The Journal of Supercomputing
https://doi.org/10.1007/s11227-023-05469-9

# Parallel algorithm design and optimization of geodynamic numerical simulation application on the Tianhe new-generation high-performance computer

**Jin Yang[1] · Wangdong Yang[1] · Ruixuan Qi[1] · Qinyun Tsai[1] · Shengle Lin[1] · Fengkun Dong[1] · Kenli Li[1] · Keqin Li[1,2]**

Accepted: 31 May 2023


**Abstract**
CitcomCu is a numerical simulation software for mantle convection in the field of geodynamics, which can simulate thermo-chemical convection in a three-dimensional domain. Due to the increasing demand for high-precision simulations and larger application scales, larger-scale computing systems are needed to solve this problem. However, the parallel efficiency of CitcomCu on large-scale heterogeneous parallel computing systems is difficult to improve, especially it cannot adapt to the current mainstream heterogeneous high-performance computing architecture with CPUs and accelerators. In this paper, we propose an geodynamics numerical simulation parallel computing framework using heterogeneous computing architecture based on the Tianhe new-generation high-performance computer. Firstly, the data partitioning mode of CitcomCu was optimized based on the large-scale heterogeneous computing architecture to reduce the overall communication overhead. Secondly, the iterative solution algorithm of CitcomCu was improved to speed up the solution process. Finally, the NEON instruction set based on SIMD is used for the sparse matrix operations in the solution process to improve parallel efficiency. Based on our parallel computing framework, the optimized CitcomCu was deployed and tested on the Tianhe new-generation high-performance computer. Experimental data showed that the performance of the optimized program was 3.3975 times higher than that of the unoptimized program on a single node. Compared with 50,000 computational cores, the parallel efficiency of the unoptimized program on one million computational cores was 36.75%, while the parallel efficiency of the optimized program was improved by 16.22% and reached 42.71%. In addition, the optimized program can be executed on 40 million computational cores, with a parallel efficiency of 36.54%.

**Keywords** Heterogeneous system · Large-scale parallelism · Efficient task scheduling NEON

Extended author information available on the last page of the article

Published online: 24 June 2023

## 1 Introduction

More than 80% of the total volume of the Earth is composed of the mantle, which plays a crucial role in the evolution of Earth's plates. On the one hand, mantle convection [1] can influence the movement of the Earth's surface continental plates and seafloor spreading. On the other hand, the process of mantle convection also controls the rate of heat loss from the Earth's interior. Therefore, mantle convection simulation has high research significance and practical value. But with viscosity strongly dependent on the temperature [2], Non-Newtonian processes, drastic changes in composition, and spherical curvature, mantle convection simulations become very difficult.

The governing equations [30] of mantle convection studies are derived from three sets of equations: the conservation laws of mass, momentum, and energy [3]. Mantle convection research exhibits strong temperature and stress dependence, as well as nonlinear characteristics due to the coupling of velocity and temperature in the energy equation. This requires the use of numerical simulation methods to solve the governing equations of mantle convection [4]. Since the late 1960s, the mantle convection numerical simulation has a long history, as Torrance and Turcotte [5]. However, due to the weak processing power and small storage capacity of early computers, it was difficult to obtain high-precision simulation solutions directly using numerical simulation methods. Recently, with the continuous progress of computer architecture and numerical simulation technology [6], the field of mantle convection is expected to advance in the area of geophysical fluid dynamics.

Parallel computing is an effective method to improve the computing speed and processing power of computer systems. Its basic idea is to decompose a complex problem into several sub-problems, each of which is independently processed in parallel by a separate processor, and then combine the solutions of each subproblem to greatly reduce the solution time of the entire problem. This method is particularly useful for fields such as scientific computing, big data processing, and deep learning, especially those that require processing large amounts of data or complex calculations. Parallel computing systems can be supercomputers with multiple processors or independent clusters of computers connected in some ways. In recent years, heterogeneous parallel computing has become increasingly popular in various fields, not just in supercomputing laboratories. For example, CUDA and OpenCL technology make it easier for programmers to write code that utilizes parallel architectures, enabling them to develop applications that process large amounts of data faster than before.

In this paper, we propose a parallel computing framework for large-scale geodynamic numerical simulation and improve the iterative algorithm of the geodynamic numerical simulation application (CitcomCu). For CPU and accelerator heterogeneous parallel computing, the underlying mathematical operations and communications are optimized, and experiments were carried out on the heterogeneous computing platform of the Tianhe new-generation high-performance computer (hereinafter referred to as the Tianhe).

The main contributions of this paper are as follows:

1. Based on the Tianhe, we provide the parallel computing framework for geodynamic numerical simulation. The parallel efficiency of the unimproved CitcomCu carried out in one million computing cores is 36.75% compared with 50,000 cores, and the parallel efficiency of the improved CitcomCu has increased by 16.22%, reaching 42.71%. Furthermore, the improved CitcomCu can be carried out in 40 million cores with the parallel efficiency reaching 36.54%.
2. For the Tianhe heterogeneous parallel processor, the iterative solving algorithm of CitcomCu has been replaced from Gauss–Seidel algorithm to CG (conjugate gradient) algorithm based on Krylov subspace. In addition, the NEON instruction set was used, which led to a performance improvement of 3.3975 times.
3. We use the performance analysis tools (Perf) to track the hot spots of the geodynamic solving process and use the NEON instruction set to accelerate the hot spots in the application solving process on single node. As a result, the performance is improved by 1.3×.

## 2 Related work

Most of the studies on numerical simulation of mantle convection convert numerical simulations into joint solutions of the approximate Stokes equation and energy conservation equation. The common methods include finite element, finite volume, and finite difference, which all solve partial differential equations after discretization. Another solution is to discretize the simulation region to be solved into individual particles, then divide the solution domain according to the associations between the particles, and solve it simultaneously.

Moresi and Gurnis [7] developed the original version of the Citcom in 1996, which can simulate thermochemical convection and mantle convection in three-dimensional Cartesian or spherical coordinates. Zhong [8] optimized the original version of the Citcom, used MPI for parallel communication, and the Boussinesq approximation, which greatly simplified the equations while retaining the original mantle convection-related features. In October 2003, the California Institute of Technology and US seismological research laboratory jointly developed geo-framework, the multi-scale deformation analysis framework tool for earth science. Geo-framework contains a large number of earth science problem solvers, among which Citicoms is the geodynamic rheology problem solver.

Martin Kronbichler et al. [9] proposed high-precision mantle convection simulation based on modern numerical methods in 2012 and developed the ASPECT framework, which also models mantle thermal convection. The ASPECT framework focuses on simulating the process of mantle motion. It uses the p4est parallel grid processing module and the Trilinos parallel linear algebra solving module, which can show small features in the mantle convection large solution domain. Gabriele Morra [10] described Pythonic Geodynamics, which used python to combine MPI, OpenMP, PETSc, and other modules to form the geodynamic solution framework.

The framework is highly integrated and easier for users to use. Nils Kohl et al. [11] developed the HyTeG finite element software framework by using a special method. The method uses particles without constructing matrix, which avoids to construct large coefficient matrices and can save a lot of resources. Fraters et al. [12] developed the Geodynamic World Builder open-source library in 2019 and added preset initial conditions and visualization module to the library. The visualization module can display complex geological structures. This open-source library can be used in almost any programming language and has excellent portability.

In recent years, more and more researchers have taken advantage of supercomputer characteristics to optimize large-scale applications in various fields. Jianyuan Xiao [13] used the symplectic electromagnetic fully kinetic PIC method on supercomputers to study crucial problems and phenomena in the magnetic confinement toroidal plasma. Yong (Alexander) Liu [14] develop a high-performance tensor-based simulator for random quantum circuits (RQCs) on the new Sunway supercomputer. It is a new milestone for classical simulation of quantum circuits. Shang [15] simulated highly accurate full quantum mechanical (QM) of Raman spectra on the new-generation Sunway high-performance computing system in 2021. These studies have important reference significance for optimizing large-scale geodynamics numerical simulation.

In the process of solving the geodynamics simulation, the supercomputer platform has been exploring the three-dimensional large-scale geodynamics numerical simulation technology. Small-scale parallel simulation often tested on distributed machines that used for personal research. Gómez [16] performed three-dimensional mantle convection simulation experiment by using the underworld2 virtual framework, which used 128 nodes (two cores per node) in the experiment. Large-scale parallel simulation often tested on high-performance supercomputing platforms. Bauer [17] solved the Stokes system on 40,960 cores using the ASPECT framework. This Stokes system has coupled velocity components and over a trillion degrees of freedom. Assunção and Sacek [18] conducted CitcomCu experiments by using 18,915 nodes on the supercomputing platform. Bauer et al. [19] used the HyTeG finite element framework to solve the surrogate element matrices constructed by low-order polynomial approximations and simulated large-scale mantle convection at 47,250 Computing cores. At present, the number of parallel cores in large-scale geodynamic numerical simulation application is at the level of ten thousand cores. With the deepening of geodynamic parallel research, the optimal solution strategy (May et al. [20]) and multi-core parallelism (Bangerth et al. [21]) have made further improvement in computing performance.

## 3 The Tianhe architecture

Currently, all computers adopt a parallel structure. Specifically, modern computers support parallel computing with at least one type of parallel feature, such as vector instructions, multi-threaded cores, multi-core processors, multiprocessing, or parallel co-processors. In order to fully exploit the performance of the Tianhe, it is required that CitcomCu closely integrates with the architecture of supercomputers

[22], fully leveraging the advantages of the Tianhe's specific architecture. As we move toward the era of exascale computing, the architecture of supercomputers is showing a diversified development trend. In addition to the computing resources of general-purpose processors (CPUs), accelerators are introduced as computing acceleration components. Currently, acceleration components such as General-Purpose Graphic Processing Unit (GPGPU), Intel's Many Integrated Core (MIC), and General Purpose Digital Signal Processing (GPDSP) are widely used in high-performance computer systems. The introduction of accelerators makes the architecture of high-performance computers more diverse and complex, and also brings new problems to the porting of high-performance computing applications.

