# swNEMO_v4.0: an ocean model NEMO4 for the next generation Sunway supercomputer

Yuejin Ye[1], Zhenya Song[2,3,4], Shengchang Zhou[2,4,5], Yao Liu[6], Qi Shu[2,3,4], Bingzhuo Wang[1], Weiguo Liu[4,5], Fangli Qiao[2,3,4], and Lanning Wang[4,7]

[1]National Supercomputing Center in Wuxi, 214000, China
[2]First Institute of Oceanography, and Key Laboratory of Marine Science and Numerical Modeling, Ministry of Natural Resources, Qingdao 266061, China
[3]Shandong Key Laboratory of Marine Science and Numerical Modeling, Qingdao 266061, China
[4]Laboratory for Regional Oceanography and Numerical Modeling, Pilot National Laboratory for Marine Science and Technology, Qingdao 266237, China
[5]Shandong University, Jinan 250101, China
[6]East China Normal University, Shanghai 200062, China
[7]Beijing Normal University, Beijing 100875, China

**Correspondence:** Fangli Qiao (email: qiaofl@fio.org.cn), Lanning Wang (email: wangln@bnu.edu.cn)

**Abstract.** The current large-scale parallel barrier of ocean general circulation models (OGCMs) makes it difficult to meet the computing demand of high resolution. Fully considering both the computational characteristics of OGCMs and the heterogeneous many-core architecture of the new Sunway supercomputer, swNEMO_v4.0 based on NEMO4 (Nucleus for European Modelling of the Ocean version 4), with ultrahigh scalability is developed. Three innovations and breakthroughs are shown in our work: (1) A highly adaptive, efficient four-level parallelization framework for OGCMs is proposed to release a new level of parallelism along the compute-dependency column dimension. (2) A many-core optimization method using blocking by remote memory access (RMA) and a dynamic cache scheduling strategy, effectively utilizing the temporal and spatial locality of data. The test shows that the actual direct memory access (DMA) bandwidth is greater than 90% of the ideal bandwidth after optimization, and the maximum is up to 95%. (3) A mixed-precision optimization method with half-, single-, and double-precision is explored, which can effectively improve the computation performance while maintaining the simulated accuracy of OGCMs. The results demonstrate that swNEMO has ultrahigh scalability, achieving up to 99.29% parallel efficiency with a resolution of 500 m using 27,988,480 cores, reaching the peak performance with 1.97 PFlops.

## 1 Introduction

Ocean general circulation models (OGCMs) are numerical models focusing on the properties of oceans based on the Navier-Stokes equations on the rotating sphere with thermodynamic terms for various energy sources (Chassignet et al., 2019). OGCMs are the most powerful tools for predicting the ocean and climate states. As shown in Figure 1, recent studies indicate that the horizontal resolution of OGCMs used for ocean research has been improved from 5 degrees (approximately 500 km) (Bryan et al., 1967; Bryan, 1969) to 1/48 degrees (approximately 2 km) (Rocha et al., 2016; Viglione et al., 2018; Dong et al., 2020; Qiu et al., 2018, 2020). However, the small-scale processes in the ocean (Chassignet et al., 2019; Lellouche et al., 2018), which are critical for further reducing the ocean simulation and prediction biases, still cannot be resolved within a 2-km resolution. Therefore, improving the spatial resolution is one of the most important directions of OGCM development.

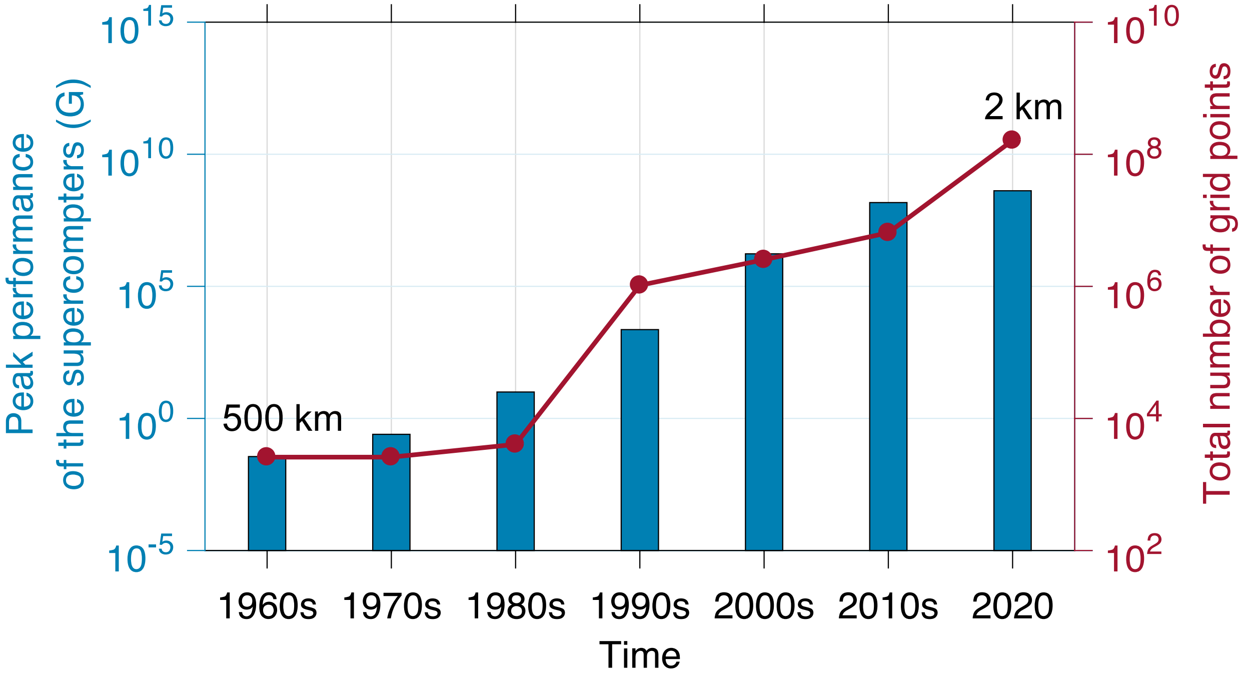

**Figure 1.** Peak performance of the supercomputers (blue bar) and the total number of OGCM grid points (red line) over the past 60 years.

The increase in OGCMs resolution results in an exponential increase in the demand for computing capabilities. With the doubled horizontal resolution, the amount of calculation increases about ten times accordingly. However, the enhanced performance of supercomputers makes it feasible to simulate oceans at higher resolutions. Compared to homogeneous systems, which cannot afford the high-power cost of the transition from PetaFlops supercomputers to ExaFlops supercomputers, heterogeneous many-core systems reduce the power loss from the perspective of system design, thus, becoming mainstream.

In the last few years, heterogeneous architectures (*i.e*, CPU+GPU (Vazhkudai et al., 2018), CPU+FPGA (Putnam et al., 2014), CPU+MIC (Liao et al., 2014) and MPE+CPE (Fu et al., 2016)) have been widely applied to speed up model computation (Table 1). GPUs are increasingly important in high-performance computing due to their high compute density. Using GPUs to accelerate computing speed shows great potential for the ocean and climate modeling. For example, based on mpiPOM, Xu et al. developed a POM (Princeton Ocean Model) GPU solution, which reduced power consumption by a factor of 6.8 times, and achieved the equivalent performance of 408 Intel Westmere cores by using four K20 GPUs (Xu et al., 2015). LICOM (LASG/IAP Climate system Ocean Model) was successfully ported and optimized to run on Nvidia and AMD GPUs, which

**Table 1.** Research on models based on heterogeneous architectures

| Year | Models | Maximum Resolutions | Platforms | Maximum Scales | Parallel Programming | Results |
|---|---|---|---|---|---|---|
| 2015 | POM.gpu | 1,922×1,442×51 | NVIDIA TESLA K20X | 4 | MPI+CUDA | Equivalent to 408 Westmere cores performance |
| 2019 | LICOM2 | 360×218×30 | NVIDIA Tesla K80 | 4 | MPI+OpenACC | 6.6× the 4 Intel Xeon CPU E5-2690 v2 GPUs. |
| 2020 | LICOM3 | 7,200×3,920×55 | AMD GFX906 GPU | 26,200 | MPI+HIP | 2.72 SYPD |
| 2020 | POP2 | 3,600×2,400×65 | TaihuLight | 1,189,500 cores | MPI+Athread | 3.8× speedup |
| **2022** | **swNEMO_v4.0** | **82,500×55,000×128** | **Next Generation Sunway** | **27,988,480 cores** | **MPI+Athread** | **∼8× speedup, 99.29% parallel efficiency** |

showed great potential compared with CPUs (Jiang et al., 2019; Wang et al., 2020). Compared with GPUs, the results of ocean models published with FPGA are relatively few, and most are in the program porting stage.

On the platform of Sunway supercomputers, Zhang et al. (2020) and Zeng et al. (2020) studied the porting of the ocean component POP2 (Parallel Ocean Program version 2), and successfully scaled it to 1 million and 4 million cores, respectively, speeding up computation by a factor of 3 to 4. To use heterogeneous architectures, parallel programming of MPI + OpenMP/CUDA/OpenACC/Athread was developed to improve the parallel efficiency of the model using finer-grained parallelism (Afzal et al., 2017). The coordination between the master and slave cores (involving master-slave cores, slave-slave cores, memory bandwidth, and register communication) is the key to parallel efficiency.

