A Scalable Parallel Algorithm for Atmospheric General Circulation Models on a Multicore Cluster

Yuzhu Wang[1,2], Jinrong Jiang[2,†], He Zhang[3], Xiao Dong[3], Lizhe Wang[4,†], Rajiv Ranjan[4], Albert Y. Zomaya[5]

1 Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, China
2 Computer Network Information Center, Chinese Academy of Sciences, China
3 International Center for Climate and Environment Sciences, Institute of Atmospheric Physics, Chinese Academy of Sciences, China
4 School of Computer Science, China University of Geosciences, China
5 School of Information Technologies, The University of Sydney, Australia
† corresponding authors: Lizhe Wang, lizhe.wang@gmail.com

Abstract

High-performance computing of atmospheric general circulation models (AGCMs) has been receiving increasing attention in earth science research. However, when scaling to large-scale multi-core computing, the parallelization of an AGCM which demands fast parallel computing for long-time integration or climate simulation becomes extremely challenging due to its inner complex numerical calculation. The previous Institute of Atmospheric Physics of the Chinese Academy of Sciences Atmospheric General Circulation Model version 4.0 (IAP AGCM4.0) with one-dimensional domain decomposition can only run on dozens of CPU cores, so the paper proposes a two-dimensional domain decomposition parallel algorithm for it. In the parallel implementation of the IAP AGCM4.0, its dynamical core utilizes a hybrid form of latitude/longitude decomposition and vertical direction/longitude circle direction decomposition. Through experiments on multi-core clusters, we confirmed that our algorithm is efficient and scalable. The parallel efficiency of the IAP AGCM4.0 can reach up to 50.88% on 512 CPU cores, and the IAP AGCM4.0 can be run long-term simulations for climate change research.

Keywords: high performance computing, parallel algorithm, domain decomposition, atmospheric general circulation model.

1. Introduction

Atmospheric general circulation models (AGCMs) are important tools for weather forecasts and climate change research [**?** **?** **?** **?**]. An AGCM is also one of the most important components of an earth system model. Because of the importance of AGCMs for climate research, the Institute of Atmospheric Physics (IAP) of the Chinese Academy of Sciences Atmospheric General Circulation Model (IAP AGCM) has been developed since the 1980s [**?** **?** **?**]. As the atmospheric component, the fourth version of the IAP AGCM (IAP AGCM4.0) [**?**] has been used in the Chinese Academy of Sciences-Earth System Model (CAS-ESM) [**?** **?** **?**].

To conduct climate simulation, AGCMs usually need to be integrated for multiple years or decades [**?**]. Wehner et al. conducted a 27-year simulation using the Community Atmosphere Model version 5.1 (CAM5.1) [**?**]. Nakaegawa et al. used the Meteorological Research Institute AGCM3.1 model to perform 25-year simulations for the present-day and future climate [**?**]. In short, AGCMs usually involve a large amount of calculation and a long computing time. Therefore, AGCMs have to be run on a high-performance computing resource in order to meet the real-time requirements of weather forecasting and climate research. Wehner et al. utilized 7680 processing cores to perform the Atmospheric Model Intercomparison Project simulations on a CRAY XE-6 supercomputer [**?**]. Meanwhile, with the rapid development of supercomputers and high-performance computing technology, AGCMs can have higher resolution [**?**]. Miyamoto et al. used a nonhydrostatic icosahedral atmospheric model to conduct a sub-kilometre global simulation on the K computer [**?**].

Before an AGCM is run on a high-performance computing platform, it is necessary to develop a parallel version of the AGCM. Parallel algorithms of AGCMs were studied by climate scientists and computer scientists. Wehner et al. used a two-dimensional latitude/longitude domain decomposition message-passing strategy to implement the UCLA AGCM in a portable parallel form [**?**]. Mechoso et al. later optimized the parallel UCLA AGCM code, after which the UCLA AGCM was about nine times faster [**?**]. Drake et al. designed a parallel global atmospheric circulation model, PCCM2. During this development, parallel spectral transform, semi-Lagrangian transport, and load balancing algorithms were researched [**?**]. Mirin and Sawyer used a message passing interface (MPI) + OpenMP hybrid paradigm to perform a parallel implementation of a finite-volume dynamical core in the CAM. A one-sided communication technique was utilized in the parallel implementation [**?**]. Similarly, the parallelization of the IAP AGCM4.0 also needed to be studied and implemented.

At first, the IAP AGCM4.0 used one-dimensional domain decomposition, where each subdomain contained all longitude lines but only a subset of latitude lines. Obviously, it is easy

to develop a parallel code for the model by using this method. However, the decomposition strategy limits the maximum number of subdomains and CPU cores which may be exploited. The previous IAP AGCM4.0 can only run on dozens of CPU cores, which is not sufficient to meet the real-time computing demand of climate simulations. Therefore, it is necessary to study more efficient parallel algorithms of the IAP AGCM4.0. To realize this goal, this paper designs and implements a two-dimensional domain decomposition parallel algorithm for the IAP AGCM4.0. The two-dimensional decomposition includes two types, latitude/longitude decomposition and vertical direction/longitude circle direction decomposition, which are both used in the implementation of the parallel algorithm. It is obvious that you must switch from one decomposition to another during the parallel computing of the model. Using the two-dimensional domain decomposition strategy, the global domain is decomposed into more subdomains, which are assigned to each process (or MPI rank). Hence, the IAP AGCM4.0 can be processed by more than one thousand processes. Because the IAP AGCM4.0 uses the physics package of CAM3.1, the focus of our research is mainly on the parallel implementation of the dynamical core. Based on a 61-day climate experiment, we evaluate the parallel performance of the IAP AGCM4.0. The results indicate that the IAP AGCM4.0 scales reasonably to 3120 CPU cores and has a desirable parallel performance.

The rest of this paper is organized as follows. The following section introduces the IAP AGCM4.0 model and its dynamical core. In section **??**, we go into detail about the design and implementation of the parallel algorithm with two-dimensional domain decomposition. Section **??** discusses the experimental analysis of the parallel algorithm performance, and the last section contains a summary.

2. Model description

2.1. IAP AGCM4.0 model

An AGCM usually consists of the "dynamics" (dynamical core) and the "physics" (physical parameterizations). The dynamical core calculates the atmospheric flow and solves the hydrodynamic equations of the atmosphere. Then the physical parameterizations for sub-grid phenomena such as long- and short-wave radiation, moist process and gravity wave drag, are approximated [**?**]. The dynamical core and physical parameterizations are expressed according to the following formula:

$$\frac{\partial \psi}{\partial t} = D(\psi) + P(\psi), \tag{1}$$

where ψ is forecast variables such as temperature, surface pressure, horizontal wind field and specific humidity, D is the tendency generated by the dynamical core and P is the tendency generated by the physical parameterizations. The tendencies generated by the dynamical core and the physical parameterizations are added to derive an overall tendency of ψ.

The IAP AGCM was developed by a group of atmospheric scientists from the IAP. After decades of research and development, the IAP AGCM has already shown a considerable capacity for simulating climatic changes. The IAP AGCM4.0 uses the CAM3.1 physics package, while its dynamical core was developed independently by the IAP. The IAP AGCM4.0 uses a finite-difference scheme with a $1.4^\circ \times 1.4^\circ$ or $0.5^\circ \times 0.5^\circ$ horizontal resolution and 26 levels in the vertical direction.

