# Improving Parallel Performance of a Finite-Difference AGCM on Modern High-Performance Computers

LI LIU,* RUIZHE LI,+ GUANGWEN YANG,# BIN WANG,@ LIJUAN LI,& AND YE PU&

* *Ministry of Education Key Laboratory for Earth System Modeling, Center for Earth System Science, Tsinghua University, Beijing, China*
+ *Department of Computer Science and Technology, Tsinghua University, Beijing, China*
# *Department of Computer Science and Technology, and Ministry of Education Key Laboratory for Earth System Modeling, Center for Earth System Science, Tsinghua University, Beijing, China*
@ *Ministry of Education Key Laboratory for Earth System Modeling, Center for Earth System Science, Tsinghua University, and State Key Laboratory of Numerical Modeling for Atmospheric Sciences and Geophysical Fluid Dynamics, Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing, China*
& *State Key Laboratory of Numerical Modeling for Atmospheric Sciences and Geophysical Fluid Dynamics, Institute of Atmospheric Physics, Chinese Academy of Sciences, Beijing, China*

(Manuscript received 18 March 2013, in final form 1 May 2014)

ABSTRACT

The rapid development of science and technology has enabled finer and finer resolutions in atmospheric general circulation models (AGCMs). Parallelization becomes progressively more critical as the resolution of AGCMs increases. This paper presents a new parallel version of the finite-difference Gridpoint Atmospheric Model of the Institute of Atmospheric Physics (IAP)–State Key Laboratory of Numerical Modeling for Atmospheric Sciences and Geophysical Fluid Dynamics (LASG; GAMIL) with various parallel optimization strategies, including two-dimensional hybrid parallel decomposition; hybrid parallel programming; parallel communications for coupling the physical packages, land surface, and dynamical core; and a cascading solution to the tridiagonal equations used in the dynamical core. The new parallel version under two different horizontal resolutions (1° and 0.25°) is evaluated. The new parallel version enables GAMIL to achieve higher parallel efficiency and utilize a greater number of CPU cores. GAMIL1° achieves 37.8% parallel efficiency using 960 CPU cores, while GAMIL0.25° achieves 57.5% parallel efficiency.

## 1. Introduction

Atmospheric general circulation models (AGCMs) are large-scale computing programs that consist of thousands lines of code. AGCMs are used to simulate and predict the evolution of the general circulation of the atmosphere (GCA), or the global patterns of air movement, on computers. Shifts in the GCA pattern are thought to be a main cause and symptom of climate change, so that AGCMs are among the most important components of climate system models. The performance of AGCMs has improved substantially under the rapid development of science and technology. Many important aspects of the GCA can be successfully captured by state-of-the-art AGCMs, but uncertainties remain an important issue. These uncertainties arise partly because of limitations in the understanding of physical processes in the GCA and partly because the relatively low resolution of the models. Clouds are a typical example: AGCMs are often unable to adequately reproduce the spatial distribution, regional characteristics, and radiative effects of clouds. The continued improvement of physical parameterizations and refinement of spatial and temporal resolutions are therefore two of the most important initiatives in current AGCM development. With respect to clouds, cloud-resolving models (Khairoutdinov and Randall 2001) with very fine resolutions have also become an important direction in AGCM development. These research directions depend critically on high-performance computing techniques.

Fortunately, the computing power of high-performance computers approximately doubles every year because of the rapid increase in the number of processor cores. This increase in computing power enables AGCMs to be run at higher and higher resolutions. The acceleration of

*Corresponding author address:* Dr. Li Liu, Center for Earth System Science, Tsinghua University, Meng Minwei Science and Technology Building, Room S-817, Beijing 100084, China.
E-mail: liuli-cess@tsinghua.edu.cn

DOI: 10.1175/JTECH-D-13-00067.1

high-resolution AGCM simulations on modern high-performance computers requires the careful application of parallelization techniques. Parallelization techniques for AGCMs have been studied for more than 10 years (Dennis et al. 2012; Lauritzen et al. 2011; Michalakes et al. 2004; Sato 2004; Schaffer and Suarez 2000). Most of these works have demonstrated that 2D parallel decomposition and parallel programming with hybrid Message Passing Interface (MPI) and Open Multiprocessing (OpenMP) are effective means of parallelizing AGCMs on modern high-performance computers. However, different AGCMs with similar parallelization strategies often achieve a different parallel performance. These differences arise from differences in numerical algorithms, especially those in the dynamical cores. For example, versions of the Community Atmosphere Model (CAM) with a spectral-element dynamical core (CAM-SE), a finite-volume dynamical core (CAM-FV), and an Eulerian global spectral dynamical core (CAM-EUL) achieved different parallel performances on the same high-performance computer (Dennis et al. 2012). CAM-SE0.25° and CAM-FV0.25° (both with 777 602 horizontal grid points) were able to effectively utilize all 172 800 CPU cores on the Oak Ridge National Laboratory Leadership Computing Facility (LCF) Cray XT5 Jaguar petaflop (PF) system, which was the world's TOP1 high-performance computer in 2009. By contrast, CAM-EUL T341 ($1024 \times 512 = 524\,288$ horizontal grid points) was only able to effectively utilize 10 800 CPU cores on the same system. Moreover, parallelization resulted in a greater speed up for CAM-SE0.25° than for CAM-FV0.25° given the same number of CPU cores.

The Gridpoint Atmospheric Model of the Institute of Atmospheric Physics (IAP)–State Key Laboratory of Numerical Modeling for Atmospheric Sciences and Geophysical Fluid Dynamics (LASG; GAMIL; Wang et al. 2004; Li et al. 2013a) is a finite-difference AGCM that has been widely used in model intercomparison projects and scientific researches. It is the atmospheric component of the gridpoint versions of the Flexible Global Ocean–Atmosphere–Land System Model (FGOALS-g) series of coupled climate system models (CSMs), including FGOALS-g1.0 and FGOALS-g2 (Li et al. 2013b). These CSMs contributed results to the model intercomparisons conducted for the Intergovernmental Panel on Climate Change (IPCC) Fourth Assessment Report (AR4) and AR5. FGOALS-g2 performs particularly well with respect to simulating the El Niño–Southern Oscillation (Bellenger et al. 2013).

The current parallel version of GAMIL can only effectively utilize a small number of CPU cores (e.g., GAMIL1° can only effectively utilize 161 CPU cores), and the parallel efficiency achieved is low. GAMIL2.8° was used in FGOALS-g2 for almost all experiments in phase 5 of the Coupled Model Intercomparison (CMIP5). The spatial resolution of this model is coarse and the time step is big; consequently, the original parallel version of GAMIL was barely able to finish all of the experiments on time. The future development of GAMIL (including finer resolution, improved physical processes, nonhydrostatic considerations, etc.) requires a reparallelization of GAMIL for better parallel performance. In this work, we construct a new parallel version of GAMIL with several parallel optimizations, including the hybrid 2D parallel decomposition (2DD), hybrid parallel programming (HPP), parallel communications (PC), and a cascading solution to the tridiagonal equations (CS) in the dynamical core. This new parallel version dramatically improves the parallel efficiency and enables GAMIL to effectively utilize a greater number of CPU cores. We also compare the new parallel version of GAMIL with a state-of-the-art AGCM to reveal the differences in scalability and parallel efficiency.

