

This discussion paper is/has been under review for the journal Geoscientific Model Development (GMD). Please refer to the corresponding final paper in GMD if available.

Improving data transfer for model coupling

C. Zhang^{2,1}, L. Liu^{1,3}, G. Yang^{2,1,3}, R. Li^{2,1}, and B. Wang^{1,3,4}

¹Ministry of Education Key Laboratory for Earth System Modeling, Center for Earth System Science (CESS), Tsinghua University, Beijing, China

²Department of Computer Science and Technology, Tsinghua University, Beijing, China

³Joint Center for Global Change Studies (JCGCS), Beijing, China

⁴State Key Laboratory of Numerical Modelling for Atmospheric Sciences and Geophysical Fluid Dynamics (LASG), Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing, China

Received: 22 September 2015 – Accepted: 30 September 2015 – Published: 20 October 2015

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

Published by Copernicus Publications on behalf of the European Geosciences Union.

GMDD

8, 8981–9020, 2015

Improving data
transfer for model
coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures

⏪

⏩

◀

▶

Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



Abstract

Data transfer, which means transferring data fields between two component models or rearranging data fields among processes of the same component model, is a fundamental operation of a coupler. Most of state-of-the-art coupler versions currently use an implementation based on the point-to-point (P2P) communication of the Message Passing Interface (MPI) (call such an implementation “P2P implementation” for short). In this paper, we reveal the drawbacks of the P2P implementation, including low communication bandwidth due to small message size, variable and big number of MPI messages, and jams during communication. To overcome these drawbacks, we propose a butterfly implementation for data transfer. Although the butterfly implementation can outperform the P2P implementation in many cases, it degrades the performance in some cases because the total message size transferred by the butterfly implementation is larger than that by the P2P implementation. To make the data transfer completely improved, we design and implement an adaptive data transfer library that combines the advantages of both butterfly implementation and P2P implementation. Performance evaluation shows that the adaptive data transfer library significantly improves the performance of data transfer in most cases and does not decrease the performance in any cases. Now the adaptive data transfer library is open to the public and has been imported into a coupler version C-Coupler1 for performance improvement of data transfer. We believe that it can also improve other coupler versions.

1 Introduction

Climate System Models (CSMs) and Earth System Models (ESMs) are fundamental tools for simulating, predicting and projecting the climate. A CSM or an ESM generally integrates several component models, such as an atmosphere model, a land surface model, an ocean model, and a sea-ice model, into a coupled system, to simulate the behaviors of and interactions between components of the climate system. More and

GMDD

8, 8981–9020, 2015

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



more ESMs have sprung up in the world. For example, the number of coupled model versions in the Coupled Model Intercomparison Project (CMIP) has increased from less than 30 (used for CMIP3) to more than 50 (used for CMIP5).

High-performance computing is an essential technical support for model development, especially for higher and higher resolutions of models. Modern high-performance computers integrate an increasing number of processor cores for higher and higher computation performance. Therefore, efficient parallelization, which enables a model to utilize more processor cores for acceleration, becomes a technical focus in model development, and a number of component models with efficient parallelization have sprung up. For example, the Community Ice CodE (CICE; Hunke et al., 2008, 2013) at 0.1° horizontal resolution can scale to 30 000 processor cores on the IBM Blue Gene/L (Dennis et al., 2008); the Parallel Ocean Program (POP; Kerbyson, 2005; Smith et al., 2010) at 0.1° horizontal resolution can also scale to 30 000 processor cores on the IBM Blue Gene/L and to 10 000 processor cores on a Cray XT3 (Dennis, 2007); the Community Atmosphere Model (CAM; Morrison et al., 2008; Neale et al., 2010, 2012) with the spectral element dynamical core (CAM-SE) at 0.25° horizontal resolution can scale to 86 000 processor cores on a Cray XT5 (Dennis et al., 2012). To achieve an efficient parallelization of a coupled model, each component model requires to be efficiently parallelized.

A coupler is an important component in a coupled system. It links component models together to construct a coupled model, and controls the integration of the whole coupled model. A number of couplers now are available for model coupling, e.g., the Model Coupling Toolkit (MCT; Jacob et al., 2015), the Ocean Atmosphere Sea Ice Soil coupling software (OASIS) coupler (Redler et al., 2010; Valcke, 2013), the Earth System Modelling Framework (ESMF; Hill et al., 2004), the CPL6 coupler (Craig et al., 2005), the CPL7 coupler (Craig et al., 2012), the Flexible Modelling System (FMS) coupler (Balaji et al., 2006), the Bespoke Framework Generator (BFG; Ford et al., 2006; Armstrong et al., 2009), and the community coupler version 1 (C-Coupler1; Liu et al., 2014), among others. Most of the existing couplers provide fundamental coupling functions

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



that include data transfer between component models and data interpolation between different model grids (Valcke et al., 2012).

A coupler generally has much smaller overhead than other component models. However, it is potentially a time-consuming component in an ESM in future. This is because there will be more and more component models (such as land-ice model, chemistry model and biogeochemical model) coupled into an ESM and the coupling between component models will be more and more frequent. Data transfer is a fundamental and most frequently used operation in a coupler. It is responsible for transferring data fields between the processes of component models and responsible for rearranging data fields among various processes of the same component model for parallel data interpolation.

A coupler may become a bottleneck for efficient parallelization of future coupled models. The most obvious reason is that the current implementation of data transfer in a state-of-the-art coupler is not efficient enough. For example, the data transfer from a component with a logically rectangular grid (of 1021×1442 grid points) to a component with a Gaussian Reduced T799 grid (with 843 000 grid points) can only scale to about 100 processor cores when using OASIS3 (Valcke, 2013) and to about 1000 processor cores when using OASIS3-MCT (Valcke et al., 2013); the data transfer from a component model with a horizontal grid (of 576×384 grid points) to another component model with another horizontal grid (of 3600×2400 grid points) can only scale to about 500 processor cores when using the CPL7 coupler (Craig et al., 2012). Therefore, it is highly desirable to improve the parallelization of couplers.

In this study, we propose a butterfly implementation of data transfer and then develop an adaptive data transfer library that is open to the public. Performance evaluation demonstrates that such a library significantly improves the performance of data transfer in most cases and does not decrease the performance in any cases. This library has been imported into C-Coupler1 with slight code modification. We believe it can be easily imported into other coupler versions for better performance of data transfer.

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



The reminder of this paper is organized as follows. We briefly introduce the implementation of data transfer in existing couplers in Sect. 2. We analyze performance bottlenecks of the existing implementation in Sect. 3. Details of the butterfly implementation and the adaptive data transfer library are presented in Sects. 4 and 5, respectively. The performance of the butterfly implementation and the adaptive data transfer library is evaluated in Sect. 6. Conclusion is given in Sect. 7.

2 Implementation of data transfer in existing couplers

In this section, we focus on the implementation of data transfer in existing couplers, including MCT (Jacob et al., 2015), the OASIS coupler (Redler et al., 2010; Valcke, 2013; Valcke et al., 2013), ESMF (Hill et al., 2004), the FMS coupler (Balaji et al., 2006), the CPL6 coupler (Craig et al., 2005), the CPL7 coupler (Craig et al., 2012), and C-Coupler1 (Liu et al., 2014). More details of these couplers can be found in the citations given.

2.1 MCT

MCT works as a library for model coupling. It can be directly used to construct a coupled model with different component models, and can also be used to develop other couplers, such as OASIS3-MCT, the CPL6 coupler and the CPL7 coupler. It provides fundamental coupling functions, i.e., data transfer and data interpolation, in parallel. To achieve a parallel data transfer, MCT first generates a communication router (known as the data mapping between processes) according to the parallel decompositions of the two component models, and next uses the point-to-point (P2P) communication of the Message Passing Interface (MPI) to transfer data. A data field will be transferred from a process of the source component model to a process of the target component model, only when the two processes have common grid points. A data transfer can serve mul-

multiple data fields that will be packed into one MPI message for better communication performance.

On the other hand, parallel interpolation can also introduce data exchange among processes of the same component model. Interpolation is generally performed by the calculation of matrix–vector multiplication. To achieve efficient parallelization of interpolation, MCT can rearrange the layout of the data field among processes, to enable the matrix-vector multiplication to be performed locally on each process. The data rearrangement is essentially a data transfer.

2.2 The OASIS coupler

The OASIS coupler is mainly developed by the European Centre for Research and Advanced Training in Scientific Computing (CERFACS) since 1991. OASIS3 (Valcke, 2013a) is a 2-D version of the OASIS coupler with broad usage. To transfer a field from one component model to another, a process of OASIS3 first gathers the field from the processes of the source component model and then scatters the field to the processes of the target component model. Each process of OASIS3 can transfer one model field, so that multiple model fields can be transferred in parallel. However, the parallelism of such an implementation is limited by the number of coupling fields. To solve this problem, MCT has been used to develop the latest version of the OASIS coupler (OASIS3-MCT).