The Tianhe system adopts a heterogeneous architecture consisting of multi-core processors and DSP accelerators, connected by a high-speed network. Each cabinet includes multiple nodes, with each node consisting of an MT-3000 processor [23]. The processor utilizes a hierarchical structure and multiple parallel features and also has a heterogeneous memory consistency model, running in an on-chip heterogeneous environment. Figure 1 shows the architecture of a single node in the Tianhe, with a general-purpose zone (host side) including 16 general-purpose CPU cores for hardware cache consistency, and an accelerator zone (device side) consisting of 96

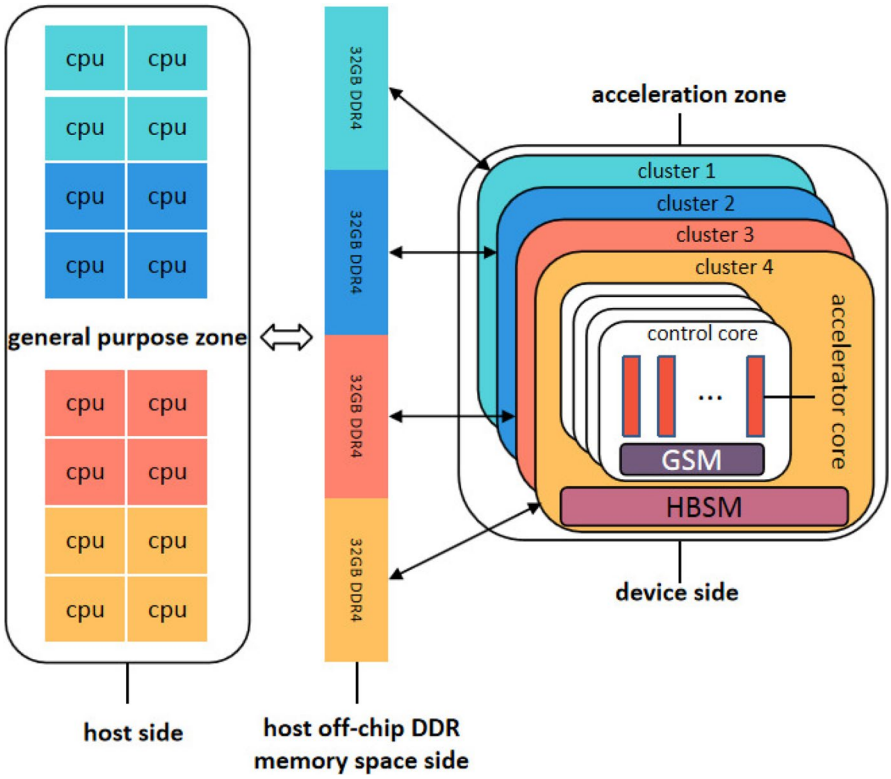


**Fig. 1** The Tianhe single-node architecture

control cores and 1536 accelerator cores. These cores are evenly divided into four independent accelerator clusters, each of which has on-chip Global Shared Memory (GSM), High Bandwidth Shared Memory (HBSM), and off-chip DDR memory space. The general-purpose area can access the entire high-bandwidth shared memory and DDR space of the four independent accelerator clusters, while the control cores and accelerator cores can only access the global shared memory, high-bandwidth shared memory, and DDR memory space within their respective accelerator clusters.

CPU cores are the foundation of operating systems, with hardware cache consistency and management of all storage spaces in the computer system, as well as handling I/O operations. Users can control the behavior of accelerators through CPU cores, including allocating space, preparing programs and data, triggering execution, querying status, and shutting down accelerator cores. The 24 control cores in each accelerator cluster can independently obtain instructions and access memory, and can run different programs independently, but they usually use the SPMD mode for collaborative computing. As the general-purpose area can access the entire high-bandwidth shared memory and DDR space of the four computing areas, data migration only occurs when necessary communication is needed between different independent accelerator clusters. The CPU cores in the general-purpose zone not only have the ability to control overall tasks and start the operating system, but also have general processing capabilities. This zone can also use the NEON instruction set to accelerate sparse matrix calculations. In contrast, the accelerator zone is specifically designed for computationally intensive tasks, aiming to provide higher computational performance and efficiency, especially suitable for computationally intensive iterative solving problems.

The Tianhe system adopts a heterogeneous architecture consisting of multi-core processors and DSP accelerators. The advantage of this architecture is that it can fully leverage the computational performance of accelerators, thereby improving the execution efficiency of application programs. Accelerators are efficient parallel processors with more computing units and memory bandwidth than traditional CPUs. Therefore, they can complete a large amount of parallel computing tasks in a relatively short time. By sending data from the host side to the device side accelerator for calculation, intensive computing tasks can be assigned to the accelerator, effectively reducing the workload on the host. In addition, due to the strong computing power of accelerators and their ability to simultaneously process more data, they are clearly superior to traditional CPUs for large-scale data operations and complex computing tasks. In summary, by efficiently completing large-scale data operations and complex computing tasks with the computational power of device-side accelerators, and optimizing the data transfer process, the execution efficiency of CitcomCu can be significantly improved.

Based on the Tianhe system, we adopt new parallel algorithms with tightly coupled architecture and run time optimization techniques to realize large-scale numerical simulation of geodynamic applications on the Tianhe. In the computing process, the key technical problems to be solved include:

1. The meshing optimization for large-scale numerical simulation.

2. The data communication optimization between and within nodes.
3. Fast iterative solving optimization combined with heterogeneous processor.

## 4 The analysis of CitcomCu performance

For the large-scale numerical simulations performed by CitcomCu, the key lies in solving large-scale nonlinear systems. In this process, CitcomCu uses iterative methods to solve the discretized Stokes system, which yields information on velocity and pressure fields. Meanwhile, it is necessary to solve for the temperature field by combining the energy conservation equation and geophysical parameters. In Sect. 5.2, we describe in detail the iterative process of the Stokes system in CitcomCu, which employs iterative solving methods such as Gauss–Seidel. Specifically, the velocity and pressure fields are solved separately at each time step, and the iteration number is controlled based on error limits. Additionally, multi-grid methods are applied to accelerate the iterative solving process.

Figure 2 illustrates the numerical simulation process of CitcomCu, which is a complex nonlinear dynamic process that requires comprehensive analysis of the interactions among various physical processes and the use of appropriate numerical algorithms to ensure computational accuracy and stability. In the simulation process, the experimental conditions need to be set in the input file of CitcomCu, including physical and numerical parameters such as temperature field, density field, viscosity coefficient, grid size, etc. Then, the three-dimensional geological body

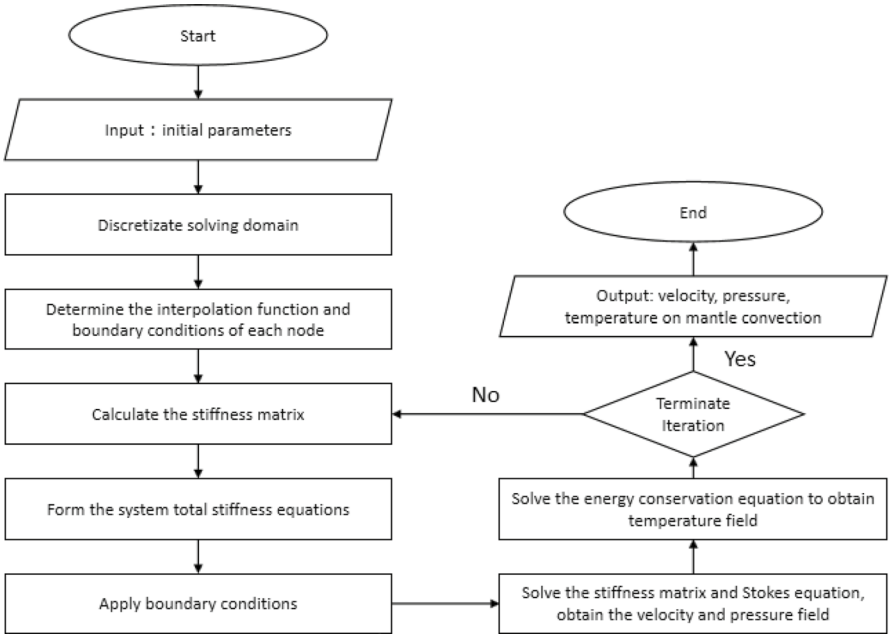


**Fig. 2** CitcomCu numerical simulation process

is partitioned into a structural mesh using the finite element method based on the numerical parameters set in the input file. This process converts the continuous nonlinear partial differential equations into a Stokes equation group composed of a large number of algebraic equations. Specifically, the calculation region is decomposed into several elements, each of which contains several nodes and elements, forming the matrix of the algebraic equation group. The node refers to the discrete point on the mesh, and the element refers to the graphical element connecting the node, such as a triangle or quadrilateral. Next, for each node mesh, the discrete expression of its internal matrix and vector is determined using interpolation functions, and it is modified according to boundary conditions. Then, the stiffness matrix of the node mesh can be calculated using these discrete matrices and vectors. Once all the stiffness matrices of the node meshes are calculated, they can be combined into the system total stiffness matrix equation group, and the given boundary conditions and load field can be applied to calculate the solution of the equation to solve the problem. After solving the system total stiffness matrix, it needs to be combined with the Stokes equation group and an iterative algorithm (such as Gauss–Seidel, SOR, etc.) is used to solve the velocity field and pressure field. Additionally, to solve the temperature field, an appropriate energy conservation equation needs to be added. In each iteration process, it is necessary to check whether the gradient of the velocity field and pressure field satisfies the convergence requirements. If it does, the iteration can be stopped; otherwise, it needs to continue until the convergence condition is met. Finally, output the mantle temperature field, velocity field, pressure field, thermal evolution information obtained from the iteration, as well as related statistical information.

