TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 09/12 pp335–346
DOI: 10.26599/TST.2019.9010057
Volume 26, Number 3, June 2021

Helmholtz Solving and Performance Optimization in Global/Regional Assimilation and Prediction System

Jianqiang Huang, Wei Xue, Haodong Bian, Wenxin Yan, Xiaoying Wang, and Wenguang Chen*

Abstract: Despite efficient parallelism in the solution of physical parameterization in the Global/Regional Assimilation and Prediction System (GRAPES), the Helmholtz equation in the dynamic core, with the increase of resolution, can hardly achieve sufficient parallelism in the solving process due to a large amount of communication and irregular access. In this paper, optimizing the Helmholtz equation solution for better performance and higher efficiency has been an urgent task. An optimization scheme for the parallel solution of the Helmholtz equation is proposed in this paper. Specifically, the geometrical multigrid optimization strategy is designed by taking advantage of the data anisotropy of grid points near the pole and the isotropy of those near memory equator in the Helmholtz equation, and the Incomplete LU (ILU) decomposition preconditioner is adopted to speed up the convergence of the improved Generalized Conjugate Residual (GCR), which effectively reduces the number of iterations and the computation time. The overall solving performance of the Helmholtz equation is improved by thread-level parallelism, vectorization, and reuse of data in the cache. The experimental results show that the proposed optimization scheme can effectively eliminate the bottleneck of the Helmholtz equation as regards the solving speed. Considering the test results on a 10-node two-way server, the solution of the Helmholtz equation, compared with the original serial version, is accelerated by $100\times$, with one-third of iterations reduced.

Key words: Global/Regional Assimilation and Prediction System (GRAPES); Helmholtz equation; Generalized Conjugate Residual (GCR); performance optimization; Incomplete LU (ILU)

1 Introduction

It is the mainstream for atmospheric scientific research and numerical weather prediction model to establish a high-resolution fine numerical weather prediction model. The Global/Regional Assimilation and Prediction System (GRAPES) is a new-generation numerical weather prediction system independently developed by China. This model adopts internationally advanced numerical forecasting technology and is embedded with an extremely complicated communication model and computation process with its codes reaching up to hundreds of thousands of lines.

The GRAPES global model uses the 2D horizontal domain partition, and each decomposed subdomain is handled by a Message Passing Interface (MPI) process. Figure 1 is a computation flow chart of the GRAPES global model that shows how the GRAPES global model integrates into a time step. Firstly, the linear and nonlinear terms are computed and the water vapor equation is solved by the piecewise rational method based on the piecewise rational function[1]. Secondly, the position and interpolation of the departure point are

• Jianqiang Huang, Wei Xue, and Wenguang Chen are with Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: hjq16@mails.tsinghua.edu.cn; xuewei@tsinghua.edu.cn; cwg@tsinghua.edu.cn.
• Jianqiang Huang, Haodong Bian, Wenxin Yan, and Xiaoying Wang are with Department of Computer Technology and Applications, Qinghai University, Xining 810016, China. E-mail: hpc_bhd@163.com; yanwenxin1@126.com; xy.wang@foxmai.com.
* To whom correspondence should be addressed.
Manuscript received: 2019-10-10; accepted: 2019-10-17

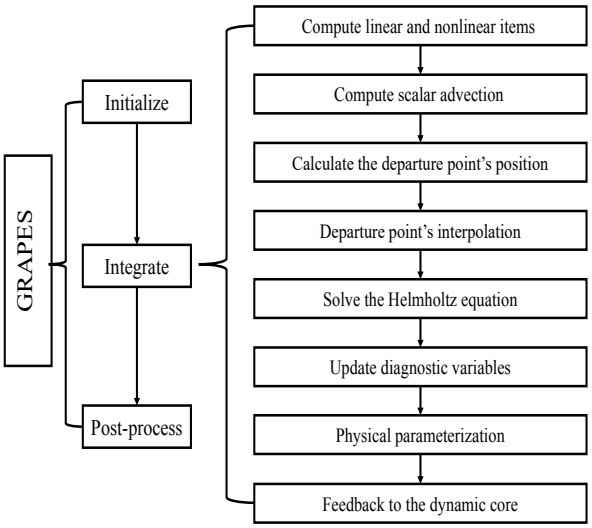


Fig. 1 Computation flowchart of GRAPES global model.

calculated by the semi-Lagrangian method. Next, the GRAPES global model solves the Helmholtz equation and updates other prognostic variables. At this point, the calculation of the dynamic core ends. Finally, the model calculates a series of physical parameterization schemes and sends the results to the dynamic core.

As the center of the GRAPES global model, the dynamic core consumes over half of the computing time of the whole model (as shown in Fig. 2). The communication of the dynamic core mainly relies on the boundary exchange of neighbors and the global protocol communication Allreduce in the Helmholtz equation solver. Before the calculation by the grid points of the boundary area, the specified weather parameters should be used to exchange the boundary data. If Q represents the number of grid points in the GRAPES global model, the time complexity of the algorithm for most modules including the semi-Lagrangian departure point calculation is $O(Q^2)$, and that of the algorithm for solving the Helmholtz equation is $O(Q^3)$. The GRAPES global model, like the currently advanced numerical weather prediction models in other countries

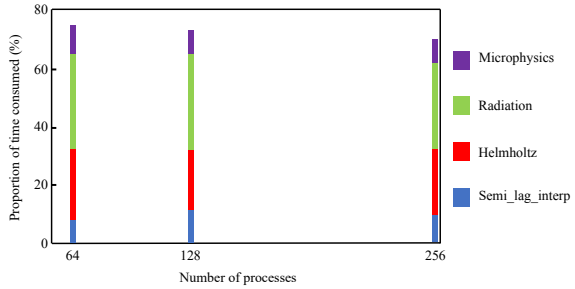


Fig. 2 Proportion of time consumed by all hotspots in GRAPES.

and institutions, is moving toward higher resolution and more complex physical parameterization solutions. As planned, by 2020, a commercial operation with a horizontal resolution of 0.1 degree will be achieved, with the forecasting timeliness remaining unchanged, which poses extremely high requirements for the calculation speed of the model.

