Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

Geoscientific Model Development Discussions

Coupling Library Jcup3: Its philosophy and application

Takashi Arakawa[1], Takahiro Inoue[1], Hisashi Yashiro[2], and Masaki Satoh[3]

[1]Research Organization for Information Science and Technology, Minato-ku, Tokyo, Japan
[2]RIKEN Center for Computational Science, Kobe, Hyogo, Japan
[3]Atmosphere and Ocean Research Institute, The University of Tokyo, Kashiwa, Chiba, Japan

Correspondence: Takashi Arakawa (arakawa@rist.jp)

Abstract. In this paper, we describe the design of the coupling library Jcup and report its various applications including the coupling between the global atmospheric and oceanic models with different grid systems. Jcup is a software library mainly focused on weather/climate models and was developed for the purpose of coupling components of various models. Jcup has flexibility in application to an unspecified number of components of earth system models. In order to achieve high order safety and versatility, we divided the processes of a general coupling program into processes of changing and not changing the values of the data, and placed the former outside the program and under the control of the user. As a result, Jcup has two features: 1) that the correspondence relationship of grid indexes is used as input information, and 2) that the user can implement an arbitrary interpolation code. Jcup was applied to atmosphere–ocean coupling, IO component coupling, and seismic model–structure model coupling, and the validity and usefulness of the design were demonstrated.

1 Introduction

Meteorological and climate models are constructed not only by the dynamic motion of the atmosphere but also by complex interactions of physical processes, such as radiation and clouds, or of various atmospheric boundaries, such as the ocean and land surface. After choosing what to include in the model among the elements constituting the phenomenon, weather/climate phenomena can be accurately reproduced and predicted using a simulation model. These elements should be modeled with the required accuracy to calculate the resolution and the integration period corresponding to the time-space scale of the phenomenon to be expressed. However, there is an upper limit to the performance of the computer executing the calculation. Consequently, there is a trade-off between the number of model components, the accuracy of the modeling of each component, and the time-space scale of the calculation. On the other hand, the modeling precision and the spatiotemporal scale are interrelated, because the phenomena to be expressed in the model are determined to some extent by the spatial scale. Alternatively, the spatial scale of the calculation is determined according to the phenomena represented by the model. Therefore, the number of components that are modeled, and at what accuracy and to what degree of the spatiotemporal scale they are calculated, is adjusted and determined by the capacity of the computer obtained at that time and the content of the modeled phenomena. Although the performance improvement of the CPU alone slows down by increasing the degree of parallelization, the computing performance of the entire system and the memory capacity increase at a speed generally following Moore's Law. Accordingly, the spatiotemporal scale on which the model can be computed also improves, and, as a result, it becomes

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

possible and necessary to incorporate more elements into the system or to model each element more precisely. For example, according to the transition of the components of NCAR's CCSM (CESM) as stated by Washington et al.(Washington et al. (2009)), at the beginning of its development, the model comprised two components, namely the atmosphere (land surface) and the ocean. In the 1990s, sea ice and aerosol were added, and vegetation and the carbon cycle are currently being incorporated.

Spatiotemporal scales can be improved relatively easily according to the performance of the computer. However, to improve the model's accuracy, a series of procedures is required, such as constructing equations suitable for the phenomenon to be expressed, building the program, executing the program, and confirming its validity. For this reason, work by experts and research institutes in this field is indispensable for incorporating new components into the system or elaborating each component. In summary, in the modern meteorological/climate model, individual component models developed by individual groups agency are executed at each spatiotemporal resolution in parallel. Such a structure is not only a natural analogy of weather and climate phenomena, but it is an appropriate mapping of the state of research communities involved in meteorological/climate research. In the structure of such a model, the components interact as is the case of expressed subjects, and it is necessary to exchange appropriate information on an appropriate spatiotemporal scale corresponding to each component when executing the model. Appropriate grid remapping is required according to the spatial scale of each component, but it is not preferable from the viewpoint of development efficiency and maintainability to separately develop and implement such a program for each component. For this reason, dedicated software responsible for coupling between components has been developed and used.

In this paper, software that executes such tasks is called coupling software or couplers. Generally, there are two types of coupling software, one of which is a program targeting a specific model. In this case, because target components are predetermined, it can be sufficient with coupling software specialized for a specific grid system, time scale, or coupling pattern. An example of this type of coupling software is the coupler used in NCAR CESM. The other type is coupling software developed for the general use. Since the specific target is not assumed, the structures of both the interface and program are determined depending on the extent of support on the grid system, the coupling pattern, and the interpolation method. As a representative of this type of coupler, an OASIS coupler has been developed mainly by CERFACS and is widely used in European meteorological research groups(Graig and Valcke (2017)). Scup was developed to target models of the Japanese Meteorological Agency and Meteorological Research Institute, but has a general-purpose interface that can be used for other models(Yoshimura and Yukimoto (2008)). Furthermore, MCT was developed as a basic software library for constructing a coupling program and cannot perform coupling alone. However, it can be thought of as a general-purpose coupling software (Larson et al. (2005), Jacob et al. (2005)). In addition to being used for CESM couplers, MCT is also utilized as a basic library for constructing the latest version of the OASIS coupler. These coupling software supports the existing grid system and interpolation method and provides coupled computing environments, but in order to deal with grid systems and interpolation calculations that software does not suppose, some kind of software modification is required. On the other hand, Jcup is a library developed to corresponds to various lattice systems and interpolation algorithms without modifing the program in the future and to enables coupled calculation of various patterns.

In this paper, we describe the design features of Jcup and the reasons for adopting such a design. We further clarify the usefulness of the design by cases where models were coupled. First, we explain Jcup's design, focusing on two features

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

in particular. Next, the reason for Jcup having adopted such a design is described from the following three aspects: 1) the characteristics of the weather/climate model as a simulation model, 2) the relationship between the research community and the development community, and 3) the essence of coupling the models. Finally, we clarify the usefulness of the design adopted by Jcup by three cases: atmosphere–ocean coupling, coupling of IO components, and coupling of earthquake structural models.

2 Overview of Jcup

The execution pattern is implemented independently of the model components in some couplers, including the older version of the OASIS coupler (Valcke (2013)) or the JAEA coupler(Nagai et al. (2011)). However, under a massively parallel computing environment, which is mainstream today, it is not preferable in terms of computational efficiency to occupy a plurality of computation nodes by a coupling process with a relatively light computing load. Therefore, Jcup adopted an execution form that operates as a part of each model component.

Furthermore, there is generally a number of data exchange patterns: a parallel exchange in which the models use data from a preceding step of target models alternatively, and a serial exchange in which subsequent models continuously use data from preceding models. There is also a number of execution patterns for each component model: each component is executed in parallel as an independent binary, or a plurality of components is sequentially executed in one binary. As shown in Fig.1, Jcup supports a total of four patterns, namely two of data exchange and two of model execution. In addition, it is applicable to the case where these patterns are complexly combined by three or more component models as shown in Fig.2. There are three binaries, namely A, B, and C, in the figure. Component A is executed in binary A, components B, C, and D are executed in binary B, and component E is executed in binary C. In binary B, component B is executed in all MPI processes. Subsequently, component C is executed in certain processes, and component D is executed in the other process in parallel. The solid line in the figure indicates parallel exchange, and the dotted line indicates serial exchange. In reality, it seems that there is no case where such complicated execution and data exchange patterns are required, but Jcup is designed to be able to deal with such complicated cases. The interface related to data exchange and the data flow in the program are detailed in Arakawa et al. (2011). As described in the next section, Jcup is not completed as a coupling software, and to use it, procedures such as implementation of interpolation code by the user and the making of a mapping table are required. Therefore, Jcup is called a "coupling library" and not a "coupler." In this paper we use "coupling program" or "coupling software" as broad terms that include couplers and coupling libraries.

3 Design philosophy

3.1 Characteristics of a climate model as a target of coupled simulation

The objective of coupling software is to provide users with a software environment that enables them to couple multidisciplinary simulation models and to perform coupled simulation. Jcup was developed for being used for various coupled calculations without limiting the field.

