1. Introduction

A coupled model regarding Earth system Modelling generally highly depends on existing couplers (引用), each of which can combine different component models into a whole system and handle data interpolation between different model grids and data transfer between different component models (引用文献). In response to more and more computation resulting from higher and higher resolutions in model development, the parallel efficiency of a coupled model on modern high-performance computers becomes more and more important. Any module in a coupled model, including the coupler, can impact or even may damage the parallel efficiency of the whole coupled model. Although most existing couplers achieve scalable data transfer and data interpolation, i.e., the data transfer and data interpolation generally can be faster when using more processor cores, there is almost no evidence of scalable initialization of a coupler. Experiences from OASIS3-MCT and C-Coupler2 have revealed that the initialization cost of a coupler increases fast with more processor cores (引用文献). To achieve scalable initialization of couplers, this paper tries to make a first step through focusing on the initialization of data transfer.

The functionality of data transfer of couplers is transferring scalar variables or fields on a model grid (called gridded fields hereafter) from one component model to another via MPI (Message Passing Interface). A component model generally has been parallelized through decomposing the cells of a model grid into distinct subsets each of is assigned to an MPI process for cooperative concurrent computation, e.g., the sample parallel decompositions in Fig. 1a and 1b. To efficiently transfer gridded fields in parallel, Jacob et al. (2005) proposed an approach of *M*x*N* communication (called *M*x*N* approach) following the routing network where a pair of processes each from the two component models should have a communication connection only when they have common grid cells (for example, Fig. 1c). This *M*x*N* approach has already been used in existing couplers for more than ten years. As the parallel decompositions of component models generally keep constant throughout the integration, a routing network can also keep constant. Thus, the *M*x*N* approach can be achieved with two major steps: generating the routing network when initializing the coupler, and transferring gridded fields based on the routing network throughout the coupled model integration. In spite of the scalability of the second major step, the first major step in existing couplers is non-scalable, introducing higher cost when using more processor cores.

In this paper, we propose a scalable approach for generating routing network, which is much faster and consumes much less memory than the existing approach. The remainder of this paper is organized as follows. We reveal the causes of non-scalability of the existing approach in Section 2, present and then evaluate new scalable approach Section 3 and 4 respectively, and discuss and conclude this work in Section 5.

.

Each component model of the earth system model is a grid-based numerical program, that is, the area to be calculated (two-dimensional surface or three-dimensional space) is divided into a calculation grid composed of several non-overlapping sub-grids (atomic regions), and perform collaborative calculations on all grid points. For a calculation area, the more grid points in the sub-grid, the higher the simulation resolution is, and the larger the calculation amount of the simulation is. The increasing number of high-performance computers with computing nodes and processor cores has brought opportunities for the accelerated operation of various calculation applications, including numerical programs. To achieve better acceleration on high-performance computers, the serial numerical programs need to be rewritten into efficient parallel versions. MPI (Message Passing Interface) is a widely used parallel programming library that can implement parallel computing between different computing nodes and between different processor cores within the same node. Most of the earth system component models have parallel versions based on MPI.

When developing a parallel version of a numerical simulation program, researchers first need to assign grid points of the computing grid to different processes (hereafter, the assignment of grid points on different processes is called parallel decomposition) to perform parallel computing. For example, Fig. 1 shows two parallel decompositions of the same 8x8 grid when using 4 processes and 8 processes respectively, each number represents the process number where the grid is located.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |   Fig 1(a). A parallel decomposition for an 8x8 grid when using 4 processors. |  | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |   Fig 1(b). A parallel decomposition for an 8x8 grid when using 8 processors. |

Due to the mutual coupling between different component models in the earth system model, the coupling variable will be transferred from one component model to another, the essence of which is the MPI communication for the coupling variable. This process is usually implemented in two steps: 1) according to the parallel decomposition of the same grid in two component models, a parallel communication routing relationship between component models is established (for example, process 1 in Fig. 1 (a) needs to transfer the coupling variable data in the upper right corner of the grid to process 7 in Fig. 1 (b)); 2) During the coupling process, according to the established communication routing relationship, finish parallel communication for several variables between component models. Because the parallel communication routine can remain unchanged during the frequent coupling procedure between component models, the parallel communication routine between two parallel decompositions usually only needs to be established once.

This paper presents a new method for quickly establishing a parallel communication routine. It does not introduce global communication and has excellent parallel scalability, and the average computational complexity of each process is O ((N / M) \* (log (M) + log (N / M))), the average storage complexity is O (N / M), and the average communication complexity is O ((N / M) \* log (M)).

