An Automatic Performance Model-based Scheduling Tool for Coupled Climate System Models

Abstract

The prediction of the climate system is highly depended on the efficient integration of observations and simulations of the Earth. This is regarded as a canonical example of the cyber-physical system. The climate system model, the simulation engine in this cyber-physical system, is one of most challenging applications in scientific computing. It utilizes the multi-physics simulation that couples multiple components, conducts decadal to millennium simulations, and has long been an important application on supercomputers. However, current climate system models suffer from the inefficient task scheduling methods resulting in an intolerable simulation time. Take the Community Earth System Model (CESM), the most widely used climate system model, as an example, one major reason that CESM suffers from bad performances is the huge overhead to rationally distribute processes among the coupled heterogeneous components. According to the report of NCAR, every percent improvement in CESM performance frees up to the equivalent of $250,000 in computing resources in their scientific experiments. To address such challenge, our paper firstly constructs a lightweight and accurate performance model for effectively capturing and predicting the heterogeneous time-to-solution performance of end-to-end CESM components with a given simulation configuration. Then, based on the performance model, we further propose an efficient scheduling strategy based on rectangular packing method to determine the best process layout among different components, and the process numbers assigned to each component. Our evaluations show that we can achieve 58% average run time reductions on CESM comparing to the widely used sequential process layout for a scale of 144 to 480 cores on typical CPU clusters. And we can save 4 million CPU hours when we conduct one standard scientific experiment (a 2870-year simulation), which equals to save $40,089 with a charge of $0.01 per CPU hour. Meanwhile, 26% extra performance improvements could also be gained in our methods comparing to the heuristic branch and bound algorithm with the guidance of the known curve-fitting performance model.

Keywords: performance model, time-to-solution, scheduling, Cyber-Physical System, automatic tool

1. Introduction

Scenes of flooding and storms show us just how much weather and climate can affect our lives. Understanding and predicting how the weather and climate will change over the next century is of vital importance for both our economy and society [1]. The climate system model is regarded as the most critical and complicated component in the observation-simulation integration system nowadays [2–6]. The physical procedures, such as fluid-dynamical, physical, and biological procedures, and software components are deeply intertwined and each component can be performed on different spatial and temporal scales, exhibiting multiple and distinct behavioral modalities [7, 8]. Therefore, the coupled climate system model is taken as one of biggest and complicated cyber-physical system (CPS) [9].

As the coupled climate model has been developing rapidly in recent years [10, 11], the problem of ineffective using of modern computing platforms has been revealed. It is because each component is developed independently as stand-alone parallel models following diversified development roadmaps and heterogeneous time-to-solution performance characteristics. In recent decades, we can see quite a lot of work on accelerating climate simulation speed, such as the efforts on GRAPES [12], CAM-SE [13], AGCM [14], GFDL [15], POP [16], CICE [17] and CESM [18]. Even though with the efforts above, every percent of the climate simulations' improvement in time-to-solution performance frees up to the equivalent of $250,000 in computing resources according to the report of NCAR [10]. One major reason for such issue is due to the huge overhead to rationally distribute processes among the coupled heterogeneous components.

Thus an efficient scheduling method, which aims at improving the time-to-solution performance by optimizing the process layouts among components, is extremely important for reducing both time and money costs. Scheduling problems have been around for decades and there are some valuable research efforts that have been made, such as multiprocessor scheduling [19], graph scheduling problem [20] and SPMD (single-problem-multi-data) program process scheduling [21–23]. However, the scheduling issue in current climate models becomes more and more challenging with the growing of the resolution scales and coupled physics procedures [10]. First of all, the huge solution space (3.60E+13 when using 500 processes) caused by various combinations of process layout among components and process numbers in each component makes it impossible to exhaustively search for the optimal process distributions. Besides, components differ greatly in implementation, numerical algorithms and load characteristics, which brings difficulties to schedule processes based on the performance modeling. At last, it challenges lightweight profiling because of the expensive cost of coupled climate models.

Researchers have already conducted some work to overcome these challenges. One of the most significant work is that Kim et al. suggest changing the number of processes during execution between the components through a synthetic CCSM benchmark on the Malleable Model Coupling Toolkit with the support of CHARM++ and Adaptive MPI [24, 25]. Despite these efforts, it takes 14 million CPU hours to conduct a scientific experiment (2870-year simulation) using the 1 degree coupled climate models [10].

Our paper targets at proposing an integrated scheduling mechanism in climate models to reduce the simulation time by optimizing the process layout among components, and determining the best process number assigned to each component. We take the Community Earth System Model (CESM) [26], one of the state-of-the-art and widely used coupled system models with a large code base for more than one million and five hundred thousand LOCs, as an example. Because the heterogeneity of CESM components can cover most of the characteristics of the coupled climate system models around the world. Our method and experience achieved from our CESM study can benefit other climate system models and even other kinds of MPMD applications by making following contributions.

- ***A lightweight and accurate performance model***. We propose a high-level analytic performance model based on the generalized algorithm analysis with domain knowledge. Such performance model is easier to be ported to other climate models compared to the detailed analytic performance model [16] while maintaining a satisfactory model accuracy. Further study shows that short-term simulations can represent the timing of long-term simulations effectively. Our performance model error of CESM is 13.2%, which can achieve close to 20% improvement compared to the curve fitting method [27] on Tianhe-1A.

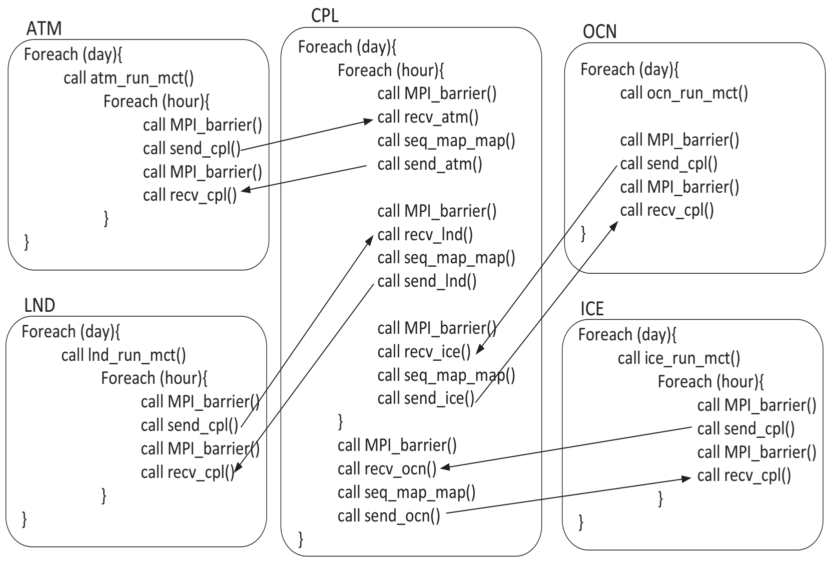
- ***An efficient scheduling strategy***. We design a two-phase scheduling strategy based on rectangular packing method to solve a mixed integer nonlinear programming (MINLP) problem [28]. Our method reduced the scheduling time complexity in one order of magnitude compared to the heuristic branch and bound strategy [18]. For a scale of 144 to 480 cores on typical CPU clusters, our approach can reduce the run time of simulation by 58% on arithmetic average, compared to the widely used sequential process layout. And we can save 4 million CPU hours when we conduct one standard scientific experiment (a 2870-year simulation), which equals to save $40,089 with a charge of $0.01 per CPU hour. When compared to the heuristic branch and bound algorithm [18] according to the known curve-fitting performance model, our approach achieves 26% extra performance improvement.

- ***An optimization tool that is easy-to-use***. An automatic scheduling tool has been designed, implemented and verified in this work, which is integrated into CESM script system. The user community can capture the computation time, communication time and message size of critical kernels of each process effectively and efficiently. They can build the performance model, look for the best process configuration and submit jobs tuned with the process configuration automatically.

2. Performance Analysis of CESM

In a real-world climate simulation, different components of the earth system have to be coupled to represent the interactions among global spheres and must run simultaneously as an MPMD system. The currently released CESM1.2 couples the Community Atmosphere Model Version 5 (ATM) [29], the Parallel Ocean Program Version 2 (OCN) [30], the Community Land Model Version 4 (CLM) [31], the Los Alamos sea ice model Version 4 (ICE) [32], and the river transport model (RTM) [33] by the coupler (CPL) [34]. They are all mature parallel models, in which we can utilize various computing resources supporting both distributed memory and shared memory. However, any two of the components do not communicate directly. They exchange their data through the coupler (CPL) [34], as Fig. 1 shows.

