Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Scalability and efficiency challenges for the exascale supercomputing system: practice of a parallel supporting environment on the Sunway exascale prototype system[*]

Xiaobin HE[†§], Xin CHEN[†§], Heng GUO, Xin LIU[†‡], Dexun CHEN[†‡], Yuling YANG, Jie GAO, Yunlong FENG, Longde CHEN, Xiaona DIAO, Zuoning CHEN

National Research Center of Parallel Computer Engineering and Technology, Beijing 100190, China

[†]E-mail: hexiaobin_1984@163.com; ischen.xin@foxmail.com; yyylx@263.net; adch@263.net

Received Sept. 25, 2022; Revision accepted Nov. 29, 2022; Crosschecked Dec. 8, 2022

Abstract: With the continuous improvement of supercomputer performance and the integration of artificial intelligence with traditional scientific computing, the scale of applications is gradually increasing, from millions to tens of millions of computing cores, which raises great challenges to achieve high scalability and efficiency of parallel applications on super-large-scale systems. Taking the Sunway exascale prototype system as an example, in this paper we first analyze the challenges of high scalability and high efficiency for parallel applications in the exascale era. To overcome these challenges, the optimization technologies used in the parallel supporting environment software on the Sunway exascale prototype system are highlighted, including the parallel operating system, input/output (I/O) optimization technology, ultra-large-scale parallel debugging technology, 10-million-core parallel algorithm, and mixed-precision method. Parallel operating systems and I/O optimization technology mainly support large-scale system scaling, while the ultra-large-scale parallel debugging technology, 10-million-core parallel algorithm, and mixed-precision method mainly enhance the efficiency of large-scale applications. Finally, the contributions to various applications running on the Sunway exascale prototype system are introduced, verifying the effectiveness of the parallel supporting environment design.

Key words: Parallel computing; Sunway; Ultra-large-scale; Supercomputer
https://doi.org/10.1631/FITEE.2200412 **CLC number:** TP302

1 Introduction

China's high-performance computing (HPC) is entering the exascale era. Compared with the existing Sunway TaihuLight—China's strongest supercomputer system, the number of computing cores of the exascale system will be greatly increased and will be much more than 10 million. The growth in the scale of the exascale system will release huge computing power. Taking into account the scalability and efficiency of parallel applications under such a huge parallel scale raises great challenges to the design of the exascale system. In addition, with the increasing demand for computing power in artificial intelligence (AI) applications, the fusion of AI and HPC applications has become an important breakthrough direction for HPC applications (Kurth et al., 2018; Jia et al., 2020). It is also a challenge to support the high scalability and efficiency of AI applications in the exascale era of HPC.

The major challenges are as follows:

1. Scalable management to support parallel applications

[§] These two authors contributed equally to this work
[‡] Corresponding authors
[*] Project supported by the Key R&D Program of Zhejiang Province, China (No. 2022C01250) and the National Key R&D Program of China (No. 2019YFA0709402)

ORCID: Xiaobin HE, https://orcid.org/0000-0001-6785-1561; Xin CHEN, https://orcid.org/0000-0002-0562-0319

In recent years, the performance improvement of the HPC system depends much more on the increase of the number of processor cores. The number of computing cores of Sunway TaihuLight (Fu et al., 2016) has exceeded 10 million, and the number of computing cores of the exascale supercomputer will be much more than 10 million. Therefore, scalable management is required to flexibly support parallel applications to achieve the partitioning, startup, detection, low-power control, and stopping of the parallel program running on tens of millions of cores, to ensure the high efficiency of application management.

2. High-concurrency input/output (I/O) to support parallel applications

In the exascale era, due to the unprecedented application scale of HPC, the amount of data generated during the running of the application also increases explosively. The process number of parallel data access may reach tens of thousands or even tens of millions. It is difficult for the traditional single construction with the shared storage system to meet the application requirements. Therefore, it is necessary to design a data access technique with large capacity, high bandwidth, high concurrency, and low cost for exascale applications, which can flexibly support concurrent data access of the parallel application running on tens of millions of computing cores.

3. Scalable debugging and tuning to support parallel applications

In the exascale era, due to the huge scale of application of HPC, it is common to encounter abnormal running of applications. In addition, to discover potential bottlenecks with respect to performance, scalability, and so on, it is necessary to collect massive application running information, so as to conduct data sampling during the application running (Lin et al., 2021). With the sharp increase in the system scale and application complexity, the amount of debugging information data also increases dramatically. Traditional methods have been unable to meet the requirements of scalability and complexity of parallel applications in the exascale era. Therefore, it is necessary to implement a debugging and tuning mechanism oriented to the characteristics of the exascale system, reducing the interference of data collection and analysis on applications, and improving the efficiency of debugging and tuning in super-large-scale scenarios.

4. Efficient parallel algorithms to support parallel applications running on tens of millions of computing cores

In the exascale era, to maximize the computing capability, HPC system developers often need to design complex heterogeneous on-chip storage and interconnection systems (Hluchý et al., 2020). Meanwhile, to support the super-large-scale system, it is necessary to build a complex network interconnection system. It is a great challenge to design parallel applications that can maximize the potential of the system in terms of computing and network interconnection. Therefore, it is necessary to refine the common requirements of HPC applications, provide a scientific computing parallel framework for specific supercomputing processors and interconnection network platforms, reduce the programming's complexity, and ensure the efficient running of applications.

5. Efficient support for AI applications

AI applications show a trend of integration with traditional HPC, which shows higher requirements for HPC systems in the exascale era. The purpose is to adapt to the special configuration of precision and computing power in the field of AI, improve the computing efficiency by reducing the precision, and reduce the pressure of data on memory and communication. Therefore, providing an AI-oriented parallel application ecology to support the efficient running of large-scale AI applications on HPC systems is a necessity.

As a representative in the field of supercomputing in China, the Sunway exascale prototype system (SEPS) is an exploration to verify the technology route of the next-generation Sunway supercomputing. The system equipped with the SW26010pro many-core processors and the self-developed high-speed network is deployed in the National Supercomputing Center in Jinan. Considering the above five challenges, the system implements a collaborative design and optimization for the parallel supporting environment, including the parallel operating system, the storage system, the debugging and tuning system, the scientific and engineering parallel application framework, and the AI ecosystem, and a series of innovative technologies are proposed. The applicability of the above techniques to exascale supercomputers has been verified by the important application achievements acquired recently on SEPS.

2 SEPS architecture

SEPS is a small-scale verification system that is designed according to the exascale application requirements, and can be scaled up to the exascale level. The architecture of SEPS is similar to that of Sunway TaihuLight. As shown in Fig. 1, SEPS consists of the computing system, the computing network, the storage system, the management cluster, the application debugging server, and the application server. The computing system is built using SW26010pro processors, which are interconnected through the self-developed Sunway computing network. The storage system consists of the burst buffer and disk storage, provides global shared storage services for the computing nodes, and is interconnected with the whole system through the self-developed Sunway network. The management cluster provides the management functions for the whole system. The application debugging server provides debugging and tuning services for the applications. The application server provides services such as application compiling, submitting, and viewing for users.

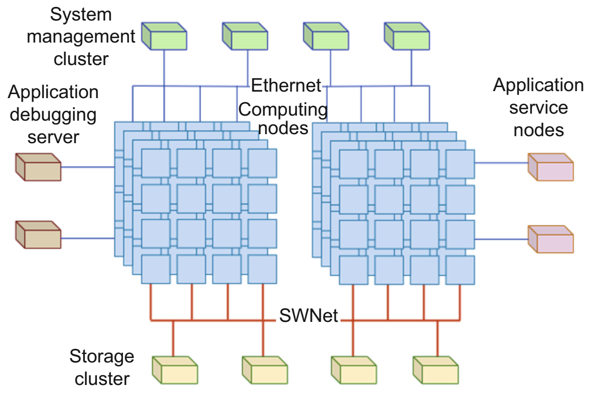

System management cluster
Application debugging server
Ethernet
Computing nodes
Application service nodes
SWNet
Storage cluster

Fig. 1 Architecture of the Sunway exascale prototype system

The system is equipped with the heterogeneous many-core processor SW26010pro, which contains 390 computing cores. The architecture of the many-core processor SW26010pro is shown in Fig. 2. All computing cores in SW26010pro are divided into six core groups (CGs), and each CG consists of one management processing element (MPE) and 64 computing processing elements (CPEs), providing powerful computing capabilities. The system achieves network interconnection through Sunway's self-developed network, supports the remote direct memory access (RDMA) communication protocol, and has large network bandwidth, which supports large-scale applications for high-speed message passing interface (MPI) communication and