2.2. Dynamical core

The IAP AGCM4.0 uses a terrain-following σ coordinate [**?**] in the vertical direction, so there are the following variable definitions,

$$\sigma \equiv \frac{p - p_t}{p_{es}}, p_{es} = p_s - p_t, \tag{2}$$

where p is the pressure, p_s is the surface pressure and $p_t = 2.194$ hPa is the pressure at the model's top layer. The equations of the dynamical core with the subtraction of standard stratification and the IAP transformation are defined as follows

In the Eqs. (**??**), t is the time, $\delta_p = p_t/p$, $f^* = 2\Omega\cos\theta + u\cot\theta/a$, Ω is the Earth's rotation angular velocity, θ is the colatitude, a as the Earth's radius, λ is the longitude, $b = 87.8$ m/s, $p_0 = 1000$ hPa, $\kappa = R/C_p$, R is the gas constant for dry air, and C_p is the specific heat of dry air at constant pressure. Here, $\delta = 0$ represents the standard stratification approximation. If it is set to 1, the set of equations becomes the same as the primitive equations that are commonly used. $\tilde{T}(p)$ is the standard atmospheric temperature, T is the temperature, $T'(\theta, \lambda, p, t) = T(\theta, \lambda, p, t) - \tilde{T}(p)$. The IAP transformation parameter $P \equiv \sqrt{p_{es}/p_0}$, $(U, V, \Phi) \equiv (Pu, Pv, PRT'/b)$, U is zonal wind velocity, V is meridional wind velocity, Φ is geopotential field. κ^* is the indication coefficient. The horizontal advection operators L_1, L_2 and the vertical convection operator L_3 are calculated according to the following formulas:

$$\begin{cases} L_1(F) \equiv \dfrac{1}{2a\sin\theta}\left(2\dfrac{\partial Fu}{\partial\lambda} - F\dfrac{\partial u}{\partial\lambda}\right) \\ L_2(F) \equiv \dfrac{1}{2a\sin\theta}\left(2\dfrac{\partial Fv\sin\theta}{\partial\theta} - F\dfrac{\partial v\sin\theta}{\partial\lambda}\right) \\ L_3(F) \equiv \dfrac{1}{2}\left(2\dfrac{\partial F\dot{\sigma}}{\partial\sigma} - F\dfrac{\partial\dot{\sigma}}{\partial\sigma}\right) \end{cases}, \tag{4}$$

where F is the diffusion operator, the vertical velocity on the σ level is $\dot{\sigma}$.

The pressure gradient terms are defined by

$$\begin{cases} P_\theta^{(1)} \equiv P\dfrac{\partial\phi'}{a\partial\theta} \\ P_\theta^{(2)} \equiv \dfrac{b\Phi(1-\delta_p)}{p_{es}}\cdot\dfrac{\partial p_{es}}{a\partial\theta} \\ P_\lambda^{(1)} \equiv P\dfrac{\partial\phi'}{a\sin\theta\partial\lambda} \\ P_\lambda^{(2)} \equiv \dfrac{b\Phi(1-\delta_p)}{p_{es}}\cdot\dfrac{\partial p_{es}}{a\sin\theta\partial\lambda} \end{cases}, \tag{5}$$

where $\phi'(\theta, \lambda, p, t) = \phi(\theta, \lambda, p, t) - \tilde{\phi}(p)$, $\phi = gz$ is the geopotential, g is the gravity acceleration, z is the height, $\tilde{\phi}(p)$ is the standard atmospheric geopotential.

The terms Ω and D are determined by

$$\begin{cases} \dfrac{\partial U}{\partial t} = -\sum\limits_{m=1}^{3} L_m(U) - P_\lambda^{(1)} - P_\lambda^{(2)} - f^* V \\ \dfrac{\partial V}{\partial t} = -\sum\limits_{m=1}^{3} L_m(V) - P_\theta^{(1)} - P_\theta^{(2)} + f^* U \\ \dfrac{\partial \Phi}{\partial t} = -\sum\limits_{m=1}^{3} L_m(\Phi) + (1-\delta_p)[b(1+\delta_c) + \delta \cdot \kappa\Phi/P](\Omega^{(1)} + \Omega_\theta^{(2)} + \Omega_\lambda^{(2)}) \\ \dfrac{\partial}{\partial t}\left(\dfrac{p'_{sa}}{p_0}\right) + D(P) + \dfrac{\partial P^2 \dot{\sigma}}{\partial \sigma} = \kappa^* D_{sa}/p_0 \end{cases} . \tag{3}$$

$$\begin{cases} \Omega^{(1)} \equiv \dfrac{P\dot{\sigma}}{\sigma} - \dfrac{1}{P}\left[D(P) + \dfrac{\partial P^2 \dot{\sigma}}{\partial \sigma}\right] \\ \Omega_\theta^{(2)} \equiv \dfrac{V}{p_{es}} \cdot \dfrac{\partial p_{es}}{a\partial\theta} \\ \Omega_\lambda^{(2)} \equiv \dfrac{U}{p_{es}} \cdot \dfrac{\partial p_{es}}{a\sin\theta\partial\lambda} \end{cases} , \tag{6}$$

$$\begin{cases} D(P) \equiv \dfrac{1}{a\sin\theta}\left(\dfrac{\partial PU}{\partial\lambda} + \dfrac{\partial PV\sin\theta}{\partial\theta}\right) \\ D_{sa} \equiv \dfrac{1}{a\sin\theta}\left(\dfrac{\partial \tilde{p}_{sa} k_{sa} D_{s\lambda}}{\partial\lambda} + \dfrac{\partial \tilde{p}_{sa} k_{sa} D_{s\theta}\sin\theta}{\partial\theta}\right) \end{cases} , \tag{7}$$

where

$$\begin{cases} D_{s\lambda} \equiv \dfrac{1}{a\sin\theta} \cdot \dfrac{\partial}{\partial\lambda}\left(\dfrac{p'_{sa}}{\tilde{\rho}_{sa}}\right) \\ D_{s\theta} \equiv \dfrac{1}{a} \cdot \dfrac{\partial}{\partial\theta}\left(\dfrac{p'_{sa}}{\tilde{\rho}_{sa}}\right) \end{cases} , \tag{8}$$

the standard atmosphere density at the surface $\tilde{p}_{sa} = \tilde{p}_s/R\tilde{T}_s$, $\tilde{p}_s$ is the standard surface pressure, $\tilde{T}_s$ is the standard surface temperature, and the dissipation coefficient $k_{sa} = 0.1$. The equations of the dynamical core are described in detail in [**?**].

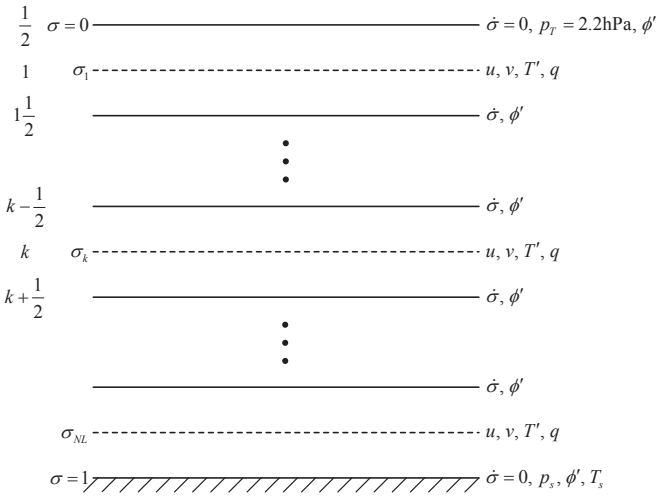

Fig. 1. Vertical distribution of variables in the IAP AGCM4.0.