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

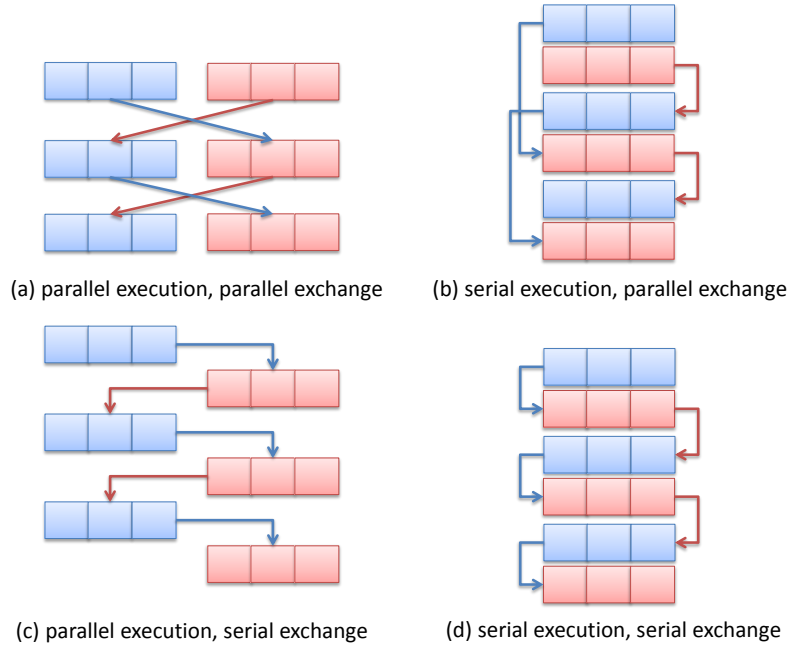

Figure 1. Coupling pattern:(a) and (b) show parallel data exchange, and (c) and (d) show sequential data exchange.(b) and (d) represent single-program coupling, and (a) and (c) represent multiprogram coupling.

For this purpose, the external conditions confronted by the coupling program can be summarized as follows:

– Each model has a grid structure suitable for the physical state expressed by the model. In addition, the optimum grid structure may change depending on external factors, such as computer architecture.

  For example, in the conventional global atmospheric model, latitude and longitude grids and spectral methods were used. However, to avoid an increase in calculation cost of Legendre transformation, models using grid structures that differ from those in conventional models, such as icosahedral grids, Yinyang grids(Baba et al. (2010)), and Cubic grid (Adcroft et al. (2004)), have recently been developed. Regarding the ocean model, grid point concentration in the polar region (Arctic Ocean) is a classical problem, and Stretch and tri-polar grids are therefore widely used. Furthermore, river models, which adopt an irregular grid system that expresses the catchment along the river channel(Yamazaki et al. (2014)) might become a coupling target.

– The interpolation method between models varies depending on the physical requirements of each model and cannot be uniquely determined.

  In cases where the integration period is relatively short or when the system is not physically closed, it is unnecessary to strictly satisfy conservativity, and in such cases, simple linear interpolation might be sufficient. On the other hand,

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

Geoscientific Model Development Discussions

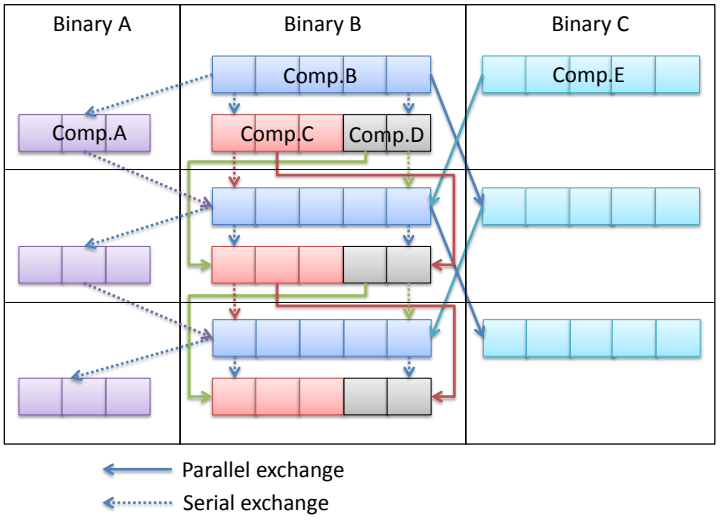

Figure 2. Example of complex coupling pattern. The solid line indicates parallel exchange, and the dotted line indicates serial exchange.

in simulations that require long integration times and have physically closed systems, such as climate simulation, the preservation of physical quantities is crucial, and interpolation algorithms that satisfy conservativity are required. Furthermore, it can be assumed that the physical quantity to be interpolated or the interpolation algorithm might change depending on the physical condition in the model, for example, the solar altitude and coverage degree of ice.

– In many cases, coupled simulation entails multiphysics and multiscale nonlinear calculations, and computational bugs are extremely difficult to find.

  In complex-system simulation, results are difficult to predict because of the system's nonlinearity and are complexly altered by slight changes in the parameters. Therefore, even if there are bugs in the program, their detection is generally difficult and requires intensive labor, except for fatal bugs where the results change radically.

– In general, climate models have a large program scale and are physically complex. It is also laborious to construct programs and verify the scientific validity. Therefore, once established, these models continue to be used for a long time while being improved.

  For example, AGCM of GFDL was developed in the 1950s. It has been improved up to the present and continues to be one of the representative general circulation models. In addition, many of the modern representative atmospheric models, such as NCAR's CESM, the ECMWF model, and the Hadley Center model, are rooted in models developed

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

in the 1960s, and, including the aforementioned GFDL model, they have long development histories, which can be expressed as a phylogenetic tree(Edwards (2011)).

3.2 Relationship between research community and development community

As mentioned in the introduction, coupling programs are roughly divided into those targeting specific models and those intended to be used generically without specifying a model. In the former case, the majority of the coupling program development community is close to or included in the model research and development community. The range of the direction of model improvement is narrow, and the development community also deals with limited partners. Consequently, the coupling program can respond quickly and appropriately to changes in model components. On the other hand, in the case of coupling software developed for general- purpose use, the relationship between the development entity and the user community varies. For example, they can belong to the same community, such as Scup, they can develop software independently without belonging to a specific research community, such as MCT, or development can be promoted by maintaining a relatively close relationship with multiple research institutions, such as OASIS. In particular, when the development community does not have a relationship with a specific research community and develops combined software targeting an unspecified number of models, the direction of model improvement is not determined. In such a case, the relationship between the research community and the development community tends to be distant. From the research community's view, this means that the required improvement of the coupling software accompanying the improvement of their own model might not be performed promptly or at all. Moreover, for research communities, the coupler code is a black box, which means that it is difficult to detect and trace a non-fatal bug.

3.3 Essential functions of coupling program

The essential functions of a coupling program in the large-scale parallel computing environment is as follows: 1) coupling between two or more component models, 2) appropriate timing, 3) performing appropriate time interpolation according to the time scale of each model, 4) performing appropriate interpolation calculations according to the lattice shape, and 5) exchanging data between appropriate nodes. Here, the condition that the time step of each model is a divisor of the data exchange interval is set. This condition seems to be reasonable in coupling components such as atmospheric and ocean models, which involve large-scale phenomena and long- term, constant interaction. In this case, an essential function of the coupling program is to execute interpolation calculation and data exchange at an appropriate time, because it is not necessary to consider time interpolation. The functions required for the coupling software are summarized into the following three functions: 1) decision of data exchange timing, 2) data exchange, and 3) interpolation calculation.

3.4 Basic design of Jcup

Jcup is a general-purpose coupler that does not cover specific models, and the relationship between coupler developer and user community is one-to-many and not dense. In this case, the performance requirements of the coupler are as follows:

- Since Jcup is a black box for users, it is desirable to minimize the possibility of mixing bugs caused by the coupler.

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

– Since Jcup is used over a wide range and long term, it must respond flexibly to various grid systems and interpolation methods that exist or will be newly developed. Jcup's design philosophy is to realize the abovementioned three functions under these requirements.

3.4.1 Time control

In judging the data exchange timing, we assumed that the data are exchanged at predetermined time intervals. Under this assumption, the condition of data exchange timing is that the elapsed time from the start of integration is a multiple of the data exchange interval. Therefore, in the coupler, it suffices to hold and manage elapsed model time from the start of integration. On the other hand, the time information handled by the weather/climate model during calculation is generally the current time and Δ_T. Therefore, there are two ways to obtain the elapsed time: 1) to find the difference between integration start time and current time and 2) to integrate Δ_T. The former method is considered to be valid when only the Gregorian calendar is used in the model. In reality, however, the calendar is not limited to the Gregorian calendar but is diverse, including the Julian calendar, the calendar without leap years, or the calendar that limits the number of days in a month to 30. Furthermore, it may be necessary to consider the changeover of the calendar to the calculation of the time difference. Therefore, the method of calculating the time difference is not preferable in terms of versatility and computational safety. Jcup consequently adopted a method of integrating Δ_T at each time step of the latter method. In this case, to avoid errors and erroneous judgment because of real number operation, input is limited to integer, and milliseconds/microseconds can be set as time unit to correspond to Δ_T of less than 1 s.