The rest of this paper is organized as follows. We briefly introduce related work in section 2, introduce the overall design and describe the implementation of our algorithm in Section 3, evaluate it in Section 4, briefly summarize this paper and discuss the future work in Section 5.

2. Related work

At present, couplers responsible for coupling between earth system models such as C-Coupler, OASIS, CPL, etc., have adopted the above two steps to establish parallel communication routine between component models, among which global communications are used to complete the establishment of the routing relationship, that is, a process P in one component model first obtains the global parallel decomposition information of all processes R\_ALL in another component model, and then finds the intersection of the local parallel decomposition of P and the global decomposition R\_ALL to establish the parallel communication routing relationship. The total number of grid points of a given grid is N, and the number of processes of a component model is M. The average computational complexity of the global algorithm on each process is O (N), the average storage complexity is O (N), and this algorithm will introduce global communication with complexity even O (NM). Therefore, global communication algorithm is time-consuming and does not have parallel scalability. Especially when the model grid has a high resolution as well as a large number of grid points, the algorithm will seriously cause the coupled model to start slowly.

3. Overall design

In this part, we introduce our method for quickly establishing a parallel communication relationship from the sender decomposition(the source component model or the source internal process of a component model of the coupling variable) to the receiver decomposition (the destination component model or the destination internal process of a component model of the coupling variable). Because the global indexes processed by previous global communication algorithms are in unordered state, they had to use global communication to obtain the global grid data, resulting in high communication complexity. By sorting the global indexes according to the idea of merge sorting, our algorithm makes the global indexes orderly and avoids global communication. In the communication relationship establishment procedure, each process of the sender and the receiver sequentially executes each subroutine:

1) Initialize module;

2) Parallel sorting module based on the global index of grid points;

3) Parallel mapping relationship establishment module;

4) Parallel sorting module based on process number;

5) Parallel communication routing relationship generation module.

3.1 Initialization module

This module first establishes the same grid point indexes for the sender and receiver processes, different grid point has different global index, while the same grid has the same global index both in sender and receiver processes, then establishes the relationship between the global index and local index of grid points for the sender and receiver processes. Based on the grid point global index, each process builds a grid point local-global mapping table, where each item in the table is a triplet < global index, process id, local index>. Fig. 2 and Fig. 3 respectively show the local-global mapping tables of the grid points in Fig. 1 (a) and Fig. 1 (b).

|  |  |
| --- | --- |
| process id | Grid Point Local-Global Mapping Table |
| P0 | <0,0,0>, <1,0,1>, <2,0,2>, <3,0,3>, <8,0,4>, <9,0,5>, <10,0,6>, <11,0,7>, <16,0,8>, <17,0,9>, <18,0,10>, <19,0,11>, <24,0,12>, <25,0,13>, <26,0,14>, <27,0,15> |
| P1 | <4,1,0>, <5,1,1>, <6,1,2>, <7,1,3>, <12,1,4>, <13,1,5>, <14,1,6>, <15,1,7>, <20,1,8>, <21,1,9>, <22,1,10>, <23,1,11>, <28,1,12>, <29,1,13>, <30,1,14>, <31,1,15> |
| P2 | <32,2,0>, <33,2,1>, <34,2,2>, <35,2,3>, <40,2,4>, <41,2,5>, <42,2,6>, <43,2,7>, <48,2,8>, <49,2,9>, <50,2,10>, <51,2,11>, <56,2,12>, <57,2,13>, <58,2,14>, <59,2,15> |
| P3 | <36,3,0>, <37,3,1>, <38,3,2>, <39,3,3>, <44,3,4>, <45,3,5>, <46,3,6>, <47,3,7>,  <52,3,8>, <53,3,9>, <54,3,10>, <55,3,11>, <60,3,12>, <61,3,13>, <62,3,14>, <63,3,15> |

Fig. 2 After the initialization module, the grid point local-global mapping table is established in parallel based on parallel decomposition in Fig. 1 (a), where the global index of the grid points is 0 ~ 63, and the index is increased one by one from left to right and top to bottom according to the grid point position.

