Communication-Avoiding for Dynamical Core of Atmospheric General Circulation Model

Junmin Xiao
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
xiaojunmin@ict.ac.cn

Shigang Li
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
lishigang@ict.ac.cn

Baodong Wu
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
wubaodong@ict.ac.cn

He Zhang
Institute of Atmospheric Physics
Chinese Academy of Sciences
Beijing, China
zhanghe@mail.iap.ac.cn

Kun Li
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
likun@ict.ac.cn

Erlin Yao
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
yaoerlin@ict.ac.cn

Yunquan Zhang
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
zyq@ict.ac.cn

Guangming Tan
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
tgm@ict.ac.cn

ABSTRACT

Dynamical core is one of the most time-consuming parts in the global atmospheric general circulation model, which is widely used for the numerical simulation of the dynamic evolution process of global atmosphere. Due to its complicated calculation procedures and the non-uniformity of latitude-longitude mesh, the parallelization suffers from high communication overhead. In this paper, we deduce the operator form of the calculating flow in the dynamical core. Furthermore, it is abstracted out that the stencil and collection alternate action is the basic operation in the dynamic core. Based on the operator form of the calculation flow, we propose the corresponding optimization strategy for each operator. In the end, we develop a communication-avoiding algorithm to reduce communication overhead in the dynamic core. Our experiments show that the communication-avoiding algorithm reduces the total runtime by 54% at most for a 50 km resolution model running 10 years. Especially for communication reduction, the new algorithm achieves 1.4x speedup on average for the collective communication and 3.9x speedup on average for the communication involved in the stencil computation.

ICPP 2018, August 13–16, 2018, Eugene, OR, USA

ACM ISBN 978-1-4503-6510-9/18/08…$15.00
https://doi.org/10.1145/3225058.3225140

CCS CONCEPTS

• **Computing methodologies** → **Parallel algorithms**; *Massively parallel algorithms*; • **Applied computing** → **Environmental sciences**; *Earth and atmospheric sciences*;

KEYWORDS

communication avoiding; operator form of calculation flow; collective communication; stencil computation; atmospheric general circulation model

ACM Reference Format:
Junmin Xiao, Shigang Li, Baodong Wu, He Zhang, Kun Li, Erlin Yao, Yunquan Zhang, and Guangming Tan. 2018. Communication-Avoiding for Dynamical Core of Atmospheric General Circulation Model. In *ICPP 2018: 47th International Conference on Parallel Processing, August 13–16, 2018, Eugene, OR, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3225058.3225140

1 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 atmosphere circulation at increasingly fine resolutions [10], high resolution global atmospheric general circulation models have been developed in recent years. In order to enable high-fidelity simulations of realistic problems, the study of parallel optimization for atmospheric solvers is becoming an urgent demand.

The dynamical core of global atmospheric general circulation model (AGCM), as one of the most time-consuming parts of AGCM, focuses on the dynamic evolution process of global atmospheric circulation, which refers to the formulation of the hydrodynamic equations of the atmosphere and the numerical algorithms to solve

them. In recent years, a lot of progress has been made on dynamical cores, including the formulation of equations, flexibilities of grids [20], discretization methods [6, 26, 27], and parallel algorithms [18, 27]. Although the finite volume and spectral element methods have been developed and adopted into the dynamical core based on unstructured grids [6] in this decade, the finite difference dynamical core using latitude-longitude meshes is always widely used due to it being able to preserve the energy conservation, which is important for the climate modeling. Since the finite difference dynamical core usually involves several kinds of complicated communications, such as neighbor communication and collective communication along different directions of the latitude-longitude mesh, the communication overhead is large, and the communication time takes up the main part of the runtime of the dynamical core [23, 27]. In order to improve the performance of the finite difference dynamical core, this work focuses on the communication optimization.

In resent decades, the communication-avoiding technique has been developed based on the lower bound analysis. Using the technique for avoiding communication, most classical algorithms in numerical computation have been optimal [3, 5]. In this paper, we consider the application of the communication-avoiding technique for the global atmospheric circulation simulation, and design a communication-avoiding algorithm for the dynamical core of IAP-AGCM 4.0, which is developed by Institute of Atmospheric Physics, Chinese Academy of Sciences [22, 27]. It includes two main parts: the dynamic core and the physical parameterizations. The dynamic core is the main computational part of the model. Our numerical test in Figure 1 demonstrates that the communication time dominates the runtime of the dynamical core indeed. Hence, the communication optimization is important for the performance improvement of the dynamical core. Following are the primary contributions and novelties of the paper:

- This paper first deduces the operator form of the calculating flow in the dynamical core of AGCM. Furthermore, it is abstracted out that the stencil and collection alternate action is the basic operation in the dynamic core, which is an important insight useful for the design of the communication-avoiding algorithm.
- For several communication operations in the dynamical core, the corresponding optimization strategies are proposed. Consequently, a communication-avoiding algorithm is designed, which significantly reduces the frequency of collective communication and local communication, and maximizes the potential overlap of the computation and communication in the stencil of the dynamical core.
- The communication-avoiding algorithm is implemented in the dynamical core of IAP-AGCM 4.0 which is a complicated program for the simulation of realistic problems. The experimental evaluation demonstrates significant performance improvement on communication. For the 50 km resolution model running 10 model years, the communication-avoiding algorithm reduces the total runtime by 54% at most. Compared the new algorithm with the original algorithm using the best decomposition scheme, the new algorithm achieves 1.4x speedup on average for the collective communication and 3.9x speedup on average for the communication involved in the stencil computation.

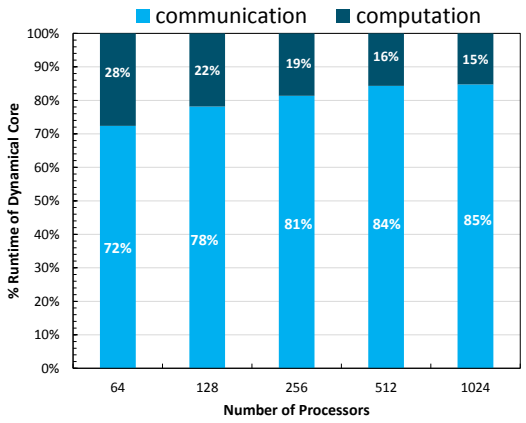

Figure 1: The percentage of times for communication and computation in the dynamical core, where the size of a latitude-longitude mesh is $n_x \times n_y \times n_z = 720 \times 360 \times 30$. Each processor on distributed-memory clusters manages a MPI process independently.

2 RELATED PRIOR WORK

2.1 Dynamic Evolution Equations

To facilitate numerical analysis and algorithmic design, the dynamical core of IAP-AGCM 4.0 uses the following variable substitution referred to a tensor transform,

$$\begin{cases} U = Pu, \\ V = Pv, \\ \Phi = PR(T - \tilde{T})/b, \\ p'_{sa} = p_s - \tilde{p}_s, \end{cases} \tag{1}$$

where u and v are two velocity vectors along latitude and longitude directions on the surface of the sphere respectively; T is the temperature; p_s is the surface pressure; and $b = 87.8$ m s^{-1} is the characteristic velocity of gravity wave propagation in the standard atmosphere; R is the gas constant for dry air; $P = \sqrt{p_{es}/p_0}$, $p_{es} = p_s - p_t$, $p_t = 2.2$ hPa is the pressure at the model top layer and $p_0 = 1000$ hPa; $\tilde{T}$ and $\tilde{p}_s$ are state variables which characterize temperature and pressure respectively on the standard stratification. After the subtraction of standard stratification, with the transform (1) and the vertical coordinate $\sigma = (p - p_t)/p_{es}$, the dynamic evolution equations (DEEs) can be written as

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

where $\delta = p_t/p$, f^* is determined by $f^* = 2\Omega \cos\theta + u \cot\theta/a$ with a as the earth radius, Ω is the angular velocity of the earth rotation. Here, δ is 0 with 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. The L_1 and L_2 are two horizontal advection terms and L_3 is the vertical convection term, which are defined by

$$\begin{cases} L_1(F) = \frac{1}{2a\sin\theta}(2\frac{\partial Fu}{\partial\lambda} - F\frac{\partial u}{\partial\lambda}), \\ L_2(F) = \frac{1}{2a\sin\theta}(2\frac{\partial Fv\sin\theta}{\partial\theta} - F\frac{\partial v\sin\theta}{\partial\theta}), \\ L_3(F) = \frac{1}{2}(2\frac{\partial F\dot{\sigma}}{\partial\sigma} - F\frac{\partial\dot{\sigma}}{\partial\sigma}), \end{cases} \tag{3}$$

where $\dot{\sigma}$ is the vertical velocity on σ level. The pressure gradient terms are calculated according to the following formulas:

$$\begin{cases} P_\theta^{(1)} = P\frac{\partial\phi'}{a\partial\theta}, \\ P_\theta^{(2)} = \frac{b\Phi(1-\delta_p)}{p_{es}}\cdot\frac{\partial p_{es}}{a\partial\theta}, \\ P_\lambda^{(1)} = P\frac{\partial\phi'}{a\sin\theta\partial\lambda}, \\ P_\lambda^{(2)} = \frac{b\Phi(1-\delta_p)}{p_{es}}\cdot\frac{\partial p_{es}}{a\sin\theta\partial\lambda}. \end{cases} \tag{4}$$