3.4.2 Data exchange

To simplify the condition and description, we assumed that the spatial interpolation calculation was performed by the receiving-side component. Furthermore, we assumed that each component was parallelized by region. In this case, data exchange means that the value of each grid point of the sending component is transmitted and received by an area that requires its value in the receiving component, that is, uses the value of that point in the interpolation calculation. Therefore, to perform appropriate data exchange, information on the grid point index of the sending-side component used for calculating each grid point value of the receiving-side component was necessary. The correspondence between the grid point index of the sending-side component and that of the receiving-side component in interpolation calculation is referred to as a mapping table here. Moreover, since each component is parallelized by region in the considered condition, it suffices to be given a point index of each grid point assigned to each region of each component to perform appropriate data exchange. The problem here is how to obtain the mapping table. In general coupling software, the position of grid points, and the grid shape are given as input information, and mapping tables and interpolation coefficients are calculated in the coupling software. This method, however, has the following problems:

1. Coupling software can only deal with grid shapes and interpolation algorithms that are already installed, and portability/extensibility is poor.

2. Depending on the lattice shape, it might be difficult to eliminate the possibility of bug contamination, for example, requiring complicated calculations or accurate judgment considering rounding error.

This can be a serious disadvantage for general-purpose coupling software, for which there is a distance between the software developer and users and which is recognized as a black box by the users. Therefore, Jcup does not calculate the mapping table internally but uses the mapping table itself as input information. This enables the elimination of problems caused by mapping table calculation from the coupling software as described above. The idea of using the mapping table as input information is not unique to Jcup, but there are other coupling software that can give a mapping table in addition to grid information, such as the YAC coupler(Hanke et al. (2016)). What is significant about Jcup is not that the mapping table can "also" be used as input information but that input information is limited to the mapping table only. As a result, increases in the size and complexity of the code of the coupling software are suppressed, and the future extensibility and security in the software are "clearly" secured. This is considered to be an essential condition for an unspecified number of research communities that cannot necessarily uniquely control the tendency of a coupling software developer.

3.4.3 Interpolation calculation

As with data exchange, in interpolation calculation, it is important to ensure future extensibility and portability and to eliminate the possibility of bug contamination. For this purpose, the interpolation calculation program should not be implemented as a black box in Jcup but should be placed in a location that can be viewed in such a way that it can be controlled by users. The difference between the mapping table and the interpolation calculation is that the former can be calculated with a program independent of the model before model execution, whereas the interpolation calculation must be performed at each data exchange step during model execution. This means that the calculation performed during model execution as part of the operation of the coupling software must be placed under the user's control. To realize this in the interpolation calculation, we ensured that the interface subroutine was provided by the coupling-software side and that the concrete interpolation calculation code was implemented in the interface subroutine by the user side. The following conditions were considered:

1. Interpolation algorithms used for one set of coupling are not necessarily limited to one type but may vary depending on data or other conditions.

2. Interpolation calculation is executed not only for each type of data, but, in addition to that, it can be computed by combining multiple types of data, such as vector calculation.

Therefore, in the interface subroutine implementing the interpolation calculation, it is indispensable to have functions for identifying individual data points and for simultaneously processing multiple kinds of data. Conversely, if these functions are provided, arbitrary interpolation calculation code can be implemented with a high degree of freedom.

3.5 Section summary

In this section, Jcup's design philosophy, which is based on three basic functions indispensable for coupling software, was described. The philosophy underlying and informing the design can be summarized as follows:

1. The relationship between the developer and the user community should be considered. In the case of Jcup, since there is a distance between the developer and the user without specifying the user, extensibility/portability and safety are required.

2. On the other hand, considering the series of coupling software operations, it is possible and important to set the classification axis on operations that may change the value of the data and that do not change their values.

3. Therefore, operations that may change the data values should be excluded from the coupling software, which is a black box for users, and should be placed in a visible location and under the control of users.

The next section explains how these basic designing points were realized.

4 Interface

In this section, we describe the interface related to the judgment of data exchange timing and the setting of the grid number and implementation of the interpolation calculation code mentioned in the previous section. Jcup specifications as a prerequisite for explanation are summarized as follows:

1. A single binary can have multiple model components.

   For example, when the program of the atmospheric model has atmospheric, land surface, and river models.

2. A single model component can have multiple grid systems.

   For example, when the model component uses a staggered grid system.

3. A set of grid systems can have multiple mapping tables.

   This specification corresponds to cases where the interpolation method varies depending on the data to be exchanged. For example, if one data point is used for linear interpolation and the other data point for nearest neighbor approximation, two mapping tables are used for a set of grid systems.

4.1 Interface for time control

There are three interface subroutines for exchanging appropriate data at appropriate times, namely jcup_def_grid, jcup_init_time, and jcup_set_time. The arguments for these three subroutines are shown in Table 1. The subroutine jcup_def_varg is for setting the attributes of the received data. The argument data_ptr is a structured type variable pointer for the model component to identify the data. The arguments comp_name, data_name, and grid_name are the components, variable names, and grid names to which the variable belongs, respectively. The arguments send_comp_name and send_data_name are the sender component and data names, respectively. The argument interval is an argument related to the control of the time exchange and represents the data exchange interval. The argument mapping_tag is a tag for specifying the mapping table to be used for interpolation calculation. The argument data_tag is a tag for identifying data in the interpolation calculation subroutine to be described later.

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

Among these arguments, intervals are used to control the timing of data exchange. This argument is an integer type and defaults to seconds, but it can be changed to milliseconds or microseconds depending on the Jcup initialization subroutine setting.

The subroutine jcup_init_time gives Jcup the initial coupling time. The argument time_array defaults to an array of size 6, representing year/month/day/hour/minute/second. However, depending on Jcup's initial setting, it is also possible to pass an array of size 7 or 8 in milliseconds or microseconds, respectively.

The subroutine jcup_set_time is used in the time integration loop and sets the current time and Δ_T of each component. Each argument is a component name, current time, and Δ_T. From these pieces of information, whether it is the time to exchange data is determined at each time step, and these data are transmitted and received.

Table 1. APIs for time control

subroutine name	argument type	argument name	description
jcup_def_varg	type(varg_type), pointer	data_ptr	data identifier
	character(len=*), intnet(IN)	comp_name	name of component
	character(len=*), intnet(IN)	data_name	name of data
	character(len=*), intent(IN)	grid_name	name of grid
	character(len=*), intnet(IN)	send_comp_name	name of send component
	character(len=*), intnet(IN)	send_data_name	name of send data
	integer, intent(IN)	interval	data exchange interval
	integer, intent(IN)	mapping_tag	tag of mapping table
	integer, intent(IN)	data_tag	tag of the data
jcup_init_time	integer, intent(IN)	time_array(:)	array of initial time
jcup_set_time	character(len=*), intent(IN)	comp_name	name of component
	integer, intent(IN)	time_array(:)	array of current time
	integer, intent(IN)	delta_t	delta t

4.2 Interface for grid index

The setting of the grid number and mapping table is realized in the following procedure:

1. Set the name of the model component of the individual binary.
2. Set the grid index of each component.
3. Set the mapping table for each grid pair.

The interface subroutines corresponding to each procedure are summarized in Table 2. These subroutines can be called multiple times.

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

The subroutine jcup_set_new_comp sets the name of each component, and the subroutine jcup_def_grid sets the grid index number. The argument grid_index is one-dimensional, and even in cases of a two- or three-dimensional grid, the grid index number is given as a one-dimensional array. The optional argument num_of_vgrid assumes the number of vertical layers. This is set, for example, where the data to be exchanged are three- dimensional, but the interpolation calculation is only in the horizontal direction, as in the case of coupling an atmospheric model with an atmospheric/chemical model having different horizontal resolutions.

The subroutine set_mapping_table sets the mapping table. Since this subroutine performs data communication between the send and receive models, it needs to be called synchronously. The arguments send_grid_index and recv_grid_index are correspondence tables of the grid numbers of the send and receive models, respectively, and they can be given on either the send or receive model. The argument mapping_tag is an identification number to identify the mapping table. By appropriately using this argument, it becomes possible to apply multiple mapping tables to a pair of grid systems. It should be noted here that the information given by set_mapping_table is only the correspondence relationship of grid numbers in the interpolation calculation, and no interpolation coefficient is given. As mentioned in the previous section, Jcup adopts the following design policy: 1) the operations of changing and not changing the value are separated, 2) the coupling software takes charge of the operation of not changing the value, and 3) the operation of changing the value is disclosed to the user. The interpolation coefficient is a value related to interpolation calculation changing the value. Therefore, it is handled separately from the correspondence relationship of the grid numbers. The handling of interpolation coefficients is described in the next section.