Due to the limitation of memory bandwidth, large-scale parallelism usually cannot maintain high-efficiency (Lellouche et al., 2018; Ruston, 2019; Chassignet et al., 2019; Lellouche et al., 2018). The classic parallelization method is the "longitude-latitude" 2D decomposition based on the MPI of OGCMs. To further improve parallelism, there are many efforts in parallel decomposition schemes and algorithm improvements of physical processes. To achieve better load balance, several decomposition schemes based on curve filling have been introduced in OGCMs (POP at el. (Smith et al., 2010) and NEMO (Nucleus for European Modelling of the Ocean) (Madec et al., 2016)). The parallel scale can reach $O(100km)$ in practice with a resolution of $O(10km)$ (Hu et al., 2013; Yang et al., 2021). However, due to the communication barrier, the large-scale parallel efficiency is still below 50%. Therefore, we need to explore new schemes that can further improve the scalability to the next level. At the current stage, developing a large-scale parallel algorithm for the OGCMs is still in two-dimensional parallelism along with the longitudinal and latitudinal horizontal directions. Therefore, integrating the vertical direction in the three-dimensional parallelism scheme is still challenging.

As most emerging Exascale systems provide support for mixed-precision arithmetic, a mixed-precision computing scheme becomes an important step to reduce the computational and memory pressure further, as well as to improve the computing performance. However, due to the weak support of mixed precision from computer systems and the difficulty of balancing computing precision and simulation accuracy, many efforts are still at an early stage. Dawson *et al.* used a reduced-precision emulator (RPE) to study the shallow water equation (SWE) model using mixed precision and found that the error caused by iterations with low precision can be solved by mixed-precision (Dawson et al., 2017). Then, based on the RPE, Prims *et al.* investigated the application of a mixed scheme of double-precision (DP) and single-precision (SP) on NEMO and verified the feasibility of half-precision (HP) in the regional ocean model of ROMS (Regional Ocean Modeling System) (Prims et al., 2019). Previous studies have shown that mixed precision can improve computational efficiency. However, RPE can only verify the feasibility of mixed-precision in theoretical models. The new generation Sunway supercomputer, supporting HP, can lay a solid foundation for applying mixed-precision OGCM.

The architecture of the new generation of Sunway processors (SW26010 Pro) adopts a more advanced DDR4 compared with the original SW26010. It not only expands the capacity but also greatly improves the DMA bandwidth of the processor. The upgrade of the on-chip communication mechanism of many-core arrays makes the interconnection between CPEs more convenient and builds a more efficient global network on the entire supercomputer. The upgrade of these key technologies provides a solid foundation for the parallelization of OGCMs on the new generation of Sunway supercomputers. In this work, we design and implement a four-level parallel algorithm on three-dimensional space based on hardware-software codesign. We also resolve the problem of memory bandwidth through fine-grained data reuse technology, thus paving the way for the ultrahigh scalability of NEMO. Furthermore, based on Sunway's heterogeneous many-core architecture, a composite block algorithm and a dynamic scheduling algorithm based on LDCache are proposed to fully exploit the performance of many-core acceleration. Finally, half-precision is introduced to further release the memory pressure of NEMO under simulation with ultrahigh resolution. We develop a highly scalable swNEMO_v4.0 based on the Nucleus for European Modelling of the Ocean version 4 (NEMO4) with the GYRE-PISCES benchmark (Madec et al., 2016), which is the benchmark abbreviation of the Gyre Pelagic Interactions Scheme for Carbon and Ecosystem Studies, where PISCES is short for Pelagic Interactions Scheme for Carbon and Ecosystem Studies (Oliver et al., 2015). The resolution is equivalent to the horizontal resolution of 500 m on a global scale.

The following section briefly introduces the new generation of Sunway heterogeneous many-core supercomputing platforms. In Section 3, we briefly review the basics of NEMO4 and describe our optimization methods in detail. The performance results are discussed in Section 4, and Section 5 concludes this paper with discussions.

## 2 The New Generation Heterogeneous Many-core Supercomputing Platform and NEMO4

Succeeded by the architecture of Sunway TaihuLight, the new Sunway generation, as shown in Figure 2, which is driven by SW26010 Pro, consists of 6 core groups, with one management processing element (MPE) and one $8 \times 8$ computing processing element (CPE) cluster in each. The core groups are connected within the loop network. Data can be transferred between CPEs via remote memory access (RMA), which significantly increases the efficiency of CPE co-working.

CPE is also based on the SW64 instruction set, with the 512-bit single-instruction multiple-data (SIMD) vector, where double-precision, single-precision, half-precision, and integer are all supported. Furthermore, double and single precision share the same computation speed, whereas half-precision performs twice faster. There are also a separate instruction cache and a scratchpad memory (SPM) in each CPE. The SPM allocates local data memory (LDM) for users, part of which can be set as a local data cache, automatically administrated by hardware. The data is transmitted between LDM and main memory via either direct memory access (DMA) or the general load and save instructions.

**Table 2.** Basic information about SW26010 Pro

| CPU | SW26010 Pro |
|---|---|
| Number of CGs | 6 |
| Number of processors of one CG | 65 (1 MPE + 64 CPE) |
| Programming language | C/C++, Fortran, Python |
| Parallel Programming Environment | MPI, Athread/OpenACC |
| Memory Size | 96GB |
| Instruction Set | sw64 |
| L1 Instruction Cache | 32KB |
| L1 Data Cache | 32KB |
| L2 Data Cache of MPE | 512KB |
| SIMD register of MPE | 256-bit |
| SIMD register of CPE | 512-bit |
| Precision Support | Double-precision |
| | Single-precision |
| | Half-precision |
| Scratch Pad Memory | 256KB |

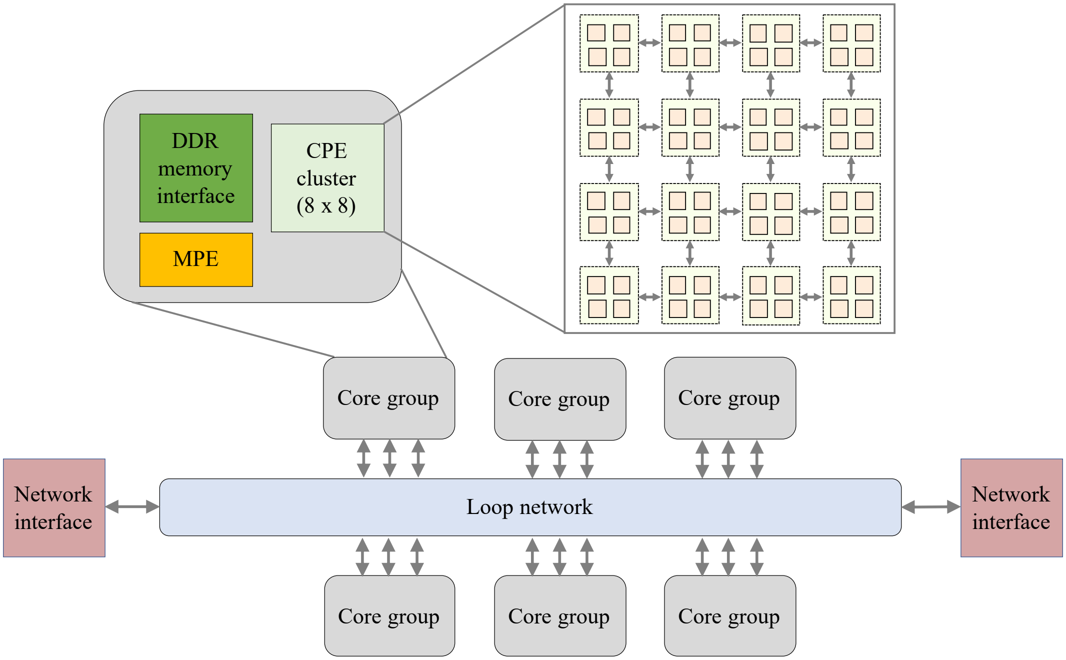

**Figure 2.** Architecture of the new generation Sunway supercomputer.

## 3 Porting and Optimizing NEMO4

NEMO is a state-of-the-art modeling framework for research activities and forecasting services in the ocean and climate sciences developed in a sustainable way by a European consortium since 2008. It has been widely used in marine science, climate change studies, ocean forecasting systems and climate models. For ocean forecasting, NEMO has been applied by many operational forecasting systems, such as the Mercator Ocean monitoring and forecasting systems (Lellouche et al., 2018), as well as the ocean forecasting system in the National Marine Environmental Forecasting Center of China with a 1/12 degree resolution (Wan, 2020). For the climate simulation and projections, approximately 1/3 of the climate models in the latest phase of Coupled Model Intercomparison Project Phase 6 (CMIP6) (Eyring et al., 2016) use the ocean component models of NEMO. The breakthroughs made in this study are based on NEMO, and thus, this work provides useful methods and ideas that can directly contribute to the NEMO community for optimizing their models and improving their simulation speeds.