As high-performance computers see increasingly enhanced computing power and expanded scale, it is an urgent task for the GRAPES model to fully utilize the computing power provided by the high-performance computing system to optimize its computational performance, realize scalability under massively parallel tasks, and solve huge scientific problems and massive data processing to meet the time limit in actual numerical forecasting. In this paper, we analyze in-depth the main factors affecting the parallel computing efficiency of the GRAPES model, focus on the efficient solving algorithm of the Helmholtz equation that performs key computation in the GRAPES model (with its time consumption accounting for 30% of that of the dynamic core), and adopt 19-diagonal incomplete LU factorization with zero fill-in (ILU(0)) preconditioner to accelerate convergence. Additionally, a parallel computing scheme for the Helmholtz equation is presented, together with an optimized scheme for the key impediments to the computational efficiency, which includes the following three challenges:

- The data storage is excessively large in scale. The coefficient matrix of the Helmholtz equation is an extra-large-scale matrix with extremely high sparsity, with each row containing a maximum of 19 non-zero elements, which causes great storage waste. Therefore, the compressed sparse row data storage format is used in this paper to store the coefficient matrix so as to reduce redundant memory access during the computational process.
- In the traditional Generalized Conjugate Residual (GCR) algorithm, each loop iteration requires two global communications. When the algorithm is extended to a larger scale, the proportion of communications will increase sharply. To this end, we design an optimization strategy for communications, which mainly includes communication avoidance, reduction in global communications, and overlapping of calculation and communications, ultimately reducing communication overhead and improving scalability.
- The multiple-iteration calculation in the GCR algorithm mainly includes the coefficient matrix-vector

multiplication and the sparse triangle algorithm that can be parallelized. In this paper, through multi-core technologies, such as OpenMP and MPI, parallelism is realized and further optimized, improving the overall solving efficiency of the Helmholtz equation.

The experimental results show that the optimized solving module of the Helmholtz equation achieves a speedup by 100× on both data scales of $360 \times 180 \times 38$ and $720 \times 360 \times 38$.

2 Background

Numerical weather prediction is closely related to high-performance parallel computers. With the development of the latter, parallel computing has become an integral part of numerical weather prediction systems[2–7]. Table 1 lists the business numerical forecasting systems of main meteorological departments that adopt parallel computing.

The Weather Research and Forecasting (WRF)[8] model is a mesoscale model and assimilation system jointly developed by scientists from various meteorological research departments and universities in USA and it adopts the regional grid pattern. In terms of parallel computing, the grid point pattern is much simpler than the spectral pattern, for the physical parameterization of the grid point pattern mainly calculates the vertical correlation. The WRF model uses horizontal two-dimensional data partitioning for parallel computing. That is, all computational variables are divided according to a given number of processors in both warp and latitude directions, and each processor only calculates the data in a sub-region designated to itself. Considering the data-dependence infinite difference and horizontal interpolation calculations, the storage area of each processor usually adds a certain-width boundary to the calculation area, called the halo area. The data in the halo area are computed by adjacent calculation processors, and the data should be synchronized for each step.

The ultimate goal of parallel computing for numerical weather prediction is to meet the real-time nature of business operations. Faced with the huge computing resources from high-performance computers, how to effectively use these resources has become a challenge in the research on parallel computing for numerical forecasting. The scalability of parallel computing is the goal pursued by high-performance computers and parallel algorithms[9] as well as the numerical weather prediction systems in parallel computation. The load balance in parallel computing is the key to parallel scalability and the bottleneck of parallel computing in numerical weather prediction[10–12]. Due to the complexity of atmospheric motion, the physical phenomena in different regions are different. Regarding the model computing, the difference in regions and physical phenomena and changes with the movement of the atmosphere result in different calculation amounts of the model. Additionally, the study of efficient methods for core computing is also an important aspect of ensuring real-time numerical weather prediction services and parallel computing scalability[13, 14]. Even from an economic viewpoint, studies on efficient computational methods are essential. Michalakes and Vachharajani[15] firstly studied the migration of WRF from the traditional parallel cluster to GPU, concluding that the running speed on NVIDIA 8800GTX is 20 times that based on CPU operation[16], due to the limitation of the high-time cost of the WSM5 module in WRF, the WSM5 module was rewritten to CUDA and transplanted to the GPU for separate operation, and no optimization actions were performed. For Ref. [17], its main contribution is that developers, who do not have any knowledge of the

Table 1 High-Performance Computing (HPC) trends for large installations of Earth System Model (ESM) Community, where PF represents Pflops.

Category	Organization	Location	Model	Previous HPC	Current HPC	Size/Cost (million)/Date
Operational and research	ECMWF	Reading, UK	IFS	IBM power	Cray XC30-x86	3.5 PF/$65/Jun 2013
	Met office	Exeter, UK	UM	IBM power	Cray XC30-x86	16 PF/$120/Oct 2014
	DWD	Offenbach, DE	COSMO, ICON	NEC SX-9	Cray XC30-x86	2 PF/$23/Jan 2013
	MF	Toulouse, FR	Arpege, Arome	NEC SX-9	Bull-x86	5 PF/$36/Nov 2012
	NOAA NCEP	Various, USA	GFS, HRRR/WRF	IBM power	IBM Cray XC50-x86	5.8 PF/$50/Oct 2015
	Env Canada	Montreal, CA	GEM-YY, WRF	IBM power	IBM Cray XC40-x86	2.4 PF/$50/Nov 2017
	JMA	Tokyo, JP	GSM, ASUCA	Hitachi power	Hitachi Cray	∼4 PF/2018
Research	DKRZ/MPI-M	Hamburg, DE	ICON, MPI-ESM	IBM power	Bull-x86	3 PF/$35/May 2014
	NCAR	Boulder, CO, USA	CESM, WRF, MPAS	IBM iDataPlex	SGI ICE XA-x86	5.34 PF/∼$60/Jan 2016
	NOAA	Fairmont, WV, USA	FV3, Radiation	Various	Cray CS-Storm-x86	∼4 PF/Jan 2016

application transplant theory, can use a simple iterative method to migrate complex scientific applications to the GPU in a progressive manner, thus reducing system overhead. Nevertheless, performance optimization was not considered. The authors in Ref. [18] mainly discussed how to transplant the Rapid Radiative Transfer Model (RRTM) module in WRF to the GPU from the aspect of performance optimization. It's main optimization strategies include the following: (1) modifying the data structure; (2) optimizing the data transfer; (3) splitting different kernel codes and kernel configurations. Since the RRTM module relies heavily on lookup tables, there is a robust inter-data correlation. Thus, using different types of memory will vitally affect the system performance.