The remainder of this paper is organized as follows. Section 2 introduces the relevant background material and related work. Section 3 presents our parallelization strategies. Section 4 empirically evaluates these optimizations and analyzes the parallelization overhead. We summarize our conclusions in section 5.

## 2. Description of model, high-performance computer, and existing parallelization

In this section, we first briefly introduce the background, including GAMIL, high-performance computers, and parallel programming. We then briefly describe the current parallel version of GAMIL.

### *a. GAMIL*

GAMIL is an AGCM developed at the LASG within the IAP at the Chinese Academy of Sciences (CAS). This model has participated in various international model intercomparison projects and has been widely used in scientific studies (e.g., Li et al. 2007a,b; Wu and Li 2008; Kuang et al. 2009; Kucharski et al. 2009; Scaife et al. 2009; Zou et al. 2009; Hodson et al. 2010; Li and Wang 2010; Guo et al. 2011; Dong et al. 2012; Mao and Li 2012; Xie et al. 2012; Lin 2007; Yu and Sun 2009; Zheng and Yu 2009; Yan et al. 2010).

GAMIL is based on an Eulerian finite-difference dynamical core. The discrete grid includes a hybrid horizontal grid with a Gaussian grid (at low resolutions) or uniform grid (at high resolutions) in low and middle latitudes and a weighted equal-area grid in high and

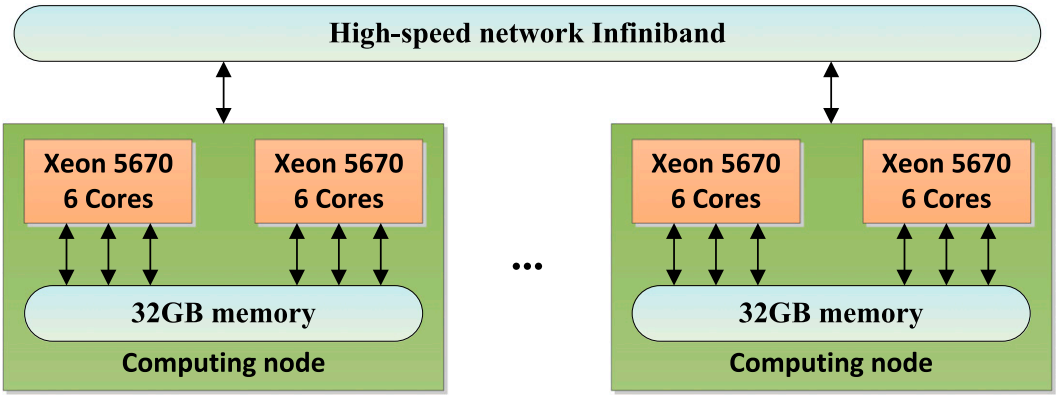



FIG. 1. Main architecture of the high-performance computer Tansuo100. The peak memory bandwidth in each computing node is 42 GB $s^{-1}$, and the peak communication bandwidth between computing nodes is 5 GB $s^{-1}$.

polar latitudes. GAMIL is run on 26 vertical sigma levels (pressure normalized by surface pressure), with a model top at 2.194 hPa. The dynamical core includes a finite-difference scheme that conserves mass and effective energy (Wang et al. 2004), and a two-step shape-preserving advection scheme for the moisture equation (Yu 1994). A more detailed description of GAMIL has been provided by Li et al. (2013a). The software features of the GAMIL AGCM will be described in section 3a.

There are currently four versions of GAMIL. All four versions use the same 26$\sigma$ vertical grid but have different horizontal resolutions: 4° × 5°, 2.8° × 2.8°, 1° × 1°, and 0.25° × 0.25°. Future development of GAMIL will include higher resolution in both the horizontal and vertical directions, improvements to the physical parameterizations to match the higher resolution, and consideration of nonhydrostatic conditions. Future developments may also include unstructured grids in the polar regions, which will allow larger time steps.

### b. High-performance computers

High-performance computers contain massive numbers of processors and usually consist of a large number of computing nodes connected by high-speed networks. Each computing node consists of CPUs with multiple cores sharing the same memory and sometimes also contains graphics processing units (GPUs) with a large number of cores for accelerating computations. We focus on CPU-only nodes without considering GPUs, because we find that GAMIL is memory intensive and communication intensive rather than computationally intensive. We also neglect vector processors, which are not popular in modern high-performance computers. Figure 1 shows the main architecture of the high-performance computer used in this paper, Tansuo100 at Tsinghua University in China. Tansuo100 consists of more than 700 computing nodes, each of which consists of two Intel Xeon 5670 six-core CPUs sharing 32 GB of main memory with a 42 GB $s^{-1}$ memory bandwidth. All computing nodes are connected by a high-speed Infiniband network with a peak bandwidth of 5 GB $s^{-1}$.

MPI and OpenMP are the most widely used parallel programming models. MPI provides standard Message Passing Interfaces for communication between processes. As different processes can run on both the same and different computing nodes, MPI is able to utilize all of the CPU cores in high-performance computers. OpenMP provides primitives for parallelization of a code segment such as a loop with multiple threads that share the same memory. OpenMP can only utilize the CPU cores that share the same memory and therefore it is limited to cores in the same computing node. We therefore have two choices for parallelizing GAMIL on high-performance computers: MPI only or a hybrid combination of MPI and OpenMP.

### c. Original parallel version of GAMIL

GAMIL currently uses an MPI-only parallelization, with a 1D parallel decomposition in which all grid points with the same latitude must be assigned to the same process. This parallelization technique has two significant drawbacks. First, the parallelization of GAMIL can only use a small number of CPU cores for acceleration. For example, GAMIL1° (i.e., the version with a horizontal resolution of 1°) can only use at most 161 CPU cores, while GAMIL0.25° (the version with a horizontal resolution of 0.25°) can only use at most 616 CPU cores. This limitation arises because these versions of the model have 161 and 616 different latitudes, respectively. Second, the parallel efficiency is low and decreases heavily as the number of CPU cores increases. When the number of CPU cores doubles, the communication