Typical coupled climate system models [35] contain the atmosphere component (ATM), the ocean component (OCN), the sea ice component (ICE), and the land component (LND). The atmosphere component is characterized by two phases, namely the dynamics and the physics. The two phases are executed in turn during each simulation time-step. The dynamics expresses the evolutionary equations for the atmospheric flow, which contains both computation and the update-halo communication. The finite-volume dynamical core (FV) is one of the most widely used atmosphere grids in the CESM. The resolution ($GLOBAL\_X$ x $GLOBAL\_Y$, also referred as horizontal grid) is decomposed into two-dimensional blocks using a Cartesian decomposition [27] as shown in Fig. 2. The third dimension (depth) extends into the page. A column is referred to as a horizontal grid with its third dimension. The size of a block is defined



```
Foreach (day){
        call atm_run_mct()
                Foreach (hour){
                        call MPI_barrier()
                        call send_cpl()
                        call MPI_barrier()
                        call recv_cpl()
                }
}
```

```
Foreach (day){
        Foreach (hour){
                call MPI_barrier()
                call recv_atm()
                call seq_map_map()
                call send_atm()

                call MPI_barrier()
                call recv_lnd()
                call seq_map_map()
                call send_lnd()

                call MPI_barrier()
                call recv_ice()
                call seq_map_map()
                call send_ice()
        }
        call MPI_barrier()
        call recv_ocn()
        call seq_map_map()
        call send_ocn()
}
```

```
Foreach (day){
        call ocn_run_mct()

        call MPI_barrier()
        call send_cpl()
        call MPI_barrier()
        call recv_cpl()
}
```

```
Foreach (day){
        call lnd_run_mct()
                Foreach (hour){
                        call MPI_barrier()
                        call send_cpl()
                        call MPI_barrier()
                        call recv_cpl()
                }
}
```

```
Foreach (day){
        call ice_run_mct()
                Foreach (hour){
                        call MPI_barrier()
                        call send_cpl()
                        call MPI_barrier()
                        call recv_cpl()
                }
}
```

Figure 1: An example of inter-communications among components. The arrows indicate the inter-communication pattern in CESM. For example, the ATM and OCN exchange interfacial flux and state data via CPL; the CPL has grid information for both ATM and OCN and carries out intergrid interpolation of state and flux data, and then sends the new data back to the ATM and OCN.

by the input parameters $BLOCK\_X$ and $BLOCK\_Y$, and presents $Block(i, j)$ along with its four neighbouring blocks in the two horizontal dimensions. Each block is assigned to a process, and it has a halo of ghost cells that permit tasks to proceed independently with minimal communication and synchronization, called update-halo. In each step, all processes have to exchange the values of physical variables by ghost cells. The main cost of physics is calculation in each independent column. Each column may be assigned with different amounts of workloads depending on the actual terrain.

Each time-step of the ocean component is divided into two phases, namely, baroclinic and barotropic. The baroclinic solver is computationally intensive. It utilizes an explicit time integration method for the three-dimensional fluid equations. This solver is a typical stencil-like program. The preconditioned

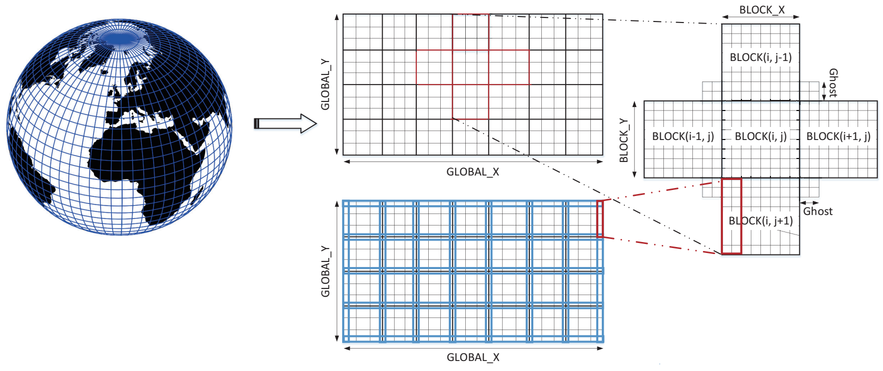


Figure 2: A block in the two-dimensional Cartesian data decomposition of FV dynamic core in ATM component.

conjugate gradient solver for the barotropic phase mainly consists of a two-dimensional nine-point stencil operator. This solver contains a lot of update-halo operations (similar to the ATM component) and collective communications. The collective communication is implemented using MPI_Allreduce.

The ice component acts as a barrier between the polar atmosphere and the ocean to hinder flux exchange, such as heat and greenhouse gas. Update-halo is a critical part with significant irregularity, and its computational cost is mainly manifested where sea ice exists. This part suffers load imbalance owing to that the sea ice changes both spatially and temporally during a climate simulation.

The land component is a single-column model with a nested sub-grid hierarchy. Its grids are composed of multiple land-units, each with multiple snow/soil with multiple plant types. The columns are grouped into blocks of nearly equal computational cost, and these blocks are subsequently assigned to MPI processes.

The grid partitions and high-level kernels are common senses in the climate models [36, 37]. Our performance model based on these high-level kernels can benefit other climate models easily.

3. Model-based Process Scheduling Optimization

Our model-based process-scheduling optimization tool uses two steps to improve time-to-solution performance, namely, predicting the runtime of the coupled climate system models, and determining the best process configuration. A lightweight performance model is built to predict runtime of each component on a given machine. The inputs of the performance model are the resolution, total process number, and the domain decomposition scheme (refers to the problem of determining the number of processes in x and y directions). The domain decomposition can be obtained in the CESM script system. The output is the estimated runtime of each component of the given number of processes. We then look for the best process configuration with the guidance of the performance model output.

3.1. Predicting the Runtime

As Fig. 3 shows that the ATM, OCN, ICE and LND components account for 95% of the total runtime. Therefore, we focus on detailed modeling of the OCN and ATM components and use a holistic modeling method for the LND and ICE components. The LND component takes a relatively short time. The ICE component has a serious load imbalance issue, and the kernels are inconspicuous. In the holistic modeling of the LND and ICE components, we estimate the computation and communication time. Their sum is the total runtime of the component because BSP programming [38] is a common method in climate models.

The computational model is built according to the horizontal grid number. We associate the number of grids of each process to the computation runtime, which has the advantages of estimating the other cases with different resolutions. The forecast for the computation time is shown is Eq. 1:

$$T\_comp(P) = a * (GLOBAL\_X * GLOBAL\_Y) * P^i * (log_2(P))^j + b \qquad (1)$$

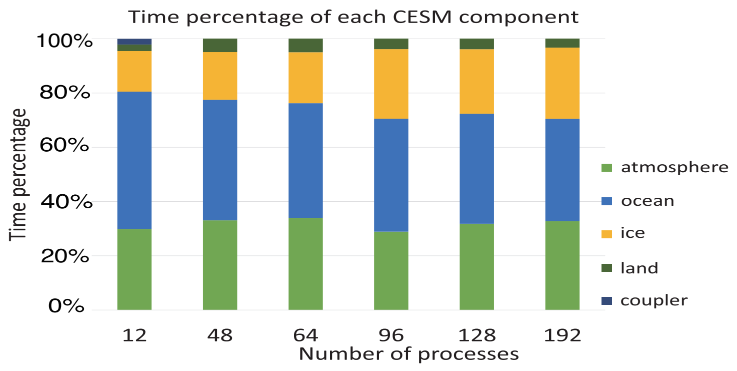


Figure 3: Time percentage of each component in a 30-day CESM sequential run using different number of processes on TH_HPCA (cf. Tabel 1). A sequential run is that the components share the same process pool.

In Eq. 1, $GLOBAL\_X$ and $GLOBAL\_Y$ are the total number of grids in x direction and y direction, P is the number of processes, a and b are the model parameters, i and j are in range of [-2,2] to determine the complexity of desired fitting functions [39].