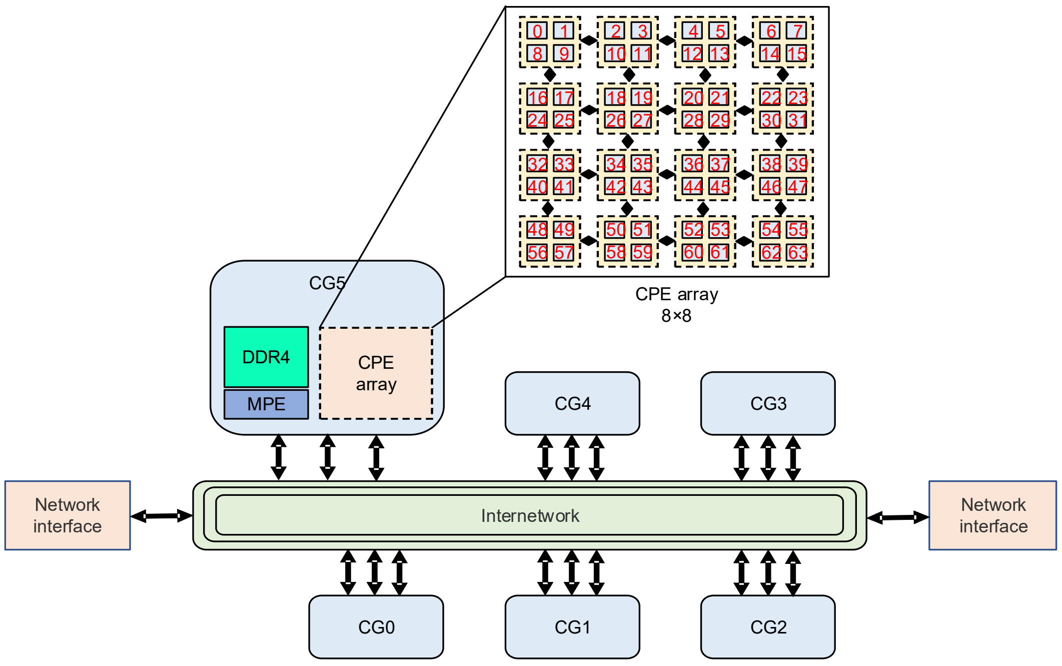


Fig. 2 Architecture of the many-core processor SW26010pro (CG: core group; CPE: computing processing element; DDR: double data rate; MPE: management processing element)

I/O. Moreover, applications running on SEPS can be scaled up to 10 million computing cores.

3 Design of the parallel supporting environment

The parallel supporting environment of SEPS provides a comprehensive basic parallel operating environment for parallel applications. As shown in Fig. 3, it is composed of the parallel operating system, the distributed storage system, the debugging and tuning system, the scientific computing parallel framework, and the AI ecosystem.

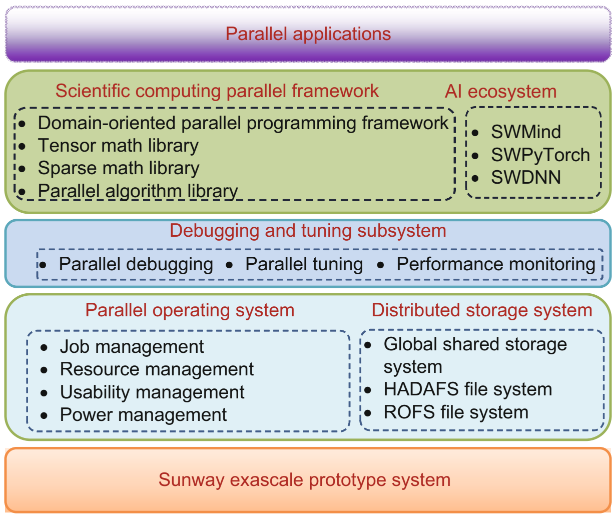

Parallel applications
Scientific computing parallel framework
AI ecosystem
• Domain-oriented parallel programming framework
• Tensor math library
• Sparse math library
• Parallel algorithm library
• SWMind
• SWPyTorch
• SWDNN
Debugging and tuning subsystem
• Parallel debugging • Parallel tuning • Performance monitoring
Parallel operating system
Distributed storage system
• Job management
• Resource management
• Usability management
• Power management
• Global shared storage system
• HADAFS file system
• ROFS file system
Sunway exascale prototype system

Fig. 3 Components of the parallel supporting environment (DNN: deep neural network; ROFS: read-only file system; HADAFS: HADA file system)

3.1 Parallel operating system

The resource management for the whole system on SEPS enables users to execute their jobs concurrently and achieve the unified management, monitoring, and on-demand allocation for many-core computing nodes. The software helps the system continue to run even if some failure occurs, and helps control the energy consumption of applications. The structure of the parallel operating system is shown in Fig. 4.

Job management adopts the adaptive multilevel parallel architecture to support the efficient management of user jobs in large-scale systems. It can achieve efficient start-up and operation control of large-scale parallel jobs and provide users with rich functions, strong scalability, and a convenient job environment.

Resource management provides the efficient management and allocation of different granularities and heterogeneous resources for the large-scale system. It adopts the hierarchical parallel and interlayer pipeline control mode to replace the traditional large-scale one-to-many control with a simpler, multiple parallel one-to-many control. In this way, the control pressure of the master is reduced, and the scalability of the system is greatly improved.

Usability management improves the fault-tolerant operation ability of the system and supports the reliable operation of large-scale applications through the fault-tolerant mechanism for typical application characteristics.

Power management adopts a job-driven power-capping control method to balance performance and energy issues effectively. With this method, the large-scale system can operate efficiently in terms of energy usage.

3.2 Storage system

SEPS carries a hybrid storage system, consisting of a global file system (GFS), a burst buffer file system (HADAFS), and a read-only file system (ROFS). The storage architecture is shown in Fig. 5. The three file systems use different mount paths, so the applications could choose different paths to store as per their needs: GFS for general application data access, HADAFS for high-performance burst data storage, and ROFS for storage of AI application datasets or dynamic libraries.

GFS is the most commonly used unit, adopting a forwarding architecture. The bottom-level GFS is built on the dedicated I/O nodes and the disk arrays based on the Lustre file system (Ma et al., 2012). The upper-level lightweight file system (LWFS) provides services for forwarding I/O requests from computing nodes to the backend parallel file system, which provides an interface compatible with portable operating system interface (POSIX) semantics. The server of LWFS is deployed to the I/O forwarding node, and the client is deployed on the computing node. HADAFS is constructed based on the non-volatile memory express solid-state drive (NVMe-SSD) of the I/O forwarding node to obtain high bandwidth, which provides a unified metadata view and a POSIX-like I/O interface. ROFS uses the SSD

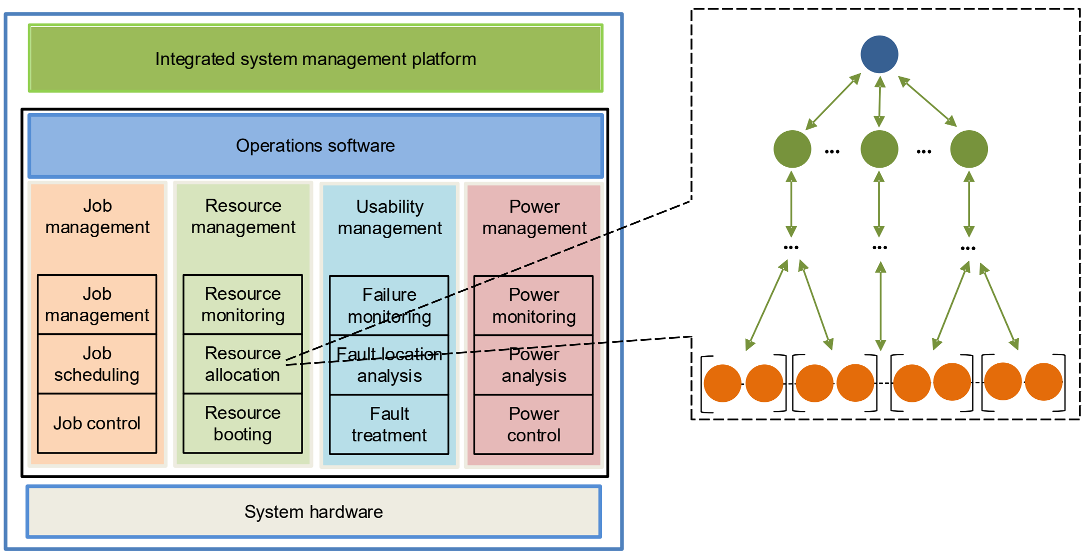


Fig. 4 Parallel operating system structure

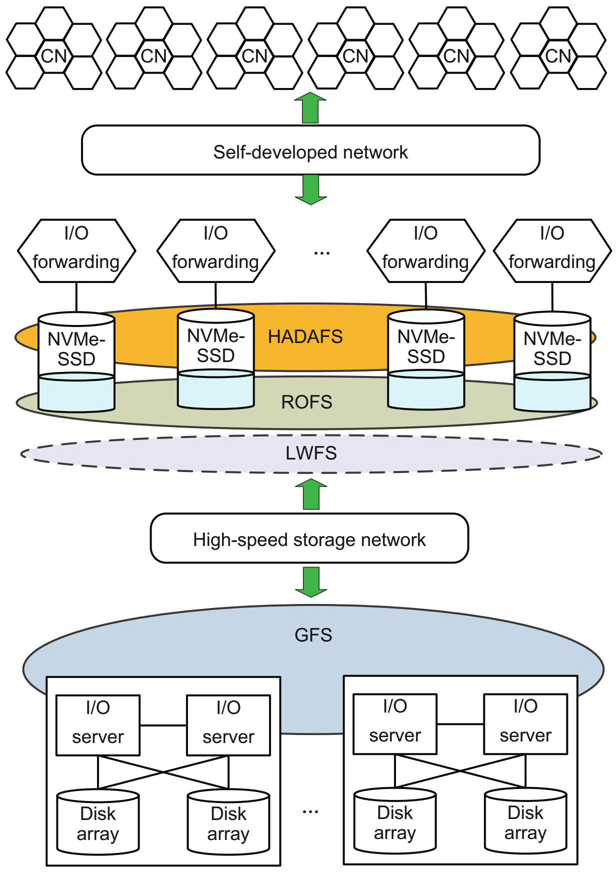


Fig. 5 Sunway storage architecture (CN: computing node; GFS: global file system; HADAFS: HADA file system; I/O: input/output; LWFS: lightweight file system; NVMe-SSD: nonvolatile memory express solid-state drive; ROFS: read-only file system)

of the I/O forwarding node and the read-only local virtual file system (VFS) provided by the computing node to provide a read-only local VFS to support the requirements of AI software and applications for large-scale data burst reads.