Performance analysis and code optimization are two essential stages in the system performance optimization process. According to the iterative solution process of CitcomCu numerical simulation, CitcomCu can be transplanted to heterogeneous computing platforms to achieve large-scale parallelism. On single node, application computing efficiency is improved with multi-thread scheduling pool and NEON instruction set [24]. On multiple nodes, the parallel efficiency is improved by data partition, communication mode, and data storage layout.

Through the analysis of the performance of CitcomCu, we can find the performance bottlenecks and hot spots of the application, so as to optimize the application quickly. The Perf [25] is a performance analysis tool in the Linux kernel, which is often used in the performance analysis stage of system optimization. Using Perf, we can obtain CitcomCu performance statistics, including function, compilation, process, or thread-level performance.

In the performance analysis stage, we use the Perf to analyze the numerical simulation process of CitcomCu, and the analysis results are shown in Fig. 3. The system process occupancy rates of the *gauss_seidel* and *n_assemble_del*2*_u* functions in CitcomCu are 52.51% and 17.44%, and they are the main computational overhead sources of CitcomCu. The *gauss_seidel* function is the key algorithm for the iterative solution of Stokes equation, which is used to iteratively update the different variables of the Stokes equation to obtain the velocity and pressure fields. To complete the solving of the Stokes equation, *gauss_seidel* function need to be called multiple times. In the numerical simulation process, the call to the *n_assemble_del*2*_u*

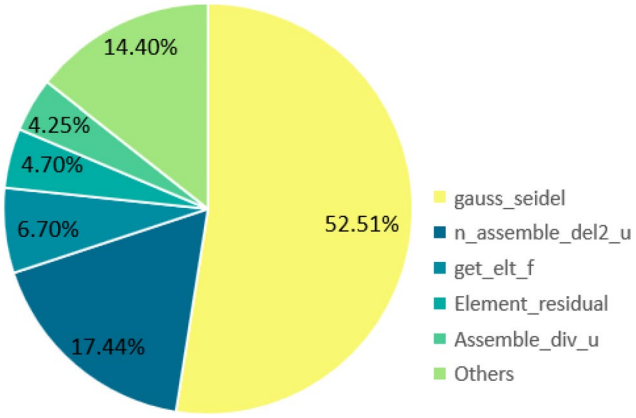


**Fig. 3** Analysis result of CitcomCu system process occupancy rates

function is determined based on the different physical materials. In addition, the *n_assemble_del*2*_u* function is also called before assembling the coefficient matrix. The other functions are tool functions to help solve the Stokes equation, as *get_elt_f* (get intermediate variable), *Element_residual* (get residual). According to the performance analysis results, it is better to optimize *gauss_seidel* function.

## 5 The parallel computing framework for large-scale geodynamic numerical simulation

In the process of building the large-scale parallel numerical simulation application on the heterogeneous computing platform, and realizing the collaborative parallel computing mode, it is necessary to design the overall parallel framework to ensure that the application is large-scale parallel and avoid the sharp decrease of parallel efficiency as much as possible. This section mainly describes the parallel computing framework of large-scale geodynamic numerical simulation designed for the Tianhe, as shown in Fig. 4.

### 5.1 The large-scale parallel computing optimization

When conducting large-scale geodynamic numerical simulation on the Tianhe, efficient parallel task scheduling is achieved by optimizing technologies such as the data partition, data storage layout, and data communication within the parallel computing framework.

#### 5.1.1 The data partition optimization

Large-scale geodynamic application can make better use of the Tianhe architecture in numerical simulations and has higher accuracy and longer iteration steps.

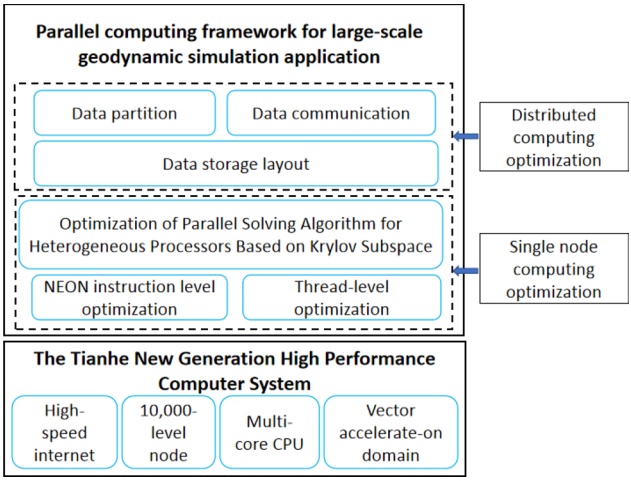


**Fig. 4** The parallel computing framework for large-scale geodynamic numerical simulation

Therefore, we optimize the data partition under the framework of parallel computing and optimize the multi-level storage access for the structured grid of data. The distributed batch-stream combined algorithm proposed by T. Weng [26] for graph decomposition also adopts a new task assignment strategy. When drawing large-scale three-dimensional model, it is necessary to effectively manage bandwidth requirements. So we need to control the scale of the data set as much as possible, reduce the amount of data to be drawn, make full use of the characteristics of multi-level storage hierarchy, and reduce the cache access mismatch rate of data. The unified scheduling management of data is realized through data encoding, and parallel load balancing can be effectively achieved.

The finite element solution method needs to perform basic structural grid division on the simulated three-dimensional geological structure. Therefore, we propose to mesh the solving domain according to the number of nodes, which is easier to perform parallel data division. According to three directions of the Cartesian coordinate system, we use node to control the data and divide the structure grid into three-dimensional topology process. Figure 5 shows the structure grid data partition of single node on the node grid. Each node only stores the parameter data for the part of the mesh associated with this node.

To accelerate the iteration speed of mantle convection simulation, we introduced a multi-grid method in the structural mesh. Specifically, we first perform arbitrary initial value calculation on the low-resolution structural mesh and then obtain a higher-resolution multi-grid through interpolation. On these higher-resolution meshes, multi-grid numerical simulation calculations are performed to obtain the final simulation results. In this process, the multi-grid structure is used for each node structural mesh at level layers, and the number of mesh points in the three directions is $M_x$, $M_y$, and $M_z$, respectively. Then, they are expanded into different levels of multi-grid structures according to expansion formula (1) in the three directional nodes. This multigrid method can accelerate the iteration speed of mantle convection simulation and improve computational efficiency.

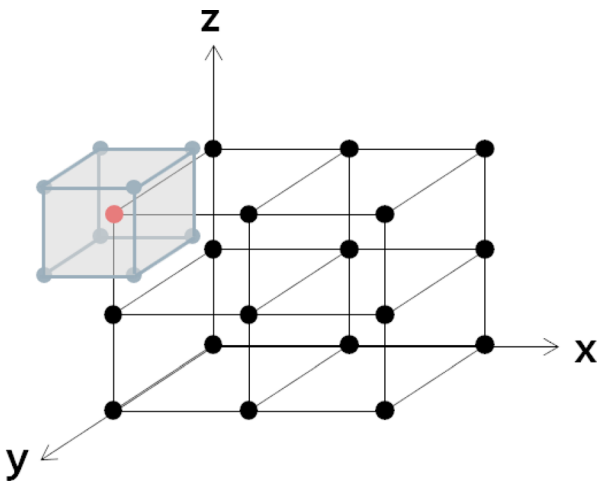


**Fig. 5** Three-dimensional structured grid and node grid

$$Lm_i = M_i * 2^{\text{level-1}} + 1, \cdots i \in \{x, y, z\} \tag{1}$$

In order to provide a clearer demonstration of the implementation of the multi-grid method, we have presented a two-dimensional multi-grid in Fig. 6. This figure distinctly illustrates the step-by-step process of obtaining a high-resolution grid by interpolating from a low-resolution one. Through the utilization of the multi-grid method, we can significantly enhance the computational efficiency and accuracy of mantle convection simulation, resulting in a more precise replication of the physical processes within the Earth.

The multi-grid data are also managed by the node, and it is divided by the local grid level to form the multi-grid data set which starting with the vertex elements of the structural grid in each node, as shown in Fig. 7.

For the vector parameters such as velocity, we carry out grid topology information progressive coding division on multi-grid data by three directions. For the scalar parameters such as temperature and pressure, we carry out partition for multi-grid data according to nodes. Finally, each level address codes of the multi-grid of different parameters are obtained. By using the multi-grid method, the extreme point can be searched faster in the iteration of solving the Stokes equation and the energy conservation equation. Geodynamic application requires a large number of iterative steps to simulate the long time of billion years. Therefore, we can make full use of

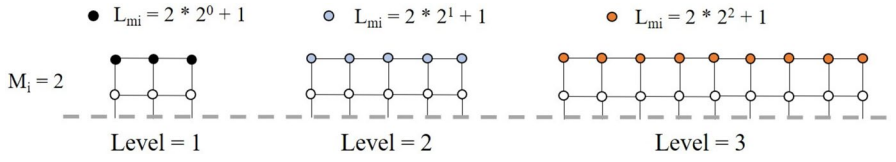


**Fig. 6** In the two-dimensional case, the extension of multi-grid

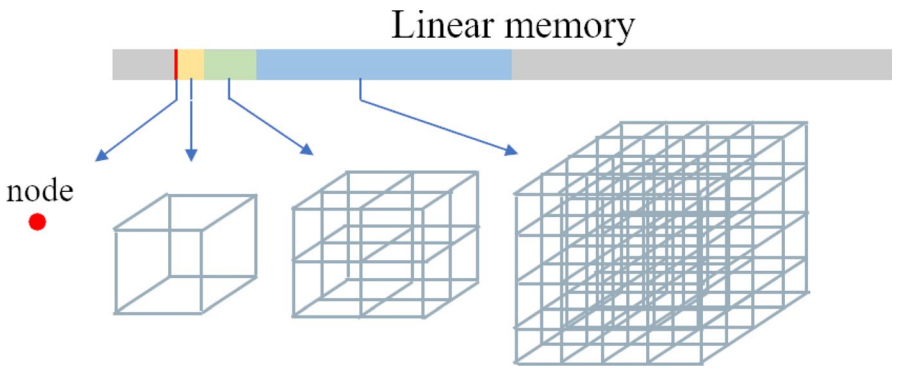


**Fig. 7** The multi-grid data

the characteristics of multi-grid to improve the computational efficiency of geodynamic numerical simulation, that is, to shorten the solving time of a single iteration.

