Manuscript prepared for Geosci. Model Dev. Discuss.
with version 2014/07/29 7.12 Copernicus papers of the LaTeX class copernicus.cls.
Date: 4 September 2014

C-Coupler1: a Chinese community coupler for Earth System Modelling

L. Liu[1], G. Yang[1,2], B. Wang[1,3], C. Zhang[2], R. Li[2], Z. Zhang[2], Y. Ji[2], and L. Wang[4]

[1]Ministry of Education Key Laboratory for Earth System Modelling, Center for Earth System Science (CESS), Tsinghua University, Beijing, China
[2]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[3]State Key Laboratory of Numerical Modelling for Atmospheric Sciences and Geophysical Fluid Dynamics (LASG), Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing, China
[4]College of Global Change and Earth System Science, Beijing Normal University, Beijing, China

Correspondence to: L. Liu (liuli-cess@tsinghua.edu.cn), G. Yang (ygw@tsinghua.edu.cn), and B. Wang (wab@tsinghua.edu.cn)

Abstract

A coupler is a fundamental software tool for earth system modelling. Targeting the requirements of 3-D coupling, high-level sharing, common model software platform and better parallel performance, we started to design and develop a community coupler (C-Coupler) from 2010 in China, and finished the first version (C-Coupler1) recently. The C-Coupler1 is a parallel 3-D coupler that achieves the same (bit-identical) results with any number of processes. Guided by the general design of the C-Coupler, the C-Coupler1 enables various component models and various coupled models to be integrated on the same common model software platform to achieve a higher-level sharing, where the component models and the coupler can keep the same code version in various model configurations for simulation. Moreover, it provides the C-Coupler platform, a uniform runtime environment for operating various kinds of model simulations in the same manner. The C-Coupler1 is ready for earth system modelling, and it is publicly available. In China, there are more and more model groups using the C-Coupler1 for the development and application of models.

1 Introduction

Climate system models (CSMs) and earth system models (ESMs) are fundamental tools for global climate change study. They play an important role in simulating and understanding the past, present, and future climate. They are always coupled models consisting of several separate interoperable component models to simultaneously simulate the variations of and interactions among the atmosphere, land surface, oceans, sea ice and other components of the climate system. Following the fast development of science and technology, more and more CSMs, ESMs and related component models have sprung up in the world. For example, more than 50 coupled models participated in the Coupled Model Intercomparison Project Phase 5 (CMIP5), while less than 30 coupled models in the previous CMIP3.

A coupler is an important software tool for model development. It links component models together to construct a coupled model, achieves parallel computation among multiple

component models, controls the integration of the whole coupled model, and even provides a software platform to make scientists and engineers cooperate together. Most state-of-the-art CSMs and ESMs are constructed with a coupler. With more and more component models (e.g., land ice model, chemistry model, and biogeochemical model) to be added into ESMs, couplers become more and more important for model development. Now, there are a number of couplers available in the world that have been widely used for model development, e.g., the Ocean Atmosphere Sea Ice Soil coupling software (OASIS) coupler (Redler et al., 2010; Valcke, 2013a), the Model Coupling Toolkit (Larson et al., 2005; Jacob et al., 2005) (MCT), the Earth System Modelling Framework (Hill et al., 2004) (ESMF), the Flexible Modelling System (FMS) coupler (Balaji et al., 2006), the CPL6 coupler (Craig et al., 2005) designed for the Community Climate System Model version 3 (Collins et al., 2006) (CCSM3), the CPL7 coupler (Craig et al., 2012) designed for the Community Climate System Model version 4 (Gent et al., 2011) (CCSM4) and the Community Earth System Model (Hurrell et al., 2013) (CESM), the Bespoke Framework Generator (Ford et al., 2006; Armstrong et al., 2009) (BFG), etc. Most of these couplers provide typical coupling functions (Valcke et al., 2012a), such as transferring coupling fields between component models, interpolating coupling fields between different grids of component models, and coordinating the execution of component models in a coupled model.

Facing the future development of CSMs and ESMs, we'd like to highlight the following ongoing requirements for coupler development:

1. 3-D coupling. Generally, coupling occurs on the common boundaries or domains between component models. In CSMs, most common interfaces between any two component models are on 2-D horizontal surfaces. For example, the common surface between an atmosphere model and an ocean model is on the skin of the ocean, which is also the bottom of the atmosphere. A CSM also needs 3D coupling, for example, between physics and dynamics in an atmospheric model. In ESMs, component models can share the same 3-D domain. For example, both atmospheric chemistry model and atmosphere model simulate in the 3-D atmosphere space. When coupling them together, especially when their 3-D grids are different, 3-D coupling, including trans-

ferring and interpolating 3-D coupling fields, needs to be achieved. Some existing couplers, such as the OASIS coupler, MCT, CPL6 coupler, CPL7 coupler and ESMF, already provide 3-D coupling function.

2. High-level sharing. A component model can be shared by different coupled model configurations for various scientific research purposes. In different coupled model configurations, the same component model may have different coupling fields and different common interfaces with other component models. For example, given an atmosphere model, when it is used as a standalone component model, there are no coupling fields. When it is used as a component of a CSM, it will provide/obtain 2-D coupling fields to/from other component models. When it is coupled with an atmospheric chemistry model, some 3-D variables with vertical levels become coupling fields. In each coupled model configuration, the atmosphere model can have a branch code version with a specific procedure for providing/obtaining the corresponding coupling fields. When more and more coupled model configurations share the atmosphere model, there will be an increasing number of branch code versions, which will introduce more complexity to the code version control. To facilitate the code version control for sharing a component model, the coupler should enable multiple configurations of coupled models without requiring source code changes to the individual component models and the coupler itself.

3. To develop a model or to achieve a scientific research target, scientists always need to run various kinds of models. For example, to develop an atmosphere model, scientists may want to use various combinations of a single-column model of physical processes, a standalone atmosphere model, a air-sea model, a nested model, a CSM or an ESM. Moreover, scientists may want to cooperatively use the models from different groups or institutions for a scientific purpose. When the software platforms for these models are not identical, scientists have to invest a lot of effort to learn how to handle model simulations on each platform. To facilitate model development and scientific research, a coupler should be able to integrate various component models and

various coupled models on a common model software platform which handles various kinds of model simulations in the same manner, i.e., the same way for creating, configuring, compiling and running model simulations.

4. Better parallel performance. In the future, the coupler's functions will become more computationally expensive for earth system modelling in several aspects. First, the coupler will be required to manage the coupling between more and more component models in a coupled model configuration. Second, 3-D coupling will introduce much higher communication and calculation overhead than 2-D coupling. Third, the resolution of component models coupled together will continually increase. Therefore, it will we increasingly important to improve the parallel performance of a coupler.

Motivated by these requirements, in 2010, we started to design and develop a new coupler named "Community Coupler (C-Coupler)", and finished its first version (C-Coupler1) at the end of 2013. The C-Coupler1 contains a library with functions for coupling a number of component models together and a uniform runtime environment (we call it the C-Coupler platform) with scripts and configuration files for creating, configuring, compiling and running model simulations. Besides the typical coupling functions mentioned above, the C-Coupler1 libraries achieve parallel 3-D coupling with flexible 3-D interpolation, provide the functionality of integrating external algorithms to enable the same code of the C-Coupler and component models shared by various coupled model configurations, and support direct coupling without a specific coupler component to improve the parallel performance. The C-Coupler platform can operate various kinds of model simulations in the same manner. Recently, we successfully used the C-Coupler1 to build several model configurations with different coupling architectures, and made these model configurations share the same code of the component models and the C-Coupler1, and the same C-Coupler platform for simulation.

In this paper, we will introduce the general design of the C-Coupler and show details of the C-Coupler1. The remainder of this paper is organized as follows: Sect. 2 briefly introduces existing couplers; Sect. 3 presents the general design of the C-Coupler; Sect. 4 presents the C-Coupler1 with details; Sect. 5 empirically evaluates the C-Coupler1; Sect. 6

discusses the future works for the C-Coupler development. We conclude this paper in Sect. 7.

2 Brief introduction to existing couplers

In this Section, we will briefly introduce the OASIS coupler, MCT, ESMF, FMS coupler, CPL6 coupler, CPL7 coupler and BFG. More details of these couplers can be found in Valcke et al. (2012a), Redler et al. (2010), Valcke (2013a), Larson et al. (2005), Jacob et al. (2005), Hill et al. (2004), Balaji et al. (2006), Craig et al. (2005), Craig et al. (2012), Ford et al. (2006) and Armstrong et al. (2009).

2.1 The OASIS coupler

CERFACS started to develop the OASIS coupler in 1991. OASIS3 (Valcke, 2013a) is a 2-D version of the OASIS coupler. It has been widely used for developing the European CSMs and ESMs. For example, it has been used in different versions of 5 European CSMs and ESMs that have participated in the CMIP5, e.g., CNRM-CM5 (Voldoire et al., 2011), IPSL-CM5 (Dufresne et al., 2013), CMCC-ESM (Vichi et al., 2011), EC-Earth V2.3 (Hazelger et al., 2011) and MPI-ESM (Giorgetta et al., 2013; Jungclaus et al., 2013). The OASIS3 uses multiple executables for a coupled model, where OASIS3 itself forms a separate executable for data interpolation tasks and each component model remains a separate executable. It provides an ASCII formatted "namcouple" configuration file, which is an external configuration file written by users, to specify some characteristics of each coupling exchange, e.g. source component, target component, coupling frequency and data remapping algorithm. For data interpolation, the OASIS3 can use the remapping weights generated by the 2-D remapping algorithms in the Spherical Coordinate Remapping and Interpolation Package (SCRIP) library (Jones, 1999). The degree of parallelism of the OASIS3 is limited to the number of coupling fields, because each process for the OASIS3 is responsible for a subset of the coupling fields.

OASIS4 is a 3-D version of the OASIS coupler, which supports both 2-D and 3-D coupling. Similar to the OASIS3, the OASIS4 also uses multiple executables for a coupled model and can use the SCRIP for 2-D interpolation. It also provides a "namcouple" configuration file, while the configuration is described in a XML format. As a 3-D coupler, OASIS4 supports transfer and interpolation for 3-D fields. For 3-D interpolation, OASIS4 itself provides pure 3-D remapping algorithms, e.g., 3-D n-neighbor distance-weighted average and trilinear remapping algorithms, and supports 2-D interpolation in the horizontal direction followed by a linear interpolation in the vertical direction. In July 2011, CERFACS stopped the development of OASIS4, and started to develop a new coupler version, the OASIS3-MCT (Valcke et al., 2012b, 2013b), which further improves the parallelism of the OASIS3.

In the following context, we use "namcouple" and "codecouple" to respectively denote the way of specifying characteristics of coupling in configuration files and in source code.

2.2 The Model Coupling Toolkit

The Model Coupling Toolkit (MCT) provides the fundamental coupling functions: data transfer and data interpolation, in parallel. The MCT represents a coupling field into a 1-D array and uses sparse matrix multiplication to achieve data interpolation. Therefore, it can be used for both 2-D and 3-D coupling. For 2-D interpolation, the MCT can use remapping weights generated by the SCRIP. However, it rarely achieves 3-D interpolation due to the lack of 3-D remapping weights generated by existing remapping software. As a result, the MCT is not user-friendly enough in 3-D coupling and users always have to implement 3-D interpolation in the code of component models.

The MCT can be viewed as a library for model coupling. It can be directly used to couple fields between two component models where no separate executable is generated for coupling tasks, and can also be used to develop other couplers, e.g., the OASIS3-MCT, CPL6 coupler and CPL7 coupler.

2.3 The Earth System Modelling Framework

The Earth System Modelling Framework (ESMF, www.earthsystemmodelling.org) is a framework for developing models, which consists of a superstructure for creating components and an infrastructure with common coupling functions. A registry of functions in a coupled system was first advanced by the ESMF. A component model can be registerred to the ESMF after its routines are organized as standard ESMF methods (initialize, run, and finalize). The ESMF can use both single executable and multiple executables for a coupled model. It uses "codecouple" configuration for model coupling. For 2-D interpolation, besides the typical remapping algorithms such as bilinear and first order conservative, the ESMF provides a higher-order finite element-based patch recovery algorithm to improve the accuracy of interpolation. For 3-D interpolation, similar to the OASIS coupler, the ESMF provides several 3-D remapping algorithms, e.g., trilinear, 3-D n-neighbor distance-weighted average and 3-D first-order conservative. For parallelism, the ESMF can remap data fields in parallel and keep bit-identical result when changing the number of processes for interpolation.