GFS: GFS is a globally unified storage system based on Lustre and compatible with the standard POSIX file system interfaces. Its client is mounted on the I/O forwarding node and provides global file-sharing services for computing nodes through the LWFS forwarding service. Lustre implements active-active hot standby at multiple levels of the network, equipment, and services, with strong reliability and availability, and no single point of failure. The system implements a data distribution algorithm based on performance cognition (Yang et al., 2019), which can effectively avoid I/O interference between different applications, and avoid faults or performance reduction of object storage technology (OST) devices, ensuring that the applications have better I/O performance. GFS supports quality-of-service (QoS) control of storage performance, and administrators can set performance limits for specific applications (Shi et al., 2017; Hua et al., 2019). The above work is also completely transparent to the application. The storage system also deploys a beacon performance monitoring tool (Chen et al., 2020; Yang et al., 2022), which can monitor the application's I/O mode and performance in real time, and provide an important reference for application I/O optimization.

Moreover, GFS supports dynamic allocation of storage resources according to application requirements (Ji et al., 2019).

HADAFS: To meet the ultra-high-bandwidth requirements of data reading and writing for some applications, SEPS deploys SSD-based burst buffer and the developed burst buffer storage software HADAFS. HADAFS aggregates the NVMe-SSD storage space on multiple I/O forwarding nodes and provides it to users through resource groups. The format of the HADAFS interface is similar to that of the POSIX system call interface. HADAFS separates data management and data access; therefore, a set of data management tools named HADASH is designed to implement metadata query and data migration between Lustre and HADAFS. To fully use the capacity and bandwidth of NVMe-SSD, HADAFS does not provide high-reliability capabilities. It is critical to make full use of the performance of SSD (Shi et al., 2017), so HADAFS does not use the data redundancy mechanism. HADAFS supports the layout of data to the nearest SSD of the I/O forwarding node, and allocates resources to the application based on the group of the I/O forwarding node. The administrator can dynamically reorganize the group according to the application needs, so as to achieve better resource utilization.

ROFS: With the development of emerging applications such as AI and data analysis, some applications need to repeatedly read a large number of files and therefore require a high level of concurrent read performance. However, it is difficult to meet these requirements with the traditional parallel file system. In response to this problem, ROFS is designed and developed based on Internet small computer systems interface (ISCSI) technology. It directly maps part of the SSD space to the computing node to provide a remote local disk for the computing node with POSIX-compatible interfaces for application. For each computing node, the reading process of the application is equivalent to local disk access, and the metadata performance is significantly improved. When running AI applications on the whole system, the data reading performance of the AI algorithm library and data set is improved by more than 100 times compared with the original version. Since ROFS is exported in read-only mode, multiple computing nodes correspond to a fixed server, and the mapping between computing nodes and servers is static.

3.3 Debugging and tuning subsystem

The debugging and tuning subsystem provides developers with tools to develop a large-scale application in less time (Fig. 6). Debugger, parallel debugging tool, and large-scale lightweight debugging tool make up the debugging part. They support single-node applications, medium-scale applications, and large-scale applications, respectively. The tuning part includes a job-level performance monitoring tool, which can quickly obtain a performance overview of the job, and a performance monitoring library, which provides fine-grained analysis capabilities.

The debugger provides a unified debugging view of MPE and CPE by threading abstraction, supporting source- and instruction-level program execution control. The parallel debugging tool is built on the debugger, which can support fine-grained execution control for thousands of parallel processes. The

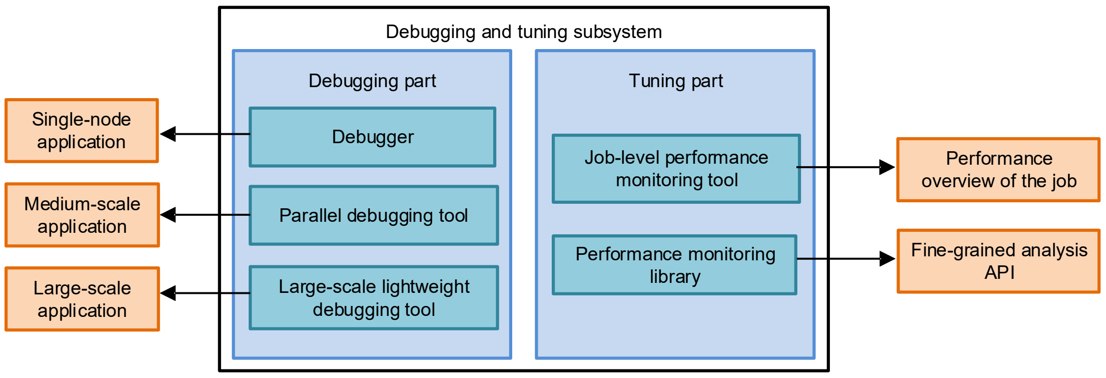

Debugging and tuning subsystem
Debugging part
Tuning part
Single-node application
Medium-scale application
Large-scale application
Debugger
Parallel debugging tool
Large-scale lightweight debugging tool
Job-level performance monitoring tool
Performance monitoring library
Performance overview of the job
Fine-grained analysis API

Fig. 6 Overview of the debugging and tuning subsystem (API: application programming interface)

large-scale lightweight debugging tool locates abnormal processes by clustering important process states and can analyze applications with tens of millions of computing cores in less than 1 min.

The job-level performance monitoring tool is available to users in the form of a job submission option, without recompiling the application. The performance monitoring library provides rich application programming interfaces (APIs) for obtaining the running information of the application, and supports efficient queries for floating point operations per second (FLOPS), instructions per cycle (IPC), cache miss ratio, and so on.

3.4 Scientific computing parallel framework

To reduce the programming's complexity caused by the heterogeneous many-core architecture on SEPS, a scientific computing parallel framework is designed to meet the common requirements for science and engineering computing. The framework provides a template or paradigm of multilevel parallel programming for applications. As shown in Fig. 7, the framework is composed mainly of two parts. One part is a domain-oriented parallel programming framework, including the SW-Gromacs software in molecular dynamics simulation (Berendsen et al., 1995; Lindahl et al., 2001) and SW-Vina software in molecular docking simulation (Trott and Olson, 2009). The other part is the kernel-level parallel programming framework for common kernels in polymorphic applications, including the tensor math library, sparse math library, and parallel algorithm library. The functions of the framework are as follows:

1. The domain-oriented parallel programming framework enables users to easily and efficiently migrate applications developed in some domains to the many-core architecture. For example, SW-Gromacs is developed based on the open-source software Gromacs (Berendsen et al., 1995; Lindahl et al., 2001). This software supports basic dynamics-related algorithms, including Newtonian mechanics, energy minimization, and regular pattern analysis. SW-Vina is designed based on the open source molecular docking software Vina (Trott and Olson, 2009). This software supports the simultaneous docking of multiple ligands. In addition, this software provides many-core optimized versions for multiple docking functions.

2. The tensor math library supports the efficient many-core implementation of common tensor operations, such as general (dense) matrix multiply (GEMM) and tensor transpose, and provides common interfaces with the standard mathematical format.

3. The sparse math library supports the efficient many-core implementation of commonly used sparse algebra operations, such as sparse matrix-vector multiplication (SPMV) (Merrill and Garland, 2017) and sparse general matrix-matrix multiplication (SPGEMM) (Buluc and Gilbert, 2012), and provides common interfaces with various sparse matrix storage formats, including compressed sparse row (CSR), compressed sparse column (CSC), and coordinate (COO) formats.

4. The parallel algorithm library provides advanced parallel algorithms, such as the dynamic load-balancing algorithm and the discrete memory access optimization algorithm. Moreover, generic interfaces and usage examples for these algorithms are given.

The parallel computing framework provides a parallel programming model on SEPS for domain-related and public computing requirements, hiding the complexity of the underlying implementation on the SW26010pro processor and network, effectively reducing the difficulty of transplanting and optimizing codes, accelerating heterogeneous mapping, and efficiently operating for multifield applications on the many-core processor.

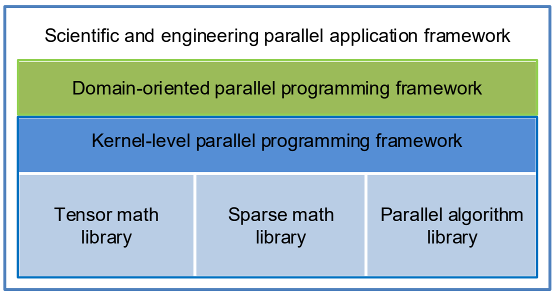

Scientific and engineering parallel application framework
Domain-oriented parallel programming framework
Kernel-level parallel programming framework
Tensor math library
Sparse math library
Parallel algorithm library

Fig. 7 Scientific computing parallel framework

3.5 SW AI ecosystem

The SW AI ecosystem is built on massively scalable AI frameworks (SWMind and SWPyTorch) and SWDNN, whose composition is shown in Fig. 8. Based on the system software and runtime, the AI ecosystem provides model-developing tools and an efficient/scalable running environment for AI applications.