### 5.1.2 The data storage layout optimization

For the geodynamic numerical simulation, it is required to discretize the continuity equations in these simulation processes, to produce grid space or plane (including the corresponding time series) consisting of a finite number of discrete points. After performing the corresponding numerical simulation computing on the discretized grid, we can obtain the approximate value of the target variable.

Zhao Tuowen proposed the two-dimensional data storage layout optimization method in 2021 and conducted experiments on three-dimensional stencils application [27]. This method eliminates the process of data packaging and moving during communication and reduces the number of communications and communication time. Combine with the Tianhe architecture, we propose the layout optimization method of three-dimensional data storage with node grids. This method uses the granularity of the accelerator to set subdomains, makes regional partition for the data in the node by fine-grained subdomains, and then optimizes the physical layout of the entire data through the sorting adjustment of local areas. This layout optimization method of data storage can save part of the data packaging process, improve the continuity of data locality, and finally save the communication time of CitcomCu and greatly improve the performance.

The optimized three-dimensional data layout structure is shown in Fig. 8. First, combined with the heterogeneous architecture of the Tianhe, we choose the size of the sub-domain for data storage according to the maximum granularity that can be computed in the acceleration domain. In addition, the data in the sub-domain and the sorting of the sub-domain preserve the logical order of the algorithm. Therefore, after the layout optimization of data storage, it will not affect subsequent heterogeneous optimization and algorithm computation. Secondly, combined with the underground structural characteristics of geodynamic simulation, the longitudinal geophysical characteristics are given priority in mantle convection simulation. So

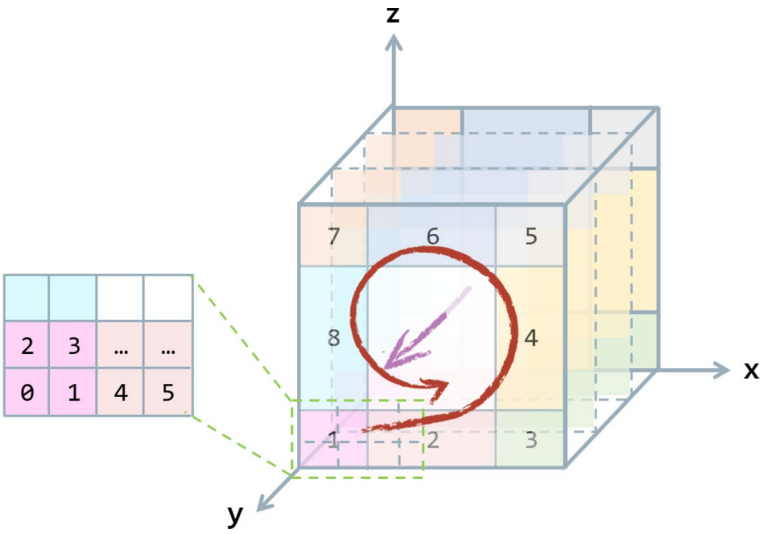


**Fig. 8** Optimized three-dimensional data storage layout

when storing the three-dimensional data on the node, we first use the x-axis and z-axis planes for two-dimensional data storage and then expand to the y-axis.

As shown in Fig. 9, after the layout optimization of data storage, we do not need to move the data when communicating in the y-axis direction. When communicating in the x-axis and z-axis directions, we only need to move, arrange, and pack the data with good local continuity in the xz-plane toward the y-axis direction. Compared

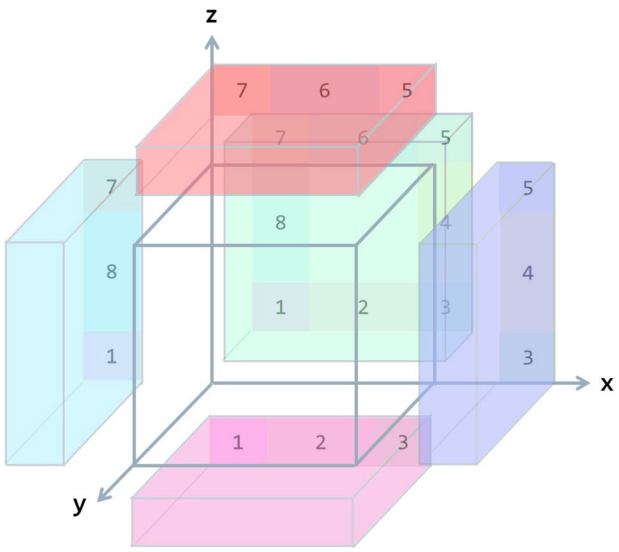


**Fig. 9** Partial communication after data layout optimization

with the original sorting method, the optimized data layout has better local continuity on the xz plane. This reduces access requirements for sequential storage, and the accelerator core is able to access data faster and more efficiently.

### 5.1.3 The data communication optimization

When the geodynamic application performs mantle convection simulation, the data in the divided grid area are solved each time, which can reduce the data communication between nodes. However, in solving the geodynamic equation, there must be communication between the results of each iteration which is in the associated data unit. Weng [28] designed an ingenious message aggregation strategy to avoid the memory overflow problem while processing large-scale graphs. We design two-level communication in the parallel computing framework, including data communication between nodes and data communication within nodes. When communicating between nodes, each node will communicate with five surrounding neighbors [29] to obtain the necessary data for computation, while when communicating within nodes, there is no communication between threads.

As shown in Fig. 10, since the Tianhe adopts tree-shaped network topology, the network bandwidth decreases as the node distance increases. There are larger bandwidth and lower delay between near-end nodes. Heterogeneity in communication speed of nodes at different distances can cause the overall communication efficiency of geodynamic applications to be limited by the communication of nodes at longer distances. Therefore, according to the communication bandwidth difference between different nodes, multi-grid data of different granularity are transmitted to maintain consistent communication efficiency. MPI communication mechanism is adopted for communication between nodes. Fine-grained multi-grid data are transmitted on

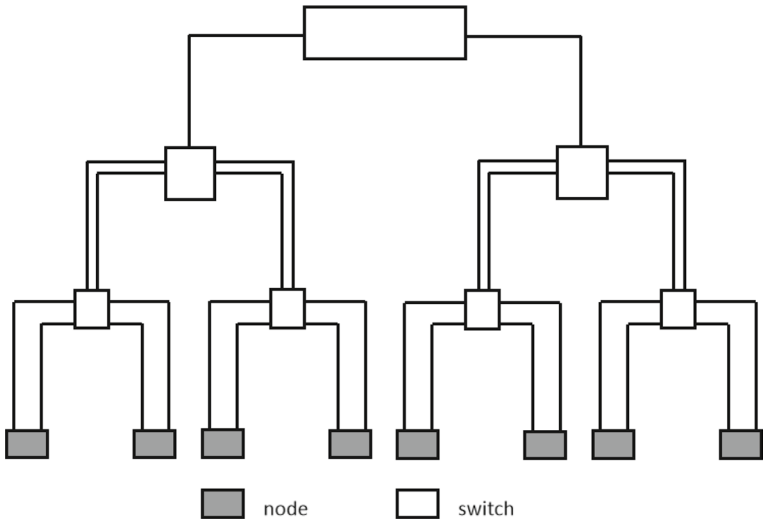


**Fig. 10** The Tianhe adopts tree network topology

adjacent nodes to maintain the accuracy of computing results. The coarse-grained multi-grid data are transmitted on remote nodes, which is used to reduce the communication bandwidth burden between the remote nodes, sacrifice accuracy to balance the communication efficiency of different nodes, maximize the use of effective bandwidth to achieve load balancing of communication.

The data communication in the node is mainly related to the heterogeneous architecture of the Tianhe. The main computing part of CitcomCu mantle convection simulation iteration is computed on the accelerator, and the memory required for communication is allocated through the dynamic allocation method at run time. The generated temporary memory space will be released after use, reducing the memory usage during program execution. By analyzing the scale of each data transmission and the delay between the computing, to set up a double buffering mechanism during communication. So the data waiting time during computing is reduced, and the pipeline of data and computing is realized, hiding a certain data communication overhead, to improve the overall performance of the program.

### 5.2 The optimization of iterative solving algorithm based on Tianhe heterogeneous processor

Finite element method is very effective in solving differential equations with complex geometric shapes and material properties. When CitcomCu uses the finite element method to solve the thermal convection control equation, it needs to solve the Stokes system iteratively to obtain the current velocity field and pressure field. Zhong [30] had proposed the solution to the finite element implementation of Stokes flow weak formula, and the matrix form of Stokes system is as follows:

$$\begin{bmatrix} K & G \\ G^T & 0 \end{bmatrix} \begin{Bmatrix} V \\ P \end{Bmatrix} = \begin{Bmatrix} F \\ 0 \end{Bmatrix} \tag{2}$$

where the vector $V$ contains the velocity at all the nodal points, the vector $P$ is the pressure at all the pressure nodes, the vector $F$ is the total force term resulting from the derivation process, and matrices $K$, $G$, and $G^T$ are the stiffness matrix, discrete gradient operator, and discrete gradient operator transpose, respectively. Explicitly, we can solve the linear system of equations iteratively to get the velocity and pressure values.