In the CMIP5, the coupled model NASA GEOS-5 uses EMSF throughout, and the coupled models CCSM4 and CESM1 use the higher-order patch recovery remapping algorithm provided by the ESMF.

2.4 The FMS coupler

The Flexible Modelling System (FMS) is a software framework that is mainly developed by and used in the Geophysical Fluid Dynamics Laboratory (GFDL) for the development, simulation and scientific interpretation of atmosphere models, ocean models, CSMs and ESMs. The coupling between component models in the FMS is achieved by the FMS coupler in parallel. Similar to the MCT and ESMF, the FMS coupler uses "codecouple" configuration for model coupling, and can keep bit-identical result across different parallel decompositions. One key feature of the FMS coupler is the "exchange grid" (Balaji et al., 2006). Given two component models, the corresponding exchange grid is determined by all vertices in the two grids of these two component models, and the coupling between these two compo-

nent models is processed on the exchange grid. For example, the coupling fields from the source component model are first interpolated onto the exchange grid and then averaged onto the grid of the target component model. In the CMIP5, the CSMs and ESMs (Donner et al., 2011; Dunne et al., 2012) from the GFDL use the FMS as well as its coupler.

2.5 The CPL6 coupler

The CPL6 coupler is the sixth version in the coupler family developed at the National Center for Atmospheric Research (NCAR). It is a centralized coupler designed for the CCSM3, where the atmosphere model, land surface model, ocean model and sea ice model are connected to the unique coupler component and the coupling between any two component models is performed by the coupler component. Through integrating the MCT, the CPL6 coupler achieves the data transfer and data interpolation in parallel. Moreover, it provides a number of numerical algorithms for calculating and merging some fluxes and state variables for component models. It uses multiple executables for a coupled model, where the coupler component forms a separate executable. Similar to the MCT, the CPL6 coupler uses "codecouple" configuration for model coupling. For data interpolation, it always uses the remapping weights generated by the SCRIP.

In the CMIP5, the CPL6 coupler as well as the model platform of CCSM3 has been widely used in Chinese coupled model versions, e.g., FGOALS-g2 (Li et al., 2013a), FGOALS-s2 (Bao et al., 2013), BNU-ESM (Ji et al., 2014), BCC-CSM, FIO-ESM, etc.

2.6 The CPL7 coupler

The CPL7 coupler is the latest coupler version from the NCAR. It has been used for the CMIP5 models CCSM4 and CESM1. It keeps most of characteristics in the CPL6 coupler, such as centralized coupler component, integration of the MCT and flux computation. The most notable advancements from the CPL6 coupler to the CPL7 coupler include: (1) the CPL7 coupler does not use multiple executables but single executable for a coupled model and provides a top-level driver to achieve various processor layout and time sequencing

of the components, to improve the overall parallel performance of the coupled model; (2) a parallel I/O library is implemented in the CPL7 coupler to improve the I/O performance.

2.7 The Bespoke Framework Generator

The Bespoke Framework Generator (BFG) aims to make the coupling framework more flexible in model composition and deployment. BFG2 is the latest version of the BFG. It provides a metadata-driven code generation system where a XML formatted metadata is designed for generating the wrapper code of a coupled model configuration. The BFG metadata is classified into 3 phases: model definition, composition and deployment. The model definition metadata describes the implementation rules of each component model, including <name>, <type>, <language>, <timestep> and <entryPoints>. The <entryPoints> consists of a set of entry points each of which corresponds to a user-specified sub-program unit that can be called by the main program in a coupled model. An entry point can contain a number of <data> elements, each of which corresponds to an argument the entry point. When a <data> element represents an array, the number of dimensions and the bounds for each dimension need to be specified. The composition metadata specifies how component models are coupled together with a number of <set> elements. Each <set> element identifies references to a coupling field, where the references are arguments of the corresponding entry points. The deployment metadata specifies how the coupled model is mapped onto the available hardware and software resources.

The BFG2 designs a "namcouple" approach to separate the code of models from the coupling infrastructure. Although it does not provide coupling functions such as data interpolation, it enables users to choose the underlying coupling functions from other couplers, such as the OASIS coupler. As models are not main programs in a coupled model with the BFG2, single executable or multiple executables can be selected for deploying a coupled model.

2.8 Summary

Some existing couplers already support 3-D coupling, such as the OASIS coupler, MCT, ESMF, CPL6 coupler and CPL7 coupler. However, these 3-D coupling functions should be further improved. The 3-D n-neighbor distance-weighted average, trilinear and linear remapping algorithms used in the OASIS coupler and ESMF can result in low accuracy in the interpolation on the vertical direction. The MCT, CPL6 coupler and CPL7 coupler generally cannot interpolate in 3-D due to a lack of remapping weights. Moreover, most of existing couplers use sparse matrix multiplication to achieve data interpolation. Limited by this implementation, some higher-order remapping algorithms for 1-D vertical direction, such as spline, cannot be used in 3-D interpolation, because these algorithms cannot be achieved purely by sparse matrix multiplication.

Existing couplers that use "codecouple" configuration, e.g., the MCT, ESMF, FMS coupler, CPL6 coupler and CPL7 coupler, are not convenient for sharing the same code version of coupler and component models among various coupled model configurations. For example, when increasing coupling fields or component models based on a coupled model configuration, the code of coupler or component models has to be modified. Although some model code could have compiler directives (such as "#ifdef") for different coupled model configurations, heavy use of compiler directives will make the model code hard to be read and maintained for further development and the change of coupled model configuration requires a recompile of model code. Regarding the OASIS coupler, its “namcouple” configuration, which specifies how to transfer and interpolate each coupling field, can decrease code modification requirements when changing the corresponding characteristics of coupling. However, the configuration files in the OASIS coupler do not specify external algorithms for calculating coupling fields, such as flux calculation algorithms. When changing the procedures for calculating coupling fields, the code of component models always has to be modified. Regarding the BFG, its metadata can further describe and manage user-specified sub-program units. However, the wrapper code requires to be regenerated and recompiled whenever the metadata changes. Moreover, the description of an array, which specifies

the number of dimensions and the bounds for each dimension, is not flexible enough for changing parallel decompositions of models.

There are several model software platforms corresponding to existing couplers which have been successfully used for model development, such as the CCSM3 platform corresponding to the CPL6 coupler, the CCSM4/CESM platform corresponding to the CPL7 coupler, and the FMS. These platforms can configure, compile and run different kinds of model configurations for simulation. The CCSM4/CESM platform can run standalone component models, CSMs, ESMs, etc. However, to make a new standalone component model run on the CCSM4/CESM platform, users have to dramatically modify the code of the component model. For example, when integrating an ocean model version, i.e., MOM4p1 (Griffies et al, 2010), onto the CCSM4/CESM platform for a standalone ocean model configuration without coupling with other component models, the code of the MOM4p1 needs to be dramatically modified to use the CPL7 coupler as the driver.

Improving parallel performance is always a focus in coupler development. It concerns the parallel performance of both the coupler and the whole coupled model. Similarly, we are concerned with the parallel performance of both the coupler and coupled models in the long-term development of the C-Coupler.

3 General design of the C-Coupler

In this section, we will briefly introduce the general design of the C-Coupler. The C-Coupler can be viewed as a family of the community coupler developed in China, and the C-Coupler1 is the first version following the general design. The future versions of the C-Coupler will also follow the general design. In the following context, we first define a general term of "*experiment model*", and then introduce the architecture of the experiment models with the C-Coupler and the general software architecture of the C-Coupler.

3.1 A general term for the C-Coupler: experiment model

An experiment model is a model configuration which can run on the C-Coupler platform for simulations. It consists of a certain set of configuration files and a certain set of model code, with a certain set of rules for precompiling. Generally, an experiment model can be any kind of model configuration, such as a single-column model, a standalone component model, a regional coupled model, an air-sea coupled model, a nested model, a CSM, an ESM, etc.

3.2 Architecture of the experiment models with the C-Coupler

To achieve the target of integrating various models on the same common model software platform for a high-level sharing of the component models and for facilitating the construction of a new experiment model, we have designed an architecture for the experiment models with the C-Coupler. Fig. 1 shows an example of this architecture with a typical CSM, where "ATM", "ICE", "LND" and "OCN" stand for the component models. The key ideas of this design include:

1. All experiment models share the same code of the C-Coupler. Given an experiment model, there could be a separate coupler component which manages the coupling between the component models, while direct coupling without a coupler component is also supported. For example, the red lines in Fig. 1 stand for the direct coupling, where no separate executable is generated for the coupling tasks, and all coupling tasks, such as data transfer, data interpolation and flux computation, are performed through the uniform C-Coupler Application Programming Interfaces (APIs) called by the component models. Compared to the approach with a separate coupler component, the direct coupling can reduce the number of data transfers for better parallel performance, but it can lower code modularity. Users can select a separate coupler component, direct coupling or hybrid for constructing a coupled model configuration. For example, users can use a separate coupler component for a CSM or ESM for better code modularity and use direct coupling for an air-sea coupled model for better parallel performance.

2. When a component model is shared by multiple experiment models, it keeps the same code version in all these experiment models. The code of coupling interfaces in the component model only specifies the input fields that the component model wants and the output fields that the component model can provide, but does not specify how to get the input fields and how to provide the output fields. For example, the source component models, the target component models and the flux calculation of the coupling fields are not specified in the code of the coupling interfaces.

3.3 General software architecture of the C-Coupler

Under the guidance of the key ideas, we designed the general software architecture of the C-Coupler, which consists of a configuration system, a runtime software system and a coupling generator, as shown in Fig. 2. In the following context of this subsection, we will further introduce the configuration system and runtime software system. The coupling generator that has not been developed in the C-Coupler1 will be further discussed in Sect. 6.

3.3.1 Configuration system

In different experiment models, a component model always has different procedures for the input and output fields. For example, given an atmosphere model, in its standalone component model configuration, the ocean surface state fields (such as sea surface temperature) are obtained from the I/O data files, while in an air-sea coupled model configuration, the ocean surface state fields are obtained from the ocean model through coupling. Moreover, in these two model configurations, the algorithms for computing the air-sea flux (such as evaporation, heat flux and wind stress) can be different. To make the same code version of a component model shared by various experiment models, the C-Coupler should make a procedure adaptively achieve different functions for different model configurations without code modification. We therefore designed a configuration system in the C-Coupler. Besides the functionality achieved by the "namcouple" configuration file in the OASIS coupler and

the BFG, the configuration system of the C-Coupler can further specify procedures for coupling. In the following context of this paper, we call such procedures *runtime procedures*. A runtime procedure consists of a list of algorithms called as *runtime algorithms*. The runtime algorithms can be classfied into two categories: internal algorithms and external algorithms. The internal algorithms are implemented intra the C-Coupler, including the data transfer algorithms, data remapping algorithms, data I/O algorithms, etc. The external algorithms are always provided by the component models, coupled models and users. They could be the private algorithms of a component model or common algorithms such as flux calculation algorithms which can be shared by various experiment models.

The configuration system manages the configuration files of software modules and the runtime configuration files of model simulations. The software modules include component models, experiment models and external algorithms. In detail, the configuration files of a component model specify some characteristics of the component model, e.g., the input and output fields, how to generate the input namelists, how to compile the code of the model, etc. The configuration files of an experiment model specify how to organize the components, e.g., the components in the experiment model, how each component gets the input fields and provides the output fields, etc. The configuration files of an external algorithm specify the input and output fields of the algorithm. The runtime configuration files specify how to run an experiment model for a simulation, e.g., how to organize the internal algorithms and external algorithms into the runtime procedures for the input and output fields of the components, the coupling frequencies, the start time and stop time of the simulation, etc.

In detail, the keyword (algorithm name) of each runtime algorithm in a runtime procedure is listed in a simple configuration file. This implementation does not reduce the readability of model code but can make the coupling process more easily understood. When a runtime procedure is called by model code, the function pointer (some programming languages such as C++ support function pointer) corresponding to each runtime algorithm will be found according to the keyword and then the runtime algorithms can be executed one by one. As the function pointer of a runtime algorithm requires to be searched only one time during the whole simulation, the extra overhead introduced by the approach of runtime

procedure is trivial. The same runtime procedure can achieve different functions in different model configurations through modifying the list of runtime algorithms that is recorded in a configuration file, without modification of model code.