The terms $\Omega^{(1)}$, $\Omega_\theta^{(2)}$, $\Omega_\lambda^{(2)}$, $D(P)$ and D_{sa} are determined by

$$\begin{cases} \Omega^{(1)} = \frac{W}{\sigma} - \frac{1}{P}[D(P) + \frac{\partial PW}{\partial\sigma}], \\ \Omega_\theta^{(2)} = \frac{V}{p_{es}}\cdot\frac{\partial p_{es}}{a\partial\theta}, \\ \Omega_\lambda^{(2)} = \frac{U}{p_{es}}\cdot\frac{\partial p_{es}}{a\sin\theta\partial\lambda}, \end{cases} \tag{5}$$

and

$$\begin{cases} D(P) = \frac{1}{a\sin\theta}(\frac{\partial PU}{\partial\lambda} + \frac{\partial PV\sin\theta}{\partial\theta}), \\ D_{sa} = \nabla\cdot(\tilde{\rho}_{sa}k_{sa}\nabla\frac{p'_{sa}}{\tilde{\rho}_{sa}p_0}), \end{cases} \tag{6}$$

where $\tilde{\rho}_{sa} = \tilde{p}_s/R\tilde{T}s$ is the density of the standard atmosphere at the surface, $k_{sa} = 0.1$ is the dissipation coefficient.

2.2 Dynamical Core And Atmospheric Circulation Simulation

Dynamical core is a key component in AGCM to solve DEEs. For discretizing the equation (2), the dynamical core uses a finite difference scheme with a terrain-following σ coordinate [12]. A 3-dimensional latitude-longitude mesh with Arakawa's C grid staggering is used in the horizontal discretization [1]. In the latitude-longitude mesh [20], latitude, longitude and normal directions of the spherical surface are denoted briefly as x, y and z directions, and numbers of nodes along the three directions are n_x, n_y and n_z respectively. Although the latitude-longitude mesh may not maintain load-balance due to the non-uniformity [13], it is able to achieve the conservation of the sum of kinetic energy, the available potential energy, and the available surface potential energy under the transform (1), which leads to high-fidelity simulations of realistic problems. However, with the development of high resolution global atmospheric general circulation models, the grid lines of the latitude-longitude mesh cluster at the pole, which creates a potentially severe Courant-Friedrichs-Lewy (CFL) restriction on the time step. In the decade, many techniques have been proposed to handle this pole problem, such as Fourier filtering [27]. They can successfully deal with this clustering but they result in the reduced parallel scalability [18]. Thus, there is a growing interest in highly scalable algorithms [26] based on less structured or unstructured grids with more uniform resolution [2, 24]. Although the finite volume and spectral element methods are developed and adopted into the dynamical core based on unstructured grids [6] in resent years, the finite difference dynamical core under latitude-longitude grids is still widely used since it can preserve the energy conservation, which is important for the climate modeling. In this paper, we propose an algorithm for avoiding communication in the finite difference dynamical core.

In the dynamical core of AGCM, the global atmosphere is defined on the earth, and exhibits a broad range of different spatial and temporal scales. These characteristics of the atmosphere require the stencil computation and the global computation involving collective communications. To avoid collective communication along some direction, 2-dimensional latitude-longitude domain decomposition schemes for message passing are usually used in the implementation of AGCM [23, 27]. Although the 2-dimensional decomposition strategies impact the parallelism of atmospheric models, they are always more efficient than 3-dimensional decomposition in real-world applications. To accelerate stencil computation in atmospheric models, recent proposed approaches focus on better utilizing the heterogeneous processors [7], such as GPU [9, 15, 26] and Intel MIC [25]. However, reducing the pure communication overhead is still a challenging problem. In order to develop scalable algorithms for the atmospheric circulation simulation, related work choose some standard atmosphere models with simple forms as the test bed, such as shallow-water equations [26], which exhibit most of the essential characteristics of the atmosphere but are much simpler than DEEs for the simulation of real problems. In order to optimize the finite difference dynamical core for solving DEEs, our research focuses on reducing the data movement between processors on distributed-memory clusters.

3 ORIGINAL ALGORITHM

As the dynamical core of AGCM is developed on the distributed-memory context, we assume that there are p processors, while p_x, p_y and p_z processors ($p = p_x \times p_y \times p_z$) are distributed respectively along x, y and z directions of the latitude-longitude mesh T with $n_x \times n_y \times n_z$ points. In the distributed computing platform, all processors are at equal status, and we do not distinguish the processors within a computation node and the processors across nodes.

In the dynamical core, the finite difference discretization of DEEs is as follows

$$[\frac{\partial\xi}{\partial t}]_{i,j,k} = [\widetilde{\mathcal{A}}(\xi)]_{i,j,k} + [\widetilde{\mathcal{L}}(\xi)]_{i,j,k}, \tag{7}$$

where the functions $\widetilde{\mathcal{A}}$ and $\widetilde{\mathcal{L}}$ are defined as

$$[\widetilde{\mathcal{A}}(\xi)]_{i,j,k} = \begin{pmatrix} (-P_\lambda^{(1)} - P_\lambda^{(2)} - f^*V)_{i-\frac{1}{2},j,k} \\ (-P_\theta^{(1)} - P_\theta^{(2)} - f^*U)_{i,j+\frac{1}{2},k} \\ ((1-\delta_p)[b(1+\delta_c) + \delta\kappa\Phi/P] \\ \qquad \cdot(\Omega^{(1)} + \Omega_\theta^{(2)} + \Omega_\lambda^{(2)}))_{i,j,k} \\ p_0\kappa^*(D_{sa})_{i,j} - p_0\sum_{k=1}^{n_z}\Delta\sigma_k\cdot D(P)_{i,j,k} \end{pmatrix},$$

and

$$[\widetilde{\mathcal{L}}(\xi)]_{i,j,k} = \begin{pmatrix} -\sum_{m=1}^{3} L_m(U)_{i-\frac{1}{2},j,k} \\ -\sum_{m=1}^{3} L_m(V)_{i,j+\frac{1}{2},k} \\ -\sum_{m=1}^{3} L_m(\Phi)_{i,j,k} \\ 0 \end{pmatrix}.$$

Denote $\mathcal{D}$ as the set of all elements of $\xi = (U, V, \Phi, p'_{sa})$. The functions $\widetilde{\mathcal{A}}$ and $\widetilde{\mathcal{L}}$ are defined on $\mathcal{D}$. Both $\widetilde{\mathcal{A}}$ and $\widetilde{\mathcal{L}}$ involve with stencil computations, while $\widetilde{\mathcal{A}}$ has to need a collective communication along z direction due to the summation in the fourth element of $\widetilde{\mathcal{A}}$.

The functions $\widetilde{\mathcal{A}}$ and $\widetilde{\mathcal{L}}$ are corresponding to the adaptation and advection processes in the atmospheric circulation respectively.