3. Parallel algorithm

3.1. Spatial discretization scheme and time-integration algorithm

The IAP AGCM4.0 uses the implicit finite difference discretization scheme; its vertical distribution of variables is shown in Fig. **??**. The forecast variables such as U, V, and Φ are put on the integer layer (model layer), while the diagnostic variables are put on the semi-integer layer (interface layer). The IAP AGCM4.0 utilizes Arakawa's C grid staggering [**?**] in its horizontal discretization. Fig. **??** illustrates the horizontal distribution of the variables in the IAP AGCM4.0.

The differential form of Eqs. (**??**) under the standard stratification approximation of $\delta = 0$ and without considering energy dissipation ($\kappa^* = 0$) is written as

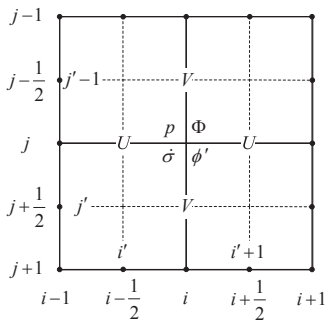

Fig. 2. Horizontal distribution of variables in the IAP AGCM4.0.

In the Eqs. (**??**), vertical velocity $W = P\dot{\sigma}$, α^*, β^* and γ^* are the flexible coefficients. Regardless of what values they are

$$
\begin{cases}
\left[\dfrac{\partial U}{\partial t}\right]_{i',j,k} = \left[-\sum\limits_{m=1}^{3}\alpha^*_{1m}L_m(U) - \beta^*_1 P^{(1)}_{\lambda} - \beta^*_{22}P^{(2)}_{\lambda} - \gamma^* f^* V\right]_{i',j,k}, \quad j \in [2, J-1] \\
\left[\dfrac{\partial V}{\partial t}\right]_{i,j',k} = \left[-\sum\limits_{m=1}^{3}\alpha^*_{2m}L_m(V) - \beta^*_1 P^{(1)}_{\theta} - \beta^*_{21}P^{(2)}_{\theta} + \gamma^* f^* U\right]_{i,j',k}, \quad j \in [1, J-1] \\
\left[\dfrac{\partial \Phi}{\partial t}\right]_{i,j,k} = \left[-\sum\limits_{m=1}^{3}\alpha^*_{3m}L_m(\Phi) + (1-\delta_p)[b(1+\delta_c) + \delta\cdot\kappa\Phi/P]\cdot(\beta^*_1\Omega^{(1)} + \beta^*_{21}\Omega^{(2)}_{\theta} + \beta^*_{22}\Omega^{(2)}_{\lambda})\right]_{i,j,k}, \quad j \in [1, J] \\
\left[\dfrac{\partial}{\partial t}\left(\dfrac{p'_{sa}}{p_0}\right)\right]_{i,j} = -\left[\beta^*_1 D(P) + \beta^*_1\dfrac{P(W_\sigma)_k}{\Delta\sigma_k} - \kappa^* D_{sa}/p_0\right]_{i,j,k}, \quad j \in [1, J]
\end{cases}
. \tag{9}
$$

assigned, the Eqs. (**??**) are conserved. By assigning different values to α^*, β^* and γ^*, it is convenient to conduct numerical experiments and design decomposition algorithms.

A nonlinear iterative time integration method is used in the model. For the sake of simplicity, Eqs. (**??**) can be written as

$$
\frac{\partial F}{\partial t} + A(F) = 0, \quad F = (U, V, \Phi, p'_{sa}), \tag{10}
$$

where A is a nonlinear operator. The following formulas represent the integration from time n to time $n+1$.

$$
\begin{cases}
F^{n+1}_{(1)} = F^n + \Delta t A(F^n) \\
F^{n+1}_{(2)} = F^n + \Delta t A(F^{n+1}_{(1)}) \\
F^{n+1}_{(3)} = F^n + \Delta t A\left[\dfrac{F^{n+1}_{(2)} + F^n}{2}\right] \\
\qquad\vdots \\
F^{n+1}_{(2m)} = F^n + \Delta t A(F^{n+1}_{(2m-1)}) \\
F^{n+1}_{(2m+1)} = F^n + \Delta t A\left[\dfrac{F^{n+1}_{(2m)} + F^n}{2}\right]
\end{cases}
, \tag{11}
$$

where $m = 1, 2, 3, \cdots$. In the IAP AGCM4.0, the number of iterative steps is 3 [**?**].

Using Eqs. (**??**) with the differential form and the time integration algorithm, we can conduct numerical computing and design parallel algorithms for the IAP AGCM4.0.

3.2. Two-dimensional domain decomposition

When running the IAP AGCM4.0 with a $1.4^\circ \times 1.4^\circ$ horizontal resolution in series, the computation time of all its components is indicated in Fig. **??**. The physical parameterizations account for 56.48% of all the computation time, and the dynamic core accounts for 38.02%. Therefore, the physical parameterizations and dynamic core take most of the total computation time in the IAP AGCM4.0. By analysing their computing characteristics, we design a two-dimensional domain decomposition parallel algorithm for them.

The computation of the physical parameterizations in the IAP AGCM4.0 features characteristics of a vertical single column model. This means that the computation of the physical parameterizations on every grid point needs to use related data on the grid points in the vertical direction k and has to be done in sequence, but it does not need related data on the grid points in the horizontal direction. Therefore, the computation task of the physical parameterizations can be decomposed in the horizontal direction. In other words, the global domain is decomposed by latitude and longitude, as shown in Fig. **??**. Because there is no data dependence in the entire computation, there exists no data communication among processors in the physical parameterizations.

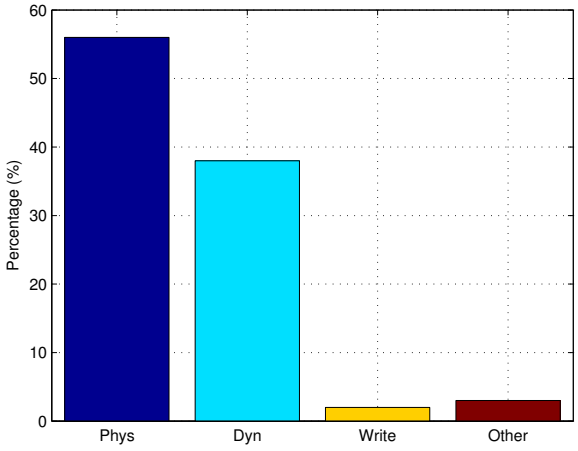

Fig. 3. Statistics about serial computing time of each component of the IAP AGCM4.0.

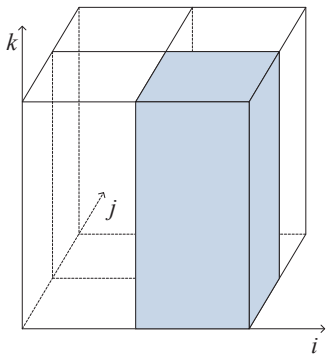

Fig. 4. Two-dimensional domain decomposition in the horizontal direction.