The communication model is divided into two parts, namely, the point-to-point (p2p) communication model and the collective communication model. We focus on modeling the update-halo communication (multi-p2p communications) in strong scaling. A large amount of update-halo communication leads to large overhead and presents a great challenge if we try to model individual sent and received messages. According to HOCKEY model [40], the p2p communication time can be modelled as Eq. 2. Variable S_total is total communication volume, which can be calculated according to the resolution and domain decomposition (cf. Section 2), as Eq. 3 shows. Model parameters a and b are taken as the transferring speed and the network latency.

$$T\_halo(S\_total) = a * S\_total + b \tag{2}$$

The total halo size S_total equals to Eq. 3. P_X and P_Y are the numbers of processes in x and y directions, and $iter$ is the iteration times that is fixed for a given resolution. The variables P_X, P_Y, $iter$, $GLOBAL\_X$, $GLOBAL\_Y$ and $ghost$ can be found in the configuration files (as Fig. 2 shows in Section. 2).

$$S\_total = iter * ghost * (\lceil \frac{GLOBAL\_X}{P\_X} \rceil + \lceil \frac{GLOBAL\_X}{P\_X} \rceil) * \lceil \frac{GLOBAL\_X}{P\_X} \rceil * \lceil \frac{GLOBAL\_X}{P\_X} \rceil \tag{3}$$

The collective communication pattern in CESM is MPI_Allreduce. Eq. 4 shows the running time for a binomial tree MPI_Allreduce.

$$T\_allreduce(P) = a * log2(P) + b \tag{4}$$

3.2. Process Scheduling

Process scheduling in our work aims to determine the best number of processes assigned to each component, and the process layout across components, as shown in Fig. 4. The input is the estimated runtime of each component using the given number of processes. The output is the best process configuration.

We design a two-phase (process layout generation and process scheduling searching) process scheduling strategy based on the rectangular packing (RP) method [41] to further improve the searching efficiency and scalability. Thus the problem is how to pack the rectangles together into a container and the container size should be minimum. However, traditional RP can only deal with rectangles with fixed length and width. In our situation, the length differs, and the width varies with the change in length.

The solution space in process scheduling is up to $(m!)^2 * p^m$, where m represents the active component numbers, p is the total number of processes. Because one has to consider the process layout across the components $(m!)^2$ and the number of processes assigned to each component p^m at the same time. As we can see, when the parallelism reaches 500 cores with four tuning components, the solution space can be as large as 3.60E+13. Traditional methods usually cannot

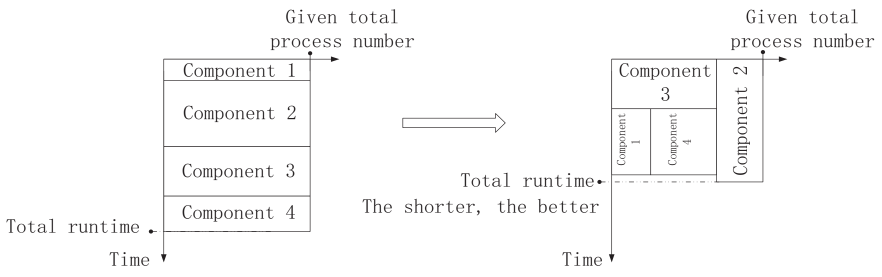


Figure 4: A general example of our process scheduling strategy based on the rectangle packing method.

solve this kind of onlinear multi-modal functions optimization (NMFO) problem [42] efficiently and effectively. One previous work uses the mixed integer nonlinear programming (MINLP) problem to search the process configuration [18]. MINLP has to give constraints of each process layout to complete the searching procedure. Besides, whether MINLP can get the collective optimal solution is sensitive to the initial value.

3.2.1. Process Layout Generation

We decouple the process layout and the number of processes of each component in the process layout generation part. The process layout generation part can be performed offline once the number of components is determined. We first enumerate all possible layouts and then remove the reduplicative layouts.

Generally, we classify the reduplicative layouts into two categories. The first category comprises the layouts with the same relative positions. In Fig. 5, the numbers indicate the relative positions among the rectangles by serialization. The three figures are considered three process layouts across components in the traditional method. However, the relative positions of the three rectangles are the same. Our strategy regards them as one process layout. The second category of reduplicative layouts includes the layouts that are symmetrical to one another, as shown in Fig. 6. The time complexity of process layout generation is $O(m!)$, where m is the component number ($m = 4$ in this paper).

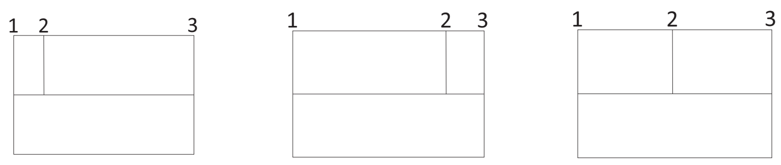


Figure 5: An example of the first kind of reduplicative layouts. Such kind of layouts have the same relative positions. Each rectangle is considered as a component. The X direction represents the number of processes and the y direction represents the time cost of the component. Three figures are considered as three process layouts across components in traditional method. The numbers on the figure indicate the relative positions among the rectangles. The relative positions of the three figures are the same. Our strategy takes them as one process layout.

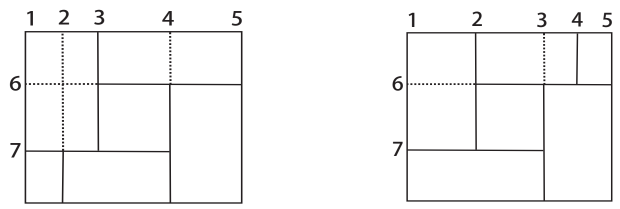


Figure 6: An example of the second kind of reduplicative layouts. Such kind of layouts are symmetrical to one another. Each rectangle is considered as a component. The X direction represents the number of processes and the y direction represents the time cost of the component. The numbers indicate the relative positions among the rectangles. Position NO.2 of the left figure and position No.4 are diagonal symmetry. So we take these two as one layout.

While the traditional method is $O(m^m)$ because it does not consider reducing the reduplicative layouts.

3.2.2. Process-scheduling Searching

In the process-scheduling searching part, we search for the best solution in the range of the total number of processes among the process layouts. Given the inherent serial parts and communication overhead, the component runtime will not decrease once it reaches a certain threshold of process number (sweet-spots). The computing resource needs not be further increased. Therefore, we can eliminate all situations that are out of the threshold. The threshold can be known through the output of the performance model.

We cache the optimal sub-layout information to avoid duplication of searching. When more than two components exist, we define a rectangle group (such as Fig. 5 and Fig. 6) that contains at least two components as a sub-layout. Such a group may be repeated in different process layouts. We determine the optimal solution for different numbers of processes through multiple times of Fibonacci searching [43], and cache the optimal results for this group, and reuse the results when we find the same group again. As Fig. 5 is shown, the two rectangles above can be considered as the sub-layout of the three rectangles. We cache the best process configuration (total run time, total process number, the number of processes of each rectangular, and the position of each rectangular) using different numbers of processes. That information can be used when searching for the best process configuration of three or more rectangles during later searching.

Finding sweet-spots and caching the sub-layout information can help us save up to 30% online searching time.As mentioned before, we assume that the runtime of a component is a convex function on the distribution of the process number; in other words, the runtime function has only one minimum point. The number of processes, that is larger than the number of processes at the minimum point, are being pruned.

The time complexity of each Fibonacci searching is $O(logp)$. So the time complexity of process scheduling searching is $O(log^m p)$. In summary, the time complexity of our process scheduling strategy is $O(m!log^m p)$. The process layout generation can be performed offline and we can use the bitwise store to save the disk space.

4. Implementation

A process configuration system is designed, implemented, and verified in this work. We integrate our system into the CESM script. As Fig.7 shows, the system consists of four modules, namely, Performance Model Builder, Process

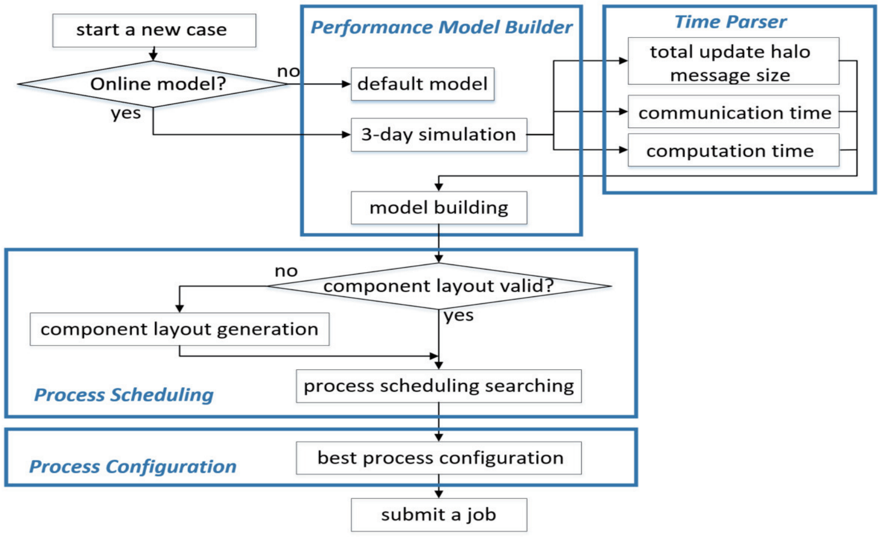


Figure 7: Implementation of the process configuration system.