Algorithm 1 Original algorithm

Require: $\xi^{(0)}$;
1: // Dynamical evolution process of AGCM with the number of time-step from $k = 1$ to K;
2: **for** $k = 1$ to K **do**
3: $\psi^0 = \xi^{(k-1)}$
4: // Adaptation process needs M nonlinear iterations, and each iteration is composed of 3 internal updates;
5: **for** $i = 1$ to M **do**
6: $\eta_1 = \psi^{i-1} + \Delta t_1 \cdot \widetilde{\mathcal{F}}\widetilde{\mathcal{A}}(\psi^{i-1})$;
7: $\eta_2 = \psi^{i-1} + \Delta t_1 \cdot \widetilde{\mathcal{F}}\widetilde{\mathcal{A}}(\eta_1)$;
8: $\eta_3 = \psi^{i-1} + \Delta t_1 \cdot \widetilde{\mathcal{F}}\widetilde{\mathcal{A}}(\frac{\psi^{i-1}+\eta_2}{2})$;
9: $\psi^i = \eta_3$;
10: **end for**
11: // Advection process requires only 1 nonlinear iteration;
12: $\zeta_1 = \psi^M + \Delta t_2 \cdot \widetilde{\mathcal{F}}(\widetilde{\mathcal{L}}(\psi^M))$;
13: $\zeta_2 = \psi^M + \Delta t_2 \cdot \widetilde{\mathcal{F}}(\widetilde{\mathcal{L}}(\zeta_1))$;
14: $\zeta_3 = \psi^M + \Delta t_2 \cdot \widetilde{\mathcal{F}}(\widetilde{\mathcal{L}}(\frac{\psi^M+\zeta_2}{2}))$;
15: // Smoothing;
16: $\xi^{(k)} = \widetilde{\mathcal{S}}(\zeta_3)$;
17: **end for**
18: **return** $\xi^{(K)}$;

The dynamical core integrates the equation (7) using Algorithm 1 which is a nonlinear time integration method [27]. In Algorithm 1, $\widetilde{\mathcal{S}}$ represents a smoothing function for numerical stability, and the function $\widetilde{\mathcal{F}}$ is associated with Fourier filtering to deal with the pole problem [21]. Fourier filtering explores 1-dimensional fast Fourier transform (FFT) as a tool for filtering high-frequency waves. Since Fourier filtering is executed at each latitude circle, the collective communication in FFT is along x direction.

As seen from Algorithm 1, different time integration methods are chosen for the adaptation and advection processes respectively at each time step of the numerical simulation in the dynamical core, due to the different physical characteristics of two processes. The adaptation process is a high frequency change in the atmospheric circulation, while the advection process is related to the large-scale and relative-stable activities of atmosphere on the earth. Hence, the nonlinear iteration for the adaptation process has to be executed for M times with $\Delta t_1 \ll \Delta t_2$, and each nonlinear iteration is composed of 3 internal updates. However, only one nonlinear iteration is required by the advection process.

4 COMMUNICATION AVOIDING FOR DYNAMICAL CORE OF AGCM

In this section, we consider the communication optimization for the dynamical core of AGCM. First of all, we deduce the operator form of the calculating flow in the dynamical core, in which each operator involves only one kind of communication. Next, the corresponding optimization strategy is proposed for each operator. Finally, a communication-avoiding algorithm is designed.

4.1 Operator Form of Calculating Flow

Since the function $\widetilde{\mathcal{A}}$ contains stencil computations and a summation operation along z direction, both the local communication and collective communication along z direction occur in $\widetilde{\mathcal{A}}$. In order to facilitate the algorithm design for improving two different kinds of communications in $\widetilde{\mathcal{A}}$, we write $\widetilde{\mathcal{A}}$ as a summation of two functions $\widetilde{\mathcal{A}} = \widehat{\mathcal{C}} + \widehat{\mathcal{A}}$ where

$$[\widehat{\mathcal{C}}(\xi)]_{i,j,k} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -p_0 \sum_{k=1}^{n_z} \Delta\sigma_k \cdot D(P)_{i,j,k} \end{pmatrix}, \widehat{\mathcal{A}} = \widetilde{\mathcal{A}} - \widehat{\mathcal{C}}.$$

It is obvious that $\widehat{\mathcal{A}}$ corresponds to stencil computations, and $\widehat{\mathcal{C}}$ is a summation function along z direction. By this way, two different kinds of communications in $\widetilde{\mathcal{A}}$ are split successfully. In conclusion, Algorithm 1 mainly involves five functions in which $\widehat{\mathcal{A}}$, $\widetilde{\mathcal{L}}$ and $\widetilde{\mathcal{S}}$ associates with stencil computations, $\widehat{\mathcal{C}}$ requires a collective communication along z direction, and $\widetilde{\mathcal{F}}$ is Fourier filtering needing a collective communication along x direction.

For a tensor $\xi \in \mathcal{D}$ on a 3-dimensional $n_x \times n_y \times n_z$ mesh T with independent directions x, y and z, we define *the update of a mesh point* $v_{i_1,i_2,i_3} \in T$ as updating $\xi|_{v_{i_1,i_2,i_3}}$ which represents the data of ξ on the point v_{i_1,i_2,i_3}. For example, in the $(2m+1)^3$-point stencil computation ($m \geq 1$), the update of each $v_{i_1,i_2,i_3} \in T$ involves all v_{j_1,j_2,j_3} such that $|j_k - i_k| \leq m$ ($k = 1, 2, 3$). If the update of each $v_{i_1,i_2,i_3} \in T$ involves all points along some direction, such as 1-dimensional FFT along x direction in which the update of $v_{i_1,i_2,i_3} \in T$ needs all $v_{j_1,i_2,i_3} \in T$ ($1 \leq j_1 \leq n_x$), we call this kind of update way as a *collective computation*.

Let $n = n_x \times n_y \times n_z$. An operator is defined as a map from $\mathcal{D}^n$ to $\mathcal{D}^n$. Denote $\mathcal{A}$ as the operator corresponding to $\widehat{\mathcal{A}}$. For $\mathbf{w} = \mathcal{A}\mathbf{v}$ with $\mathbf{w}$ and $\mathbf{v}$ in $\mathcal{D}^n$, the operator $\mathcal{A}$ updating the k-th element of $\mathbf{v}$ corresponds to the update of v_{i_1,i_2,i_3} using the function $\widehat{\mathcal{A}}$, where $k = i_1 + (i_2 - 1)n_x + (i_3 - 1)n_x n_y$. Similarly, we can define operators $\mathcal{C}$, $\mathcal{L}$, $\mathcal{F}$ and $\mathcal{S}$ of $\widehat{\mathcal{C}}$, $\widetilde{\mathcal{L}}$, $\widetilde{\mathcal{F}}$ and $\widetilde{\mathcal{S}}$ respectively. Since the addition and scalar-multiplication operations for updating each mesh point v_{i_1,i_2,i_3} are independent with other points, the addition and scalar-multiplication in Algorithm 1 can be ignored. Hence, the K iterations of the dynamical core can be written as the following operator form

$$\xi^{(K)} = [\mathcal{S}(\mathcal{F}\mathcal{L})^3(\mathcal{F}\mathcal{C}\mathcal{A})^{3M}]^K \xi^{(0)}, \tag{8}$$

where each operator only involves one kind of communication. Based on (8), we can obtain the operator form of the calculating flow (Figure 2), which clearly shows that each iterative step performs a stencil computation and a collective computation alternatively. Therefore, the stencil-collective alternate action is the basic operation in the dynamic core.

4.2 Optimization for Collective Computation

Fourier filtering operator $\mathcal{F}$ and the summation operator C involve with collective communication along x and z directions respectively. In order to avoid one of these two collective communications, it is natural to set $p_z = 1$ or $p_x = 1$, which yields two different strategies: X-Y decomposition and Y-Z decomposition strategies [27]. Although the 2-dimensional decomposition schemes impact the

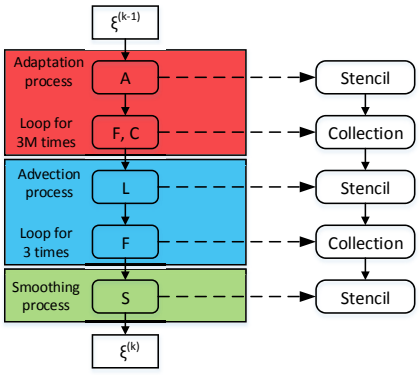

Figure 2: The operator form of the calculating flow.

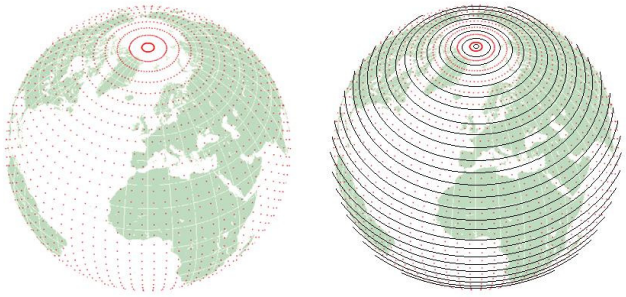

Figure 3: A latitude-longitude grid (left), and the domain decomposition scheme (right) for mesh points where all points on a latitude circle are distributed to the same processor.

parallelism of atmospheric models, they are wildly used, and more efficient than 3-dimensional decomposition. In the following, we use the data movement lower bounds to reason about domain decomposition schemes, and select an optimal choice for minimizing the communication cost.