The computation of the dynamical core on each grid point needs to use not only related data on the grid points in the ver-

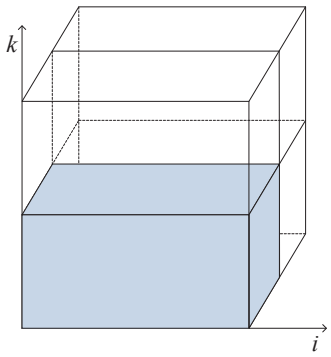

Fig. 5. Two-dimensional domain decomposition in the vertical direction.

tical direction k and the direction j of the longitudinal circle, but also related data on the grid points in the direction i of the latitudinal circle. Meanwhile, some computation in the directions i and k needs to be done in sequence. Therefore, when the computation in the direction i needs to be done in sequence, the task of the dynamic core is decomposed in the vertical direction. This means that the global domain is decomposed by latitude and level, as shown in Fig. **??**. When the computation in the direction k needs to be done in sequence, the way that the task is decomposed is shown in Fig. **??**. For example, the subroutine *sltb* in the one-dimensional decomposition version of the IAP AGCM4.0 is designed to drive the semi-Lagrangian transport algorithm on a given latitude slice in the extended data arrays using information from the entire latitudinal extent of the arrays. In the two-dimensional decomposition of the IAP AGCM4.0, the *sltb* is decomposed into two subroutines *sltb1* and *sltb2*. The *sltb1* with vertical direction/longitude circle direction decomposition is used for the horizontal interpolation of scalar fields, and the *sltb2* with latitude/longitude decomposition is used for the vertical interpolation of scalar fields. Obviously, it is necessary to convert the data from one way of decomposition to another during the entire computation of the dynamical core.

In the two-dimensional domain decomposition, the latitude and longitude boundaries of each subdomain are both constrained to have no fewer than three latitude and longitude lines to form a halo zone. The aim of this limitation is to reduce data communication costs, because data communication among subdomains only occurs in the halo zone, not in the whole zone. The same limitation also exists in the fvCAM [**?**].

According to the parallel algorithm above, the latitude and longitude boundaries of each subdomain must have no fewer than three latitude and longitude lines, so the maximum number of processes (or MPI ranks) in the directions of latitudinal circle and longitudinal circle is $\lfloor mi/3 \rfloor$ and $\lfloor mj/3 \rfloor$, respectively, where mi is the number of grid points in the direction of latitudinal circle, mj is the number of grid points in the direction of longitudinal circle. The maximum number of processes in the vertical direction is mk, where mk is the number of grid points in the vertical direction. In the decomposition strategy in Fig. **??**, the maximum number of processes is $\lfloor mi/3 \rfloor \times \lfloor mj/3 \rfloor$. Similarly, in the decomposition strategy in Fig. **??**, the maximum number of processes is $\lfloor mj/3 \rfloor \times mk$. The two decomposition strategies are both utilized in the model, so the maximum number of processes used to run the IAP AGCM4.0 is calculated by the following formula:

$$\lfloor mj/3 \rfloor \times \min(mk, \lfloor mi/3 \rfloor). \tag{12}$$

Therefore, when the horizontal resolution is $1.4^\circ \times 1.4^\circ$, the IAP AGCM4.0 can run on 1092 ($\lfloor 128/3 \rfloor \times \min(26, \lfloor 256/3 \rfloor)$) cores; when the horizontal resolution is $0.5^\circ \times 0.5^\circ$, it can run on 3120 ($\lfloor 361/3 \rfloor \times \min(26, \lfloor 720/3 \rfloor)$) cores. The maximum number P_y of processes in the direction of longitudinal circle can be assigned the value 42 or 120 and the maximum number P_z of processes in the vertical direction can be assigned the value 26. The processors in the physical parameterizations are decomposed by $P_x \times P_y$, where P_x is the number of processes used in the direction of latitudinal circle. The processors in the dynamical core are decomposed by $P_y \times P_z$ or $P_x \times P_y$, and $P_x \times P_y = P_y \times P_z$, so the maximum value of P_x is 26.

3.3. Algorithm implementation

Fig. **??** illustrates the running flow of the IAP AGCM4.0, which mainly consists of the initialization and core phases. The initializations for the parallel decomposition of the grid, data structure, I/O management module and restarting management module are performed at the initialization phase, where the initialization and restarting data is also read. The forward integrals in the model time direction are mainly performed at the core phase.

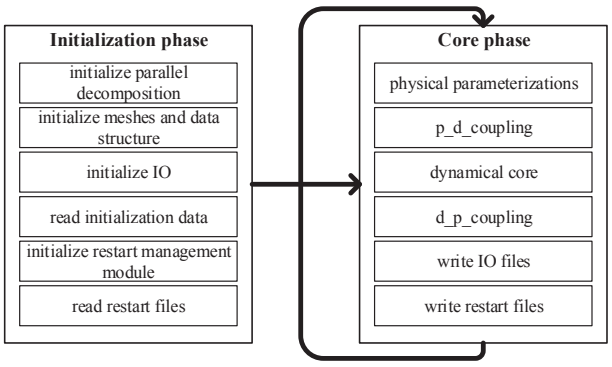

Fig. 6. The whole running flow of the IAP AGCM4.0.

The dynamical core of the IAP AGCM4.0 also consists primarily of the initialization and core phases, as indicated in Fig. **??**. The subroutine *spmd_dyn* in the initializing program is used to decompose the grid of the dynamical core. The subroutine *p_d_coupling* in the core program is used to couple from the physical parameterizations to the dynamical core. In contrast, the subroutine *d_p_coupling* is used to couple from the dynamical core to the physical parameterizations. It is necessary to create a transformation of variables between the physical parameterizations and the dynamical core while running the IAP AGCM4.0. Generally, the variables transmitted include zonal velocity U, meridional velocity V, temperature T, water vapour field Q, vertical velocity W, the pressure at the bottom layer

PS, and the geopotential height *PHIS*. The subroutine *dynpkg*, which mainly consists of the subroutines *dyfram* and *qpdata*, is used to solve dynamic partial differential equations. The call graph of the main subroutines in the IAP AGCM4.0 program is described in Fig. **??**.

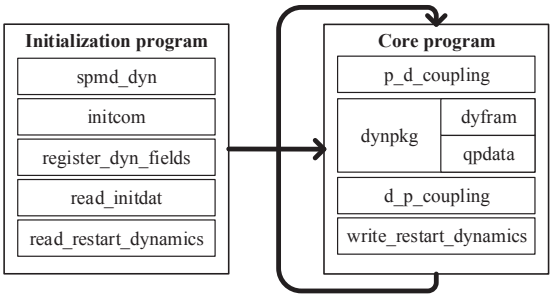

Fig. 7. Running flow of the dynamical core.

In the parallelization of the dynamical core, the PILGRAM library [**?**] is used for the domain decomposition of mesh, and the *mod_comm* library [**?**], which encapsulates MPI_Send and MPI_Recv, is used for data communication among the subdomains. The two libraries were used for the parallelization of the dynamical core in the CAM [**?**]. Similarly, they can be utilized to parallelize other AGCMs.

The implementation of the two-dimensional domain decomposition in Fig. **??** is shown in Algorithm 1, where *npr_y* is the number of subdomains in *y* (the direction of the longitudinal circle), *npr_z* is the number of subdomains in *z* (the vertical direction), *npes* is the total number of MPI tasks, *ydist*(:) is the number of latitudes per subdomain, *nlat_p*(:) is the number of latitudes per subdomain, *cut*(:, :) is the partition for MPI tasks, *plat* is the number of latitudes, *myid_y* is the subdomain index (0-based) in latitude (*y*), *numlats* is the number of latitudes owned by a given MPI rank, *beglatdyn* is the starting latitude for dynamical framework, *endlatdyn* is the ending latitude for dynamical framework, *beglatdynex* is the extended starting latitude for dynamical framework, *endlatdynex* is the extended ending latitude for dynamical framework, and [*loc_JB*, *loc_JE*] of a processor is related to [*JB*, *JE*]. The algorithm implementation of the decomposition in Fig. **??** is similar to Algorithm 1.

