AGCM3D: A Highly Scalable Finite-Difference Dynamical Core of Atmospheric General Circulation Model based on 3D Decomposition

Baodong Wu[1,2], Shigang Li[1,✉], Hang Cao[1,2], Yunquan Zhang[1], He Zhang[3], Junmin Xiao[1], and Minghua Zhang[3]

[1]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Institute of Atmospheric Physics, Chinese Academy of Sciences
[✉]Corresponding author (shigangli.cs@gmail.com)

***Abstract*—It is commonly recognized that the dynamical core of the atmospheric model based on latitude-longitude mesh has poor parallel scalability, since it has to perform the costly polar or high-latitude filtering to dump out the unwanted modes. To parallelize the algorithm, only two dimensions can be partitioned even for a 3-dimensional mesh because of the costly filtering, which hinders the scalability of the algorithm. In this paper, we develop a highly scalable finite-difference dynamical core based on the latitude-longitude mesh using a 3D decomposition method, named as AGCM3D. Different from the traditional methods, our method releases the parallelism in all three dimensions, namely latitude, longitude, and level. To replace the costly Fast Fourier Transform (FFT) filtering, we propose a novel adaptive Gaussian filtering scheme, whose filtering strength increases as the latitude increases. Compared with the parallel FFT filtering, the parallel adaptive Gaussian filtering is far more efficient. In addition, we use the techniques of communication avoiding and message aggregation to further reduce the communication overhead. Experiments are conducted on Tianhe-2 supercomputer, and the resolution of the model is set as $0.5^\circ \times 0.5^\circ$ (50 km). Results show that our implementation scales up to 32,768 CPU cores in strong scaling and achieves the maximal simulation speed of 15.6 simulation-year-per-day (SYPD).**

***Keywords*-3D decomposition; parallel scalability; adaptive filtering; communication avoiding; message aggregation; atmospheric model**

I. Introduction

Numerical simulation of the global atmospheric circulation is important in climate modeling, and is also a great challenge in scientific computing. As it is essential for the atmospheric physics research to understand dynamic behaviors of the global atmospheric circulation at increasingly fine resolutions, high resolution Atmospheric General Circulation Models (AGCM) have been developed. Some recently developed atmospheric models include CAM5 [1] of the Community Earth System Model (CESM) [2] from NCAR, ECHAM-5 [3], and IAP AGCM [4]. In order to enable high-fidelity simulation of realistic problems, the study of high-performance atmospheric solvers is becoming an urgent demand.

The dynamical core is one of the most time-consuming modules of AGCM. It focuses on the dynamic evolution process of the global atmospheric circulation, which refers to the formulation of the hydrodynamic equations of the atmospheric and the numerical algorithms to solve them. In recent years, a lot of progress has been made on dynamical cores, including the flexibilities of meshes [5], discretization methods [4], [6]–[9], and parallel algorithms and optimizations [10]–[12].

Typically, the dynamical core can be numerically solved by two types of mesh, including the quasi-uniform polygonal mesh and the equal-interval latitude-longitude mesh. CAM-SE [7] is a spectral element dynamical core, which uses the quasi-uniform polygonal mesh. CAM-SE has good parallel scalability, since the quasi-uniform polygonal mesh does not require the costly polar filtering. Fu et al. [11] have migrated the CAM-SE to the Sunway TaihuLight, and scaled the performance of the model up to 10 million accelerator cores. CAM-FV [1], [13] is a finite-volume dynamical core, which uses the equal-interval latitude-longitude mesh. Similar to CAM-FV, IAP AGCM [4], [6] is a finite-difference dynamical core, which also uses the equal-interval latitude-longitude mesh. The dynamical cores based on the latitude-longitude mesh have the advantages of easily preserving the energy conservation, dealing with the discontinuous variables, and coupling with other components. However, both CAM-FV and IAP AGCM are considered to have poor parallel scalability, since they have to perform the costly polar or high-latitude filtering along the longitude dimension to dump out the unwanted modes when using the latitude-longitude mesh. Recent results [8], [12], [14] show that the dynamical cores based on the latitude-longitude mesh can only scale to thousands of CPU cores. Our work focuses on improving the parallel scalability for the dynamical cores based on the latitude-longitude mesh, and scales the performance to tens of thousands of CPU cores.

For both CAM-FV and IAP AGCM, only two dimensions (longitude and level) of the mesh are partitioned to execute on multiple cores because of the costly filtering along the

longitude dimension, which hinders the scalability of the algorithm. We develop a highly scalable finite-difference dynamical core based on latitude-longitude mesh using a 3D decomposition method, named as AGCM3D. AGCM3D releases the parallelism in all three dimensions, namely latitude, longitude, and level. A novel adaptive Gaussian filtering scheme is proposed to replace the costly FFT filtering. We also use the techniques of communication avoiding and message aggregation to further reduce the communication overhead. Experiments are conducted on Tianhe-2 supercomputer [15]. The resolution of the model is set as $0.5^\circ \times 0.5^\circ$ (50 km). On Tianhe-2, our implementation scales up to 32,768 CPU cores in strong scaling and achieves the maximal simulation speed of 15.6 simulation-year-per-day (SYPD).

1) For the finite-difference dynamical core based on the latitude-longitude mesh, we present a 3D decomposition method which releases the parallelism in all three dimensions. Our 3D decomposition method greatly improves the parallelism and the scalability of the algorithm.
2) To keep the polar or high-latitude region computationally stable, we propose a novel adaptive Gaussian filtering scheme to replace the costly FFT filtering, whose filtering strength increases as the latitude increases. The parallel adaptive Gaussian filtering significantly outperforms the parallel FFT filtering.
3) We further use the techniques of communication avoiding and message aggregation to reduce the communication overhead involved in the stencil computation, collective operations, and the adaptive Gaussian filtering.

In the next section, we discuss the motivation of our work. Section III discusses the 3D decomposition method and the adaptive Gaussian filtering scheme. Section IV discusses the communication optimizations for the 3D decomposition method. Experimental results and analysis on Tianhe-2 supercomputer are presented in Section V, and Section VI concludes.

II. Motivation

The baseline we use in this paper is the dynamical core of the fourth-generation IAP AGCM (IAP AGCM-4) code developed by Zhang et al. [4]. IAP AGCM-4 uses the finite-difference method based on the latitude-longitude mesh to solve the dynamical core. The results show that IAP AGCM-4 captures the main feature of global climatology very well [16], and has the similar simulation performance with the finite-volume dynamical core of the community atmospheric model (CAM-FV) [13]. Next, we discuss the continuous form of the governing equations and the two-dimensional (2D) decomposition scheme in IAP AGCM-4.