OASIS4 is a 3-D version of the OASIS coupler. The data exchange library in the PRISM System Model Interface Library (PSMILe; Redler, 2010), which performs communication with MPI, is used to perform the data transfer in OSIS4. Similar to MCT, each process only needs to send or receive the data of its local decomposition.

In OASIS3, the interpolation of a field is carried out by only one process. Like the implementation of data transfer in OASIS3, the data needed interpolation will be gathered from all processes of the corresponding component model before the interpolation, and will be scattered to all processes after the interpolation. In OASIS4 and OASIS3-MCT, the interpolation is performed in parallel, where all processes of the corresponding

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



component model cooperatively perform the interpolation at the same time. The data rearrangement for the parallel interpolation is implemented by PSMILe in OASIS4 and by MCT in OASIS3-MCT.

2.3 ESMF

5 Earth System Modeling Framework (ESMF) is a widely used software framework for model development, which defines a superstructure for the architecture of component models and an infrastructure with common coupling functions for model coupling. In ESMF, the coupler components are responsible for regridding and transferring data among component models. The coupler components build the corresponding relationship between the data of the source model and the data of the target model according to their parallel decomposition. Then, the data are transferred in parallel according to the corresponding relationship.

2.4 The FMS coupler

15 FMS is a software framework developed by the Geophysical Fluid Dynamics Laboratory (GFDL). It supports the development, construction, execution, and scientific interpretation of models. The FMS coupler deploys an exchange grid to perform the coupling. Given the grids of two component models, their exchange grid is generated by all the vertices in the two grids. The coupling fields from a source component model to a target component model are first interpolated onto the exchange grid, and then averaged onto the target grid. Data transfer among different processors is performed with MPI P2P communications.

2.5 The CPL6 coupler

25 The CPL6 coupler is a centralized coupler for the Community Climate System Model version 3 (CCSM3; Collins et al., 2006) developed at the National Center for Atmospheric Research (NCAR). The data transfer between component models must go

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures

⏪

⏩

◀

▶

Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



through the coupler. The CPL6 coupler integrates MCT for data transfer and data interpolation. Therefore, the data transfer between component models is processed in parallel with MPI P2P communications and can serve multiple model fields at the same time for better communication performance.

2.6 The CPL7 coupler

The CPL7 coupler is the latest coupler version from the NCAR. It has been used for the ESMs of the Community Climate System Model version 4 (CCSM4; Gent et al., 2011) and the Community Earth System Model (CESM; Hurrell et al., 2013). Similar to the CPL6 coupler, the CPL7 coupler is also a centralized coupler, where the data transfer between component models must go through the coupler. The CPL7 coupler also integrates MCT for data transfer and data interpolation. Moreover, the CPL7 coupler supports the coupling interface based on ESMF and can use the coupling functions in ESMF for data transfer and data interpolation.

2.7 C-Coupler1

C-Coupler1 is a Chinese community coupler for Earth system modeling. It achieves 3-D coupling with flexible 3-D interpolation, and supports direct coupling without a specific coupler component to improve the parallel performance. Its implementation of data transfer is derived from the corresponding implementation in MCT. In other words, C-Coupler1 first generates a communication router according to the parallel decompositions of the component models, and then uses the MPI P2P communication to transfer the coupling fields in parallel. To further improve the communication performance, model fields with different data types, different model grids, or different parallel decompositions can be served by the same data transfer.

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



3 Performance bottlenecks of existing implementations

The implementations of data transfer in the state-of-the-art couplers are similar, which can be concluded as the MPI P2P communication that transfers data among the processes according to the two corresponding parallel decompositions. In the following context, we call such an implementation “P2P implementation” for short. To reveal why the P2P implementation is inefficient, we first derive a benchmark from a real coupled model version GAMIL2-CLM3, where GAMIL2 (Li et al., 2013) is an atmosphere model and CLM3 (Oleson et al., 2004) is a land surface model. GAMIL2 and CLM3 share the same horizontal grid of 7680 (128 × 60) grid points.

In this benchmark, there is only the data transfer with P2P implementation between two data models with the same grid as the horizontal grid of GAMIL2-CLM3. The parallel decompositions of the source and target data models are the same as those of CLM3 and GAMIL2, respectively. A high-performance computer named Tansuo100 at Tsinghua University, China is used for the performance testing. It has 700 computing nodes, each of which contains two six-core Intel Xeon X5670 CPUs and 32 GB main memory. All computing nodes are connected by a high-speed InfiniBand network with peak communication bandwidth of 5 GBs⁻¹.

To evaluate the parallel performance of the P2P implementation, 14 2-D coupling fields are transferred between the two data models. In each test, the two data models have the same number of processes. As there are 12 CPU cores on each computing node, the number of processes is set to be an integral multiple of 12. When the process number is less than 12, the two data models are located on two different computing nodes. The two data models do not share the same computing node, so the communication of the P2P implementation must go through the InfiniBand network.

Figure 1 demonstrates the poor performance of the P2P implementation. It is well known that the performance of communication heavily depends on message size. As shown in Fig. 2, the communication bandwidth achieved generally increases with message size; so when the message size is small (for example, smaller than 4 KB), the

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



communication bandwidth achieved is very low. The message size in the P2P implementation decreases with increment of process number of models (Fig. 3), indicating that the communication bandwidth gets lower with increase of process number. The performance of a data transfer also heavily depends on the number of MPI messages. As shown in Fig. 4, the number of MPI messages in the P2P implementation increases with increment of process number. Here, we may conclude that the decrease of message size and the increase of number of MPI messages are primary reasons for the poor performance of the P2P implementation when increasing the process number. However, the ideal performance shown in Fig. 5 is much better than the actual performance. The ratio between the ideal performance and actual performance significantly increases with the increment of processor number. The significant gap between the ideal performance and actual performance is due to the jam of network communication. For example, when multiple P2P communications share the same source process or target process, they must wait in an order.

4 Butterfly implementation for better performance of data transfer

To improve the performance of data transfer, a new implementation should be able to overcome the drawbacks of the P2P implementation, which can be concluded as low communication bandwidth due to small message size, variable and big number of MPI messages, and jams in communications. We therefore propose a new implementation called the butterfly implementation. As shown in Fig. 6, it is similar to the butterfly diagram in Fast Fourier Transform (FFT; Heckbert, 1995). The most significant challenge to the butterfly implementation is that the process number needs to be 2^n , where n is a non-negative integer, while the process number of data transfer generally can be any positive integer. To resolve this challenge, we investigated how to efficiently map processes between the butterfly implementation and the sender/receiver. Next, we will introduce the butterfly implementation and the process mapping.

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



4.1 The butterfly implementation

The butterfly implementation aims to rearrange the data from the source parallel decomposition to the target parallel decomposition. As shown in Fig. 6, there are multiple stages in the butterfly implementation. Given the process number $N = 2^n$, the number of stages is $n + 1$. Each stage has a unique parallel decomposition. The parallel decompositions of the first stage and last stage are determined by the source and target parallel decompositions, respectively, while the parallel decompositions of the other stages are determined by the first and last stages. Between any two successive stages, all processes are split into a number of pairs and the two processes of each pair exchange data according to the corresponding parallel decompositions using MPI P2P communication.

Compared to the existing implementations of data transfer, the butterfly implementation has the following advantages:

1. Bigger message size for better communication bandwidth. The message size is $M/(2N)$ on average, where M is the total size of data to be transferred and N is the process number.
2. Balanced number of MPI messages among processes. Each process performs $\log_2 N$ times of MPI communication.
3. Ordered communications among processes and fewer communications operated concurrently. The jam of network communication can be dramatically reduced.

4.2 Process mapping

Process number of the butterfly kernel must be 2^n , where n is a non-negative integer, while process number of sender or receiver can be any positive integer. The first question is how to decide the number of processes of the butterfly kernel? Any process of the sender or receiver can be used as a process of the butterfly kernel. Given that

GMDD

8, 8981–9020, 2015

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



GMDD

8, 8981–9020, 2015

Improving data transfer for model coupling

C. Zhang et al.

[Title Page](#)

[Abstract](#)

[Introduction](#)

[Conclusions](#)

[References](#)

[Tables](#)

[Figures](#)



[Back](#)

[Close](#)

[Full Screen / Esc](#)

[Printer-friendly Version](#)

[Interactive Discussion](#)



the total number of unique processes of the sender and receiver is N_T , the process number of the butterfly kernel (N_B) can be any power of 2, which is no larger than N_T . For example, we can select the maximum number in order for maximum utilization of resources. When $N_B < N_T$, we prefer to pick out processes first from the sender, and then from the receiver if the sender does not have enough processes, in order to save the overhead of process mapping from the sender to the butterfly kernel.