Because there are zero-element blocks on the diagonal of Eq. (2), the current system of equations is singular, but it is symmetric. In addition, the stiffness matrix $K$ is symmetric positive definite, so in numerical simulation, matrix Eq. (2) can be split into two coupled equations:

$$KV + GP = F \tag{3}$$

$$G^T V = 0 \tag{4}$$

In the case of the initial pressure $P_0 = 0$, the initial velocity $V_0$ can be obtained by Eq. (3):

$$V_0 = K^{-1}F \tag{5}$$

After further derivation, we finally get Algorithm 1 Uzawa Algorithm. For large-scale numerical simulation, the key is to solve large-scale linear systems, so that the iterative solution algorithm can adapt to large-scale parallel computing systems. The Jacobi iterative algorithm has achieved parallel design on the ternary optical computer [31]. We develop an iterative parallel solver within the framework of parallel computing for geodynamic numerical simulations based on the CG algorithm of the Krylov subspace method.

**Algorithm 1** Uzawa Algorithm

**Require:** stiffness matrix, discrete gradient operator, and discrete divergence operator transposes $K$, $G$, $G^T$

**Ensure:** the vector $V$, $P$ after iterative computing

1: $k = 0, P_0 = 0$
2: Solve $KV_0 = F$
3: $r_0 = H = G^T V_0$
4: **while** $|r_k| > \varepsilon$ **do**
5: $\quad k = k + 1$
6: $\quad$ **if** $k = 1$ **then**
7: $\quad\quad s_1 = r_0$
8: $\quad$ **else**
9: $\quad\quad \cdots$
10: $\quad$ **end if**
11: $\quad$ Solve $Ku_k = Gs_k$
12: $\quad \cdots$
13: $\quad$ compute the value of $P_k, V_k$
14: $\quad \cdots$
15: **end while**
16: $P = P_k, V = V_k$

The iterative parts in algorithm 1 (lines 2 and 11) are solved by *gauss_seidel* function, and $P_k$ and $V_k$ are the pressure and velocity of iteration step k, respectively. The residual error of iteration step k is $r_k$, $u_k$ is the computed column vector, and $\varepsilon$ is the given accuracy, which is also the exit condition for the loop. Since $K$ is a symmetric positive definite matrix, the convergence speed of the CG algorithm based on the Krylov subspace is faster than that of the Gauss–Seidel algorithm. Taking $Ku_k = Gs_k$ as an example, we use the Tianhe series machine interface to change the original solving method to the CG algorithm based on the Krylov subspace, as shown in Algorithm 2, where $r_n$ represents the residual in the $n$-th iteration step, $p_n$ represents the search direction in the $n$-th iteration step, and $Kp_n$ represents the linear combination of all vectors in the corresponding Krylov subspace multiplied by the $K$ matrix. As shown in Fig. 11, the application code is mainly divided into two parts, the host side code and the device

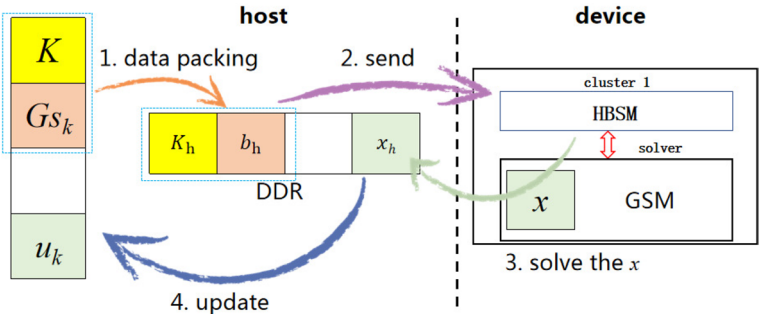


**Fig. 11** Heterogeneous algorithm running interactive relationship

side code. The host side code mainly initializes the data in the DDR memory space, packages the data, and then sends it to the HBSM of acceleration cluster (device side). The device side code receives the incoming data from the host side and then uses the accelerator core to accelerate the computing part. The GSM of accelerator core (device side) obtains data from HBSM and sends the results to HBSM after the calculation is completed; finally, control core sends the results to the host side.

**Algorithm 2** Optimization of Krylov subspace in iterative solving algorithm

**Require:** Symmetric positive definite matrix $K$, right-hand side vector $b$, and initial guess value $x_0$, satisfying $b = Gs_k$ and $x_0 = u_k$

**Ensure:** The number of iterations $n$, and the solution $x_{n+1}$, i.e., the solution of $u_k$

Initial value $r_0 = b - Kx_0$, $p_0 = r_0$ and $n = 0$
**repeat**
  **for** $n = 0, 1, \ldots$ **do**
    **if** $p_n = 0$ **then**
      return $x_n$
    **else**
      $\alpha_n = r_n^T r_n / (p_n^T K p_n)$
      $x_{n+1} = x_n + \alpha_n p_n$
      $r_{n+1} = r_n - \alpha_n K p_n$
      $\beta_n = r_{n+1}^T r_{n+1} / (r_n^T r_n)$
      $p_{n+1} = r_{n+1} + \beta_n p_n$
    **end if**
  **end for**
**until** convergence criteria satisfied
Output the number of iterations $n$, and the solution $x_{n+1}$ after iteration

### 5.3 The optimization of matrix operation based on vector accelerator

It is another characteristic of geodynamics application to realize the register-level parallelism of floating point operation on ten million core large-scale heterogeneous parallel computing system. Floating point operation is the most basic operation unit of numerical simulation. In order to optimize the floating point operation, it is necessary to consider the multi-core shared cache, the shared memory in heterogeneous processors storage characteristics, or optimize key algorithms to adapt to the Tianhe multi-level storage architecture. We optimize the floating point operation in CitcomCu key algorithms through the NEON instruction set.

To apply CitcomCu to the large-scale heterogeneous parallel computing platform, it is necessary to use the performance optimization technology combining physical process and the Tianhe platform to efficiently achieve the parallel simulation of physical process. Therefore, we designed the large-scale parallel computing framework for geodynamic numerical simulation application that can realize multi-level parallelism on the Tianhe, so as to make full use of the powerful computing performance and parallel characteristics of the Tianhe. The SIMD component is a part of the parallel computing framework. It can identify task scale and mark, autonomously schedule the computing part of the task to the suitable computing core according to the computational scale. In large-scale tasks, the acceleration zone is used as the computing unit, and the accelerator is fully used through the heterogeneous multi-thread library. In medium-scale tasks, the accelerator core of the acceleration zone is used as the computing unit. In small-scale tasks, the vector accelerator is used as the computing unit, and the NEON interface is used to avoid data transmission overhead.

In Sect. 5.2, we only did the Krylov optimization of the *gauss_seidel* algorithm, so the proportion of the algorithm as a whole in the calculation process is fixed. We replaced the *gauss_seidel* algorithm with the CG algorithm of Krylov. And *gauss_seidel* can be regarded as an orthogonal projection method similar to CG, but the two subspaces are the same. Therefore, the matrix-based NEON optimization method is also suitable for CG algorithm.

The vector accelerator of the Tianhe supports SIMD technology. SIMD makes multiple operation units to perform parallel operations on multiple data under the control of one instruction. These vector registers typically have a width of 128 bits and can store multiple data elements of the same type, such as 8 16-bit integers or 4 32-bit single-precision floating-point numbers. To achieve the goal of processing multiple data elements simultaneously, SIMD extension instructions pack multiple data elements of the same type into a vector, which is then stored in a vector register for parallel computation. During actual computation, if the number of data elements being processed is not evenly divisible by the width of the vector register, special handling is required. When the remainder is slightly smaller than the width of the vector register, special instructions can be used to process the last few data elements and ensure that all data can be processed simultaneously by the vector register. When the remainder is much smaller than the width of the vector register, the CPU core can directly execute the computation. The advantages of SIMD technology

include low hardware cost, simple control, and the ability to achieve higher performance at lower power consumption.

NEON [32] is an ARM technology that combines 64-bit and 128-bit SIMD instruction sets. It has a vectorized operation, that is packs several source data into a source register, and then performs the same operation on the data to generate several destination data. The numerical computing on single grid block is optimized by NEON technology, which can reduce data dependencies, reduce unnecessary branch statements. Register reusing can improve the instruction-level parallelism of arithmetic operations in the loop as much as possible, and multiple instructions can be pipelined.

**Algorithm 3** NEON Optimization of Krylov Algorithm

**Require:** Local vector Input in the original function

**Ensure:** Computed vector Out

1: $vdupq\_n\_type$: Initialize registers $neon\_UU$, $neon\_B$, $neon\_ans$
2: $vld1q\_dup\_type$, $vld1q\_lane\_type$: Extract input data in three directions from memory and load it into registers $neon\_UU$, $neon\_B$
3: In the register, single instruction $vmlaq\_type$ completes three operation: $ans_1 = ans_1 + B1_i * UU$, $ans_2 = ans_2 + B2_i * UU$, $ans_3 = ans_3 + B3_i * UU$
4: $vst1q\_type$: Data is loaded from register $neon\_ans$ to memory parameter Out

NEON optimization in CitcomCu is shown in algorithm 3 (part of the code). In CitcomCu, the iterative solution of the Stokes equation is to calculate the three directions of the velocity field. Therefore, three instructions are needed to calculate the velocity field for each iteration. Through the NEON instruction set, the source code is optimized at the assembly layer, and the data are loaded into the register. Compared with the original algorithm, single instruction can complete three operations. However, the NEON technology brings additional load and stores instruction time. Compared with these time overheads, the optimization effect of NEON instructions is better, so that the overall computing efficiency can be improved.