|  |  |
| --- | --- |
| process id | Grid Point Local-Global Mapping Table |
| P0 | <0,0,0>, <8,0,1>, <16,0,2>, <24,0,3>, <32,0,4>, <40,0,5>, <48,0,6>, <56,0,7> |
| P1 | <1,1,0>, <9,1,1>, <17,1,2>, <25,1,3>, <33,1,4>, <41,1,5>, <49,1,6>, <57,1,7> |
| P2 | <2,2,0>, <10,2,1>, <18,2,2>, <26,2,3>, <34,2,4>, <42,2,5>, <50,2,6>, <58,2,7> |
| P3 | <3,3,0>, <11,3,1>, <19,3,2>, <27,3,3>, <35,3,4>, <43,3,5>, <51,3,6>, <59,3,7> |
| P4 | <4,4,0>, <12,4,1>, <20,4,2>, <28,4,3>, <36,4,4>, <44,4,5>, <52,4,6>, <60,4,7> |
| P5 | <5,5,0>, <13,5,1>, <21,5,2>, <29,5,3>, <37,5,4>, <45,5,5>, <53,5,6>, <61,5,7> |
| P6 | <6,6,0>, <14,6,1>, <22,6,2>, <30,6,3>, <38,6,4>, <46,6,5>, <54,6,6>, <62,6,7> |
| P7 | <7,7,0>, <15,7,1>, <23,7,2>, <31,7,3>, <39,7,4>, <47,7,5>, <55,7,6>, <63,7,7> |

Fig. 3 After the initialization module, the grid point local-global mapping table is established in parallel based on parallel decomposition in Fig. 1 (b), where the global index of the grid points is 0 ~ 63, and the index is increased one by one from left to right and top to bottom according to the grid point position.

The characteristics of our algorithm limit the number of processes participating in communication must be a power of two, which can effectively improve the performance of communication of our algorithm. So, if the processes number is not a power of two, the data distribution needs to be adjusted. Assuming the number of processes is 10, the extra processes (processes which process index greater than 7) need to send their own grid point data to the corresponding processes in processes 0 to 7(process 8 and process 9), as shown in Fig. 4, these extra processes do not participate in subsequent communications.

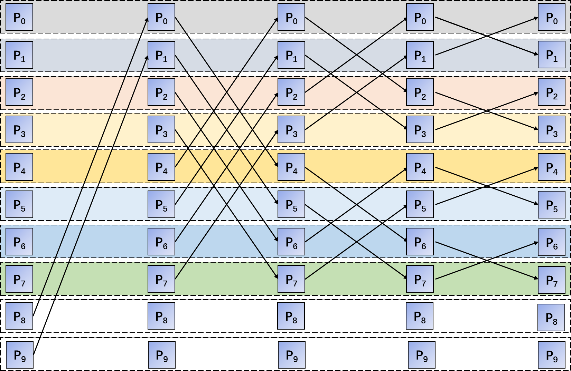


Fig. 4 An example of our algorithm with ten processes. Each row stands for one process (P0–P9). There are multiple stages throughout the communication process. Process 8 and Process 9 need to send their grid point data to Process 0 and Process 1 correspondingly, these two processes will not participate in subsequent communications.

3.2 Parallel sorting module based on the global index of grid points

This module is responsible for the parallel sorting of the grid point local-global index entries in the sender (or the receiver) processes based on the grid point global indexes among all processes. This module first determines the expected global index range of the grid points of each process after sorting according to the total number of grid points of the grid and the number of processes of the sender(or the receiver) processes, as well as ensures that the number of grid points in each process is roughly equal after sorting, that is, ensure load balancing of calculation. After that, sorts the grid point local-global index entries within each process; then sorts the grid point local-global index entries among all processes according to the idea of ​​merge sort based on global index. Recursively divide the entire processes into two groups, communicate between corresponding process pairs in two groups, and exchange data using MPI communication functions, then filter and merge the received entries with its original entries according to the grid point range of the process calculated before, as shown in Fig. 5.