The second question is how to decide process mapping from the sender to the butterfly kernel and from the butterfly kernel to the receiver. To minimize the overhead of process mapping from the butterfly kernel to the receiver, we make one or multiple processes of the butterfly kernel map to a process of the receiver if the butterfly kernel has more processes than the receiver; otherwise, we make a process of the butterfly kernel map to one or multiple processes of the receiver. In other words, there is no multiple-to-multiple process mapping between the butterfly kernel and the receiver. Similarly, there is no multiple-to-multiple process mapping between the sender and the butterfly kernel. Processes of the sender or receiver may be unbalanced in terms of size of the data transferred, which may result in unbalanced communications between processes of the butterfly kernel.

As mentioned in Sect. 4.1, at each stage of the butterfly kernel, all processes are split into a number of pairs, each of which is involved in P2P communications. To improve the balance of communications among the processes, one solution is to try to make the process pairs at each stage more balanced in terms of data size of P2P communications. To achieve balanced data size among process pairs, we propose to take consideration of the sorting order of the processes in terms of data size. For example, for the remaining processes that have not been paired, we can pair the process with the largest data size and the process with the smallest data size. The pairing of the processes should be conducted iteratively among stages of the butterfly kernel. All processes are taken as the input for the first stage, while output of the pairing for one stage will be the input for the next stage. After finishing the iterative pairing through all stages, all processes of the sender or receiver are reordered.

Improving data transfer for model coupling

C. Zhang et al.

[Title Page](#)[Abstract](#)[Introduction](#)[Conclusions](#)[References](#)[Tables](#)[Figures](#)[Back](#)[Close](#)[Full Screen / Esc](#)[Printer-friendly Version](#)[Interactive Discussion](#)

The iterative pairing also requires the number of processes be a power of 2. Given that the number of processes of the sender (or receiver) is N_C and the process number of the butterfly kernel is N_B , we propose to first pad empty processes (the data size is 0) before the iterative pairing to make the number of the processes for the sender (or receiver) be a power of 2 (denoted N_P), which is no smaller than N_B . Therefore, the reordered N_P processes after the iterative pairing can be divided into N_B groups, each of which contains N_P/N_B processes with consecutive reordered indexes and maps to a unique process of the butterfly kernel. Figure 7 shows an example for further illustration of process mapping.

5 Adaptive data transfer library

Now, we have two implementations (the P2P implementation and the butterfly implementation) for data transfer. Although the butterfly implementation can effectively improve the performance of data transfer, it still has some drawbacks: (1) it generally has a larger total message size of communications than the P2P implementation; (2) its stage number is $\log_2 N$ (N is the number of processes for the butterfly kernel), which may be bigger than the average number of MPI messages per process in the P2P implementation. Therefore, it is possible that the P2P implementation outperforms the butterfly implementation in some cases (examples are given in Sect. 6). To achieve optimal performance for data transfer, we propose an adaptive data transfer library that can keep the advantages of the two implementations in all cases.

As introduced in Sect. 4, the butterfly implementation is divided into multiple stages. Each stage has a unique intermediate parallel decomposition. Actually, the data transfer between two successive stages can be viewed as a P2P implementation with only one MPI message per process. Inspired by this fact, we try to design an adaptive approach that can combine the butterfly and P2P implementations, where some stages in the butterfly implementation are skipped with the P2P implementations of more MPI messages per process. Figure 8 shows an example of the adaptive data transfer li-

brary with 8 processes, where Stage 1 of the butterfly implementation is skipped with the P2P implementation of 3 MPI messages per process.

The most significant challenge to such an adaptive approach is how to determine which stage(s) of the butterfly implementation should be skipped. The first solution is to design a cost model that can accurately predict the performance of data transfer in various implementations. We eventually gave up this solution because it was almost impossible to accurately predict the performance of the communications on a high-performance computer, especially when a lot of users share the computer to run various applications. Profiling which means directly measuring the performance of data transfer is more practical to determine an appropriate implementation, because the simulation for earth system modeling always takes a long time to run. To obtain an appropriate implementation of the adaptive data transfer library, we try to successively skip the stages of the butterfly implementation. If skipping one stage can achieve better performance, this stage will be skipped; otherwise, it will be kept. Figure 9 shows a flowchart for determining an appropriate implementation of the adaptive data transfer library. In the algorithm, a stage mask array (Stage_mask in the flowchart) specifies which stages are skipped. In detail, each array element corresponds to a stage of the butterfly implementation. If the value of an array element is false, its corresponding stage is skipped with a P2P implementation. Otherwise, its corresponding stage is kept.

The source code of the adaptive data transfer library is mainly written in C++, while the application programming interfaces (APIs) are written in Fortran because most couplers and models are programmed in Fortran. Table 1 lists the APIs, and Fig. 10 shows an example of how to use these APIs. The adaptive data transfer library can transfer 2-D and 3-D fields at the same time. Now, it is publicly available at a website (see the code availability section).

GMDD

8, 8981–9020, 2015

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



6 Performance evaluation

In order to improve the performance of data transfer for model coupling, we propose the butterfly implementation and an adaptive data transfer library that combines the butterfly implementation and the traditional P2P implementation. In this section, we empirically evaluate the adaptive data transfer library, through comparing it to the butterfly implementation and P2P implementation. Both toy models and realistic models (GAMIL2-CLM3 and CESM) are used for the performance evaluation. GAMIL2-CLM3 has been introduced in Sect. 3. CESM is a state-of-the-art ESM developed by the NCAR. All the experiments are run on the high performance computer Tansuo100 that has been introduced in Sect. 3.

In the following context, we will respectively evaluate the overhead of initialization, the performance in data transfer and the performance in data rearrangement for interpolation.

6.1 Overhead of initialization

We first evaluate the overhead of initialization of different implementations of data transfer. As shown in Fig. 11, the overheads of initialization of all the three implementations increase with the increment of core number. The initialization overhead of the butterfly implementation is a little higher than that of the P2P implementation, while the initialization overhead of the adaptive data transfer library is 4–5 folds higher than that of the P2P implementation, because the adaptive data transfer library uses extra time on performance profiling. Considering that one data transfer instance should be initialized only one time at the beginning and executed many times in an ESM, we can conclude that the initialization overhead of the adaptive data transfer library is reasonable, especially when the simulation is executed for a very long time.

GMDD

8, 8981–9020, 2015

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



6.2 Performance of data transfer between toy models

In this subsection, we evaluate the performance of data transfer (excluding the initialization overhead) with two toy models that use the same logically rectangular grid (of 192×96 grid points). In each test, the two toy models have the same process number and each process has the same MPI message number. The MPI message number of one process can be modified through adjusting the parallel decompositions of the toy models. The factors that impact the performance of a data transfer implementation include the commutation number, the size of the data to be transferred (also known as the number of fields in this evaluation) and the number of processes. Next, we evaluate the performance of data transfer through varying these factors.

Given a fixed process number of 192 and a fixed 2-D coupling field number of 10, Fig. 12 shows the execution time of one data transfer of different implementations when varying the MPI message number of each process from 1 to 96. The P2P implementation can outperform the butterfly implementation when the MPI message number is small (say, smaller than 12 in Fig. 12), while the butterfly implementation can outperform the P2P implementation when the MPI message number is big (say, bigger than 12 in Fig. 12). Our adaptive data transfer library can completely keep the best performance of the P2P and butterfly implementations. Moreover, it further improves the performance based on the butterfly implementation when the MPI message number is big, because some butterfly stages in the adaptive data transfer library have been skipped with the P2P implementation. When the MPI message number per process is 96, the adaptive data transfer library can achieve a 13.9-fold performance speedup compared to the P2P implementation.

Given different numbers of processes and different numbers of MPI messages per process, Fig. 13 shows the execution time of one data transfer in different implementations when varying the number of 2-D coupling fields to be transferred. The results show that the execution time of each implementation increases with the increment of data size. When the MPI message number per process is small (Fig. 13a and b), the

GMDD

8, 8981–9020, 2015

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



performance of the butterfly implementation is poorer than that of the P2P implementation, especially when the number of 2-D coupling fields gets bigger. However, the adaptive data transfer library achieves similar performance with the P2P implementation. When the MPI message number per process is big (Fig. 13c and d), both the butterfly implementation and adaptive data transfer library significantly outperform the P2P implementation, and the adaptive data transfer library further achieves better performance than the butterfly implementation.

Given a fixed MPI message number per process 24 and a fixed 2-D coupling field number 10, Fig. 14 shows the execution time of one data transfer in different implementations when varying the number of cores. The results show that both the butterfly implementation and adaptive data transfer library achieve better parallel scalability than the P2P implementation. The execution time of the P2P implementation slightly increases with the increment of the number of cores used. However, the execution times of the butterfly implementation and adaptive data transfer library slightly decrease with the increment of the number of the cores used. The butterfly implementation outperforms the P2P implementation, and the adaptive data transfer library achieves better performance than the butterfly implementation.