Firstly, in the related work for computing the discrete Fourier transform of m values based on the directed acyclic graph (DAG) of m-input FFT, when no processor computes more than a constant fraction of the total number of vertices of the DAG, the number of words of data W moved by each processor (communication cost) is $\Omega(m \log m/(p \log(m/p)))$ [4, 14]. This bound does not assume any particular I/O protocol, and only requires that every input resides in the local memory of exactly one processor before the computation begins. For Fourier filtering $\mathcal{F}$, both the FFT and inverse one would be executed once respectively. As the inverse FFT is the same as FFT in the computation flow, except for using different coefficients for multiplication, the communication lower bound for n_x-input Fourier filtering $\mathcal{F}$ can be obtained by a simple application of the result for the bound of FFT.

Theorem 4.1. *Any algorithm computes the n_x-input Fourier filtering $\mathcal{F}$ with p_x processors, where $1 \le p_x \le n_x$. If each processor computes at most $\varepsilon(n_x \log n_x)$ vertices in the DAG of FFT (the same for the inverse FFT) with an arbitrary constant $\varepsilon \in (0, 1)$, then the communication cost of any algorithm is*

$$W = \Omega(\frac{2n_x \log n_x}{p_x \log(n_x/p_x)} \cdot \eta_x), \tag{9}$$

where

$$\eta_x = \begin{cases} 0, & if\, p_x = 1, \\ 1, & if\, p_x \ge 2. \end{cases} \tag{10}$$

Secondly, according to the analysis on collective communication operations in [19], we can deduce the data movement lower bound of the summation operation C, which can be attained by Ring algorithms [19]. Hence, the following result is valid.

Theorem 4.2. *Any parallel execution of the operator C, requires the communication cost,*

$$W = \Omega(2(p_z - 1)n_x n_y). \tag{11}$$

Thirdly, in the practical application, n_x, n_y are always larger than n_z, and $p_x \le n_x/2, p_y \le n_y/2, p_z \le n_z/2$, Hence, $\frac{n_x n_y n_z \log n_x}{p_x \log(n_x/p_x)} \gg (p_z - 1)n_x n_y$. Furthermore, Theorem 4.1 and Theorem 4.2 indicate that the communication cost of Fourier filtering is the high order term in the communication lower bound of the dynamical core. Since the increase of communication cost in the dynamical core is determined by the change of the high order term in the lower bound, an important optimization principle is avoiding communication of the operation corresponding to the high order term in the lower bound as much as possible. Hence, it is necessary to minimize the data movement in Fourier filtering at first.

In conclusion, Y-Z decomposition strategy is a better choice for the collective computation in the dynamical core. However, it should be noted that, X-Y decomposition is more suitable for the stencil computation, because the communication volume of the stencil on Y-Z decomposition is lager than that based on X-Y decomposition (see Section 5.2).

4.2.1 Domain decomposition scheme for $\mathcal{F}$. From the equations (9) and (11), it is obvious that, if we set $\eta_x = 0$, the high order term would be canceled. Based on this observation, it is reasonable to use only one processor to deal with all computation/processing tasks on x direction in the dynamical core, which leads to $p_x = 1$ and $\eta_x = 0$. It means that both 1-dimensional FFT and the inverse one along each latitude circle are executed by one processor. Consequently, there is no data movement in Fourier filtering. By this way, we avoid communication in Fourier filtering. Therefore, this work chooses the Y-Z decomposition ($p_x = 1$) as the optimization scheme for $\mathcal{F}$ (Figure 3).

4.2.2 Approximate nonlinear iteration for C. Since the existing implementation for collective communication has been optimal on the communication cost [19], this work focuses on reducing the frequency of the operator C in the dynamic core. In the adaptation process, one nonlinear iteration can be approximately expanded as

$$\begin{aligned} \psi^i \quad &\approx \psi^{i-1} + \alpha\Delta t_1 \widetilde{\mathcal{F}}(\widehat{C} + \widehat{\mathcal{A}})(\psi^{i-1}) \\ &+\beta(\Delta t_1)^2 \widetilde{\mathcal{F}}(\widehat{C} + \widehat{\mathcal{A}})\widetilde{\mathcal{F}}(\widehat{C} + \widehat{\mathcal{A}})(\psi^{i-1}) \\ &+\gamma(\Delta t_1)^3 \widetilde{\mathcal{F}}(\widehat{C} + \widehat{\mathcal{A}})\widetilde{\mathcal{F}}(\widehat{C} + \widehat{\mathcal{A}})\widetilde{\mathcal{F}}(\widehat{C} + \widehat{\mathcal{A}})(\psi^{i-1}). \end{aligned} \tag{12}$$

On the right-hand side of the expansion (12), the second through fourth terms are three corrections for ψ^{i-1}, and constitute a power series on Δt_1. According to the strong stability analysis of high-order time integration methods [8], the fourth term is the highest

order correction term, which can be computed inexactly by approximation methods. Since $\widehat{C}(\psi^{i-1})$ is associated with the summation of pressures along z direction, and the change of pressures is relatively slow, $\widehat{C}(\psi^{i-2})$ is a good approximation of $\widehat{C}(\psi^{i-1})$ at a small time step. Hence, in the highest order minim term of (12), we can use $\widehat{C}(\psi^{i-2})$ to replace $\widehat{C}(\psi^{i-1})$ for reducing the number of the execution of C. Accordingly, we propose an approximate nonlinear iteration approach for the adaptation process as:

$$\begin{aligned}\psi^i \quad &\approx \psi^{i-1} + \alpha\Delta t_1 \widetilde{\mathcal{F}}(\widehat{C}+\widehat{\mathcal{A}})(\psi^{i-1}) \\ &+\beta(\Delta t_1)^2 \widetilde{\mathcal{F}}(\widehat{C}+\widehat{\mathcal{A}})\widetilde{\mathcal{F}}(\widehat{C}+\widehat{\mathcal{A}})(\psi^{i-1}) \\ &+\gamma(\Delta t_1)^3 \widetilde{\mathcal{F}}(\widehat{C}+\widehat{\mathcal{A}})\widetilde{\mathcal{F}}(\widehat{C}+\widehat{\mathcal{A}})\widetilde{\mathcal{F}}(\widehat{C}(\psi^{i-2})+\widehat{\mathcal{A}}(\psi^{i-1})),\end{aligned} \tag{13}$$

where $1 \le i \le M$ and $\psi^0 = \xi^{(k-1)}, \psi^{-1} = \xi^{(k-2)}$. In this iterative approach, C is executed only twice in each nonlinear iteration of the adaptation process. Compared with the original method, M summation operations can be avoided successfully in the new approach, and one third of communication costs are reduced.

4.3 Optimization for Stencil Computation

Table 1: Stencil Computation in Adaptation Process

Term	x direction	y direction	z direction
$P_\lambda^{(1)}$	$i, i\pm1, i-2$	j	$k, k+1$
$P_\lambda^{(2)}$	$i, i\pm1, i-2$	j	k
f^*V	$i, i-1$	$j, j-1$	k
$P_\theta^{(1)}$	i	$j, j+1$	$k, k+1$
$P_\theta^{(2)}$	i	$j, j+1$	k
f^*U	$i, i+1$	$j, j+1$	k
Ω^1	i	j	$k, k+1$
Ω_θ^2	i	$j, j\pm1$	k
Ω_λ^2	$i, i\pm1\ i-2, i\pm3$	j	k
$D(P)$	$i, i-1\ i+2, i\pm3$	$j, j-1$	k
D_{sa}	$i, i\pm1$	$j, j\pm1$	k

Table 2: Stencil Computation in Advection Process

Term	x direction	y direction	z direction
$L_1(U)$	$i, i\pm1, i\pm2, i\pm3$	j	$k, k+1$
$L_2(U)$	$i, i-1$	$j\ j\pm1,$	k
$L_3(U)$	$i, i-1$	j	$k, k\pm1$
$L_1(V)$	$i, i\pm1, i+2, i\pm3$	$j, j+1$	k
$L_2(V)$	i	$j, j\pm1$	k
$L_3(V)$	i	$j, j+1$	$k, k\pm1$
$L_1(\Phi)$	$i, i\pm1, i+2, i\pm3$	j	k
$L_2(\Phi)$	i	$j, j\pm1$	k
$L_3(\Phi)$	i	j	$k, k\pm1$

4.3.1 Halo areas and partition computing for $\mathcal{A}$ and $\mathcal{L}$. From the definitions of $\widehat{\mathcal{A}}$ and $\widetilde{\mathcal{L}}$, the functions $\widehat{\mathcal{A}}$ and $\widetilde{\mathcal{L}}$ consist of several terms which involve with the stencil computations in the advection and advection processes. Tables 1 and 2 respectively present how the updates of a point $v_{i,j,k}$ in stencils of the advection and advection processes depend on its neighboring mesh points. Under the Y-Z decomposition scheme for the latitude-longitude mesh, we only need to consider the behaviors of $\mathcal{A}$ and $\mathcal{L}$ along y and z directions to optimize the stencil computations in the adaptation and advection processes.