3.3.2 Runtime software system

The runtime software system can be viewed as a common, flexible and extendible library for constructing experiment models and for running model simulations. It enables various kinds of experiment models to share the same code of the C-Coupler.

First, it provides a set of uniform APIs for integrating component models. With these APIs, a component model can register the model grids, parallel decompositions and input/output fields, and get/put the input/output fields from/to I/O data files or other components, etc.

Second, similar to the ESMF, the runtime software system supports the registry of functions in a coupled system. It provides uniform APIs for integrating external algorithms. A component model can register its private subroutines as external algorithms of the C-Coupler. Common algorithms like flux calculation algorithms can also be registered as external algorithms. Therefore, the runtime software system is an extendible library which can integrate more and more common external algorithms, and thus users can have more choices for model simulations. For example, given an air-sea coupled model, if there are several different algorithms for calculating air-sea flux, users can select one of them in a simulation, or two or more of them for sensitivity experiments.

Third, the runtime software system consists of a number of managers (shortened as MGR in Fig. 2), including a communication manager, grid manager, parallel decomposition manager (shortened as decomposition MGR in Fig. 2), remapping manager, timer manger, data manager, restart manager, runtime process manager (shortened as process MGR in Fig. 2), etc. In detail, the communication manager is responsible for allocating and managing the communicators of each component and the whole experiment model. The grid manager manages the grids registered by the component models. The grid can be 1-D, 2-D, 3-D even 4-D. The parallel decomposition manager manages the parallel decompositions registered by the component models. The remapping manager manages the remapping

algorithms used for coupling. Users can select different remapping algorithms in different simulations of the same experiment model. The timer manager provides timers for triggering the execution of internal and external algorithms. Each algorithm has a timer, and it is executed only when its timer is on. The timer manager can also provide time information for the whole experiment model through the corresponding APIs. The data manager provides uniform APIs for getting the attributes and memory space of fields. A component model can register model fields (including the memory space) as external fields to the data manager. For the internal fields, the data manager will allocate their memory space automatically. The restart manager is responsible for reading fields from the restart I/O data files in a restart run of a model simulation, and writing fields into the restart I/O data files when the timer for the restart writing is on. The runtime process manager manages the internal algorithms and the registered external algorithms, organizes these runtime algorithms into a number of runtime procedures, and executes the runtime procedures in a model simulation.

Modularity is an important characteristic of software quality. A coupler is a software tool for improving modularity of coupled models. The software architecture with a set of managers targets better modularity of the runtime software system. It can enhance independence of each manager, so as to facilitate the advancement of the C-Coupler. For example, when one manager is upgraded, the whole C-Coupler is upgraded. Moreover, it can enhance the testing and reliability for each manager. For example, diagnoses can be inserted into the source code of the C-Coupler for detecting potential errors in the input and output of a manager.

4 C-Coupler1: first released version of the C-Coupler

The C-Coupler1 is the first version of the C-Coupler for public use. Many ideas and concepts from existing couplers have been considered for its design and implementation. As the initial version of the C-Coupler, it does not include the coupling generator currently. However, we carefully developed the configuration system and the runtime software system, which makes the C-Coupler1 achieve most of characteristics in the general design of

the C-Coupler and reach the target of sharing the same code version of the C-Coupler and component models among different experiment models. Moreover, we designed and developed the C-Coupler platform, which enables users to operate various model simulations in the same manner.

The C-Coupler1 is a 3-D coupler, where the coupling fields can be 0-D, 1-D, 2-D, or 3-D. It uses multiple executables for the coupled models, each component of which has a separate executable. It can be used to construct a simple coupler component with a few lines of code, and can also be used for direct coupling between component models without separate executable specifically for the coupling tasks. It does not use the "codecouple" configuration but develops a powerful configuration system. As a 3-D coupler, it can interpolate both 2-D and 3-D fields. The runtime software system of the C-Coupler1 has been parallelized using the Message Passing Interfaces (MPI) library, while the bit-identical result is kept when changing the number of processes for the C-Coupler1.

In the following context of this section, we will respectively present the runtime software system, configuration system and C-Coupler platform in the C-Coupler1, and then introduce how to couple a component model and the enhancement for reliability of software.

4.1 The runtime software system

The runtime software system is a parallel software library programmed mainly in C++ for better code modularity. It provides APIs mainly in Fortran because most of component models for earth system modelling are programmed in Fortran. In the following context, we will introduce the technical features of the runtime software system, including the APIs, each manager, and parallelization.

4.1.1 The APIs

Table 1 lists out the APIs provided by the C-Coupler1, which can be classified into four categories: the main driver, registration, restart function and time information. Besides

the brief description of each API in Table 1, we would like to further introduce two APIs, *c_coupler_execute_procedure* and *c_coupler_register_model_algorithm*.

The API *c_coupler_execute_procedure* takes the name of a runtime procedure as an input parameter, while the algorithm list for the runtime procedure is specified in the corresponding runtime configuration files. Thus, a runtime procedure can keep the same name in various experiment models, and users can make the same runtime procedure perform different works through modifying the runtime configuration files that can be viewed as a part of input of a model simulation. As a result, a component model can keep the same code version in various experiment models sharing it.

Almost all APIs are in Fortran the *c_coupler_register_model_algorithm*. The *c_coupler_register_model_algorithm* is in C++ because most Fortran versions do not support function pointer. A private external algorithm (or subroutine) registered by a component model through this API does not have explicit input and output fields. The input and output of such a algorithm are specified implicitly in the code (always Fortran code) through using modules' public variables. In the future version of the C-Coupler, we may enable the private external algorithms to have explicit input and output fields.

4.1.2 The implementation of managers

The communication manager

The communication manager adaptively allocates and manages the MPI communicators for the MPI communications intra and between the components of an experiment model. It also provides some utilities for other managers, such as getting the id of a process in the communicator of a component or in the global communicator.

The grid manager

The grid manager utilizes a multi-dimensional **Co**mmon **R**emapping (CoR) software (Liu et al., 2013a, b; it can be downloaded through "*svn –username=guest –password=guest co http://thucpl1.3322.org/svn/coupler/CoR1.0*") to manage the grids with dimensions from

1-D to 4-D. In a model simulation, the grids of component models are registered to the grid manger with a script of the CoR, where the grid data (such the latitude, longitude and mask corresponding to each grid cell) is always read from I/O data files. Besides the support of multiple dimensions of grids, another advantage of using the CoR for grid management is that it can detect the relationship between to two grids, for example, a 2-D grid is the horizontal grid of a 3-D grid with vertical levels. Moreover, for horizontal 2-D grid, the CoR can support logically rectangular and unstructured grids, such as a cubic spherical grid.

The parallel decomposition manager

Most of component models for earth system modelling have been parallelized using the MPI library, where the whole domain of each component model, which always is a 3-D grid with vertical levels, is decomposed into a number of sub domains for parallelization and each process of the component model is responsible for a sub domain. We call the decomposition from the whole domain into sub domains as parallel decomposition. In the C-Coupler1, each parallel decomposition managed by the parallel decomposition manager is based on a 2-D horizontal grid that has been registered to the grid manager, while the parallel decomposition on the vertical sub-grid of the 3-D grid is not supported yet. To register a parallel decomposition, the C-Coupler1 uses an implementation derived from the MCT: each process of a component model enumerates the global index (the unique index in the whole domain) of each local cell (each cell in the sub domain of this process) in the corresponding horizontal grid. A component model can register multiple parallel decompositions on the same horizontal grid, while each parallel decomposition has a unique name which is treated as the keyword of it.

The remapping manager

The remapping manager utilizes the CoR to achieve the data interpolation function. There are several remarkable advantages of using the CoR for interpolation. First, it can help the Coupler1 to remap the field data on 1-D, 2-D and 3-D grids. Second, it can generate

remapping weights using its internal remapping algorithms and can also use the remapping weights generated by other software, such as the SCRIP. Third, it is designed to be able to interpolate field data between two grids with any structures, which makes the C-Coupler1 able to be used more extensively. Similar to other couplers such as the OASIS coupler, MCT, ESMF and CPL6/CPL7 coupler, the C-Coupler1 can utilize the remapping weights generated offline by remapping software such as the CoR and SCRIP. The I/O data files for the offline remapping weights are specified in a script of the CoR, the same script for registering the grids. In different simulations of the same experiment model, users can select different remapping algorithms through modifying the script.

Considering that the vertical grids (such as SIGMA-P grid) of component models may be changed during the model execution, the 1D remapping weights for the "2-D + 1-D" interpolation can be generated online in parallel by the C-Coupler1 in the execution of a model simulation. This online weight generation can scale well because the vertical grid is not decomposed for parallelization and each process can generate the 1-D remapping weights independently.

The CoR makes the C-Coupler1 more flexible in 3-D interpolation when compared with existing couplers. First, the CoR supports the "2-D + 1-D" approach to interpolate data between two 3-D grids, where a 2-D remapping algorithm is used for the interpolation between the 2-D horizontal sub-grids and a 1-D remapping algorithm is used for the interpolation between the 1-D vertical sub-grids. Given that there are M 2-D remapping algorithms and N 1-D remapping algorithms, there are MxN selections for 3-D interpolation. Second, the CoR supports both sparse matrix multiplication and equation group solving for interpolation calculation. Thus, it can provide some higher-order remapping algorithms such as spline that requires solving a tri-diagonal equation group. Moreover, it makes the C-Coupler1 able to handle some unstructured grids. For example, when the source grid is a cubic spherical grid, the bilinear remapping algorithm in the SCRIP cannot be used, while the CoR can handle this case.

At the present time, there are several remapping algorithms available in the CoR. For the 2-D horizontal interpolation, the CoR provides three remapping algorithms: first order

conservative, bilinear and 2-D n-neighbor distance-weighted average. For the 1-D vertical interpolation, the CoR provides two remapping algorithms: linear and spline. Currently, there are no pure 3-D remapping algorithms implemented and 3-D interpolation is achieved by the "2-D + 1-D" approach. In the detailed implementation for this approach, the 2-D and 1-D remapping weights are managed separately, and there could be a number of matrixes for 2-D remapping weights and for 1-D remapping weights respectively. For some cases, the matrixes for the 2-D and 1-D remapping weights can be merged into one big matrix of 3-D remapping weights. We do not select this implementation because it can dramatically increase the calculation for 3-D interpolation. Moreover, this implementation cannot handle the vertical spline interpolation which requires equation group solving. For each spline interpolation on the vertical direction, the coefficient matrix of the equations can be precalculated by the CoR when generating the offline remapping weights. During the model execution, the C-Coupler1 will call the CoR functions to solve the equation group. The vertical interpolation scales well because the vertical grid is not decomposed for parallelization and each process can handle the vertical interpolation independently.

The timer manager

The timer manager makes the component models in a coupled model advanced in simulation time in an orderly fashion. In the runtime software system, each coupling field can have a set of timers for periodically triggering the operations on it. For example, a coupling field always has a timer for data transfer and a timer for data interpolation. Moreover, each external algorithm has a timer to periodically trigger its execution. There are three elements in a timer: the unit of frequency, the count of frequency and the count of delay. The unit and the count of frequency specify the period of the timer. The count of delay specifies a lag of the time during which the corresponding operation or algorithm will not be executed. The unit of frequency can be "years", "months", "days", "seconds" and "steps", where "steps" means the time step of calling the API c_coupler_advance_timer. For example, timer < 10, steps, $15 >$ means that the corresponding operation or algorithm will be executed at the steps with number $10 \times N + 15$, where N is a nonnegative integer. Sequential and concur-

rent runs between component models can be achieved through cooperatively setting the delay of the timers.

Besides managing all the timers, the timer manager provides interfaces for getting the information of model time (e.g., calendars) during a simulation.

The data manager

The fields managed by the data manager include the external fields which are registered by the components with APIs and the internal fields which are automatically allocated by the data manager. A coupling field, such as Sea Surface Temperature (SST), can have different instances in a coupled model configuration due to different parallel decompositions and different grids. There is a keyword for each field instance, which consists of the name of the field, the name of the corresponding component model, the name of the corresponding parallel decomposition and the name of the corresponding grid. To define a field instance, the corresponding four names must have been defined or registered to the runtime software system. For an external field instance, these four names are specified when a component model registers this field instance through calling the corresponding API, while for an internal field instance, these four names are specified in configuration files. For the scalar field which is not on a grid, the corresponding parallel decomposition and grid are marked as "NULL".