6.3 Performance of data transfer between realistic models

Previous evaluation with toy models reveals that the adaptive data transfer library can achieve the best performance among different implementations. In this subsection, we evaluate the performance with two realistic models: GAMIL2-CLM3 (horizontal resolution of $2.8^\circ \times 2.8^\circ$) and CESM (resolution of $1.9 \times 2.5_{\text{gx1v6}}$). For CESM, we use the data transfer between the coupler CPL7 (Craig et al., 2012) and the land surface model CLM4 (Oleson et al., 2004), where 32 2-D coupling fields on the CLM4 horizontal grid (the grid size is $144 \times 96 = 13824$) are transferred. Figure 15 shows the performance of one data transfer of different implementations when increasing the process number of both CPL7 and CLM4 from 6 to 192. When the process number is small (say, smaller than 24 in Fig. 15), the butterfly implementation is much poorer than the P2P

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



implementation, and the adaptive data transfer library achieves similar performance as the P2P implementation. However, when the process number gets bigger (say, larger than 24 in Fig. 15), the adaptive data transfer library dramatically outperforms the P2P implementation with more speedup and also outperforms the butterfly implementation.

5 When each component uses 192 cores, the adaptive data transfer library is 4.01 times faster than the P2P implementation.

For GAMIL2-CLM3, we use the data transfer from CLM3 to GAMIL2 where 14 2-D coupling fields on the GAMIL2 horizontal grid (the grid size is $128 \times 60 = 7680$) are transferred. Figure 16 shows the execution time of one data transfer of each implementation when increasing the process number of both GAMIL2 and CLM3 from 6
10 to 192. The results in Fig. 16 confirm that the adaptive data transfer library can constantly keep the best performance among different implementations. Compared to the P2P implementation, the adaptive data transfer library achieves an 11.68-fold performance speedup when the process number is 96, but achieves a much lower speedup (only 3.48-fold) when the process number is 192. This is because that the average MPI message number per process reduces from 32 to 18 when the number of process increases from 96 to 192.

6.4 Performance of data rearrangement for interpolation

For model coupling, besides the data transfer between different component models,
20 there is the other kind of data transfer that rearranges the data inside a model in order for parallel interpolation of fields between different grids. Here, we use the data rearrangement for the parallel interpolation from the atmosphere grid (the grid size is $144 \times 96 = 13\,824$) to the ocean grid (the grid size is $320 \times 384 = 122\,880$) in the coupled model CESM for further evaluation. The results show that the butterfly implementation
25 is much poorer than the P2P implementation (Fig. 17). This is because the MPI message number is very small (for example, average MPI message number per process is only 6.49 when each model uses 96 cores) for data rearrangement. As a result,

GMDD

8, 8981–9020, 2015

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



the adaptive data transfer library achieves almost the same performance as the P2P implementation.

7 Conclusion

Data transfer is the fundamental and most frequently used operation in a coupler. This paper demonstrated the current implementation (which is named as the P2P implementation in this paper) of data transfer in most state-of-the-art couplers is not efficient. To improve the parallel performance of data transfer, we proposed a butterfly implementation. However, the butterfly implementation has advantages and disadvantages, comparing with the P2P implementation. The evaluation results showed that the butterfly implementation did not always outperform the P2P implementation. To completely achieve better parallel performance of data transfer, we built an adaptive data transfer library, which combines the advantages of the butterfly implementation and P2P implementation. The evaluation results demonstrated the adaptive data transfer library can always achieve the best performance, comparing with the butterfly implementation and P2P implementation. That is to say the adaptive data transfer library can effectively improve the performance of data transfer in model coupling.

Code availability

The source code of the adaptive data transfer library is available at https://github.com/zhang-cheng09/Data_transfer_lib.

Acknowledgement. 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).

GMDD

8, 8981–9020, 2015

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



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.:
5 The Exchange Grid: a mechanism for data exchange between Earth system components on independent grids, in: *Parallel Computational Fluid Dynamics 2005 Theory and Applications*, 2006, 179–186, doi:10.1016/B978-044452206-1/50021-5, 2006.
- 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.,
10 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., Kauffman, B., Bettge, T., Larson, J., Ong, E., Ding, C., 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.
- 15 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.
- Dennis, J. M.: Inverse space-filling curve partitioning of a global ocean model, In *IEEE International Parallel and Distributed Processing Symposium*, Long Beach, CA, 2007.
- 20 Dennis, J. M. and Tufo, H. M.: Scaling climate simulation applications on the IBM Blue Gene/L system, *IBM J. Res. Dev.*, 52, 117–126, doi:10.1147/rd.521.0117, 2008.
- Dennis, J. M., Edwards, J., Evans, K. J., Guba, O., Lauritzen, P. H., Mirin, A. A., St-Cyr, A., Taylor, M. A., and Worley, P. H.: CAM-SE: a scalable spectral element dynamical core for the Community Atmosphere Model, *Int. J. High Perform. C.*, 26, 74–89, doi:10.1177/1094342011428142, 2012.
- 25 Dickinson, R. E., Oleson, K. W., Bonan, G., Hoffman, F., Thornton, P., Vertenstein, M., Yang, Z. L., and Zeng, X.: The community land surface model and its climate statistics as a component of the Community Climate System Model, *J. Climate*, 19, 2302–2324, 2006.
- 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.
- 30 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

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



Zhang, M.: The Community Climate System Model version 4, *J. Climate*, 24, 4973–4991, 2011.

Heckbert, P.: Fourier transforms and the Fast Fourier Transform (FFT) algorithm, *Comp. Graph.*, 2, 15–463, 1995.

5 Hill, C., DeLuca, C., Balaji, V., Suarez, M., and da Silva, A.: The 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.,
10 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.

Hunke, E. C. and Lipscomb, W. H.: CICE: the Los Alamos Sea Ice Model Documentation and Software User's Manual 4.0, Tech. Rep. LA-CC-06-012, Los Alamos National Laboratory, T-3 Fluid Dynamics Group, 2008.

15 Hunke, E. C., Lipscomb, W. H., Turner, A. K., Jeffery, N., and Elliott, S.: CICE: the Los Alamos Sea Ice Model Documentation and Software User's Manual Version 5.0, LA-CC-06-012, Los Alamos National Laboratory, Los Alamos NM, 87545, 115, 2013

Jacob, R., Larson, J., and Ong, E.: M × 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.

20 Kerbyson, D. J. and Jones, P. W.: A performance model of the parallel ocean program, *Int. J. High Perform. C.*, 19, 261–276, doi:10.1177/1094342005056114, 2005.

Li, L. J., Wang, B., Dong, L., Liu, L., Shen, S., Hu, N., Sun, W., Wang, Y., Huang, W., Shi, X., Pu, Y., and Yang, G.: 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, 2013.

25 Liu, L., Yang, G., Wang, B., Zhang, C., Li, R., Zhang, Z., Ji, Y., and Wang, L.: C-Coupler1: a Chinese community coupler for Earth system modeling, *Geosci. Model Dev.*, 7, 2281–2302, doi:10.5194/gmd-7-2281-2014, 2014.

Morrison, H. and Gettelman, A.: A new two-moment bulk stratiform cloud microphysics scheme in the Community Atmosphere Model, version 3 (CAM3). Part I: Description and numerical tests, *J. Climate*, 21, 3642–3659, doi:10.1175/2008JCLI2105.1, 2008.

30 Neale, R. B., Richter, J. H., Conley, A. J., Park, S., Lauritzen, P. H., Gettelman, A., Williamson, D. L., Rasch, P. J., Vavrus, S. J., Taylor, M. A., Collins, W. D., Zhang, M., and Lin, S.: Description

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures



Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



of the NCAR Community Atmosphere Model (CAM 4.0), National Center for Atmospheric Research Ncar Koha Opencat, TN-485+STR, 222 p., 2010.

Neale, R. B., Chen, C. C., Gettelman, A., Lauritzen, P. H., Park, S., Williamson, D. L., Conley, A. J., Garcia, R., Kinnison, D., Lamarque, J. F., Marsh, D., Mills, M., Smith, A. K., Tilmes, S., Vitt, F., Morrison, H., Cameron-Smith, P., Collins, W. D., Iacono, M. J., Easter, R. C., Ghan, S. J., Liu, X., Rasch, P. J., and Taylor, M. A.: Description of the NCAR Community Atmosphere Model (CAM 5.0), National Center for Atmospheric Research Ncar Koha Opencat, TN-486+STR, 289 p., 2012.

Oleson, K. W., Dai, Y., Bonan, G., Bosilovich, M., Dickinson, R., Dirmeyer, P., Hoffman, F., Houser, P., Levis, S., Niu, G. Y., Thornton, P., Vertenstein, M., Yang, Z. L., and Zeng, X.: Technical Description of the Community Land Surface Model (CLM), National Center for Atmospheric Research Ncar Koha Opencat, TN-461+STR, 186 p., 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.