Fully considering the characteristics of the three-dimensional spatial computation of OGCMs and the heterogeneous many-core architecture of the new generation Sunway supercomputer, we develop a highly scalable NEMO4 named swNEMO_v4.0, with the following three major contributions based on the concepts of hardware-software co-design:

- A highly efficient four-level parallelization framework is proposed for OGCMs to release a new level of parallelism along the column dimension that was originally not parallel-friendly due to the computational dependency;
- A raised many-core optimization method that uses an effective dynamic cache scheduling strategy to utilize the temporal and spatial locality of data effectively;
- A multi-level mixed-precision optimization method that uses half-, single-, and double-precision is explored, which can effectively improve the computation performance while maintaining the same level of accuracy;

We then elaborate on the above algorithms in the following.

### 3.1 An Adaptive Four-level Parallelization Framework

To utilize the many heterogeneous cores of the new generation Sunway supercomputer, the load of the expanded C-grid computation should be assigned in a balanced way. Considering the characteristics of the three-dimensional spatial computation of NEMO and the heterogeneous many-core architecture of a new generation Sunway supercomputer, we propose an adaptive four-level parallel framework that can realize computation load dispatching among different levels. Figure 3 demonstrates our adaptive four-level parallel framework.

### 3.1.1 Level 1: Load-balanced "Longitude-Latitude" Decomposition among MPEs

With land grids eliminated in NEMO, keeping the horizontal data in a traditional way benefits the load balance among processes. The variables are stored as 3-D arrays, and the $x$, $y$ and $z$ axes represent the longitude, latitude and depth, respectively. Since the scales of the $x$-axis and $y$-axis are considerably larger than that of the $z$-axis, we decompose the data along the $x$-axis

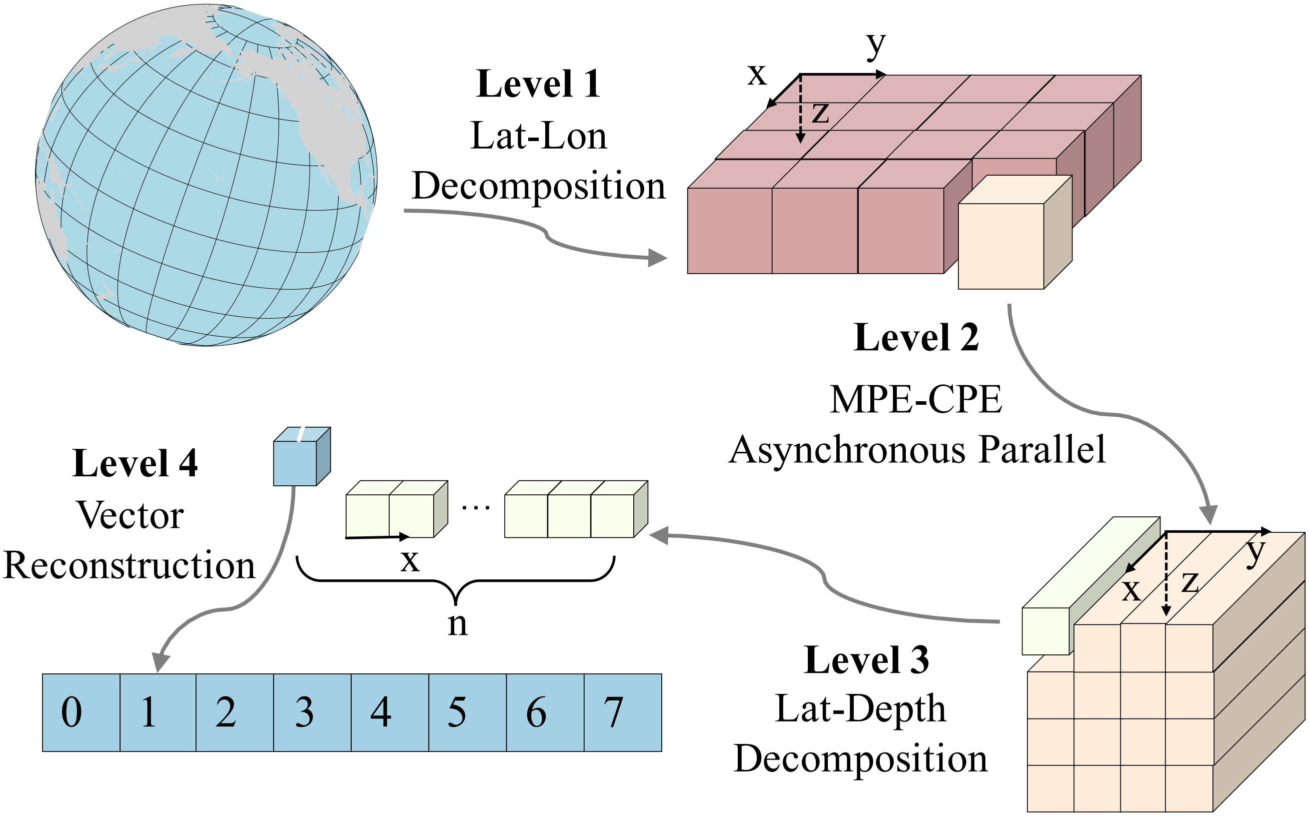

**Figure 3.** The adaptive four-level parallel framework, where $x$-, $y$-, and $z$-axes indicate latitude, longitude and depth, respectively. Level 1 is the load-balanced "longitude-latitude" decomposition among MPEs, level 2 is the asynchronous parallel between a MPE and a CPE cluster, level 3 is the "Latitude-depth" decomposition in a CPE clusters and level 4 is the vector reconstruction in a CPE.

and $y$-axis and dispatch each data block to different MPI processes (first level of the four-level parallelization framework). Our major strategy is to guarantee that (a) the grid sizes of the subdomain in each process are similar, achieving a good load balance; and (b) the $x$-axis and $y$-axis dimension sizes are close to each other. The $x$-axis and $y$-axis of each process are the closest in all options, thus minimizing the halo areas that require communication among processes.

### 3.1.2 Level 2: Asynchronous Parallelization between a MPE and a CPE Cluster

Gu et al. (2022) proposed a multi-level optimization method to use the heterogeneous architecture. In the first level of the method, they designed pre-communication, communication, and post-communication in the code based on MPE-CPE asynchronous parallelism architecture. As the default barotropic solver in NEMO4 is an explicit method with a small time step, which is more accurate and more suitable for high resolution without filter, instead of implicit or split-explicit methods (e.g., PCG), there is no global communication. Therefore, it is only necessary to update the information of the halo region between different processes. Moreover, in the explicit method, most boundary information exchanges are independent of the partition data in the process and can be used for asynchronous parallelization. So boundary data exchange can be parallelized aside from the computation. Considering that MPE is asynchronous with a CPE cluster, we propose an asynchronous parallelization design between the MPE and the CPE cluster using efficient DMA (second level of the four-level parallelization framework). The MPE is in charge of the boundary data exchange, I/O, and a small amount of computation, while the CPE cluster performs the computation of most kernels. Such an asynchronous communication pattern can make full use of the asynchronous parallelism between the MPE and the CPE cluster, thus improving the parallel efficiency.

### 3.1.3 Level 3: "Latitude-depth" Decomposition in the CPE Cluster

Furthermore, we design the third level of parallelism in the CPE cluster by utilizing the fine-grained data-sharing features within the CPE cluster, releasing a new level of parallelism along the column dimension that was originally not parallel-friendly due to the computational dependency. RMA is a unique on-chip communication mechanism on the next-generation Sunway supercomputer, which enables high-speed communication among CPEs. We realize the "latitude-depth" decomposition with LDM and the data exchange with RMA. By utilizing the row and column communication features of RMA to achieve fine-grained data sharing among the CPE threads, we can accomplish an efficient parallelization on the column (*i.e.*, the depth) dimension and release a new level of parallelism for the underlying hardware.

In the data structure of swNEMO, data items along the longitude axis are stored in a continuous way. Following such a memory layout, we divide the data block along the latitude and depth into smaller blocks and copy them into the LDM. More details are shown in Section 3.2.

### 3.1.4 Level 4: Data Layout Reconstruction for Vectorization

To further improve the computational efficiency within each CPE, we design the fourth level of parallelism for vectorization. SW26010 Pro has a 512-bit SIMD instruction set, one of which can compute eight double-precision digits simultaneously. To adapt our computation pattern for the SIMD instruction, we perform a data layout reconstruction to achieve the most suitable vectorization arrangement. Moreover, methods such as instruction rearrangement, branch prediction, and cycle expansion are adopted to improve the execution efficiency of the instruction pipeline.

## 3.2 Performance Optimization for the Many-core Architecture

### 3.2.1 A Composite Blocking Algorithm Based on RMA

To further improve the parallel scale and efficiency, we implement the fine-grained "lat-depth" decomposition. There are many stencils and temporal dependency computations existing in NEMO, which results in high demand for DMA bandwidth. A stencil computation is a class of algorithms that updates elements in a multidimensional grid based on neighboring values using a fixed pattern (hereafter called stencil). In a stencil operation, each point in a multidimensional grid is updated with the weighted contributions from a subset of its neighbors in both time and space, thereby representing the coefficients of the partial differential equation (PDE) for that data element. Therefore, we take advantage of the RMA provided by SW26010 Pro to relieve the pressure of the DMA bandwidth. RMA is an on-chip communication mechanism with superior bisection bandwidth within a CPE cluster. RMA enables direct remote LDM access among different CPEs within one CPE cluster. The efficient batch communication mechanism of RMA is highly adaptable for solving typical *x*-pointer problems. For example, in the diffusion process of the *tracer* in NEMO, upstream points are needed for data exchange, which includes horizontal unidirectional grid information exchange (3-pointer) and grid information exchange along with longitude and latitude directions (5-pointer). Figure 4 represents the grid communication process between CPE #0 and CPE #1. Each point represents a multidimensional tensor composed of different variables that are irrelevant to each other. When updating $A(u)$, variables $v$ and $w$ from the surrounding 8 points are required to participate in the calculation. We first send $v$ and $w$ required by adjacent CPEs to the buffer in the corresponding CPEs while updating the local variables whose surrounding points are on the same CPE. After all the needed variables in the halo are transferred into the buffer, the remaining variables in the points located at the edge of the CPE can be updated.