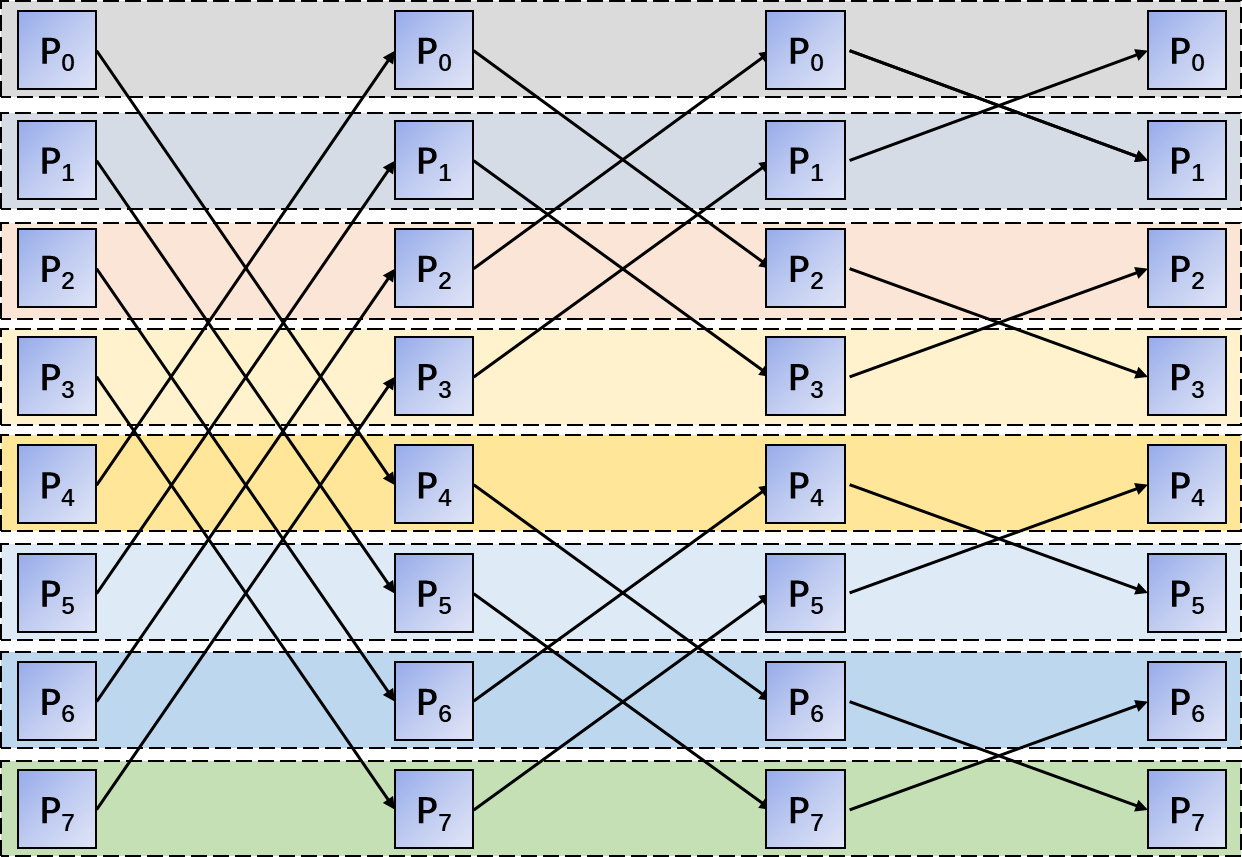


Fig. 5 An example of our algorithm with eight processes. Each row stands for one process (P0–P7). There are multiple stages throughout the communication process. Each process needs to communicate a total of log2(M) times.

The number of grid points in a given grid is N, and the number of processes at the sender (or receiver) is M. One process involves log (M) one-to-one communication with other processes during the sorting process. In this module, the average time complexity of sorting each process is O ((N / M) \* (log (M) + log (N / M))), the average storage complexity is O (N / M), and the average communication The complexity is O ((N / M) \* log (M)). Fig. 6 and Fig. 7 are the results of the grid point local-global mapping tables in Fig. 2 and Fig. 3 after parallel sorting, respectively.

|  |  |
| --- | --- |
| process id | Grid Point Local-Global Mapping Table |
| P0 | <0,0,0>, <1,0,1>, <2,0,2>, <3,0,3>, <4,1,0>, <5,1,1>, <6,1,2>, <7,1,3>,  <8,0,4>, <9,0,5>, <10,0,6>, <11,0,7>, <12,1,4>, <13,1,5>, <14,1,6>, <15,1,7>, |
| P1 | <16,0,8>, <17,0,9>, <18,0,10>, <19,0,11>, <20,1,8>, <21,1,9>, <22,1,10>, <23,1,11>, <24,0,12>, <25,0,13>, <26,0,14>, <27,0,15>, <28,1,12>, <29,1,13>, <30,1,14>, <31,1,15> |
| P2 | <32,2,0>, <33,2,1>, <34,2,2>, <35,2,3>, <36,3,0>, <37,3,1>, <38,3,2>, <39,3,3>, <40,2,4>, <41,2,5>, <42,2,6>, <43,2,7>, <44,3,4>, <45,3,5>, <46,3,6>, <47,3,7> |
| P3 | <48,2,8>, <49,2,9>, <50,2,10>, <51,2,11>, <52,3,8>, <53,3,9>, <54,3,10>, <55,3,11>, <56,2,12>, <57,2,13>, <58,2,14>, <59,2,15>, <60,3,12>, <61,3,13>, <62,3,14>, <63,3,15> |