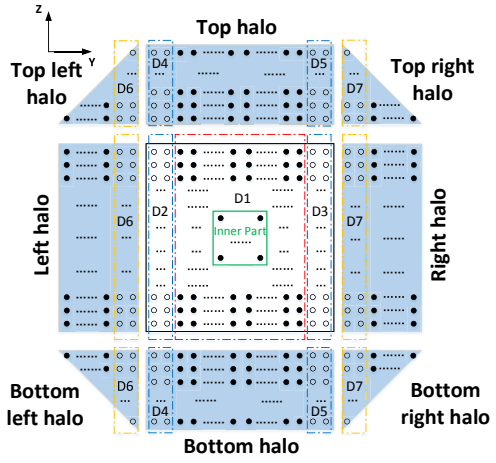

Figure 4: The computational block owned by each processor and eight halo areas for $3M$ stencil updates.

For the adaptation process, based on the idea of avoiding communication in sparse matrix-vector multiplication [3, 5], eight halo areas are allocated around the computational block (black box in Figure 4) of each processor to store the data from its neighbors for $3M$ stencil updates (Figure 4). Because some updates on points near boundaries are computed by both processors, this approach performs some redundant work. But, the communication occurs only once for $3M$ stencil updates.

In order to overlap some parts of the computation time and communication time, we separate the computation block into an inner part (green box in Figure 4) where the execution of $\mathcal{A}$ does not require the information from halo areas, and an outer part that is the remaining part of the block. At the beginning of the adaptation process, each processor communicates once with eight neighbors to send all the components of ξ needed by the neighboring processors for $3M$-times stencil computation. Simultaneously, each processor computes the stencil update of inner part. As long as all messages from eight neighbors arrive at the halo areas, the stencil computations on the outer part and halo areas are executed.

Similar to the adaptation process, we can use the same approach to improve the communication of $\mathcal{L}$.

4.3.2 Operator splitting for $\mathcal{S}$. For any $\xi = (U, V, \Phi, p'_{sa})$, $\widetilde{\mathcal{S}}$ is defined as $\widetilde{\mathcal{S}}(\xi) = (P_1(U), P_1(V), P_2(\Phi), P_2(p'_{sa}))$, where

$$P_1(\varphi) = \varphi - \beta\frac{1}{2^4}\delta_\lambda^4\varphi,$$

and

$$P_2(\varphi) = \varphi - \beta\frac{1}{2^4}[\delta_\lambda^4\varphi + \delta_\theta^4\varphi] + \frac{1}{2^8}\beta^2\delta_\theta^4\delta_\lambda^4\varphi, \forall\varphi \in \{U, V, \Phi, p'_{sa}\}.$$

The three smoothing operations $\delta_\lambda^4\varphi$, $\delta_\theta^4\varphi$ and $\delta_\theta^4\delta_\lambda^4\varphi$ depend linearly on $\varphi_{i',j',k}$ ($i' = i, i\pm1, i\pm2, j' = j, j\pm1, j\pm2$). From Table 3, it is obvious that the stencil computation in the smoothing is just associated with x and y directions.

Table 3: Stencil Computation in Smoothing

Term	x direction	y direction	z direction
P_1	$i, i \pm 1, i \pm 2$	j	k
P_2	$i, i \pm 1, i \pm 2$	$j, j \pm 1, j \pm 2$	k

Since the smoothing $\mathcal{S}$ only involves the stencil along x and y directions, the communication along the y direction only needs to be considered for $\mathcal{S}$ under Y-Z decomposition scheme. As each element in $\widetilde{\mathcal{S}}(\xi)$ is a line combination of some components of ξ, we can rewrite $\widetilde{\mathcal{S}}(\xi)$ as follows

$$\widetilde{\mathcal{S}}(\xi) = \widetilde{\mathcal{S}}_{j-2}(\xi) + \widetilde{\mathcal{S}}_{j-1}(\xi) + \widetilde{\mathcal{S}}_{j}(\xi) + \widetilde{\mathcal{S}}_{j+1}(\xi) + \widetilde{\mathcal{S}}_{j+2}(\xi), \quad (14)$$

where $\widetilde{\mathcal{S}}_l(\xi)$ corresponds to the contribution of all $\xi|_{v_{i',l,k}}$ ($i' = i, i \pm 1, i \pm 2$) for the update of $\xi|_{v_{i,j,k}}$, and $l = j, j \pm 1, j \pm 2$. Furthermore, we set $\widetilde{\mathcal{S}}_L = \widetilde{\mathcal{S}}_j + \widetilde{\mathcal{S}}_{j-1} + \widetilde{\mathcal{S}}_{j-2}$, $\widetilde{\mathcal{S}}'_L = \widetilde{\mathcal{S}}_{j+1} + \widetilde{\mathcal{S}}_{j+2}$, and $\widetilde{\mathcal{S}}_R = \widetilde{\mathcal{S}}_j + \widetilde{\mathcal{S}}_{j+1} + \widetilde{\mathcal{S}}_{j+2}$, $\widetilde{\mathcal{S}}'_R = \widetilde{\mathcal{S}}_{j-1} + \widetilde{\mathcal{S}}_{j-2}$. Based on the important observation (14), we split $\mathcal{S}$ into two operators which correspond to two different stages: former smoothing and later smoothing. Former smoothing happens before each processor sends information to halo areas of neighbors for the next adaptation process, and later smoothing is executed after halo areas receives all messages. By this way, we can fuse two communications for the adaptation and smoothing processes together. Figure 5 shows that a processor executes both former smoothing and later smoothing with its left neighbor.

During the first stage, the former smoothing is executed in the computational block of each processor (black box in Figure 4). At the beginning of former smoothing, $\xi|_{v_{i,j,k}}$ with $v_{i,j,k}$ in D_2 and D_3 (Figure 4) are copied for later smoothing. In Figure 5, the green and yellow bars highlight the data copied by a processor and its left neighbor respectively. Next, the full smoothing process $\widetilde{\mathcal{S}}(\xi)$ is able to be executed on D_1 (Figure 4), because the stencil of smoothing on each mesh point only needs at most two points on its left or right side along y direction (Table 3). Meanwhile, $\widetilde{\mathcal{S}}_R(\xi)$ and $\widetilde{\mathcal{S}}_L(\xi)$ are done on D_2 and D_3 respectively. For this process, the corresponding function can be represented as

$$\widetilde{\mathcal{S}}_1(\xi|_{v_{i,j,k}}) = \begin{cases} \widetilde{\mathcal{S}}(\xi|_{v_{i,j,k}}), & v_{i,j,k} \in D_1, \\ \widetilde{\mathcal{S}}_R(\xi|_{v_{i,j,k}}), & v_{i,j,k} \in D_2, \\ \widetilde{\mathcal{S}}_L(\xi|_{v_{i,j,k}}), & v_{i,j,k} \in D_3. \end{cases}$$

After the former smoothing is finished, the message transfer for smoothing occurs, which can be combined with the communication for the next adaptation process. For example, in Figure 5, a processor gains all messages from the left neighbor through one communication. The received messages contain the yellow bar for later smoothing, and the data for the next adaptation process which are some elements of $\xi' = \widetilde{\mathcal{S}}_1(\xi)$ on the left neighbor's $3M$ right layers of computational block. The data for the next adaptation process would be stored in the left halo of the processor.