3 Design of Parallelism Scheme for GRAPES Model

To solve the large-scale problem in parallel computing, decomposing the problem area to explore the parallelism potential of multiple processors is firstly necessary. As the GRAPES model adopts a global longitude-latitude grid system, the grid space can be divided along the longitude, latitude, and vertical directions during parallel processing. In general, for large-scale isotropic parallel communications, 3D partitioning can achieve better load balancing and parallelism than 1D and 2D partitioning. However, in numerical weather prediction, the role of atmospheric vertical motion is critical to the prediction of the entire model. From the entire life cycle of the cumulonimbus to the classical three-circle ideal model, vertical motion is a focus of atmospheric science, for it is directly related to many key processes, such as the gas expansion process and latent heat transfer in water vapor heterogeneous condensation. Therefore, a subdivision in the vertical direction will result in the simulative vertical motion of intensive communications between adjacent processes, which will greatly increase communication overhead and inevitably limit the model scalability. For the above reasons, GRAPES adopts a two-dimensional area decomposition scheme, which divides the whole problem space into several subareas along the longitude and latitude directions, with each subarea handled by one process. This partitioning scheme implements load balance on the division of the mesh number in the computation of GRAPES.

As shown in Fig. 3, ims represents the start address of the storage area, which stores the array spaces of

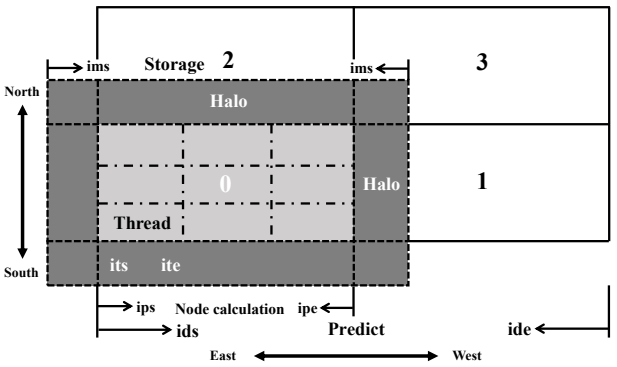


Fig. 3 Two-dimensional areas partitioning[19].

the patch area and the overlap area; its represents the start address of the tile area, which is divided in the patch area according to the number of available threads of this area; ite represents the end address of the tile area; ips represents the start address of the patch area, which is a calculation sub-area that divided the global forecast area according to the number of computer processors; ipe represents the end address of the patch area; ids represents the start address of the entire global forecast domain; ide represents the end address of the entire global forecast domain. To make the model of parallel computing adaptable to the development of modern parallel computer architecture, the parallel implementation of GRAPES was considered from two aspects: One is the distributed memory computer system patch area. There is need to calculate the whole area in accordance with the number of divisions. The division of a subarea is called a patch. Each computer node only completes the patch calculation and only the corresponding patch + halo array space. When the calculation involves other areas, information exchange in the corresponding halo area must be carried out in advance. The other is the tile computation performed on a shared memory computer system, commonly known as multithreaded parallel computing. The tile is divided into regions within a patch according to the number of available threads of this node. Therefore, memory sharing between different tiles on the same node is different only in calculation regions. The two parallel computing solutions of the regional mode are complementary to each other and can be used independently or in combination for different computer systems.

3.1 Optimization of the solution of Helmholtz equation

GRAPES performs excellently in improving the

forecasting accuracy, the available forecasting timeliness, and the fineness of forecasting, with the fineness of 1–2 km and the available grid range of 1–100 km. For such a multi-scale unified model, parallel computing involves the calculations of both the regional and global model and considers not only low resolution but also HPC efficiency under an extremely large grid size at the fine resolution. As the Helmholtz equation contains the most complicated calculation in GRAPES, we firstly focus on the solution of the three-dimensional Helmholtz discrete grid equation. It involves 19 points at three layers, with five grid points each along the three directions of x, y, and z. In terms of the boundary, one grid point is reduced at each end of the three directions. Because the earth is an approximate sphere and it both rotates and revolves, the motion of the atmosphere on the earth's surface is relative to the motion of the earth's surface. A stationary coordinate system, such as the Cartesian coordinate system, is no longer applicable. The basic equations of atmospheric motion are given as follows:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{C_p\theta}{r\cos\varphi}\frac{\partial\prod}{\partial\lambda} + fv + F_u + \delta_M\left\{\frac{u\cdot v\cdot\tan\varphi}{r} - \frac{u\cdot w}{r}\right\} - \delta_\varphi\left\{f_\varphi w\right\} \quad (1)$$

where u, v, w, $\prod$, C_p, θ, r, λ, f, F_u, δ_M, φ, δ_φ, and $f_\varphi w$ represent the x direction horizontal wind speed component, y direction horizontal wind speed component, the vertical wind speed component in z coordinates, Exner pressure variable, specific heat of air at constant pressure, potential temperature, the vector in spherical coordinates, longitude of spherical coordinates, coriolis force, turbulent diffusion, curvature correction term, latitude in spherical coordinates, earth deflection force correction term, and Coriolis force derivative, respectively. Bringing in intermediate pressure variable at the same time,

$$\frac{\mathrm{d}v}{\mathrm{d}t} = -\frac{C_p\theta}{r}\frac{\partial\prod}{\partial\lambda} - fu + F_v - \delta_M\left\{\frac{u^2\cdot\tan\varphi}{r} + \frac{u\cdot w}{r}\right\} \quad (2)$$

$$\delta_{\mathrm{NH}}\frac{\mathrm{d}w}{\mathrm{d}t} = -C_p\theta\frac{\partial\prod}{\partial r} - g + F_w + \delta_M\left\{\frac{u^2+v^2}{r}\right\} + \delta_\varphi\left\{f_\varphi u\right\} \quad (3)$$

$$(\gamma-1)\frac{\mathrm{d}\prod}{\mathrm{d}t} = -\prod\cdot D_3 + \frac{F_\theta^*}{\theta} \quad (4)$$

$$\nabla\cdot V = \frac{1}{r\cos\varphi}\frac{\partial u}{\partial\lambda} + \frac{1}{r\cos\varphi}\frac{\partial v\cos\varphi}{\partial\varphi} + \frac{1}{r^2}\frac{\partial\left(r^2w\right)}{\partial r} \quad (5)$$

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{F_\theta^*}{\prod} \quad (6)$$

$$f = 2\Omega\sin\varphi \quad (7)$$

$$f_\varphi = \partial f/\partial\varphi = 2\Omega\cos\varphi \quad (8)$$

$$\gamma = \frac{C_p}{R_2} \quad (9)$$

$$D_3 = +\nabla\cdot V \quad (10)$$

where δ_{NH} represents the vertical acceleration switch, D_3 represents the three-dimensional divergence, V represents three-dimensional wind speed vector, Ω represents the Angular velocity of the earth's rotation, R_2 represents the dry gas constants, g represents the top height, and F_θ^* represents the net reservoir heat sink term.