Table 2. APIs for grid setting

subroutine name	argument type	argument name	description
jcup_set_new_comp	character(len=*), intent(IN)	component_name	name of search model component
jcup_def_grid	integer, intent(IN)	grid_index(:)	array of grid index
	character(len=*), intnet(IN)	comp_name	name of component
	character(len=*), intent(IN)	grid_name	name of grid
	integer, optional, intent(IN)	num_of_vgrid	number of vertical layers
jcup_set_mapping_table	character(len=*), intent(IN)	my_comp_name	name of my component
	character(len=*), intent(IN)	send_comp_name	name of send component
	character(len=*), intent(IN)	send_grid_name	name of send grid
	character(len=*), intent(IN)	recv_comp_name	name of receive component
	character(len=*), intent(IN)	recv_grid_name	name of receive grid
	integer, intent(IN)	mapping_tag	identifier of mapping table
	integer, optional, intent(IN)	send_grid_index(:)	index of send grid
	integer, optional, intent(IN)	recv_grid_index(:)	index of receive grid

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

4.3 Interface for spatial interpolation

Jcup's design is that the interpolation calculation code is implemented by the user, and the coupling software provides only the interface. Subroutine interfaces for interpolation calculation are as shown in Table 3. The argument mapping_tag corresponds to the argument of the subroutine jcup_set_mapping_table. By the arguments send_comp_name, recv_comp_name, and mapping_tag, users can identify the mapping_table used for interpolation calculation. The arguments sn1 and sn2, and rn1 and rn2 indicate the size of array send_data and recv_data, respectively. Jcup treats the spatial dimension as interpolated in one dimension. Therefore, for example, assuming horizontal interpolation, sn1 and rn1 are the number of horizontal grid points of the send- and receive-components respectively. However, Jcup extracts only grid points related to interpolation calculation and exchanges, and sn1 and rn1 therefore do not necessarily match the size of the grid defined by the subroutine jcup_def_grid. The arguments sn2 and rn2 representing the size of the array of the second dimension have two meanings depending on the condition. The first is the number of vertical layers, for example, a three-dimensional variable that performs only horizontal interpolation corresponds to this condition. Here, the maximum value of the number of vertical layers is defined by the optional argument num_of_vlayer of the subroutine jcup_def_grid, but apart from that, the number of layers can be specified for each data point. The number of layers for each data point is determined by the optional argument of the API subroutine jcup_def_varg, which defines the receive data. The second is the number of data sent and received together and brought into the interpolation subroutine. To reduce the number of calls of the MPI subroutines, Jcup has the function of exchanging multiple pieces of data at once, which is realized through the argument data_tag of the API subroutine jcup_def_varg. Data having the same value of data_tag are grouped together, sent and received, and passed to the interpolation subroutine interpolate_data. The last argument data_tag is an identifier for identifying the type of data, and the value given by jcup_def_varg is passed. For future extension data_tag is to be given as an array, but each value of array is the same.

Table 3. Interface of interpolation subroutine

subroutine name	argument type	argument name	description
interpolate_data	character(len=*), intent(IN)	recv_comp_name	name of receive component
	character(len=*), intent(IN)	send_comp_name	name of send component
	integer, intent(IN)	mapping_tag	identifier of mapping table
	integer, intent(IN)	sn1, sn2	array size of send_data
	real(kind=8), intnet(IN)	send_data(sn1, sn2)	array of send data
	integer, intent(IN)	rn1, rn2	array size of receive data
	real(kind=8), intent(INOUT)	recv_data(rn1, rn2)	array of receive data
	integer, intent(IN)	num_of_data	number of data
	integer, intent(IN)	tn	number of data tag
	integer, intent(IN)	data_tag(tn)	array of data tag

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

As described above, Jcup exchanges only data used for interpolation calculation and passes to an interpolation subroutine. Furthermore, to reduce the calculation for data rearrangement, the region divided data are passed to the interpolation subroutine as one block for each region. Fig.3 illustrates these situations. In the figure, data are exchanged from component A to component B. Both of these components have a grid with a size of 6 x 6, and the positions of the grid points are the same. Therefore, the interpolation calculation is a copy of the value for each grid point. The red lines drawn on each grid represent the dividing line of the region. In the figure, component A is divided into six parts of 2 x 3, and component B is divided into three parts of 1 x 3. The grid points of component B are color-coded because this simulates the ocean–land distribution. In this case, only the value of the lattice point represented by the blue circle is meaningful and is to be transmitted and received. In the figure, transmission and reception and interpolation calculations relating to Rank 2 of component B are shown in detail. The grid point values required by Rank 2 of Component B are 33 for Rank 3 of Component A, 28, 33, and 34 for Rank 4, and 29, 30, 35, and 36 for Rank 5. Therefore, Ranks 3, 4, and 5 of Component A send these lattice point values held by itself to Rank 2 of Component B. Rank 2 of Component B receives these values and passes them to the interpolation subroutine in the received order. After receiving the data, the interpolation calculation is executed inside the interpolation subroutine of the receiving-side component. At this time, rearranging the array is the index conversion table is, ir of equation (1). The calculation result is held in Jcup's data buffer, and it is passed from the data buffer to the component when requested by the component.

Most importantly, the data array of the send component passed to the interpolation subroutine is arranged according to the divided region of the send component. As shown in the example, since the order of the array passed to the interpolation subroutine is different from that of the grid, even if the grid points of the sending component and those of the receiving component are the same, the interpolation calculation is not a simple array copy, and a grid index conversion table is necessary. For example, we assumed that the value of a certain grid point on the receiving component can be calculated from the grid point value on the sending component and the interpolation coefficients as shown in equation (1).

In this case, the code representing the equation is as follows:

```
do i = 1, N
  R(ir(i)) = R(ir(i)) + S(is(i)) * C(i)
end do
```

Here, the arrays ir and is correspond to the conversion tables of the grid indexes. This conversion table is calculated in Jcup from the two information sets given by the interface subroutines jcup_def_grid and jcup_set_mapping_table, that is, the number of the grid point in charge of each area and the correspondence of grid point indexes in the interpolation calculation. The calculated table can be obtained through Jcup's interface subroutine jcup_get_interpolation_table. Since Jcup assumes that the grid shape does not change over time, it is sufficient for the user to call this subroutine only once before time integration.

Next, we describe how interpolation coefficients are treated in Jcup. As mentioned above, Jcup's basic functions are to send and receive individual grid point values between appropriate components with appropriate timing. The grid point indexes assigned to each region of the component and the correspondence relation between the grid point indexes on the interpolation calculation are necessary to realize these basic functions, and the management of interpolation coefficients is left to the user.

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

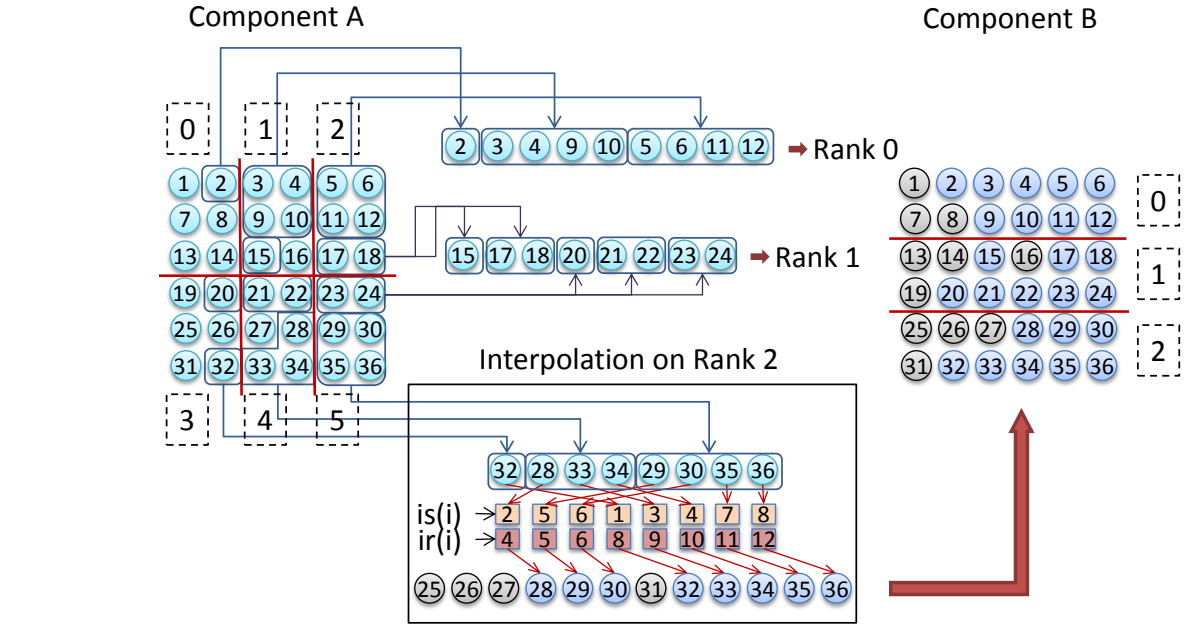

Figure 3. Data exchange diagram. Component B is assumed to be an ocean model, and gray circles represent grid points of the land area. Therefore, only the grid point value indicated by the blue circle is passed from Component A to Component B.