A. The Governing Equations

In IAP AGCM-4, the dynamic core revolves around the solutions of the baroclinic primitive equations. After the subtraction of the standard stratification and several transformations [16], the primitive equations are discretized using the finite difference scheme under the latitude-longitude mesh, which are written as

$$\begin{cases} \left[\frac{\partial U}{\partial t}\right]_{x',y,z} = \left[-\alpha^* \widetilde{L}(U) - \beta^* \widetilde{P}(\lambda) + \gamma^* f^* V\right]_{x',y,z} \\ \left[\frac{\partial V}{\partial t}\right]_{x,y',z} = \left[-\alpha^* \widetilde{L}(V) - \beta^* \widetilde{P}(\Theta) + \gamma^* f^* U\right]_{x,y',z} \\ \left[\frac{\partial \Phi}{\partial t}\right]_{x,y,z} = \left[-\alpha^* \widetilde{L}(\Phi) + \widetilde{\delta} \cdot \beta^* \widetilde{\Omega}\right]_{x,y,z} \\ \left[\frac{\partial}{\partial t}\left(\frac{p'_{sa}}{p_0}\right)\right]_{x,y} = \left[\beta^* \widetilde{P}(W) - \kappa^* \frac{D_{sa}}{P_0}\right]_{x,y} \end{cases} \tag{1}$$

In Equation (1), the U, V, P, T are the basic prognostic variables, which represent the zonal wind, meridional wind, the pressure, and the temperature, respectively. The derivatives $\frac{\partial U}{\partial t}, \frac{\partial V}{\partial t}, \frac{\partial P}{\partial t}, \frac{\partial \phi}{\partial t}$ are used to calculate the values of U, V, P, T. According to the time splitting method [16], the finite difference equations are divided into advection and adaptation processes. The advection form $\widetilde{L}(\varepsilon)$ is defined as

$$\widetilde{L}(\varepsilon) = \sum_{m=1}^{3} L_m(\varepsilon) \tag{2}$$

and adaptation forms $\widetilde{P}, \widetilde{\Omega}, \widetilde{\delta}$ are defined as

$$\begin{cases} \widetilde{P}(\lambda) = P_\lambda^{(1)} + P_\lambda^{(2)} \\ \widetilde{P}(\Theta) = P_\Theta^{(1)} + P_\Theta^{(2)} \\ \widetilde{\Omega} = \Omega^{(1)} + \Omega_\Theta^{(2)} + \Omega_\lambda^{(2)} \\ \widetilde{P}(W) = D(P) + \frac{P(W_\sigma)_k}{\Delta \sigma_k} \end{cases} \tag{3}$$

and

$$\widetilde{\delta} = (1 - \delta_p)\left[b(1 + \delta_c) + \delta \cdot \kappa \frac{\Phi}{P}\right] \tag{4}$$

For the sake of briefness, the longitude, latitude and level dimensions of the mesh are denoted as X, Y and Z dimensions in this paper, respectively. In the latitude-longitude mesh system, the central spatial difference solution method makes all the prognostic variables depend on the data in the X, Y, and Z dimensions. The values of these variables are calculated by the multi-point 3D stencil computation. As shown in Fig.1, the calculation of the variable $U_{x,y,z}$ requires the values of $U_{x\pm2,y,z}, U_{x,y\pm2,z}, U_{x,y,z\pm1}$, which is typical 3D stencil computation. The calculations for the other variables are similar.

B. Two-dimensional Decomposition

The two-dimensional (2D) decomposition is used to parallelize the dynamical core in IAP AGCM-4 and other models based on the latitude-longitude mesh, like CAM-FV [8]. For the polar or high-latitude region in the mesh space, the mesh points are very small and dense. Thus,

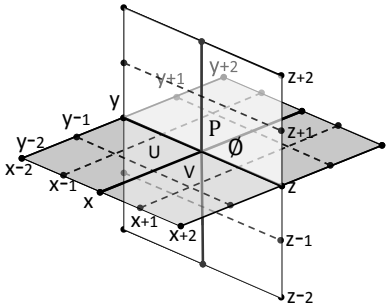
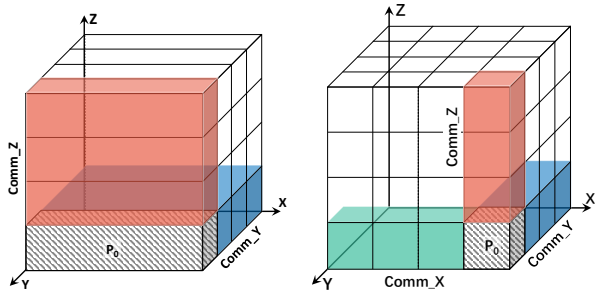


Figure 1. Stencil computation for the prognostic variables.



(a) Communication pattern of the 2D decomposition.

(b) Communication pattern of the 3D decomposition.

Figure 2. 2D decomposition vs 3D decomposition.

both IAP AGCM-4 and CAM-FV use the one-dimensional FFT filtering along the longitude (X) dimension in the high-latitude region to dump out the short-wave modes. Thus, the decomposition of X dimension inevitably causes the problem of FFT parallelization, which leads to expensive all-to-all collective communication. To avoid the high communication overhead, both IAP AGCM-4 and CAM-FV use 2D (Y-Z) decomposition to parallelize the dynamical core.

However, as the computing resources of supercomputers grow rapidly, the traditional 2D decomposition method is no longer effective enough to utilize the the rich computing resources efficiently. This is mainly because only the parallelism of the Y and Z dimensions is exploited, while the X dimension, which contains the most number of mesh points among the three dimensions, is serialized. Thus, the total degree of parallelism of the 2D decomposition is not enough, which hinders the parallel scalability. To the best of our knowledge, the state-of-the-art finite-volume dynamical core based on the latitude-longitude mesh can only scale up to 1664 MPI processes (1664 MPI processes $\times$ 4 OpenMP threads = 6656 cores) at the resolution of $0.5^\circ \times 0.5^\circ$ [14]. For IAP AGCM-4, the dynamical core can only scale up to 1024 MPI processes at the resolution of $0.5^\circ \times 0.5^\circ$ [12], with 64 processes along the Y dimension and 16 processes along the Z dimension. To fully release the parallelism of all three dimensions, we propose a novel 3D decomposition method together with an efficient filtering algorithm. In addition, the overhead of the point-to-point communication caused by the stencil computation and the overhead of collective communication along the Z dimension is also reduced significantly using the 3D decomposition method.