The basic equations for controlling atmospheric motion are Eqs. (1)–(6). After simplifying the reference atmosphere semi-implicit, semi-Lagrangian discretization, and separation of variables, the basic prediction equations of the model dynamic framework[19, 20] are obtained:

$$u^{n+1} = \left[\xi_{u_1}\frac{1}{a\cos\phi}\frac{\partial}{\partial\lambda} + \xi_{u_2}\frac{1}{a}\frac{\partial}{\partial\phi} + \xi_{u_3}\frac{\partial}{\partial\hat{z}}\right]\cdot\left(\prod'\right)^{n+1} + \xi_{u_0} \quad (11)$$

$$v^{n+1} = \left[\xi_{v_1}\frac{1}{a\cos\phi}\frac{\partial}{\partial\lambda} + \xi_{v_2}\frac{1}{a}\frac{\partial}{\partial\phi} + \xi_{v_3}\frac{\partial}{\partial\hat{z}}\right]\cdot\left(\prod'\right)^{n+1} + \xi_{v_0} \quad (12)$$

$$\hat{w}^{n+1} = \left[\xi_{w_1}\frac{1}{a\cos\phi}\frac{\partial}{\partial\lambda} + \xi_{w_2}\frac{1}{a}\frac{\partial}{\partial\phi} + \xi_{w_3}\frac{\partial}{\partial\hat{z}}\right]\cdot\left(\prod'\right)^{n+1} + \xi_{w_0} \quad (13)$$

$$(\theta')^{n+1} = \left[\xi_{\theta_1}\frac{1}{a\cos\phi}\frac{\partial}{\partial\lambda} + \xi_{\theta_2}\frac{1}{a}\frac{\partial}{\partial\phi} + \xi_{\theta_3}\frac{\partial}{\partial\hat{z}}\right]\cdot\left(\prod'\right)^{n+1} + \xi_{\theta_0} \quad (14)$$

$$\left(\prod'\right)^{n+1} = \left[\frac{\xi_{\prod 1}}{a\cos\phi}\frac{\partial}{\partial\lambda} + \frac{\xi_{\prod 2}}{a}\frac{\partial}{\partial\phi} + \xi_{\prod 3}\frac{\partial}{\partial\hat{z}} + \frac{\xi_{\prod H_1}}{(a\cos\phi)^2}\frac{\partial^2}{\partial^2\lambda} + \frac{\xi_{\prod H_2}}{a^2}\frac{\partial^2}{\partial^2\phi} + \xi_{\prod H_3}\frac{\partial^2}{\partial^2\hat{z}} + \frac{\xi_{\prod H_4}}{a^2\cos\phi}\frac{\partial^2}{\partial\phi\partial\lambda} + \frac{\xi_{\prod H_5}}{a\cos\phi}\frac{\partial^2}{\partial\lambda\partial\hat{z}} + \frac{\xi_{\prod H_6}}{a}\frac{\partial^2}{\partial\phi\partial\hat{z}}\right]\left(\prod'\right)^{n+1} + \xi_{\prod 0} \quad (15)$$

where ξ_{u_1}, ξ_{u_2}, ξ_{u_3}, ξ_{v_1}, ξ_{v_2}, ξ_{v_3}, ξ_{w_1}, ξ_{w_2}, ξ_{w_3}, $\xi_{\prod H_1}$, $\xi_{\prod H_2}$, $\xi_{\prod H_3}$, $\xi_{\prod H_4}$, $\xi_{\prod H_5}$, $\xi_{\prod H_6}$, $\xi_{\prod 1}$, $\xi_{\prod 2}$, $\xi_{\prod 3}$, ξ_{θ_1}, ξ_{θ_2}, and ξ_{θ_3} represent the coefficients of the equation

that do not change with time. ξ_{u_0}, ξ_{v_0}, ξ_{w_0}, $\xi_{\prod 0}$, and ξ_{θ_0} represent the coefficient terms at the previous moment that change with time. a represents radius of the earth, ϕ represents the spherical curvature, and n represents the coefficient term at the previous moment that changes with time and at some point.

We firstly focus on the solution of the three-dimensional Helmholtz discrete grid equation. As shown in Fig. 4, it involves 19 points at three layers, with five grid points each along the three directions of x, y, and z. In terms of the boundary, one grid point is reduced at each end of the three directions.

The coefficient matrix of the Helmholtz equation is a large-scale sparse matrix, and its order equals the total number of all discrete grid points in all directions in the three dimensions. Due to a very sparse nature of the whole matrix, each row (column) has only 19 non-zero elements distributed on 19 diagonal lines that are symmetrical with respect to the main diagonal of the matrix. Such a sparse matrix with a diagonal structure is of great significance both in storage and computation. In this paper, the GCR iterative method is used as the basic method for solving the Helmholtz equation. The solution process with the preconditioner is shown in Algorithm 1.

In each iteration step of the GCR method, the computational core is the sparse matrix-vector multiplication at the third row, the preconditioner at the seventh row, the vector dot product at the third, eighth, ninth, and tenth rows, and some other vector multiplications. In the case of a parallel solution, one neighbor communication is required before each sparse matrix-vector multiplication to fill the boundary data

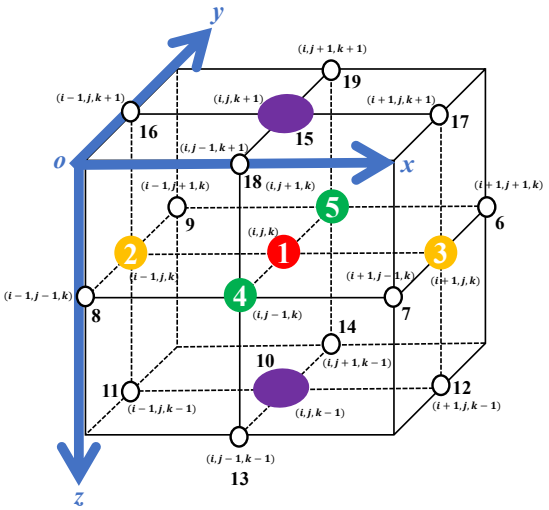


Fig. 4 Distribution of discrete grid points and correlation coefficients of three-dimensional Helmholtz equation[21].

Algorithm 1 GCR solution process with the preconditioner

Input: Matrix A, initial solution S_0, the right vector b, the inverse of the preprocessing matrix M^{-1}, subspace dimension k, and the initial residual r_0

Output: approximate solution S

1: $r_0 = b - AS_0$, $r'_0 = M^{-1}r_0$, $p_0 = r'_0$
2: **for** i = 1: Maximum number of iterations **do**
3: $\alpha_{i-1} = (r_{i-1}, Ap_{i-1})/(Ap_{i-1}, Ap_{i-1})$
4: $S_i = S_{i-1} + \alpha_{i-1}p_{i-1}$
5: $r_i = r_{i-1} - \alpha_{i-1}Ap_{i-1}$
6: **if** converged **then** exit;
7: $r'_i = M^{-1}r_i$
8: **for** $j = \text{int}[\frac{i-1}{k}]k, ..., i-1$ **do**
9: $\beta_{ij} = -(Ar'_i, Ap_j)/(Ap_j, Ap_j)$
10: $p_i = r'_i + \sum_{j=\text{int}[\frac{i-1}{k}]k}^{i-1} \beta_{ij}p_j$
11: $Ap_i = Ar'_i + \sum_{j=\text{int}[\frac{i-1}{k}]k}^{i-1} \beta_{ij}Ap_j$

of the sub-region calculated by each process. After each process performs the local vector dot product, all processes carry out global Allreduce communication to obtain a global dot product result.