Based on RMA, we design different parallel blocking algorithms for computing kernels with different characteristics to maintain an efficient performance in various application scenarios. In the following , $\alpha_1$, $\beta_1$, and $\beta_2$ are only the coefficient, $f$ is the mathematical formula for the loop segment, $x$ means the $x$-axis (longitude axis) in the coordinate system.

- Temporal dependency in computing along the *z*-axis. We suppose $u_{i,j,k-1}^{iter}$ is an original value of grid $(i, j, k-1)$ in the coordinate system $(x, y, z)$ (where the *x*-axis is the most continuous one and the *z*-axis is the least) and $u_{i,j,k}^{iter}$ is computed as follows:

  $$\begin{cases} u_{i,j,k}^{iter} = f(u_{i,j,k-1}^{iter}) \\ \qquad f = \alpha_1 \times u_{i,j,k-1}^{iter} + \beta_1 \times (u_{i\pm1,j,k}^{iter} + u_{i,j\pm1,k}^{iter}) \end{cases}$$

  where $0 < i < n_x$, $0 < j < n_y$, and $1 < k < n_k$. In this equation, $\alpha_1$ and $\beta_1$ are coefficients that indicates the linear characteristic of this mapping $f$. To fully utilize the DMA bandwidth, we decompose data along the *y*-axis and send them into different CPEs. Then, continuous data along the *x-axis* are copied into the LDM at a time.

- Temporal dependency in computation along the *y*-axis. In this case, the computation is as follows:

  $$\begin{cases} u_{i,j,k}^{iter} = f(u_{i,j-1,k}^{iter}) \\ \qquad f = \alpha_2 \times u_{i,j-1,k}^{iter} + \beta_2 \times (u_{i\pm1,j,k}^{iter} + u_{i,j,k\pm1}^{iter}) \end{cases}$$

  Here $\alpha_2$ and $\beta_2$ are also coefficients that indicates the linear characteristic of this mapping $f$. Restricted by the size of the LDM, we decompose data along the *z*-axis and then decompose data along the *y*-axis into the proper size $m$ and dispatch each data block into different CPEs as follows:

  $$m = \frac{\text{size(LDM)}}{\text{size}(x)}$$

  Therefore, blocks on the same *z*-layer are dispatched to the same CPE one-by-one.

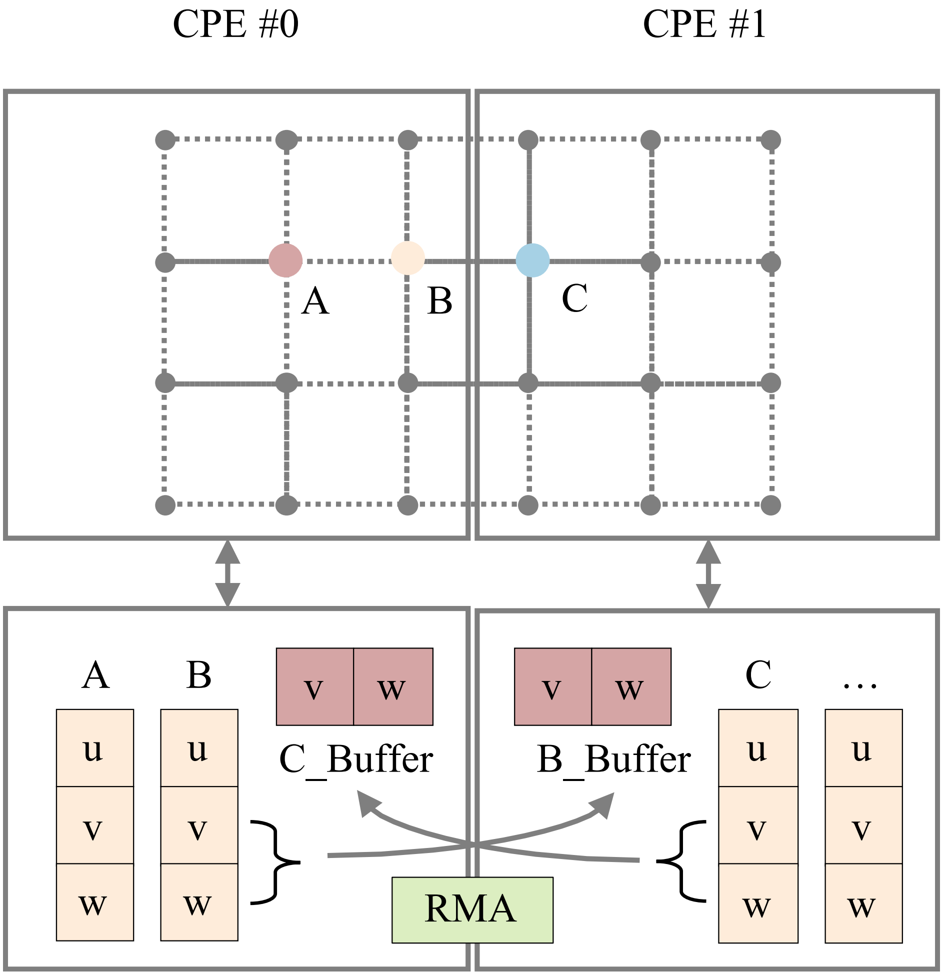

**Figure 4.** Tensor distribution in different CPEs (upper) and data transportation between CPEs (lower).

- Nontemporal dependency in computation. In this case, we decompose the data along with the *z*-axis and *y*-axis. Due to the elaborate design of size ($x$), each block can be copied into LDM at once to reduce redundant halo transfer and improve the asynchronous parallel of stencil calculations.

To increase the utility of the RMA bandwidth, we pack the data before sending it for better utilization of the bandwidth in an aggregated manner. While sending, CPEs compute, in this way, we can realize the overlap between RMA communication and computation and thus, make efficient utilization of the bandwidth by reducing redundant DMA references. By utilizing RMA and the composite blocking strategy, for nontemporal dependency and temporal-dependency computation, we finally achieve over 90% effective utilization of the RMA bandwidth and over 80% utilization of the DDR4 memory bandwidth. With such an efficient memory scheme, the average speedup of most computing kernels (comparing the performance of a CPE cluster to an MPE) can be up to 40 $times$.

### 3.2.2 A Dynamic LDCache Scheduling Algorithm

To access discrete data items in swNEMO, the traditional global load/global store (gld/gst) method becomes a major performance hindrance. LDCache, provided by SW26010 Pro, stores data with a cache_line of 256 bytes in CPE. In one CPE, LDM and LDCache share an SPM with a total capacity of 256 KB, and the size of LDM and LDCache can be manually adjusted. Since the amount of data required for a round of computation in different kernels in NEMO is varied, if the cache space is fixed, the utilization of the LDM becomes less efficient. Therefore, we design a dynamic LDCache scheduling algorithm that can realize efficient and fine-grained memory access. One feature of this algorithm is to dynamically adjust the size of LDCache to achieve a balance between LDM and LDcache. Furthermore, the algorithm has a time-division update technique since LDCache cannot guarantee data consistency with memory. We regularly update the stored data and eliminate the outdated data in the LDM simultaneously. As shown in Figure 5, the data that needs to be refreshed is packed on the CPE and sent to the designated buffer of the MPE. The MPE then uses the MPE-CPE message mechanism to find the buffers that need to be updated in a round-robin way and update them, while the CPE eliminates the corresponding data in the cache. By applying the dynamic LDCache scheduling algorithm with both an adjustable cache and a manual time-division update technique, we can improve the memory bandwidth utilization rate to approximately 88.7% for DDR4, with an 88 times speedup (comparing the performance of a CPE cluster to an MPE) for most computing kernels, which is a substantial improvement compared with the 5.1 times speedup when using the traditional gld/gst method.

## 3.3 Mixed-precision Optimization

Because the NEMO, as one of the OGCMs, is memory intensive, the memory bandwidth limits the computational efficiency of NEMO to a large extent. A reduced-precision method can be a promising solution. However, using low-precision data is a double-edged sword. On the one hand, it can effectively resolve the performance obstacles ; on the other hand, the reduced precision brings errors and uncertainties. According to (Dawson et al., 2017), 95% of NEMO variables support the single-precision floating-point format (SP). Therefore, we specifically reconstruct the data structure and introduce a new three-level mixed precision scheme in NEMO. To achieve a higher performance (Figure 6), we reconstruct two "half-precision + single-precision" (HP+SP) computing kernels *tracer_fct* and *tracer_iso* of NEMO on CPE, which account for 50% of the hotspot runtime in total. As HP is only supported on the CPEs of SW26010 Pro, we store HP data in the format of char or short types of memory on the MPE and use HP format to read them to LDM by DMA to ensure the correctness on the CPE.