III. Three-dimensional decomposition and adaptive Gaussian filtering scheme

We present a 3D decomposition method, named as AGCM3D, to improve the parallel scalability and efficiency of the finite-difference dynamical core. A novel adaptive Gaussian filtering scheme is also discussed in this section.

A. The 3D decomposition method

The 3D decomposition method is implemented by partitioning all the three dimensions of the mesh and the corresponding variable arrays. The mesh points and the variable arrays are then mapped to a three-dimensional process topology. Suppose there are M, N, H mesh points and P_x, P_y, P_z processes for X, Y and Z dimensions, respectively. Fig.2(a) shows the communication pattern of the Y-Z decomposition method. The communication domains, $comm\_Y$ and $comm\_Z$, are used for data communication in Y and Z dimensions, respectively. For the 2D decomposition, each process is responsible for $MNH/P_y/P_z$ mesh points. A new communication domain $comm\_X$ along the X dimension is added in the 3D decomposition, as shown in Fig.2(b). The parallelism is increased by P_x fold and each process is responsible for $MNH/P_x/P_y/P_z$ mesh points.

The 3D decomposition not only increases the parallelism, but also decreases the communication overhead. A detailed comparison between the 2D and 3D decompositions at $0.5^\circ \times 0.5^\circ$ resolution is shown in Table I. The 3D decomposition brings extra point-to-point communication overhead along the X dimension caused by the 3D stencil computation. In practice, there are 23 variables stored in 2D (X-Y) arrays and 36 variables stored in 3D arrays required to be exchanged along the X dimension for each process, and the total size is $(23+36\times\frac{361}{P_y})\times\frac{30}{P_z}\times$*sizeof*(DOUBLE) bytes. However, compared with the traditional 2D decomposition, the volume of point-to-point communications in Y and Z dimensions caused by the stencil computation are reduced by P_x times. The volume of collective communications along the Z dimension, which are used to calculate the total energy, are also reduced by P_x times compared with the 2D decomposition. Overall, the communication overhead is reduced significantly by the 3D decomposition method. Note that the communication overhead caused by the polar or high-latitude filtering for the 3D decomposition is not presented in Table I. We will discuss our novel filtering scheme in the next section.

Table I
THE COMPARISON BETWEEN 2D AND 3D DECOMPOSITIONS

Comparison items	***2D***	***3D***
Horizontal Resolution	$0.5° \times 0.5°$	$0.5° \times 0.5°$
Number of mesh points: $M \times N \times H$	$720 \times 361 \times 30$	$720 \times 361 \times 30$
Processes number of X dimension	1	P_x
Processes number of Y dimension	P_y	P_y
Processes number of Z dimension	P_z	P_z
The theoretical parallelism	361×30	$720 \times 361 \times 30$
Per core P2P communication volume along X	0	$(23+36\times\frac{30}{P_z})\times\frac{361}{P_y}$
Per core P2P communication volume along Y	$(15+18\times\frac{30}{P_z})\times 720$	$(15+18\times\frac{30}{P_z})\times\frac{720}{P_x}$
Per core P2P communication volume along Z	$6\times\frac{361}{P_y}\times 720$	$6\times\frac{361}{P_y}\times\frac{720}{P_x}$
Per core collective communication volume along Z	$\frac{361}{P_y}\times\frac{30}{P_z}\times 720$	$0\ (if\ P_z=1)$; $\frac{361}{P_y}\times\frac{30}{P_z}\times\frac{720}{P_x}$ $(if\ P_z>1)$

B. Adaptive Gaussian filtering scheme

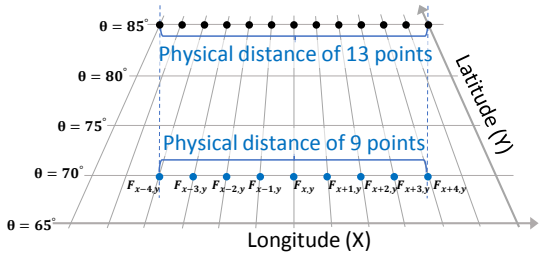


Figure 3. The latitude mesh lines cluster at the high-latitude region.

As shown in Fig.3, the latitude mesh lines cluster at the polar region. The physical distance of 9 mesh points at 70° is equal to the physical distance of 13 mesh points at 85°. Commonly, the dynamical cores based on the latitude-longitude mesh have a severe restriction on the time step. The time step must be small enough to meet the stability requirements of the governing equations, which result in high computational cost [8]. To alleviate the problem caused by the mesh lines clustering along the X dimension, the filtering module is applied in the finite-difference dynamical core. Poleward of $\pm 70°$, FFT filtering is used on the tendencies of U, V, P, T to dump out the short-wave modes in the original IAP AGCM-4 code. The key problem when implementing the 3D decomposition method is the parallelization of FFT filtering, since the data transposition of parallel FFT causes all-to-all communication along the X dimension. Recall that P_x is the number of processes and M is the number of mesh points along the X dimension. The all-to-all communication of parallel FFT incurs at least $\log_2 P_x$ number of communications and total M communication size for each process [17], which is too high to be amortized by the benefit of the 3D decomposition. We propose a new adaptive filtering scheme, which has the same filtering effect as FFT and ensures the stability and accuracy of the simulation. Compared with parallel FFT, our adaptive filtering scheme has a much lower communication overhead. The implementation of the adaptive filtering scheme is based on Gaussian filtering, whose impulse response is a Gaussian function [18]. The one-dimensional Gaussian filtering has the following form:

$$F_{x,y} = \sum_{n=-2K}^{2K} F_{(x+n),y} * W_{x,y;x+n}, n = 0, \pm 1, ..\pm 2K \quad (5)$$

where K is a positive integer. The width of Gaussian filtering $B = 4K + 1$. (x, y) are the coordinates of the central mesh point, and $W_{x,y;x+n}$ are the weighting coefficients, which are calculated by

$$W_{x,y;x+n} = C_0 e^{-\frac{n^2}{K^2}} \quad (6)$$

where C_0 is given by

$$C_0 = \frac{1}{\sum_{n=-2K}^{2K} e^{-\frac{n^2}{K^2}}} \quad (7)$$

Based on Equations (6) and (7), we can easily get

$$\sum_{n=-2K}^{2K} W_{x,y;x+n} = 1 \quad (8)$$

The value of the central mesh point after filtering is the sum of the multiplications of the weighting coefficients and the corresponding mesh point values. From x=0 to 720, 1D Gaussian filtering $F_{x,y}$ is sequentially performed on a circle of mesh points for a specific value of y. As shown in Fig.3, the 9-points Gaussian filtering value for $F_{x,y}$ is equal to the sum of the points from $F_{x-4,y}$ to $F_{x+4,y}$ multiplied by the weighting coefficients from $W_{x,y;x-4}$ to $W_{x,y;x+4}$, respectively. If K=1, we can get the weighting coefficients of 5 points Gaussian filtering. However, this simple Gaussian filtering method cannot guarantee the computational stability at the high-latitude or polar region, since the short-wave modes are not fully dumped out. Poleward of $\pm 70°$, we propose an adaptive Gaussian filtering method to adaptively adjust the width, the weighting coefficients, and the number of filtering calls according to the latitude (namely the value of y). As a result, the filtering strength increases with the increasing of the latitude.