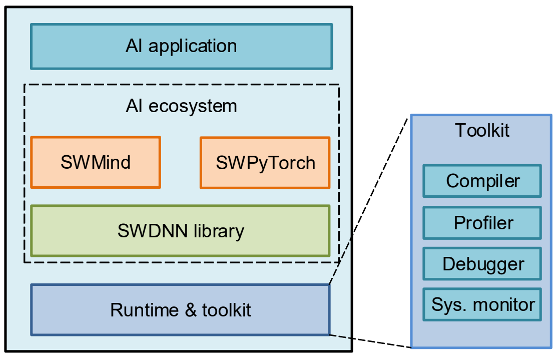


Fig. 8 The position of the AI ecosystem in Sunway software (AI: artificial intelligence; DNN: deep neural network; SW: Sunway; Sys.: system)

1. SWDNN. SWDNN is an accelerated library based on SEPS for DNNs (Liu S et al., 2021). It provides highly optimized 32- or 16-bit floating point (FP32 and FP16, respectively) function interfaces for these frequently used operators in DNN applications, including convolution, normalization, pooling, and activation functions. By invoking the SWDNN library, users can focus on the construction, training, and application of NNs, without spending time on performance optimization. Users need only to replace the function of the original program with the corresponding operator in the SWDNN library to achieve model acceleration. The following optimization techniques are developed in SWDNN:

On-chip memory access optimization: Since the memory access bandwidth between CPEs in a CG is higher than that between the CPE and the main memory, we need to make full use of the on-chip communication to obtain useful data. In many DNN operators, row and column broadcasts in a CG are effective ways to share data. In addition, the dual buffering technique is widely used to hide communications and calculations. We also optimize malloc specifically to ensure that the assigned data addresses are aligned to achieve efficient memory access.

On-chip CG memory sharing: Typically, the optimized operators in SWDNN are designed based on a single CG. However, for some special operators, such as the "embedding" and "bmm" operators, their required memory far exceeds the capacity of a single CG. To solve this problem, we develop a CG memory sharing model; i.e., one MPE occupies the memory of the entire processor, and all CPEs on the whole processor are jointly scheduled. Assuming that N CGs are used, we need to divide the computation tasks into $N \times 64$ components to achieve high-efficiency parallelism.

2. SWMind. For simplicity, efficiency, and ease of use, we also design a lightweight deep learning framework, SWMind, which features simple interfaces, optimized communication, and lightweight mode. Most importantly, the framework provides many-core accelerated operators and supports large-scale parallelism on SEPS. By invoking the designed efficient operators, users can easily construct their own NN models and write various custom operators. Compared with existing frameworks, such as Tensorflow and PyTorch, SWMind has higher performance due to elimination of nonessential modules.

3. SWPyTorch. Similarly, the main mission of SWPyTorch on SEPS is to achieve many-core acceleration of PyTorch and scalability on large-scale computing nodes. To achieve many-core acceleration, the original serial implementation is replaced by the corresponding operator in the SWDNN library. The backend to support distributed data-parallel (DDP) in SWPyTorch is the MPI. Due to the support of DDP, model parallelism and data parallelism are achieved. Moreover, we modify the implementation of Megatron (Shoeybi et al., 2019), a natural language processing (NLP) model, to enable it to run on the Sunway platform with SWPyTorch and achieve an efficient, large-scale hybrid model and data parallel pretraining with mixed precision.

4 Optimization technologies

With the expansion of the system scale in the exascale era, achieving high scalability and efficiency has become an important challenge for the efficient operation of applications. As a bridge between applications and the supercomputing system, the parallel supporting environment has become the key to overcoming this challenge. Therefore, the parallel supporting environment on SEPS provides comprehensive support for the application in terms of scalability and efficiency.

4.1 Scalability optimizations of the exascale parallel supporting environment

Scalability optimization includes the I/O optimization technology and the debugging technique for large-scale parallel applications. The goal of I/O optimization is to provide diversified storage options and solve the problem of concurrent data reading and writing for the parallel application running on tens of millions of computing cores. The debugging technique is a solution to replace the traditional debugging tools that cannot support parallel debugging for the application running on tens of millions of computing cores.

4.1.1 I/O optimization

The data forwarding software used by the Sunway series supercomputers is LWFS, whose server runs in the data forwarding node and the client runs in the computing node. Using Filesystem in USErspace (FUSE) to support Linux-standard file system interfaces, LWFS has the advantage of high compatibility. In this way, the I/O requests from the application running on the computing nodes need to first enter the kernel FUSE module, and then be transferred from the kernel mode to the user mode. In practice, the two switchings (between the kernel mode and user mode) and memory copies lead to a large software overhead. Therefore, we propose a user-mode direct data access LIBIO library for LWFS users (Fig. 9).

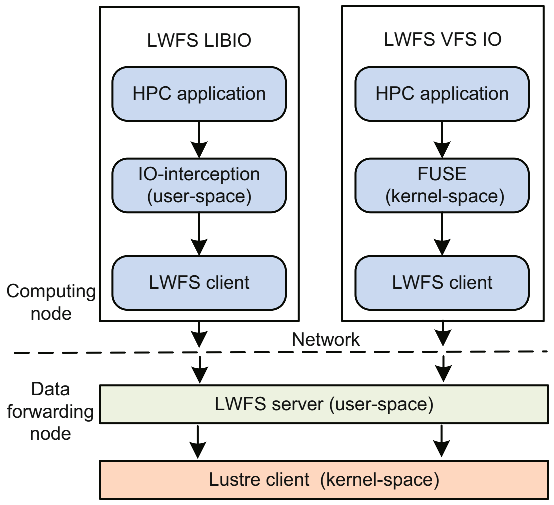


Fig. 9 Software architecture of LWFS LIBIO (FUSE: Filesystem in USErspace; HPC: high-performance computing; IO: input-output; LWFS: lightweight file system; VFS: virtual file system)

LIBIO calls the functions of the LWFS client port in a library manner, including an application request intercepting component and a standard LWFS client port component. When the application is running, the request intercepting component can intercept all I/O requests of the application and transfer them to the LWFS client port integrated into LIBIO, and then send them to the server port to execute. When LIBIO is used, the user does not need to modify the code; the user has to just link the LIBIO library in the compilation stage. Moreover, the access mode in the running application is exactly the same as that of the traditional kernel file system. Through the LIBIO library, the single-process bandwidth of a computing node is increased by more than twice, and the aggregate bandwidth of a single computing node is increased by more than five times.

Facing the endless I/O requirements of applications, for the new generation of Sunway storage system, optimization schemes have been designed for different I/O modes. A new file system HADAFS is designed and developed. The software stack of HADAFS is loosely coupled with the traditional VFS, which relaxes the POSIX semantics and retains only the necessary interface design. The main features of HADAFS are as follows: (1) adopting a flat data organization method, which breaks the limitation of the traditional directory tree structure and achieves a high degree of scalability; (2) using the rocksdb-based key-value database to store metadata, which significantly improves metadata performance; (3) separating data access and data management, as well as streamlining storage semantics, which yields better I/O performance; (4) loose coupling between software stack and traditional VFS, which guarantees a simple extension interface and strong flexibility in cache management and data management. Considering the hierarchical data format-5 (HDF5), network common data form (NetCDF), and other file formats widely used in scientific computing, we also transplant HADAFS to support reading and writing HDF5 and NetCDF files through HADAFS.

For massive data read scenarios in AI applications, we develop ROFS. Data updates for ROFS are done by users with distributed dedicated tools. The mounting of ROFS is automatically completed by the job scheduling system. ROFS is mounted before the starting of the job and unmounted after the completeness of the job. Users need only to use a fixed mount point to access the ROFS. With the help of ROFS, the load speed of the dynamic library of the AI ecosystem in the large-scale application is 100 times higher than that of the GFS scheme based on Lustre and LWFS.

4.1.2 Scalable debugging technique for parallel applications

As the scale of the parallel application grows, it becomes more difficult and time-consuming to locate errors that occur during runtime. Many attempts have been made to address this problem, but the current advances in debugging techniques are far outpaced by the increasing need for complex debugging, and the gap between debugging tools and debugging requirements is getting wider and wider.

We propose a method that can quickly locate abnormal processes in very-large-scale applications (Fig. 10). Our approach includes mainly two aspects: (1) the important state data of the program are obtained through the baseboard management controller (BMC) interface; (2) the abnormal process will be located using our analysis strategies. The BMC interface provides access to hardware registers and can obtain data on the entire system quickly by a hierarchical architecture. We can obtain more data by extending the interface to more registers, such as the power consumption, the performance monitoring unit (PMU), and the status of the memory component on the nodes where the process resides. By optimizing our focus registers, we are able to collect data from tens of millions of computing cores in a few seconds (Peng et al., 2022).