However, since information for properly processing interpolation coefficients are held in Jcup, subroutines for processing interpolation coefficients are provided to the user.

Among the interpolation coefficient subroutines, the most frequently used subroutine is jcup_set_local_coef.

This subroutine is used when the interpolation coefficient Cg for interpolation calculation of the whole area is defined by the following code:

```
Do i = 1, Ng
   R (irg (i)) = R (irg (i)) + S (isg (i)) * Cg(i)
End do
```

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

Geoscientific Model Development Discussions

The subroutine jcup_set_local_coef returns the interpolation coefficients for each region represented by Cl in the lower code from Cg:

```
Do i = 1, Nl
   R (ir (i)) = R (ir (i)) + S (is (i)) * Cl(i)
End do
```

The arguments of the subroutine are as shown in Table 4.

Table 4. Arguments of jcup_set_local_coef

subroutine name	argument type	argument name	description
jcup_set_local_coef	character(len=*), intent(IN)	recv_comp_name	name of receive component
	character(len=*), intent(IN)	send_comp_name	name of send component
	integer, intent(IN)	mapping_tag	mapping table indentifier
	real(kind=8), intent(IN)	Cg(:)	global coefficient
	real(kind=8), pointer	Cl(:)	local coefficient

To perform interpolation calculation, the interpolation calculation code is added to the subroutine interpolate_data after acquiring the necessary information using the subroutines jcup_get_interpolation_table and jcup_get_local_coef.

5 Application case studies

In this section we describe cases where Jcup was applied, thereby showing that the Jcup features mentioned above are effective for coupling various models.

5.1 MIROC coupling

The first case is the coupling of the atmospheric and ocean models in the climate model MIROC. MIROC is a global climate model jointly developed by the University of Tokyo, Japan Agency for Marine-Earth Science and Technology, and the Institute for Environmental Studies. It is a representative climate model in Japan, and its results have been referenced in the Intergovernmental Panel on Climate Change report(Watanabe et al. (2010)). MIROC consists of four subcomponents: atmosphere, ocean, land surface, and rivers. Among these components, the atmosphere, land surface, and rivers are executed sequentially in one binary. On the other hand, the atmospheric model "MIROC AGCM" and the ocean model "COCO" (Hasumi (2007)) are usually executed in parallel as different binaries and are coupled by MIROC's proprietary coupling program. The first test case in Jcup's development is to replace MIROC's original coupling program with Jcup. Importantly, when replacing the coupling program with Jcup, the calculation results do not change before and after the replacement, that is, the results of both

are identical at the binary level. By guaranteeing matching at the binary level, there will be no bug in Jcup internally or in the coupling code using Jcup.

In the coupling of the original MIROC, the coupling procedure consists of three steps, namely 1) collecting data to the root processor and transmitting it to the root processor of the other component, 2) executing the interpolation calculation by the root processor of the receiving-side component, and 3) distributing the result to each processor. On the other hand, in Jcup, data exchange is performed local-to-local, and interpolation calculation is individually performed in each processor on the receiving side.

Since there is no change in values during data transfer, the interpolation calculation must at least be the same, including the calculation order, to match the results at the binary level. This was made possible by Jcup's design in that users can arbitrarily implement interpolation calculation code. A part of interpolation calculation code implemented in MIROC is shown in Fig.4.

The figure shows an interpolation calculation code when the atmosphere receives information on sea ice calculated by COCO. In this calculation, coefficients corresponding to the proportion of sea ice that changes with time are calculated at grid points where both the ocean area ratio and the sea ice area ratio are larger than 0; the grid point value of the reception side is calculated according to the calculation equation $R = R + S * C$. Although not shown in the figure, another type of calculation code is also used depending on the type of data.

One of the advantages of Jcup's design is that it is possible to shift the coupling with such complicated interpolation calculation to coupling using local calculation, which has higher efficiency while maintaining compatibility at the binary level.

5.2 NICAM–COCO coupling

As a second example, we describe the coupling of the atmospheric model NICAM and the ocean model COCO. NICAM is a global atmospheric model using a non- hydrostatic equation system (Tomita and Satoh (2004), Satoh et al. (2008), Satoh et al. (2014)). The discretization method is a finite volume method, and the grid system employs an icosahedral grid. This model has been mainly utilized for research on phenomena with time scales of several days to months, such as typhoons and MJO, and in such research, simulation is carried out with the atmosphere only model by specifying sea surface temperature (SST) or with a slab ocean model (Miyakawa et al. (2014)). However, NICAM is expected to be applied not only to such short- to medium- term simulations but also to climate simulations with time scales of several decades to centuries (Kodama et al. (2015), Haarsma et al. (2016)). In such long-term simulations, although the current simulations are conducted with the atmosphere only model under specified sea surface temperature (SST) condition, it is natural to extend NICAM to be coupled with an ocean model to be used as an atmosphere-ocean coupled model to internally reproduce SST in the model. The ocean model coupled to NICAM in this study was COCO. COCO is a global ocean model using the Boussinesq approximate hydrostatic equation and a general orthogonal coordinate system. The version of COCO used for this coupling adopts the Tri-polar grid. At a certain latitude, the Tri-polar grid does not have a regular latitude and longitude grid but an irregular grid shape, such that the North Pole is moved onto the North American and Eurasian continents. In fact, the study with the coupled model with NICAM and COCO using Jcup is already conducted by Miyakawa et al. (2017) and Kubokawa et al. (2018).

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

Geoscientific Model Development Discussions

```
DO N = 1, IJ_OMAX
   DO M = IJ_OHEAD(N), IJ_OHEAD(N+1) - 1
         LA = IJRECOV_O2A(M)
         IF (BFSOCNG(LA).GT.0.0D0
  $           .and.BFOxxG(LA,2).GT.0.0D0) THEN
           LO   = IJO2C(LA,N)
           AOCN2 = SOCN(LA,N) * BFOyyG(LO,2)
  &                / (BFSOCNG(LA)* BFOxxG(LA, 2))
           BFOxxG(LA,3) = BFOxxG(LA,3)
  &           + ( BFOyyG (LO,3) * 1.D-2 ) *AOCN2
           BFOxxG(LA,4) = BFOxxG(LA,4)
  &           + ( BFOyyG (LO,4) * 1.D-2 ) *AOCN2
           BFOxxG(LA,5) = BFOxxG(LA,5)
  &           + ( BFOyyG (LO,5) + kelvin )*AOCN2
           BFOxxG(LA,6) = BFOxxG(LA,6)
  &           + ( BFOyyG (LO,6) + kelvin )*AOCN2
           ENDIF
   END DO
END DO
```

Figure 4. Example of MIROC interpolation code.

In this way, both NICAM and COCO are irregular grid-system models covering a spherical surface, and a large number of calculations is required to determine the grid point correspondence and interpolation coefficients necessary for coupling. This is because it is necessary to search for polygons, including individual grid points in the calculation of the grid correspondence, and to calculate the area of the overlapped portion of the polygons to obtain the interpolation coefficient. On the other hand, in meteorological simulations, the shape of the grid is generally constant irrespective of time, and the resolution does not change frequently. In addition, the resolution used has a certain pattern. For example, in an atmospheric model of the spectral method, frequently used resolutions are T42, T85, T128, and T256, and those in ocean models are 1degree, 0.25degree, and 0.1degree. Particularly in NICAM, there is a constraint condition that the number of grid points is specified by the power of 2, and the pattern of resolution that can normally be employed is limited to five to six patterns. For these reasons, Jcup's design, involving computation of expensive and infrequently updated mapping tables by an external program beforehand and the coupler using the mapping table as input information, is well adapted to NICAM-COCO coupling.