Method (*a*) - *Adaptive filtering width*: Let θ denote the latitude. The filtering width B_θ is set to 9 at $\pm 70°$ latitude, namely $B_{\pm 70°} = 9$ and $K_{\pm 70°} = 2$. We use the

filtering width at $\pm 70°$ latitude as a baseline, and adaptively increase the width as the latitude increases. Poleward of $\pm 70°$ ($\pm 70° \leq \theta \leq \pm 90°$), the adaptive filtering width is described as:

$$B_\theta = 4K_\theta + 1, \text{where } K_\theta = K_{70°} \left\lfloor \frac{sin(90° - 70°)}{sin(90° - |\theta|)} \right\rfloor \quad (9)$$

Method (*b*) - *Adaptive weighting coefficients*: At $\pm 70°$, the weighting coefficients $W_{x,y=\pm 70°;x+n}$ can be calculated by Equation (6) and (7). Poleward of $\pm 70°$, we sine-linearly adjust the coefficients with $\pm 70°$ as the baseline.

$$W_{x,y;x+n} = W_{x,y=\pm 70°;x+n} L_\theta + \frac{1}{1 + 2K_{\pm 70°}} (1 - L_\theta) \quad (10)$$

where L_θ is defined as

$$L_\theta = \frac{sin(90° - 70°)}{sin(90° - |\theta|)} \quad (11)$$

Method (*c*) - *Adaptive number of filtering calls*: To further improve filtering strength near the pole, the number of Gaussian filtering calls is determined by

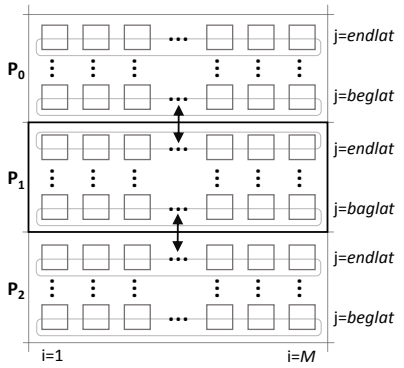
$$N_\theta = \left\lfloor \frac{sin(90° - \delta)}{sin(90° - |\theta|)} \right\rfloor, \quad 70° \leq \delta \leq 90° \quad (12)$$

in which δ is the baseline of the latitude value where the number of filtering calls begins to adaptively increase.

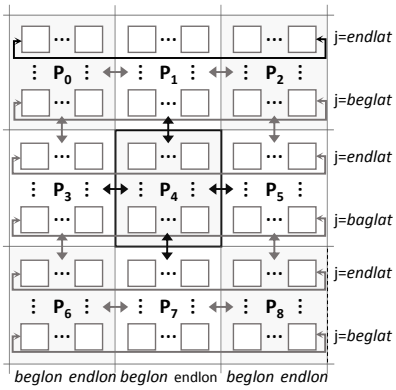
Table II
ADAPTIVE GAUSSIAN FILTERING SCHEME FOR HIGH-LATITUDE

Filtering scheme	***Iteration times***	***Latitude***
$\sum_{n=-4}^{4} F_{(x+n),y} * W_{x,y=\pm 70°;x+n}$	1	$\theta = \pm 70°$
$\sum_{n=-4}^{4} F_{(x+n),y} * W_{x,y;x+n}$	1	$\pm 70° < \theta < \pm 87°$
$\sum_{n=-6}^{6} F_{(x+n),y} * W_{x,y;x+n}$	$N_\theta(\delta = 87°)$	$\pm 87° \leq \theta \leq \pm 90°$

Our adaptive Gaussian filtering scheme for high-latitude or polar region is a combination of the above three methods, as shown in Table II. At $\pm 70°$, the 9-points Gaussian filtering is called by one time. From $\pm 71°$ to $\pm 86°$, the 9-point Gaussian filtering with adaptive weighting coefficients (Method (b)) is called by one time. Poleward of $\pm 87°$, the 13-point Gaussian filtering with adaptive weighting coefficients (Method (b)) and adaptive number of filtering calls (Method (c)) is used. As shown in Section V-B, our adaptive filtering scheme has the same filtering effect as FFT. In addition, the communication overhead of our scheme is much lower than the parallel FFT filtering in 3D decomposition, since only the point-to-point communications with the neighbor processes are needed by the Gaussian filtering. Although the Gaussian filtering (so does the point-to-point communication) is called by multiple times when using Method (c), we will discuss how to reduce the overhead of this situation by communication avoiding in the next section.

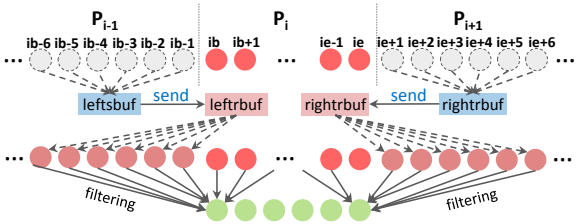


(a) Communication of stencil computation in 2D decomposition.

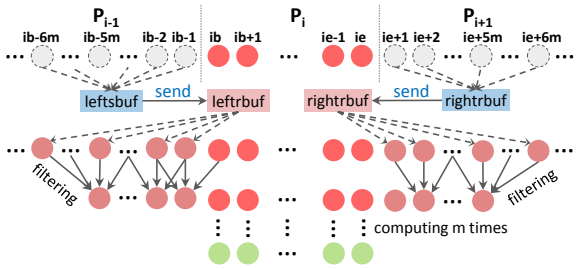


(b) Communication of stencil computation in 3D decomposition.

Figure 4. Communication of stencil computation in horizontal direction. The arrows represent communications.



(a) Without communication avoiding, m times P2P communication and m times filtering computation.