Smith, R., Jones, P., Briegleb, B., Bryan, F., Danabasoglu, G., Dennis, J., Dukowicz, J., Eden, C., Fox-Kemper, B., Gent, P., Hecht, M., Jayne, S., Jochum, M., Large, W., Lindsay, K., Maltrud, M., Norton, N., Peacock, S., Vertenstein, M., and Yeager, S.: The Parallel Ocean Program (POP) reference manual ocean component of the Community Climate System Model (CCSM) and Community Earth System Model (CESM), Los Alamos National Laboratory, LAUR-10-01853, available at <http://www.cesm.ucar.edu/models/cesm1.1/pop2/doc/sci/POPRefManual.pdf> (last access: 15 October 2015), 141 p., 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, 2013.

Valcke, S., Balaji, V., Craig, A., DeLuca, C., Dunlap, R., Ford, R. W., Jacob, R., Larson, J., O’Kuinghtons, 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, 2012.

Valcke, S., Craig, T., and Coquart, L.: The OASIS3-MCT parallel coupler, in: The Second Workshop on Coupling Technologies for Earth System Models (CW2013), available at: https://wiki.cc.gatech.edu/CW2013/images/a/a0/OASIS_MCT_abstract.pdf (last access: 15 October 2015), 2013.

Improving data transfer for model coupling

C. Zhang et al.

[Title Page](#)

[Abstract](#)

[Introduction](#)

[Conclusions](#)

[References](#)

[Tables](#)

[Figures](#)

[⏪](#)

[⏩](#)

[◀](#)

[▶](#)

[Back](#)

[Close](#)

[Full Screen / Esc](#)

[Printer-friendly Version](#)

[Interactive Discussion](#)



Table 1. The application program interfaces (APIs) of the adaptive data transfer library. Couplers or component models can improve the performance of data transfer through calling these APIs.

API	Brief description	Parameter description
instance_id = data_transfer_register_instance (local_comm, global_rank_remote_root, action)	This API registers one data transfer instance and returns the index of this data transfer instance. A component model can register multiple different data transfer instances.	This API takes local communicator <i>local_comm</i> , global rank of the root process in the remote model <i>global_rank_remote_root</i> and the transfer direction <i>action</i> (send, recv or sendrecv) as input, and returns the instance index <i>instance_id</i> .
call data_transfer_register_decomp (instance_id, num_grid_cells, num_local_cells, local_cells_global_index)	This API registers one parallel decomposition to one data transfer instance.	This API takes the instance index <i>instance_id</i> , the number of grid cells <i>num_grid_cells</i> , the number of local cells <i>num_local_cells</i> and the global index of local cells <i>local_cells_global_index</i> as input.
call data_transfer_register_field (instance_id, data_buf, input)	This API registers a coupling field to enable one data transfer instance to access the memory space of this field. One data transfer instance can register multiple coupling fields.	This API takes the instance index <i>instance_id</i> , the memory space of this field <i>data_buf</i> and the action of this field <i>input</i> (true stands for input field and false stands for output field) as input.
call data_transfer_register_mask (instance_id, mask_array)	This API registers a mask array to enable one data transfer instance to transfer different coupling fields at different coupling steps.	This API takes the instance index <i>instance_id</i> and the mask array <i>mask_array</i> as input.
call data_transfer_init_instance (instance_id)	This API initializes one data transfer instance.	This API takes the instance index <i>instance_id</i> as input.
call data_transfer_exec_instance (instance_id)	This API executes one data transfer instance.	This API takes the instance index <i>instance_id</i> as input.
call data_transfer_final_instance (instance_id)	This API finalizes one data transfer instance.	This API takes the instance index <i>instance_id</i> as input.

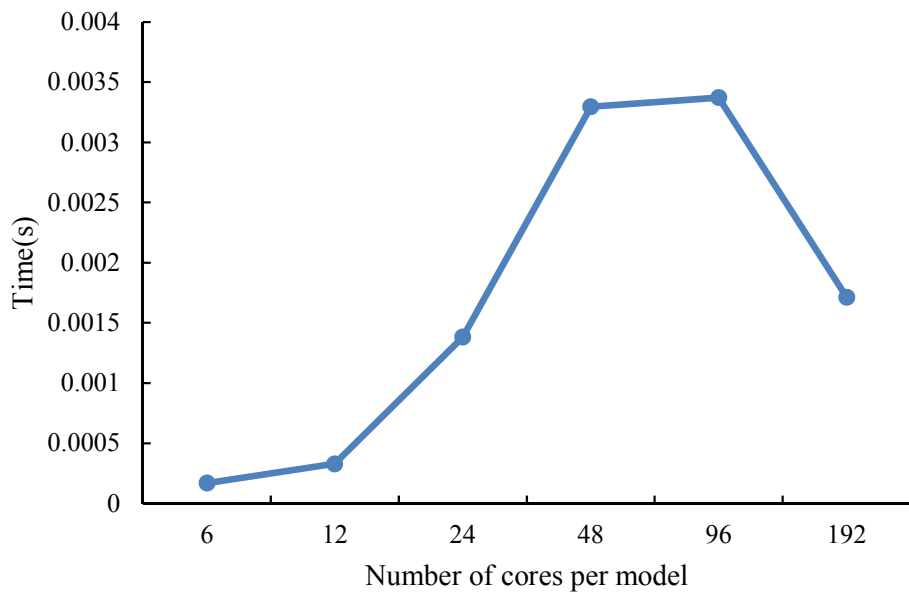


Figure 1. Average execution time of the P2P implementation when transferring 14 2-D fields from CLM3 to GAMIL2. In each test, the atmosphere model GAMIL2 and the land surface model CLM3 use the same number of cores and do not share the same computing node. The horizontal grid of the 14 2-D fields contains 7680 (128 × 60) grid points.

Improving data transfer for model coupling

C. Zhang et al.

[Title Page](#)

[Abstract](#) [Introduction](#)

[Conclusions](#) [References](#)

[Tables](#) [Figures](#)

[◀](#) [▶](#)

[◀](#) [▶](#)

[Back](#) [Close](#)

[Full Screen / Esc](#)

[Printer-friendly Version](#)

[Interactive Discussion](#)



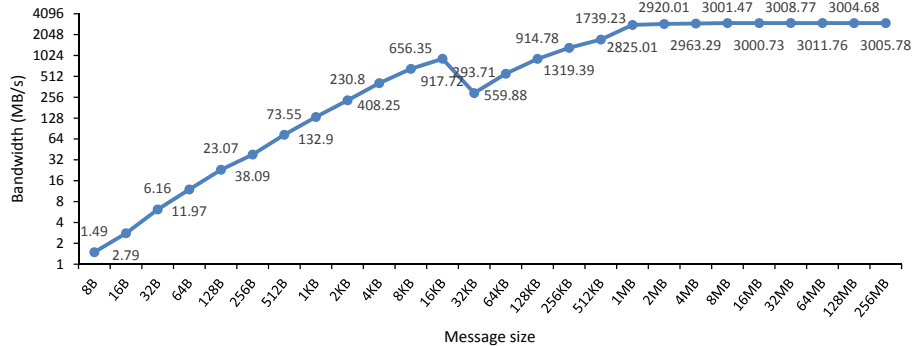


Figure 2. Variation of bandwidth (y axis) of an MPI P2P communication with the increment of message size. The two processes of the P2P communication run on two different computing nodes.

GMDD

8, 8981–9020, 2015

Improving data transfer for model coupling

C. Zhang et al.

Title Page	
Abstract	Introduction
Conclusions	References
Tables	Figures
⏪	⏩
◀	▶
Back	Close
Full Screen / Esc	
Printer-friendly Version	
Interactive Discussion	



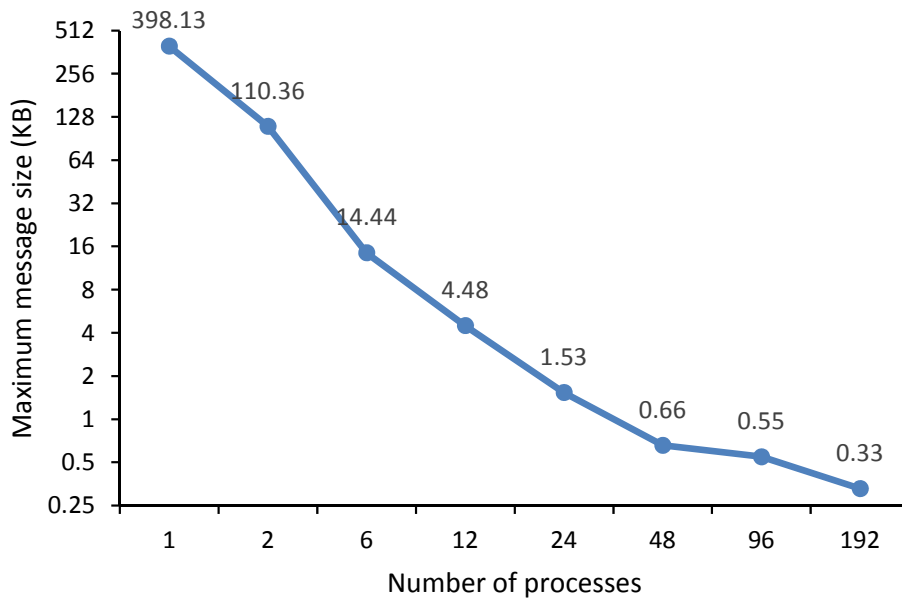


Figure 3. Variation of maximum message size of the P2P implementation in GAMIL2-CLM3 with the increment of process number. The experimental setup here is similar with that in Fig. 1.