The design of the analysis strategy depends on the metrics we choose. During the running of the application, there will be a corresponding program execution space, which contains a large number of program execution states; this space can reflect the program anomalies. Therefore, we can locate abnormal processes by analyzing anomalies in certain metrics. The program counter (PC) value, which indicates where the process is currently running, is the metric we choose. Our analysis strategy includes two parts: vertical and horizontal. In the vertical aspect, we analyze the abnormal PC change sequence; in the horizontal aspect, we cluster the PC values between different task processes to find the abnormal classes.

Our method achieves the co-design of software and hardware by combining the hardware BMC system. The proposed method has good scalability, and its capabilities can be extended by collecting other program execution states and designing new analysis strategies (Hofer and Mössenböck, 2014).

4.2 Parallel efficiency optimizations of the exascale parallel supporting environment

For an exascale-oriented new-generation supercomputer, the efficient mapping of computing tasks contained in the specific application to 10-million-level computing cores is the key to high-efficiency operation for the large-scale application. In this subsection, we propose ultra-large-scale parallel algorithms for the heterogeneous many-core architecture to ensure efficient implementation of mapping.

4.2.1 Multilevel parallelization mode

To adapt to the hierarchical memory design of the heterogeneous many-core architecture, the corresponding multilevel parallelization mode is proposed for large-scale parallel computing (Fig. 11).

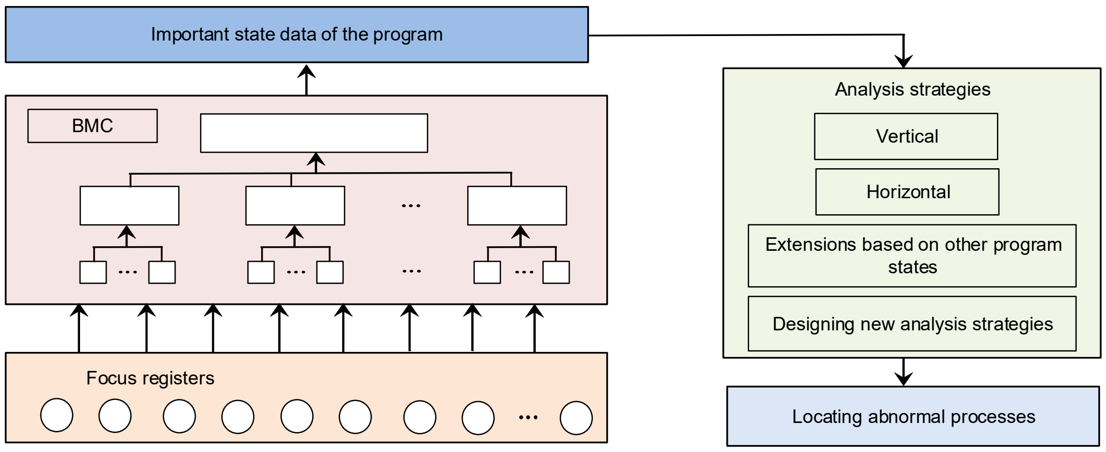

Important state data of the program
BMC
Focus registers
Analysis strategies
Vertical
Horizontal
Extensions based on other program states
Designing new analysis strategies
Locating abnormal processes

Fig. 10 Main idea of the very-large-scale debugging method (BMC: baseboard management controller)

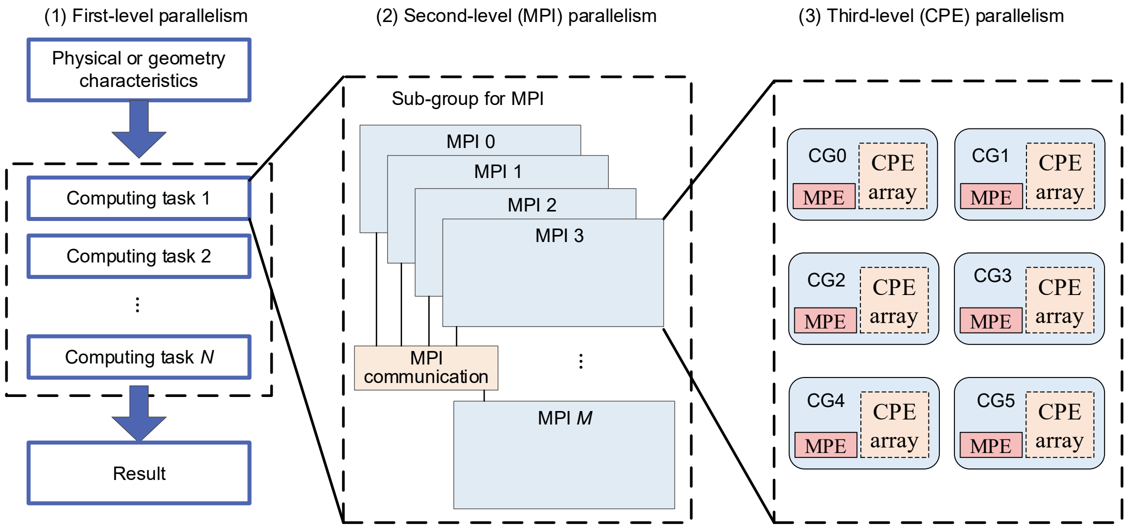


Fig. 11 Illustration of the multilevel parallelization mode (CG: core group; CPE: computing processing element; MPE: management processing element; MPI: message passing interface)

At the first level, the computing tasks in the specific application are divided into multiple independent tasks according to physical or geometry characteristics. In this stage, the whole processor pool is divided into multiple subgroups, and each subgroup is responsible for calculating an independent task. Usually, no communication between subgroups is required at this level. At the second level, i.e., process-level (MPI) parallelism, the independent task within the subgroup is mapped to each MPI process, which generally involves communication between MPI processes. The third level is thread-level (CPE) parallelism, where computing tasks are further subdivided into CPEs within the MPI process. In this way, the basic mapping of the computing tasks from the application level to the core level has been completed.

4.2.2 Adaptive load-balancing algorithm

At the process level (MPI), the possibility of computational load imbalance is high, especially for those dynamic applications, in which computation and data movement change over time. To solve the problem, an adaptive load-balancing algorithm is proposed, and the particle-in-cell (PIC) simulation (Madduri et al., 2011; Derouillat et al., 2018) is used as an example to introduce in detail.

For the PIC method, it is easy to raise the computational load imbalance due to the unbalanced particle distribution. The proposed adaptive load-balancing algorithm is illustrated in Fig. 12. The idea of the original spatial partitioning method (subfigure on the left) is that each process tracks and computes the particles in its range. The improved load-balancing method involves particle sharing, which ensures that the number of particles calculated by each process is as equal as possible (different colors indicate different processes in the figure). Finally, the particles responsible for this process are further distributed into the CPE array for parallel computation.

4.3 Mixed-precision optimization for AI applications

When the scale of applications gradually increases, the performance of the program is usually limited by bandwidth and memory; thus, we adopt a mixed-precision method (Micikevicius et al., 2018) to reduce the demand for both. The mixed-precision method refers to a mixture of single-precision float32 and half-precision float16, which uses half-precision to store data with less memory, reducing memory and bandwidth pressure. The floating-point data format used on the SW26010pro processor follows the IEEE-754 standard. FP32 and FP16 are shown in Fig. 13.

Regarding the mixed-precision method, two innovations are proposed:

First, we design an adaptive scaling method to dynamically adjust the data, so that the representation of the main data is always within the range of

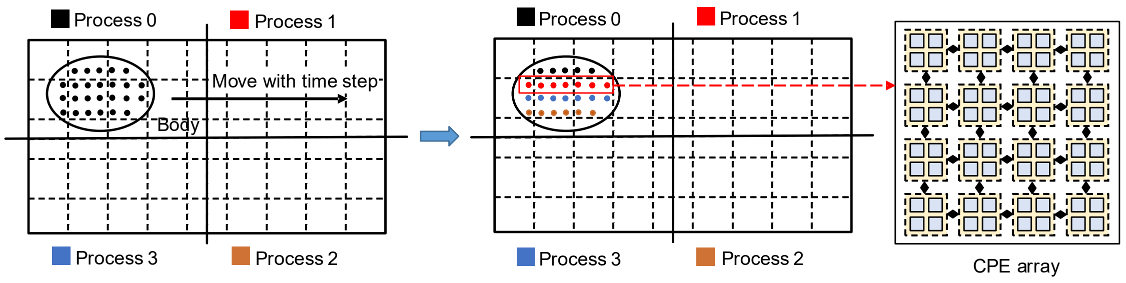


Fig. 12 Illustration of the adaptive load-balancing algorithm (CPE: computing processing element) (References to color refer to the online version of this figure)

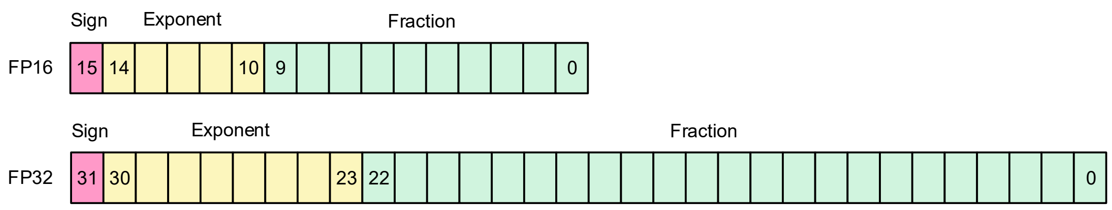


Fig. 13 FP16 and FP32 (16- and 32-bit floating points, respectively) schemes

FP16, and the error is kept at a level similar to that of using only single-precision floating-point numbers. When the error r is greater than the maximum, i.e., "max," we will reduce the data; when the error r is less than the minimum, i.e., "min," we will enlarge the data (Table 1). This method effectively prevents data underflow and achieves a better balance between accuracy and efficiency.