(b) With communication avoiding, one time P2P communication and m times filtering computation.

Figure 5. 13-point filtering before and after communication avoiding.

Algorithm 1 Message aggregation algorithm

Input: **sbuf_x**: Data buffer to be sent along X; **xlen**: Length of sbuf_x; **rbuf_x**: Data buffer to be received along X; **myid_x**: Process ID in the X dimension;**comm_x**: Total number of Processes along X;**PT,UT,VT,TT**: Mode variables.

Output: DPsa,DU,DV,DT: The tendencies of PT,UT,VT,TT.

```
1:  for j← beglat to endlat do
2:    sbuf_x← PT,Psa(endlon,j)
3:    for k← beglev to endlev do
4:      sbuf_x← UT,VT,TT(endlon,j,k)
5:    end for
6:  end for
7:  mpi_isend(sbuf_x,xlen,rbuf_x,myid_x-1,comm_x)
8:  mpi_irecv(rbuf_x,xlen,sbuf_x,myid_x+1,comm_x)
9:  unpack(rbuf_x,PT,Psa,UT,VT,TT)        ▷ unpack received buffer
10: computing_tend(DPsa,DU,DV)            ▷ compute tendencies
11: for k← beglev to endlev do
12:   O2P,D1K,DSK ← 0                     ▷ the variables required by DT
13:   for i← beglon to endlon do
14:     O2P(K)← O2P(K)+Pstar(i,k)         ▷ summing Pstar in X
15:     D1K(K)← D1K(K)+pv1(i)             ▷ summing pv1 in X
16:     DSK(K)← DSK(K)+DsaY1(i)           ▷ summing DsaY1 in X
17:   end for
18:   albuf_x ← O2P,D1K,DSK               ▷ packing data to send buffer
19: end for
20: mpi_allreduce(albuf_x,3*(endlev-beglev+1),MPI_SUM)
21: unpack(albuf_x,O2P,D1K,DSK)           ▷ unpack received buffer
22: computing_tend(DT)
```

IV. Communication optimizations for 3D decomposition

In this section, we discuss the techniques of message aggregation and communication avoiding used to reduce the communication overhead of the 3D decomposition method, involving the point-to-point communication incurred by the 3D stencil computation, the collective communications, and the point-to-point communication incurred by the adaptive Gaussian filtering scheme.

A. Message aggregation for Multiple Variables

The communication in the horizontal direction of the 2D decomposition is illustrated in Fig.4(a), where M is the number of mesh points in the longitude (X) dimension, $beglat$ and $endlat$ are starting and ending mesh points in the latitude (Y) dimension for each process. Due to the mesh points in the X dimension ($1 \sim M$) are all stored in the local process memory for the 2D decomposition, the communication only occurs in the Y dimension. However, the ghost region, which would be exchanged by point-to-point communication, includes all the mesh points in the X dimension. The communication in the horizontal direction of 3D decomposition is illustrated in Fig.4(b). The total M mesh points in the X dimension are divided by P_x processes, and the local mesh points in the X dimension of each process is from $beglon$ to $endlon$. Compared with the 2D decomposition, the 3D decomposition decreases the message size of the communication in the Y dimension by $\frac{M}{P_x}$ times. However, the 3D decomposition adds point-to-point communication between the direct neighbor processes along the X dimension, and periodic border communication between the first process and the last process along the X dimension. However, the same communication pattern, as shown in Fig.4(b), is needed by calculations of multiple variables, and the messages are very short. Taking 4096 processes ($32 \times 64 \times 2$) as an example, the size for each message in the X or Y dimension is only about 500 bytes. However, only the messages for more than 32 KB can achieve good bandwidth utilization for MPI over InfiniBand [19]. Therefore, we package all the short messages with the same destination as a long message, and send it by one communication to improve bandwidth utilization. Algorithm 1 shows the details of the message aggregation algorithm in the tendencies computation along the X dimension. The ghost data of PT, UT, VT, and TT along the X dimension are packaged into a communication buffer $sbuf\_x$, as shown in Lines 1-6. $sbuf\_x$ is sent to the neighbor processes using the MPI nonblocking point-to-point communication functions, mpi_isend and mpi_irecv, as shown in Lines 7-8.

Collective communication along the X dimension is also applied in the 3D decomposition, because of the tendency computation of variable DT requiring all the mesh points data of the X dimension. A similar approach is applied on the messages involved in the collective communication. Variables of $O2P$, $D1K$, and DSK are packaged into a single buffer, $albuf\_x$, as shown in Lines 11-19. As a result, the collective communication ($mpi\_allreduce$) is called only once, as shown in Line 20.

B. Communication avoiding

Due to the adaptive Gaussian filtering used in the 3D decomposition, the communication volume and the number of communication calls caused by the parallel filtering are different at different latitudes. The closer to the polar region, the more number of communications. The problem of communication imbalance severely damages the filtering performance. We use the technique of communication avoiding to minimize the communication cost, especially for the high-latitude or polar region. Let m indicate the number of calls for the adaptive filtering at latitude j. The value of m can be obtained from Equation (12). It is obvious that m is

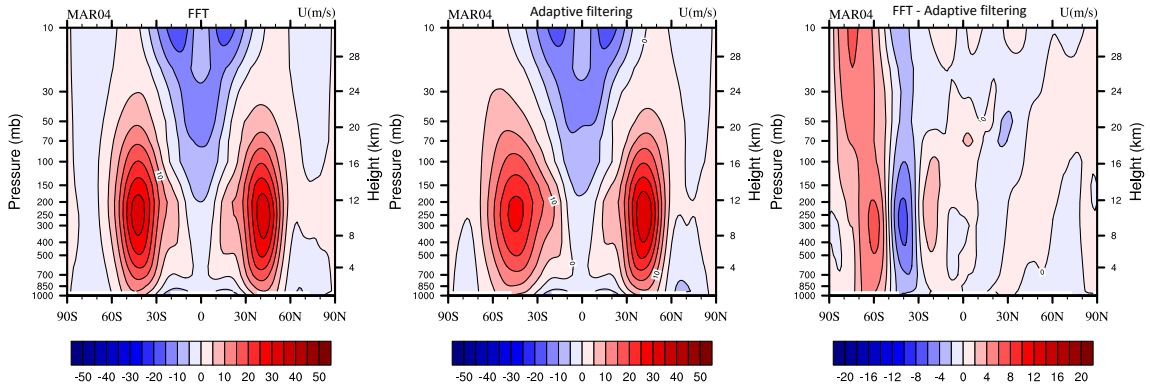


Figure 6. Distribution of zonal wind from the Held-Suarez tests.

larger for higher latitude region. Thus, we package the ghost data required by the m times filtering in advance, and send them to the related process by one communication. The same technique is also used in [20] to reduce the communication overhead for the stencil computation.