By analyzing the calculation characteristics of NEMO, we find that the HP format subtracts operations between adjacent grids, which makes the final results diverge due to the round error of low precision. Therefore, a high-precision format must be used when calculating adjacent grids. Based on the above analysis in other calculations, we adopt the SP floating-point format for calculations between adjacent grids and the HP floating-point format with BF16 (consisting of 1 sign bit, 8 exponent bits, and 7 mantissa bits). After optimization, we achieve a speedup close to 2× compared with NEMO using DP, while

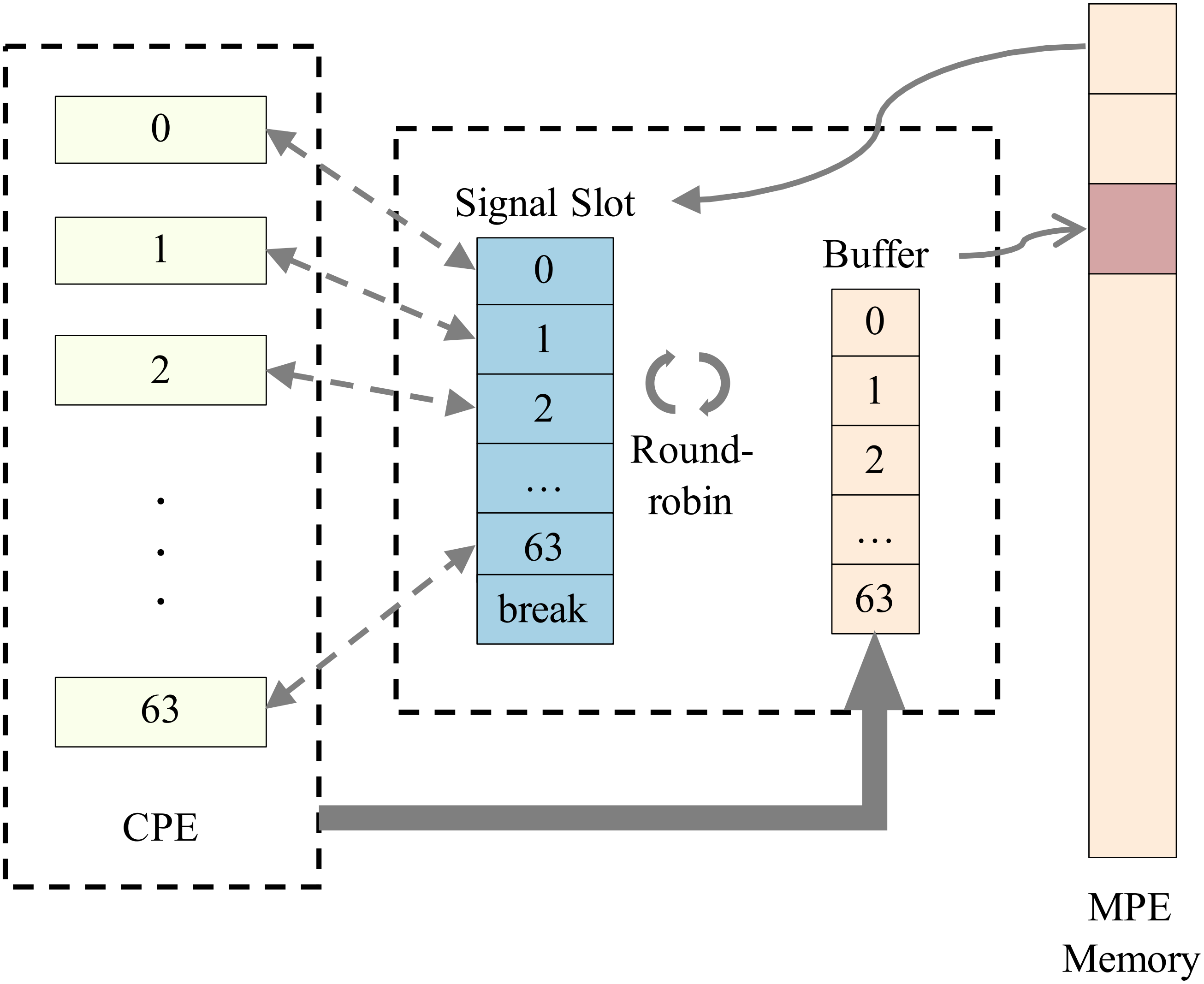

**Figure 5.** Time-division refresh technology for LDM.

the maximum biases of temperature, salinity and velocity are within 0.05% (figures not shown). Therefore, utilizing mixed-
225 precision can effectively increase the computational intensity and improve the scalability while maintaining the simulated accuracy of NEMO.

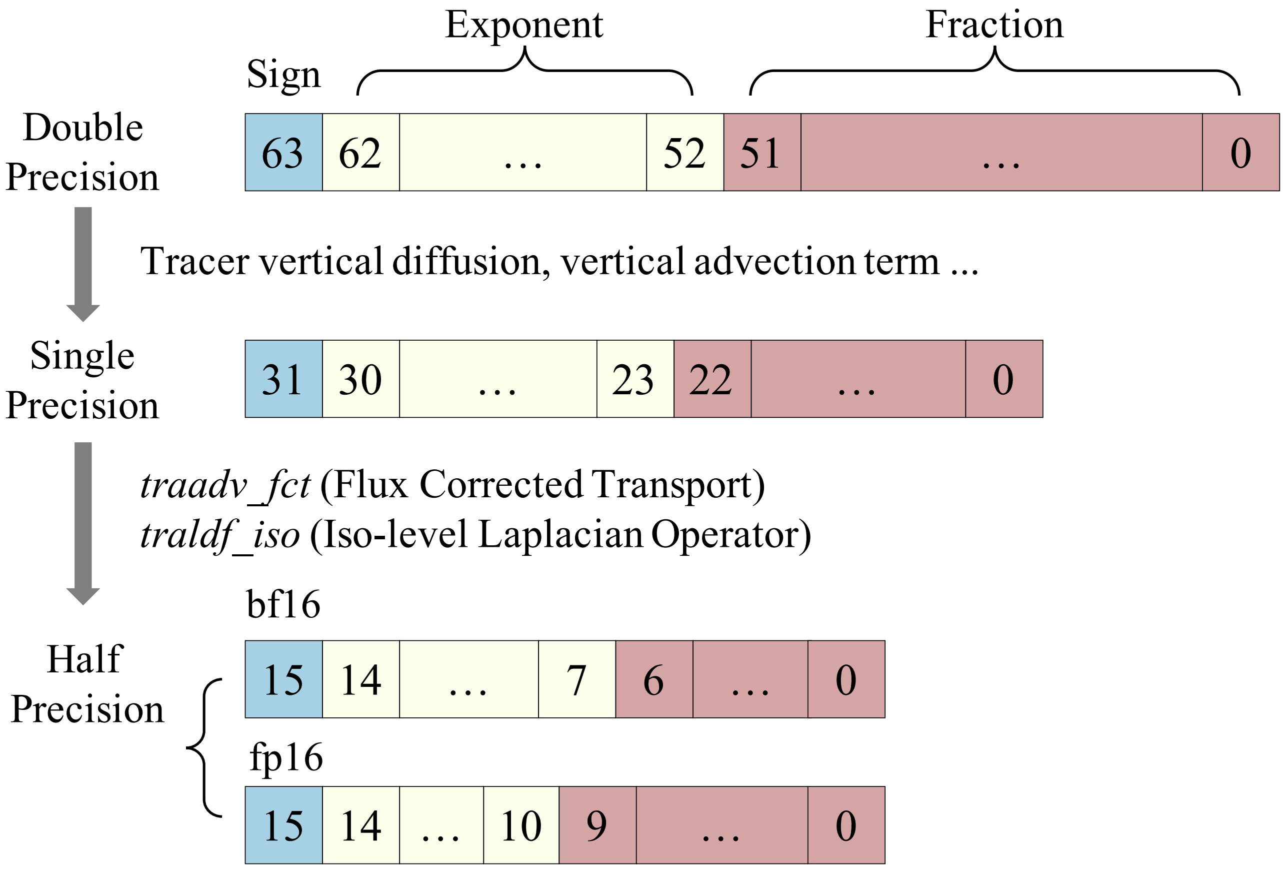
Exponent
Fraction
Sign
Double Precision
63
62
…
52
51
…
0
Tracer vertical diffusion, vertical advection term ...
Single Precision
31
30
…
23
22
…
0
traadv_fct (Flux Corrected Transport)
traldf_iso (Iso-level Laplacian Operator)
bf16
Half Precision
15
14
…
7
6
…
0
fp16
15
14
…
10
9
…
0

**Figure 6.** Data reconstruction for three-level mixed precision.

In order to validate the results of optimized swNEMO_v4.0, we carried on the perturbation experiments for the temperature results and analyzed the RMSZ (Root-Mean-Square Z-score) proposed by Baker et al. (2016). The experimental sample set can be represented as an ensemble $E = \{X_1, X_2, X_3, ..., X_m\}, m = 1, 2, ..101$, where each sample $X_m(j)$ represents the global
230 given points in all grids for each month $j = 1, 2, ..120$. $\mu(j)$ and $\delta(j)$ represent the mean and standard deviations of the initial temperature fields for all grid points at each time point.

$$RMSZ(X) = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(\frac{X(j)-\mu(j)}{\delta(j)})^2}$$

Since the experiment in this work is the ocean basin benchmark, it is not feasible to directly compare with observed values. Hence random disturbance conditions for the initial temperature are configured within the NEMO program. At the same time, we also selected 101 sets of data as experimental simulation results to validate the results of optimized NEMO_v4.0, of which 100 sets are temperature results with perturbation conditions, and the remaining one is the mixed-precision temperature field experimental results of swNEMO_v4.0.