Scheduling, Process Configuration, and Time Parser. The Performance Model Builder Module (corresponding to Section 3.1) estimates the computation and communication performances using the effective performance data providing by the Time Parser Module. The computation and communication performance models are then generated and their outputs are fed into the Process Scheduling Module. The Process Scheduling (corresponding to Section 3.2) is used to find a solution to improve the CESM performance. The Process Configuration Module is used to rearrange the processes configuration according to the Process Scheduling Module.

Our profiling tool is integrated into Time Parser Module. We mark each MPI operation by a unique ID (e.g. MPI_SEND ID is 15). We use PROFILE_START (ID) and PROFILE_STOP (ID, message_size) to collect the time cost and communication size of each MPI command. All profiling data are written into the file at MPI_Finalize in order to reduce the profiling overhead. We capture the CESM profile information by instrumenting PROFILE_INIT() and PROFILE_FINISH() into the main programs of each kernel of CESM files to control the profile scope.

Once the new case is created, the build script will add the common environment definition of the profiling tool to the *Macros* and *Makefile*. It will be automatically propagated to each component setup and build scripts. The user community can choose to use either the existing offline model to rearrange the process configuration, or build a new performance model online.

The Process Configuration Module includes a post-verification step. The module will parse the timing files of each run of the CESM and check the deviation between the time of model run and the predicted time of our performance model. If the deviation reaches a user-defined threshold, then our tools will restart the performance model building phase. We use this method to follow the change of the given computing environment.

5. Performance Evaluation

We conduct our experiments on three parallel platforms as is shown in Table 1. As a petascale supercomputer, Tianhe-1A features a massively parallel processor (MPP) architecture of hybrid CPU-GPU computing. A proprietary high-speed interconnection network, the TH-net, is designed and implemented to enhance the communication capabilities of the system. The topology of the TH-net is an optoelectronic hybrid, hierarchical fat tree. The MPI implementation on the Tianhe-1A is customized to achieve high-bandwidth and low latency data transfers. TH_HPCA is a dedicated-use cluster that only has 16 compute nodes for a total of 192 cores. HP cluster is also a dedicated-use cluster with 1024 nodes. The networks of TH_HPCA and HP Cluster are both InfiniBand QDR.

We use control run simulation (named B1850) with the resolution of f19_g16 (ATM and LND: 144 x 96 horizontal grid; OCN and ICE: 384 x 320 x 78) as our test case. B1850 represents all active components for pre-industrial simulation, and is used to place the climate model into a stable state before any historical experiments and projection experiments. The test case is scientifically validated

Table 1: Parallel platforms used in the evaluation.

	Tianhe-1A	TH_HPCA	HP cluster
CPU	2× Intel Xeon X5670 (6 cores)	2× Intel Xeon X5650 (6 cores)	2× Intel Xeon X5550 (4 cores)
Frequence	2.93GHz	2.67GHz	2.67GHz
Complier	icc 11.1	icc 11.0.069	icc 11.0.069
MPI	MPICH2 Version 1.4.1p1	Intel MPI Version 3.2	Intel MPI Version 3.2
File System	Lustre	NFS	Lustre
Network	TH-net (Fat-tree)	InfiniBand QDR (Fat-tree)	InfiniBand DDR (Fat-tree)
Node Number	7168	16	128

according to the CESM website [44]. The case is also the control run experiment of the IPCC AR5 experiments[45]. We do not consider I/O in current method because of the large disturbance of I/O.

In this paper, we performance modeling the B1850_f19_g16 on TH_HPCA, and compare the performance model error with the simple curve-fitting model [27]. We test the performance improvement of B1850_f19_g16 on Tianhe-1A and HP cluster.

We profile the case B1850_f19_g16 at two parallelisms (12 and 96 processes) to determine the model parameters. The performance model is verified in two aspects, namely, profiling overhead and model accuracy. In Section 5.1, we discuss the profiling overhead and the reason for using a 3-day simulation to predict the CESM runtime. In Section 5.2, we show the performance results of case B1850_f19_g16.

5.1. Profiling Overhead

Fig. 8 shows the time variance of each model hour of the ATM, LND, and ICE components at different model days in a half model year simulation, which is the arithmetic average of the three measurements on Tianhe-1A. The large variance is due to the imbalance computation load from a diurnal cycle. The time cost is large during the noon for the three components because of the increasing computation of radiation. At the end of each day, the components usually have

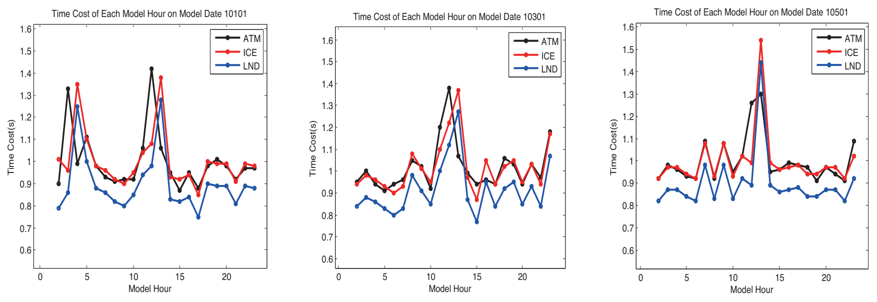


Figure 8: Time cost of each model hour on model date 10101, 10301 and 10501. For example, '10301' means that the first day ('01') of the third month ('03') of the first year ('1') in the model.

Table 2: Time cost of the process scheduling strategy.

# of cores	144	192	256	512	1024	2048	3.12M
Our strategy(s)	0.07	0.08	0.09	0.16	0.26	0.70	52.90
Branch and bound(s)	0.09	0.20	0.37	1.95	12.81	33.28	>3600

a global communication to exchange the value of the tracers, so the runtime becomes larger. Model date 10101 is the first day of the simulation when each component. The time consumption of every model hour in one day fluctuates greatly, and the characteristics of model hours in different days are not the same. In this situation, we use the model day to profile the CESM. The entire model remains stable after the third day. We conduct a three-day simulation and pick up the last two days to gather the performance data. In addition, we conduct a 500-year experiment on TH_HPCA. The 500-year experiment takes 45.7 days, and the predicted time using our performance model is 42.8 days. The difference implies a 6.26% prediction error, which is satisfied for practical use.

The process scheduling time is shown in Table 2. We search for the result using four components, and compare our strategy to Branch and Bound [23] method. When the process number reaches 3.12e6 (equals to the total process number of Tianhe-1A), the Branch and Bound method fails to find a solution within an acceptable time. In contrast, the time of our strategy is within one

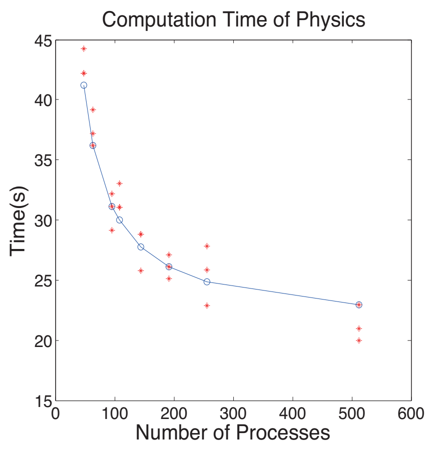
Table 3: The performance model on TH_HPCA.

	kernel	computation model	communication model
ATM	physics	$T\_comp(P) = 0.07*(144*96)/P + 21.07$	
	dynamic	$T\_comp(P) = 0.09*(144*96)/P + 14.68$	$T\_halo(S\_total) = 0.000006 * S\_total - 21.7$ $S\_total = nsteps * 2 * 26 * (P\_Y * 144 + P\_X * 96 - P\_X * P\_Y)$
OCN	baroclinic	$T\_comp(P) = 0.02 * (384 * 320)/P + 18.97$	$T\_halo(S\_total) = 0.0000025 * S\_total - 0.68$ $S\_total = nsteps*2*2*60*(P\_Y * 384 + P\_X * 320 - P\_X * P\_Y)$
	barotropic	$T\_comp(P) = 0.01 * (384 * 320)/P + 0.14$	$T\_coll(P) = 26.48*log2(P) - 93.88$
ICE		$T\_comp(P) = 0.01 * (384 * 320)/P - 4.96$	$T\_comm = (0.16 * P - 2.83)/P$
LND		$T\_comp(P) = -0.01 * (144 * 96)/P + 69.27$	$T\_comm = (0.12 * P - 0.31)/P$

minute which can be ignored compared with the simulation time.