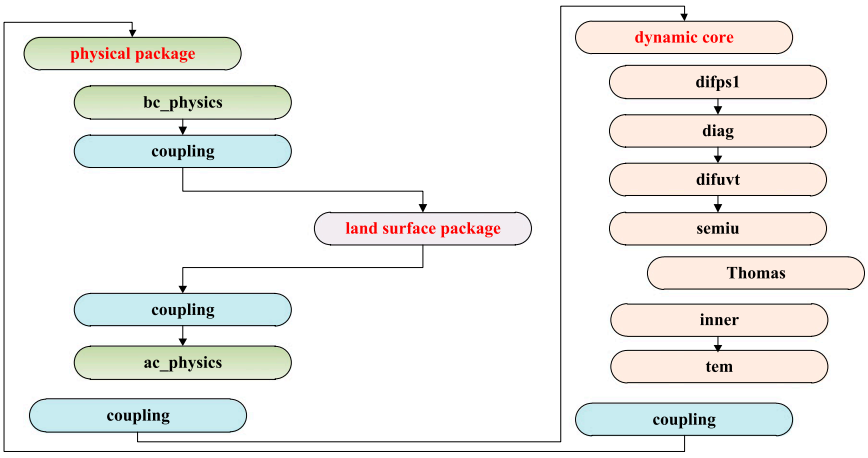



FIG. 2. Main procedures in GAMIL. The subroutine bc_physics consists of physical processes before coupling (including convection, cloud macrophysical/microphysical processes, radiation, etc.). The subroutine ac_physics consists of physical processes after coupling (including surface flux, vertical diffusion, planetary boundary layer, gravity wave drag, etc.). The subroutine difps1 computes the tendency of surface pressure and diagnoses vertical velocity using a central difference technique in space. The subroutine diag is a diagnostic program that converts basic model variables to real physical variables and diagnoses potential height. The subroutine difuvt calculates the tendencies of horizontal velocity and temperature based on central differences in space. The subroutine semiu obtains solutions to the zonal surface gravity wave equations using the Thomas algorithm. The subroutine inner defines the inner product of basic model variables. The subroutine tem calculates total energy and total mass.

overhead of each process remains almost identical, while the computation time for each process halves. This results in a rapid decrease in the parallel efficiency when more CPU cores are engaged.

## 3. Parallel optimizations of GAMIL

In this section, we describe the new parallel version of GAMIL. We first examine the software features of GAMIL and then individually present each of the optimization strategies used in the new parallel version.

### *a. Software features of GAMIL*

The GAMIL kernel can be partitioned into three packages (Fig. 2): the physical package, the land surface package, and the dynamical core. The land surface package is the Community Land Model, version 3 (CLM3), land surface model (Vertenstein et al. 2004). Most of the computational load is generally due to the physical package and dynamical core, which each range from 30% to 60%. The land surface package always accounts for less than 7% of the computational load. These packages are relatively independent (with different grids, algorithms, data structures, etc.) and must therefore be coupled. The physical package and land surface package use the Arakawa A grid, while the dynamical core uses the Arakawa C grid. The physical package and land surface package are column procedures (i.e., grid points are independent in the horizontal direction), while grid points in the dynamical core interact in both the horizontal and vertical directions. The physical package and land surface package decompose the horizontal grid into a number of columns and use column data structures to record all of the variables in each column for better data access locality. By contrast, the dynamical core uses array data structures to record each variable on the whole grid.

The dynamical core consists of several subroutines (Fig. 2), including difps1, diag, difuvt, semiu, inner and tem, among others. Grid points are highly interdependent in the dynamical core. These procedures are therefore communication intensive and memory intensive. Table 1 lists the communication behavior of each subroutine according to three types: neighbor, partial, and global. Each subroutine involves complex communication. Moreover, semiu is a special subroutine. This subroutine uses the Thomas algorithm (Thomas 1949) to solve tridiagonal equations, each of which uses information from all horizontal grid points with the same latitude. The Thomas algorithm is a simplified algorithm of the Gauss elimination for tridiagonal equations with only $O(n)$ operations (where $n$ is the number of equations). This algorithm is a sequential method, consisting of a forward and a backward substitution with a loop-carried dependency. A detailed description of the Thomas algorithm has been provided by Press et al. 2007.

TABLE 1. Communication behaviors of each subroutine in the dynamical core: neighbor means that each grid point depends on the grid points around it, partial means that grid points with the same latitude depend on each other, and global means that all points depend on each other.

| Subroutine | Neighbor | Partial | Global |
|---|---|---|---|
| Difps1 | √ | √ | |
| Diag | √ | | |
| Difuvt | √ | √ | |
| Semiu | √ | √ | |
| Inner | | | √ |
| Tem | | | √ |

In summary, the main procedures in GAMIL can be partitioned into three packages: the physical package, the land surface package, and the dynamical core. The physical package and the land surface package are column procedures with column data structures to improve data access locality. The dynamical core is memory intensive and communication intensive and contains complex communication behavior.

### b. *Hybrid 2D parallel decomposition*

The existing 1D parallel decomposition of GAMIL limits the number of CPU cores and results in low parallel efficiency (section 2c). We therefore propose a 2D parallel decomposition in this work. Figure 3 shows an example of a regular 2D parallel decomposition, in which the horizontal grid on the left side is decomposed into $2 \times 4$ regions on the right side. The communication overhead of a halo (the perimeter of a region) decreases to half its original size when the number of regions in the longitude and latitude directions both double. In other words, the 2D parallel decomposition decreases the communication overhead when using more CPU cores. We do not propose to decompose the vertical direction, because the number of vertical levels is generally small and the column procedures included in the physical package do not need to be parallelized.

A straightforward approach for 2D parallel decomposition is using a unique decomposition for the whole GAMIL. Given the regular 2D parallel decomposition in Fig. 3, it will introduce load imbalance to the physical package, because the computation overhead on each horizontal grid point varies with day and night, rain and sunshine, etc. Similarly, the land surface package will also encounter the problem of load imbalance, because the land cover varies heavily on different horizontal grid points. Given an irregular 2D parallel decomposition, which can achieve load balance in the physical package or land surface package, it will result in more point-to-point communications and smaller message size in the dynamical core, which is a communication-intensive procedure almost without the problem of load imbalance.

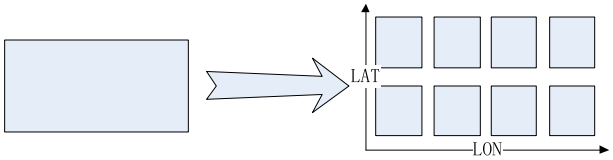



FIG. 3. An example of regular 2D parallel decomposition.

We therefore propose a hybrid 2D parallel decomposition of GAMIL (Fig. 4), where each package has its own parallel decomposition. We have tried several load assignment strategies to reduce the load imbalance in the physical package, finally selecting the strategy of assigning geocentrically symmetrical grid points to the same process. For the land surface package, we implement a round-robin strategy that sequentially assigns each grid point to all processes in a circular order. Note that only grid points with land are concerned. For the dynamical core, we decompose the horizontal grid into regions with an equal number of grid points. To minimize communication overhead, the number of regions is required to be the same as the number of processes. The size of each halo region can be computed using Eq. (1), where x_proc is the number of processes in the zonal direction and y_proc is the number of processes in the meridional direction:

$$\text{Halo\_size} = 2(\text{num\_lon}/\text{x\_proc} + \text{num\_lat}/\text{y\_proc}) \quad (1)$$

According to the dynamical core algorithms, each boundary of the halo region has only one level. The specific parallelization for the physical package and land surface package is easy to implement because these packages consist of column procedures with column data structure. For the dynamical core, we have implemented a parallel library tool to simplify the work.