In addition, there is Fourier filtering above 70 degrees towards the poles. The FFT99 [**? ?**], a commonly used software package for the Fast Fourier Transform (FFT) in atmospheric numerical models, works for any transform length of the form $N = 2^p \times 3^q \times 5^r (p > 0, q \geqslant 0, r \geqslant 0; p, q, r \in Z)$. The previous IAP AGCM4.0 also used the FFT99. To improve the computational efficiency of the polar filtering in the IAP AGCM4.0, according to the special needs of atmospheric numerical models, we have designed and implemented a new package called SC_FFT, based on the Fastest Fourier Transform in the West (FFTW) library [**?**]. The FFTW takes account of the influence of hardware configuration and FFT transformation parameters, so the SC_FFT is more efficient than the FFT99. Ideal experiments show that the SC_FFT is 2.5 to 3.5 times faster than the FFT99. In the IAP AGCM4.0, it is 39% faster [**?**]. The algo-

Algorithm 1: Implementation of the two-dimensional domain decomposition in the vertical direction

```
//Compute y decomposition
allocate (ydist(npr_y));
allocate (nlat_p(0:npes-1));
allocate (cut(2, 0:npes-1));
ydist(:) = 0;
nlat_p(0:npes-1) = 0;
lat = plat / npr_y;
workleft = plat - lat * npr_y;
if lat < 3 then
    call endrun ('SPMDINIT_DYN: less than 3 latitudes
    per subdomain');
for procid=1, npr_y do
    ydist(procid) = lat;
if workleft /= 0 then
    procids = (npr_y+1) / 2;
    procidn = procids + 1;
    while workleft /= 0 do
        if procids == 1 then
            procids = npr_y;
        ydist(procids) = ydist(procids) + 1;
        workleft = workleft - 1;
        if workleft /= 0 then
            ydist(procidn) = ydist(procidn) + 1;
            workleft = workleft - 1;
        procidn = procidn + 1;
        procids = procids - 1;
//Set the data structures
lat = 0;
for procid=0, npr_y-1 do
    cut(1, procid) = lat+1;
    lat = lat + ydist(procid+1);
    cut(2, procid) = lat;
    nlat_p(procid) = ydist(procid+1);
    if myid_y == procid then
        beglat = cut(1, myid_y);
        endlat = cut(2, myid_y);
        numlats = ydist(procid+1);
        beglatdyn = plat+1-endlat;
        endlatdyn = plat+1-beglat;
        beglatdynex = max(1, beglatdyn - 1);
        endlatdynex = min(plat, endlatdyn + 1);
        loc_JB = max(2, beglatdyn);
        loc_JE = min(plat-1, endlatdyn);
for k = 1, npr_z-1 do
    for j = 0, npr_y-1 do
        procid = j + k*npr_y;
        cut(1, procid) = cut(1, j);
        cut(2, procid) = cut(2, j);
        nlat_p(procid) = nlat_p(j);
//Compute z decomposition
```

The algorithm of the *z* decomposition is similar to that of the *y* decomposition.

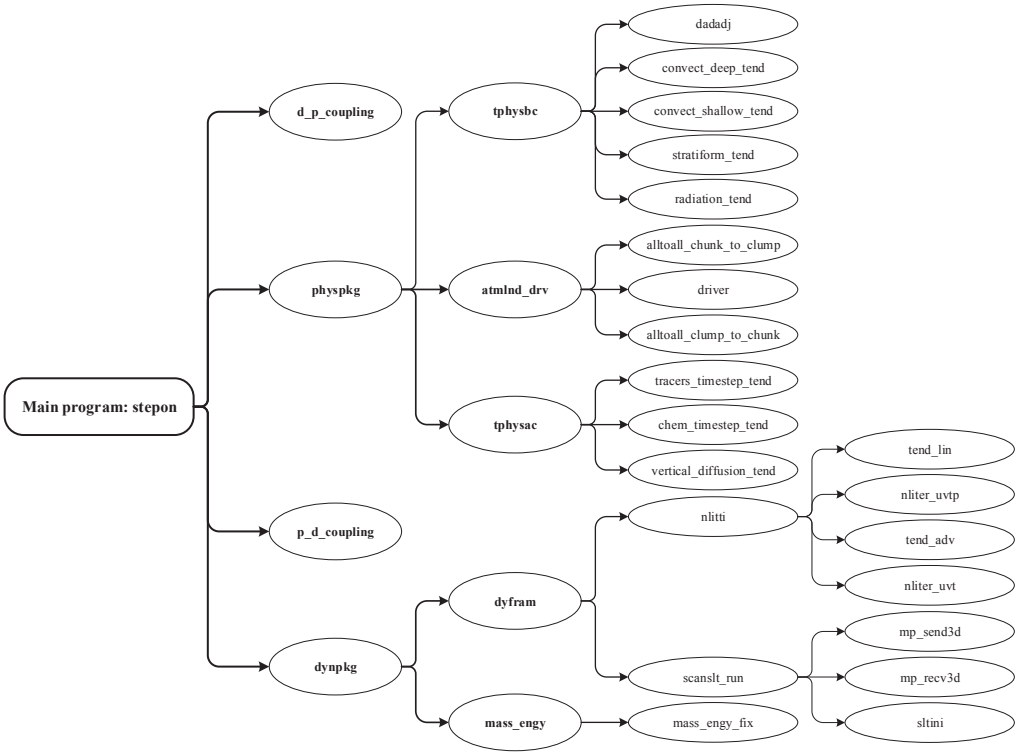

Fig. 8. The call graph of the main subroutines in the IAP AGCM4.0.

rithm of the SC_FFT is as follows.

(1) In the FFTW library, the planner is one of its three components. In a FFT with a length N, the planner factors N in various decomposition ways, or *plans*. Then, the planner uses some measuring methods to determine which *plan* is the fastest [**?**]. Therefore, the *plan* is one parameter of subroutines for fast fourier transform in the FFTW. Once a *plan* is produced, it can be utilized for many times. In atmospheric models, a data sequence may be processed by the FFT for many times. To produce such a *plan* for a data sequence of atmospheric models, we create a subroutine or function called SC_SETPLAN based on the FFTW.

(2) Then, based on the FFTW, the subroutines SC_FFT99 and SC_FFT991 with the same parameters of corresponding subroutines in the FFT99 are created. Here, they can perform a number of simultaneous real/half-complex periodic fourier transforms or corresponding inverse transforms, using ordinary spatial order of grid point values (SC_FFT991) or explicit cyclic continuity in the grid point values (SC_FFT99).

4. Result and discussion

4.1. Experimental setup

To evaluate the parallel performance of the IAP AGCM4.0, an ideal climate simulation experiment for 61 model days is designed. The time step of the IAP AGCM4.0 in the experiment is 600 seconds. The simulating result is output once every month. The initial conditions are from an earlier run of control simulations, and boundary conditions (sea surface temperatures and sea ice concentrations) are from the global Hadley Centre Sea Ice and Sea Surface Temperature (HadISST) dataset [**?**]. In the study, the IAP AGCM4.0 is tested and evaluated at both $1.4^\circ \times 1.4^\circ$ and $0.5^\circ \times 0.5^\circ$ horizontal resolutions.

The experimental platform for the simulation is the Sugon TC4600H blade cluster in the Computer Network Information Center of the Chinese Academy of Sciences, which now has hundreds of compute nodes, each compute node having 20 or 24 CPU cores. The CPU is the Intel Xeon E5-2680 v2 or E5-2680 v3 processor. In each compute node, CPU cores share a 64 or 128 GB DDR3 system memory through QuickPath Interconnect. As the basic compiler in the tests, we used an Intel C/Fortran compiler version 13.1.3 with the optimizing level of O1. For MPI communication routines, we used Intel MPI 4.1.3 implementation binding with the Intel compiler.