5.2. Accuracy of the Performance Model

The performance model of case B1850_f19_g16 on TH_HPCA is shown in Table 3. Take physics in ATM component and baroclinic in OCN component as an example, the computation model parameter $a = 0.07$ of the physics is larger than the one of baroclinic ($a = 0.02$). This result is influenced by the computation amount, which relates to the resolution in the performance model. The horizontal resolution of the OCN component is 384 x 320 horizontal grid, and that of physics in the ATM component is 144 x 96, which makes the actual time of baroclinic scales faster than that of the physics as Fig. 9 shows. Fig. 10 illustrates that the update-halo of the dynamic in the ATM component and that of the baroclinic in the OCN component share the similar runtime performance of strong scaling. The parameter values of the baroclinic are smaller than those of the dynamic. This result is due to that the update-halo happens among the blocks covered by the sea, and those who are covered by land are ignored, which



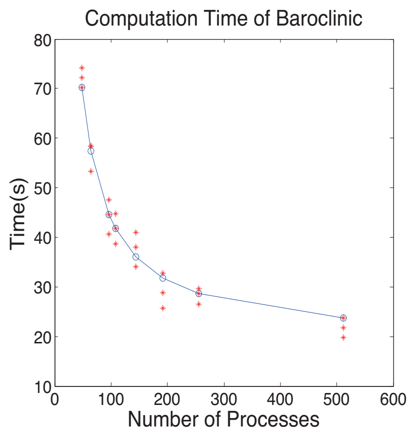
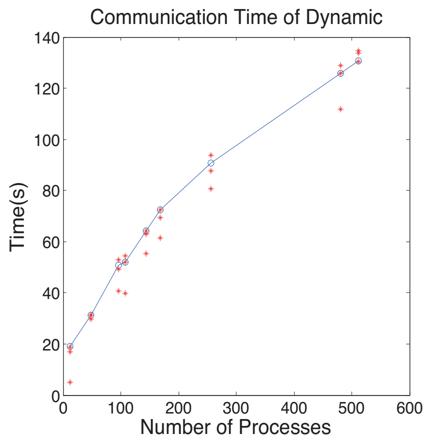


Figure 9: Computation time of physics and baroclinic. The blue lines are the predicted runtime of performance model. The red stars are the real time cost using different number of processes.



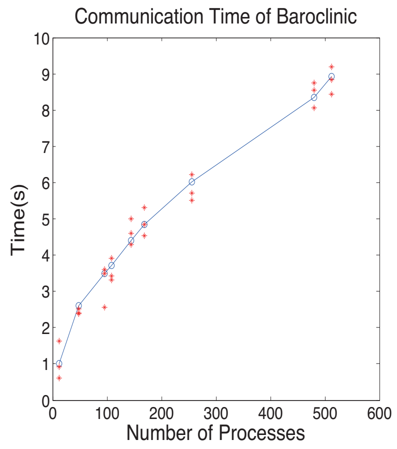


Figure 10: Update-halo time of dynamic and baroclinic. The blue lines are the predicted runtime of performance model. The red stars are the real time cost using different number of processes.

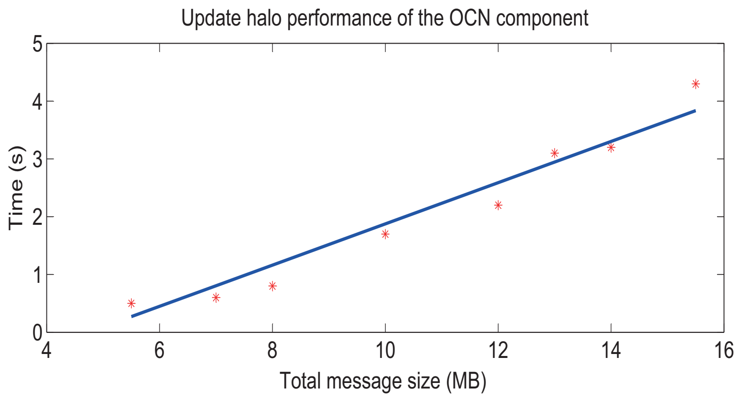


Figure 11: Performance of update-halo in baroclinic of the OCN component for 3-day run on TH_HPCA.

in turn makes the total message size and communication time small.

In Table 3, P_X and P_Y are the numbers of processes in x and y directions, respectively. The total message size equals to the amount of communication in one update-halo step multiplied by the number of steps ($nsteps$). Fig. 11 shows the communication time is nearly linearly scaling with the communication size. The performance model is listed in Table 3.

The interpolation and extrapolation model errors on TH_HPCA are shown in Fig. 12. The ICE component has a relatively large error up to 26%, but the error of curve-fitting model is even larger. The error of the ICE component comes from the serious load imbalance. The ice component has to balance the time cost of radiation computation from Arctic and Antarctic, as well as the horizontal transport tracer halo and elastic-viscous-plastic sub-cycling halo [46]. The halo cost dependents on the distribution of the ice block, and as the computation time falls, the update-halo time rises. Such an imbalance causes a significant challenge to performance modeling. Nonetheless, the ICE component reaches its peak performance at 168 processes. We then ignore all of the predicted ICE component results with process number larger than 168. Currently we are further investigating this issue in advance.

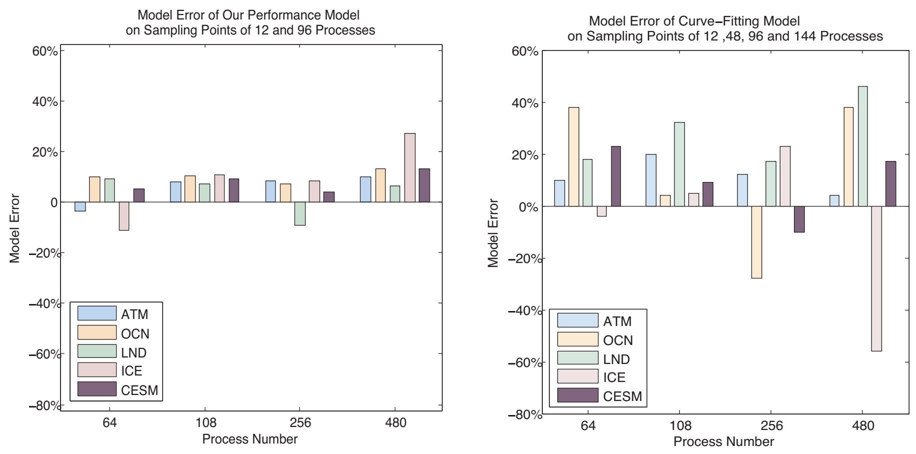


Figure 12: Performance model error of our model and the curve-fitting model.

Fig. 12 shows the curve-fitting [27] model error. Worley *et al.* use Amdahl's law to build the performance model for each component. The model is $T\_component = a/P + b * P^c + d$, where P is the number of compute nodes. The entire CESM error of the curve-fitting model can be as high as 20% and the component error is up to 56% at 480 processes. Such coarse-grained curve-fitting modeling lacks insight into the communication roadmap and characteristics of each kernel. Moreover, the higher order terms of the curve-fitting performance model according to the Amdahl's law are redundant, which in turn increases the profiling times but cannot improve the model precision. Our performance model uses 12 and 96 processes as the profiling points, and the curve-fitting model uses 12, 48, 96, and 144 processes as the profiling points. Our model error is 30% smaller, and we further reduce the profiling points from 4 to 2 during the model construction.

Table 4 shows our model error of each component using different profiling number of processes on TH_HPCA. The model errors may become larger when performing extrapolation of the performance model. When we use profiling points of 48 and 192 processes to build the performance model, the model error of the ATM component with 12 processes increases to 15.37%. The model error

Table 4: Performance model error of B1850_f19_g16 using different profiling points.