**Table 1** Environment configuration for design and operation

| Environment | Related configuration |
|---|---|
| Development hardware environment | The Tianhe series machine 16 CPU cores + accelerator cores, memory ddr4: 128 GB |
| Operating hardware environment | The Tianhe series machine 16 CPU cores + accelerator cores, memory ddr4: 128 GB |
| Developed operating system | YH Aquila 2.0.1.0 (based on gcc 10.2.0), mpich 3.4.1 |
| Running platform/Operating system | YH Kylin 10.1.8.3 |
| Running support environment/Support software | YH Kylin 10.1.8.3 |

## 6 Experiment

### 6.1 Experimental environment

As shown in Table 1, optimized CitcomCu was designed and tested based on the CPU cores and DSP accelerator cores of the Tianhe series machine. Each Tianhe node contains 16 general-purpose CPUs, 96 control cores, and 1536 accelerator cores, working at frequencies of 1.8 and 2.2 GHz, respectively. The simulation peak of the MT-3000 node is 9.6 TFLOPS, and the theoretical peak is 19 TFLOPS. Each HBSM has a capacity of 48 MB, with a bandwidth of up to 307 GBps, while a DDR memory provides 32 GB of memory capacity and a memory bandwidth of 204 GBps. In addition, each accelerator cluster is equipped with 6 MB of private high-bandwidth GSM. The entire operating environment supports YH Kylin 10.1.8.3. It should be noted that one accelerator core in the accelerator cluster is equivalent to one computing core. Therefore, each Tianhe node is equivalent to 1536 computing cores. In addition, Tianhe uses multi-level storage access, which can greatly accelerate data reading and computation speed.

### 6.2 Experimental results

CitcomCu is a weak scaling application that increases data scale by changing the grid resolution to simulate mantle structures of the same size. Each parameter in CitcomCu is generated by a single node, and each node has an independent base grid. Vector data contain points in three directions. In multi-grid expansion, the base grid is expanded into different levels of multi-grids. Taking a three-level multi-grid as an example, at the first level, the scalar parameter comprises (nx + 1) * (ny + 1) * (nz + 1) data; at the second level, the scalar parameter comprises (2 * nx + 1) * (2*ny + 1) * (2*nz + 1) data, and at the third level, the scalar parameter comprises (4 * nx + 1) * (4 * ny + 1) * (4 * nz + 1) data. Here, nx, ny, and nz represent the number of grids in the three directions of the independent base grid of each node, satisfy nx=ny=nz=2. We combine the data structure grid with the node grid. As the number of nodes increases, the data grid will be subdivided more finely to ensure an increase in the proportion of data and nodes. Therefore, when calculating parallel efficiency, the calculation formula can be obtained based on Amdahl's law:

$$S(n) = \frac{T(1)}{T(n)} \tag{6}$$

Among them, S is the parallel efficiency of n nodes relative to 1 node (the benchmark), T(1) is the serial execution time of the program, and T(n) is the parallel execution time (Including serial part execution time).

We compared the performance of code with and without NEON instruction set optimization by recording their run time to calculate their performance improvement ratio. In the experiment, we set the number of iterations *maxstep* to 500, double precision to 1.0e−6, data grid size to $32^3$, and set the number of layers in the multi-grid

**Table 2** Time comparison of NEON-optimized code and original application

| Nodes | NEON optimization run time/s | Original run time/s | Performance improvement ratio |
|---|---|---|---|
| 1 | 12.4281 | 16.6825 | 1.3423 |
| 8 | 18.6076 | 24.3759 | 1.3100 |
| 64 | 22.0273 | 28.7651 | 1.3059 |
| 512 | 25.8927 | 33.6213 | 1.2985 |
| 4096 | 28.7554 | 37.5978 | 1.3075 |

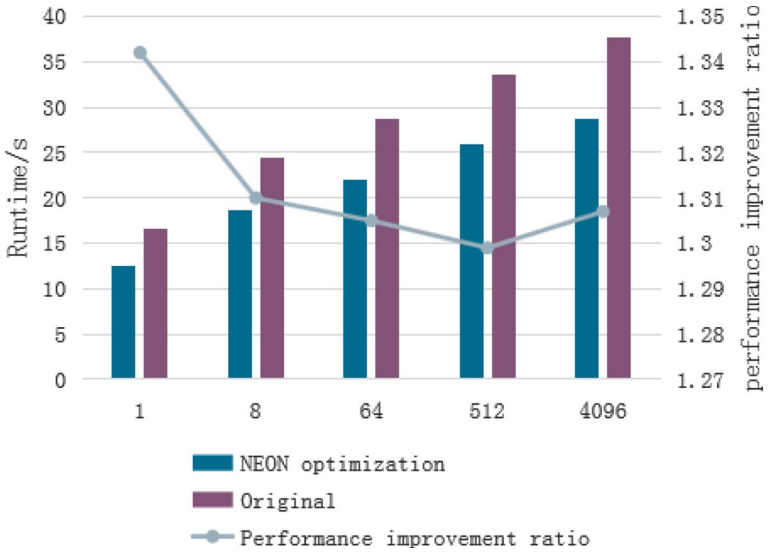


**Fig. 12** Comparison of iteration run time between original application and NEON-optimized code under different scale

to 3. The experimental results are shown in Table 2, and further details are shown in Fig. 12.

The experimental results show that the performance improvement effect of NEON optimization is better than the original application; the performance improvement ratio is about 1.3×.

To verify the optimization effect of the Krylov subspace-based CG algorithm, we conducted a series of experiments on a heterogeneous single node. In the experiments, we tested the performance of CitcomCu with no optimization, NEON instruction set optimization alone, and both NEON instruction set and Krylov subspace optimizations applied, and recorded the corresponding experimental data. The optimized algorithm was deployed on the Tianhe platform for single-node numerical simulation experiments in geodynamics applications. The final experimental results are shown in Fig. 13. It can be seen that the optimized algorithm is 3.3975 times faster than the unoptimized algorithm.

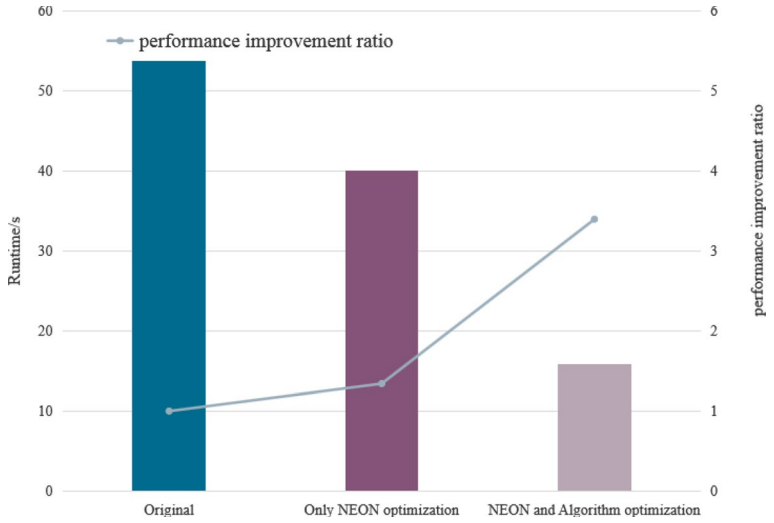


**Fig. 13** Comparison of run time on single node under different conditions

To explore the scalability of CitcomCu optimized by NEON and Krylov subspace on multiple nodes, we deployed the optimized CitcomCu on Tianhe and conducted a series of tests on different node scales. In this parallel efficiency test, we set the maximum number of iterations *maxstep* to 1000, double precision to 1.0e−6, and data grid size to $32^3$, and set the number of layers in the multi-grid to 3, with node scales ranging from 32 to 1024. The final experimental results are detailed in Table 3.

The parallel efficiency curve of CitcomCu is shown in Fig. 14. Compared with the scale of 32 nodes (about 50,000 computing cores), the parallel efficiency of unoptimized CitcomCu on 1024 nodes (about 1 million computing cores) is only 36.75%. However, after optimization with NEON and Krylov subspace, the parallel efficiency of CitcomCu for mantle convection numerical simulation was improved by 16.22% to reach 42.71%. It can be seen that with progressive data storage format encoding, modular design of parallelism, and efficient task scheduling improvements, the application program can perform large-scale numerical simulation at the level of millions of cores. At such a large scale, additional parallel overhead will lead to a decrease in parallel efficiency. However, a parallel efficiency of 42.71% is acceptable for weak scaling applications. Chen Xin and Gao Yingxiang tested large-scale parallel sparse matrix multiplication on the new-generation Sunway supercomputer. When expanded to 104,000 cores (1600 CGs), the parallel efficiency was about 58.2% (47.4%, 43.2%) for a system with 6146 (3074, 1538) atoms.

In order to fully utilize the high-performance advantages of the Tianhe platform and obtain the performance of large-scale parallel computing for geodynamics applications, we conducted large-scale numerical simulation experiments on 3328 to 26,624 nodes and tested the parallel efficiency under different node quantities. In this test, we set the maximum number of iterations *maxstep* to 500, double precision

**Table 3** Parallel efficiency test result of different scale below 1024 nodes (about one million computing cores)

| Nodes | Original run time/s | Parallel efficiency/% | Optimized run time/s (Gauss–Seidel) | Parallel efficiency/% (Gauss–Seidel) |
|---|---|---|---|---|
| 32 | 25.9307 | 100 | 16.5030 | 100 |
| 64 | 32.6262 | 79.48 | 19.9915 | 82.55 |
| 128 | 34.462 | 75.24 | 21.3024 | 77.47 |
| 256 | 40.7018 | 63.71 | 24.6903 | 66.84 |
| 512 | 53.4133 | 48.55 | 31.6574 | 52.13 |
| 1024 | 70.5615 | 36.75 | 41.7586 | 39.52 |

| Nodes | Optimized run time/s (NEON+Krylov) | Parallel efficiency/% (NEON+Krylov) |
|---|---|---|
| 32 | 7.9260 | 100 |
| 64 | 9.1715 | 86.42 |
| 128 | 9.7419 | 81.36 |
| 256 | 11.1807 | 70.89 |
| 512 | 14.2785 | 55.51 |
| 1024 | 18.5577 | 42.71 |