Fig.5 compares the communication and computation overhead between the naive communication strategy and the communication avoiding. For the process P_i, ib to ie is the range of the local mesh points in the X dimension, and processes P_{i-1} and P_{i+1} are the neighbors of P_i along the X dimension. For the naive communication strategy illustrated in Fig.5(a), P_{i-1} and P_{i+1} send the mesh points $ib-6 \sim ib-1$ and $ie+1 \sim ie+6$ to P_i by point-to-point communication, respectively. Then, the computation of one 13-point Gaussian filtering is performed. The above computation-after-communication procedure will be iterated m times to complete the adaptive filtering. Thus, the number of communication is very high for the polar or high-latitude region. Fig.5(b) illustrate the technique of communication avoiding used in the adaptive Gaussian filtering. All the data required by the m times filtering of P_i process, including the mesh points $ib-6m \sim ib-1$ of P_{i-1} and the mesh points $ie+1 \sim ib+6m$ of P_{i+1}, are packaged together. The packaged message is sent to P_i by only one communication. After that, the computation of the 13-point Gaussian filtering is performed by m times. The technique of communication avoiding slightly increases the computation overhead, since the number of mesh points to be filtered (for the first $m-1$ times) is larger than the number of local mesh points. However, this extra computation overhead can be amortized by the communication overhead reduction.

V. Experimental Results

A. Experiment Configurations

The experiments are conducted on Tianhe-2 supercomputer [15] . Each computational node of Tianhe-2 is equipped with two Intel Xeon E5-2692 processors (total 24 cores) and 64 GB memory. The computational nodes of Tianhe-2 are connected by TH Express-2 interconnected network. The communication library is a customized MPI, called mpi3-dynamic, which complies with the MPI 3.0 [21] standard. The backend compiler is Intel 15.0 compiler.

We compare our 3D decomposition method, AGCM3D, with the original Y-Z 2D decomposition method for IAP AGCM-4 based on the latitude-longitude mesh. To evaluate the accuracy of the simulation results and the performance, both our 3D decomposition method and the original 2D method use the idealized dry-model experiments proposed by Held and Suarez [22]. The boundary and initial conditions are interpolated from the same input data for both methods. The horizontal resolution of the atmosphere model is $0.5^\circ \times 0.5^\circ$, which is the highest resolution of IAP AGCM-4. In this resolution, the total number of mesh points is 7,797,600 ($720 \times 361 \times 30$). Table. III shows the number of processes in each dimension for the original 2D decomposition method and our 3D decomposition method. The Y-Z 2D decomposition can use 1024 processes at most, while our 3D decomposition method can use up to 65,536 processes. Considering that the number of mesh points in the X and Y dimensions is much larger than the number of levels in Z dimension, we give priority to assign the processes to X and Y dimensions first.

Table III
Processes topology of 2D decomposition and 3D decomposition

Number of processes	***2D*** $(P_y \times P_z)$	***3D*** $(P_x \times P_y \times P_z)$
128	32×4	$32 \times 4 \times 1$
256	32×8	$32 \times 8 \times 1$
512	32×16	$32 \times 16 \times 1$
1024	64×16	$32 \times 32 \times 1$
2048	–	$32 \times 64 \times 1$
4096	–	$32 \times 64 \times 2$
8192	–	$32 \times 64 \times 4$
16384	–	$32 \times 64 \times 8$
32768	–	$32 \times 64 \times 16$
65536	–	$64 \times 64 \times 16$

B. The Correctness and the Performance Advantage of the Adaptive Gaussian Filtering

We verify our adaptive Gaussian filtering scheme through the Held-Suarez test [22] for the dynamical core of IAP AGCM-4 for 2D and 3D decompositions. This test is a unified inspection standard for the dynamical core, in which the complex physical parameterization are replaced by the simple expressions. The distribution of zonal wind is one of the fundamental measures of the climate simulations. We conduct the Held-Suarez test on the original IAP AGCM-4 with the FFT filtering and the IAP AGCM-4 with our adaptive Gaussian filtering scheme. Both codes run for 40 simulation months. Fig.6 illustrates the zonal-mean zonal wind from the Held-Suarez tests, with FFT filtering on the left, our adaptive Gaussian filtering in the middle, and their difference on the right. The results show that both the FFT filtering and our adaptive Gaussian filtering can produce a reasonably realistic zonal mean circulation with westerly jet cores located near 250 hPa over the middle-latitudes of both hemispheres. Although there is a little difference between the FFT filtering and our adaptive Gaussian filtering, the overall patterns of both filtering methods are similar and reasonable.

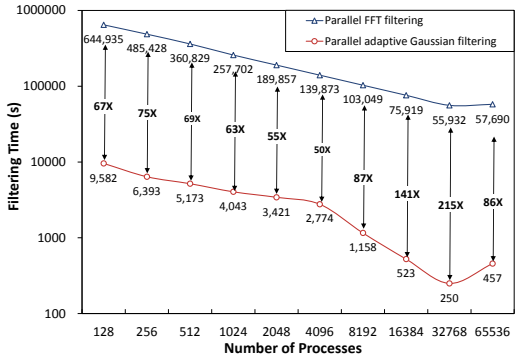


Figure 7. Performance comparison between the parallel FFT filtering and the parallel adaptive Gaussian filtering.

Fig.7 compares the performance of the parallel FFT filtering and the parallel adaptive Gaussian filtering used in the 3D decomposition. Note that the parallel adaptive Gaussian filtering is optimized by communication avoiding discussed in Section IV-B. Compared with the parallel FFT filtering, our parallel adaptive Gaussian filtering improves the performance by an average of 90x. The one-dimensional parallel FFT is call many times for filtering the values of multiple variables at each time step, which involves many all-to-all collective communications. Thus, the parallel FFT filtering results in very poor performance. Our parallel adaptive Gaussian filtering only involves several point-to-point communications with the neighbor processes, whose overhead is much lower than the parallel FFT filtering.

C. Communication Optimizations

Message aggregation is used to improve the performance of point-to-point communication and the collective communication, as discussed in Section IV-A. Fig.8 compares the performance of the naive communication and the optimized communication by message aggregation of the 3D decomposition. We can see that the optimized communication improves the performance by 10x on average. The main reason is that the long messages replace many short messages using the message aggregation optimization, and thus the bandwidth utilization is improved and communication times are reduced. The minimum communication overhead is 55s at 2048 cores for the optimized communication. As the number of cores increases from the 2048, the communication overhead increases gradually. This is because the decomposition along the Z dimension is added for more than 2048 cores, which leads to extra point-to-point communication and collective communication along the Z dimension.