As the dynamical core is communication intensive and memory intensive, we also implement a number of secondary optimizations. These secondary optimizations include packing transferred variables and performing loop transformations (such as loop interchange and loop fusion). Multiple variables in point-to-point communications are packed together to enlarge message size and to improve the communication bandwidth. Loop interchange means interchanging loop levels so that arrays are continuously accessed. This modification improves data access locality and memory performance. Loop fusion merges adjacent loops into one loop to reduce the number of times arrays must be accessed from memory and to improve the overall memory performance.

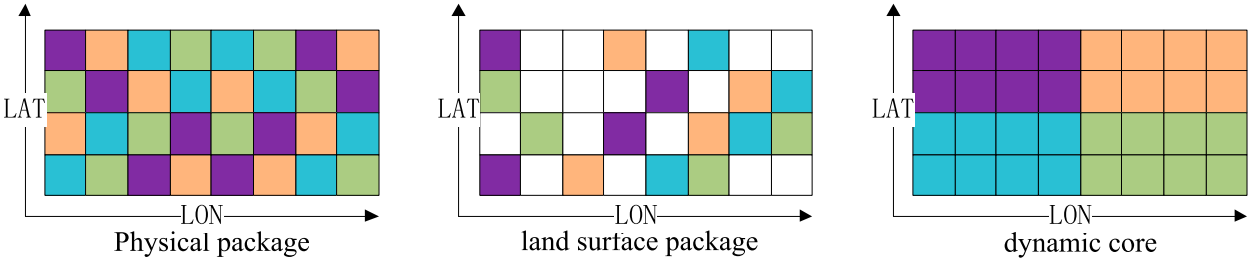



FIG. 4. Hybrid 2D parallel decomposition of GAMIL with four processes for the (left) physical package, (middle) land surface package, and (right) dynamical core. Each cell stands for a horizontal grid point. White cells stand for ocean-only points neglected by the land surface package, while other colors indicate different processes.

These optimizations improve GAMIL1° performance by 7.3% when using 960 CPU cores.

The initialization overhead is small (especially when the model is used to simulate long periods, such as tens of years), but we still apply the hybrid 2D parallel decomposition. The initialization of each model package is parallelized using the corresponding 2D parallel decomposition.

The hybrid 2D decomposition results in additional communication in the coupling procedures relative to the unique 2D decomposition. However, only a few variables are involved in the coupling, and the coupling procedures are accelerated using parallel communication (section 3d). The additional overhead related to coupling the hybrid 2D decomposition can therefore be neglected compared to the benefit.

### c. *Hybrid parallel programming with MPI and OpenMP*

To further improve the parallel efficiency of GAMIL, we implement hybrid parallel programming with MPI and OpenMP (Fig. 5). MPI manages all of the computing nodes, each of which runs only one process. The hybrid 2D parallel decomposition determines the grid points and the computations involved in each process. OpenMP manages all of the CPU cores that share the same memory and parallelizes the computations across all of the grid cells in the process. In particular, we use OpenMP loop parallelization to parallelize more than 100 loops in the GAMIL code. The detailed implementation of MPI uses nonblocking asynchronous point-to-point communication to hide communication latency. All MPI communications are master only and take place in the master thread, outside of the regions of code parallelized using OpenMP. We do not use one-sided MPI communication in our implementation, because it cannot currently achieve nonblocking asynchronous communication and it does not force partial synchronization. The latter feature is essential for guaranteeing the correct halo boundaries for every process.

The new hybrid MPI and OpenMP implementation has the following advantages over the existing MPI-only parallel programming:

1) It minimizes the number of processes, which reduces the communication overhead and improves the parallel efficiency.
2) The use of OpenMP inside individual processes further parallelizes GAMIL in both the horizontal and vertical directions and increases the parallelism of the model.
3) The dynamic scheduling of OpenMP can easily achieve load balance in each process, which further improves the parallel efficiency. We use OpenMP dynamic scheduling in the physical and land surface packages.

### d. *Parallel communication*

The hybrid 2D parallel decomposition means that the same process operates on different horizontal grid points in the physical package, land surface package, and dynamical core. It is therefore necessary to transfer data between processes during the coupling step. For example, referring to Fig. 4, the dynamical core in the purple grid points requires data from the same grid points from the physical package, which is scattered across four processes. The original communication approach in the coupling between the physical package and land surface package is sequential and consists of two steps (Fig. 6). First, the root process (the purple process in the middle left of Fig. 6) gathers all data needed by the land surface package from the processes decomposed for the physical package. Second, the root process scatters the gathered data to the processes decomposed for the land surface package. The root process has to transfer all of the data and represents a bottleneck in the coupling procedure. We use a parallel communication optimization to improve the performance, as shown in the right panel of Fig. 6. Processes send and receive data according to the mapping of grid points between the

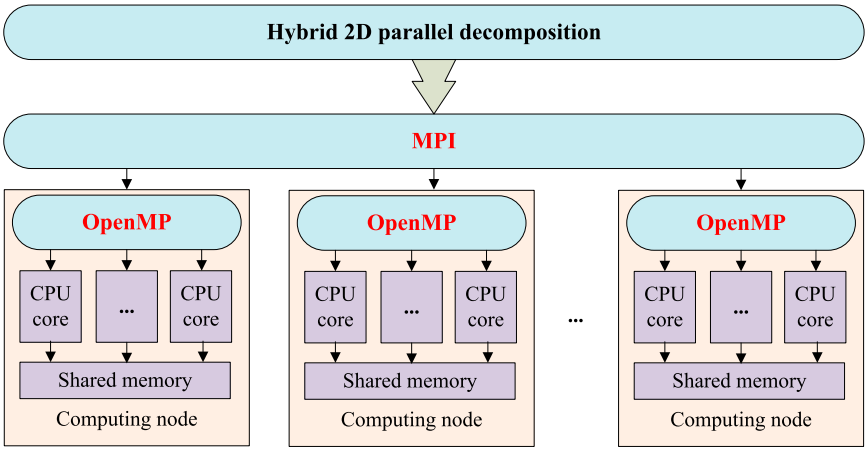



FIG. 5. Hybrid parallel programming using MPI and OpenMP.

physical package and the land surface package. This parallel communication eliminates the root process bottleneck by allowing different process pairs to transfer data in parallel. For example, the purple process can transfer data with the pink process at the same time as the blue process transfers data with the green process. Similarly, parallel communication has been used in the coupling procedures between the physical package and dynamical core.