Improving data transfer for model coupling

C. Zhang et al.

Title Page	
Abstract	Introduction
Conclusions	References
Tables	Figures
⏪	⏩
◀	▶
Back	Close
Full Screen / Esc	
Printer-friendly Version	
Interactive Discussion	



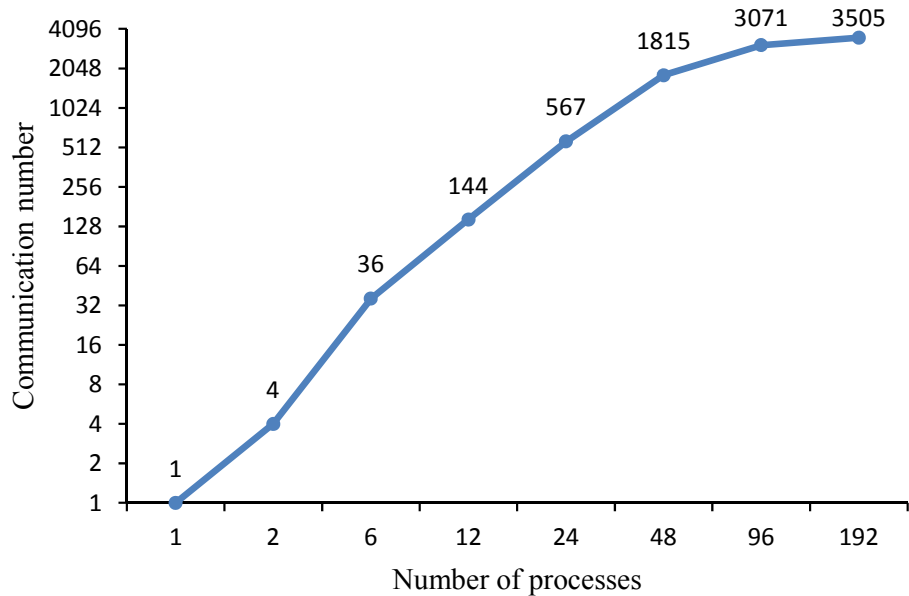


Figure 4. Variation of total number of MPI messages (y axis) of the P2P implementation in GAMIL2-CLM3 with the increment of process number (x axis). The experimental setup here is similar with that in Fig. 1.

Improving data transfer for model coupling

C. Zhang et al.

Title Page	
Abstract	Introduction
Conclusions	References
Tables	Figures
⏪	⏩
◀	▶
Back	Close
Full Screen / Esc	
Printer-friendly Version	
Interactive Discussion	



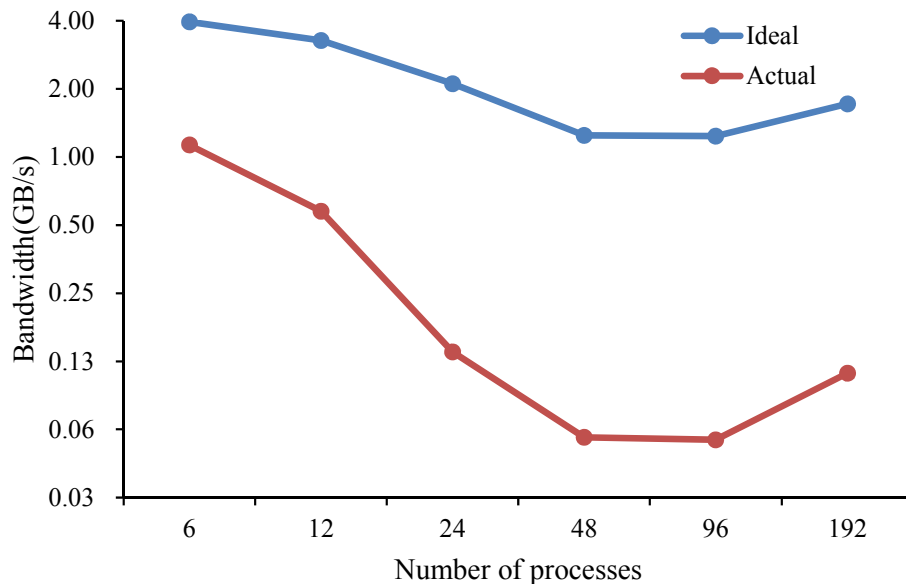


Figure 5. The ideal and actual bandwidths of the P2P implementation in GAMIL2-CLM3 when gradually increasing the number of processes for each component model. The experimental setup here is similar with that in Fig. 1. The ideal bandwidth is calculated from the message size and the MPI bandwidth measured in Fig. 2, and the actual bandwidth is calculated from Fig. 1.

Improving data transfer for model coupling

C. Zhang et al.

Title Page	
Abstract	Introduction
Conclusions	References
Tables	Figures
⏪	⏩
◀	▶
Back	Close
Full Screen / Esc	
Printer-friendly Version	
Interactive Discussion	



Improving data transfer for model coupling

C. Zhang et al.

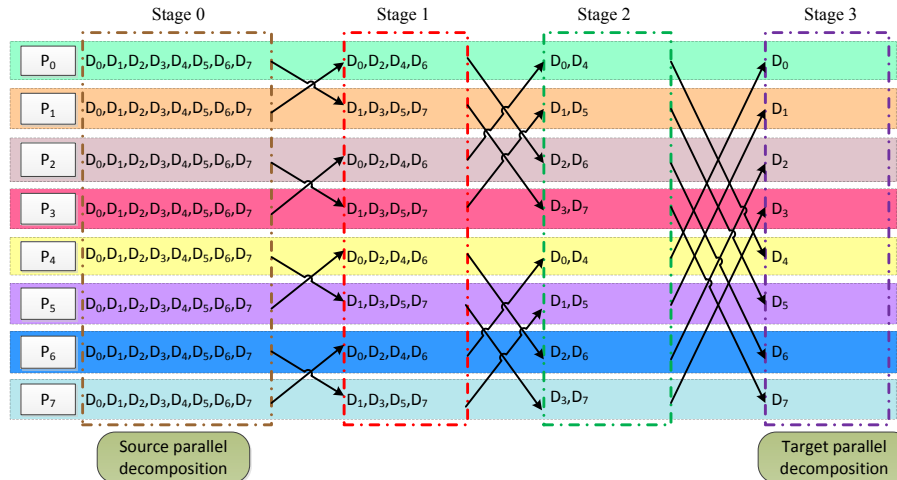


Figure 6. An example of the butterfly implementation with 8 processes. The butterfly implementation targets to rearrange the data from the source parallel decomposition to the target parallel decomposition. Each colored row stands for a process (P_0 – P_7). D_i represents the subset of data corresponding to process P_i , determined by the target parallel decomposition. There are multiple stages (each colored column represents a stage (Stage 0 to Stage 3)) in the butterfly implementation, and each stage has a unique parallel decomposition. Each arrow stands for an MPI P2P communication from one process to another.

Title Page

Abstract Introduction

Conclusions References

Tables Figures

◀ ▶

◀ ▶

Back Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



Improving data transfer for model coupling

C. Zhang et al.

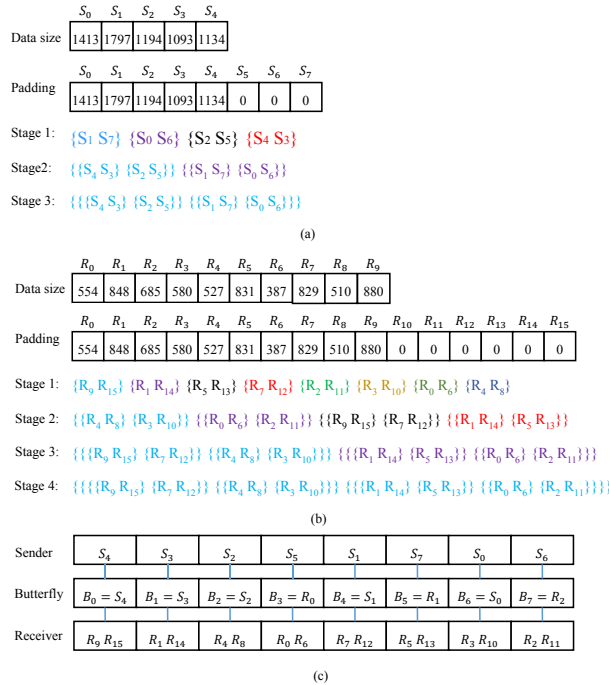


Figure 7. An example of process mapping, given that the sender has 5 processes (S_0 – S_4), the receiver has 10 processes (R_0 – R_9) (there is no common process between the sender and receiver), and the butterfly kernel contains 8 processes (B_0 – B_7). Panels (a) and (b) show how to iteratively pair processes of the sender and receiver, respectively. There are multiple stages in the iterative pairing of processes of the sender and receiver. In each stage, the processes in the same color are grouped into one pair. Panel (c) shows how to map the reordered processes of the sender and receiver to processes of the butterfly kernel. All the 5 processes of the sender are used for the butterfly kernel. Each process of the sender is mapped to a process of the butterfly kernel, while each two processes of the receiver are mapped to one process of the butterfly kernel.