Profiling points	#of cores	ATM	OCN	LND	ICE	CESM
12,	48	2.15%	5.37%	5.17%	5.17%	2.91%
192	96	1.01%	7.44%	-0.3%	3.83%	3.78%
48,	12	13.29%	-8.0%	-10%	8.62%	-1.2%
96	192	-3.2%	1.90%	2.77%	-3.3%	-1.1%
48,	12	15.37%	-10%	-8.0%	-10%	-3.2%
192	96	2.91%	2.17%	-5.1%	1.11%	2.03%
96,	12	8.17%	-14%	-3.5%	-16%	-7.9%
192	48	1.30%	-7.6%	2.78%	-3.9%	-3.8%

of the ICE component increases to 16.33%, and the model error of the OCN component increases to 14.65% using profiling points of 96 and 192 processes to extrapolate the solution time with 12 processes. According to Table 4, the model errors are less than nearly 15% with different profiling points, which prove the effectiveness of our performance model.

In summary, our model error lies in two aspects. The first aspect is decomposition. The data decomposition in a CESM run is set to *AUTO*. Various decomposition can lead to performance fluctuation, including the update-halo performance and even the computation load of each process. The second aspect is the system effect [47], which refers to the influence of the system and other applications, including the memory and network contention.

5.3. Performance Improvement

With the guidance of our performance model, the runtime performance is improved by up to 58.49% on Tianhe-1A, and 57.98% on HP cluster (Fig. 13 and Fig. 14). We can save 4 million CPU hours when we conduct one 2870-year simulation, which equals to save \$40,089 with a charge of \$0.01 per CPU hour. We compare the CESM default process configuration and the curve-fitting model on Tianhe-1A using our process scheduling strategy. Our model has 26.15% performance improvement compared with the curve-fitting model on Tianhe-1A, and 23.47% on HP cluster.

The performance model accuracy can greatly affect the tuning results. The

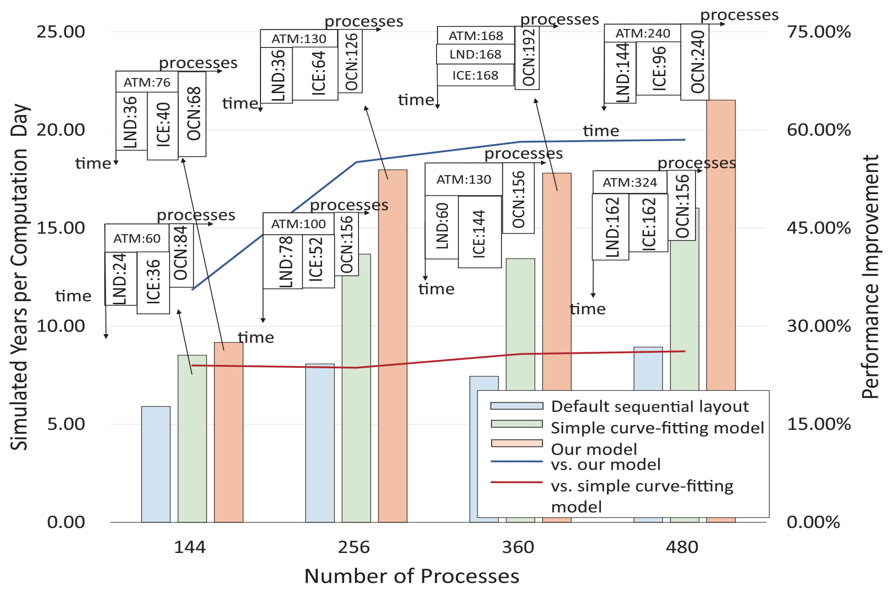


Figure 13: Performance improvement of B1850_f19_g16 on Tianhe-1A. Simulated year per computation day is a performance indicator of the coupled climate models, the higher, the better. The process number of each component and the process layout across component are marked on the figure. The layout is different from others on 360 processes. It is because the process number allocated to the ICE and LND components are 168 which reaches their peak performance according to the performance model. We have 58% performance improvements using 480 cores, which can help to save $40,089 for one 2870-year scientific experiment with a charge of $0.01 per CPU hour.

DLB[24, 48] method has 33.8% performance improvement on CCSM benchmark. On the contrary, our process scheduling strategy ensures to search optimal solution within the feasible solutions for process layout and number of processes. The DLB method can only search for the best number of process of each component under the condition of a given total process number and the same process layout across components.

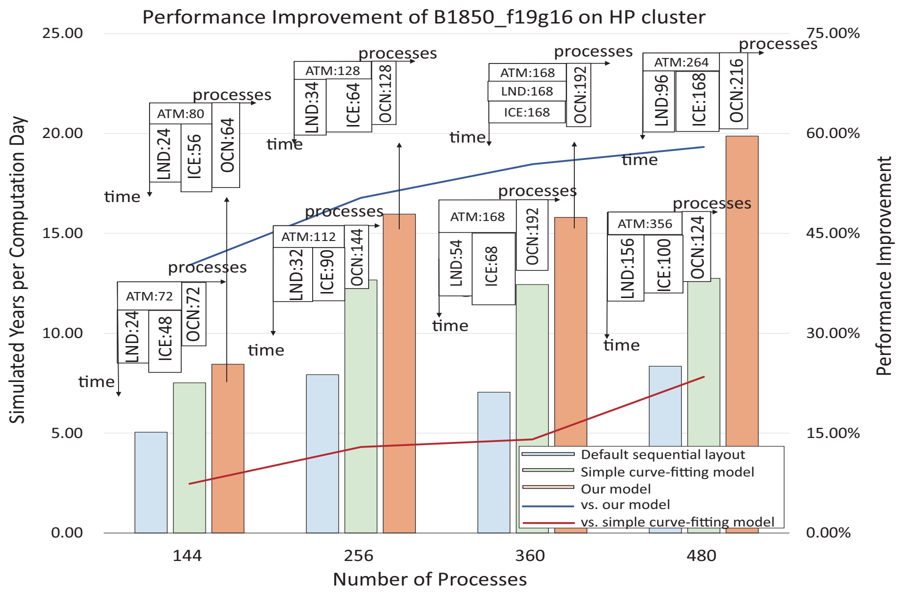


Figure 14: Performance improvement of B1850_f19_g16 on HP cluster. The process number of each component and the process layout across component are marked on the figure.

6. Related Work

6.1. Process Scheduling

425 There are two common process scheduling methods: dynamic process scheduling (DPS) and static process scheduling (SPS). DPS is applied in the D.Kim *et al.*'s work [25, 48]. D.Kim *et al.* change the process number during execution both within and between the components through a synthetic CCSM benchmark on the Malleable Model Coupling Toolkit [24] with the support of
430 CHARM++ and Adaptive MPI. They build and adjust the performance model by using the measurement data continuously collected from previous iterations during the each model simulation day by tracing the operations in CPL of CCSM benchmark. Predictions from the performance model are used to guide process scheduling decisions. They re-allocate the process every model day in a
435 range of 2^0 to 2^5 processes until the algorithm is convergence, as such a method can be converged at a local optimal value in solving such complex multi-modal

function optimization problems. Although such a DPS technique needs limited prior knowledge about the performance characteristics, it requires that the real coupled climate models have the ability to dynamically re-allocate computational resources at runtime, which is not supported in the current climate models. Moreover, their DPS strategy can be easily trapped at a local optimal value in solving such complex multi-modal function optimization problems. In general, DPS usually has a strong reliance on the software environment. It can easily leave the burden of providing accurate load information on when and how to re-balance the application. Various process scheduling strategies and algorithms have been proposed to support DPS. Centralized strategies, which collect the information onto one process, and decision algorithms run sequentially. Such methods may show good performance in a small parallelism, but when it comes to thousands of processes, it raises the performance bottleneck because of the memory capacity of the single process. Hierarchical strategies [21, 22] are proposed. They generate a group of processes and collect information at the root of them. A higher level of the hierarchy receives the aggregate information, and although reducing the memory cost of profiling, it suffers excessive data collection at the lower level of the hierarchy. Soon-Heum Ko and Felipe Bertrand employ the dynamic process scheduling on the multi-physics coupled components[49, 50].

SPS is a common approach to improve the performance of the large-scale scientific computing applications. A heuristic static process scheduling algorithm is applied to the Fragment Molecular Orbital (FMO) Method [23] along with a curve-fitting based performance model. Alexeev *et al.* [23] use branch and bound algorithm to look for a suitable process number for the application. The authors used the Branch and Bound (BB) method to determine the best process number with one component. Such a method shows great performance on FMO since it is a one-dimensional optimization problem. It is not suitable for CESM. Because there are four important components in CESM that must be considered into load balancing, which results in a three-dimensional optimization problem. The solution space of our load balance problem is up to

10^{13}, and the overhead of the BB method is beyond the limitation. What's more, the higher order terms of the curve-fitting performance model according to the Amdahl's law[23, 51] are redundant, which increases the profiling times but cannot improve the model precision. The model error is 30% larger than ours.