At the second stage, the later smoothing updates the mesh points on $D_2, D_3, \cdots, D_7$, while it is not needed on the solid points of the block and eight halos (Figure 4). Using $\widetilde{\mathcal{S}}'_L$ and $\widetilde{\mathcal{S}}'_R$, the later

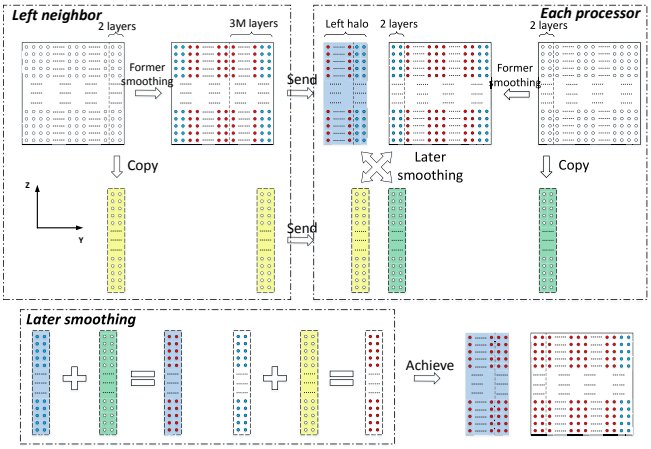

Figure 5: Former smoothing and later smoothing. Before the smoothing starts, the mesh points are drawn as empty dots. The blue solid dots represent the mesh points which still need later smoothing, while former smoothing on them has been executed. The red solid dots show the areas where the smoothing is finished.

smoothing $\widetilde{\mathcal{S}}_2$ can be defined as

$$\widetilde{\mathcal{S}}_2(\widetilde{\mathcal{S}}_1(\xi|_{v_{i,j,k}})) = \begin{cases} \xi'|_{v_{i,j,k}}, & v_{i,j,k} \in D_1, \\ \xi'|_{v_{i,j,k}} + \widetilde{\mathcal{S}}'_R(\xi|_{v_{i,j,k}}), & v_{i,j,k} \in D_2, D_4, D_7, \\ \xi'|_{v_{i,j,k}} + \widetilde{\mathcal{S}}'_L(\xi|_{v_{i,j,k}}), & v_{i,j,k} \in D_3, D_5, D_6, \end{cases}$$

where $\xi' = \widetilde{\mathcal{S}}_1(\xi)$. From (14), we have $\widetilde{\mathcal{S}} = \widetilde{\mathcal{S}}_2 \cdot \widetilde{\mathcal{S}}_1$. It means that former smoothing and later smoothing accomplish the mission of smoothing (Figure 5), while only one communication is just needed for data preparation for both the smoothing operation and the next adaptation process.

4.4 Communication-Avoiding Algorithm

Together with the optimized communication schedules proposed above, Algorithm 2 shows the details of a new algorithm for the dynamical core, which is based on the Y-Z decomposition scheme. Since the Y-Z decomposition scheme is chosen in the new algorithm, $\widetilde{\mathcal{F}}$ involves no communication (Section 4.2.1). In each iterative step, an approximate nonlinear iteration is proposed to reduce the frequency of using the summation operation $\widehat{C}$ (Section 4.2.2). By using $\widehat{C}(\psi^{i-2})$ to replace $\widehat{C}(\psi^{i-1})$ in the highest order correction term, 2 collective communications along z direction are only needed in each nonlinear iteration, and one third of communication costs are reduced successfully.

For the optimization of stencil computations in the dynamical core, halo areas and partition computation strategy are designed to reduce the times of local communication and maximize the potential overlap of computation and communication (Section 4.3.1). Furthermore, an operator splitting technique is developed for fusing the communications in the adaptation process and the smoothing (Section 4.3.2). Therefore, two communications are only required for stencil updating, which are overlapped with computation.

Algorithm 2 A communication-avoiding algorithm

Require: $\xi^{(0)}$;
$\xi^{(-1)} = \xi^{(0)}$;
for $k = 1$ to K **do**
  $\psi^{-1} = \xi^{(k-2)}$, $\psi^0 = \xi^{(k-1)}$;
  if $k \geq 2$ **then**
    $\psi^0 = S_1(\psi^0)$ in D_1, D_2, D_3; // former smoothing
  end if
  send data for the smoothing operation and $3M$ stencil computations in the adaptation process;
  $\eta_1 = \psi^0 + \Delta t_1 \cdot \widetilde{\mathcal{F}}(\widehat{\mathcal{C}}(\psi^{-1}) + \widehat{\mathcal{A}}(\psi^0))$ in inner part of the block;
  receive all messages from eight neighbors;
  if $k \geq 2$ **then**
    $\psi^0 = S_2(\psi^0)$ in D_2, D_3, $\cdots$, D_7; // later smoothing
  end if
  $\eta_1 = \psi^0 + \Delta t_1 \cdot \widetilde{\mathcal{F}}(\widehat{\mathcal{C}}(\psi^{-1}) + \widehat{\mathcal{A}}(\psi^0))$ in outer part of the block and halo areas;
  $\eta_2 = \psi^0 + \Delta t_1 \cdot \widetilde{\mathcal{F}}(\widehat{\mathcal{C}}(\eta_1) + \widehat{\mathcal{A}}(\eta_1))$ in the block and halo areas;
  $\eta_3 = \psi^0 + \Delta t_1 \cdot \widetilde{\mathcal{F}}(\widehat{\mathcal{C}}(\frac{\psi^0+\eta_2}{2}) + \widehat{\mathcal{A}}(\frac{\psi^0+\eta_2}{2}))$ in the block and halo areas;
  $\psi^1 = \eta_3$;
  for $i = 2$ to M **do**
    $\eta_1 = \psi^{i-1} + \Delta t_1 \cdot \widetilde{\mathcal{F}}(\widehat{\mathcal{C}}(\psi^{i-2}) + \widehat{\mathcal{A}}(\psi^{i-1}))$ in the block and halo areas;
    $\eta_2 = \psi^{i-1} + \Delta t_1 \cdot \widetilde{\mathcal{F}}(\widehat{\mathcal{C}}(\eta_1) + \widehat{\mathcal{A}}(\eta_1))$ in the block and halo areas;
    $\eta_3 = \psi^{i-1} + \Delta t_1 \cdot \widetilde{\mathcal{F}}(\widehat{\mathcal{C}}(\frac{\psi^{i-1}+\eta_2}{2}) + \widehat{\mathcal{A}}(\frac{\psi^{i-1}+\eta_2}{2}))$ in the block and halo areas;
    $\psi^i = \eta_3$;
  end for
  send data for 3 stencil computations in the advection process;
  $\zeta_1 = \psi^M + \Delta t_2 \cdot \widetilde{\mathcal{F}}(\widetilde{\mathcal{L}}(\psi^M))$ in inner part of the block;
  receive all messages from eight neighbors;
  $\zeta_1 = \psi^M + \Delta t_2 \cdot \widetilde{\mathcal{F}}(\widetilde{\mathcal{L}}(\psi^M))$ in outer part of the block and halo areas;
  $\zeta_2 = \psi^M + \Delta t_2 \cdot \widetilde{\mathcal{F}}(\widetilde{\mathcal{L}}(\zeta_1))$ in the block and halo areas;
  $\zeta_3 = \psi^M + \Delta t_2 \cdot \widetilde{\mathcal{F}}(\widetilde{\mathcal{L}}(\frac{\psi^M+\zeta_2}{2}))$ in the block and halo areas;
end for
$\xi^{(K)} = \widetilde{\mathcal{S}}\xi^{(K)}$;
return $\xi^{(K)}$;

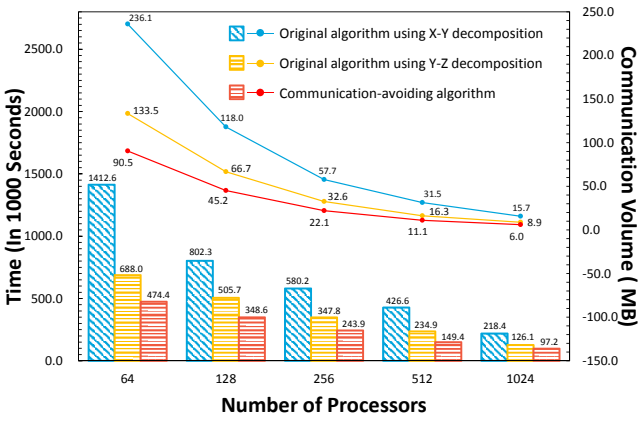

Figure 6: Time for Collective Communication.

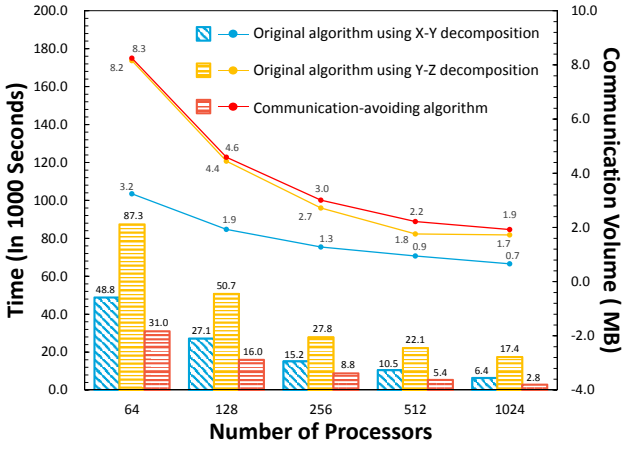

Figure 7: Communication Time of Stencil.