Title Page	
Abstract	Introduction
Conclusions	References
Tables	Figures
⏪	⏩
◀	▶
Back	Close
Full Screen / Esc	
Printer-friendly Version	
Interactive Discussion	



Improving data transfer for model coupling

C. Zhang et al.

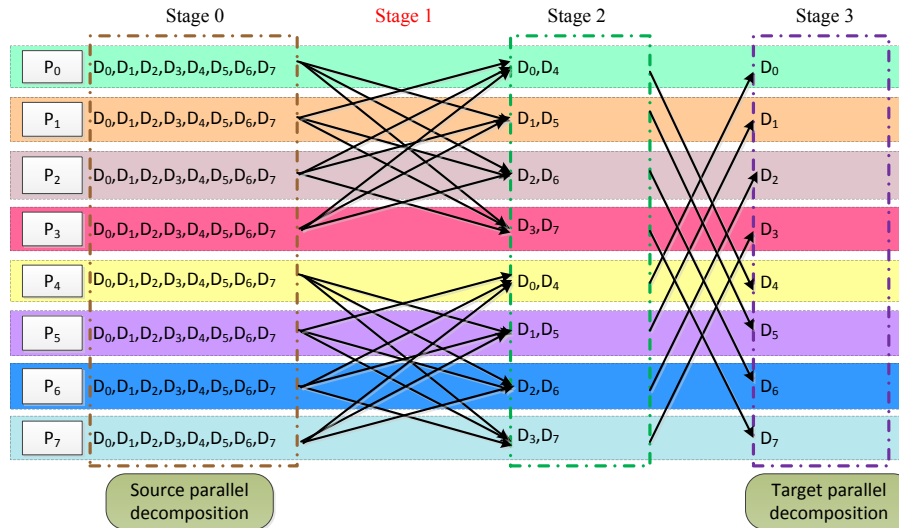


Figure 8. An example of the adaptive data transfer library given 8 processes, where Stage 1 of the butterfly implementation is skipped with the P2P implementation of 3 MPI messages per process.

Title Page

Abstract Introduction

Conclusions References

Tables Figures

◀ ▶

◀ ▶

Back Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures

⏪

⏩

◀

▶

Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



```

Input: Process number of the butterfly implementation Proc_num
Output: Stage mask array of the butterfly implementation Stage_mask
Program Profiling
Begin
  Set the stage number Stage_num to be  $\log_2 Proc\_num + 1$ 
  For  $i=0$  to Stage_num-1; then set Stage_mask[i] to be true
    Execute the butterfly instance with the stage mask array Stage_mask, and
    record the execution time as best_timer

    For  $i=0$  to Stage_num-1
      Do
        Set Stage_mask[i] to be false

        Execute the butterfly instance with the stage mask array Stage_mask,
        and record the execution time as cur_timer

        If best_timer is larger than cur_timer
          Set best_timer to be cur_timer

          Else set Stage_mask[i] to be true

        End do
      End do
    End do
  End

```

Figure 9. A flowchart for determining an appropriate implementation of the adaptive data transfer library.

Improving data transfer for model coupling

C. Zhang et al.

```

Program X_model
  use data_transfer_library_mod

  implicit none

  integer :: instance_id, decomp_id, end_step

  integer :: local_comm, remote_root_global_rank, direction

  integer :: num_grid_cells, num_local_cells, local_cells_global_index(:)

  logical :: mask_array(:), input

  real :: send_buf1(:,,:), send_buf2(:,,:)

  instance_id = data_transfer_register_instance(local_comm, remote_root_global_rank,
direction)

  call data_transfer_register_decomp(instance_id, num_grid_cells, num_local_cells,
local_cells_global_index)

  call data_transfer_register_field(instance_id, send_buf1, input)
  call data_transfer_register_field(instance_id, send_buf2, input)
  call data_transfer_register_mask(instance_id, mask_array)
  call data_transfer_init_instance(instance_id)

  do i=1, end_step
    call data_transfer_exec_instance(instance_id)
  end do

  call data_transfer_final_instance(instance_id)

End program X_model

```

Figure 10. An example of how to implement data transfer with the APIs of the adaptive data transfer library. The APIs of the adaptive data transfer library are marked in red.

[Title Page](#)
[Abstract](#)
[Introduction](#)
[Conclusions](#)
[References](#)
[Tables](#)
[Figures](#)
[⏪](#)
[⏩](#)
[◀](#)
[▶](#)
[Back](#)
[Close](#)
[Full Screen / Esc](#)
[Printer-friendly Version](#)
[Interactive Discussion](#)

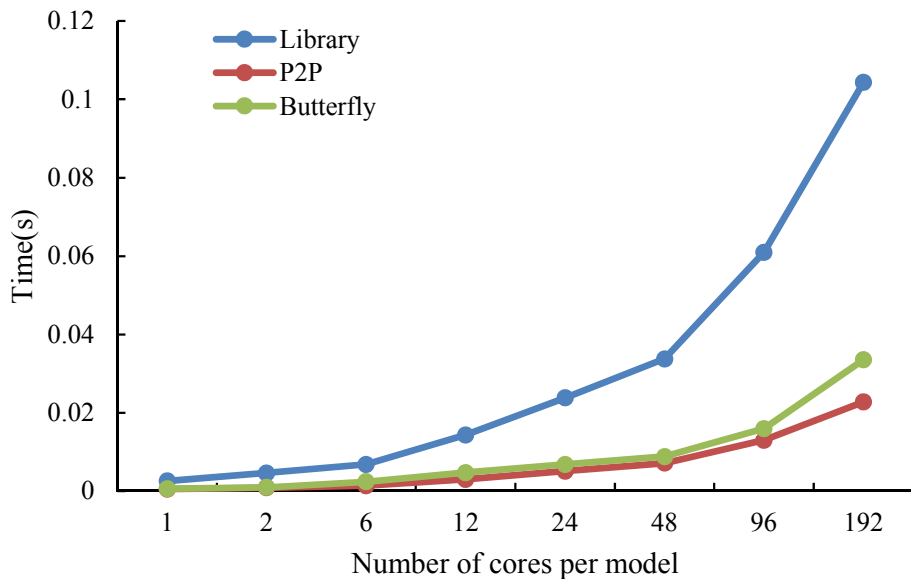



Figure 11. Initialization time (y axis) of one data transfer between two toy models using a rectangular grid (of 192×96 grid points) when varying the number of cores used by each toy model (x axis). There are 10 2-D coupling fields transferred from the source toy model to the target toy model. If the number of cores per toy model is less than 24, the MPI message number per process is set to be the number of cores. Otherwise, the MPI message number per process is set to 24.