6.2. Performance Modeling

In general, there are three methods of performance modeling: simulation modeling, empirical performance modeling, and analytic performance modeling. Simulation modeling [52, 53] uses the simulator to reconstruct the behavior of the application, which usually takes an excessive amount of time. An empirical performance model [54] is established by observing the macro performance of an application on specific machines. An analytical performance model[55, 56] usually counts the number of basic operations, including float-point operations, memory accesses, etc. Such models have to be designed carefully to trade off the model parameters versus the required accuracy[57]. A great deal of work has been done using semi-analytical modeling[57–59]. This approach models the performance of kernels empirically and composes them analytically as the whole performance model.

Considering the complexity of CESM, lots of work has been done on performance modeling. Alexeev *et al.* [23] use the curve-fitting method to predict the performance. Although their performance model has low overhead, the model accuracy cannot be satisfied due to the lack of considering the kernel characteristics and the hardware and software interaction, the error of their performance model can achieve even up to 56%, which leads to a 26% performance decrease when using the model for CESM tuning. Kerbyson *et al.* [60] built a fine-grained sample-based performance model for the POP component in the fat-tree InfiniBand network. They modeled the performance of kernels of POP by analyzing the algorithm and using the mathematical expressions for predicting performance metrics. This work has inspired us in how to build a lightweight and accurate performance model.

7. Conclusion

We are at a critical juncture in the evolution of high-performance computing, from tera-scale to peta-scale, even to exa-scale, with an increasing amount of parallelism. The era of increasing the concurrency among components is coming with the approach of combing multiple MPI programs to simulate one complex scenario becoming more and more common. We demonstrate a promising approach that can significantly reduce the computation time of complex multi-model simulation with a smart process scheduler.

We design and implement a model-based process scheduler for the CESM – a coupled climate system model. We believe our work and experience provide an important base for automatic process scheduling among multi-model physical simulations. We use a rectangular packing method to schedule the process layouts among components, which lowers the complexity of the problem from $O((mp)^m)$ to $O(m!log^m p)$, where m is the number of components, and p is the number of processes. For a scale of 144 cores to 480 cores on typical CPU clusters, our approach runs for less than one minute and can reduce the run time of simulation by 58% on arithmetic average and save \$40,089 with a charge of \$0.01 per CPU hour, compared with the widely used sequential process layout. Compared with the heuristic branch and bound searching algorithm according to the known curve-fitting performance model, our approach achieves 26% additional performance benefits.

With this advanced scheduler in hand, we will reduce the performance variance across clusters obviously and take the best utilization of different parallel computers, thus brings us a better simulation system, which can improve the computing efficiency of observation-simulation integration system enabling faster climate knowledge discovery and prediction.

Considering that memory/network contention may lead to decreased performance, we will concentrate on taking them in our future work to improve our performance model and scheduling strategy.

References

[1] Why climate important, https://www.ncas.ac.uk/en/why-is-climate-important.

[2] T. C. Peterson, D. R. Easterling, T. R. Karl, P. Groisman, N. Nicholls, N. Plummer, S. Torok, I. Auer, R. Boehm, D. Gullett, et al., Homogeneity adjustments of in situ atmospheric climate data: a review, International Journal of Climatology 18 (13) (1998) 1493–1517.

[3] J. W. Hurrell, M. M. Holland, P. R. Gent, S. Ghan, J. E. Kay, P. J. Kushner, J.-F. Lamarque, W. G. Large, D. Lawrence, K. Lindsay, et al., The community earth system model: a framework for collaborative research, Bulletin of the American Meteorological Society 94 (9) (2013) 1339–1360.

[4] Z. Deng, W. Han, L. Wang, R. Ranjan, A. Y. Zomaya, W. Jie, An efficient online direction-preserving compression approach for trajectory streaming data, Future Generation Computer Systems 68 (2017) 150–162.

[5] Y. Ma, H. Wu, L. Wang, B. Huang, R. Ranjan, A. Zomaya, W. Jie, Remote sensing big data computing: Challenges and opportunities, Future Generation Computer Systems 51 (2015) 47–60.

[6] A. Barbu, A. Segers, M. Schaap, A. Heemink, P. Builtjes, A multi-component data assimilation experiment directed to sulphur dioxide and sulphate over europe, Atmospheric Environment 43 (9) (2009) 1622–1631.

[7] E. Burakowski, S. Ollinger, G. Bonan, C. Wake, J. Dibb, D. Hollinger, Evaluating the climate effects of reforestation in new england using a weather research and forecasting (wrf) model multiphysics ensemble, Journal of Climate 29 (14) (2016) 5141–5156.

[8] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, et al., Multiphysics simulations: Challenges and opportunities, The International Journal of High Performance Computing Applications 27 (1) (2013) 4–83.

[9] R. Baheti, H. Gill, Cyber-physical systems, The impact of control technology 12 (2011) 161–166.

[10] R. A.Gerber, H. J.Wasserman, Large scale computing and storage requirements for biological and environ-mental science: Target 2017, Tech. rep., DOE Office of Sciennce, Office of Biological and Environmental Research, Office of Advanced Scientific Computing Research, and National Energy Research Scientific Computing Center (2012).

[11] P. Bauer, A. Thorpe, G. Brunet, The quiet revolution of numerical weather prediction, Nature 525 (7567) (2015) 47.

[12] Z. Wang, X. Xu, N. Xiong, L. T. Yang, W. Zhao, Gpu acceleration for grapes meteorological model, in: High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on, IEEE, 2011, pp. 365–372.

[13] I. Carpenter, R. Archibald, K. J. Evans, J. Larkin, P. Micikevicius, M. Norman, J. Rosinski, J. Schwarzmeier, M. A. Taylor, Progress towards accelerating homme on hybrid multi-core systems, The International Journal of High Performance Computing Applications 27 (3) (2013) 335–347.

[14] Y. Wang, J. Jiang, H. Zhang, X. Dong, L. Wang, R. Ranjan, A. Y. Zomaya, A scalable parallel algorithm for atmospheric general circulation models on a multi-core cluster, Future Generation Computer Systems 72 (2017) 1–10.

[15] V. Balaji, R. Benson, B. Wyman, I. Held, Coarse-grained component concurrency in earth system modeling: parallelizing atmospheric radiative transfer in the gfdl am3 model using the flexible modeling system coupling framework, Geoscientific Model Development 9 (10) (2016) 3605.

[16] B. van Werkhoven, J. Maassen, M. Kliphuis, H. Dijkstra, S. Brunnabend, M. Van Meersbergen, F. Seinstra, H. Bal, A distributed computing approach to improve the performance of the parallel ocean program (v2. 1), Geoscientific Model Development 7 (1) (2014) 267–281.

[17] P. Balaprakash, Y. Alexeev, S. A. Mickelson, S. Leyffer, R. Jacob, A. Craig, Machine-learning-based load balancing for community ice code component in cesm, in: International Conference on High Performance Computing for Computational Science, Springer, 2014, pp. 79–91.

[18] D. Nan, X. Wei, J. Xu, X. Haoyu, S. Zhenya, Cesmtuner: An auto-tuning framework for the community earth system model, in: High Performance Computing and Communications (HPCC), 2014 IEEE Intl Conf on, IEEE, 2014, pp. 282–289.

[19] J. C. Liou, M. A. Palis, A comparison of general approaches to multiprocessor scheduling, in: Parallel Processing Symposium, 1997. Proceedings., International, 1997, pp. 152–156.

[20] C. Wu, R. Tobar, K. Vinsen, A. Wicenec, D. Pallot, B. Lao, R. Wang, T. An, M. Boulton, I. Cooper, Daliuge: A graph execution framework for harnessing the astronomical data deluge, Astronomy & Computing 20.

[21] J. Lifflander, S. Krishnamoorthy, L. V. Kale, Work stealing and persistence-based load balancers for iterative overdecomposed applications, in: Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing, ACM, 2012, pp. 137–148.

[22] G. Zheng, A. Bhatel, E. Meneses, L. V. Kal, Periodic hierarchical load balancing for large supercomputers, International Journal of High Performance Computing Applications 25 (4) (2011) 371–385.

[23] Y. Alexeev, A. Mahajan, S. Leyffer, G. Fletcher, D. G. Fedorov, Heuristic static load-balancing algorithm applied to the fragment molecular orbital method, in: High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for, IEEE, 2012, pp. 1–13.