Second, we design a filter whose main function is to filter out the overflow in the calculation process. According to our method, the percentage of overflow is less than 2%, so only a small part of the overflow result will be discarded, which has little effect on the result.

5 Contributions to various applications

5.1 Scientific and engineering computing applications

Benefiting from the high performance of SEPS and the support of the proposed parallel support environment, hundreds of applications in various fields

Table 1 Adaptive precision scaling

Error analysis	Scaling statistics
$\Delta_r > \Delta_{\max}$	Reduce
$\Delta_{\min} \le \Delta_r \le \Delta_{\max}$	Unchanged
$\Delta_r < \Delta_{\min}$	Enlarge

have been deployed on the system. These applications show great scalability potential in large-scale systems. Fig. 14 shows the expected scalability and efficiency of several outstanding applications running on SEPS, and deployment of the application in a much larger similar system has proved that the expected scalability and efficiency have been reached (Gu et al., 2021; Li F et al., 2021; Liu Y et al., 2021; Shang et al., 2021a, 2021b, 2021c; Xiao et al., 2021; Ye et al., 2022). Among them, the random quantum circuit (RQC) simulation, Raman spectra simulation, and tokamak plasmas simulation have all been shortlisted for the 2021 Gordon Bell Prize due to significant progress in these fields. An overview of the first two applications is listed here, and the details of the quantum simulator will be elaborated in Section 5.3.

1. Whole-volume magnetic confinement toroidal plasmas simulation (Xiao et al., 2021)

A large-scale simulation of magnetic toroidal plasmas with 1.5 trillion particles is performed on SEPS. It is the first time that such an unprecedented high-resolution evolution of six-dimensional (6D) electromagnetic fully kinetic plasmas has been presented. This work can also investigate edge micro-instabilities directly. This application requires highly concurrent data read and write during running. The HADAFS proposed in this paper is used

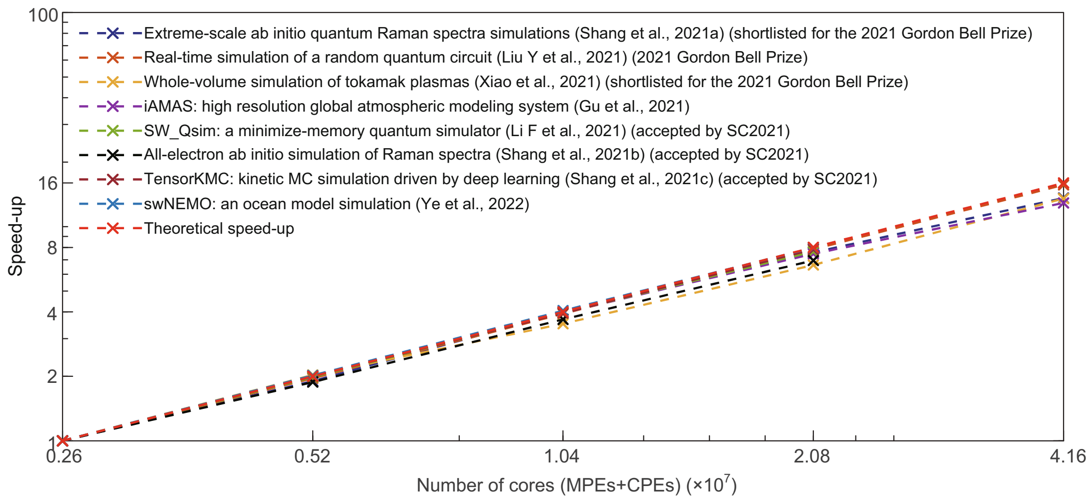


Fig. 14 Expected scalability and efficiency of multiple outstanding applications on SEPS (CPE: computing processing element; KMC: kinetic Monte Carlo; MPE: management processing element; SEPS: Sunway exascale prototype system; SC: International Conference for High Performance Computing, Networking, Storage and Analysis). References to color refer to the online version of this figure

to build a high-speed file system based on SSD, which supports burst data output, ensures application scalability, and reduces the time cost of the I/O segment. In addition, the application uses the proposed scalable debugging technology, effectively supporting the expansion of the parallel scale. Meanwhile, multilevel parallelism mode (process level and thread level) is used to improve parallel efficiency. Finally, the scale of the simulation is expanded to about 40 million computing cores, with strong scalable efficiency of up to 87.5% and weak scalable efficiency of up to 95.6%.

2. Extreme-scale ab initio quantum Raman spectra simulation (Shang et al., 2021a)

An accurate and massively parallel ab initio Raman spectra simulation for a real biological system (consisting of 3006 atoms) is performed on SEPS with great strong and weak scaling. This work has also shown the possibility to use the quantum mechanical (QM) method for virtual drug screening. The main challenge of this application is to parallelize the perturbation method. The proposed multilevel parallelism mode is used to construct a three-level parallelism strategy, which achieves large-scale application expansion and ensures a parallel efficiency of more than 80%. Moreover, a similar adaptive load-balancing algorithm is used to reduce the overhead caused by the unbalanced batch distribution, and the parallel efficiency is further improved. In the algorithm, the running time and space distribution of each batch are collected in the first iteration. Then, the distributions of all batches in all processes are recalculated according to the result of the first iteration. Finally, load balancing is achieved by keeping the distance between batches within a process as small as possible and the running time of each process as close as possible. In both strong and weak scaling tests, the parallel efficiency exceeds 80% when the number of computing cores reaches about 20 million.

5.2 AI applications

1. Wu Dao 2.0

Wu Dao (meaning enlightenment in Chinese) 2.0 trained on SW HPC was the world's biggest natural language processing (NLP) model when it was released. The model size of Wu Dao 2.0 is 10 times larger than that of generative pretrained transformer (GPT) 3, using 1.75 trillion parameters. Wu Dao 2.0 efficiently expands to tens of millions of cores on the whole HPC system. In addition, Wu Dao 2.0 explores the scalable boundary of the pretrained model, and is close to passing the Turing test over multiple tasks such as image generation, machine Q&A, and image description. Wu Dao 2.0 is multimodal, consists of WenLan, WenYuan, WenHui, and WenSu, and provides different skills, including text generation, image recognition, and image generation.

WuDaoCorpora is a super-large-scale Chinese corpora for pretraining language models. Wu Dao 2.0 acquires skills covering both Chinese and English by studying 4.9 terabytes (TB) of images and texts, including 1.2 TB of Chinese and English texts. Wu Dao 2.0 can also learn from text and images and tackle tasks that include both types of data (something GPT-3 cannot do).

2. Fusion of AI and HPC (Shang et al., 2021c; Li MF et al., 2022)

The fusion of AI and HPC has made new progress. In a recent study, through a deep convolutional neural network (CNN) used to fit the quantum multi-body variational wave function, the quantum multi-body simulation for the two-dimensional (2D) highly frustrated J1–J2 Heisenberg model is implemented. It is worth emphasizing that in this study the optimized ROFS supported in the parallel supporting environment is used to improve the efficiency of loading TensorFlow dynamic libraries by approximately 15–100 times compared with the original version. Finally, this study investigates various quantum systems with different lattice sizes (including 10×10, 16×16, and 24×24), as shown in Fig. 15. The parallel scalability is expected to extend to 31 850 000 computing cores, and the parallel efficiency of CNN calculation reaches 92.2% with lattice size 24×24.

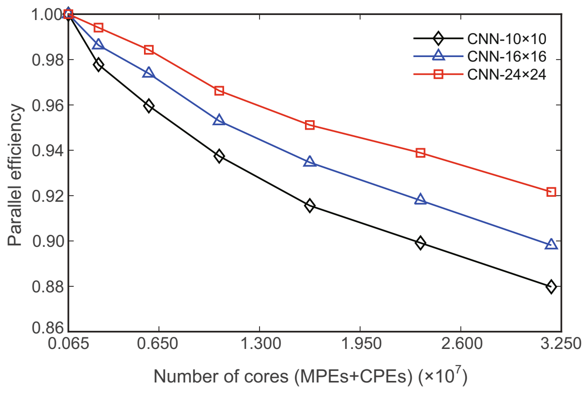


Fig. 15 Expected scalability test for CNN calculation (CNN: convolutional neural network; CPE: computing processing element; MPE: management processing element)

Another research team has proposed a TensorKMC method, combining atomic kinetic Monte Carlo (AKMC) and neural network potential (NNP, used for potential energy prediction) and performed a dynamic simulation of 54 trillion atoms on SEPS. In multi-layer neural network computing, each layer needs data input (from the main memory) and output (to the main memory) once, which leads to the overall speed being limited by the memory access speed. To solve this problem, a big-fusion operator is constructed using the proposed two-level parallelization mode. The tasks of multiple layers are merged into a kernel, so that the data are only input in the first layer and output in the last layer, and all computing tasks are mapped to CPEs. The big-fusion operator transforms the original memory-intensive task into a computation-intensive task. Finally, the number of computing cores increases to >20 million computing cores with a good parallel efficiency of 82%.