3.2 Preconditioner optimization

In an iterative algorithm, the preconditioner $M^{-1}S = b$ is critical[22]. A proper preconditioner can greatly accelerate the iterative convergence process at a fair implementation cost. As for the Helmholtz equation solver of the GRAPES model, Incomplete LU (ILU) factorization is used as the preconditioner. Incomplete LU factorization, i.e., an approximation of LU factorization, is the decomposition of matrix A into an upper triangular matrix U and a lower triangular matrix L. As shown in Eq. (16), the residual matrix R obtained by the difference between product and the original matrix meets certain requirements, such as the same location of non-zero elements as that of A. Moreover, ILU can be divided into ILU (p), $p = 0, 1, 2, \ldots$, and other forms according to the non-zero element fill-in during factorization, where ILU(0) refers to the ILU model without any filling. In other words, the distribution of the non-zero elements of the residual matrix R is the same as that of the original matrix A, and the choice of ILU form depends on the computational overhead and convergence performance.

$$R = L \cdot U - A \tag{16}$$

The calculation process of the ILU preconditioner is to firstly perform ILU on the original matrix A ($A \approx L \cdot U$)

to get $LUS = b$, and then solve it in each iteration step. For each solution process, the lower triangular matrix of the previous generation is calculated to obtain $US = L^{-1}b$, and then the upper triangular matrix of the back-generation is computed to get $S = U^{-1}(L^{-1}b)$, where S is the approximate solution obtained.

There are many options of the generation of the ILU preconditioner matrix according to the characteristics of the diagonal elements of the coefficient matrix A. By analysis, when the order of magnitudes of the coefficient of the center grid points at the vertical layer $x(i, j, k)$, $x(i, j, k-1)$, and $x(i, j, k+1)$ reaches about $10^{-1} - 1$, the order of magnitudes of the coefficient of $x(i-1, j, k)$, $x(i, j-1, k)$, $x(i, j+1, k)$, and $x(i+1, j, k)$ is about 10^{-3} , and those of other grid points are about 10^{-7}. In the original GRAPES model, the three-diagonal matrix with the largest order of magnitudes was used as the preprocessing matrix. Applying it to the GCR iterative method can accelerate the convergence, but the preconditioner can be further improved[23]. We discuss the effect of the 7-diagonal matrix and the 19-diagonal matrix as the coefficient matrix of the ILU preconditioner on the entire GCR iterative algorithm.

Figure 5 shows the residual variations of the GCR algorithm when a 7-diagonal matrix and a 19-diagonal matrix were applied to the ILU preconditioner. It can be seen that when the convergence criterion was set as 1×10^{-12}, the entire GCR iteration converged at Step 225 when the 7-diagonal ILU preconditioner was used, and at Step 134 when the 19-diagonal ILU preconditioner was used. The more complicated preconditioner accelerates the convergence but costs higher in computation. In large-scale calculations, due to a small number of grid points handled by each process, the calculation is no longer the focus. The overhead of global communication

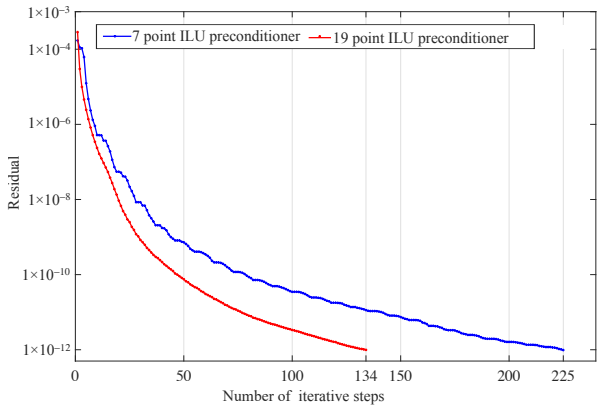


Fig. 5 Residual curve of 7-diagonal and 19-diagonal preconditioners[21].

is not related to the number of grid points but grows rapidly as the number of processes increases. The 19-diagonal ILU preconditioner can greatly reduce the total number of iteration steps, thereby reducing the number of global communication steps, which is very helpful for the overall performance improvement of the GCR algorithm.

3.3 Improved generalized conjugate residual algorithm

As can be seen from Algorithm 1, there are two vector dot product operations in each iteration step of the GCR algorithm, found at Rows 3, 8–10, which need to be implemented by global Allreduce communication in the parallel algorithm. The global Allreduce communication has huge overhead during massive parallelism. Considering the work of Zhao and Tian[24], we implement an improved GCR algorithm with preconditioners that can reduce the number of global Allreduce communication steps from two to one. The Krylov subspace algorithm[25] characterizes a feature whereby the vectors that make up the subspace are orthogonal to each other. Based on this property, the principle of the improved GCR is to deduce the equivalent mathematical formula based on the orthogonality of the vector Ap_j in the GCR algorithm to obtain the vector dot product (Ap_i, Ap_j) of the original algorithm through the iteration of floating-point number at Row 14 in Algorithm 2, where M^{-1} represents the inverse of the preprocessing matrix; r_0 represents the initial residual; k represents subspace dimension; and α_0 represents the initial inner product. Compared with the original GCR algorithm, the improved GCR algorithm maintains the same convergence characteristics while reducing the number of global Allreduce communication steps by once at the cost of adding several floating-point numbers; thus, the improved algorithm outperforms the original in both performance and scalability.

