which discretize the local partition computing domain on the sphere are then defined by providing their corner points (vertices); the communication and the repartitioning of gridded data between two coupled component models will be based on this geographical description of the local partition. For non-gridded data, the transfer and repartitioning between two component models will be based on the description of the local partition in terms of indices in the global index space.

In a second step, different sets of points on which the variables will be evaluated can then be placed in these volume elements. For vectors, three sets of points can be defined so that different vector components can be placed at different locations (staggered grids). For subgrid variables, the fraction of each subgrid class for the volume elements can also be defined, introducing a 4th dimension within the volume.

4.3 Declaration of Transient Variables

Each transient input or output variable is then declared and associated with the previously defined grids. Bundle variables, i.e. a set of related variables that can be ordered along a 4th dimension, can be declared and transferred as one coupling variable.

4.4 Neighborhood search and determination of communication patterns

The detailed communication pattern among the participating parallel component models is established by the PSMILe. It is based on the source and target identified for each coupling exchange by the user in the SMIOCs and on the local domain covered by each component process, and on the result of the neighbourhood search. This search is done in parallel in the source PSMILe in order to save communication time, decrease the memory and CPU consumption.

In an initial step, bounding boxes are locally computed for each local volume grid. The bounding boxes of all grids of all component processes are collected and distributed to all processes; each component process then has a global view on grids and processes with which it may have common coupling interfaces. Using this collected data, the processes can locally create a sequence of simplified grids, currently coarsened by a factor of 2 (similar to a Multigrid Algorithm) and used for the neighbouring search. Using the collected data, the processes can also decide which grid points have to be transferred to a “source” process for the neighbourhood search. After receiving these coordinates, the “source” process performs the neighbourhood search and informs the

“target” process which target points were found in its local grid volume. At this point, the source process knows which points can be sent directly to the target process since a matching target grid point was found and for which points the Transformer has to perform the interpolation.

Coupling data produced by one source component model may be only partially consumed by the target component model. The neighbourhood calculation performed in the source component PSMILe allows the automatic extraction of useful subdomains only and therefore minimizes the amount of data transferred.

4.5 Coupling exchange and file I/O of transient variables

The PSMILe exchanges are based on the principle of “end-point” data exchange. When producing data, no assumptions are made in the source component code concerning which other component will consume these data or whether they have to 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 it or whether they are read in from a file. Furthermore source data can be directed to more than one target (other component models and/or disk files).

The PSMILe “put” and “get” instructions can be placed anywhere in the source and target code and possibly at different locations for the different coupling fields. Each process of a parallel component model sends or receives only its local partition of the data, corresponding to its local grid defined previously (see Section 4.2). The destination or the source (another component model or a file) for each field is defined by the user in the SMIOC (see Section 2) and the coupling exchanges and/or the I/O 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.

Other arguments of the “put” and “get” instructions are the actual date at which the call is performed and the date bounds for which it is valid. These routines can be called by the model at each timestep. The sending/receiving is actually performed only if the date and date bounds corresponds to a time at which it should be activated, given the field coupling or I/O period indicated by the user in the SMIOC; a change in the period is therefore also totally transparent for the component model itself.

Local operations specified by the user in the SMIOC may also be performed automatically in the PSMILe, such as time accumulation or averaging, general algebraic operation, combination, data, reduction, statistics, scattering, gathering, etc.

The coupling exchange may then occur directly between two component models or via additional Transformer processes if needed (see Section 5), which is based on the pre-established communication patterns (see Section 4.4). The communication is based on MPI [2], a widely used and portable standard available for every architecture used by the climate modelling community. MPI is available either as open source public domain code or as a proprietary implementation using the architecture most efficient network.

When the user specifies that the source or target is a file (I/O exchange), the MPI send/receive operations are “replaced” by read and write operations respectively. The supported file format is NetCDF according to the CF convention which ensures: 1- portability of data between different computer hardware platform, 2- a self-explaining

description of data according to the needs of climate modelling community, 3- the seamless usage of post-processing and visualization tools developed in the PRISM project.

For those I/O exchanges, we exploit the MPP package [6] developed by the FMS project (Flexible Model System) ¹ at GFDL Princeton. The MPP package is proven technology and is used for instance by the ocean model MOM4. Moreover, the MPP package supports the two general I/O modes: 1- distributed (each component process works on an individual file containing the field information corresponding to the process local domain), 2- parallel (the data are collected, stitched together, and written to one file by the main process; or the data are read from one file by the main process and distributed to the parallel component processes). As a PSMILe further enhancement, there are plans to evaluate a very recent development of a parallel NetCDF library called Par-Netcdf [7].