4.2. Processors' different ways of decomposition

When the number of processors used is constant, processors' different ways of decomposition have an impact on the parallel performance of a model. In this case, the computing time of the IAP AGCM4.0 ($1.4^\circ \times 1.4^\circ$) is shown in Table **??**. The result indicates that the larger P_y is, the more quickly the IAP AGCM4.0 will run.

4.3. Parallel analysis

We conducted an experiment for the IAP AGCM4.0 on 32, 64, 128, 256 and 512 CPU cores respectively to test its parallel computing performance. In these simulations, we employed the processors' optimal way of decomposition. The computing time and speedup of the IAP AGCM4.0 with increasing the

Table 1
Computing time (s) of the IAP AGCM4.0 with processors' different ways of decomposition.

Cores	IAP AGCM4.0 ($P_x \times P_y$)	Time
32	16×2	6198.94
	8×4	4239.79
	4×8	3525.89
	2×16	2974.48
	1×32	2787.82
64	16×4	3204.28
	8×8	2368.41
	4×16	1756.35
	2×32	1577.49
128	16×8	2186.15
	8×16	1259.12
	4×32	1038.58
256	16×16	1125.78
	8×32	739.07
512	16×32	775.03

number of cores are also plotted in Fig. **??** and Fig. **??**. From the two figures, the study can draw certain conclusions as follows.

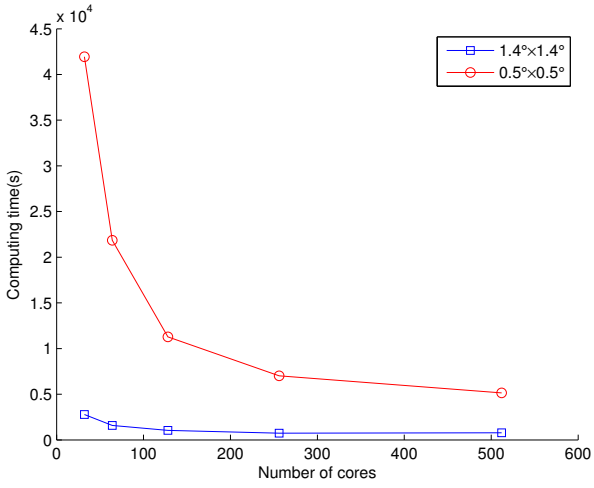

Fig. 9. Computing time of the IAP AGCM4.0.

First, when the resolution is $1.4^\circ \times 1.4^\circ$, the parallelization of the IAP AGCM4.0 with MPI reduces the computing time of 2787.82 seconds on 32 cores to 739.07 seconds on 256 cores, for a speedup of about 3.8x. But the computing time on 512 cores begins to increase. This means that the IAP AGCM4.0 with $1.4^\circ \times 1.4^\circ$ resolution on 256 cores has the fastest computing speed on the Sugon cluster.

Second, when the resolution is $0.5^\circ \times 0.5^\circ$, in comparison with the 32 CPU cores, the speedup of the IAP AGCM4.0 with 50.88% parallel efficiency on 512 CPU cores can reach 8.14x. The computing time on 512 cores still reduces as well. Therefore, the IAP AGCM4.0 with $0.5^\circ \times 0.5^\circ$ resolution has stronger scalability.

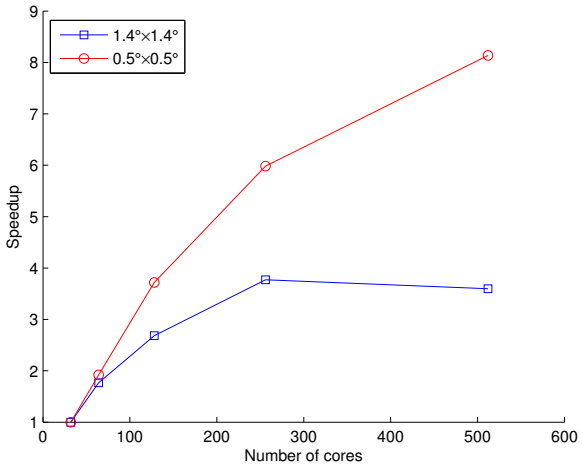

Fig. 10. Speedup of the IAP AGCM4.0.

To discover the reason why the IAP AGCM4.0 with $1.4^\circ \times 1.4^\circ$ resolution slows down on 512 cores or more, we counted the computing time and speedup of both the physical parameterizations and dynamical core, as indicated in Fig. **??** and Fig. **??** respectively. From the curves, we know that the computing time of the physical parameterizations decreases much more quickly than does that of the dynamical core. The physical parameterizations can still speed up on 512 cores, while the dynamical core cannot. There is no data communication during the entire computation, so the physical parameterizations can have a better speedup. Accordingly, it is the dynamical core which results in the decline in the IAP AGCM4.0 computing speed on 512 cores.

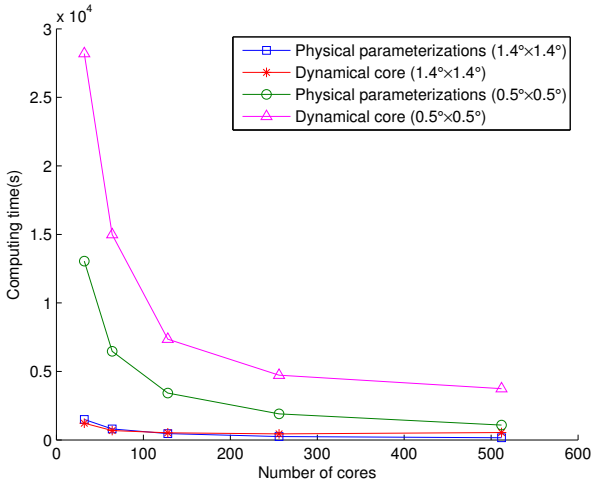

Fig. 11. Computing time of the physical parameterizations and dynamical core.

There are two main reasons why the dynamical core has an unsatisfactory speedup. The first reason is the small meshes (128×256×26) of the IAP AGCM4.0. When running the model on larger-scale cores, the data communication cost significantly increases, and the computational advantage of nodes cannot be well-represented. The second reason is that there are too many small computing tasks in the dynamical core. They are not appropriate for being computed on larger-scale CPU cores.

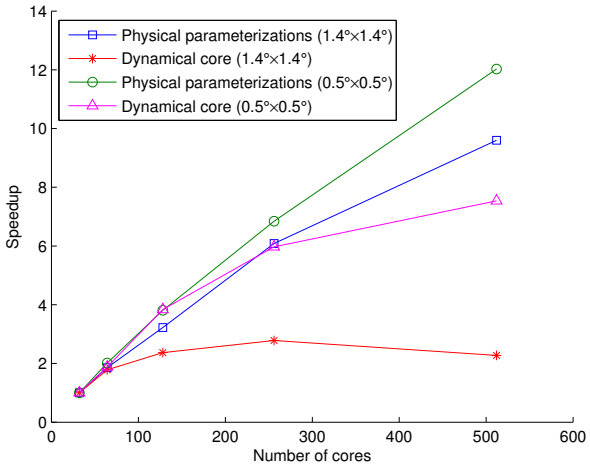

Fig. 12. Speedup of the physical parameterizations and dynamical core.