### e. *Cascading solution to tridiagonal equation*

As mentioned in section 3a, the subroutine "semiu" in the dynamical core solves tridiagonal equations using the Thomas algorithm. The 2D parallel decomposition makes it more challenging to solve these equations. This difficulty results because each equation uses information from all the zonal grid points at the same latitude, which have been distributed among different processes by the 2D parallel decomposition of the dynamical core. Moreover, the Thomas algorithm is a sequential method with low time complexity. A straightforward solution to this problem is to use only one process to solve the equations, as shown in the left panel of Fig. 7. This approach gathers and scatters almost all of the data used in the equations, and therefore requires large data transfers (especially as the resolution of the model increases). Moreover, the gather and scatter procedures belong to collective communication and lead to significant communication overhead, especially when a large number of computing nodes are used. We therefore propose a cascading solution without a gather-and-scatter step, as shown in the right panel of Fig. 7. This approach guarantees the sequence of the Thomas algorithm. It also significantly reduces communication overhead because it only transfers data on the boundary grid points between adjacent processes in the longitude direction. In this case, each process contains only one boundary grid point. This means that the amount of data transferred in the cascading solution is very small.

Under the 2D parallel decomposition of the dynamical core, each process must solve multiple tridiagonal equations. We use OpenMP to ensure that each process can solve multiple tridiagonal equations in parallel.

These tridiagonal systems can also be solved in parallel using iterative schemes, which can achieve better parallelism. We choose the ordered cascading solution

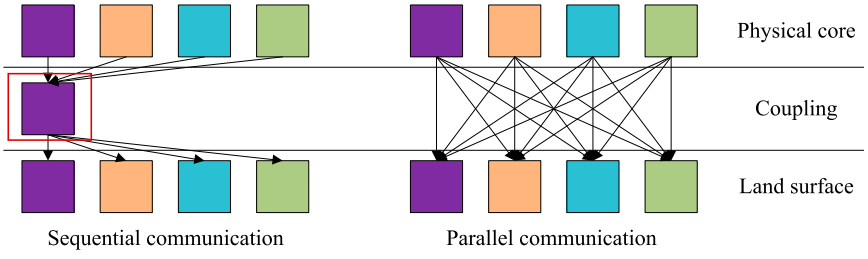



FIG. 6. (left) Original sequential communication and (right) proposed parallel communication in the coupling between the physical and land surface packages. Each box is a process. The purple process represents the root process for sequential communication.


FIG. 7. (left) Straightforward gather/scatter approach and (right) cascading approach to solve tridiagonal equations after a 2D parallel decomposition. Blocks stand for processes sharing the same latitudes (and hence the same tridiagonal equations). The red process is the root process for the gather/scatter approach.

rather than an iterative scheme because the cascading solution accounts for only a small portion of the entire execution time (less than 3% in our current application when using 960 CPU cores). An iterative scheme would have to be balanced against a potentially significant increase in the number of flops/memory transfers and would lower the precision of the solution. We will reconsider implementing an iterative scheme in the future when the cascading solution becomes a bottleneck with respect to parallel performance.

## 4. Experimental evaluation

We combine all of the optimization strategies described in section 3 to implement a new parallel version of GAMIL. In this section, we evaluate this new parallel version. We first describe the experimental design, followed by the results of our evaluation.

### *a. Experimental design*

We evaluate the new parallel version of GAMIL under two different horizontal resolutions (GAMIL1° and GAMIL0.25°) running on the high-performance computer Tansuo100 at the Tsinghua National Laboratory for Information Science and Technology (section 2b). Our experiments for this evaluation are restricted to only about 1000 CPU cores because of limitations in the computing resources available to each user. We do not account for input/output (I/O) overhead because our optimizations do not yet include it. We compile the model code using version 11.1 of the Intel C/C++/FORTRAN compiler and version 4.0 of the Intel MPI library.

### *b. Correctness of parallelization*

Before evaluating the parallel performance, we verify whether the new parallel version is correct. Simulations using the new parallel version of GAMIL are bit identical under different parallel settings (such as the number of CPU cores, the number of processes, and the number of OpenMP threads), even after long simulations spanning tens of model years. These results demonstrate that the parallelization is correct.

### *c. Performance comparison with original work*

We initially evaluate the performance of the new parallel version by comparing it to that of the original parallel version (Figs. 8 and 9). We use the performance on four CPU cores (four computing nodes, each of which provides a single CPU core) as a baseline in Fig. 9 because GAMIL0.25° requires more memory than that available on a single computing node. When the CPU core number is small (such as 48 CPU cores in Fig. 9), the original parallel version performs slightly better than the new parallel version. This is because of the additional overhead introduced by the hybrid 2D decomposition and the implementation of OpenMP in the new parallel version. The results shown in Figs. 8 and 9 indicate that the new parallel version enables GAMIL to more efficiently utilize a greater number of CPU cores. GAMIL1° simulates about 4.3 model years per day when using 240 cores, 7.3 model years per day when using 480 cores, and 11.5 model years per day when using 960 cores.

### *d. Impact of each parallel optimization strategy*

We next evaluate the relative impact of each proposed parallelization strategy, including the hybrid 2D decomposition, hybrid parallel programming with MPI and OpenMP, parallel communication, and the cascading solution for the tridiagonal equations (Fig. 10). Figure 10a shows that the hybrid 2D decomposition enables GAMIL to effectively utilize more CPU cores. We use only the parallel version with the hybrid 2D decomposition rather than the original parallel version as the baseline in Fig. 10b, because the original parallel version cannot utilize a large number of CPU cores (Fig. 10a). When scaling the CPU core number from 1 to 120, the hybrid parallel programming does not improve parallel performance (i.e., the speedup is less than or equal to 1) when scaling the number of CPU cores from

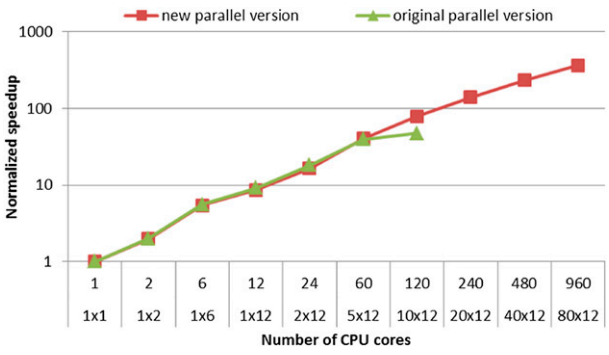



FIG. 8. Speedup ($y$ axis) of the original parallel version (green) and new parallel version (red) of GAMIL1°, using various numbers of CPU cores ($x$ axis). Labels along the $x$ axis indicate (top) the number of total CPU cores used and (bottom) the number of computing nodes $N$ times the number of CPU cores on each node M. Results for the original parallel version are omitted for 240–960 CPU cores because this version cannot use more than 161 cores. All speedup values have been normalized to the original parallel version using one CPU core.