At first, the perturbation coefficient is selected as $O(10^{-14})$). However, we found that the RMSZ obtained by mixed-precision simulation was completely out of the shadow area formed by 100 simulations with double-precision (Figure 7a). Therefore, we increased the perturbation coefficient, and the mixed-precision Z-score gradually approached the shadow area. We found that the disturbance had a greater influence in the first few years, then gradually decreased and tended to be stable after several years (Figure 7a and b). As shown in Figure 7b, when the perturbation coefficient is $O(10^{-11})$, the Z-score of mixed-precision simulation falls partially in the region formed by double-precision simulations. When the perturbation coefficient equals $O(10^{-10})$, the Z-score completely falls in the shadow area (Figure 7c), which indicates the effects of mixed-precision are similar to these of the perturbation coefficient as $O(10^{-10})$. It shows that the mixed-precision affects the results, but the effects are small (the perturbation coefficient is around $O(10^{-10})$) and can be accepted (Figure 7c).

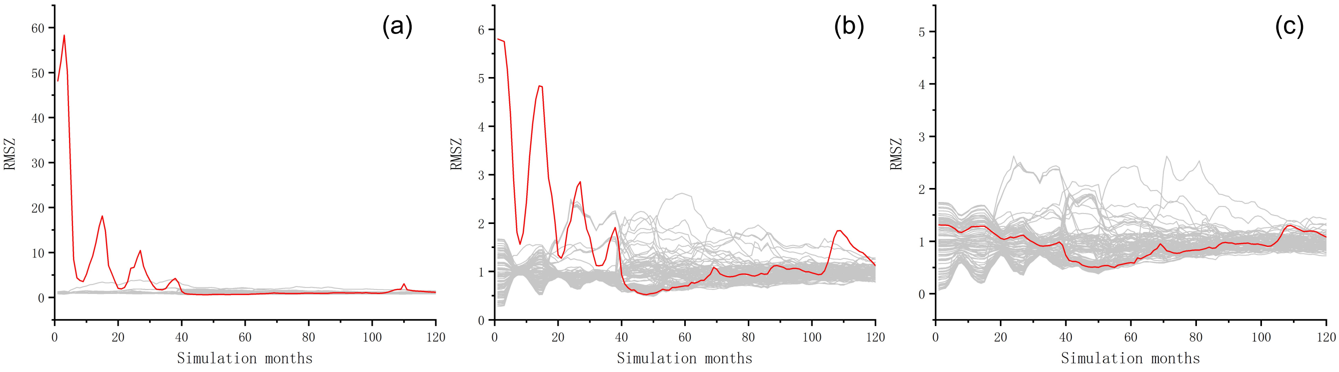

**Figure 7.** RMSZ biases of the sea surface temperature in ensemble experiments (grey lines) with perturbation coefficient as $(a)O(10^{-14})$, $(b)O(10^{-11})$, $(c)O(10^{-10})$ and in the mixed-precision experiment (red line)

## 4 Performance Results

We choose the benchmark named GYRE-PISCES to test the swNEMO_v4.0 performance. The domain geometry is a closed rectangular basin on the $\beta$-plane centered at $\sim$ 30°N and rotated by $45°$, 3,180 km long, 2,120 km wide, and 4 km deep. The circulation is forced by analytical profiles of wind and buoyancy fluxes. This benchmark represents an idealized North Atlantic or North Pacific basin (Madec et al., 2016). In addition, the East-West periodic conditions and the North Pole folding of the global ocean with a tripolar grid have a large impact on performance. Therefore, we activate the BENCH option to include these periodicity conditions and reproduce the communication pattern of the global ocean with a tripolar grid between two

North Pole subdomains. It is equivalent to a global ocean with a tripolar grid with the same number of grid points from the perspective of computational cost and computational characteristics, although the physical meaning is limited. In the following content, the resolution of the benchmark is equivalent to that of the global ocean.

In this part, we have added a more specific description. We designed three groups of experiments with different resolutions of 2 km, 1 km, and 500 m for strong expansibility. Besides, we also carried out weak scalability experiments. The corresponding relationship among resolution , Computing grid points and data scale of weak scalability can also be seen from Table 3 information. The speedup equals to the clock time in different scales divided by the baseline record of the minimum scale with 2,129,920 cores. For the weak scalability analysis, we design the experiment with 8 resolutions (Table 3). All experiments are run for 1 model day without I/O. We adopt the following methods to perform real-time statistics and floating points to ensure measurement accuracy.

- For time statistics, we use two methods to proofread:
  - We use the MPI_Wtime() function provided by MPI to obtain the wall-clock time;
  - We use the assembly instructions to count the cycle time.
- Similarly, two methods are used in the statistics of floating-point operations:
  - We use the loader to count the floating-point operation of the program when submitting the job;
  - We use performance interface functions to perform the program instrumentation to count the operations.

**Table 3.** Eight different scales used in weak scalability, the conversion between resolution in *degree* and resolution in *km*, and the total number of computed grid points used in weak scalability.

| Scales in weak scalability (cores) | Resolution ($^\circ$) | Resolution (km) | Computed grid points (horizontal) |
|---:|---:|---:|---:|
| 2,129,920 | 1/12 | 9.0 | 13,515,004 |
| 4,259,840 | 1/16 | 7.0 | 24,020,004 |
| 8,519,680 | 1/24 | 4.5 | 54,030,004 |
| 12,779,520 | 1/32 | 3.5 | 96,040,004 |
| 17,039,360 | 1/44 | 2.5 | 181,555,004 |
| 21,299,200 | 1/64 | 2.0 | 384,080,004 |
| 25,559,040 | 1/96 | 1.2 | 864,120,004 |
| 27,988,480 | 1/116 | 1.0 | 1,261,645,004 |

### 4.1 Parallel-Working Performance on a Many-Core Architecture

The cycle time and speedup ratio of the hotspots are shown in Figure 8, where different parallel methods are applied for comparison with the original method. While the CPEs parallel method (level 3 of four-level parallel framework), which introduces all 64 slave kernels to help with acceleration, is 12 times faster than the original method, we find the master-slave asynchronous parallel-working mode (MPE-CPEs multilevel parallel, level 2 of four-level parallel framework) shows quite satisfying results with a 65 times speedup. The four-level parallel framework fits well in Sunway architecture, and the master-slave asynchronous parallel-working strategy combines data parallelism with task parallelism. By making great use of the existing Sunway architecture, this parallel-working strategy overlaps the computation part with the communication and I/O part, thus considerably improving hotspot efficiency.

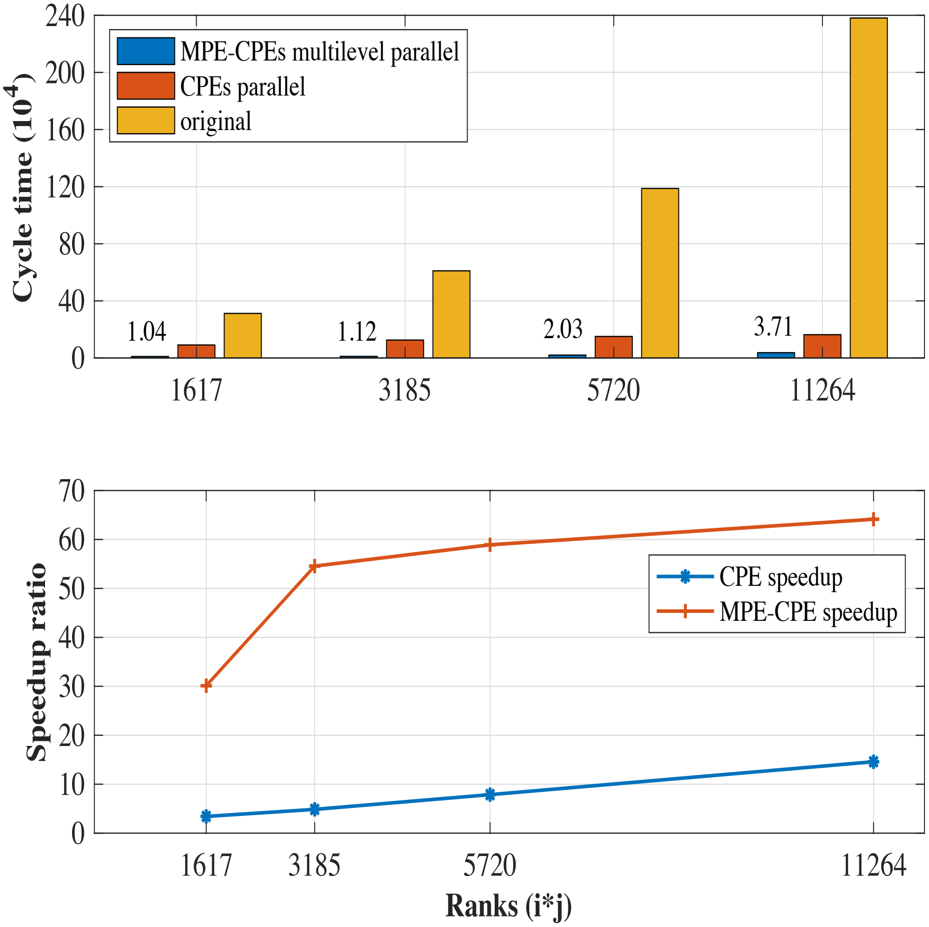

**Figure 8.** Performance of MPE-CPE asynchronous parallelization

The Sunway architecture suggests storing data into slave kernels before further computation. Therefore, the performance of the many-core architecture is related to the actual amount of memory access, the bandwidth of DMA and the FLOPS performance of slave kernels. We assume the total amount of time of the hotspot part in slave kernels is represented by

$T = T_c^{'} + T_l^{'}$, where $T_c^{'}$ is the time cost of the actual floating-point operation in this part, and $T_l^{'}$ is the time cost of direct memory access. To speed up the many-core architecture, we can seek methods for both DMA optimization and floating-point operation optimization.