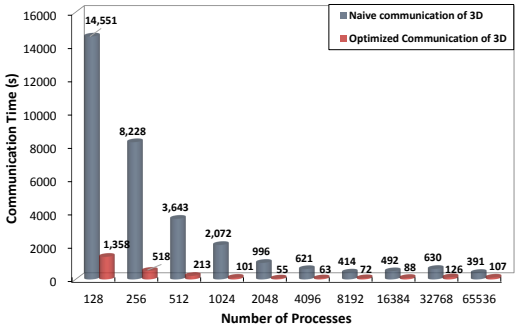


Figure 8. Communication time comparison between the naive communication and the optimized communication by message aggregation.

D. Scalability and Overall Performance Test

In the strong scaling tests, we set the resolution to be $0.5^\circ \times 0.5^\circ$ and simulate for 2 model months. The number of processes is increased from 128 to 65,536. The execution time of both 2D decomposition and 3D decomposition methods is presented in Fig.9, including the computation time, the communication time, and the filtering time. To facilitate comparison, the communication time caused by the filtering is included in the filtering time for the 3D decomposition method. The communication time for the 2D and 3D decomposition methods includes the point-to-point communication caused by the stencil computation and the collective communications.

From Fig.9, we can see that the dynamical core using 2D decomposition only scales up to 1024 processes, which is limited by the parallelism of the Y-Z decomposition. On the contrary, the 3D decomposition method can scale the performance up to 32,768 processes. Compared with the 2D decomposition, the communication time for the 3D decomposition is reduced by more than 50% on average over

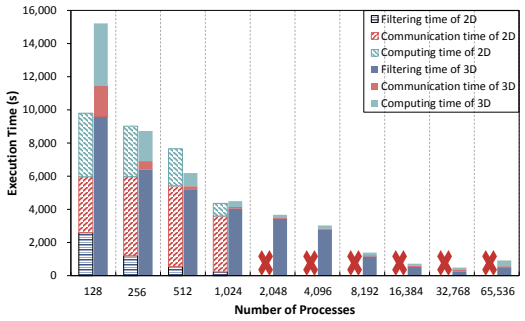


Figure 9. Strong scaling comparison for IAP AGCM-4 based on 2D and 3D decompositions.

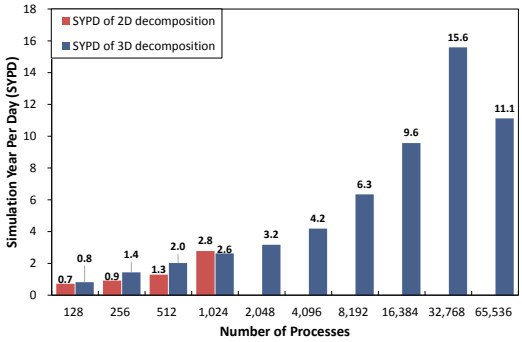


Figure 11. Simulation speed comparison between the 2D and 3D decomposition methods.

the process number from 128 to 1024. The two main factors contribute to the reduction of communication time: (1) the volume of the point-to-point communication is reduced significantly because of the partition of the X dimension; (2) the collective communication along the Z dimension is thoroughly eliminated from 128 processes to 1024 processes, since we give priority to partition the mesh points in X and Y dimensions for the 3D decomposition. However, the overall performance of the 3D decomposition is lower than the original 2D decomposition for 128 processes, because of the higher filtering time for the polar region. However, from 2048 processes to 32,768 processes, the filtering time is linearly reduced. This is because the number of processes along the Z dimension increases, and the workload of filtering for each process decreases proportionally. However, the execution time on the 65,536 processes is higher than 32,768 processes for 3D decomposition. This is because the number of processes along the X dimension increases from 32 to 64, which leads to higher communication time for the adaptive Gaussian filtering. Overall, the experimental results demonstrate that the 3D decomposition has excellent scalability. Fig.10 shows the 3D decomposition method scales from 128 processes to 32,768 processes, and achieves 30.3x speedup and 13% parallel efficiency.

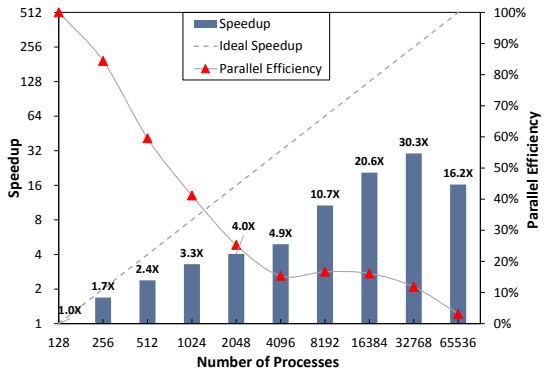


Figure 10. Speedup and parallel efficiency of the 3D decomposition method.

We also evaluate the simulation speed of dynamical core for both the 2D and 3D decomposition methods. In the area of atmospheric simulation, Simulation Year Per computing Day (SYPD) is a common indicator for measuring the simulation speed. Fig.11 presents the simulation speed comparison between the traditional 2D decomposition method and the 3D decomposition method. At the resolution of $0.5^\circ \times 0.5^\circ$, our 3D decomposition method achieves the maximum rate of 15.6 SYPD on 32,768 cores, while the traditional 2D decomposition methods can only achieve the maximum rate of 2.8 SYPD on 1024 cores. Thus, our 3D decomposition method improves the maximum simulation speedup by 5.7x compared with the traditional 2D decomposition.

VI. Conclusion

We present AGCM3D, a 3D decomposition method for the finite-difference dynamical core based on latitude-longitude mesh. AGCM3D increases the parallelism of dynamical core significantly by adding decomposition on the longitude dimension. High-latitude FFT filtering is replaced by the new adaptive Gaussian filtering, which has the same filtering effect as FFT. The adaptive Gaussian filtering only needs to communicate with neighbor processes when being parallelized, rather than the collective communication required by the parallel FFT. Furthermore, using the techniques of message aggregation and communication avoiding, the overhead of the point-to-point communication and the collective communication in the dynamical core is significantly reduced. Compared with the original IAP AGCM-4 based on 2D decomposition, AGCM3D achieves much better parallel scalability and significant performance improvement on the Tianhe-2 supercomputer.