Fig. 6 The grid point local-global mapping table in Fig 2 is sorted based on the grid point global index.

|  |  |
| --- | --- |
| process id | Grid Point Local-Global Mapping Table |
| P0 | <0,0,0>, <1,1,0>, <2,2,0>, <3,3,0>, <4,4,0>, <5,5,0>, <6,6,0>, <7,7,0> |
| P1 | <8,0,1>, <9,1,1>, <10,2,1>, <11,3,1>, <12,4,1>, <13,5,1>, <14,6,1>, <15,7,1> |
| P2 | <16,0,2>, <17,1,2>, <18,2,2>, <19,3,2>, <20,4,2>, <21,5,2>, <22,6,2>, <23,7,2> |
| P3 | <24,0,3>, <25,1,3>, <26,2,3>, <27,3,3>, <28,4,3>, <29,5,3>, <30,6,3>, <31,7,3> |
| P4 | <32,0,4>, <33,1,4>, <34,2,4>, <35,3,4>, <36,4,4>, <37,5,4>, <38,6,4>, <39,7,4> |
| P5 | <40,0,5>, <41,1,5>, <42,2,5>, <43,3,5>, <44,4,5>, <45,5,5>, <46,6,5>, <47,7,5> |
| P6 | <48,0,6>, <49,1,6>, <50,2,6>, <51,3,6>, <52,4,6>, <53,5,6>, <54,6,6>, <55,7,6> |
| P7 | <56,0,7>, <57,1,7>, <58,2,7>, <59,3,7>, <60,4,7>, <61,5,7>, <62,6,7>, <63,7,7> |

Fig. 7 The grid point local-global mapping table in Fig 3 is sorted based on the grid point global index.

3.3 Parallel mapping relationship establishment module

The grid point mapping relationship means that the relationship between two processes which a process on the sender and a process on the receiver have the same grid point. This module is responsible for establishing the sender-receiver grid point mapping relationship table based on grid point global index in parallel, where the entries in the table are quintuple <grid point global index, sender process id, sender process grid point local index, receiver process id, receiver process grid point local index>. This module first completes the exchange of the grid point local-global mapping table of each process on the sender side and the corresponding process on the receiver side through MPI point-to-point communication based on the global index range of the grid points of the sender and receiver processes. Because number of processes of both sender and receiver are the power of 2, so we can build a regular communication relationship, as shown in Fig. 10. Within each process, a sender-receiver grid point mapping relationship table is established based on the grid point local-global mapping tables of the sender and receiver. The average time complexity, the average storage space complexity, and the average communication complexity of each model's processes are O (N / M). Fig. 8 and 9 are the sender-receiver grid point mapping tables of the sender (Fig. 1 (a)) and the receiver (Fig. 1 (b)).

|  |  |
| --- | --- |
| process id | Grid Point Local-Global Mapping Table |
| P0 | <0,0,0,0,0>, <1,0,1,1,0>, <2,0,2,2,0>, <3,0,3,3,0>, <4,1,0,4,0>, <5,1,1,5,0>, <6,1,2,6,0>, <7,1,3,7,0>, <8,0,4,0,1>, <9,0,5,1,1>, <10,0,6,2,1>, <11,0,7,3,1>, <12,1,4,4,1>, <13,1,5,5,1>, <14,1,6,6,1>, <15,1,7,7,1>, |
| P1 | <16,0,8,0,2>, <17,0,9,1,2>, <18,0,10,2,2>, <19,0,11,3,2>, <20,1,8,4,2>, <21,1,9,5,2>, <22,1,10,6,2>, <23,1,11,7,2>, <24,0,12,0,3>, <25,0,13,1,3>, <26,0,14,2,3>, <27,0,15,3,3>, <28,1,12,4,3>, <29,1,13,5,3>, <30,1,14,6,3>, <31,1,15,7,3> |
| P2 | <32,2,0,0,4>, <33,2,1,1,4>, <34,2,2,2,4>, <35,2,3,3,4>, <36,3,0,4,4>, <37,3,1,5,4>, <38,3,2,6,4>, <39,3,3,7,4>, <40,2,4,0,5>, <41,2,5,1,5>, <42,2,6,2,5>, <43,2,7,3,5>, <44,3,4,4,5>, <45,3,5,5,5>, <46,3,6,6,5>, <47,3,7,7,5> |
| P3 | <48,2,8,0,6>, <49,2,9,1,6>, <50,2,10,2,6>, <51,2,11,3,6>, <52,3,8,4,6>, <53,3,9,5,6>, <54,3,10,6,6>, <55,3,11,7,6>, <56,2,12,0,7>, <57,2,13,1,7>, <58,2,14,2,7>, <59,2,15,3,7>, <60,3,12,4,7>, <61,3,13,5,7>, <62,3,14,6,7>, <63,3,15,7,7> |