4.6 Termination

At the end of the run, each process reaches the termination phase and waits for other processes participating in the coupling to reach it as well. At this point, all open units under control of the PSMILe are closed, the output to standard out is flushed, and all memory under control of PSMILe is deallocated. After this phase, no coupling exchange is possible anymore for this process and no further I/O actions under control of the PSMILe can be performed.

Furthermore, if a process needs to abort before the end of the run, the PSMILe provides a specific abort call which guarantees a clean and well-defined shut down of the whole coupled model.

5 The Transformer

The Transformer performs the calculation of the weights and the regridding of the coupling fields (often called the interpolation), i.e. the expression on the grid of a target component of a field given by a source component on its grid.

The work flow of the Transformer is mainly determined in a loop over the receptions of predefined arrays sent by the component PSMILe giving a clear description of what has to be done by the Transformer. Following pre-defined sequences, the PRISM Transformer is thus able to react to the corresponding sequence activated on the sender side.

During the initialisation phase, all information given through the SMIOC files is set up in the Driver/Transformer structures. Then the Transformer receives from the different component PSMILes the grid information resulting from the different neighbouring searches. As described in Section 4.4, the information at this stage (and the corresponding data arrays sent later during the simulation) is transferred separately for the intersection of each pair of target and source process local domains. This design will allow an easy parallelisation of the Transformer. The Transformer then calculates the weight corresponding to each source neighbour depending on the regridding method chosen by the user.

¹ <http://www.gfdl.gov/~fms>

During the simulation timestepping, the Transformer receives orders from the components processes to receive data for transformation (source component process) or to send transformed data (target component process). After a reception, the PRISM Transformer applies the appropriate transformations or regridding following the information collected during the initialisation phase. In case of request of fields, the PRISM Transformer controls if the requested field has already been received and transformed; if so, the data field is sent; if not the data field will be sent as soon as it is received and treated.

6 Conclusion

In this paper, the general design of a new parallel coupling and I/O system for Earth System Models currently under development in the framework of PRISM project was presented. Its different parts, i.e. the Driver, the PSMILe and the Transformer were described in more details.

A first prototype of this coupler has been delivered at end of April 2004. Performance evaluation has not been done yet. The final coupler gathering all the functionality described here above will be available at the end of PRISM project, i.e. in December 2004.

With this new coupling and I/O system design, PRISM hopes to ensure efficient and parallel 3D coupling for full climate models built on complex component models representing the different parts of the Earth System running on the present and next generation of advanced high performance computing facilities.

References

1. Guilyardi, E., Budich, R., Komen, G., Brasseur, G.: PRISM System Specification Handbook, Version 1. PRISM report series No.1, 230 pp. ISBN: 90-369-2217-8. (2003) <http://prism.enes.org/Results/Documents/Handbook/Handbookv1.0.1.pdf>.
2. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J.: MPI – The Complete Reference. Second edn. Volume 1, The MPI Core. MIT Press (1998)
3. Dehning, C., Post, P., Rümpler, C., Wolf, K.: MpCCI - Coupled Simulations through Code Coupling, NAFEMS Seminar on 'Simulation of Complex Fluid-Problems' May 3-4, Niederrhausen, Germany (2004)
4. Buja, L., Craig, T.: Community Climate System Model CCSM2.0.1 User Guide, National Center of Atmospheric Research, Boulder CO. (2002)
5. Gropp, W., Huss-Lederman, S., Lumsdaine, A., Lusk, E., Nitzberg, B., Saphir, W., Snir, M.: MPI – The Complete Reference. Volume 2, The MPI Extensions. MIT Press (1998)
6. Balaji, V.: Parallel Numerical Kernels for Climate Models. In: ECMWF TeraComputing Workshop 2001, ECMWF, World Scientific Press (2001)
7. Li, J., Liao, W., Choudhary, A., Ross, R., Thakur, R., Gropp, W., Latham, R., Siegel, A., Gallagher, B., Zingale, M.: Parallel netCDF: A Scientific High-Performance I/O Interface. In: SC2003 Proceedings, Supercomputing (2003)