We foresee that our method will achieve even better scalability for the higher-resolution simulation (such as $0.25^\circ \times 0.25^\circ$), since it has higher arithmetic intensity. Besides, our 3D decomposition method and adaptive Gaussian filtering scheme can also be used to optimize the scalability of other models, such as CAM-FV. For the future work, we

will couple AGCM3D with the physical process, and utilize many-core architectures to further speedup the simulation.

VII. Acknowledgments

This work was supported by National Key R&D Program of China under Grant No. 2016YFB0200800; National Natural Science Foundation of China under Grant No. 61502450 and Grant No. 61432018.

References

[1] X. Liu, R. C. Easter, S. J. Ghan, R. Zaveri, P. Rasch, X. Shi, J.-F. Lamarque, A. Gettelman, H. Morrison, F. Vitt *et al.*, "Toward a minimal representation of aerosols in climate models: Description and evaluation in the Community Atmosphere Model CAM5," *Geoscientific Model Development*, vol. 5, no. 3, p. 709, 2012.

[2] J. W. Hurrell, M. M. Holland, P. R. Gent, S. Ghan, J. E. Kay, P. J. Kushner, J.-F. Lamarque, W. G. Large, D. Lawrence, K. Lindsay *et al.*, "The community earth system model: a framework for collaborative research," *Bulletin of the American Meteorological Society*, vol. 94, no. 9, pp. 1339–1360, 2013.

[3] E. Roeckner, G. Bäuml, L. Bonaventura, R. Brokopf, M. Esch, M. Giorgetta, S. Hagemann, I. Kirchner, L. Kornblueh, E. Manzini *et al.*, "The atmospheric general circulation model ECHAM 5. PART I: Model description," 2003.

[4] H. Zhang, Z. Lin, and Q. Zeng, "The computational scheme and the test for dynamical framework of IAP AGCM-4," *Chinese J. Atmos. Sci*, vol. 33, pp. 1267–1285, 2009.

[5] P. A. Ullrich, P. H. Lauritzen, and C. Jablonowski, "Geometrically Exact Conservative Remapping (GECoRe): Regular latitude–longitude and cubed-sphere grids," *Monthly Weather Review*, vol. 137, no. 6, pp. 1721–1741, 2009.

[6] Q. Zeng, X. Zhang, X. Liang, C. Yuan, and S. Chen, "Documentation of IAP two-level atmospheric general circulation model," *USDo Energy (Ed.)*, p. 383, 1989.

[7] J. M. Dennis, J. Edwards, K. J. Evans, O. Guba, P. H. Lauritzen, A. A. Mirin, A. St-Cyr, M. A. Taylor, and P. H. Worley, "CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model," *The International Journal of High Performance Computing Applications*, vol. 26, no. 1, pp. 74–89, 2012.

[8] A. A. Mirin and W. B. Sawyer, "A scalable implementation of a finite-volume dynamical core in the community atmosphere model," *The International Journal of High Performance Computing Applications*, vol. 19, no. 3, pp. 203–212, 2005.

[9] M. F. Wehner, A. A. Mirin, P. G. Eltgroth, W. P. Dannevik, C. R. Mechoso, J. D. Farrara, and J. A. Spahr, "Performance of a distributed memory finite difference atmospheric general circulation model," *Parallel Computing*, vol. 21, no. 10, pp. 1655–1675, 1995.

[10] C. Yang, W. Xue, H. Fu, L. Gan, L. Li, Y. Xu, Y. Lu, J. Sun, G. Yang, and W. Zheng, "A peta-scalable CPU-GPU algorithm for global atmospheric simulations," in *ACM SIGPLAN Notices*, vol. 48, no. 8. ACM, 2013, pp. 1–12.

[11] H. Fu, J. Liao, N. Ding, X. Duan, L. Gan, Y. Liang, X. Wang, J. Yang, Y. Zheng, W. Liu *et al.*, "Redesigning CAM-SE for peta-scale climate modeling performance and ultra-high resolution on Sunway TaihuLight," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2017, p. 1.

[12] J. Xiao, S. Li, B. Wu, H. Zhang, K. Li, E. Yao, Y. Zhang, and G. Tan, "Communication-Avoiding for Dynamical Core of Atmospheric General Circulation Model," in *Proceedings of the 47th International Conference on Parallel Processing*. ACM, 2018, p. 12.

[13] R. B. Neale, C.-C. Chen, A. Gettelman, P. H. Lauritzen, S. Park, D. L. Williamson, A. J. Conley, R. Garcia, D. Kinnison, J.-F. Lamarque *et al.*, "Description of the NCAR community atmosphere model (CAM 5.0)," *NCAR Tech. Note NCAR/TN-486+ STR*, vol. 1, no. 1, pp. 1–12, 2010.

[14] J. M. Dennis, M. Vertenstein, P. H. Worley, A. A. Mirin, A. P. Craig, R. Jacob, and S. Mickelson, "Computational performance of ultra-high-resolution capability in the Community Earth System Model," *The International Journal of High Performance Computing Applications*, vol. 26, no. 1, pp. 5–16, 2012.

[15] X. Liao, L. Xiao, C. Yang, and Y. Lu, "MilkyWay-2 supercomputer: system and application," *Frontiers of Computer Science*, vol. 8, no. 3, pp. 345–356, 2014.

[16] Z. He, L. Zhaohui, and Z. Qingcun, "The mutual response between dynamical core and physical parameterizations in atmospheric general circulation models," *Climatic and Environmental Research (in Chinese)*, vol. 16, no. 1, pp. 15–30, 2011.

[17] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in MPICH," *The International Journal of High Performance Computing Applications*, vol. 19, no. 1, pp. 49–66, 2005.

[18] K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE transactions on automatic control*, vol. 45, no. 5, pp. 910–927, 2000.

[19] J. Liu, J. Wu, and D. K. Panda, "High performance RDMA-based MPI implementation over InfiniBand," *International Journal of Parallel Programming*, vol. 32, no. 3, pp. 167–198, 2004.

[20] J. Demmel, M. Hoemmen, M. Mohiyuddin, and K. Yelick, "Avoiding communication in sparse matrix computations," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1–12.

[21] M. Forum, "MPI: A message-passing interface standard version 3.0," in *Technical report, University of Tennessee, Knoxville*, 2012.

[22] I. M. Held and M. J. Suarez, "A proposal for the intercomparison of the dynamical cores of atmospheric general circulation models," *Bulletin of the American Meteorological Society*, vol. 75, no. 10, pp. 1825–1830, 1994.