Fig. 8 sender (Fig. 1(a)) grid point local-global mapping table

|  |  |
| --- | --- |
| process id | Grid Point Local-Global Mapping Table |
| P0 | <0,0,0,0,0>, <1,0,1,1,0>, <2,0,2,2,0>, <3,0,3,3,0>, <4,1,0,4,0>, <5,1,1,5,0>, <6,1,2,6,0>, <7,1,3,7,0> |
| P1 | <8,0,4,0,1>, <9,0,5,1,1>, <10,0,6,2,1>, <11,0,7,3,1>, <12,1,4,4,1>, <13,1,5,5,1>, <14,1,6,6,1>, <15,1,7,7,1> |
| P2 | <16,0,8,0,2>, <17,0,9,1,2>, <18,0,10,2,2>, <19,0,11,3,2>, <20,1,8,4,2>, <21,1,9,5,2>, <22,1,10,6,2>, <23,1,11,7,2> |
| P3 | <24,0,12,0,3>, <25,0,13,1,3>, <26,0,14,2,3>, <27,0,15,3,3>, <28,1,12,4,3>, <29,1,13,5,3>, <30,1,14,6,3>, <31,1,15,7,3> |
| P4 | <32,2,0,0,4>, <33,2,1,1,4>, <34,2,2,2,4>, <35,2,3,3,4>, <36,3,0,4,4>, <37,3,1,5,4>, <38,3,2,6,4>, <39,3,3,7,4> |
| P5 | <40,2,4,0,5>, <41,2,5,1,5>, <42,2,6,2,5>, <43,2,7,3,5>, <44,3,4,4,5>, <45,3,5,5,5>, <46,3,6,6,5>, <47,3,7,7,5> |
| P6 | <48,2,8,0,6>, <49,2,9,1,6>, <50,2,10,2,6>, <51,2,11,3,6>, <52,3,8,4,6>, <53,3,9,5,6>, <54,3,10,6,6>, <55,3,11,7,6> |
| P7 | <56,2,12,0,7>, <57,2,13,1,7>, <58,2,14,2,7>, <59,2,15,3,7>, <60,3,12,4,7>, <61,3,13,5,7>, <62,3,14,6,7>, <63,3,15,7,7> |