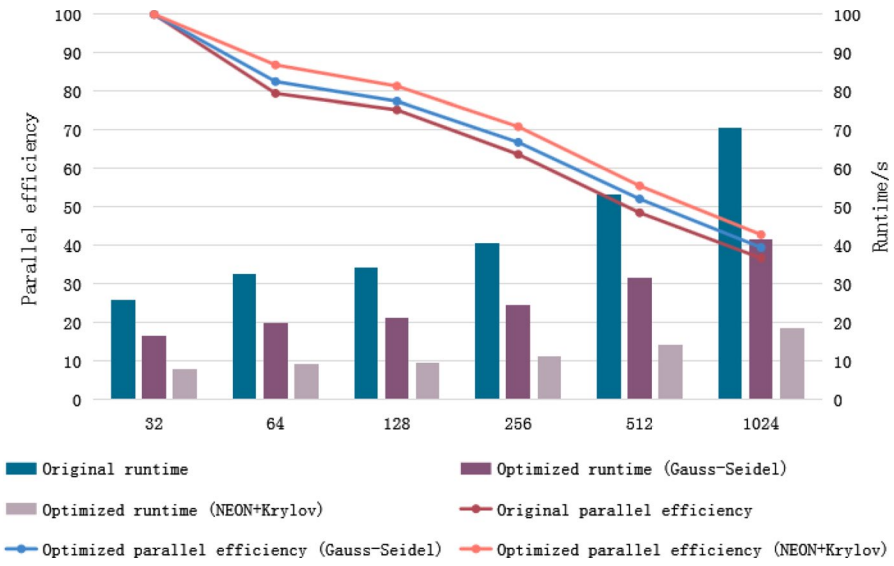


**Fig. 14** Parallel efficiency test result of different scale below 1024 nodes (about one million computing cores)

**Table 4** Larger-scale parallel run time and parallel efficiency

| Nodes | Run time/s | Parallel efficiency/% |
|---|---|---|
| 3328 | 37.5751 | 100 |
| 6656 | 57.3929 | 65.47 |
| 13,312 | 71.5444 | 52.52 |
| 26,624 | 102.8328 | 36.54 |

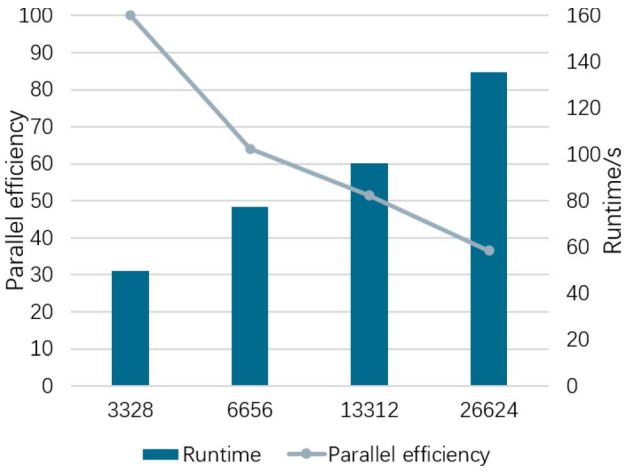


**Fig. 15** Larger-scale parallel efficiency of different nodes

to 1.0e−6, and data grid size to $32^3$, and set the number of layers in the multi-grid to 3. The final experimental results are shown in Table 4.

The test results are based on 3328 nodes, and the run time is the total time of 500 iterations. We computed the parallel efficiency of other nodes with 3328 nodes as the benchmark and obtained the parallel efficiency curve of geodynamic application in 3328 to 26,624 computing nodes, as shown in Fig. 15.

It can be seen that after the progressive data storage format encoding, parallel modular design, and efficient task scheduling improvements, large-scale operations can be successfully run, and geodynamic application can be extended to the whole Tianhe system. The parallel efficiency will drop significantly after reaching a certain scale. The main reason is that the MPI communication in the application will have greater loss performance under the influence of the bandwidth limitation of the communication network between nodes and the burden of large-scale data IO on the nodes with the expansion of the scale.

## 7 Conclusion

For high-performance heterogeneous computing platforms, important applications in different fields adapt large-scale parallel acceleration are the key requirement of current high-performance scientific computing numerical simulation. CitcomCu is an important application of geodynamics for the mantle convection simulation. In this paper, we implement the deployment and optimization of geodynamic application on current large-scale heterogeneous parallel computing platforms and provide the parallel computing framework for geodynamic numerical simulation. Firstly, based on the large-scale heterogeneous computing architecture, the data storage of CitcomCu is optimized, the overall communication overhead is reduced, and the load balance of data communication is ensured. Secondly, we improve the iterative solution algorithm of CitcomCu to accelerate the solving process. Finally, the NEON instructions based on SIMD are used for the matrix operation in the solving process to improve the parallel efficiency.

In the large-scale test at the level of ten millions cores, the imbalance between the element and the node solving domain will lead to poor scalability of large-scale data, and the scalability will have a step-like decline when it exceeds 10,000 cores. After more than one million cores, there will be next step-like decline. Nonetheless, the Tianhe new-generation high-performance computer can further improve the computational scale of numerical simulation applications in various fields and ensure certain parallel efficiency. These optimization strategy is suitable for common heterogeneous high-performance computing platform.

## Appendix A: Appendix

### A.1. Noun introduction table

See Appendix Table 5.

**Table 5** Noun introduction table

| Abbreviation | Full name |
|---|---|
| CitcomCu | The geodynamic numerical simulation application |
| Tianhe | Tianhe new-generation high-performance computer |
| Host side | General purpose zone |
| Device side | Acceleration zone |
| GSM | Global Shared Memory |
| HBSM | High Bandwidth Shared Memory |
| NEON | An ARM technology that combines 64-bit and 128-bit SIMD instruction sets |
| $M_x$, $M_y$, $M_z$ | Nodes number in three directions of the grid |
| K | The stiffness matrix |
| G | The discrete gradient operator |
| $G^T$ | The discrete gradient operator transpose |
| V | The velocity at all the nodal points |
| P | The pressure at all the pressure nodes |
| F | The total force term resulting from the derivation process |
| $V_k$ | The velocity of iteration step k |
| $P_k$ | The pressure of iteration step k |
| $r_k$ | The residual for pressure equation of iteration step k |

## A.2. Input File

```
# 1. Input and Output Files Information
   datafile="CASE2/caseA"
   use_scratch="local"
   oldfile="CASE2/caseA"
   restart=0
   restart_timesteps=20000
   stokes_flow_only=0
   maxstep=1000
   storage_spacing=50
   # 2. Geometry, Ra numbers, Internal heating, Thermochemical/Purely thermal convection
   Solver=multigrid node_assemble=1
   rayleigh=10.97394e5
   rayleigh_comp=1e6
   composition=0
   Q0=0
   Q0_enriched=0
   markers_per_ele=15
   comp_depth=0.605
   visc_heating=0
   adi_heating=0
```

```
# 3. Grid And Multiprocessor Information
nprocx=16
nprocz=16
nprocy=8
nodex=33 nodez=33 nodey=33
mgunitx=32
mgunitz=32
mgunity=16
levels=3
# 4. Coordinate Information
Geometry=cart3d
dimenx=1.0
dimenz=1.0
dimeny=1.0
z_grid_layers=2
zz=0.0,1.0
nz=1,129
x_grid_layers=2
xx=0,1
nx=1,129
y_grid_layers=2
yy=0,1
ny=1,65
z_lmantle=0.76655052
z_410=0.857143
z_lith=0.9651568
# 5. Rheology
rheol=0
TDEPV=off
VISC_UPDATE=off
update_every_steps=2
num_mat=4
visc0=1.0e0,1.0e0,1.0e0,1.0e0
viscE=6.9077553,6.9077553,6.9077553,6.9077553
viscT=273,273,273,273
viscZ=5e-6,5e-6,5e-6,5e-6
SDEPV=off
sdepv_misfit=0.010
sdepv_expt=1,1,1,1
sdepv_trns=1.e0,1.e0,1.e0,1.e0
VMIN=on visc_min=5.0e-2
VMAX=on visc_max=2.0e04
visc_smooth_cycles=1
Viscosity=system
# 6. DIMENSIONAL INFORMATION and Depth-dependence
layerd=2870000.0
```

```
radius=6370000.0
ReferenceT=3800.0
refvisc=1.0e20
density=3300.0
thermdiff=1.0e-6
gravacc=9.8
thermexp=5e-5
cp=1250
wdensity=0.0
visc_factor=1.0
thermexp_factor=1.0
thermdiff_factor=1.00
dissipation_number=2.601
surf_temp=0.078947
# 7. phase changes: to turn off any of the phase changes, let Ra_XXX=0
Ra_410=0.0
Ra_670=0.0
clapeyron410=3.0e6
clapeyron670=-3.0e6
width410=3.5e4
width670=3.5e4
# 8. BOUNDARY CONDITIONS and Initial perturbations
topvbc=0
topvbxval=0.0
topvbyval=0.0
botvbc=0
botvbxval=0.0
botvbyval=0.0
toptbc=1 bottbc=1
toptbcval=0.0 bottbcval=1.0
periodicx=off
periodicy=off
flowthroughx=off
flowthroughy=off
num_perturbations=1
perturbmag=0.001
perturbk=1.0
perturbl=6.0
perturbm=0.0
# 9. SOLVER RELATED MATTERS
Problem=convection
aug_lagr=on
aug_number=1.0e3
precond=on
orthogonal=off
maxsub=1
```

```
viterations=2
mg_cycle=1
down_heavy=3
up_heavy=3
vlowstep=20
vhighstep=3
piterations=375
accuracy=1.0e-2
tole_compressibility=1e-7
# Tuning of energy equation
adv_sub_iterations=2
finetunedt=0.75
ll_max=20
nlong=180
nlati=90
# Data input and program debugging
DESCRIBE=off
BEGINNER=off
VERBOSE=off
verbose=off
COMPRESS=off
see_convergence=1
# vim:ts=8:sw=8
```

**Author contributions** The paper properly credits the meaningful contributions of all authors. All authors have been personally and actively involved in substantial work leading to the paper and will take public responsibility for its content.

**Funding** The research was partially funded by Key-Area Research and Development Program of Guangdong Province (2021B0101190004), the National Key R &D Program of China (Grant Nos. 2021YFB0300800), The Key Program of National Natural Science Foundation of China (Grant Nos. U21A20461, 92055213), The National Natural Science Foundation of China (Grant No. 61872127), Research on High Precision Numerical Simulation and Parallel Computing Method for Ion Implanted Silicon Carbide Semiconductor Doping Process (U21A20461).