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract Introduction

Conclusions References

Tables Figures

◀ ▶

◀ ▶

Back Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



Improving data transfer for model coupling

C. Zhang et al.

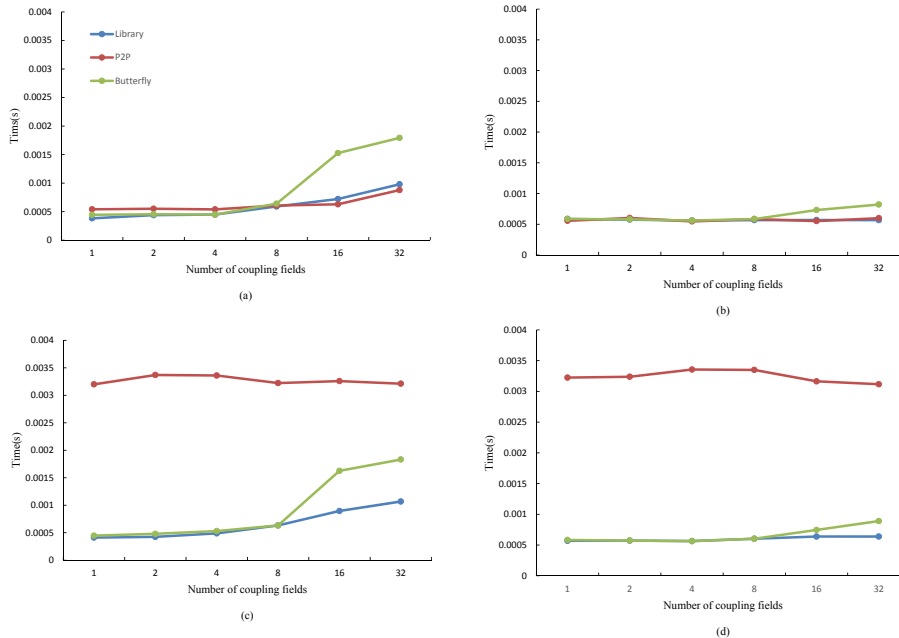


Figure 13. Average execution time (y axis) of one data transfer between two toy models with the same rectangular grid (of 192×96 grid points) when varying the number of coupling fields transferred (x axis). There are four simulation tests for the evaluation. In simulation **(a)**, each toy model is run with 48 cores and MPI message number per process is 12. In simulation **(b)**, each toy model is run with 192 cores and MPI message number per process is 12. In simulation **(c)**, each toy model is run with 48 cores and MPI message number per process is 48. In simulation **(d)**, each toy model is run with 192 cores (or processes) and the MPI message number per process is 48.

[Title Page](#)

[Abstract](#)

[Introduction](#)

[Conclusions](#)

[References](#)

[Tables](#)

[Figures](#)

⏪

⏩

◀

▶

[Back](#)

[Close](#)

[Full Screen / Esc](#)

[Printer-friendly Version](#)

[Interactive Discussion](#)



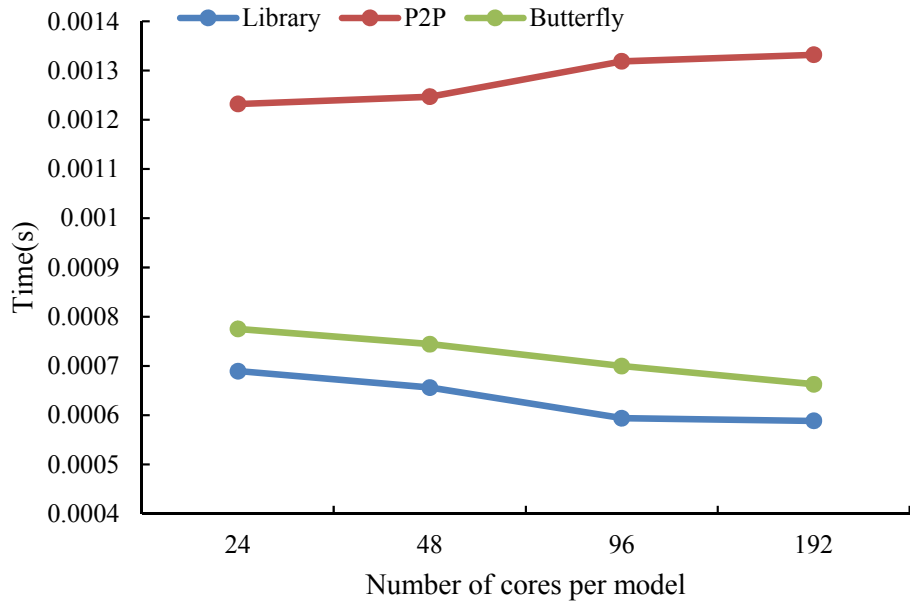


Figure 14. Average execution time (y axis) of one data transfer between two toy models with the same rectangular grid (of 192×96 grid points) when varying the number of cores used by each toy model (x axis). There are 10 2-D coupling fields transferred from the source toy model to the target toy model. In each test, the MPI message number per process is set to 24.

Improving data transfer for model coupling

C. Zhang et al.

[Title Page](#)

[Abstract](#) [Introduction](#)

[Conclusions](#) [References](#)

[Tables](#) [Figures](#)

[◀](#) [▶](#)

[◀](#) [▶](#)

[Back](#) [Close](#)

[Full Screen / Esc](#)

[Printer-friendly Version](#)

[Interactive Discussion](#)



Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract

Introduction

Conclusions

References

Tables

Figures

◀

▶

◀

▶

Back

Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion

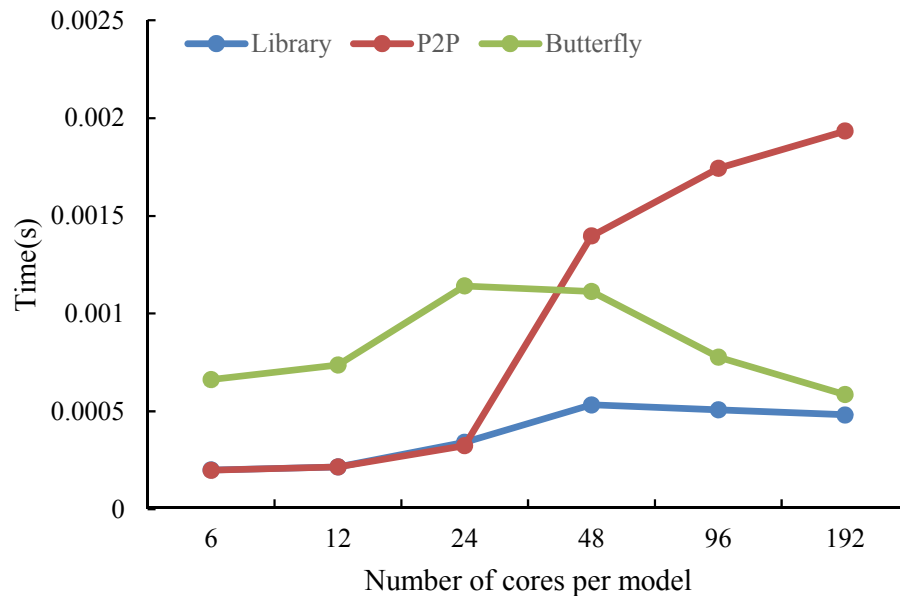


Figure 15. Average execution time (y axis) of one data transfer between the land surface model CLM4 and the coupler CPL7 in CESM when varying the number of cores used by each model (x axis): 32 coupling fields on the CLM horizontal grid (the grid size is $144 \times 96 = 13\,824$) are transferred from the land surface model CLM4 to the coupler CPL7.

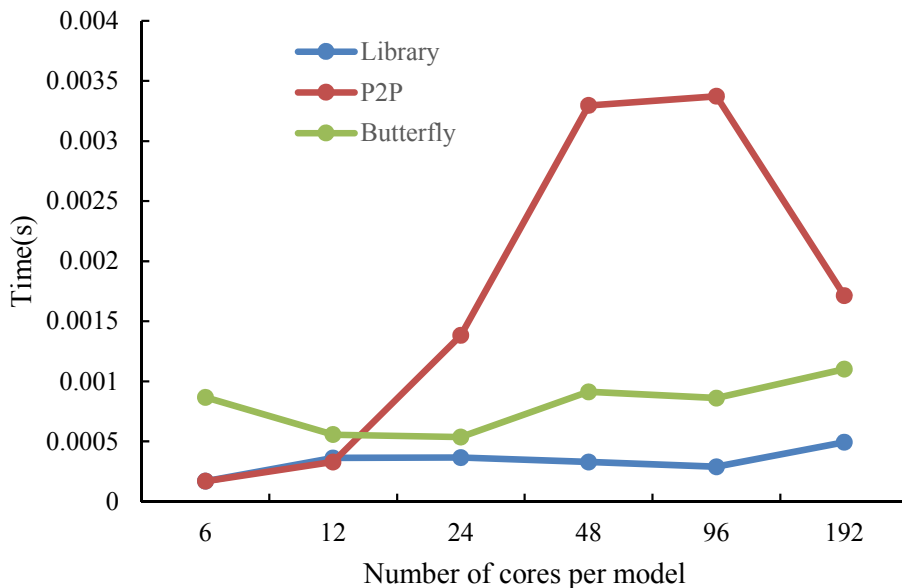


Figure 16. Average execution time (*y* axis) of one data transfer between the atmosphere model GAMIL2 and the land surface model CLM3 in GAMIL2-CLM3 when varying the number of cores used by each model (*x* axis): 14 coupling fields on the GAMIL2 horizontal grid (the grid size is $128 \times 60 = 7680$) are transferred from the land surface model CLM3 to the atmosphere model GAMIL2.

Improving data transfer for model coupling

C. Zhang et al.

Title Page

Abstract Introduction

Conclusions References

Tables Figures

◀ ▶

◀ ▶

Back Close

Full Screen / Esc

Printer-friendly Version

Interactive Discussion