5.3 Quantum simulation exploration for new computing forms

SW_Qsim (Li F et al., 2021; Liu Y et al., 2021) is the first attempt of SW HPC in the field of quantum computing. It is a high-performance tensor-based simulator (Markov and Shi, 2008) for random quantum circuits (RQCs). It can simulate up to 10×10 (qubits) $\times$ $(1 + 40 + 1)$ (depth) RQCs, which is the world's largest RQC to be simulated using a classic supercomputer. In 2019, Google developed the quantum processor Sycamore (Arute et al., 2019), and claimed that it took about 200 s to sample a QC 1 million times on Sycamore, while the equivalent task cost about 10 000 years in the world's fastest supercomputer Summit at that time. In 2018, the National Aeronautics and Space Administration (NASA) and Google implemented qFlex (Villalonga et al., 2020), a circuit simulator, which has been redesigned and reimplemented to use the graphics processing unit (GPU) accelerated Summit HPC architecture efficiently; it achieved a peak performance of 92% when simulating circuits of 7×7 (qubits) $\times (1+40+1)$ (depth). SW_Qsim uses the Projected Entangled-Pair States (PEPS) method (Guo et al., 2019, 2021) and three-dimensional contraction skills (Huang et al., 2020; Pan and Zhang, 2021) to perform quantum simulations and achieves a peak performance of >90% when simulating circuits of $10 \times 10 \times (1+40+1)$, reducing the simulation time from 10 000 years to 304 s, which closes the "quantum supremacy" gap between quantum computers and classical supercomputers. The results show a near-linear strong and weak scaling pattern when the number of cores increases from 200 000 to 41 million under different circuits.

The realization of SW_Qsim on the new-generation Sunway supercomputer is due mainly to two innovations mentioned in the previous section, namely, the ultra-large-scale parallel algorithm and the mixed-precision method.

1. Ultra-large-scale parallel computing of tensor contraction

As the core of RQC simulation based on the PEPS method, the calculation process of contracting a closed quantum tensor network to a scalar is the bottleneck. We have encountered the double challenges of computing and storage. Taking the 10×10 (qubits) $\times$ $(1+40+1)$ (depth) RQC simulation as an example, to solve the memory problem, we adopt a trick in tensor contraction called "cut" (Villalonga et al., 2019). We cut the legs of the tensor network and divide the tensor contraction task into 32^6 embarrassing parallel tasks, thus alleviating the storage pressure of the tensor network on the central processing unit (CPU). Secondly, we adopt thread-level parallelism on CPEs, assigning each task to each CPE for calculation, with an unprecedented parallel scale of 41 932 800 cores. The process is shown in Fig. 16.

2. Mixed-precision computation method in tensor multiplication

For each step of tensor contraction, we use the mixed-precision method via adaptive precision scaling (Fig. 17), which greatly reduces the calculation time. The mixed-precision method is used mainly in the core section, that is, the part of tensor multiplication. First, we convert the part of the input two tensors that need to be multiplied into half-precision, and analyze whether it is within the range of FP16. If not, then we scale the data and set a global variable to record the zoom factor. Secondly, we import the half-precision input data into the local directive memory (LDM) through direct memory access (DMA), and use vectorization to accelerate the transformation of half-precision data to single-precision data to perform the calculations. Finally,

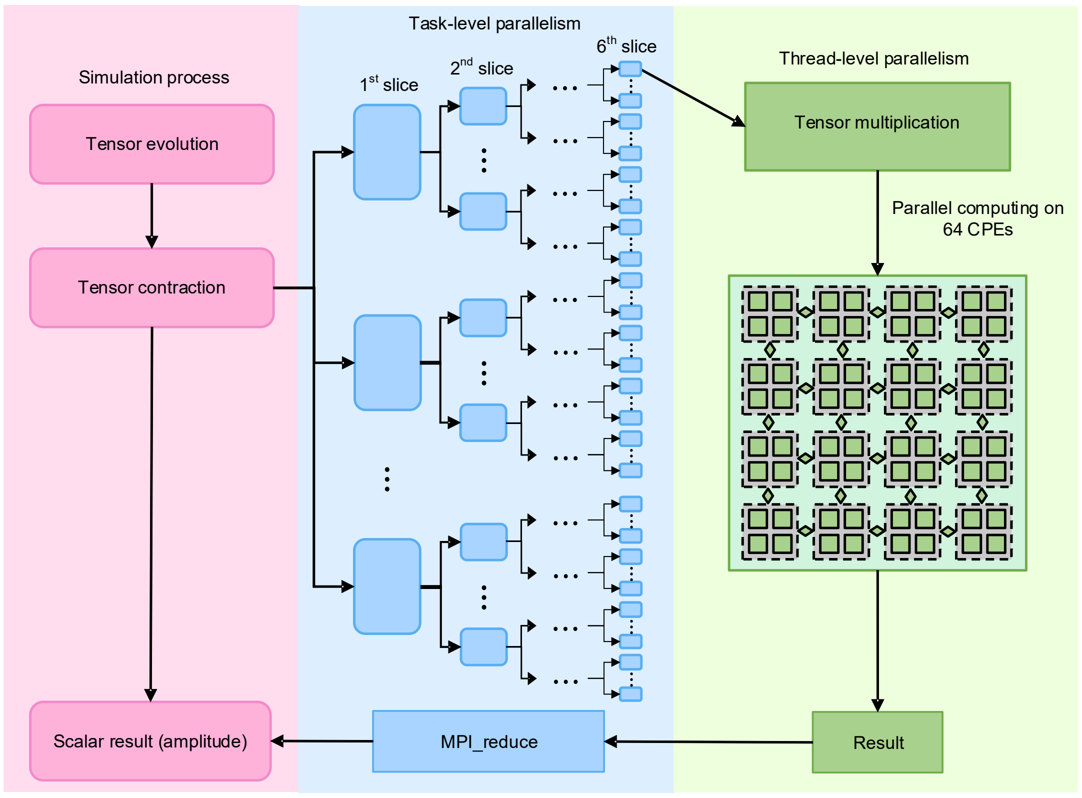


Fig. 16 Three-level parallelism of quantum simulation (CPE: computing processing element; MPI: message passing interface)

the single-precision calculation result is used as the input for the next step to continue the loop.

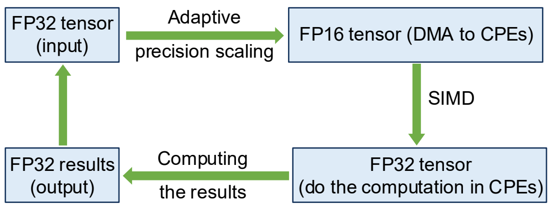


Fig. 17 Flowchart of the mixed-precision method (DMA: direct memory access; FP32 and FP16: 32- and 16-bit floating points; SIMD: single instruction multiple data; CPE: computing processing element)

6 Conclusions and future work

In the exascale era of HPC, scalability and efficiency become the key to determining the application value of the exascale supercomputer, and the challenges are growing rapidly with the expansion of the scale of the parallel system. This paper introduces the design of a parallel application support environment of SEPS and analyzes the key technologies of software in guaranteeing the scalability and efficiency of exascale applications. At present, various contributions have been made, and nearly 30 applications are expected to efficiently scale to tens of millions of computing cores, in which three applications were shortlisted for the highest award in the field of HPC applications "2021 Gordon Bell Award," five applications were accepted by SC 2021, and two applications were accepted by PPoPP 2022. Looking to the future, new complex applications reveal novel characteristics, such as computation and data movement varying with time, discreteness and sparseness caused by data non-locality, macro-scale complex task flow, micro-scale complex instruction flow, and mixed or variable precision. Given these new features, we will carry out research on novel parallel algorithms and parallel supporting software design and optimization methods to prosper application development in the field of HPC and improve the domestic supercomputer software ecosystem.

Contributors

Xin LIU and Dexun CHEN designed the research. Heng GUO, Yuling YANG, and Jie GAO performed the simulations. Yunlong FENG and Longde CHEN analyzed the results. Xiaobin HE and Xin CHEN drafted the paper. Xiaona DIAO and Zuoning CHEN helped organize the paper. Xiaobin HE and Xin CHEN revised and finalized the paper.

Compliance with ethics guidelines

Xiaobin HE, Xin CHEN, Heng GUO, Xin LIU, Dexun CHEN, Yuling YANG, Jie GAO, Yunlong FENG, Longde CHEN, Xiaona DIAO, and Zuoning CHEN declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding authors upon reasonable request.