4 Optimization

4.1 Thread-level parallelism and vectorization

To solve the Helmholtz equation, 2D or 3D grid points were processed. Specifically, in the first step of parallel, the OpenMP primitive was used to divide the computational tasks into different threads. For multi-core CPU, the threads in the same number as the physical cores provided good scalability. In this paper, as the platform contained two CPUs, the memory access had a non-uniform memory access architecture effect. The

Algorithm 2 Improved GCR iterative method with preconditioners

Input: Matrix A, initial solution S_0, the right vector b, the inverse of the preprocessing matrix M^{-1}, subspace dimension k, the initral residual r_0, and the initial inner product α_i

Output: approximate solution S

$r_0 = b - AS_0$, $r'_0 = M^{-1}r_0$, $p_0 = r'_0$, $\alpha_0 = (r_0, q_0)$, $\gamma_0 = (q_0, q_0)$

for $i = 1$: Maximum number of iterations **do**

  $\alpha_{i-1} = \alpha_{i-1}/\gamma_{i-1}$

  $S_i = S_{i-1} + \alpha_{i-1}p_{i-1}$

  $r_i = r_{i-1} - \alpha_{i-1}q_{i-1}$

  if converged **then** exit;

  $r'_i = M^{-1}r_i$

  $ar = Ar'_i$

  $\alpha_i = (r_i, ar)$

  $c_i = (ar, ar)$

  for $j = \text{int}[\frac{i-1}{k}]k, ..., i-1$ **do**

    $\beta_{ij} = -(ar, q_j)/(q_j, q_j)$

  $\gamma_i = c_i - \sum_{j=\text{int}[\frac{i-1}{k}]k}^{i-1} \beta_{ij}^2 \gamma_j$

  $p_i = r'_i + \sum_{j=\text{int}[\frac{i-1}{k}]k}^{i-1} \beta_{ij} p_j$

  $Ap_i = ar + \sum_{j=\text{int}[\frac{i-1}{k}]k}^{i-1} \beta_{ij} q_j$

numactl was adopted to increase bandwidth utilization and enhance performance. Next, the vectorization component in the kernel achieved the parallelism of the second step. Vectorization and better performance were effectively achieved by adding Single Instruction Multiple Data (SIMD) single instruction multiple data to compile the guiding statement in the program, or by overriding some of the looping codes.

4.2 Optimization of data reuse

When all the processor cores and vectorization components were used to maximize the parallelism of the program, memory requirements became our bottleneck. With better data layout, we could better reuse data and improve performance in a multi-tier cache. The common way to optimize cache usage is to partition the used matrix so that each thread continuously accesses the data suitable for each layer of caches. When access is a bottleneck for performance, proper partitioning brings significant enhancement.

5 Experimental Result

5.1 Experimental platform

There were 10 Intel Xeon E5-2680 v3@ 2.50 GHz CPUs (Sandy Bridge architecture) nodes. Each Intel Xeon E5-2680 CPU was equipped with 12 cores running at 2.5 GHz, and each core had two vector processing elements that could perform 256-bit floating-point operations. The peak computing power of the floating points of the CPU was 1.2 Tflops (for double-precision, there was one multiplication and one addition floating-point arithmetic component; for single precision, each calculation unit could store 4 floating points with single precision and perform two calculations due to the adoption of Streaming SIMD Extensions (SSE) with a length of 128 bit, both double-precision and single-precision presented high parallelism). To approach and reach a theoretical peak, we must make full use of all computing resources. Each core of the multicore CPU had L1 data cache of 32 KB, L1 instruction cache of 32 KB, and L2 cache of 256 KB. Twelve cores shared an L3 cache of 20 MB that enabled quick data exchange among processor cores. Each processor was connected to DDR3's memory via a 4-channel memory controller, which provided nearly 100 GB/s of memory bandwidth.

5.2 Analysis of experimental result

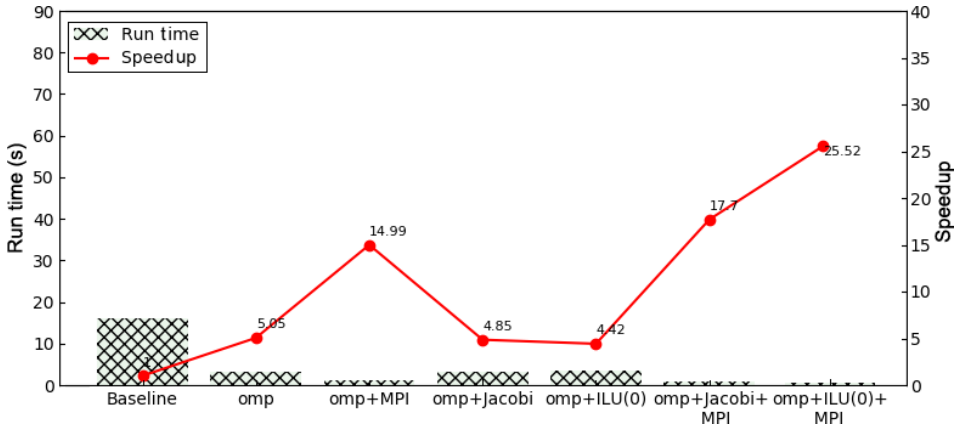
The single-process version of GCR was firstly implemented; then the multi-process version was extended, and the preconditioners of ILU and Jacobi were added to speed up the convergence. The GCR algorithm had data sizes of $360 \times 180 \times 38$ and $720 \times 360 \times 38$. Taking the GCR iteration time of two cases as benchmark, the run time and the speedup of the benchmark are shown in Fig. 6.

Jacobi preconditioner was implemented as a comparison. After two Jacobi iterations, $R' = R - (L + U)R$.

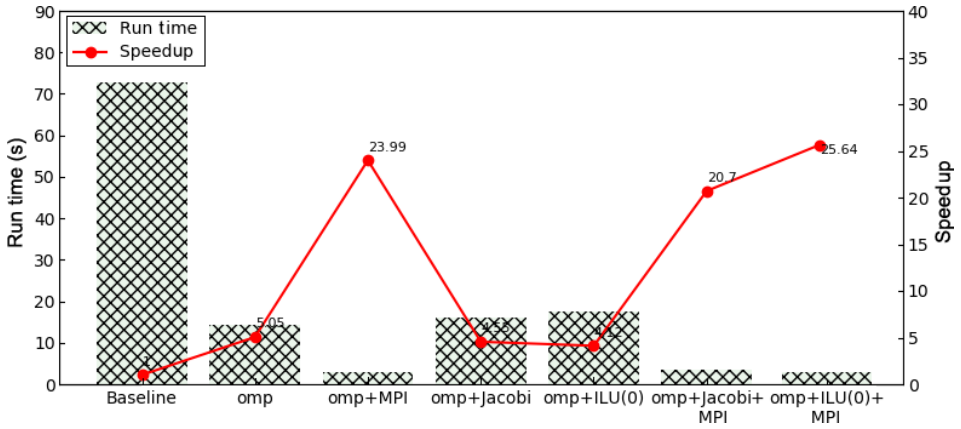
Table 2 shows that for data sizes of $360 \times 180 \times 38$ and $720 \times 360 \times 38$, the use of ILU(0) preconditioner, the optimized GCR algorithm with the number of processes $n = 20$, the number of nodes $N = 2$, and the number of threads $c = 2$ can achieve the best performance with the run time of 0.6298 and 2.932 s.