5.2.1 Performance evaluation

The experiment was divided into three cases according to the number of horizontal grid points of NICAM, which could be calculated by a function $10 * (2^{glevel})^2$. Correspondence between the cases and grid points is shown in Table 5

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

Table 5. Correspondence between the cases and grid points

case	glevel	grid points
Case 1	5	10240
Case 2	7	163840
Case 3	9	2621440

Experimental conditions are listed in Table 6. The conditions of COCO were fixed in all experiments. Size m52 corresponded to the number of horizontal grid points 360 (West– East) x 256 (South–North). The number of the process was 16, and ΔT was 15 min. The number of NICAM processes was represented by r-level as equation $10 * 4^{rlevel}$. $Delta$T and duration time are shown in Table. The data exchange interval between NICAM and COCO occurred every hour.

Table 6. Experimental conditions of NICAM–COCO coupling

		Case 1			Case 2			Case 3		
COCO	size	m52	m52	m52	m52	m52	m52	m52	m52	m52
	process	16	16	16	16	16	16	16	16	16
	ΔT[min]	15	15	15	15	15	15	15	15	15
NICAM	glevel	5	5	5	7	7	7	9	9	9
	rlevel	0	1	2	0	1	2	1	2	3
	process	10	40	160	10	40	160	40	160	640
	ΔT[min]	15	15	15	4	4	4	1	1	1
	duration time[day]	10	10	10	2	2	2	1	1	1

Results of the performance measurement are shown in Fig.5. The bar graph indicates the number of days that can be simulated by one day's calculation. The line graph is a scaling factor on the basis of 10 (or 40) processes.

The most notable difference between the cases was the state of the change in the scaling factor. In Case 1, the scaling factor was reduced according to the number of processors, but it was almost constant in Case 3.

Two reasons can be postulated for this difference:

1. Efficiency of NICAM itself

   The number of grid points per processor was smaller in Case 1 than in Cases 2 or 3 as shown in Table 7. Therefore, calculation time, which was scalable, became relatively short, and the non-parallelized process became longer. This could have caused the low scalability.

2. Load imbalance between NICAM and COCO

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

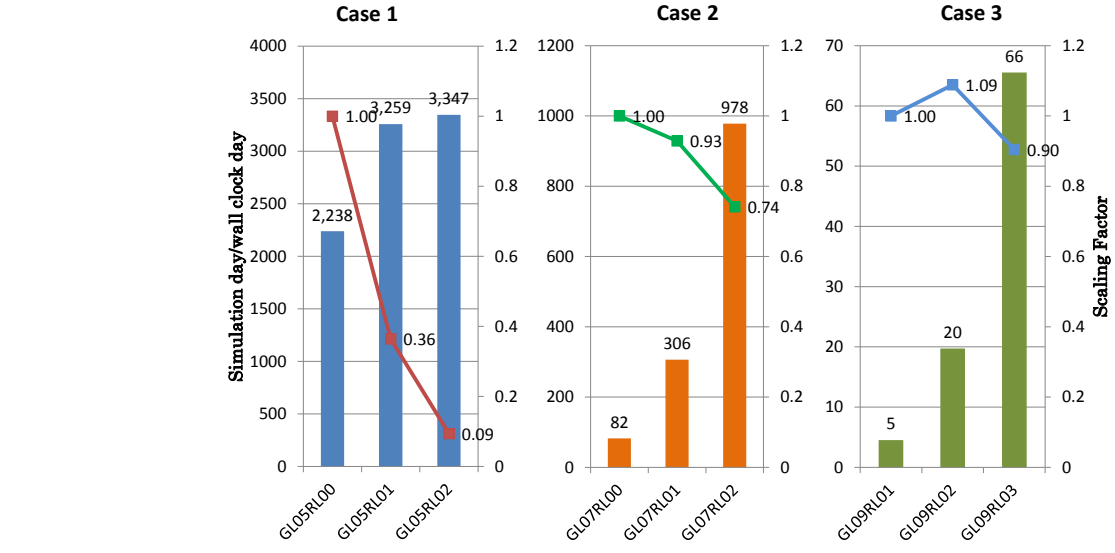

Figure 5. Results of performance measurement on NICAM–COCO coupling. Bar graph is the number of days that can be simulated in one days calculation. Line graph is a scaling factor.

In Case 1, time step calculation of NICAM was faster, and NICAM had to wait for data from COCO. Execution time therefore did not decrease with an increase in the number of processors. In contrast, in Case 3, NICAM did not need to wait for the data. Execution time was therefore determined by the number of processors assigned to NICAM.

Table 7. Number of horizontal grid points per processor

	Case 1			Case 2			Case 3		
glevel	5	5	5	7	7	7	9	9	9
rlevel	0	1	2	0	1	2	1	2	3
The number of grid points	1024	256	64	16384	4096	1024	65536	16384	4096

To confirm these assumptions, the execution time of NICAM's time integration loop is listed in Table 8. "Main ALL" is the execution time of the entire time integration loop. "Coupler Put" is the time for moving the data from NICAM to the coupler, and "Coupler Get" is the reverse. "Coupler Exchange" is the time for data exchange; load imbalance is included in this time. "Atmos" is the calculation time without coupling.

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

Geoscientific Model Development Discussions

To examine the scalability of NICAM itself, we calculated the scalability factor from the execution time of "Atmos." The result is shown in Fig.6. In Case 1, the scalability factor decreased even though there was no data exchange.

On the other hand, it should be noted that to examine load imbalance the time "Coupler Exchange" in Case 1 was significantly longer than that in the other cases. This suggests that the reception waiting occurred on the NICAM side during data exchange. The latency became larger as the execution time became shorter so as to keep the time of "Main All" constant. This constant time was determined by the execution time of COCO, and it can be concluded that the decrease of parallel efficiency was mainly caused by load imbalance.

Table 8. Execution time of NICAM (s)

	Case 1			Case 2			Case 3		
glevel	5	5	5	7	7	7	9	9	9
rlevel	0	1	2	0	1	2	1	2	3
Main ALL	386	265	258	2096	564	177	19070	4379	1318
Coupler Put	2	3	3	3	12	6	305	127	106
Coupler Exchange	36	114	172	4	1	1	18	17	3
Coupler Get	1	2	3	1	1	1	2	6	19
Atmos	338	141	75	2039	538	163	17613	3969	1078

In contrast to Case 1, the scaling factor in high-resolution Case 3 remained mostly constant. It can therefore be concluded that the effect of coupling was minimal.

5.3 NICAM–IO Coupling

The coupling of NICAM and IO components shows that a "user can implement their own interpolation code", which is one of Jcup's features that worked effectively.

Firstly, IO components are described. As mentioned above, NICAM is a model of an icosahedral grid. Since this grid system covers the whole globe with substantially uniform polygons, it is possible to avoid the CLF condition, because there is no singular point as in the latitude–longitude grid. On the other hand, there are disadvantages, such as lack of suitable drawing tools or difficulty in calculating zonal mean values. Therefore, a program named "ico2ll" is provided as one of NICAM's software packages. This program converts each physical quantity of NICAM outputted as an icosahedral grid into the latitude–longitude grid. Ico2ll is executed as a post process after NICAM calculations have been completed. In addition, it is not parallelized and operates in a single process. Consequently, converting calculation results requires time and effort to move and calculate data, and, as a result, it is one of the bottlenecks in research using NICAM. For this reason, a program was developed to more efficiently convert the results of NICAM into a latitude–longitude grid. This program is called NICAMIO. The overview of NICAMIO is shown in Fig.7.