All instances of a coupling field share the same field name. The field values in one instance can be transformed into another instance through data transferring between two component models and data interpolation intra a component model. All component models in a coupled model share the names of the coupling fields. All legal field names are listed in the configuration files, with other attributes such as the long name (also known as the description of the field) and the unit.

The data manager achieves several advantages beyond existing solutions. First, a 2-D field and a 3-D field can share the same 2-D parallel decomposition while their corresponding grids are different, where the 2-D grid corresponding to the 2-D field is a sub grid of the 3-D grid corresponding to the 3-D field. Second, the data manager unifies the man-

agement of different kinds of field instances, such as with different grids, different parallel decompositions, and different data types (i.e., integer and floating point). As a result, the data transfer algorithm can transfer different kinds of field instances at the same time for better communication performance. Third, the data manager helps improve the reliability for model coupling. For example, the remapping manager can examine whether the grids of source fields and target fields match the grids of the remapping weights.

The restart manager

For reading/writing fields from/into the restart I/O data files, the restart manager iterates on each field managed by the data manager. For the internal fields, the restart manager can automatically detect the fields which are necessary for restarting the model simulation. For an external field, the corresponding component can specify whether this field is necessary for restarting when registering it with the C-Coupler API. As a result, a component model has more selections for achieving the restart function. It can still use its own restart system or register all fields for restart as external fields to the data manager.

The runtime process manager

The runtime process manager is responsible for running the list of runtime algorithms in each runtime procedure during a model simulation. Besides the external algorithms including the private algorithms registered by the component models and the common flux calculation algorithms, there are several algorithms internally implemented in the runtime software system, e.g., the data transfer algorithm, data remapping algorithm, data I/O algorithm, etc. The data transfer algorithm is responsible for transferring a number of fields from one component to another. The fields transferred by the same data transfer algorithm can have different number of dimensions, different data types, different parallel decompositions, different grids, different frequency of transfer, etc. The data transfer algorithm packs all fields that are to be transferred at the current time step into one package to improve the communication performance.

The data remapping algorithm uses the corresponding algorithm in the CoR as a kernel for implementation. It can remap several fields at the same time for better parallel performance. Multiple fields in a data remapping algorithm share the same parallel decomposition, the same grid and the same timer, while the data types (i.e., single-precision and double-precision floating point) can be different.

The data I/O algorithm currently utilizes the serial I/O to read/write multiple fields which are managed by the data manager from/into the data I/O files. The multiple fields in a data I/O algorithm share the same timer, while can have different parallel decompositions, different grids and different data types. The fields of a data I/O algorithm are specified in the corresponding runtime configuration file. For the future version of the C-Coupler, we will further improve the I/O performance with parallel I/O for higher-resolution models.

4.1.3 Parallelization

As aforementioned, the runtime software system has been parallelized using the MPI library and achieves bit-identical result when changing the number of processes. Here we would like to further introduce some details, including the parallelization of the data transfer algorithm, the parallelization of the data remapping algorithm and the default parallel decomposition.

Parallelization of the data transfer algorithm

This parallelization is derived from existing couplers such as the MCT. For a coupling field instance transferred by the data transfer algorithm, it has a parallel decomposition in the source component and another parallel decomposition in the target component, and these two parallel decompositions share the same horizontal grid. A process in the source component will transfer the data of this field to a process in the target component only when the corresponding sub domains on these two processes have common cells. As this implementation does not involve collective communications and there are always multiple processes

to execute the source component and the target component respectively, the data transfer algorithm can transfer the coupling fields in parallel.

Parallelization of the data remapping algorithm

The data remapping algorithm interpolates a number of fields from the source grid to the target grid. To make the fields interpolated in parallel, the C-Coupler1 uses an approach from the MCT, which generates an internal parallel decomposition for rearranging coupling fields before interpolating. The internal parallel decomposition is on the source grid and determined by the remapping weights and the parallel decomposition corresponding to the target grid. For example, given that global cell y of the target grid is assigned to process p, and given a remapping weight <x,y,w> where x specifies a global cell in the source grid and w is a weight value, the internal parallel decomposition for process p will include the global cell x of the source grid. After rearranging the fields according to the internal parallel decomposition using the data transfer algorithm, process p can interpolates the fields locally. This implementation avoids the reduction for sum between multiple processes of a component model, so as to make the data remapping algorithm achieve bit-identical result when using different numbers of processes. Although it can increase the overhead when rearranging coupling fields, the collective communication for the reduction for sum between multiple processes can be avoided.

Default parallel decomposition

In an experiment model, not all parallel decompositions are specified by the component models through registration. For example, the parallel decompositions in a coupler component are not determined by any component model. Therefore, the runtime software system provides a default parallel decomposition. Given the number of processes N, the default parallel decomposition partitions a horizontal grid into N distinct sub domains without common cells, and the number of cells in each sub domain is around the average number.

4.2 Configuration system

As we did not develop the coupling generator in the C-Coupler1, we did not develop the configuration files of experiment models accordingly. In the following context of this sub section, we will respectively introduce the configuration files of the component models, the configuration files of the external algorithms and the runtime configuration files of the model simulations.

4.2.1 The configuration files of the component models

Each component model has a set of configuration files which specify the following information:

1. How to generate input namelist files. The generation of input namelist files is specified in a script named *config.sh*. When users configure a model simulation, *config.sh* will be invoked to generate the namelist files.

2. Where the source code is. The locations of source code are specified in a script named *form_src.sh*. A location can be a specific code file or a directory which means that all code files under it require to be compiled. When users compile the model code for a simulation, *form_src.sh* will be invoked.

3. How to compile the model code. The compilation of model code is specified in a script name *build.sh*. It enhances flexibility for compilation: a component model can use the compilation utility provided by the C-Coupler platform or use its own compilation system. When users compile the model code for a simulation, *build.sh* will be invoked.

4. Compiling options for the component model, which are specified in a configuration file named compiler.cfg.

5. The information of the field instances that will be registered to the C-Coupler1. Fig. 3 shows an example for the corresponding configuration file, where each line corresponds to a field instance. The first column specifies the field names, the second

column specifies the parallel decompositions and the last column specifies the grids. A component model can register multiple instances for the same field, on different grids or different parallel decomposition. From Fig. 3, we can find that *atm_2D_grid1* is a sub grid of *atm_3D_grid1* because they can share the same parallel decomposition *atm_2D_decomp1*. Similarly, *atm_2D_grid2* is a sub grid of *atm_3D_grid2*. This configuration file will be queried when the atmospheric model registers a field instance.

4.2.2 The configuration files of the external algorithms

As shown in Fig. 4, an external algorithm has a configuration file for the main information. In the main information (Fig. 4a), the first line is the algorithm name for searching the corresponding function pointer, and the second line specifies a timer for triggering the execution of the external algorithm. The last two lines specify the name of two configuration files for the input and output field instances respectively. For the private external algorithm that does not have input and output fields, these two lines are set to "*NULL*". For a field instance that are both an input and output of the external algorithm, it should be referenced in the two configurations files. Fig. 4b shows an example for how to describe the input (or output) field instances, where each line corresponds to a field instance. Columns 1 4 specify the keyword for each field instance, while the last column specifies the data type.

4.2.3 The runtime configuration files of the model simulations

Corresponding to the design and implementation of the runtime software system, the runtime configuration files contain the following information about a model simulation: 1) the configuration files of the corresponding components and external algorithms; 2) a CoR script to specify the grids and the weights for remapping algorithms; 3) the configuration files for each runtime procedure in each component that will be further illustrated in Sect. 5.1; 4) the namelist of the model simulation that is common to all components, including the start time, stop time, run type (initial run or restart run), etc.

4.2.4 Summary

Similar to the OASIS coupler and BFG, the C-Coupler develops a "namcouple" configuration system for specifying the coupling characteristics of an experiment model. The BFG defines the metadata of coupling characteristics in 3 phases: model definition, composition and deployment. Regarding the C-Coupler, the configuration files of a component model function similarly to the model definition metadata and the configuration files of an experiment model function simularly to metadata composition. Generally, the "namcouple" implementation in the C-Coupler is different from that in the OASIS coupler and BFG as follows:

1. The procedure registration system with runtime procedures and runtime algorithms can support a wide number of coupled model configurations while maintaining the same codebase for component models.

2. Grids and parallel decompositions are referenced by name in the configuration system. As a result, most of configurations files of an experiment model can keep the same when changing the parallel settings, model resolutions or model grids.

3. Under the help from the CoR, the configuration system can bridge relationship between parallel decompositions, grids, remap weights, etc. Therefore more kinds of diagnoses can be conducted to make the C-Coupler and experiment models more reliable. For example, the remapping manager can examine whether the grids of source fields and target fields match the grids of remapping weights.

4.3 The C-Coupler platform

To facilitate operating the simulation of various experiment models with the C-Coupler1, we designed and developed the C-Coupler platform. Fig. 5 shows its general architecture. It manages the input data and the software modules for the experiment models, including standardized component models, external algorithms, runtime configuration files of the model simulations and the runtime software system. There are four steps for operating a

model simulation on the C-Coupler platform: “create case”, “configure”, “compile” and “run case”. “create case” means creating a model simulation. There are two approaches for creating a model simulation: creating a default simulation of an experiment model using the script “create_newcase” and creating a model simulation from an existing model simulation. The C-Coupler platform facilitates the second approach. At each time of “configure” of a model simulation, a package of the corresponding experimental setup is automatically generated and stored. This package can be used to reproduce the existing model simulation or develop new model simulations. After creating a model simulation, users can modify the experimental setup, such as the namelist, parallel settings, hardware platform, compiling options, output settings, start and stop time, etc. After the modification of the experiment setup, users should “configure” the model simulation, and then users can “compile” and “run case”. For various experiment models on various hardware platforms, users can use the same operations for various model simulations. For more information about the C-Coupler platform, please read its users’ guide (Liu et al., 2014a).

The model platforms of the CCSM3 and CCSM4/CESM have demonstrated that the four steps, i.e., “create case”, “configure”, “compile” and “run case”, are sufficient and user-friendly for model simulations. We therefore used a similar four-step design for the C-Coupler platform. The most unique feature of the C-Coupler platform is the enhancement for reproducibility of bit-identical simulation result for earth system modelling. Please refer Liu et al. (2014b) for details.

4.4 How to couple a component model

Generally, it takes the following steps to couple a component model with the C-Coupler1:

1. Generate remapping weights if necessary.
2. Write a CoR script to register the grids of the component model and read in the remapping weights.
3. Initialize the C-Coupler runtime software system and get the MPI communicator through calling the API c_coupler_initialize, finalize the runtime software system

through calling the API c_coupler_finalize and advance the simulation time through calling the API c_coupler_advance_timer in the source code of the component model.

4. Register each parallel decomposition to the C-Coupler through calling the API c_coupler_register_decomposition, register each field instance through calling the API c_coupler_register_model_data, and provide or obtain coupling fields through calling the API c_coupler_execute_procedure in the source code of the component model.

5. Write configuration files (Sect. 4.2) for the component model, to integrate the component model into the C-Coupler platform.

We note that the steps similar to the above 1, 3 and 4 are always required when coupling a new component model with other couplers. Steps 2 and 5, which produce configuration files for coupling, are specific to the C-Coupler. In the C-Coupler1, these configuration files for the coupling procedures are written manually by scientists. In the future C-Coupler2, they will be generated automatically by the coupling generator.

4.5 Enhancement for reliability of software

Reliability is an important characteristic of software quality. To make the C-Coupler1 and experiment models more reliable, more than 900 diagnoses are inserted into the source code (about 30,000 lines) of the C-Coupler1. These diagnoses focus on the following functions:

1. Trace the behavior of coupling during a model simulation. The C-Coupler1 can trace the flow of coupling for each component model and the input/output fields for each runtime algorithm at each coupling step.

2. Detect errors (software bugs) in the C-Coupler1. There are more than 600 diagnoses for detecting potential errors in the input and output of managers in the runtime software system of the C-Coupler1.

3. Detect errors in the configuration files. The C-Coupler1 can check whether a configuration file is right in format and content, and whether configuration files are consistent

between component models. For example, given that a runtime algorithm transfers a number of coupling field instances from component model A to B, each of A and B has a configuration file for this data transfer. The C-Coupler1 will check the consistency between these two configuration files.