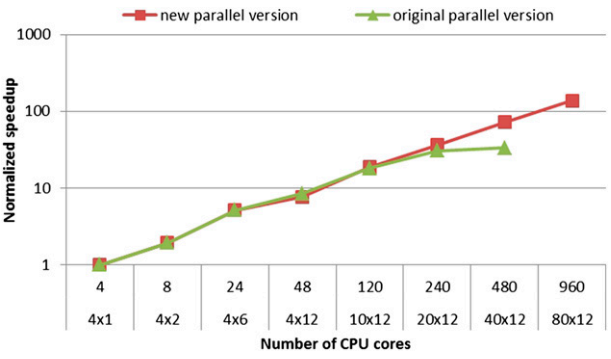



FIG. 9. As in Fig. 8, but for GAMIL0.25°. Results using the original parallel version are omitted for 960 CPU cores because this version cannot use more than 616 cores. All speedup values have been normalized to the original parallel version using $4 \times 1$ CPU cores.

1 to 120. This lack of improvement results from the extra computational costs of using OpenMP. However, the hybrid parallel programming dramatically improves parallel performance when scaling the number of CPU cores from 240 to 960. For example, the hybrid parallel programming achieves a speedup of 1.56 relative to the MPI-only baseline when GAMIL1° is run on 960 cores. According to the red bar and purple bar, we find that the optimizations of parallel communication and cascading solution further dramatically improve the parallel performance, especially when the CPU core number is large. With all of the three parallelization strategies, we achieve a 2.10-fold speedup when using 960 CPU cores.

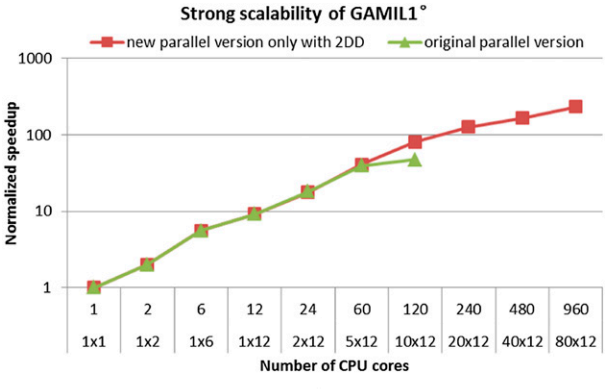
### e. *Overall efficiency of parallelization*

We now evaluate the overall parallel efficiency of our proposed parallel optimization strategies. This quantity is computed according to Eq. (2), where $E_P$ is the parallel efficiency, $T_B$ is the baseline execution time, $C_B$ is the number of cores used for the baseline execution, $T_P$ is the parallel execution time, and $C_P$ is the number of cores used for the parallel execution. Figure 11 shows the overall, physical package, and dynamic core parallel efficiency of GAMIL1° when the number of CPU cores is gradually increased from 1 to 960. The physical package achieves high parallel efficiency because it is computationally intensive. The dynamical core achieves much lower parallel efficiency because it is memory intensive and communication intensive. The overall parallel efficiency falls between the parallel efficiencies of the physical package and the dynamical core. When using 960 CPU cores, GAMIL1° achieves a parallel efficiency of 37.8% relative to the single-core (serial run) baseline. This parallel efficiency is relatively high because GAMIL1° has only 57 960 horizontal grid points, resulting in low computational cost on each CPU core:

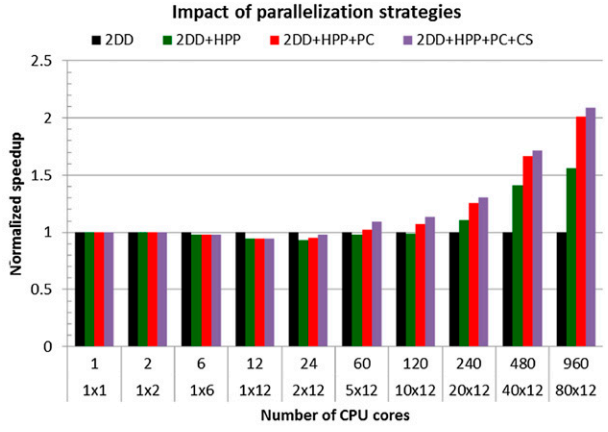
$$E_P = (T_B C_B)/(T_P C_P). \tag{2}$$

The $x$ axis in Fig. 11 can be partitioned into three segments according to the change in parallel efficiency with respect to core number. The first segment corresponds to CPU core numbers between 1 and 12 (i.e., only one computing node). The dynamical core and overall parallel efficiency decrease rapidly as the number of CPUs increases from 1 to 12 because the dynamical core is memory intensive and the total memory bandwidth is constant (only one computing node is used). The second segment corresponds to CPU core numbers between 12 and 120. The parallel efficiency degrades more slowly along this segment because the total memory bandwidth increases as the number of computing nodes increases. The third segment corresponds to CPU core numbers between 120 and 960. The parallel efficiency degrades more rapidly along this segment. This rapid decrease in parallel efficiency reflects a rapid increase in communication overhead, which becomes an increasingly significant contributor to the overall execution time as the number of computing nodes increases.

Figure 12 shows the parallel efficiency of GAMIL0.25° when the number of CPU cores is gradually increased from 4 to 960. GAMIL0.25° achieves a much higher parallel efficiency than GAMIL1° for the same number of CPU cores. This is because GAMIL0.25° has a greater number of horizontal grid points than GAMIL1°, which




a




b

FIG. 10. Impact of each parallelization strategy in GAMIL1°. The *x* axis here is as in Fig. 8. (a) The performance speedup (*y* axis) achieved using HPP (green), PC (red), and CS for tridiagonal equations (purple) parallelization strategies. The baseline is the parallel version of GAMIL using only the hybrid 2DD (black) parallelization strategy. (b) The scalability of the new parallel version when using 2DD alone, without implementing other parallelization strategies.

increases the number of computations on each CPU core. GAMIL0.25° achieves a parallel efficiency of 57.5% when run on 960 cores.