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

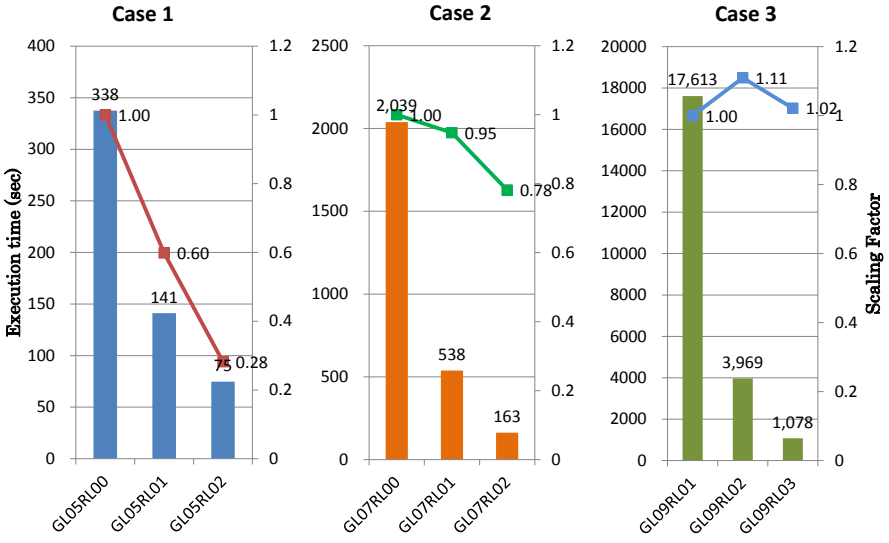

Figure 6. Execution time and scaling factor of "Atmos." Bar graph illustrates execution time (s). Line graph represents scaling factor.

NICAMIO operates in parallel with NICAM in multiple processes, simultaneously converts the result to latitude–longitude grid with the calculations of NICAM, and outputs them to files. Jcup is utilized for coupling NICAM and NICAMIO, and the code for grid conversion is implemented in Jcup's interpolation interface.

The interpolation algorithms implemented for grid transformation of NICAMIO are as follows:

1. Distance weighting method by three grid points
2. Area weighting method
3. Nearest neighbor method

The distance weighting method is an interpolation method used in ico2ll. The mapping table and interpolation coefficient using this formula can be utilized the one calculated for ico2ll.

The area weighting method is an interpolation method used for NICAM-COCO coupling, and the mapping table and interpolation coefficient can be prepared by using the table calculation program for NICAM-COCO coupling as it is.

In the nearest neighbor method, the user can implement an arbitrary interpolation calculation code (in Jcup's design), which worked effectively. When the nearest neighbor method is strictly applied, the nearest point of the lattice point group of NICAM is searched for each latitude and longitude point of NICAMIO, and the result is provided to Jcup as a mapping table. Since this

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

method uses another type of table in addition to the above two kinds of mapping tables, resources, such as a table calculation, a coupler initialization process, or a memory for holding table information, are required. However, as an approximate nearest neighbor method, when using the algorithm of selecting the nearest point from three points of 1), it is possible to respond easily by using the mapping table of 1) and by only partially modifying the distance weighting interpolation code. The interpolation calculation code thus implemented in Jcup is shown in Fig.8. In the left figure, the code that performs the interpolation calculation by the distance weighting method. Here, the following simple calculation is executed: R = R + S * C. In the right figure, among the three points used for distance weight interpolation, the grid point value having the largest coefficient is substituted for the receiving-side grid point value as it is. As shown in the example above, Jcup's design, where interpolation calculation code can be implemented, enables the use of not only a standard interpolation algorithm but also an appropriate interpolation algorithm according to the situation.

An example of outputting NICAM's ground surface temperature by NICAMIO is shown in Fig.9. The upper figure shows the case of interpolation by the distance weighting method and the lower figure shows that of outputting the grid point value of NICAM as it is with the nearest neighbor approximation. In the distance weighting method, the contour lines are smoothed by interpolation, but in the nearest neighbor approximation, the contour lines become more stepped because they are not smoothed.

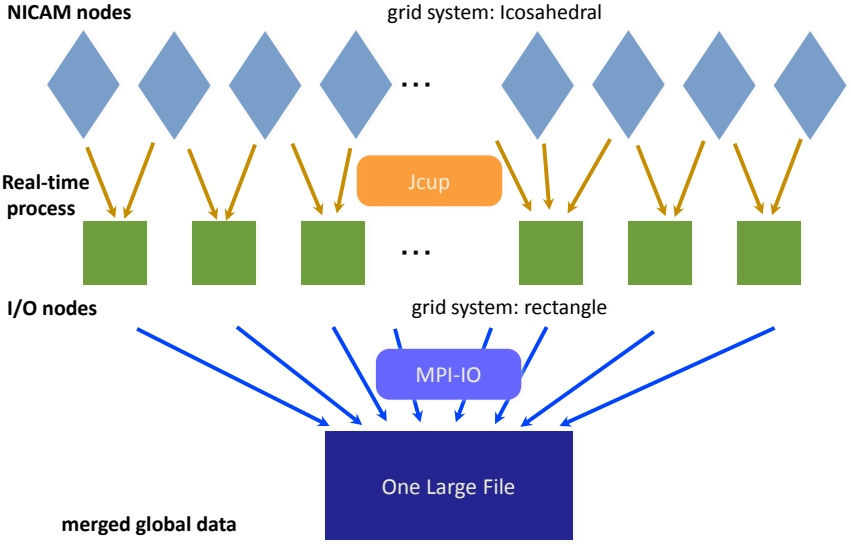

Figure 7. The overview of IO component for NICAM

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

Geoscientific Model Development Discussions

```
do i = 1, size(coi%send_data_index)
  send_point = coi%send_data_index(i)
  recv_point = coi%recv_data_index(i)
  send_coef = coi%send_coef_index(i)

  if (recv_data(recv_point) == UNDEF) cycle

  if (send_data(send_point) == UNDEF) then
    recv_data(recv_point) = UNDEF
  else
    recv_data(recv_point) = recv_data(recv_point)
   &                + send_data(send_point)*coi%coef(i)
  end if

end do
```

```
do j = 1, size(coi%recv_data_index)/3
  max_coef = 0.d0
  max_point = 0

  do i = 1, 3
    send_coef = coi%coef(i + 3*(j-1))
    if (send_coef > max_coef) then
      max_coef = send_coef
      max_point = i + 3*(j-1)
    end if
  end do

  send_point = coi%send_data_index(max_point)
  recv_point = coi%recv_data_index(1+3*(j-1))
  recv_data(recv_point) = send_data(send_point)

end do
```

Figure 8. Interpolation code implemented in NICAMIO. Left panel: distance weighting method; Right panel: nearest neighbor method

6 Conclusions

In this paper, we discussed the design concept and implementation of a coupling software to couple multiple model components for a meteorological/climate model. The main aim of this paper was to indicate how coupling software should be under external and internal conditions. The external conditions are that the developer does not belong to a specific research community, and the software is used generically. The internal conditions are that the coupling software must adapt to changes in the grid system and interpolation algorithm, and the possibility of bug contamination is as low as possible. Conceptual answers to these issues can be summarized as dividing the function of the coupling software into changing and not changing the values of the data and enabling users to manage and implement the function of changing the value as a glass box. Based upon this basic concept, Jcup is constructed so that 1) correspondence relations of grid points in interpolation calculation (mapping table) are utilized as input information and 2) interpolation calculation codes can be freely implemented by users. Jcup is flexible and multifunctional with respect to the function that does not change the value, while leaving the functions changing the value to users. It is consequently possible to couple multiple model components in series and parallel. Furthermore, there is no limit to the assignment pattern of each component in the MPI process. As described above, Jcup has high flexibility with respect to coupling methods and component combinations, but there are restrictions on time. That is, data exchange is performed at predetermined time intervals for each data point, and the time must match the model time. This constraint is unlikely to be

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

Geoscientific Model Development Discussions

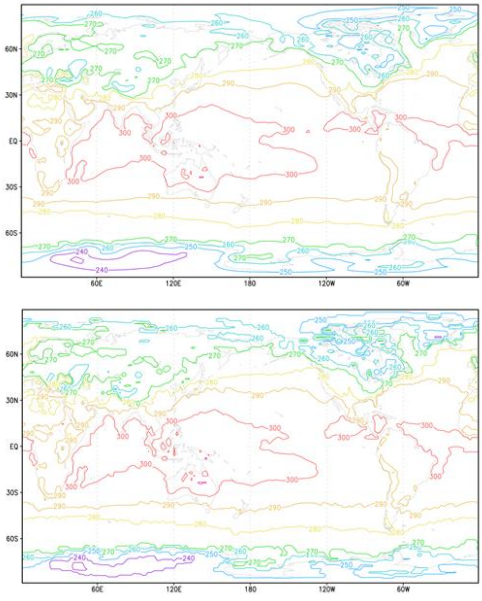

Figure 9. Surface temperature of NICAMIO output. Upper panel: distance weighting method; Lower panel: nearest neighbor method

a problem when the time constant of interaction is long, such as in atmosphere–ocean coupling. However, when physically and computationally intensive interaction is required, such as with the atmosphere and land surface, and the component Δ_T varies irregularly depending on the internal situation of each component, the current Jcup cannot cope. The refinement of each component of the climate model and enlargement of the code will continue, and, as a result, it can be anticipated that components that were not conventionally coupled can also be targets of coupling. Therefore, in the future, Jcup will be extended so that data at arbitrary time intervals can be freely exchanged.