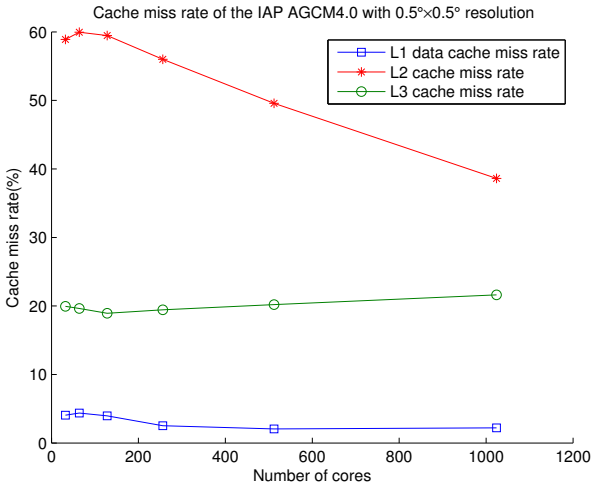

Fig. 13. Cache miss rate of the IAP AGCM4.0 with $0.5^\circ \times 0.5^\circ$ resolution.

Testing and analysing the code of the dynamical core have shown that the subroutines *tend_lin* and *scanslt_run* take most of the total computational time of the dynamical core, as shown in Table **??**. The former is designed to compute the tendencies of P, T, U, and V, while the latter is designed to handle semi-Lagrangian transport in the context of Eulerian spectral dynamics. In the two subroutines, too many *mp_send3d* and *mp_recv3d* operations, which encapsulate MPI_Send and MPI_Recv, are called to transport some computing variables and to communicate boundary information. In the meantime, MPI_Allgatherv is also invoked in the *tend_lin*. In short, as the computing time of the two subroutines on more CPU cores increases, the computing speed of the dynamical core slows down. When the resolution is $0.5^\circ \times 0.5^\circ$, the computing mesh is $361 \times 720 \times 26$. However, the speedup of the dynamical core slows down when the number of cores is more than 512. To further speed up the dynamical core, the two subroutines can continue to be optimized in the future.

To improve fully the performance of the IAP AGCM4.0, the cache miss rate of the parallel algorithm is evaluated by the Performance Application Programming Interface (PAPI), a commonly used performance analysis tool [**?**]. Fig. **??** shows that the L2 cache miss rate of the algorithm ranges from 35 percent to 60 percent, so the algorithm can be optimized on data locality later.

4.4. Long-time integration simulation

To evaluate the performance of the IAP AGCM4.0 with $1.4^\circ \times 1.4^\circ$ horizontal resolution for long-term climate change, a numerical experiment simulating the global surface air temperature (SAT) during the twentieth century is conducted. In the experiment, the observed sea surface temperature and sea ice during the twentieth century are used to drive the IAP AGCM4.0; the observational dataset GISS [**?**] is also used in this study. The simulated and observed time series of the global mean SAT anomaly are shown in Fig. **??**. The correlation coefficient (CC) between the observation and the simulation is 0.89, so the IAP AGCM4.0 can simulate the global warming trend well. This means the model has good performance for long-term climate change. It will take a lot of time to simulate long-term climate change, so it makes sense to run the model in parallel on a cluster. Therefore, the parallel algorithm presented in the paper is valuable for long-time integration simulations of the IAP AGCM4.0.

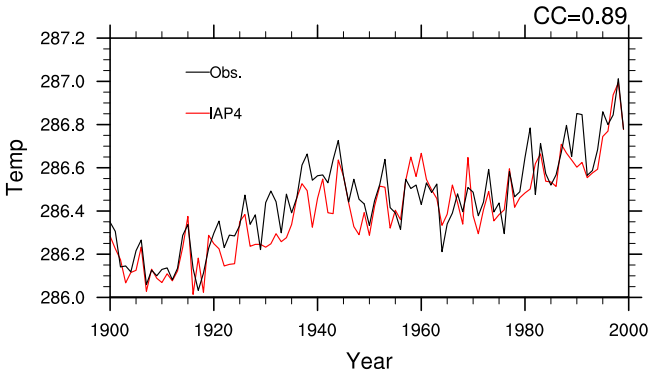

Fig. 14. Time series of the SAT anomaly during the twentieth century for the simulation (red solid line) and observation (black solid line) (units: K).

5. Conclusions and future work

The scalable parallelization of an AGCM turns out to be quite challenging for atmospheric scientists. It needs not only many parallel algorithms of numerical computing but also huge amounts of code implementation. Therefore, computer scientists or software engineers are needed to provide professional help during the design and development of an AGCM. As demonstrated above in this paper, we put forward a two-dimensional domain decomposition parallel algorithm for the IAP AGCM4.0. The algorithm is mainly used in the parallelization of the dynamical core implemented with MPI. After the parallelization, the IAP AGCM4.0 can run on 3120 CPU cores. The speedup of the IAP AGCM4.0 with 50.88% parallel efficiency on 512 CPU cores can reach 8.14x. The experimental

Table 2
Computing time (s) of dynamical core, *tend_lin* and *scanslt_run*.

Resolution	Cores	IAP AGCM4.0 ($P_x \times P_y$)	Dynamical core	*tend_lin*	*scanslt_run*
$1.4^\circ \times 1.4^\circ$	32	1×32	1223.31	403.04	293.16
	64	2×32	685.93	259.84	157.68
	128	4×32	516.16	224.36	95.04
	256	8×32	439.79	190.56	76.41
	512	16×32	537.98	264.98	95.85
$0.5^\circ \times 0.5^\circ$	32	1×32	28195.02	11271.45	7005.48
	64	1×64	14980.62	6520.45	3353.91
	128	2×64	7351.81	3457.66	2003.58
	256	4×64	4723.65	2274.81	1145.48
	512	8×64	3739.79	1889.30	894.67
	1024	16×64	3421.69	1646.44	916.00

result shows that our parallel algorithm is efficient and scalable, and it will show a more desirable parallel performance in the model with higher resolution and with longer-time integration. Our work is meaningful for real-time climate simulation. The parallel algorithm can also be used in the parallelization of other AGCMs, because their parallel ideas are all similar. Although the IAP AGCM4.0 does not have very high parallel efficiency on thousands of cores, we have created a precedent which will promote the development of climate simulation with computing on thousands of cores. We will continue to optimize the IAP AGCM4.0.

The dynamical core has many small computing tasks, so its parallelization can be implemented with OpenMP in the future. In this way, the IAP AGCM4.0, implemented with the MPI + OpenMP hybrid paradigm which exploits two-level parallelism, will run more quickly on a multi-core cluster. The physical parameterizations have quite a good parallel performance, so we can expect to assign more processors to the physical parameterizations in long-time computing with large-scale cores. At present, some physical parameterization schemes of the Weather Research and Forecasting model (WRF) with a Graphics Processing Unit (GPU) version have been implemented [**? ? ? ? ?**]. The IAP AGCM4.0 shares physical parameterization schemes that are similar to the WRF, so running the physical parameterizations on the GPU cores may also be considered. In other words, the GPU version of the physical parameterizations may be developed later. Moreover, a parallel I/O strategy with high data throughput [**?**] for the IAP AGCM4.0 should be also researched. In this way, the IAP AGCM4.0 will be more efficient and scalable in large-scale multi-core computing.

6. Acknowledgement

This work is supported by National Natural Science Foundation of China (No. 61602477, No. 61432018), China Postdoctoral Science Foundation (No. 2016M601158), and National Key Research and Development Program of China (No. 2016YFB0200800).

[1] A.A. Mirin, W.B. Sawyer, A scalable implementation of a finite-volume dynamical core in the community atmosphere model, International Journal of High Performance Computing Applications 19(3)(2005)203–212.