**Data availability statement** Not applicable.

**Code availability** Not applicable.

## Declarations

**Conflict of interest** This material is the authors' own original work, which has not been previously published elsewhere. The paper is not currently being considered for publication elsewhere. The paper reflects the authors' own research and analysis in a truthful and complete manner.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

**Consent to participate** Not applicable.

**Consent for publication** Not applicable.

## References


1. Zhong S, Michael G, Moresi L (1998) Role of faults, nonlinear rheology, and viscosity structure in generating plates from instantaneous mantle flow models[J]. J Geophys Res 103:15255–15268. https://doi.org/10.1029/98JB00605
2. Zhong S, Zuber M, Moresi L, Gurnis M (2000) Role of temperature-dependent viscosity and surface plates in spherical shell models of mantle convection. J Geophys Res 105:11063–11082. https://doi.org/10.1029/2000JB900003
3. Assunção J, Sacek V (2017) Heat transfer regimes in mantle dynamics using the CitcomCU software. In: 15th International Congress of the Brazilian Geophysical Society and EXPOGEF, Rio de Janeiro, Brazil, 31 July-3. Brazilian Geophysical Society, pp 1636–1639. https://doi.org/10.1190/sbgf2017-318
4. Yang T, Moresi L, Gurnis M et al (2019) Contrasted East Asia and South America tectonics driven by deep mantle flow. Earth Planet Sci Lett 517:106–116. https://doi.org/10.1016/j.epsl.2019.04.025
5. Parmentier EM, Turcotte DL, Torrance KE (1976) Studies of finite amplitude non-Newtonian thermal convection with application to convection in the Earth's mantle. J Geophys Res 81(11):1839–1846. https://doi.org/10.1029/JB081i011p01839
6. Van Zelst I, Crameri F, Pusok AE et al (2022) 101 geodynamic modelling: how to design, interpret, and communicate numerical studies of the solid Earth. Solid Earth 13(3):583–637. https://doi.org/10.5194/se-13-583-2022
7. Moresi L, Gurnis M (1996) Constraints on the lateral strength of slabs from three-dimensional dynamic flow models. Earth Planet Sci Lett 138(1–4):15–28. https://doi.org/10.1016/0012-821X(95)00221-W
8. Zhong S (2005) constraints on thermochemical convection of the mantle from plume-related observations. In: AGU Spring Meeting Abstracts, V42A-01
9. Kronbichler M, Heister T, Bangerth W (2012) High accuracy mantle convection simulation through modern numerical methods. Geophys J Int 191(1):12–29. https://doi.org/10.1111/j.1365-246X.2012.05609.x
10. Morra G (2019) Pythonic geodynamics: implementations for fast computing on Jupyter notebooks. In: AGU Fall Meeting Abstracts. ED53F-0902
11. Kohl N, Thönnes D, Drzisga D et al (2019) The HyTeG finite-element software framework for scalable multigrid solvers. Int J Parallel Emergent Distrib Syst 34(5):477–496. https://doi.org/10.1080/17445760.2018.1506453
12. Fraters M, Thieulot C, Van Den Berg A et al (2019) The Geodynamic World Builder: a solution for complex initial conditions in numerical modeling. Solid Earth 10(5):1785–1807
13. Xiao J, Chen J, Zheng J, An H, Huang S, Yang C, Li F, Zhang Z, Huang Y, Han W, Liu X, Chen D, Liu Z, Zhuang G, Chen J, Li G, Sun X, Chen Q (2021) Symplectic structure-preserving particle-in-cell whole-volume simulation of tokamak plasmas to 111.3 trillion particles and 25.7 billion grids. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21). Association for Computing Machinery, New York, NY, USA, Article 2, pp 1–13. https://doi.org/10.1145/3458817.3487398
14. Liu Y, Liu X, Li F, Fu H, Yang Y, Song J, Zhao P, Wang Z, Peng D, Chen H, Guo C (2021) Closing the "quantum supremacy" gap: achieving real-time simulation of a random quantum circuit using a new Sunway supercomputer. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21). Association for Computing Machinery, New York, NY, USA, Article 3, pp 1-12. https://doi.org/10.1145/3458817.3487399
15. Shang H, Li F, Zhang Y, Zhang L, Fu Y, Gao Y, Wu Y, Duan X, Lin R, Liu X, Liu Y, Chen D (2021) Extreme-scale ab initio quantum Raman spectra simulations on the leadership HPC system in China. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21). Association for Computing Machinery, New York, NY, USA, Article 6, pp 1-13. https://doi.org/10.1145/3458817.3487402
16. Gómez CD (2019) Particle-in-cell finite element models of deformation surrounding the bend in the San Andreas fault. California State University, Northridge

17. Bauer S, Bunge HP, Drzisga D, et al. (2016) Hybrid parallel multigrid methods for geodynamical simulations. In: Software for Exascale Computing-SPPEXA 2013–2015. Springer, Cham, pp 211–235. https://doi.org/10.1007/978-3-319-40528-5_10
18. Assunção J, Sacek V (2017) Benchmark comparison study for mantle thermal convection using the CitcomCU numerical code. In: 15th International Congress of the Brazilian Geophysical Society and EXPOGEF, Rio de Janeiro, Brazil, 31 July-3. Brazilian Geophysical Society, pp 1630–1635. https://doi.org/10.1190/sbgf2017-317
19. Bauer S, Huber M, Ghelichkhan S et al (2019) Large-scale simulation of mantle convection based on a new matrix-free approach. J Comput Sci 31:60–76. https://doi.org/10.1016/j.jocs.2018.12.006
20. May D A, Sanan P, Rupp K, et al. Extreme-scale multigrid components within PETSc. In: Proceedings of the Platform for Advanced Scientific Computing Conference, pp 1–12. https://doi.org/10.1145/2929908.2929913
21. Bangerth W, Burstedde C, Heister T et al (2012) Algorithms and data structures for massively parallel generic adaptive finite element codes. ACM Trans Math Softw (TOMS) 38(2):1–28. https://doi.org/10.1145/2049673.2049678
22. Chen W, Dong X, Chen H et al (2021) Performance evaluation of convolutional neural network on Tianhe-3 prototype. J Supercomput 77(11):12647–12665. https://doi.org/10.1007/s11227-021-03759-8
23. Lu K, Wang Y, Guo Y, et al. (2022) MT-3000: a heterogeneous multi-zone processor for HPC. CCF Trans High Perform Comput. https://doi.org/10.1007/s42514-022-00095-y
24. Li J J, Li J, Yang Y, et al. (2022) A parallel ETD algorithm for large-scale rate theory simulation. J Supercomput. https://doi.org/10.1007/s11227-022-04434-2
25. Maccabe AB (2017) Operating and runtime systems challenges for HPC systems. In: Proceedings of the 7th International Workshop on Runtime and Operating Systems for Supercomputers ROSS 2017, p 1. https://doi.org/10.1145/3095770.3095771
26. Weng T, Zhou X, Li K, Peng P, Li K (2022) Efficient distributed approaches to core maintenance on large dynamic graphs. IEEE Trans Parallel Distrib Syst 33(1):129–143. https://doi.org/10.1109/TPDS.2021.3090759
27. Zhao T, Hall M, Johansen H, et al. (2021) Improving communication by optimizing on-node data movement with data layout. In: Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp 304–317. https://doi.org/10.1145/3437801.3441598
28. Weng T, Zhou X, Li K, Tan K-L, Li K (2023) Distributed approaches to butterfly analysis on large dynamic bipartite graphs. IEEE Trans Parallel Distrib Syst 34(2):431–445. https://doi.org/10.1109/TPDS.2022.3221821
29. Žaloudek L, Sekanina L (2011) Increasing fault-tolerance in cellular automata-based systems. In: Calude CS, Kari J, Petre I, Rozenberg G (eds) Unconventional Computation. UC 2011. Lecture Notes in Computer Science, vol 6714. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21341-0_26
30. Zhong S, Yuen D, Moresi L (2007) Numerical methods for mantle convection. In: Treatise on geophysics, vol 7. Elsevier, pp 227–252. https://doi.org/10.1016/B978-044452748-6.00118-8
31. Song K, Li W, Zhang B, et al. (2022) Parallel design and implementation of Jacobi iterative algorithm based on ternary optical computer. J Supercomput. https://doi.org/10.1007/s11227-022-04471-x
32. Zhang K, Ding L, Cai Y, et al. (2017) A high performance real-time edge detection system with NEON. In: 2017 IEEE 12th International Conference on ASIC (ASICON). IEEE, pp 847–850. https://doi.org/10.1109/ASICON.2017.8252609
33. Chen X, Gao Y, Shang H et al (2022) Increasing the efficiency of massively parallel sparse matrix-matrix multiplication in first-principles calculation on the new-generation Sunway supercomputer. IEEE Trans Parallel Distrib Syst 33(12):4752–4766. https://doi.org/10.1109/TPDS.2022.3202518

## Authors and Affiliations

**Jin Yang[1] · Wangdong Yang[1] · Ruixuan Qi[1] · Qinyun Tsai[1] · Shengle Lin[1] · Fengkun Dong[1] · Kenli Li[1] · Keqin Li[1,2]**

✉ Wangdong Yang
yangwangdong@hnu.edu.cn

Jin Yang
yanjin96@hnu.edu.cn

Ruixuan Qi
s18010010@s.upc.edu.cn

Qinyun Tsai
hnutsai@hnu.edu.cn

Shengle Lin
lsl036@hnu.edu.cn

Fengkun Dong
dongfengkun@hnu.edu.cn

Kenli Li
lkl@hnu.edu.cn

Keqin Li
lik@newpaltz.edu

[1] College of Information Science and Engineering, Hunan University, Changsha 410082, Hunan, China

[2] Department of Computer Science, State University of New York, New Paltz 12561, NY, USA