Table 3 shows the scalability analysis results. We can see that the increase of computing nodes presents linear speedup, although the multi-computer communication is a huge cost. The proposed improvement of GCR to reduce communication obtains outstanding scalability.

Comparison of time consumed for the test cases: It can be seen that in each case test, the actual iteration time is better than the benchmark iteration time, indicating the series of optimizations yield satisfactory results.



(a) Data size of 360 × 180 × 38



(b) Data size of 720 × 360 × 38

Fig. 6 Comparison of run time and speedup for different optimization methods under different data sizes.

Table 2 Comparison of time consumed for the cases in tests.

Data size	Benchmark time (s)	Actual time (s)	Number of iterations
360 × 180 × 38	16.07	0.62	29
720 × 360 × 38	76.23	2.86	33

Table 3 Scalability analysis results.

Number of nodes	Data size	Time (s)	Speedup
2	360 × 180 × 38	0.62	25
2	720 × 360 × 38	2.86	25
4	360 × 180 × 38	0.34	48
4	720 × 360 × 38	1.49	48
10	360 × 180 × 38	0.16	121
10	720 × 360 × 38	0.61	118

Finally, the speedup of each algorithm optimized in the Helmholtz equation solution in the 25 km case was compared with that of the original algorithm. After a series of parallel optimization techniques, such as OPENMP, MPI, and precondition, the improved GCR algorithm always outperformed the original GCR algorithm under each parallel scale.

6 Related Work

The Krylov subspace method is essential for solving large-scale sparse linear equations, which are widely used in scientific applications, such as numerical weather prediction[7] and ocean simulation[21]. Linear systems to be solved are usually obtained by the discretization of partial differential equations by implicit or semi-implicit time-integration schemes. With the continuous improvement of people's living standards and the advancement of science and technology, the requirements for the analog resolution of such applications are getting increasingly higher, and thus, high-performance computers are urgently needed. As parallelism improves, the Krylov subspace approach presents a new performance bottleneck, namely, frequent global communication. To better utilize the computing resources available in modern clusters, recent research has proposed algorithms that mitigate the performance pressures of global communications. Zhao and Tian[24] proposed an improved parallel GCR algorithm that

can reduce global communication in each iteration step. The pipeline's Krylov subspace method[25, 26] allows the overlapping of global communication, sparse matrix-vector multiplication, preconditioner, and other calculation cores. These methods reduce the impact of global communication to a certain extent but do not solve the problem in essence. At the extreme scale, global communication will still be the main factor limiting the algorithm performance. On this basis, a Krylov subspace method was proposed for communication avoidance[27–29]. The method, based on the s-Step Krylov subspace algorithm[30], reduces the number of global communication steps to $1/s$ of the original at the cost of additional calculations. There are many methods for solving the Helmholtz equation, such as relaxation iteration, multigrid method, and GCR. Among them, the GCR method converges quickly and is easy to implement; thus, it is widely used in solving the Helmholtz equation. The GCR algorithm is used in the GRAPES model, with details seen in Ref. [31]. Regarding the choice of iterative methods, Lin[32] discussed and compared the performance of GCR, GMRES, Bi-CGSTAB algorithm, and IDR[33], and finally concluded the GCR algorithm performed best with respect to the numerical stability, convergence property, and computation amount.

7 Conclusion

In this paper, the solution of the Helmholtz equation was optimized in terms of the time consumption of the GRAPES model, and the effects of different forms of ILU preconditioner and different regional partition schemes on the convergence effect of the iterative solution algorithm were discussed. The 19-diagonal incomplete preconditioner was found to result in better convergence despite more complex operations. Additionally, an improved GCR algorithm is proposed, which reduces the number of global communication steps in each iteration from two to one. The optimized solving module of the Helmholtz equation achieves acceleration by up to 100× with different parallelisms in different cases.

Acknowledgment

This paper was partially supported by the Open Project of State Key Laboratory of Plateau Ecology and Agricuture, Qinghai University (No. 2020-ZZ-03), the Qinghai Province High-End Innovative Thousand Talents Program Leading Talents, the National Natural Science Foundation of China (Nos. 61762074 and 61962051), and the National Natural Science Foundation of Qinghai Province (No. 2019-ZJ-7034).

References

[1] Y. Su, X. S. Shen, X. D. Peng, X. L. Li, X. J. Wu, S. Zhang, and X. Chen, Application of PRM scalar advection scheme in GRAPES global forecast system, (in Chinese), *Chin. J. Atmos. Sci.*, vol. 37, no. 6, pp. 1309–1325, 2013.

[2] T. Yanagawa and K. Suehiro, Software system of the earth simulator, *Parallel Comput.*, vol. 30, no. 12, pp. 1315–1327, 2004.

[3] S. Habata, K. Umezawa, M. Yokokawa, and S. Kitawaki, Hardware system of the earth simulator, *Parallel Comput.*, vol. 30, no. 12, pp. 1287–1313, 2004.

[4] H. Ishizaki and I. Ishikawa, High parallelization efficiency in barotropic-mode computation of ocean models based on multi-grid boundary ghost area, *Ocean Modelling*, vol. 13, nos. 3&4, pp. 238–254, 2006.

[5] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, M. Ohlberger, and O. Sander, A generic grid interface for parallel and adaptive scientific computing. Part I: Abstract framework, *Computing*, vol. 82, nos. 2&3, pp. 103–119, 2008.

[6] P. Lynch, The origins of computer weather prediction and climate modeling, *J. Comput. Phys.*, vol. 227, no. 7, pp. 3431–3444, 2008.

[7] N. Raba, E. Stankova, and N. Ampilova, On investigation of parallelization effectiveness with the help of multi-core processors, *Procedia Computer Science*, vol. 1, no. 1, pp. 2763–2768, 2010.

[8] J. Michalakes, J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, and W. Wang, The weather research and forecast model: Software architecture and performance, in *Proc. 11th ECMWF Workshop on the Use of Parallel Processors in Meteorology*, Reading, UK, 2005, pp. 156–168.

[9] Y. R. Chen, Research on key technigues of performance models for high performace computing, (in Chinese), PhD dissertation, National University of Defense Technology, Changsha, China, 2007.

[10] Z. Y. Jin and D. X. Wang, Diffusion algorithm of dynamic load balancing for heterogeneous system, (in Chinese), *Chin. J. Comput.*, vol. 26, no. 11, pp. 1487–1493, 2003.

[11] Z. Y. Jin and D. X. Wang, An optimal method of diffusion algorithm for hetergeneous system, (in Chinese), *J. Softw.*, vol. 14, no. 5, pp. 904–910, 2003.