4. Detect errors in the C-Coupler API calls. When a component model calls a C-Coupler API, the C-Coupler1 will check the consistency between the API call and the corresponding configuration files. Moreover, the C-Coupler1 can detect the field instances which are required as input for coupling but not provided by any component models.

In a default setting, the trace for coupling behavior is disabled because it is time-consuming and will produce a large mount of data. This detection of errors in the C-Coupler1, configuration files and C-Coupler API calls is always enabled because it only slows down the initialization for the runtime software system.

5 Evaluation

To evaluate the C-Coupler1, we used it to construct several experiment models, including the FGOALS-gc, GAMIL2-sole, GAMIL2-CLM3, MASNUM-sole, POM-sole, MASNUM-POM and MOM4p1-sole. The FGOALS-gc is a CSM version based on the CSM FGOALS-g2 (Li et al., 2013a), where the original CPL6 coupler in the FGOALS-g2 is replaced by the C-Coupler1. The GAMIL2-sole is a standalone component model configuration of the atmosphere model GAMIL2 (Li et al., 2013b), the atmosphere component in the FGOALS-g2, which participated in the Atmosphere Model Intercomparison Project (AMIP) in the CMIP5. The GAMIL2-CLM3 is a coupled model configuration consisting of the GAMIL2 and the land surface model CLM3 (Oleson et al., 2004). The MASNUM-sole is a standalone component model configuration of the wave model MASNUM (Yang et al., 2005). The POM-sole is a standalone component model configuration based on a parallel version of the ocean model POM (Wang et al., 2010). The MASNUM-POM is a coupled model configuration consisting

of the MASNUM and POM. The MOM4p1-sole is a standalone component model version of the ocean model MOM4p1 (Griffies et al, 2010).

In the following context of this section, we will evaluate the C-Coupler1 in several aspects, including the coupler component, direct coupling, 3-D coupling, code sharing, parallel performance and the work amount for integrating a standalone component model version onto the C-Coupler platform.

5.1 Coupler component and direct coupling

To construct the FGOALS-gc, we use the C-Coupler1 to develop a separate and centralized coupler component according to the CPL6 coupler. Then the four component models in the FGOALS-g2, i.e., atmospheric model GAMIL2, land surface model CLM3, ocean model LICOM2 (Liu et al., 2012) and an improved version of the sea ice model CICE4 (Liu, 2010), are coupled together with the C-Coupler1 coupler component. All flux calculation algorithms in the CPL6 coupler are integrated into the C-Coupler1 as external algorithms. These algorithms can be treated as public algorithms that can be shared by other experiment models. Fig. 6 shows the main driver of the coupler component in the FGOALS-gc. It is very simple with a few lines of code, most of which call the C-Coupler APIs, while the main driver of the CPL6 coupler has about 1,000 lines of code. This is because the coupling flow derived from the CPL6 has been described by a set of configuration files that are manually defined by us. Fig. 7 shows a part of the runtime configuration file of the algorithm list for the coupler component. In detail, each line is the keyword of a runtime algorithm. The first column in the keyword specifies the type of the runtime algorithm. “Transfer” denotes a data transfer algorithm, “remap” denotes a data interpolation algorithm, and “normal” denotes an external algorithm. The second column specifies the configuration file of the runtime algorithm. In sum, this runtime configuration file clearly lists out 91 runtime algorithms. For each runtime algorithm, there are configuration files to specify the input and output field instances.

Fig. 8 shows the runtime configuration file of the runtime procedures for the coupler component, where each line corresponds to a runtime procedure. All runtime procedures share the same runtime configuration file of the algorithm list in Fig. 7. For a runtime procedure,

the first column is the procedure name, which is the input parameter when model code calls the C-Coupler API c_coupler_execute_procedure. The second column and third column respectively specify the start index and end index of the runtime algorithms in the algorithm list in Fig. 7. Then a runtime procedure can find the keywords of all runtime algorithms in it.

Our tests show that the FGOALS-gc achieves the same (bit-identical) simulation result with the FGOALS-g2. This result demonstrates that the C-Coupler1 can be used to construct a coupler component for a complicated coupled model, such as a CSM, without changing the simulation result. The FGOALS-g2 and FGOALS-gc can be downloaded through "*svn–username=guest–password=guest co http://thucpl1.3322.org/svn/coupler/CCPL_CPL6_consistency_checking*."

When constructing the experiment models GAMIL2-CLM3 and MASNUM-POM, we did not build a separate coupler component but used the direct coupling where no separate executable is generated for coupling. For the GAMIL2-CLM3, as the GAMIL2 and CLM3 share the same horizontal grid, there is only data transfer between them. For the MASNUM-POM, as the grid of the MASNUM is different from the grid of the POM, there are both data transfer and data interpolation between these two component models.

5.2 Parallel 3-D coupling

In the MASNUM-POM, there is only one coupling field, the wave-induced mixing coefficient (Qiao et al., 2004), a 3-D field from the MASNUM to POM. As the horizontal grids and vertical grids in these two component models are different, 3-D interpolation is required during coupling. In detail, we use the CoR to generate the remapping weights for the 3-D interpolation. The corresponding 3-D remapping algorithm is generated through cascading two remapping algorithms: a bilinear remapping algorithm for the horizontal grids and a 1-D spline remapping algorithm for the vertical grids. For the 1-D vertical interpolation, the MASNUM and POM have different kinds of vertical grids: a z grid for the MASNUM and a *sigma* grid for the POM.

As introduced in Sect. 5.1, the MASNUM-POM uses direct coupling without a coupler component. As the resolution of the MASNUM is lower than the resolution of the POM,

we put the calculation of the 3-D interpolation in the runtime procedure of the POM, in order for better parallel performance. Therefore, the 3-D interpolation shares the same processes with the POM. When the POM runs with multiple processes, the 3-D interpolation is computed in parallel. Our evaluation shows that the 3-D interpolation keeps the same (bit-identical) result with different numbers of processes.

5.3 Code sharing

The experiment models FGOALS-gc, GAMIL2-CLM3 and GAMIL2-sole share the same atmosphere model GAMIL2. In the FGOALS-gc, the surface fields required by the GAMIL2 are provided by other component models and computed by the coupler component with the C-Coupler1. In the GAMIL2-CLM3, the surface fields required by the GAMIL2 are provided by the CLM3 and the I/O data files which contain ocean fields and sea ice fields, and computed by the private flux algorithms in the GAMIL2. The GAMIL2-sole is similar to the GAMIL2-CLM3, while the difference is that the GAMIL2-sole directly calls a land surface package to simulate the surface fields from the land. Therefore, in these three experiment models, the GAMIL2 has different procedures for the surface fields.

However, we make the GAMIL2 share the same code version in these three experiment models. All algorithms for computing the input surface fields in the GAMIL2 have been registered to the C-Coupler1 as the private external algorithms. In different experiment models, the same runtime procedures of the GAMIL2 have different lists of runtime algorithms. As a result, all these experiment models keep the same (bit-identical) simulation result with the original model versions without the C-Coupler1.

The MASNUM-POM and MASNUM-sole share the same wave model MASNUM, while the MASNUM-POM and POM-sole share the same ocean model POM. Similarly, we respectively make the MASNUM and POM share the same code in these experiment models that keep the same (bit-identical) simulation result with the original model versions without the C-Coupler1.

5.4 Parallel performance

To evaluate the parallel performance of the C-Coupler1, we use a high-performance computer named Tansuo100 in Tsinghua University in China. It consists of more than 700 computing nodes, each of which contains two Intel Xeon 5670 6-core CPUs and 32GB main memory. All computing nodes are connected by a high-speed Infiniband network with peak communication bandwidth 5GB/sec. We use the Intel C/C++/Fortran compiler of version 11.1 and Intel MPI library of version 4.0 for compiling the experiment models, with optimization level O2 O3.

This evaluation focuses on the internal algorithms implemented in the runtime software system, including the data transfer algorithm and data remapping algorithm, without the consideration of the serial data I/O algorithm. Although the Tansuo100 computer has more than 8000 CPU cores, we can only use less than 1000 CPU cores because a lot of other users are using this computer. As each computing node has 12 CPU cores, we set the number of processes (each process runs on a CPU core) to an integral multiple of 12 when the number is no less than 12. A data transfer occurs between two components. When evaluating its parallel performance, the maximum number of processes for each of the corresponding two components is 384 and the process number of the two components keeps the same in each test. The two components do not share the same computing node, so as that the data transfer must go through the infiniband network. A data interpolation is processed intra a component. When evaluating its parallel performance, the maximum number of processes for the corresponding component is 768.

5.4.1 Parallel performance of the data transfer algorithm

We first use the MASNUM-POM for this evaluation, where a 3-D field, the wave-induced mixing coefficient, is directly transferred from the MASNUM to the POM. Both the horizontal grids of the MASNUM and POM have about 400,000 (721×625) grid cells. For the vertical grid, the MASNUM has 18 vertical levels while the POM has 30 vertical levels. Fig. 9 shows the performance of a data transfer, when gradually increasing the process number of the

MASNUM and POM from 1 to 384. When increasing the process number from 1 to 12 intra a computing node, the performance is slightly improved. However, from process number 12 to 384, almost linear speedup is achieved. This is because at 384 processes, the decomposition is still coarse (more than 1000 horizontal grid cells per process, with 18 vertical levels).

Next we use the data transfer from the atmospherical model GAMIL2 to the coupler component in the FGOALS-gc for further evaluation, where 19 fields on the GAMIL2 horizontal grid (the grid size is 128×60=7680) are transferred. Fig. 10 shows the performance of a data transfer. The performance is also slightly improved when increasing the process number from 1 to 12. From process number 12 to 192, the performance is improved slowly. From process number 192 to 384, the performance gets much worse. This relatively low speedup is because, when the process number gets bigger, the package size for MPI communication gets smaller so as that the communication bandwidth achieved in each MPI communication gets smaller.

5.4.2 Parallel performance of the data remapping algorithm

Similarly, we first use the MASNUM-POM for this evaluation, where the 3-D field, the wave-induced mixing coefficient, is interpolated from the 3-D grid of MASNUM to the 3-D grid of POM on the processes for POM. Fig. 11 shows the performance of a data interpolation when gradually increasing the process number of the POM from 1 to 768, including the total time, the time for rearranging the field and the time for interpolation calculation. For the performance of the total and interpolation calculation, almost linear speedup is achieved. However, the data rearrangement achieves much poorer performance. When increasing the process number from 12 to 768, only 16.x-fold speedup is achieved. This is because the package size for MPI communication gets smaller when increasing the process number.

For further evaluation, we use the horizontal data interpolation from the atmospherical grid (GAMIL2 grid) to ocean grid (LICOM2 grid with 360×196=70560 cells) on the coupler component in the FGOALS-gc, where 19 fields are interpolated. As shown in Fig. 12, the data rearrangement achieves much poorer performance speedup than the interpolation

calculation. As a result, the performance speedup of the total interpolation is poor. From process number 1 to 96, only 16-fold speedup is achieved. From processor number 96 to 768, the total performance gets poorer. This is because the package size for MPI communication is very small when rearranging the fields.

5.5 Integration of a standalone component model version

A common model software platform can prospectively facilitate model development and scientific research through unifying the manner for operating various model simulations. As the C-Coupler platform is targeted to be a common model software platform, it should be able to flexibly integrate various models for simulations. Besides the coupled model configurations that use the C-Coupler for coupling, the standalone component model versions without coupling and the coupled model versions that use other couplers but not the C-Coupler1 for coupling can also be integrated onto the C-Coupler platform. For the corresponding evaluation, we integrated a standalone component model version of the MOM4p1 onto the C-Coupler platform.

Sect. 4.4 shows the major 5 steps for coupling a component model with the C-Coupler1. It only takes steps 3 and 5 to integrate the standalone MOM4p1. For step 3, less than 10 lines of source code are added to the main driver of the MOM4p1. For step 5, we wrote 5 simple configuration files according to Sect. 4.2.1. Finally, the standalone MOM4p1 can be operated on the C-Coupler platform. When further coupling the MOM4p1 and other component models together on the C-Coupler platform, the other steps, i.e., step 1, 2 and 4 requires to be conducted.