5 EVALUATION

In this section, we compare the performance of the new algorithm with two original algorithms in the dynamical core of AGCM, and present numerical results on Tianhe-2 supercomputer [25].

5.1 Platform and Benchmark

Tianhe-2 was the world's fastest supercomputer from 2013 to 2015. Each node of Tianhe-2 is equipped with two Intel Ivy Bridge CPUs (24 cores). The interface nodes are connected using an InfiniBand network. The system software includes a 64-bit Kylin OS, an Intel 14.0 compiler, a customized MPICH-3.1 for TH Express-2, and a self-designed hybrid hierarchy file system H2FS.

To evaluate our optimized algorithm for dynamical core, we conduct idealized dry-model experiments proposed by Held and Suarez [11], referred to as H-S. H-S test is a widely used benchmark calculation for evaluating the dynamical cores of AGCM independently of the physical parameterizations. In the 3-dimensional latitude-longitude mesh, $n_x \times n_y \times n_z = 720 \times 360 \times 30$ corresponding to the 50 km resolution, which is the highest resolution version we can obtain. In this version, the number of processes used under Y-Z decomposition is 1024 at most. Hence, we use the 1024 CPU cores (rather than KNC co-processors) of Tianhe-2 in our numerical test, where each processor is responsible for managing a MPI process independently. We set the number of nonlinear iterations in each step as $M = 3$, and execute the atmospheric simulation for 10 model years.

5.2 Experimental Results

From Figure 6, it is obvious that the communication time for $\mathcal{F}$ in X-Y decomposition is much longer than that for $\mathcal{C}$ in Y-Z decomposition due to $n_x \gg n_z$, which is consistent with our analysis in Section 4.2.1. In addition, compared with the original algorithm using Y-Z decomposition, the communication-avoiding algorithm achieves on average 1.4x speedup for the collective communication, and about 30% of the communication volumes are reduced, because

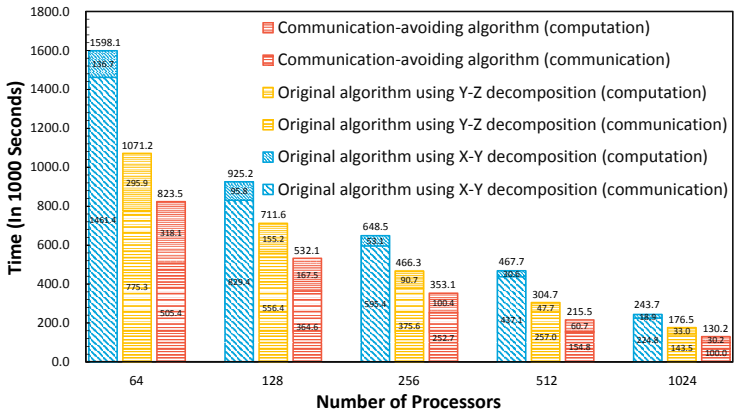

Figure 8: Total Runtime of Dynamical Core.

one third of summation operations along z direction are eliminated by using an approximate nonlinear iteration (Section 4.2.2).

From Algorithm 1, it is easy to deduce the approximate estimates for the communication costs of stencil computations in the original algorithms based on X-Y and Y-Z decomposition schemes respectively, $W_{XY}^{stencil} = \Theta(2(3M+4)(n_z \cdot \frac{n_y}{p_x} + n_z \cdot \frac{n_y}{p_y}))$, and $W_{YZ}^{stencil} = \Theta(2(3M+4)(n_x \cdot \frac{n_y}{p_y} + n_x \cdot \frac{n_z}{p_z}))$. Since $n_x \gg n_y, n_z$, the communication volume of stencil on X-Y decomposition is much smaller than that based on Y-Z decomposition. This fact is demonstrated again by Figure 7. Accordingly, the communication time of stencil using X-Y decomposition is the shortest. Compared with the original algorithm using Y-Z decomposition, the communication-avoiding algorithm needs a little more communication volumes for stencil, because some redundant data from four small triangle halos (Figure 4) have to be transferred for executing several iterations of stencil computation with no communication. But, the new strategy reduces the communication frequency from 13 to 2 in each iterative step ($M = 3$, and see Section 4.3.1). As one communication involves about 20 MPI_Isend and MPI_Recv operations (due to the length of ξ being ten), over 200 communication operations are avoided successfully in Algorithm 2. Consequently, the new algorithm achieves 3x-6x (on average 3.9x) speedup for the communication in the stencil updating (Figure 7). Compared the new algorithm with the original algorithm using Y-Z decomposition scheme, the communication time for the stencil computation decreases from 17,400 seconds to 2,800 seconds on the 1024 cores.

In Figure 8, we give the runtime of the dynamical core. Compared with the original algorithm with X-Y decomposition, the communication-avoiding algorithm reduces the total runtime by 54% at most, when $p = 512$. Compared with the original algorithm with Y-Z decomposition, the new algorithm achieves on average 1.4x speedup for the runtime, due to the time for collective communication dominating the total runtime and about 30% of the communication volumes of the collective communication being reduced. On 1024 cores, about 113,500 seconds and 46,300 seconds are saved by the new algorithm respectively compared with the original algorithms under X-Y and Y-Z decomposition schemes.

5.3 Theoretical Analysis

In order to analyze the synchronization, communication, and computation of processors in the parallel computation, a general theoretical model is proposed in [16]. This theoretical model counts the amount of work and data movement as a maximum of any execution path during the parallel computation (refer to [16]), which is elegant and realistic. Based on this model, by the similar analysis in [17], we can deduce the number of words of data W moved by each processor (communication cost) and the number of synchronizations S (network latency cost) of the communication-avoiding algorithm as follows:

$$W_{CA} = \Theta(2MK(n_x \frac{n_y}{p_y} \frac{n_z}{p_z} \cdot \log(p_z))), S_{CA} = \Theta((2M+2)K),$$

while the corresponding costs of two original algorithms are

$$W_{YZ} = \Theta(3MK(n_x \frac{n_y}{p_y} \frac{n_z}{p_z} \cdot \log(p_z))), S_{YZ} = \Theta((6M+4)K),$$

and

$$W_{XY} = \Theta(6MK(n_z \frac{n_y}{p_y} \frac{n_x}{p_x} \cdot \log(p_x))), S_{XY} = \Theta((9M+10)K).$$

Since n_x is usually larger than n_z, it is obvious that $W_{XY} \gg W_{YZ} > W_{CA}$, and $S_{XY} > S_{YZ} > S_{CA}$, which is consistent with the numerical results. Hence, the communication and latency costs of our communication-avoiding algorithm are more close to the lower bounds than those of the original ones.

For the collective communication, the new algorithm avoids collective communication along x direction in Fourier filtering, and eliminates one third of summation operations along z direction by using an approximate nonlinear iteration. The performance improvement of the new algorithm on the collective communication is able to be well kept for more processors. For the communication in stencil computation, each processor only needs to communicate with its neighbors, which is not changed with the increasing of the total number of processors. Hence, our optimization method for the communication in stencil computation is scalable. Thus, we assert that the performance improvement on communication would be always achieved by our communication-avoiding algorithm even

when a much larger number of processors are used for higher resolution simulations.

6 CONCLUSIONS

This paper has obtained the operator form of the calculating flow in the dynamical core of AGCM. Furthermore, it is abstracted out that the stencil-collective alternate action is the basic operation in the dynamic core. Furthermore, several optimization strategies have been proposed to improve collective communication and local communication in the dynamical core respectively, which leads to an efficient communication-avoiding algorithm. This work also has shown an experimental evaluation of the new algorithm, demonstrating significant performance improvement on communication.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. The work is supported by the National Key Research and Development Program of China under Grant No. 2016YFB0200800 and National Natural Science Foundation of China under Grant No. 61502450.