Figures 12 and 13 evaluate the strong scalability of the new parallel version on two different horizontal resolutions. We can discuss the weak scalability by combining these two figures. The number of grid points in the 0.25° grid is about 16 times the number of grid points in the 1° grid. It is therefore reasonable to compare the GAMIL0.25° run on 960 CPU cores with the GAMIL1° run on 60 CPU cores. GAMIL1° achieves parallel efficiencies of 93.7% for the physical package, 46.5% for the dynamical core, and 67.2% overall when run on 60 CPU cores. GAMIL0.25° achieves parallel efficiencies of 94.2% for the physical package, 39.8% for the dynamical core, and 57.5% overall. The parallel efficiency

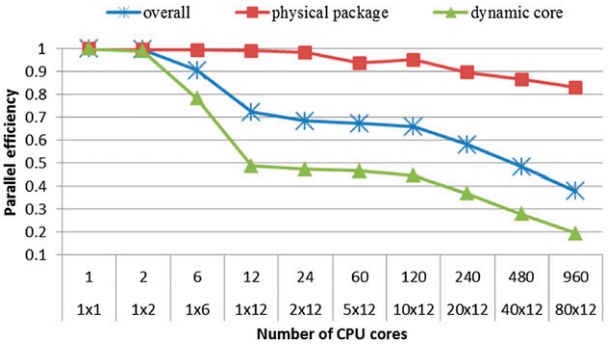



FIG. 11. The overall (blue), physical package (red), and dynamical core (green) parallel efficiencies (*y* axis) for GAMIL1° using different numbers of CPU cores (*x* axis). The *x* axis here is as in Fig. 8. When using one CPU core (a serial run), the physical package and dynamical core account for 61.7% and 33.6% of the overall execution time, respectively.

of the physical package is not decreased because it is computationally intensive and the hybrid 2D decomposition ensures load balance. The parallel efficiency of the dynamical core decreases because of the presence of collective communications within the dynamical core. During each time step of the dynamical core, all processes are involved in four collective communications and processes involving polar grid points are involved in 15 collective communications. The overall parallel efficiency of GAMIL is decreased from 67.2% to 57.5%. This decrease is larger than the decrease in the parallel efficiency of the dynamical core because the computational cost (measured using a sequential run) of the physical package and the land surface package is a smaller proportion of the overall cost in GAMIL0.25° than in GAMIL1°. The increase in the portion of the total computational cost attributable to the dynamical core means that the overall parallel efficiency approaches the parallel efficiency of the dynamical core. For example, increasing the horizontal resolution from 1° to 0.25° increases the number of grid points by a factor of 16. The computational cost of the dynamical core also increases by a factor of 16, while the computational costs of the physical package and the land surface package increase by factors of 9.95 and 12.27, respectively.

### *f. Comparison with parallel CAM-SE*

Finally, we evaluate the new parallel version of GAMIL against the parallel version of CAM-SE, which is a state-of-the-art AGCM. These two models have different dynamical cores. Figure 13 shows the parallel efficiency of CAM-SE1°. CAM-SE1° contains $30 \times 30 \times 6 \times 4 \times 4 = 86\,400$ horizontal grid points and 30 vertical levels, more than GAMIL1° ($360 \times 161 = 57\,960$

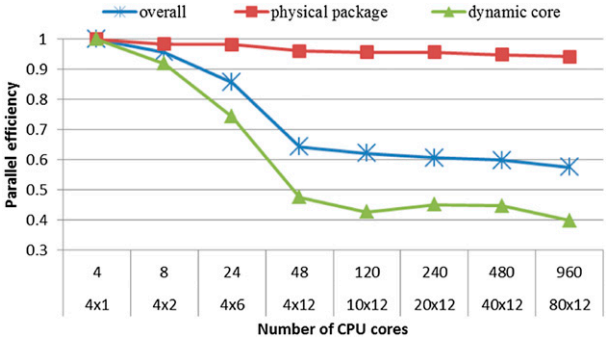



FIG. 12. As in Fig. 11, but for GAMIL0.25°. When using $4 \times 1$ CPU cores, the physical package and dynamical core account for 51.5% and 41.1% of the overall execution time, respectively.

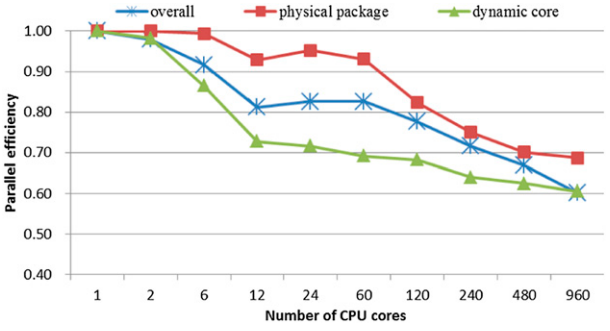



FIG. 13. As in Fig. 11, but for CAM-SE1°. When using one CPU core (a serial run), the physical package and dynamical core account for 47.5% and 51.5% of the overall execution time, respectively.

horizontal grid points and 26 vertical levels). The CAM-SE code and related data have been obtained from the National Center for Atmospheric Research (NCAR) Community Earth System Model (CESM) website (http://www2.cesm.ucar.edu/models/current). We run CAM-SE using hybrid MPI and OpenMP parallelization. A comparison of Figs. 13 and 11 reveals that CAM-SE1° achieves better overall parallel efficiency than GAMIL1° for the same number of CPU cores, even though the parallel efficiency of the physical package is lower. This is because the spectral element method used in the CAM-SE1° dynamical core achieves much better parallel efficiency than the finite-difference method used in GAMIL1°. The GAMIL dynamical core is more communication intensive because of its lower computational overhead and greater number of collective communications per time step. The lower computational overhead of the GAMIL dynamical core is attributable to the lower horizontal and vertical resolutions and the reduced computational complexity of the finite-difference method. The increase in the number of collective communications results from the treatment of the poles and the mechanism by which energy conservation is maintained. For example, the computational overhead per time step is 1.70 s for the GAMIL1° dynamical core and 28.44 s for the CAM-SE1° dynamical core (measured using a sequential execution). By contrast, the average total point-to-point communication overhead when using 960 CPU cores (including all point-to-point communication in all processes) is 2.43 s in GAMIL1° and 4.50 s in CAM-SE1°, on average. Moreover, the GAMIL dynamical core contains four collective communications for all processes and 15 collective communications for processes including polar grid points, while the CAM-SE dynamical core only contains one collective communication for all processes.

## 5. Conclusions

In this paper, we have endeavored to improve the parallel performance of GAMIL, a finite-difference AGCM. The original parallel version of GAMIL, which used a 1D parallel decomposition and MPI-only parallel programming, could only utilize a small number of CPU cores with low efficiency. These limitations restrict the development of higher-resolution versions of the model. We have therefore implemented a new parallel version using multiple parallelization strategies, including hybrid 2D decomposition, hybrid parallel programming with MPI and OpenMP, parallel communication, and a cascading solution to the tridiagonal equations in the dynamical core. Evaluation of the new parallel version demonstrates that these parallelization strategies enable GAMIL to more effectively utilize a greater number of CPU cores.

*Acknowledgments.* This work is partly supported by the Natural Science Foundation of China (41275098).

REFERENCES

Bellenger, H., E. Guilyardi, J. Leloup, M. Lengaigne, and J. Vialard, 2013: ENSO representation in climate models: From CMIP3 to CMIP5. *Climate Dyn.,* **42,** 1999–2018, doi:10.1007/s00382-013-1783-z.

Dennis, J., and Coauthors, 2012: CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model. *Int. J. High-Perform. Comput. Appl.,* **26,** 74–89, doi:10.1177/1094342011428142.

Dong, L., L.-J. Li, W.-Y. Huang, Y. Wang, and B. Wang, 2012: Preliminary evaluation of the cloud fraction simulations by GAMIL2 using COSP. *Atmos. Oceanic Sci. Lett.,* **5,** 258–263.