6 Discussion of future developments

Now, the C-Coupler is ready for model development with its first version C-Coupler1. As the C-Coupler is a community coupler, we welcome more and more scientists and engineers to use it and contribute to it in various aspects, such as providing component models, coupled models, flux algorithms, model simulations, bug reports, etc. We wish more and more model

groups will use the C-Coupler for model development. Any new requirement about the C-Coupler from scientists, engineers and model groups is possible to be considered into the future plan for the C-Coupler development. In China, there are more and more users of the C-Coupler, and more and more component models and coupled models integrated on the same C-Coupler model platform.

As the first version, the C-Coupler1 does not achieve all targets of the C-Coupler. For the future versions of the C-Coupler, we will consider about at least the following several aspects:

1. Coupling generator. The runtime software system takes the runtime configuration files of a model simulation as input. In the C-Coupler1, the runtime configuration files have to be written manually by users. To facilitate the works for building a model simulation, we propose to design and develop a coupling generator in the future, which can automatically generate the runtime configuration files according to the configuration files of the component models, experiment model and external algorithms. Moreover, the coupling generator can automatically optimize the parallel performance of the whole model simulation. For example, for the direct coupling between two component models with different horizontal resolutions, the coupling generator can make the corresponding data interpolation run in the runtime procedures of the component model with higher resolution, to minimize the data size of the fields transferred between these two component models. We plan to develop the coupling generator in the C-Coupler2, the second version of the C-Coupler.

2. Single executable. The C-Coupler1 uses multiple executables for a coupled model. A typical problem of this approach is that the processor time will be wasted when the components do not run concurrently. The CPL7 coupler has demonstrated that, the approach of single executable with a top-level driver, which manages the processor layout and time sequencing between the components, can solve this problem so as to improve the overall parallel performance of the whole coupled model. For the future versions of the C-Coupler, we will think about how to achieve a similar top-level driver

in a simple way, under the general design of the C-Coupler. For example, the top-level driver should be as simple as the main driver of the coupler component in the FGOALS-gc, as shown in Fig. 6.

3. Parallel performance optimization. As shown in Sect. 5.45, the parallel performance of the data transfer algorithm is not as good asand the data remapping algorithm is not good enough even poor when the grid size is small and the parallel decomposition is fine. For the future work, we will try to further improve these data transfer algorithm. Moreover, we will think about how to make the C-Coupler help improve the parallel performance of the whole experiment model, especially when the resolution of models increases in the future. For example, the C-Coupler will provide a parallel I/O library and the coupling generator will automatically improve the overall performance when generating runtime configuration files.

4. More functions. As demonstrated in Sect. 5, the C-Coupler1 is able to unify various standalone component models and coupled models onto the same model platform. In order to unify more kinds of models onto the same model platform, the future versions of the C-Coupler will provide more functions to support one-way even two-way model nesting, interactive coupled ensemble (Kirtman and Shukla, 2002), etc. In addition, we will consider how to make the C-Coupler able to integrate various assimilation systems and diagnostic systems onto the same model platform.

5. More remapping algorithms. Recently, we merged the development of the CoR and C-Coupler together. The available 2-D remapping algorithms in the CoR are of low order currently. In future, we will develop more 2-D remapping algorithms with higher order, to provide more selections for model coupling and scientific researches. Moreover, we will consider developing some pure 3-D remapping algorithms in the CoR, such as the 3-D n-neighbor distance-weighted average and trilinear remapping algorithms used in the OASIS coupler, and compare them with the “2-D + 1-D” approach.

7 Conclusion

The C-Coupler1 is a parallel 3-D coupler, which achieves bit-identical simulation result with different numbers of processes. Guided by the general design of the C-Coupler, the C-Coupler1 enables the same code of runtime software system and component models to be shared by multiple experiment models, and enables multiple experiment models to work on the same model platform. It can be used to construct a coupler component with a few lines of code for constructing a complex coupled model like CSM, and can also be used as direct coupling for better parallel performance of the whole coupled model. The C-Coupler1 is ready for developing CSMs and ESMs. In China, the C-Coupler begins to serve more and more model development, e.g., the development of an CSM in Tsinghua University targeting the CMIP6, the development of the atmospheric model GAMIL in the State Key Laboratory of Numerical Modelling for Atmospheric Sciences and Geophysical Fluid Dynamics (LASG), Institute of Atmospheric Physics, and the development of the coupled model MASNUM-POM in the first Institute of Oceanography, State Oceanic Administration). We are starting to develop a uniform model platform which will integrate various component models, experiment models and model simulations together. We believe that the C-Coupler will help advance earth system modelling.

8 Code availability

The source cde of CoR can be downloaded through "*svn–username=guest–password=guest co http:// thucpl1.3322.org/ svn/ coupler/ CoR1.0*". The source code of the C-Coupler1 can be downloaded with the FGOALS-gc through "*svn–username=guest–password=guest co http:// thucpl1.3322.org/ svn/ coupler/ CCPL_CPL6_consistency_checking*.". If you encounter any problem about the code, please feel free to contact us (liuli-cess@tsinghua.edu.cn).

Acknowledgements. This work is supported in part by the Natural Science Foundation of China (no. 41275098), the National Grand Fundamental Research 973 Program of China (no. 2013CB956603) and the Tsinghua University Initiative Scientific Research Program (no. 20131089356).

References

Armstrong, C. W., Ford, R. W., and Riley, G. D.: Coupling integrated Earth System Model components with BFG2, Concurr. Comp.-Pract. E, 21, 767–791, doi:10.1002/cpe.1348, 2009.

Balaji, V., Anderson, J., Held, I., Winton, M., Durachta, J., Malyshev, S., and Stouffer, R. J.: The Exchange Grid: a mechanism for data exchange between Earth System components on independent grids, in: Proceedings of the 2005 International Conference on Parallel Computational Fluid Dynamics, College Park, MD, USA, Elsevier, 2006.

Bao, Q., Lin, P., Zhou, T., Liu, Y., Yu, Y., Wu, G., He, B., He, J., Li, L., Li, J., Li, Y., Liu, H., Qiao, F., Song, Z., Wang, B., Wang, J., Wang, P., Wang, X., Wang, Z., Wu, B., Wu, T., Xu, Y., Yu, H., Zhao, W., Zheng, W., and Zhou, L.: The Flexible Global Ocean–Atmosphere–Land System Model, spectral version 2: FGOALS-s2, Adv. Atmos. Sci., 30, 561–576, 2013.

Collins, W. D., Bitz, C. M., Blackmon, M. L., Bonan, G. B., Bretherton, C. S., Carton, J. A., Chang, P., Doney, S. C., Hack, J. J., Henderson, T. B., Kiehl, J. T., Large, W. G., McKenna, D. S., Santer, B. D., and Smith, R. D.: The Community Climate System Model Version 3 (CCSM3), J. Climate, 19, 2122–2143, 2006.

Craig, A. P., Jacob, R. L., Kauffman, B., Bettge, T., Larson, J. W., Ong, E. T., Ding, C. H. Q., and He, Y.: CPL6: the new extensible, high performance parallel coupler for the Community Climate System Model. Int. J. High Perform. C., 19, 309–327, 2005.

Craig, A. P., Vertenstein, M., and Jacob, R.: A new flexible coupler for Earth System Modelling developed for CCSM4 and CESM1, Int. J. High Perform. C, 26, 31–42, doi:10.1177/1094342011428141, 2012.

Donner, L. J., Wyman, B. L., Hemler, R. S., Horowitz, L. W., Ming, Y., Zhao, M., Golaz, J. C., Ginoux, P., Lin, S. J., Schwarzkopf, M. D., Austin, J., Alaka, G., Cooke, W. F., Delworth, T. L., Freidenreich, S. M., Gordon, C. T., Griffies, S. M., Held, I. M., Hurlin, W. J., Klein, S. A., Knutson, T. R., Langenhorst, A. R., Lee, H.-C., Lin, Y., Magi, B. I., Malyshev, S. L., Milly, P. C. D., Naik, V., Nath, M. J., Pincus, R., Ploshay, J. J., Ramaswamy, V., Seman, C. J., Shevliakova, E., Sirutis, J. J., Stern, W. F., Stouffer, R. J., Wilson, R. J., Winton, M., Wittenberg, A. T., and Zeng, F.:

The dynamical core, physical parameterizations, and basic simulation characteristics of the atmospheric component AM3 of the GFDL Global Coupled Model CM3, J. Climate, 24, 3484–3519, 2011.

Dufresne, J.-L., Foujols, M.-A., Denvil, S., Caubel, A., Marti, O., Aumont, O., Balkanski, Y., Bekki, S., Bellenger, H., Benshila, R., Bony, S., Bopp, L., Braconnot, P., Brockmann, P., Cadule, P., Cheruy, F., Codron, F., Cozic, A., Cugnet, D., de Noblet, N., Duvel, J.-P., Eth'e, C., Fairhead, L., Fichefet, T., Flavoni, S., Friedlingstein, P., Grandpeix, J.-Y., Guez, L., Guilyardi, E., Hauglustaine, D., Hourdin, F., Idelkadi, A., Ghattas, J., Joussaume, S., Kageyama, M., Krinner, G., Labetoulle, S., Lahellec, A., Lefebvre, M- P., Lefevre, F., Levy, C., Li, Z. X., Lloyd, J., Lott, F., Madec, G., Mancip, M., Marchand, M., Masson, S., Meurdesoif, Y., Mignot, J., Musat, I., Parouty, S., Polcher, J., Rio, C., Schulz, M., Swingedouw, D., Szopa, S., Talandier, C., Terray, P., and Viovy, N.: Climate change projections using the IPSL-CM5 Earth System Model: from CMIP3 to CMIP5, Clim. Dynam., 40, 2123–2165, 2013.

Dunne, J. P., John, J. G., Adcroft, A. J., Griffies, S. M., Hallberg, R. W., Shevliakova, E. N., Stouffer, R. J., Cooke, W., Dunne, K. A., Harrison, M. J., Krasting, J. P., Levy, H., Malyshev, S. L., Milly, P. C. D., Phillipps, P. J., Sentman, L. T., Samuels, B. L., Spelman, M. J., Winton, M., Wittenberg, A. T., and Zadeh, N.: GFDL's ESM2 global coupled climate-carbon Earth System Models, Part I: Physical formulation and baseline simulation characteristics, J. Climate, 25, 6646–6665, doi:10.1175/JCLI-D-11-00560.1, 2012.

Ford, R. W., Riley, G. D., Bane, M. K., Armstrong, C. W., and Freeman, T. L.: GCF: a general coupling framework, Concurr. Comp.-Pract. E, 18, 163–181, 2006.

Gent, P. R., Danabasoglu, G., Donner, L. J., Holland, M. M., Hunke, E. C., Jayne, S. R., Lawrence, D. M., Neale, R. B., Rasch, P. J., Vertenstein, M., Worley, P. H., Yang, Z.-L., and Zhang, M.: The Community Climate System Model version 4, J. Climate, 24, 4973–4991, 2011.

Giorgetta, M. A., Jungclaus, J. H., Reick, C. H., Legutke, S., Bader, J., Böttinger, M., Brovkin, V., Crueger, T., Esch, M., Fieg, K., Glushak, K., Gayler, V., Haak, H., Hollweg, H.-D., Ilyina, T., Kinne, S., Kornblueh, L., Matei, D., Mauritsen, T., Mikolajewicz, U., Mikolajewicz, U., Mueller, W., Notz, D., Pithan, F., Raddatz, T., Rast, S., Redler, R., Roeckner, E., Schmidt, H., Schnur, R., Segschneider, J., Six, K. D., Stockhause, M., Timmreck, C., Wegner, J., Widmann, H., Wieners, K.-H., Claussen, M., Marotzke, J., and Stevens, B.: Climate and carbon cycle changes from 1850 to 2100 in MPI-ESM simulations for the Coupled Model Intercomparison Project phase 5, J. Adv. Model. Earth Syst., 5, 572–597, 2013.

Griffies, S. M., Schmidt, M., and Herzfeld, M.: Elements of MOM4p1, GFDL Ocean Group Technical Report 6, Geophysical Fluid Dynamics Laboratory, Princeton, 2010.

Hazeleger, W., Wang, X., Severijns, C., Stefanescu, S., Bintanja, R., Sterl, A., Wyser, K., Semmler, T., Yang, S., van den Hurk, B., van Noije, T., van der Linden, E., and van der Wiel, K.: EC-Earth V2.2: description and validation of a new seamless earth system prediction model, Clim. Dynam., 39, 2611–2629, doi:10.1007/s00382-011-1228-5, 2011.