[24] D. Kim, J. W. Larson, K. Chiu, Malleable model coupling with prediction, in: Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, IEEE, 2012, pp. 360–367.

[25] D. Kim, J. W. Larson, K. Chiu, Automatic performance prediction for load-balancing coupled models, in: Cluster, Cloud and Grid Computing (CC-Grid), 2013 13th IEEE/ACM International Symposium on, IEEE, 2013, pp. 410–417.

[26] S. Vadlamani, Current efforts for performance analysis and enhancements of cesm, Tech. rep., National Center for Atmospheric Research (2014). URL `http://www.vecpar.org/posters/vecpar2014_submission_49.pdf`

[27] P. H. Worley, A. P. Craig, J. M. Dennis, A. Mirin, M. Taylor, M. Vertenstein, et al., Performance of the community earth system model, Tech. rep. (2011).

[28] I. Quesada, I. E. Grossmann, An lp/nlp based branch and bound algorithm for convex minlp optimization problems, Computers &amp; chemical engineering 16 (10) (1992) 937–947.

[29] R. Smith, P. Gent, Reference manual for the parallel ocean program (pop), ocean component of the community climate system model (ccsm2. 0 and 3.0), Tech. rep., Technical Report LA-UR-02-2484, Los Alamos National Laboratory, Los Alamos, NM, http://www. ccsm. ucar. edu/models/ccsm3.0/pop (2002).

[30] R. B. Neale, C. Chen, A. Gettelman, P. Lauritzen, S. Park, D. Williamson, A. Conley, R. Garcia, D. Kinnison, J. Lamarque, et al., Description of the ncar community atmosphere model (cam 5.0), NCAR Tech. Note NCAR/TN-486+ STR.

[31] K. W. Oleson, D. M. Lawrence, B. Gordon, M. G. Flanner, E. Kluzek, J. Peter, S. Levis, S. C. Swenson, E. Thornton, J. Feddema, et al., Technical description of version 4.0 of the community land model (clm).

[32] E. Hunke, W. Lipscomb, Cice: the los alamos sea ice model, documentation and software user's manual, version 4.1.

[33] C. S. E. Group, Cesm user's guide (cesm1.2 release series user's guide), Tech. rep., the National Science Foundation, the Department of Energy, the National Aeronautics and Space Administration, and the University Corporation for Atmospheric Research National Center for Atmospheric Research, http://www.cesm.ucar.edu/models/cesm1.2 (2013).

[34] A. P. Craig, M. Vertenstein, R. Jacob, A new flexible coupler for earth system modeling developed for ccsm4 and cesm1, International Journal of High Performance Computing Applications 26 (1) (2012) 31–42.

[35] P. R. Gent, G. Danabasoglu, L. J. Donner, M. M. Holland, E. C. Hunke, S. R. Jayne, D. M. Lawrence, R. B. Neale, P. J. Rasch, M. Vertenstein, et al., The community climate system model version 4, Journal of Climate 24 (19) (2011) 4973–4991.

[36] W. Yuqing, W. Hui, et al., Improvements in climate simulation with modifications to the tiedtke convective parameterization in the grid-point atmospheric model of iap lasg (gamil), Advances in Atmospheric Sciences 24 (2) (2007) 323–335.

[37] S. A. Klein, X. Jiang, J. Boyle, S. Malyshev, S. Xie, Diagnosis of the summertime warm and dry bias over the us southern great plains in the gfdl climate model using a weather forecasting approach, Geophysical research letters 33 (18).

[38] M. Goudreau, K. Lang, S. Rao, T. Suel, T. Tsantilas, Towards efficiency and portability: Programming with the bsp model, in: Proceedings of the eighth annual ACM symposium on Parallel algorithms and architectures, ACM, 1996, pp. 1–12.

[39] A. Calotoiu, D. Beckingsale, C. W. Earl, T. Hoefler, I. Karlin, M. Schulz, F. Wolf, Fast multi-parameter performance modeling.

[40] C.-Y. Chou, H.-Y. Chang, S.-T. Wang, K.-C. Huang, C.-Y. Shen, An im-

proved model for predicting hpl performance, in: International Conference on Grid and Pervasive Computing, Springer, 2007, pp. 158–168.

[41] Y.-L. Wu, W. Huang, S.-c. Lau, C. Wong, G. H. Young, An effective quasi-human based heuristic for solving the rectangle packing problem, European Journal of Operational Research 141 (2) (2002) 341–358.

[42] K. Deb, Multi-objective genetic algorithms: Problem difficulties and construction of test problems, Evolutionary computation 7 (3) (1999) 205–230.

[43] M. Zhou, G. Bao, K. Pahlavan, Measurement of motion detection of wireless capsule endoscope inside large intestine, in: Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference on, IEEE, 2014, pp. 5591–5594.

[44] W. C. Porter, Community earth system model: Implementation, validation, and applications.

[45] A. IPCC, Intergovernmental panel on climate change (2007).

[46] A. P. Craig, S. A. Mickelson, E. C. Hunke, D. A. Bailey, Improved parallel performance of the cice model in cesm1, International Journal of High Performance Computing Applications 29 (2) (2015) 154–165.

[47] A. Bhatele, K. Mohror, S. H. Langer, K. E. Isaacs, There goes the neighborhood: Performance degradation due to nearby jobs, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '13, ACM, New York, NY, USA, 2013, pp. 41:1–41:12. `doi:10.1145/2503210.2503247`.
URL `http://doi.acm.org/10.1145/2503210.2503247`

[48] D. Kim, J. W. Larson, K. Chiu, Malleable model coupling with prediction, in: Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, IEEE, 2012, pp. 360–367.

[49] F. Bertrand, R. Bramley, D. E. Bernholdt, J. A. Kohl, A. Sussman, J. W. Larson, K. B. Damevski, Data redistribution and remote method invocation for coupled components, Journal of Parallel and Distributed Computing 66 (7) (2006) 931–946.

[50] S.-H. Ko, N. Kim, J. Kim, A. Thota, S. Jha, Efficient runtime environment for coupled multi-physics simulations: Dynamic resource allocation and load-balancing, in: Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on, IEEE, 2010, pp. 349–358.

[51] J. M. Dennis, M. Vertenstein, P. H. Worley, A. A. Mirin, A. P. Craig, R. Jacob, S. Mickelson, Computational performance of ultra-high-resolution capability in the community earth system model, International Journal of High Performance Computing Applications 26 (1) (2012) 5–16.

[52] W. Zhang, X. Ji, Y. Lu, H. Wang, H. Chen, P. C. Yew, Prophet: A parallel instruction-oriented many-core simulator, IEEE Transactions on Parallel & Distributed Systems PP (99) (2017) 1–1.

[53] W. Zhang, X. Ji, B. Song, S. Yu, H. Chen, T. Li, P. C. Yew, W. Zhao, Varcatcher: A framework for tackling performance variability of parallel workloads on multi-core, IEEE Transactions on Parallel & Distributed Systems 28 (4) (2017) 1215–1228.

[54] B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. De Supinski, M. Schulz, A regression-based approach to scalability prediction, in: Proceedings of the 22nd annual international conference on Supercomputing, ACM, 2008, pp. 368–377.

[55] A. Agarwal, J. Hennessy, M. Horowitz, An analytical cache model, ACM Transactions on Computer Systems (TOCS) 7 (2) (1989) 184–215.

[56] Y. Lu, X. Wang, W. Zhang, H. Chen, L. Peng, W. Zhao, Performance analysis of multimedia retrieval workloads running on multicores, IEEE Transactions on Parallel & Distributed Systems 27 (11) (2016) 3323–3337.

[57] T. Hoefler, W. Gropp, W. Kramer, M. Snir, Performance modeling for systematic performance tuning, in: State of the Practice Reports, ACM, 2011, p. 6.

[58] G. Bauer, S. Gottlieb, T. Hoefler, Performance modeling and comparative analysis of the milc lattice qcd application su3_rmd, in: Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, IEEE, 2012, pp. 652–659.

[59] D. J. Kerbyson, H. J. Alme, A. Hoisie, F. Petrini, H. J. Wasserman, M. Gittings, Predictive performance and scalability modeling of a large-scale application, in: Proceedings of the 2001 ACM/IEEE conference on Supercomputing, ACM, 2001, pp. 37–37.

[60] D. J. Kerbyson, P. W. Jones, A performance model of the parallel ocean program, International Journal of High Performance Computing Applications 19 (3) (2005) 261–276.