REFERENCES

[1] Akio Arakawa and Vivian R. Lamb. 1977. Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model. *Methods in Computational Physics: Advances in Research and Applications* 17 (1977), 173–265. https://doi.org/10.1016/B978-0-12-460817-7.50009-4

[2] Yuya Baba, Keiko Takahashi, Takeshi Sugimura, and Koji Goto. 2010. Dynamical Core of an Atmospheric General Circulation Model on a Yin-Yang Grid. *Monthly Weather Review* 138, 10 (2010), 3988–4005. https://doi.org/10.1175/2010MWR3375.1

[3] Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. 2011. Minimizing communication in Numerical Linear Algebra. *SIAM J. Matrix Anal. Appl.* 32, 3 (2011), 866–901. https://doi.org/10.1137/090769156

[4] Gianfranco Bilardi, Michele Scquizzato, and Francesco Silvestri. 2012. A lower bound technique for communication on BSP with application to the FFT. In *Euro-Par 2012 Parallel Processing*. Springer, Berlin, Heidelberg, 676–687. https://doi.org/10.1007%2F978-3-642-32820-6_67

[5] J. Demmel, M. Hoemmen, M. Mohiyuddin, and K. Yelick. 2008. Avoiding communication in sparse matrix computations. In *2008 IEEE International Symposium on Parallel and Distributed Processing*. IEEE, Miami, FL, USA, 1–12. https://doi.org/10.1109/IPDPS.2008.4536305

[6] John M. Dennis, Jim Edwards, Katherine J. Evans, Oksana Guba, Peter H. Lauritzen, Arthur A. Mirin, Amik Stcyr, Mark A. Taylor, and Patrick H. Worley. 2012. CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model. *International Journal of High Performance Computing Applications* 26, 1 (2012), 74–89. https://doi.org/10.1177/1094342011428142

[7] Haohuan Fu, Junfeng Liao, Wei Xue, Lanning Wang, Dexun Chen, Long Gu, Jinxiu Xu, Nan Ding, Xinliang Wang, Conghui He, Shizhen Xu, Yishuang Liang, Jiarui Fang, Yuanchao Xu, Weijie Zheng, Jingheng Xu, Zhen Zheng, Wanjing Wei, Xu Ji, He Zhang, Bingwei Chen, Kaiwei Li, Xiaomeng Huang, Wenguang Chen, and Guangwen Yang. 2016. Refactoring and Optimizing the Community Atmosphere Model (CAM) on the Sunway Taihulight Supercomputer. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16)*. IEEE Press, Piscataway, NJ, USA, Article 83, 12 pages. http://dl.acm.org/citation.cfm?id=3014904.3015016

[8] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. 2001. Strong Stability-Preserving High-Order Time Discretization Methods. *Siam Review* 43, 1 (2001), 89–112. https://doi.org/10.1137/S003614450036757X

[9] Tobias Gysi, Tobias Grosser, and Torsten Hoefler. 2015. MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures. In *Proceedings of the 29th ACM on International Conference on Supercomputing (ICS '15)*. ACM, New York, NY, USA, 177–186. https://doi.org/10.1145/2751205.2751223

[10] Kevin Hamilton and Wataru Ohfuchi. 2008. *High Resolution Numerical Modelling of the Atmosphere and Ocean*. Springer, New York. https://doi.org/10.1007/978-0-387-49791-4

[11] Isaac M. Held and Max J. Suarez. 1994. A proposal for the Intercomparison of the Dynamical Cores of Atmospheric General Circulation Models. *Bulletin of the American Meteorological Society* 75, 10 (1994), 1825–1830. https://doi.org/10.1175/1520-0477(1994)075<1825:APFTIO>2.0.CO;2

[12] Norman A. Phillips. 1957. A coordinate system having some special advantages for numerical forecasting. *Journal of Meteorology* 14, 2 (1957), 184–185. https://doi.org/10.1175/1520-0469(1957)014<0184:ACSHSS>2.0.CO;2

[13] William M. Putman. 2007. *Development of the Finite-Volume Dynamical Core on the Cubed-Sphere*. PhD thesis, The Florida State University, Tallahassee, Florida. http://purl.flvc.org/fsu/fd/FSU_migr_etd-0511

[14] Michele Scquizzato and Francesco Silvestri. 2014. Communication Lower Bounds for Distributed-Memory Computations. In *31st International Symposium on Theoretical Aspects of Computer Science (STACS '2014) (Leibniz International Proceedings in Informatics (LIPIcs))*, Ernst W. Mayr and Natacha Portier (Eds.), Vol. 25. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 627–638. https://doi.org/10.4230/LIPIcs.STACS.2014.627

[15] Takashi Shimokawabe, Takayuki Aoki, Tomohiro Takaki, Toshio Endo, Akinori Yamanaka, Naoya Maruyama, Akira Nukada, and Satoshi Matsuoka. 2011. Petascale Phase-field Simulation for Dendritic Solidification on the TSUBAME 2.0 Supercomputer. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*. ACM, New York, NY, USA, Article 3, 11 pages. https://doi.org/10.1145/2063384.2063388

[16] Edgar Solomonik, Erin Carson, Nicholas Knight, and James Demmel. 2014. Tradeoffs Between Synchronization, Communication, and Computation in Parallel Linear Algebra Computations. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '14)*. ACM, New York, NY, USA, 307–318. https://doi.org/10.1145/2612669.2612671

[17] Edgar Solomonik, Erin Carson, Nicholas Knight, and James Demmel. 2017. Trade-Offs Between Synchronization, Communication, and Computation in Parallel Linear Algebra Computations. *ACM Transactions on Parallel Computing* 3, 1, Article 3 (Jan. 2017), 47 pages. https://doi.org/10.1145/2897188

[18] M. A. Taylor, J. Edwards, and A. St. Cyr. 2008. Petascale atmospheric models for the Community Climate System Model: new developments and evaluation of scalable dynamical cores. *Journal of Physics: Conference Series* 125, 1 (2008), 12023–12032. http://stacks.iop.org/1742-6596/125/i=1/a=012023

[19] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. 2005. Optimization of Collective Communication Operations in MPICH. *International Journal of High Performance Computing Applications* 19, 1 (2005), 49–66. https://doi.org/10.1177/1094342005051521

[20] Paul A. Ullrich, Peter H. Lauritzen, and Christiane Jablonowski. 2009. Geometrically Exact Conservative Remapping (GECoRe): Regular Latitude-Longitude and Cubed-Sphere Grids. *Monthly Weather Review* 137, 6 (2009), 1721–1741. https://doi.org/10.1175/2008MWR2817.1

[21] Ludwig Umscheid JR. and M. Sankar-Rao. 1971. Further tests of a grid system for global numerical prediction. *Monthly Weather Review* 99, 9 (1971), 686–690. https://doi.org/10.1175/1520-0493(1971)099<0686:FTOAGS>2.3.CO;2

[22] Yuzhu Wang, Jinrong Jiang, He Zhang, Xiao Dong, Lizhe Wang, Rajiv Ranjan, and Albert Y. Zomaya. 2017. A scalable parallel algorithm for atmospheric general circulation models on a multi-core cluster. *Future Generation Computer Systems* 72 (2017), 1–10. https://doi.org/10.1016/j.future.2017.02.008

[23] M. F. Wehner, J. J. Ambrosiano, J. C. Brown, W. P. Dannevik, P. G. Eltgroth, A. A. Mirin, J. D. Farrara, C. C. Ma, C. R. Mechoso, and J. A. Spahr. 1993. Toward a high performance distributed memory climate model. In *Proceedings of The 2nd International Symposium on High Performance Distributed Computing*. IEEE, Spokane, WA, USA, 102–113. https://doi.org/10.1109/HPDC.1993.263852

[24] David L. Williamson. 2007. The Evolution of Dynamical Cores for Global Atmospheric Models. *Journal of The Meteorological Society of Japan* 85B (2007), 241–269. https://doi.org/10.2151/jmsj.85B.241

[25] Wei Xue, Chao Yang, Haohuan Fu, Xinliang Wang, Yangtong Xu, Junfeng Liao, Lin Gan, Yutong Lu, Rajiv Ranjan, and Lizhe Wang. 2015. Ultra-Scalable CPU-MIC Acceleration of Mesoscale Atmospheric Modeling on Tianhe-2. *IEEE Trans. Comput.* 64, 8 (Aug 2015), 2382–2393. https://doi.org/10.1109/TC.2014.2366754

[26] Chao Yang, Wei Xue, Haohuan Fu, Lin Gan, Linfeng Li, Yangtong Xu, Yutong Lu, Jiachang Sun, Guangwen Yang, and Weimin Zheng. 2013. A Peta-scalable CPU-GPU Algorithm for Global Atmospheric Simulations. In *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '13)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/2442516.2442518

[27] He Zhang, Minghua Zhang, and Qingcun Zeng. 2013. 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. https://doi.org/10.1175/MWR-D-11-00367.1