Guo, Z., C. Wu, T. Zhou, and T. Wu, 2011: A comparison of cloud radiative forcings simulated by LASG/IAP and BCC atmospheric general circulation models (in Chinese). *Chin. J. Atmos. Sci.,* **35,** 739–752.

Hodson, D. L. R., R. T. Sutton, C. Cassou, N. Keenlyside, Y. Okumura, and T. Zhou, 2010: Climate impacts of recent

multidecadal changes in Atlantic Ocean sea surface temperature: A multimodel comparison. *Climate Dyn.,* **34,** 1041–1058, doi:10.1007/s00382-009-0571-2.

Khairoutdinov, M. F., and D. A. Randall, 2001: A cloud resolving model as a cloud parameterization in the NCAR Community Climate System Model: Preliminary results. *Geophys. Res. Lett.,* **28,** 3617–3620, doi:10.1029/2001GL013552.

Kuang, X. Y., Y. C. Zhang, J. Liu, and L. L. Guo, 2009: A numerical study of the effect of anomalous surface heating in the Kuroshio current region in winter on the East Asian subtropical westerly jet (in Chinese). *Chin. J. Atmos. Sci.,* **33,** 81–89.

Kucharski, F., and Coauthors, 2009: The CLIVAR C20C project: Skill of simulating Indian monsoon rainfall on interannual to decadal timescales. Does GHG forcing play a role? *Climate Dyn.,* **33,** 615–627, doi:10.1007/s00382-008-0462-y.

Lauritzen, P. H., C. Jablonowski, M. A. Taylor, and R. D. Nair, Eds., 2011: *Numerical Techniques for Global Atmospheric Models.* Lecture Notes in Computational Science and Engineering, Vol. 80, Springer, 556 pp.

Li, L. J., and B. Wang, 2010: Influences of two convective schemes on the radiative energy budget in GAMIL1.0. *Acta Meteor. Sin.,* **24,** 318–327.

——, ——, Y. Q. Wang, and H. Wan, 2007a: Improvements in climate simulation with modifications to the Tiedtke convective parameterization in the Grid-point Atmospheric Model of IAP LASG (GAMIL). *Adv. Atmos. Sci.,* **24,** 323–335, doi:10.1007/s00376-007-0323-3.

——, ——, and T. J. Zhou, 2007b: Impacts of external forcing on the 20th century global warming. *Chin. Sci. Bull.,* **52,** 3148–3154, doi:10.1007/s11434-007-0463-y.

——, and Coauthors, 2013a: Evaluation of grid-point atmospheric model of IAP LASG version 2 (GAMIL2). *Adv. Atmos. Sci.,* **30,** 855–867, doi:10.1007/s00376-013-2157-5.

——, and Coauthors, 2013b: The Flexible Global Ocean-Atmosphere-Land System Model: Grid-point version 2: FGOALS-g2. *Adv. Atmos. Sci.,* **30,** 543–560, doi:10.1007/s00376-012-2140-6.

Lin, J. L., 2007: The double-ITCZ problem in IPCC AR4 coupled GCMs: Ocean–atmosphere feedback analysis. *J. Climate,* **20,** 4497–4525, doi:10.1175/JCLI4272.1.

Mao, J.-Y., and L.-J. Li, 2012: An assessment of MJO and tropical waves simulated by different versions of the GAMIL model. *Atmos. Oceanic Sci. Lett.,* **5,** 26–31.

Michalakes, J., J. Dudhia, D. Gill, T. B. Henderson, J. B. Klemp, W. Skamarock, and W. Wang, 2004: The Weather Research and Forecast Model: Software architecture and performance. *Proc. 11th Workshop on the Use of High Performance Computing in Meteorology,* Reading, United Kingdom, ECMWF, 156–168.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, 2007: *Numerical Recipes: The Art of Scientific Computing*. 3rd ed. Cambridge University Press, 1256 pp.

Sato, T., 2004: The earth simulator: Roles and impacts. *Parallel Comput.,* **30,** 1279–1286, doi:10.1016/j.parco.2004.09.003.

Scaife, A. A., and Coauthors, 2009: The CLIVAR C20C project: Selected twentieth century climate events. *Climate Dyn.,* **33,** 603–614, doi:10.1007/s00382-008-0451-1.

Schaffer, D. S., and M. J. Suarez, 2000: Design and performance analysis of a massively parallel atmospheric general circulation model. *Sci. Comput.,* **8,** 49–57.

Thomas, L. H., 1949: Elliptic problems in linear differential equations over a network. Watson Scientific Computing Laboratory Rep., Columbia University.

Vertenstein, M., K. Oleson, S. Levis, and F. Hoffman, 2004: Community Land Model Version 3.0 (CLM3.0) user's guide. NCAR, 38 pp. [Available online at http://www.cgd.ucar.edu/tss/clm/distribution/clm3.0/UsersGuide/UsersGuide.pdf.]

Wang, B., H. Wan, Z. Ji, X. Zhang, R. Yu, Y. Yu, and H. Liu, 2004: Design of a new dynamical core for global atmospheric models based on some efficient numerical methods. *Sci. China,* **47A,** 4–21, doi:10.1360/04za0001.

Wu, Z. W., and J. Li, 2008: Prediction of the Asian-Australian monsoon interannual variations with the grid-point atmospheric model of IAP LASG (GAMIL). *Adv. Atmos. Sci.,* **25,** 387–394, doi:10.1007/s00376-008-0387-8.

Xie, X., B. Wang, L.-J. Li, and L. Dong, 2012: MJO simulations by GAMIL1.0 and GAMIL2.0. *Atmos. Oceanic Sci. Lett.,* **5,** 48–54.

Yan, L., P. X. Wang, Y. Q. Yu, L. J. Li, and B. Wang, 2010: Potential predictability of sea surface temperature in a coupled ocean–atmosphere GCM. *Adv. Atmos. Sci.,* **27,** 921–936, doi:10.1007/s00376-009-9062-y.

Yu, R. C., 1994: A two-step shape-preserving advection scheme. *Adv. Atmos. Sci.,* **11,** 479–490, doi:10.1007/BF02658169.

Yu, Y., and D. Z. Sun, 2009: Response of ENSO and the mean state of the tropical Pacific to extratropical cooling and warming: A study using the IAP coupled model. *J. Climate,* **22,** 5902–5917, doi:10.1175/2009JCLI2902.1.

Zheng, W. P., and Y. Q. Yu, 2009: The Asian monsoon system of the middle Holocene simulated by a coupled GCM. *Quat. Sci.,* **29,** 1135–1145.

Zou, L. W., T. J. Zhou, B. Wu, H. M. Chen, and L. J. Li, 2009: The interannual variability of summertime western Pacific subtropical high hindcasted by GAMIL CliPAS experiments (in Chinese). *Chin. J. Atmos. Sci.,* **33,** 959–970.