[2] A. Molod, L. Takacs, M. Suarez, J. Bacmeister, Development of the GEOS-5 atmospheric general circulation model: evolution from MERRA to MERRA2, Geoscientific Model Development 8(5)(2015)1339–1356.

[3] M. Satoh, Standard experiments of atmospheric general circulation models, In: Atmospheric Circulation Dynamics and General Circulation Models, Springer Berlin Heidelberg, 2014, pp. 689–702.

[4] T. Nakaegawa, A. Kitoh, Y. Ishizaki, et al., Caribbean low-level jets and accompanying moisture fluxes in a global warming climate projected with CMIP3 multi-model ensemble and fine-mesh atmospheric general circulation models, International Journal of Climatology 34(4)(2014)964–977.

[5] H. Zhang, Development of IAP atmospheric general circulation model version 4.0 and its climate simulations (in Chinese), Institute of Atmospheric Physics, Chinese Academy of Sciences, 2009.

[6] Q. Zeng, X. Zhang, X. Liang, et al., Documentation of IAP two-level atmospheric general circulation model, DOE/ER/60314-H1, TR044, 1989, pp. 383.

[7] X. Liang, Description of a nine-level grid point atmospheric general circulation model, Advances in Atmospheric Sciences 13(3)(1996)269–298.

[8] H. Zhang, M. Zhang, Q. Zeng, Sensitivity of simulated climate to two atmospheric models: Interpretation of differences between dry models and moist models, Monthly Weather Review 141(5)(2013)1558–1576.

[9] H. Sun, G. Zhou, Q. Zeng, Assessments of the climate system model (CAS-ESM-C) using IAP AGCM4 as its atmospheric component, Chinese Journal of Atmospheric Sciences (in Chinese) 36(2)(2012)215–233.

[10] X. Dong, T. Su, J. Wang, R. Lin, Decadal variation of the Aleutian Low-Icelandic Low seesaw simulated by a climate system model (CAS-ESM-C), Atmospheric and Oceanic Science Letters 7(2)(2014)110–114.

[11] Y. Wang, J. Jiang, H. Ye, J. He, A distributed load balancing algorithm for climate big data processing over a multi-core CPU cluster, Concurrency and Computation: Practice and Experience 28(15)(2016)4144–4160.

[12] H. Wan, P.J. Rasch, K. Zhang, et al., Short ensembles: an efficient method for discerning climate-relevant sensitivities in atmospheric general circulation models, Geoscientific Model Development 7(5)(2014)1961–1977.

[13] M.F. Wehner, K.A. Reed, F. Li, et al., The effect of horizontal resolution on simulation quality in the Community Atmospheric Model, CAM5.1, Journal of Advances in Modeling Earth Systems 6(4)(2014)980–997.

[14] S. Watanabe, K. Sato, Y. Kawatani, et al., Vertical resolution dependence of gravity wave momentum flux simulated by an atmospheric general circulation model, Geoscientific Model Development Discussions 8(6)(2015)1637–1644.

[15] Y. Miyamoto, Y. Kajikawa, R. Yoshida, et al., Deep moist atmospheric convection in a subkilometer global simulation, Geophysical Research Letters 40(18)(2013)4922–4926.

[16] M.F. Wehner, J.J. Ambrosiano, J.C. Brown, et al., Toward a high performance distributed memory climate model, In: Proceedings the 2nd International Symposium on High Performance Distributed Computing, 1993,

pp. 102–113.
[17] C.R. Mechoso, L.A. Drummond, J.D. Farrara, J.A. Spahr, The UCLA AGCM in high performance computing environments, In: Proceedings of the 1998 ACM/IEEE conference on Supercomputing, IEEE Computer Society, 1998, pp. 1–7.
[18] J. Drake, I. Foster, J. Michalakes, et al., Design and performance of a scalable parallel community climate model, Parallel Computing 21(10)(1995)1571–1591.
[19] N.A. Phillips, A coordinate system having some special advantages for numerical forecasting, Journal of Meteorology 14(2)(1957)184–185.
[20] A. Arakawa, V.R. Lamb, Computational design of the basic dynamical process of the UCLA general circulation model, Methods in Computational Physics 17(1977)173–265.
[21] M. Wehner, L. Oliker, J. Shalf, Towards ultra-high resolution models of climate and weather, International Journal of High Performance Computing Applications 22(2)(2008)149–165.
[22] W. Sawyer and P. Messmer, Parallel Grid Manipulations for General Circulation Models, In: International Conference on Parallel Processing and Applied Mathematics, Springer Berlin Heidelberg, 2002, pp. 564–571.
[23] W.M. Putman, S.J. Lin, B.W. Shen, Cross-Platform Performance of a Portable Communication Module and the NASA Finite Volume General Circulation Model, International Journal of High Performance Computing Applications 19(3)(2005)213–223.
[24] C. Temperton, A generalized prime factor FFT algorithm for any $n = 2^p 3^q 5^r$, SIAM Journal on Scientific and Statistical Computing 13(3)(1992)676–686.
[25] C. Temperton, A self-sorting in-place prime factor real/half-complex FFT algorithm, Journal of Computational Physics 75(1)(1988)199–216.
[26] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, Proceedings of the IEEE 93(2)(2005)216–231.
[27] Y. Wang, J. Jiang, X. Chi, et al., Design and implementation of FFT algorithm in atmospheric numerical model (in Chinese), Jounal on Numerical Methods and Computer Applications 34(4)(2013)312–322.
[28] R. Vuduc, J.W. Demmel, Code generators for automatic tuning of numerical kernels: Experiences with FFTW, In: International Workshop on Semantics, Applications, and Implementation of Program Generation, Springer Berlin Heidelberg, 2000, pp. 190–211.
[29] N.A. Rayner, D.E. Parker, E.B. Horton, et al., Global analyses of sea surface temperature, sea ice, and night marine air temperature since the late nineteenth century, Journal of Geophysical Research: Atmospheres 108(D14)(2003).
[30] PAPI: Performance Application Programming Interface: User's Guide, http://icl.cs.utk.edu/papi, 2005.
[31] X. Dong, F. Xue, H. Zhang, et al., Evaluation of surface air temperature change over China and the globe during the twentieth century in IAP AGCM4.0, Atmospheric and Oceanic Science Letters 5(5)(2012)435–438.
[32] J. Mielikainen, B. Huang, H.L.A. Huang, et al., GPU acceleration of the updated Goddard shortwave radiation scheme in the weather research and forecasting (WRF) model, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 5(2)(2012)555–562.
[33] J. Mielikainen, B. Huang, H.L.A. Huang, et al., GPU implementation of Stony Brook University 5-class cloud microphysics scheme in the WRF, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 5(2)(2012)625–633.
[34] J. Mielikainen, B. Huang, H.L.A. Huang, et al., Improved GPU/CUDA based parallel weather and research forecast (WRF) single moment 5-class (WSM5) cloud microphysics, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 5(4)(2012)1256–1265.
[35] J. Mielikainen, B. Huang, H.L.A. Huang, et al., Speeding up the computation of WRF double-moment 6-class microphysics scheme with GPU, Journal of Atmospheric and Oceanic Technology 30(12)(2013)2896–2906.
[36] J. Michalakes, M. Vachharajani, GPU acceleration of numerical weather prediction, Parallel Processing Letters 18(04)(2008)531–548.
[37] Y. Zou, W. Xue, S. Liu, A case study of large-scale parallel I/O analysis and optimization for numerical weather prediction system, Future Generation Computer Systems 37(2014)378–389.