For stencil computation, due to the low ratio of computation to memory access, we have $T_c^{'} < T_l^{'}$. We can minimize $T_c^{'}$ using computation-communication overlap, yet $T_l^{'}$ is related to the actual amount of memory access and the bandwidth of DMA $T_l^{'} \geq \frac{M}{BW^{'}_{DMA}}$, where $BW_{DMA}$ is the theoretical value of DMA bandwidth and $M$ is the total valid amount of memory access. In the real NEMO4 case, we have $T_c^{'} << T_l^{'}$. Assuming the following equation holds true, $BW^{'}_{DMA} = \frac{M}{T_c{'} + T_l{'}}$, then the ratio of actual DMA bandwidth to theoretical bandwidth should be

$$\alpha = \frac{\frac{M}{T_c{'} + T_l{'}}}{BW_{DMA}} = \frac{M}{(T_c^{'} + T_l^{'})BW_{DMA}}$$

The theoretical time cost of DMA when M is fixed is $T = \frac{M}{BW_{DMA}}$. Since $T_l^{'}$ is affected by the frequency and size of DMA, $T < T_l^{'}$ and $T < T_c^{'} + T_l^{'}$ hold. Therefore, we have

$$\alpha = \frac{M}{(T_c^{'} + T_l^{'})BW_{DMA}} = \frac{T}{T_c^{'} + T_l^{'}} < 1$$

when $T_c^{'} \to 0$ or $T_c^{'} \to T$, $\alpha \to 1$. The more the ratio of actual memory bandwidth to theoretical bandwidth approaches 1, the faster the whole architecture works, and thus, the better the performance becomes.

To analyze the performance of the algorithm mentioned in Section 3.2.1, we use five kernels to simulate the test, with 49×65×128 grids on average per kernel. The five kernels are the most time-consuming chunks, which are only a fraction of the physical process. However, they are all specific implementations of Stencil computing. We calculate the average clock cycle of running one simulation, and the results are shown in Figure 9(see the appendix for the detailed codes of all the following kernels 1∼5), where the red bars are the original results and the blue bars are the optimized results using this algorithm. It is clearly shown that the time cost is largely reduced via this method, as with $3^{rd}$ kernel, the clock cycle is reduced from $61 \times 10^{-3}$ seconds to $0.6 \times 10^{-3}$ seconds, which means the result becomes 70.9 times faster. Meanwhile, according to the simulation test, this algorithm raises the ratio of actual DMA bandwidth to theoretical bandwidth up to above $92\%$, as shown in the $3^{rd}$ kernel, and even reaches the theoretical limit $95\%$ as a whole.

### 4.2 Mixed-precision Optimization

Introducing low-precision format data helps reduce memory use. Since the LDM in the slave kernel has limited storage, lowering the data precision could better use the given space and accelerate communication with the same actual DMA bandwidth. In terms of NEMO, due to its memory constraint, DMA data communication contributes to greater than $95\%$ of the total running time on the many-core architecture, and as a result, we can enhance the overall performance by lowering the time cost of data communication. Focusing on *tracer_fct* in NEMO, we simulate a 4-kernel test with a 49x65x128 grid on average per kernel, and the results, which include the optimizations of four-level parallelization framework and mixed-precision approaches, are shown in Figure 10, where the red and orange bars represent the average wall time of SP and SP+HP in one simulation. Since

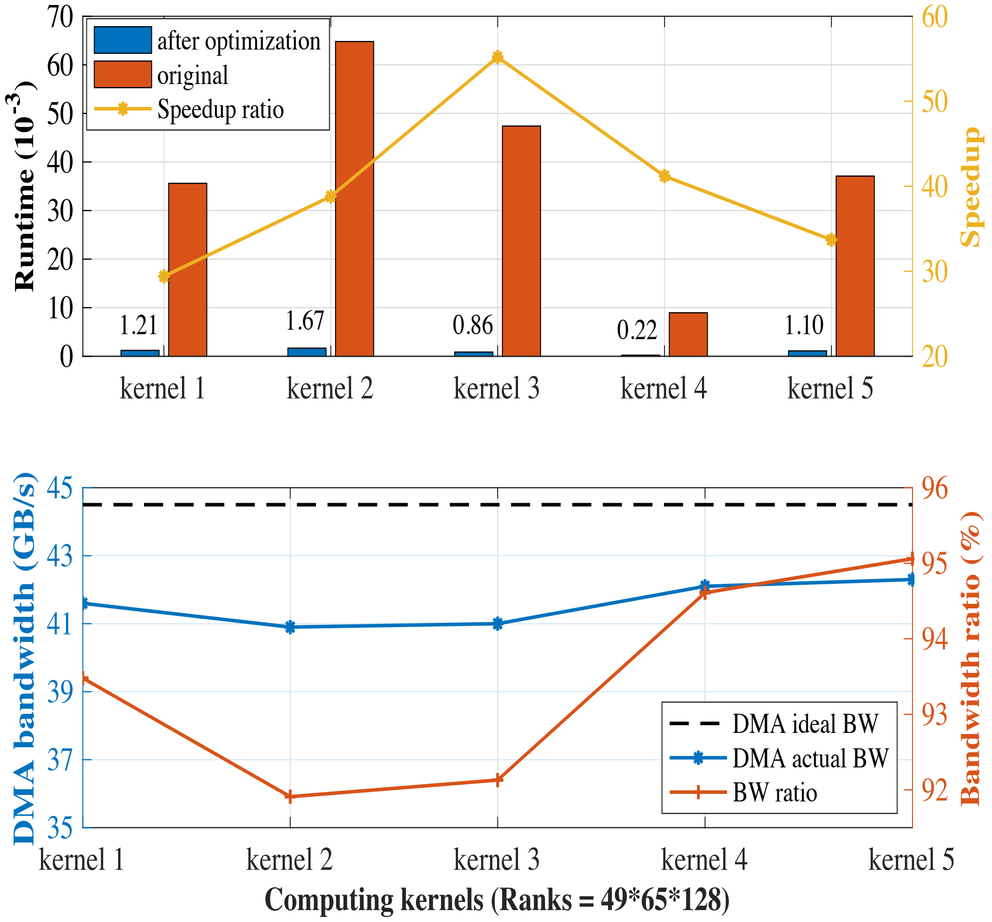

**Figure 9.** The performance of the 5-kernel simulation test and the 5-kernel are shown in the appendix

NEMO is sensitive to precision, we cannot use HP only. We can see that the SP+HP method shortens the total time by half compared to the original DP method. To further explore this topic, we simulate another test comparing the bandwidth usage and the ratio in both SP and SP+HP precision, which is shown in Figure 11. The result shows that both methods surpass the $90\%$ ratio threshold, with the $4^{th}$ kernel performing best with a ratio of $95\%$.

### 4.3 Strong Scaling

Figure 12 shows the results of strong scaling. We conduct experiments with resolutions of 2 km, 1 km, and 500 m, increasing the number of cores from 2,129,920 to 27,988,480. The numbers of grids at resolutions of 2 km, 1 km, and 500 m are