Hill, C., DeLuca, C., Balaji, V., Suarez, M., and da Silva, A.: Architecture of the Earth System Modelling Framework, Comput. Sci. Eng., 6, 18–28, 2004.

Hurrell, J. W., Holland, M. M., Gent, P. R., Ghan, S., Kay, J. E., Kushner, P. J., Lamarque, J.-F., Large, W. G., Lawrence, D., Lindsay, K., Lipscomb, W. H., Long, M. C., Mahowald, N., Marsh, D. R., Neale, R. B., Rasch, P., Vavrus, S., Vertenstein, M., Bader, D., Collins, W. D., Hack, J. J., Kiehl, J., and Marshall, S.: The Community Earth System Model: a framework for collaborative research, B. Am. Meteorol. Soc., 94, 1339–1360, 2013.

Jacob, R., Larson, J., and Ong, E.: $M \times N$ Communication and Parallel Interpolation in Community Climate System Model version 3 using the Model Coupling Toolkit, Int. J. High Perform. C, 19, 293–307, 2005.

Ji, D., Wang, L., Feng, J., Wu, Q., Cheng, H., Zhang, Q., Yang, J., Dong, W., Dai, Y., Gong, D., Zhang, R.-H., Wang, X., Liu, J., Moore, J. C., Chen, D., and Zhou, M.: Description and basic evaluation of BNU-ESM version 1, Geosci. Model Dev. Discuss., 7, 1601–1647, doi:10.5194/gmdd-7-1601-2014, 2014.

Jones, P.: First and second-order conservative remapping schemes for grids in spherical coordinates, Mon. Weather Rev., 127, 2204–2210, 1999.

Jungclaus, J. H., Fischer, N., Haak, H., Lohmann, K., Marotzke, J., Matei, D., Mikolajewicz, U., Notz, D., and von Storch, J. S.: Characteristics of the ocean simulations in MPIOM, the ocean component of the MPI-Earth System Model, J. Adv. Model. Earth Syst., 5, 422–446, 2013.

Kirtman, B. P. and Shukla, J.: Interactive coupled ensemble: a new coupling strategy for CGCMs, Geophys. Res. Lett., 29, 5-1–5-4, 2002.

Larson, J., Jacob, R., and Ong, E.: The Model Coupling Toolkit: a new Fortran90 Toolkit for building multiphysics parallel coupled models, Int. J. High Perform, C, 19, 277–292, 2005.

Li, L. J., Lin, P. F., Yu, Y. Q., Wang, B., Zhou, T. J., Liu, L., Liu, J. P., Bao, Q., Xu, S. M., Huang, W. Y., Xia, K., Pu, Y., Dong, L., Shen, S., Liu, Y. M., Hu, N., Liu, M. M., Sun, W. Q., Shi, X. J., Zheng, W. P., Wu, B., Song, M.-R., Liu, H. L., Zhang, X. H., Wu, G. X., Xue, W., Huang, X. M., Yang, G. W.,

Song, Z. Y., and Qiao, F. L.: The Flexible Global Ocean–Atmosphere–Land System Model: grid-point version 2: FGOALS-g2, Adv. Atmos. Sci., 30, 543–560, 2013a.

Li, L. J., Wang, B., Dong, L., Liu, L., Shen, S., Hu, N., Sun, W., Wang, Y., Huang, W., Shi, X., Pu, Y., G. and Yang.: Evaluation of Grid-point Atmospheric Model of IAP LASG version 2 (GAMIL2), Adv. Atmos. Sci., 30, 855–867, doi:10.1007/s00376-013-2157-5, 2013b.

Liu, H. L., Lin, P. F., Yu, Y. Q., and Zhang, X. H.: The baseline evaluation of LASG/IAP Climate system Ocean Model (LICOM) version 2.0, Acta Meteorol. Sin., 26, 318–329, 2012.

Liu, J.: Sensitivity of sea ice and ocean simulations to sea ice salinity in a coupled global climate model, Sci. China Ser. D, 53, 911–916, 2010.

Liu, L., Yang, G., and Wang, B.: CoR: a multi-dimensional common remapping software for Earth System Models, in: The Second Workshop on Coupling Technologies for Earth System Models (CW2013), available at: https://wiki.cc.gatech.edu/CW2013/index.php/Program (last access: 8 May 2014), 2013a.

Liu, L., Yang, G., and Wang, B.: Common Multi-dimensional Remapping Software CoR (**Co**mmon **R**emap) V1.0 User Reference Manual, available at: http://www.cess.tsinghua.edu.cn/publish/ess/7687/20120515135108504806682/Common%20%20Multi-dimensional%20Remapping%20Software%20CoR%20(Common%20Remap)%20V1.0.pdf (last access: 8 May 2014), 2013b.

Liu, L., Li, R., Zhang, C., Yang, G., and Wang, B.: Community coupler C-Coupler1 User's Guide, available at: http://www.cess.tsinghua.edu.cn/publish/ess/7687/20120515135108504806682/Community%20Coupler%20C-Coupler1%20User%20Guide.pdf (last access: 8 May 2014), 2014a.

Liu, L., Li, R., Zhang, C., Yang, G., and Wang, B.: Enhancing reproducibility of numerical simulation result on the C-Coupler platform, Geosci. Model Dev. Discuss., 7, 4429-4461, doi:10.5194/gmdd-7-4429-2014, , 2014b.

Oleson, K. W., Dai, Y., Bonan, G. B.,Bosilovichm, M., Dickinson, R., Dirmeyer, P., Hoffman, F., Houser, P., Levis, S., Niu, G.-Y., Thornton, P., , Vertenstein, Yang, Z., and Zeng, X.: Technical Description of the Community Land Model (CLM), NTIS #PB2004-105836, 2004.

Qiao, F., Yuan, Y., Yang, Y., Zheng, Q., Xia, C., and Ma, J.: Wave-induced mixing in the upper ocean: distribution and application in a global ocean circulation model, Geophys. Res. Lett., 31, L11303, doi:10.1029/2004GL019824, 2004.

Redler, R., Valcke, S., and Ritzdorf, H.: OASIS4 – a coupling software for next generation Earth System Modelling, Geosci. Model Dev., 3, 87–104, doi:10.5194/gmd-3-87-2010, 2010.

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

Valcke, S.: OASIS3-MCT, a coupler for climate modelling, in: The Second Workshop on Coupling Technologies for Earth System Models (CW2013), available at: https://wiki.cc.gatech.edu/CW2013/index.php/Program (last access: 8 May 2014), 2013b.

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

Valcke, S., Craig, T., and Coquart, L.: OASIS3-MCT User Guide, OASIS3-MCT 1.0, CERFACS Technical Report, TR/CMGC/12/49, Toulouse, France, available at: http://pantar.cerfacs.fr/globc/publication/technicalreport/2012/oasis3mct_UserGuide.pdf (last access: 8 May 2014), 46 pp., 2012b.

Vichi, M., Manzini, E., Fogli, P. G., Alessandri, A., Patara, L., Scoccimarro, E., Masina, S., and Navarra, A.: Global and regional ocean carbon uptake and climate change: sensitivity to a substantial mitigation scenario, Clim. Dynam., 37, 1929–1947, doi:10.1007/s00382-011-1079-0, 2011.

Voldoire, A., Sanchez-Gomez, E., Salas y M'elia, D., Decharme, B., Cassou, C., S'en'esi, S., Valcke, S., Beau, I., Alias, A., Chevallier, M., D'equ'e, M., Deshayes, J., Douville, H., Fernandez, E., Madec, G., Maisonnave, E., Moine, M.-P., Planton, S., Saint-Martin, D., Szopa, S., Tyteca, S., Alkama, R., Belamari, S., Braun, A., Coquart, L., and Chauvin, F.: The CNRM-CM5.1 global climate model: description and basic evaluation, Clim. Dynam., 40 , 2091–2121, 2013.

Wang, G., Qiao, F., and Xia, C.: Parallelization of a coupled wave-circulation model and its application, Ocean Dynam., 60, 331–339, 2010.

Yang, Y., Qiao, F., Zhao, W., Y. Teng and Yuan, Y.: MASNUM ocean wave numerical model in spherical coordinates and its application, Acta Oceanol. Sin., 27, 1–7, 2005.

Table 1. The APIs provided by the C-Coupler1.

Classification	API	Brief description
Main driver	c_coupler_initialize	This API initializes the runtime software system. A component model can obtain its MPI communicator with this interface.
	c_coupler_finalize	This API finalizes the Runtime software system.
	c_coupler_execute_procedure	This API invokes the runtime software system to run the corresponding runtime procedure which consists of a list of runtime algorithms specified by the corresponding runtime configuration files. The corresponding runtime procedure could be empty without any runtime algorithms. A component model can have multiple different runtime procedures.
Registration	c_coupler_register_model_data	This API registers a field of component model to enable the runtime software system to access the memory space of this field.
	c_coupler_withdraw_model_data	This API withdraws a field of component model from the runtime software system which has been registered before.
	c_coupler_register_decomposition	This API registers a parallel decomposition to the runtime software system. A component model can register multiple different parallel decompositions, even on the same horizontal grid.
	c_coupler_register_model_algorithm	This API registers an algorithm (also known as a subroutine) of a component model as an external algorithm of the runtime software system.
Restart function	c_coupler_do_restart_read	This API reads in the data value of fields in a restart run of model simulation.
	c_coupler_do_restart_write	This API writes out the data value of fields for restart run.

Table 1. Continued.

Classification	API	Brief description
Time information	c_coupler_get_current_calendar_time	This API gets the calendar time of the current step.
	c_coupler_get_nstep	This API gets the number of the current step from the start of the model simulation.
	c_coupler_get_num_total_step	This API gets the number of total steps of the model simulation.
	c_coupler_get_step_size	This API gets the number of seconds of the time step.
	c_coupler_is_first_restart_step	This API checks whether the current step is the first step of a restart run.
	c_coupler_is_first_step	This API checks whether the current step is the first step of an initial run, which also means whether the number of the current step is 0.
	c_coupler_advance_timer	This API advances the time of simulation.
	c_coupler_check_coupled_run_finished	This API checks whether the model simulation ends.
	c_coupler_check_coupled_run_restart_time	This API checks whether the current step is time for writing fields into I/O data files for restart run.
	c_coupler_get_current_num_days_in_year	This API gets the number of days elapsed since the first day of the current year.
	c_coupler_get_current_year	This API gets the year of the current step.
	c_coupler_get_current_date	This API gets the date of the current step.
	c_coupler_get_current_second	This API gets the second of the current step.
	c_coupler_get_start_time	This API gets the start time of the model simulation
	c_coupler_get_stop_time	This API gets the end time of the model simulation.
	c_coupler_get_previous_time	This API gets the time of the previous step.
	c_coupler_get_current_time	This API gets the time of the current step.
	c_coupler_get_num_elapsed_days_from_start	This API gets the number of days elapsed since the start time of the model simulation.
	c_coupler_is_end_current_day	This API checks whether the current step is the last step of the current day.
	c_coupler_is_end_current_month	This API checks whether the current step is the last step of the current month.

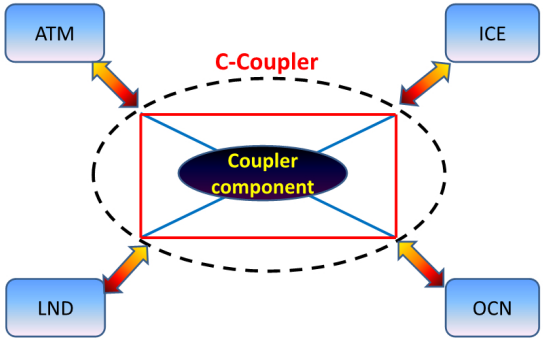

Fig. 1. The architecture of the models with the C-Coupler.

Figure 1. The architecture of the models with the C-Coupler.