Fig. 9 receiver (Fig. 1(b)) grid point local-global mapping table

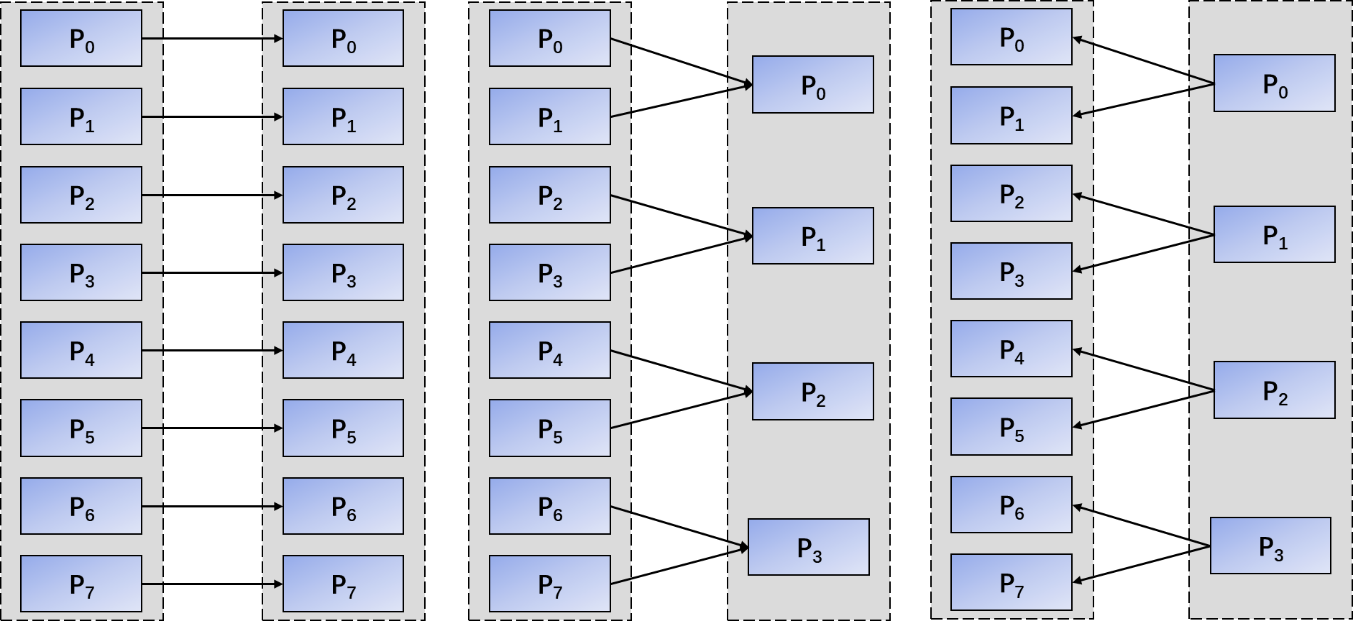


Fig. 10 The communication relationship when the sender has 8 processes and the receiver has 8 processes, the sender has 8 processes and the receiver has 4 processes, the sender has 4 processes and the receiver has 8 processes.

3.4 Parallel sorting module based on process number

According to the idea of merge sort, this module completes the sorting of the sender-receiver grid point mapping relationship table between all processes at the sender (or receiver) according to the sender process number (or receiver process number), so that the sender-receiver grid point mapping relationship table of each process at the sender (or receiver) contains only the entries related to the local grid points of the exact process. In this module, the average time complexity of sorting each process is O ((N / M) \* log (M)), the average storage complexity is O (N / M), and the average communication complexity is O ((N / M ) \* log (M)). Fig. 11 and 12 are the sender-receiver grid point mapping relationship tables after the sender (Fig. 1 (a)) and receiver (Fig. 1 (b)) are sorted in parallel by process numbers, respectively.

|  |  |
| --- | --- |
| process id | Grid Point Local-Global Mapping Table |
| P0 | <0,0,0,0,0>, <1,0,1,1,0>, <2,0,2,2,0>, <3,0,3,3,0>,<8,0,4,0,1>, <9,0,5,1,1>, <10,0,6,2,1>, <11,0,7,3,1>, <16,0,8,0,2>, <17,0,9,1,2>, <18,0,10,2,2>, <19,0,11,3,2>, <24,0,12,0,3>, <25,0,13,1,3>, <26,0,14,2,3>, <27,0,15,3,3> |
| P1 | <4,1,0,4,0>, <5,1,1,5,0>, <6,1,2,6,0>, <7,1,3,7,0>, <12,1,4,4,1>, <13,1,5,5,1>, <14,1,6,6,1>, <15,1,7,7,1>, <20,1,8,4,2>, <21,1,9,5,2>, <22,1,10,6,2>, <23,1,11,7,2>, <28,1,12,4,3>, <29,1,13,5,3>, <30,1,14,6,3>, <31,1,15,7,3> |
| P2 | <32,2,0,0,4>, <33,2,1,1,4>, <34,2,2,2,4>, <35,2,3,3,4>,<40,2,4,0,5>, <41,2,5,1,5>, <42,2,6,2,5>, <43,2,7,3,5>, <48,2,8,0,6>, <49,2,9,1,6>, <50,2,10,2,6>, <51,2,11,3,6>, <56,2,12,0,7>, <57,2,13,1,7>, <58,2,14,2,7>, <59,2,15,3,7> |
| P3 | <36,3,0,4,4>, <37,3,1,5,4>, <38,3,2,6,4>, <39,3,3,7,4>, <44,3,4,4,5>, <45,3,5,5,5>, <46,3,6,6,5>, <47,3,7,7,5>, <52,3,8,4,6>, <53,3,9,5,6>, <54,3,10,6,6>, <55,3,11,7,6>, <60,3,12,4,7>, <61,3,13,5,7>, <62,3,14,6,7>, <63,3,15,7,7> |