[12] L. L. Zhang, H. Ye, J. P. Wu, and J. Q. Song, Parallel load-balancing performance analysis based on maximal ratio of load offset, (in Chinese), *J. Comput. Res. Dev.*, vol. 47, no. 6, pp. 1125–1131, 2010.

[13] Z. Y. Mo, X. P. Liu, and Z. M. Liao, Research on key techniques for parallelization and optimization of applied codes, (in Chinese), *J. Numer. Methods Comput. Appl.*, vol. 23, no. 1, pp. 31–40, 2002.

[14] Y. Q. Zhang, DRAM(*h*): A parallel computation model for high performance numerical computing, (in Chinese), *Chin. J. Comput.*, vol. 26, no. 12, pp. 1660–1670, 2003.
[15] Michalakes J and Vachharajani M, GPU acceleration of numerical weather prediction, *Parallel Process. Lett.*, vol. 18, no. 4, pp. 531–548, 2008.
[16] Z. W. Wang, X. B. Xu, W. Q. Zhao, S. B. He, and Y. P. Zhang, Parallel acceleration and performance optimization for GRAPES model based on GPU, (in Chinese), *J. Comput. Res. Dev.*, vol. 50, no. 2, pp. 401–411, 2013.
[17] J. Michalakes, J. Hacker, R. Loft, M. O. McCracken, A. Snavely, N. J. Wright, T. Spelce, B. Gorda, and R. Walkup, WRF nature run, in *Proc. 2007 ACM/IEEE Conf. Supercomputing*, Reno, NV, USA, 2007, pp. 1–6.
[18] G. Ruetsch, E. Phillips, and M. Fatica, GPU acceleration the long-wave rapid radiative transfer model in WRF using CUDA Fortran, in *Proc. 2010 Many-Core and Reconfigurable Supercomputing Conf.*, Rome, Italy, 2010, pp. 1–11
[19] P. Xu, Research on performance optimization of GRAPES dynamic core on sunway Taihu light, (in Chinese), Master dissertation, Tsinghua University, Beijing, China, 2019.
[20] J. S. Xue and D. H. Chen, *Scientific Design and Application of Numerical Prediction System*, (in Chinese). Beijing, China: Science Press, 2008.
[21] X. J. Wu, Study on the parallel computing in GRAPES high resolution numerical weather prediction mode, (in Chinese), PhD dissertation, National University of Defense Technology, Changsha, China, 2011.
[22] D. H. Chen, J. S. Xue, X. S. Yang, H. L. Zhang, X. S. Shen, J. L. Hu, Y. Wang, L. R. Ji, and J. B. Chen, New generation of multi-scale NWP system (GRAPES): General scientific design, *Chin. Sci. Bull.*, vol. 53, no. 22, pp. 3433–3445, 2008.
[23] D. H. Chen, X. S. Yang, H. L. Zhang, and J. L. Hu, Strategy for designing a non-hydrostatic multi-scale community model dynamic core, (in Chinese), *J. Appl. Meteor. Sci.*, vol. 14, no. 4, pp. 452–461, 2003.
[24] L. B. Zhao and Y. X. Tian, Improved parallel generalized conjugate residual algorithm, (in Chinese), *Comput. Eng.*, vol. 35, no. 4, pp. 80–82, 2009.
[25] Y. Saad, *Iterative Methods for Sparse Linear Systems*. 2nd ed. Philadelphia, PA, USA: SIAM, 2003.
[26] X. M. Huang, Q. Tang, Y. H. Tseng, Y. Hu, A. H. Baker, F. O. Bryan, J. Dennis, H. H. Fu, and G. W. Yang, P-CSI v1.0, an accelerated barotropic solver for the high-resolution ocean model component in the Community Earth System Model v2.0, *Geosci. Model Dev.*, vol. 9, no. 11, pp. 4209–4225, 2016.
[27] S. Cools and W. Vanroose, The communication-hiding pipelined BiCGstab method for the parallel solution of large unsymmetric linear systems, *Parallel Computing*, vol. 65, pp. 1–20, 2017.
[28] H. A. Van der Vorst, *Iterative Krylov Methods for Large Linear Systems*: *Volume 13*. Cambridge, UK: Cambridge University Press, 2003.
[29] P. Sanan, S. M. Schnepp, and D. A. May, Pipelined, flexible Krylov subspace methods, *SIAM J. Sci. Comput.*, 2016, vol. 38, no. 5, pp. C441–C470, 2016.
[30] J. Demmel, M. F. Hoemmen, M. Mohiyuddin, and K. A. Yelick. *Avoiding Communication in Computing Krylov Subspaces*. EECS Department, University of California, Berkeley, CA, USA, 2007.
[31] M. F. Hoemmen, *Communication-Avoiding Krylov Subspace Methods*. EECS Department, University of California, Berkeley, CA, USA, 2010.
[32] F. L. Lin, Performance optimization technology of global numerical weather forecasting system, (in Chinese), Master dissertation, Tsinghua University, Beijing, China, 2012.
[33] S. Williams, M. Lijewski, A. Almgren, B. Van Straalen, E. Carson, N. Knight, and J. Demmel, *s*-Step Krylov subspace methods as bottom solvers for geometric multigrid, presented at 2014 IEEE 28th Int. Parallel and Distributed Processing Symp., Phoenix, AZ, USA, 2014, pp. 1149–1158.

Jianqiang Huang received the master degree in computer technology from Anhui University in 2011. He is an associate professor at Qinghai University, China. He is currently a PhD candidate at Department of Computer Science and Technology, Tsinghua University. His research interest includes high performance computing.

Wei Xue received the BE and PhD degrees in electrical engineering from Tsinghua University in 1998 and 2003, respectively. Currently, he is an associate professor at Department of Computer Science and Technology, Tsinghua University. His research interests include scientific computing and parallel I/O.

Haodong Bian received the BS degree in computer and information from Anhui Polytechnic University, China in 2018. Now, he is a master student at Department of Computer Technology and Applications, Qinghai University, China. His research interests include high performance computing and graph computing systems.

Wenxin Yan is a master student at Department of Computer Technology and Applications, Qinghai University, China. Her research interest includes high performance computing.

Xiaoying Wang is a professor at Department of Computer Technology and Applications, Qinghai University, China. She received the PhD degree from Tsinghua University in 2008. Her research interests include cloud computing and parallel computing.

Wenguang Chen received the BS and PhD degrees in computer science from Tsinghua University in 1995 and 2000, respectively. He is a professor and associate head at Department of Computer Science and Technology, Tsinghua University. His research interest is in parallel and distributed computing and programming model.