Code availability. The version of Jcup described in this paper is v.3.150100. Jcup code (doi:10.5281/zenodo.1297240) is available from supplement files, and also, available from github website. In addition, for readers who want to tray Jcup, sample programs (doi:10.5281/zenodo.1297250) also available from github website.

The urls of github are as follows.
https://github.com/Jcuplib/jcup/releases/tag/3.150100
https://github.com/Jcuplib/jcup_sample/releases/tag/3.150100

Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-147
Manuscript under review for journal Geosci. Model Dev.
Discussion started: 2 July 2018

Acknowledgements. This research was supported by the Integrated Research Program for Advancing Climate Models (TOUGOU program) of the Ministry of Education, Culture, Sports, Science and Technology, Japan, and is supported by the Japan Science and Technology Agency/Core Research for Evolutional Science and Technology, and the Open Source Infrastructure for Development and Execution of Large-Scale Scientific Applications on Post-Peta-Scale Supercomputers with Automatic Tuning. This research used the computational resources of the High Performance Computing Infrastructure (HPCI) system provided by the Information Technology Center of the University of Tokyo, through the HPCI System Research Project (Project ID:hp120190).

References

Adcroft, A., Campin, J.-M., Hill, C., and Marshall, J.: Implementation of an Atmosphere-Ocean General Circulation Model on the Expanded Spherical Cube, Monthly Weather Review, 132, 2845–2863, 2004.

Arakawa, T., Yoshimura, H., Saito, F., and Ogochi, K.: Data exchange algorithm and software design of KAKUSHIN coupler Jcup, Procedia Computer Science, 4, 1516–1525, 2011.

Baba, Y., Takahashi, K., Sugimura, T., and Goto, K.: Dynamical Core of an Atmospheric General Circulation Model on a Yin-Yang Grid, Monthly Weather Review, online, https://doi.org/10.1175/2010MWR3375.1, 2010.

Edwards, P. N.: History of climate modeling, Wiley Interdisciplinary Reviews: Climate Change, 2, 128–139, https://doi.org/10.1002/wcc.95, http://dx.doi.org/10.1002/wcc.95, 2011.

Graig, A. and Valcke, S.: Development and performance of a new version of the OASIS coupler OASIS3 MCT 3.0, Geoscientific Model Development, 10, 3297–3308, https://doi.org/doi:10.5194/gmd-10-3297-2017, 2017.

Haarsma, R. J., Roberts, M., Vidale, P. L., Senior, C., Bellucci, A., Corti, S., Fuckar, N., Guemas, V., von Hardenberg, J., Hazeleger, W., Kodama, C., Koenigk, T., Leung, R., Lu, J., Luo, J.-J., Mao, J., Mizielinsky, M., Mizuta, R., Nobre, P., Satoh, M., Scoccimarro, E., Semmler, T., Small, J., and von Storch, J.-S.: High Resolution Model Intercomparison Project (HighResMIP v1.0) for CMIP6., Geosci. Model Dev., 9, 4185–4208, https://doi.org/doi:10.5194/gmd-2016-66, 2016.

Hanke, M., Redler, R., Holfeld, T., and Yastremsky, M.: YAC 1.2.0: new aspects for coupling software in Earth system modelling, Geoscientific Model Development, 9, 2755–2769, https://doi.org/10.5194/gmd-9-2755-2016, 2016.

Hasumi, H.: Documentaion for CCSR Ocean Component Model (COCO) Version 4.0s, Center for Climate System Research, 2007.

Jacob, R., Larson, J., and Ong, E.: MxN Communication and Parallel Interpolation in Community Climate System Model Version 3 Using the Model Coupling Toolkit, International Journal of High Performance Computing Applications, 19, 293–307, https://doi.org/10.1177/1094342005056116, 2005.

Kodama, C., Yamada, Y., Noda, A. T., Kikuchi, K., Kajikawa, Y., Nasuno, T., Tomita, T., Yamaura, T., Takahashi, T. G., Hara, M., Kawatani, Y., Satoh, M., and Sugi, M.: A 20-year climatology of a NICAM AMIP-type simulation., J. Meteor. Soc. Japan, 93, 393–424, https://doi.org/doi:10.2151/jmsj.2015-024, 2015.

Kubokawa, H., Satoh, M., Arakawa, T., Suzuki, T., and Hasumi, H.: Air-sea interactions associated with typhoons revealed by a stretched non-hydrostatic atmosphere and ocean coupled model., J. Meteor. Soc. Japan, submitted, 2018.

Larson, J., Jacob, R., and Ong, E.: The Model Coupling Toolkit: A New Fortran90 Toolkit for Building Multiphysics Parallel Coupled Models, International Journal of High Performance Computing Applications, 19, https://doi.org/10.1177/1094342005056115, 2005.

Miyakawa, T., Satoh, M., Miura, H., Tomita, H., Yashiro, H., Noda, A. T., Yamada, Y., Kodama, C., Kimoto, M., and Yoneyama, K.: Madden-Julian Oscillation prediction skill of a new-generation global model., Nature Commun., 5, 3769, https://doi.org/doi:10.1038/ncomms4769, 2014.

Miyakawa, T., Yashiro, H., Suzuki, T., Tatebe, H., and Satoh, M.: A Madden-Julian Oscillation event remotely accelerates ocean upwelling to abruptly terminate the 1997/1998 super El Nino., Geophys. Res. Lett., 44, 9489–9495, https://doi.org/doi:10.1002/2017GL074683, 2017.

Nagai, H., Kobayashi, T., Tsuduki, K., and Terada, H.: Development of Coupled Modeling System for Regional Water Cycle and Material Transport in the Atmospheric, Terrestrial, and Oceanic Environment, Progress in Nuclear Science and Technology, 2, 556–567, 2011.

Satoh, M., Matsuno, T., Tomita, H., Miura, H., Nasuno, T., and Iga, S.: Nonhydrostatic Icosahedral Atmospheric Model (NICAM) for global cloud resolving simulations., Journal of Computational Physics, the special issue on Predicting Weather, Climate and Extreme events, 227, 3486–3514, https://doi.org/doi:10.1016/j.jcp.2007.02.006, 2008.

Satoh, M., Tomita, H., Yashiro, H., Miura, H., Kodama, C., Seiki, T., Noda, A. T., Yamada, Y., Goto, D., Sawada, M., Miyoshi, T., Niwa, Y., Hara, M., Ohno, T., ichi Iga, S., Arakawa, T., Inoue, T., and Kubokawa, H.: The Non-hydrostatic Icosahedral Atmospheric Model: Description and Development, Progress in Earth and Planetary Science, pp. 1–18, https://doi.org/10.1186/s40645-014-0018-1, 2014.

Tomita, H. and Satoh, M.: A new dynamical framework of nonhydrostatic global model using the icosahedral grid., Fluid Dyn. Res., 34, 357–400, 2004.

Valcke, S.: The OASIS3 coupler: a European climate modelling community software, Geoscientific Model Development, 6, 373–388, https://doi.org/doi:10.5194/gmd-6-373-2013, www.geosci-model-dev.net/6/373/2013/, 2013.

Washington, W. M., Buja, L., and Craig, A.: The computational future for climate and Earth system models: on the path to petaflop and beyond, Philosophical Transactions of the Royal Society A, 367, 833–846, 2009.

Watanabe, M., Suzuki, T., O'ishi, R., Komuro, Y., Watanabe, S., Emori, S., Takemura, T., Chikira, M., Ogura, T., Sekiguchi, M., Takata, K., Yamazaki, D., Yokohata, T., Nozawa, T., Hasumi, H., Tatebe, H., and Kimoto, M.: Improved Climate Simulation by MIROC5: Mean States, Variability, and Climate Sensitivity, Journal of Climate, 23, 6312–6335, https://doi.org/10.1175/2010JCLI3679.1, 2010.

Yamazaki, D., Sato, T., Kanae, S., Hirabayashi, Y., and Bates, P. D.: Regional flood dynamics in a bifurcating mega delta simulated in a global river model, Geophysical Research Letters, 41, 3127–3135, https://doi.org/10.1002/2014GL059774, 2014.

Yoshimura, H. and Yukimoto, S.: Development of a Simple Coupler (Scup) for Earth System Modeling, Pap Met Geophys, 59, 19–29, 2008.