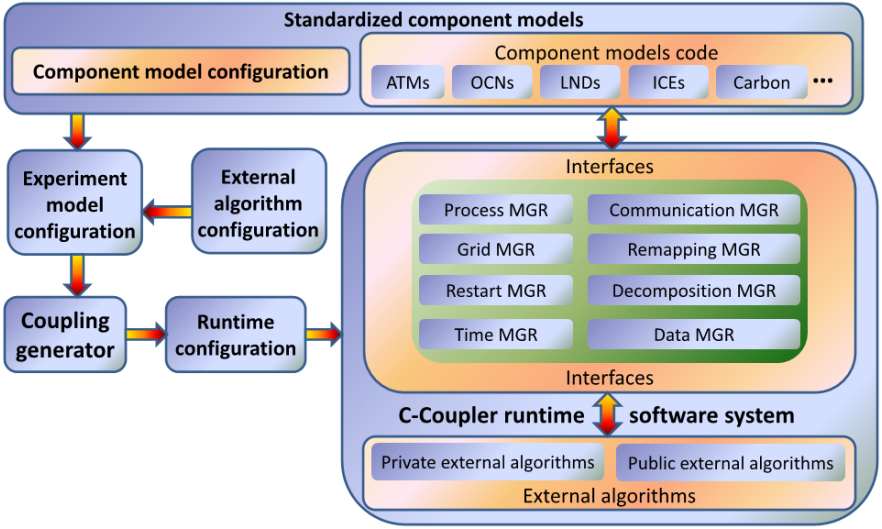

Fig. 2. The general software architecture of the C-Coupler.

Figure 2. The general software architecture of the C-Coupler.

itime	NULL	NULL
tbot	atm_2D_decomp1	atm_2D_grid1
tbot	atm_2D_decomp2	atm_2D_grid1
tbot	atm_2D_decomp3	atm_2D_grid2
tbot	atm_2D_decomp4	atm_2D_grid2
qpert	atm_2D_decomp1	atm_3D_grid1
qpert	atm_2D_decomp3	atm_3D_grid2

Fig. 3. A configuration file for an atmospherical model to register field instances to the C-Coupler

Figure 3. A configuration file for an atmospherical model to register field instances to the C-Coupler.

```
fields_mult
seconds         3600            0
fields_mult_input_fields.cfg
fields_mult_output_fields.cfg
```

(a) Main information for the external algorithm

itime	atm_model	NULL	NULL	integer
tbot	atm_model	atm_2D_decomp1	atm_2D_grid1	real8
qpert	atm_model	atm_2D_decomp1	atm_3D_grid1	real8

(b). Configuration file fields_mult_input_fields.cfg for specifying the input fields of the external algorithm.

Fig. 4. Configuration files for an external algorithm.

Figure 4. Configuration files for an external algorithm.

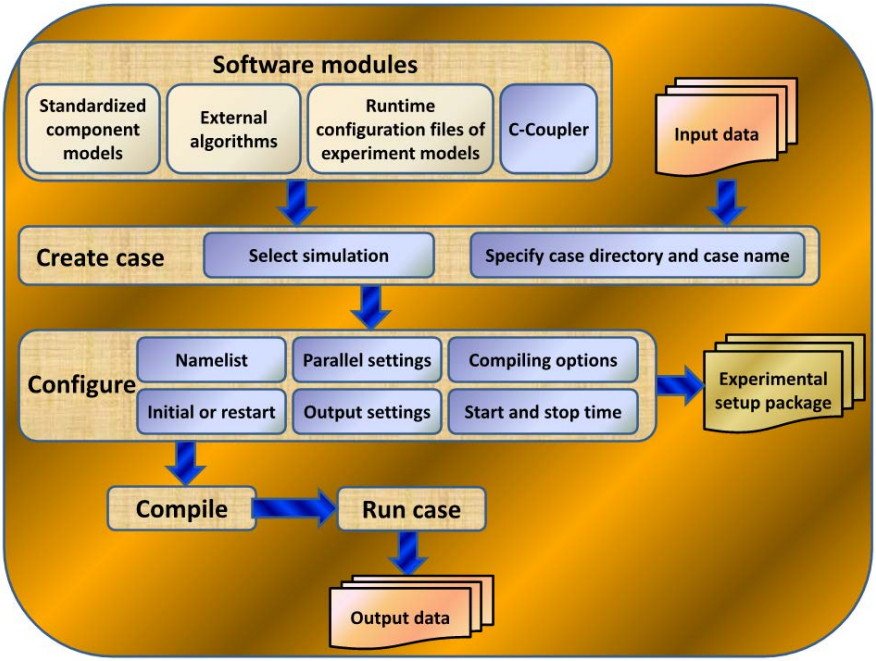

Fig. 5. The general software architecture of the C-Coupler platform.

Figure 5. The general software architecture of the C-Coupler platform.

```
program cpl

  use cpl_read_namelist_mod
  use c_coupler_interface_mod

  implicit none
  integer comm

  call c_coupler_initialize(comm)

  call parse_cpl_nml

  call c_coupler_execute_procedure("calc_frac", "initialize")
  call c_coupler_execute_procedure("sendalb_to_atm", "initialize")
  call c_coupler_execute_procedure("check_stage", "initialize")
  call c_coupler_do_restart_read
  if (c_coupler_is_first_restart_step())  call c_coupler_advance_timer

  do while (.not. c_coupler_check_coupled_run_finished())
    call c_coupler_execute_procedure("kernel_stage", "kernel")
    call c_coupler_do_restart_write()
    call c_coupler_advance_timer()
  enddo

  call c_coupler_finalize()

stop
end program cpl
```

Fig. 6. The code of the main driver of the coupler component in the FGOALS-gc. The C-Coupler APIs are marked in blue.

Figure 6. The code of the main driver of the coupler component in the FGOALS-gc. The C-Coupler APIs are marked in blue.

transfer	runtime_transfer_cpl_a2c_areac_recv.cfg
transfer	runtime_transfer_cpl_o2c_areac_recv.cfg
transfer	runtime_transfer_cpl_i2c_areac_recv.cfg
normal	frac_init_step1.cfg
remap	frac_init_remap.cfg
normal	frac_init_step2.cfg
transfer	runtime_transfer_cpl_c2lg_2D_send.cfg
transfer	runtime_transfer_cpl_r2c_areac_recv.cfg
normal	areafact_init.cfg
transfer	runtime_transfer_cpl_i2c_2D_recv.cfg
transfer	runtime_transfer_cpl_l2c_2D_recv.cfg
transfer	runtime_transfer_cpl_o2c_scalar_recv.cfg
transfer	runtime_transfer_cpl_o2c_2D_recv.cfg
transfer	runtime_transfer_cpl_a2c_2D_recv.cfg
normal	areafact_o2c.cfg
normal	areafact_i2c.cfg
normal	areafact_a2c.cfg
normal	areafact_l2c.cfg
normal	areafact_r2c.cfg
remap	runtime_remap_Xr2c.cfg

Fig. 7. Part of the runtime configuration file of the algorithm list for the coupler component in the FGOALS-gc. The first column specifies the type of each runtime algorithm. *Transfer* specifies the data transfer algorithms, *remap* specifies the data interpolation algorithms, and *normal* specifies the external algorithms. The second column specifies the configuration file of each runtime algorithm.

Figure 7. Part of the runtime configuration file of the algorithm list for the coupler component in the FGOALS-gc. The first column specifies the type of each runtime algorithm. *Transfer* specifies the data transfer algorithms, *remap* specifies the data interpolation algorithms, and *normal* specifies the external algorithms. The second column specifies the configuration file of each runtime algorithm.

calc_frac	0	6
sendalb_to_atm	7	39
check_stage	40	44
kernel_stage	45	90

Fig. 8. The runtime configuration file of the runtime procedures for the coupler component in the FGOALS-gc.

Figure 8. The runtime configuration file of the runtime procedures for the coupler component in the FGOALS-gc.

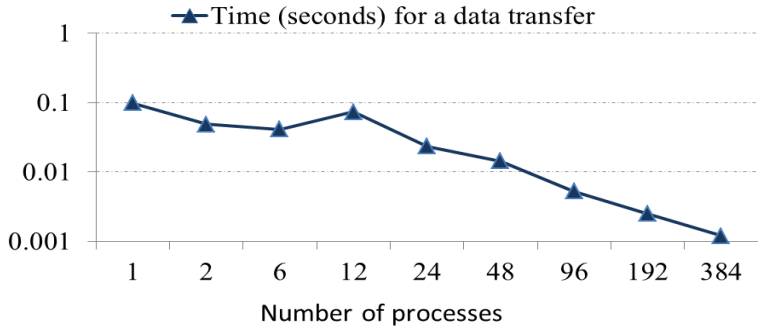

Fig. 9. The scaling performance of a data transfer in the MASNUM-POM: transferring the 3-D field wave-induced mixing coefficient from the MASNUM to the POM.

Figure 9. The scaling performance of a data transfer in the MASNUM-POM: transferring the 3-D field wave-induced mixing coefficient from the MASNUM to the POM.

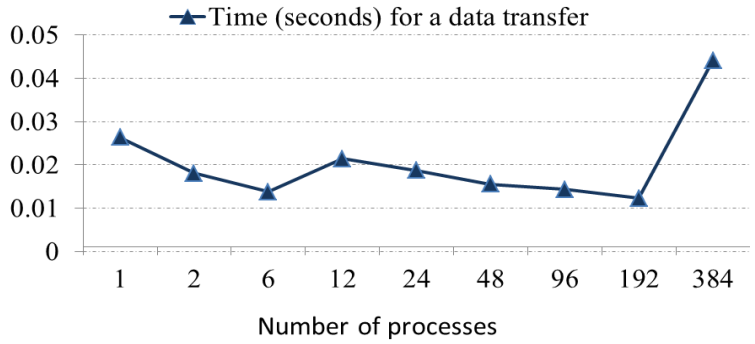

Fig. 10. The scaling performance of a data transfer in the FGOALS-gc: transferring 19 fields on the GAMIL2 horizontal grid from the GAMIL2 to the coupler component.

Figure 10. The scaling performance of a data transfer in the FGOALS-gc: transferring 19 fields on the GAMIL2 horizontal grid from the GAMIL2 to the coupler component.

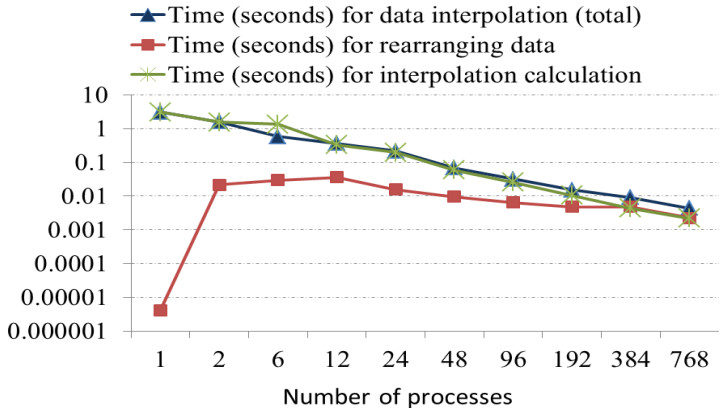

Fig. 11. The scaling performance of a 3-D interpolation in the MASNUM-POM: interpolating the 3-D field wave-induced mixing coefficient from the 3-D grid of MASNUM to the 3-D grid of POM on the POM.

Figure 11. The scaling performance of a 3-D interpolation in the MASNUM-POM: interpolating the 3-D field wave-induced mixing coefficient from the 3-D grid of MASNUM to the 3-D grid of POM on the POM.

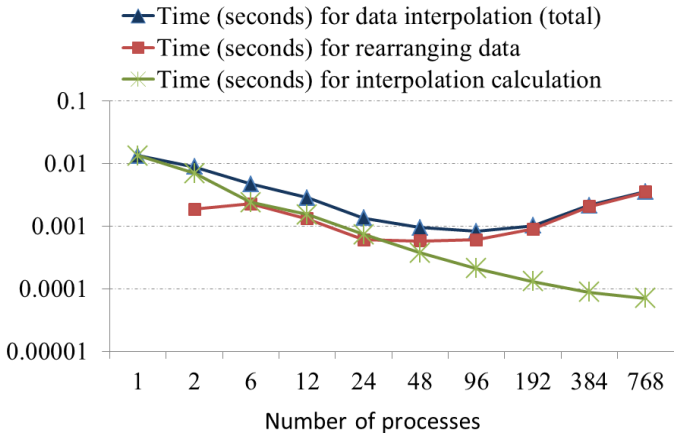

Fig. 12. The scaling performance of a 2-D interpolation in the FGOALS-gc: interpolating 19 horizontal fields from the GAMIL2 grid to the LICOM2 grid on the coupler component.

Figure 12. The scaling performance of a 2-D interpolation in the FGOALS-gc: interpolating 19 horizontal fields from the GAMIL2 grid to the LICOM2 grid on the coupler component.