Fig. 11 After parallel merge sorting based on process numbers, the sender-receiver grid point mapping relationship table of the sender (Fig. 1 (a))

|  |  |
| --- | --- |
| process id | Grid Point Local-Global Mapping Table |
| P0 | <0,0,0,0,0>, <8,0,4,0,1>, <16,0,8,0,2>, <24,0,12,0,3>, <32,2,0,0,4>, <40,2,4,0,5>, <48,2,8,0,6>, <56,2,12,0,7> |
| P1 | <1,0,1,1,0>, <9,0,5,1,1>, <17,0,9,1,2>, <25,0,13,1,3>, <33,2,1,1,4>, <41,2,5,1,5>, <49,2,9,1,6>, <57,2,13,1,7> |
| P2 | <2,0,2,2,0>, <10,0,6,2,1>, <18,0,10,2,2>, <26,0,14,2,3>, <34,2,2,2,4>, <42,2,6,2,5>, <50,2,10,2,6>, <58,2,14,2,7> |
| P3 | <3,0,3,3,0>, <11,0,7,3,1>, <19,0,11,3,2>, <27,0,15,3,3>, <35,2,3,3,4>, <43,2,7,3,5>, <51,2,11,3,6>, <59,2,15,3,7> |
| P4 | <4,1,0,4,0>, <12,1,4,4,1>, <20,1,8,4,2>, <28,1,12,4,3>, <36,3,0,4,4>, <44,3,4,4,5>, <52,3,8,4,6>, <60,3,12,4,7> |
| P5 | <5,1,1,5,0>, <13,1,5,5,1>, <21,1,9,5,2>, <29,1,13,5,3>, <37,3,1,5,4>, <45,3,5,5,5>, <53,3,9,5,6>, <61,3,13,5,7> |
| P6 | <6,1,2,6,0>, <14,1,6,6,1>, <22,1,10,6,2>, <30,1,14,6,3>, <38,3,2,6,4>, <46,3,6,6,5>, <54,3,10,6,6>, <62,3,14,6,7> |
| P7 | <7,1,3,7,0>, <15,1,7,7,1>, <23,1,11,7,2>, <31,1,15,7,3>, <39,3,3,7,4>, <47,3,7,7,5>, <55,3,11,7,6>, <63,3,15,7,7> |

Fig. 12 After parallel merge sorting based on process numbers, the sender-receiver grid point mapping relationship table of the sender (Fig. 1 (b))

3.5 Parallel communication routing relationship generation module

This module generates the communication routing relationship between the process and the corresponding receiver (or sender) process according to the sender-receiver grid point mapping relationship table in the process of the sender (or receiver). For example, a communication routing relationship between process 0 on the sending side (Fig. 1 (a)) and processes 0 to 3 on the receiving side (Fig. 1 (b)) will be established, where process 0 on the sending side will send variable data on grid points whose local indexes are 1, 5, 9, 13 to process 1 on the receiving side; also will establish communication routing relationship between process 0 of the receiving side (Fig. 1 (b)) and processes 0 and 2 of the sending side (Fig. 1 (a)), where the process 0 will receive the variable data on the grid points whose local index is 4 ~ 7 from the process 2 of the sending side. In this module, the average time complexity of each process is O (N / M), the average storage complexity is O (N / M), and there is no communication.

|  |  |
| --- | --- |
| **Algorithm** 1. Generating the parallel decompositions of the sender and the receiver according to an  average number of MPI messages of the sender in the P2P implementation. | |
| Input | Number of processes of the sender: *M*  Number of processes of the receiver: *N*  Number of points in the grid: *Grid\_points* |
| Output | Grid point local-global mapping table between sender and receiver of each process. |
| 1. Determine the parallel information of model  Determine the parallel decomposition information of each process. If the processes number of component model is not a power of 2, reset it to a power of 2.  Sort local routing mapping table based on grid point local index.    2. Parallel sorting based on the global index of grid points intra component model  For each process, find the corresponding process in each communication stage within a component model;  Exchange routing mapping tables between processes;  Merge routing mapping tables of two processes based on the range of grid index calculated by process index.  3. Parallel mapping relationship establishment  For each process, find the corresponding process in another component model;  Exchange routing mapping tables between processes between two component models;  Merge routing mapping tables of two processes.  4. Parallel sorting based on the process index of grid points intra component model  For each process, find the corresponding process in each communication stage within a component model;  Exchange routing mapping tables between processes;  Merge routing mapping tables of two processes based on the process index.  Sort the routing mapping tables.  5. Generate communication routing relationship | |

Jacob, R., J. Larson, and E. Ong, 2005: M x 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.