References

Arute F, Arya K, Babbush R, et al., 2019. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505-510. https://doi.org/10.1038/s41586-019-1666-5

Berendsen HJC, van der Spoel D, van Drunen R, 1995. Gromacs: a message-passing parallel molecular dynamics implementation. *Comput Phys Commun*, 91(1-3):43-56. https://doi.org/10.1016/0010-4655(95)00042-E

Buluc A, Gilbert JR, 2012. Parallel sparse matrix-matrix multiplication and indexing: implementation and experiments. *SIAM J Sci Comput*, 34(4):C170-C191. https://doi.org/10.1137/110848244

Chen Q, Chen K, Chen ZN, et al., 2020. Lessons learned from optimizing the Sunway storage system for higher application I/O performance. *J Comput Sci Technol*, 35(1):47-60. https://doi.org/10.1007/s11390-020-9798-5

Derouillat J, Beck A, Pérez F, et al., 2018. SMILEI: a collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation. *Comput Phys Commun*, 222:351-373. https://doi.org/10.1016/j.cpc.2017.09.024

Fu HH, Liao JF, Yang JZ, et al., 2016. The Sunway TaihuLight supercomputer: system and applications. *Sci China Inform Sci*, 59(7):072001. https://doi.org/10.1007/s11432-016-5588-7

Gu J, Feng JW, Hao XY, et al., 2021. Establishing a non-hydrostatic global atmospheric modeling system (iAMAS) at 3-km horizontal resolution with online integrated aerosol feedbacks on the Sunway supercomputer of China. https://arxiv.org/abs/2112.04668v1

Guo C, Liu Y, Xiong M, et al., 2019. General-purpose quantum circuit simulator with projected entangled-pair states and the quantum supremacy frontier. *Phys Rev Lett*, 123(19):190501. https://doi.org/10.1103/PhysRevLett.123.190501

Guo C, Zhao YW, Huang HL, 2021. Verifying random quantum circuits with arbitrary geometry using tensor network states algorithm. *Phys Rev Lett*, 126(7):070502. https://doi.org/10.1103/PhysRevLett.126.070502

Hluchý L, Bobák M, Müller H, et al., 2020. Heterogeneous exascale computing. In: Kovács L, Haidegger T, Szakál A (Eds.), Recent Advances in Intelligent Engineering. Springer, Cham, p.81-110. https://doi.org/10.1007/978-3-030-14350-3_5

Hofer P, Mössenböck H, 2014. Efficient and accurate stack trace sampling in the Java hotspot virtual machine. Proc 5th ACM/SPEC Int Conf on Performance Engineering, p.277-280. https://doi.org/10.1145/2568088.2576759

Hua Y, Shi X, Jin H, et al., 2019. Software-defined QoS for I/O in exascale computing. *CCF Trans High Perform Comput*, 1(1):49-59. https://doi.org/10.1007/s42514-019-00005-9

Huang C, Zhang F, Newman M, et al., 2020. Classical simulation of quantum supremacy circuits. https://arxiv.org/abs/2005.06787

Ji X, Yang B, Zhang TY, et al., 2019. Automatic, application-aware I/O forwarding resource allocation. Proc 17th USENIX Conf on File and Storage Technologies, p.265-279.

Jia WL, Wang H, Chen MH, et al., 2020. Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, p.1-14. https://doi.org/10.1109/SC41405.2020.00009

Kurth T, Treichler S, Romero J, et al., 2018. Exascale deep learning for climate analytics. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, p.649-660. https://doi.org/10.1109/SC.2018.00054

Li F, Liu X, Liu Y, et al., 2021. SW_Qsim: a minimize-memory quantum simulator with high-performance on a new Sunway supercomputer. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, p.1-13.

Li MF, Chen JS, Xiao Q, et al., 2022. Bridging the gap between deep learning and frustrated quantum spin system for extreme-scale simulations on new generation of Sunway supercomputer. *IEEE Trans Parall Distrib Syst*, 33(11):2846-2859. https://doi.org/10.1109/TPDS.2022.3145163

Lin F, Liu Y, Guo YY, et al., 2021. ELS: emulation system for debugging and tuning large-scale parallel programs on small clusters. *J Supercomput*, 77(2):1635-1666. https://doi.org/10.1007/s11227-020-03319-6

Lindahl E, Hess B, van der Spoel D, 2001. GROMACS 3.0: a package for molecular simulation and trajectory analysis. *J Mol Model*, 7(8):306-317. https://doi.org/10.1007/s008940100045

Liu S, Gao J, Liu X, et al., 2021. Establishing high performance AI ecosystem on Sunway platform. *CCF Trans High Perform Comput*, 3(3):224-241. https://doi.org/10.1007/s42514-021-00072-x

Liu Y, Liu X, Li F, et al., 2021. Closing the "quantum supremacy" gap: achieving real-time simulation of a random quantum circuit using a new Sunway supercomputer. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, Article 3. https://doi.org/10.1145/3458817.3487399

Ma YJ, Lv S, Liu YQ, 2012. Introduction and application of cluster file system Lustre. *Sci Technol Inform*, (5):139-140 (in Chinese).

Madduri K, Ibrahim KZ, Williams S, et al., 2011. Gyrokinetic toroidal simulations on leading multi- and many-core HPC systems. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, p.1-12. https://doi.org/10.1145/2063384.2063415

Markov IL, Shi YY, 2008. Simulating quantum computation by contracting tensor networks. *SIAM J Comput*, 38(3):963-981. https://doi.org/10.1137/050644756

Merrill D, Garland M, 2017. Merge-based parallel sparse matrix-vector multiplication. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, p.678-689. https://doi.org/10.1109/SC.2016.57

Micikevicius P, Narang S, Alben J, et al., 2018. Mixed precision training. Proc 6th Int Conf on Learning Representations.

Pan F, Zhang P, 2021. Simulating the Sycamore quantum supremacy circuits. https://arxiv.org/abs/2103.03074v1

Peng D, Feng Y, Liu Y, et al., 2022. Jdebug: a fast, non-intrusive and scalable fault locating tool for ten-million-scale parallel applications. *IEEE Trans Parall Distrib Syst*, 33(12):3491-3504. https://doi.org/10.1109/TPDS.2022.3157690

Shang HH, Li F, Zhang YQ, et al., 2021a. Extreme-scale ab initio quantum Raman spectra simulations on the leadership HPC system in China. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, Article 6. https://doi.org/10.1145/3458817.3487402

Shang HH, Li F, Zhang YQ, et al., 2021b. Accelerating all-electron ab initio simulation of Raman spectra for biological systems. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, Article 41. https://doi.org/10.1145/3458817.3476160

Shang HH, Chen X, Gao XY, et al., 2021c. TensorKMC: kinetic Monte Carlo simulation of 50 trillion atoms driven by deep learning on a new generation of Sunway supercomputer. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, Article 73. https://doi.org/10.1145/3458817.3476174

Shi X, Li M, Liu W, et al., 2017. SSDUP: a traffic-aware SSD burst buffer for HPC systems. Proc Int Conf on Supercomputing, p.1-10. https://doi.org/10.1145/3079079.3079087

Shoeybi M, Patwary M, Puri R, et al., 2019. Megatron-LM: training multi-billion parameter language models using model parallelism. https://arxiv.org/abs/1909.08053

Trott O, Olson AJ, 2009. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J Comput Chem*, 31(2):455-461. https://doi.org/10.1002/jcc.21334

Villalonga B, Boixo S, Nelson B, et al., 2019. A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware. *NPJ Quant Inform*, 5(1):86. https://doi.org/10.1038/s41534-019-0196-1

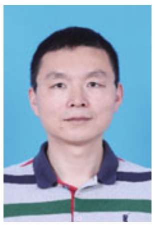
Villalonga B, Lyakh D, Boixo S, et al., 2020. Establishing the quantum supremacy frontier with a 281 Pflop/s simulation. *Quant Sci Technol*, 5(3):034003. https://doi.org/10.1088/2058-9565/ab7eeb

Xiao JY, Chen JS, Zheng JS, et al., 2021. Symplectic structure-preserving particle-in-cell whole-volume simulation of tokamak plasmas to 111.3 trillion particles and 25.7 billion grids. Proc Int Conf for High Performance Computing, Networking, Storage and Analysis, Article 2. https://doi.org/10.1145/3458817.3487398

Yang B, Ji X, Ma XS, et al., 2019. End-to-end I/O monitoring on a leading supercomputer. Proc 16th USENIX Conf on Networked Systems Design and Implementation, p.379-394.

Yang B, Zou YL, Liu WG, et al., 2022. An end-to-end and adaptive I/O optimization tool for modern HPC storage systems. IEEE Int Parallel and Distributed Processing Symp, p.1294-1304. https://doi.org/10.1109/IPDPS53621.2022.00128

Ye YJ, Song ZY, Zhou SC, et al., 2022. swNEMO_v4.0: an ocean model based on NEMO4 for the new-generation Sunway supercomputer. *Geosci Model Dev*, 15(14):5739-5756. https://doi.org/10.5194/gmd-15-5739-2022



Xiaobin HE received his BE degree from the Harbin Institute of Technology, Harbin, China, in 2006, and his MS degree from Shanghai Jiao Tong University, Shanghai, China, in 2009. He is currently an associate researcher at the National Research Center of Parallel Computer Engineering and Technology, Beijing, China. His main research interests include high-performance computing and distributed storage systems.

Xin CHEN received his BE degree from the National Digital Switching System Engineering & Technological Research Center (NDSC), Zhengzhou, China, in 2016, and his MS degree from NDSC in 2018. He is a research assistant at the National Research Center of Parallel Computer Engineering and Technology, Beijing, China. His research activities focus on high-performance parallel computation and applications.

Xin LIU received her PhD degree from PLA Information Engineering University, Zhengzhou, China, in 2006. She is currently a research fellow at the National Research Center of Parallel Computer Engineering and Technology, Beijing, China. She is a designer of the scientific and engineering application platform of the Sunway TaihuLight System, responsible for the large-scale parallel algorithm research and application software development. Her research interests include parallel algorithms and parallel application software.

Dexun CHEN received his PhD degree from Tsinghua University, Beijing, China, in 2021. He is currently a research fellow at the National Research Center of Parallel Computer Engineering and Technology, Beijing, China. His research interests include high-performance computing and parallel application software.