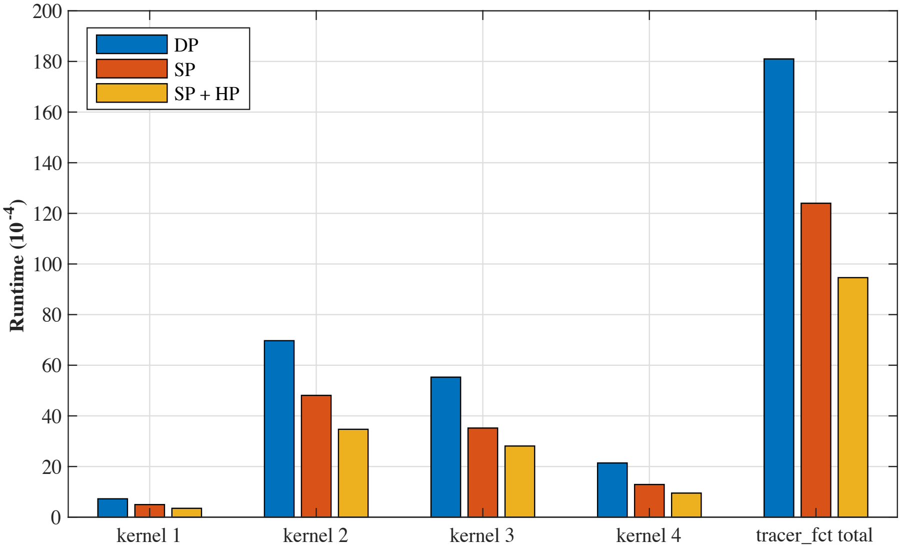

**Figure 10.** Time cost of the tracer_fct process

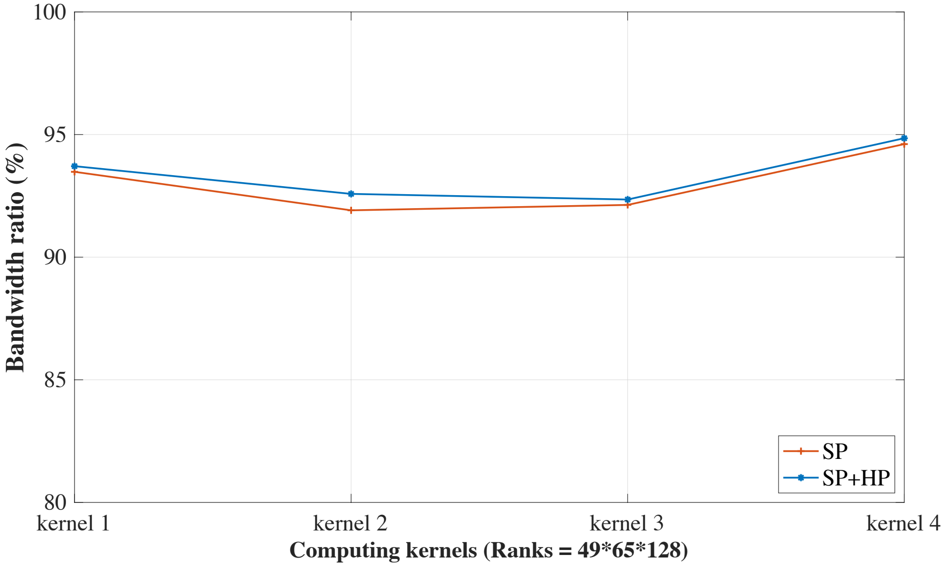

**Figure 11.** Bandwidth ratio of tracer_fct process

24,002×16,002×128, 43,502×29,002×128 and 82,502×55,002×128, respectively. We set 2,129,920 cores as the baseline of strong scaling, and the final parallel efficiencies are 74.18%, 83.40% and 99.29%, respectively.

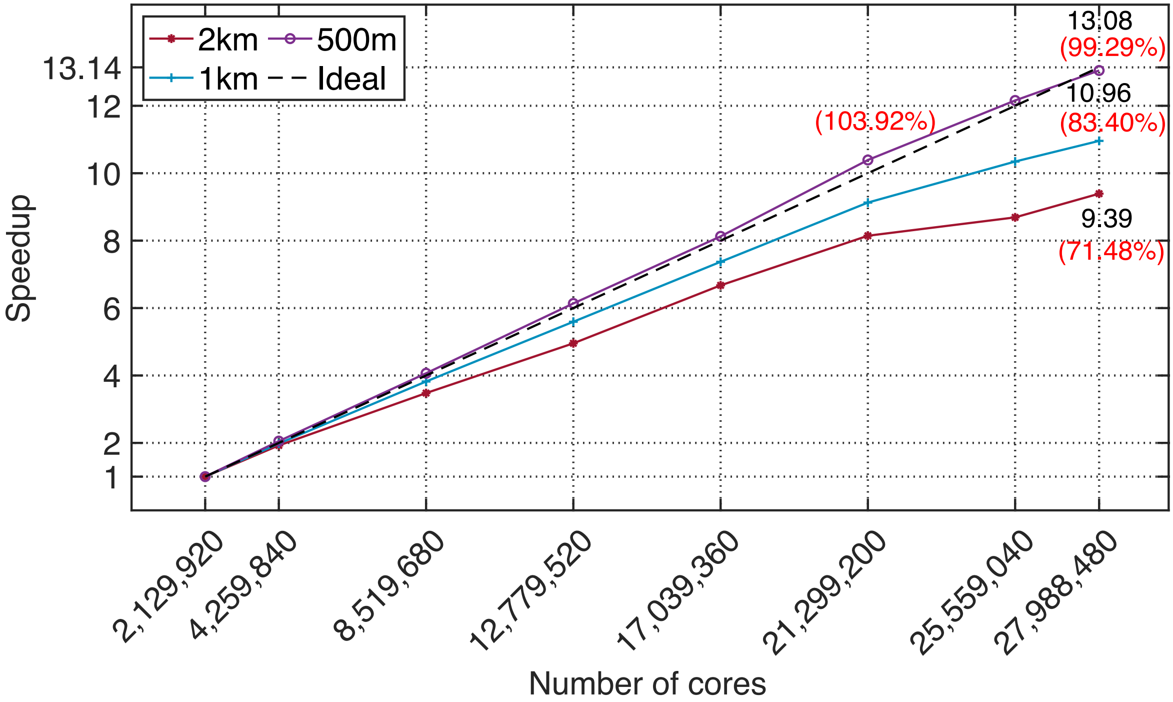

**Figure 12.** The strong scalability results, scaling from 2,129,920 to 27,988,480 cores with 2 km, 1 km, and 500 m resolutions.

When the number of cores is 2,129,920, the average computing task of each process (*i.e.*, CG) is approximately 325×432, including a halo in the horizontal direction and 128 grids in the vertical direction. The mixed-precision optimization proposed in this paper considerably speeds up computations and reduces memory overhead. However, to guarantee the accuracy of the results, there are still some unavoidable double-precision floating-point arithmetics in the key steps in swNEMO_v4.0. During computing, the size of a four-dimensional variable with double-precision has exceeded 1 GB. To maintain a reasonable efficiency of the memory system, we select 2,129,920 cores as the minimum available baseline.

For two-dimensional stencil computations, a grid with similar sizes of the *x*-axis and *y*-axis can effectively reduce the amount of communication and speed up the stencil computation. When the grid sizes of the *x*-axis and the *y*-axis are the same, the amount of communication touches the bottom. According to the four-level parallel architecture, the process-level task assignment does not involve grid partitioning in the vertical direction. For the strong scalability test, with a resolution of 500 m, the size of the horizontal grid is 82,502×55,002, and the size of the vertical grid is 128. When using 21,299,200 cores, the process division in the horizontal direction is 640×512. At this time, each process (*i.e.* CG) computes 128×108 grids, and the grid sizes of the *x*-axis and the *y*-axis of each process are the closest in all options. Therefore, the best parallel efficiency, 104%, is achieved when the number of cores is 21,299,200.

In summary, the strong scalability of the three resolutions has always maintained high performance with ten million cores, and the speedup is still nearly linear at an ultra-large scale. By using 27,988,480 cores, swNEMO_v4.0 achieves up to 99.29% parallel efficiency with a high resolution of 500 m.

### 4.4 Weak Scaling

The choice of resolution of the GYRE needs to follow strict disciplines. Therefore, to ensure that the workload in a single process is fixed, the number of grids of the *x*-axis and *y*-axis is proportional to the number of cores, and the automatic scheme of domain decomposition in swNEMO_v4.0 is replaced with a manual scheme at the same time, which is shown in Figure 13 and Table 3.

As we build roof-line models on NEMO, we find that when the horizontal grid surpasses $49 \times 65$ and the vertical grid surpasses $128$, the computation efficiency of the floating-point for a single core group performs the best. Therefore, our following experiments on weak scaling are conducted on a single core group with $49 \times 65 \times 128$ grid size each. Based on the abovementioned principle, we choose resolutions of 9 km, 7 km, 4.5 km, 3.5 km, 2.5 km, 2.0 km, 1.2 km, and 1.0 km. According to the expansion of the workload of each process (*i.e.*, CG), the number of cores increases from 299,520 to 27,988,480. With a resolution of 1.0 km, the total number of grids is 43,502×29,002×128. As shown in Figure 13, the performance is stable with different resolutions, and the computation efficiency of the floating point is still 1.99‰the resolution of 1 km, which is very close to the baseline. The nearly linear trend indicates that the model has good weak scalability.

### 4.5 Peak Performance

Figure 14 shows the peak performance of swNEMO_v4.0 with a resolution of 1 km. When the total grid size is 43,502×29,002×128, the number of cores increases from 2,129,920 to 27,988,480. It is obvious that the optimal performance is 1.97 PFlops by using 27,988,480 cores, and the performance of the optimized version is 10.25 times faster than that of the original version.

In swNEMO_v4.0, we fully parallelize 70.58% of the hotspots according to the performance profiling tools. The remaining 29.42% of hotspots are mainly serial, which can hardly be parallelized. Therefore, mix-precision optimization is used to optimize the serial part. According to Amdahl's law:

$$\text{Speedup} = \frac{1}{(1-P)+P/N}$$

the theoretical speedup is approximately by a factor of 6.8.

Combining the breakthroughs in an adaptive four-level parallelization design, CPE cluster optimization, and mixed-precision optimization, the performance of our optimized version surpasses the theoretical values after further refactoring the code to reduce conditional judgement, instruction branching, and the complexity of the code.

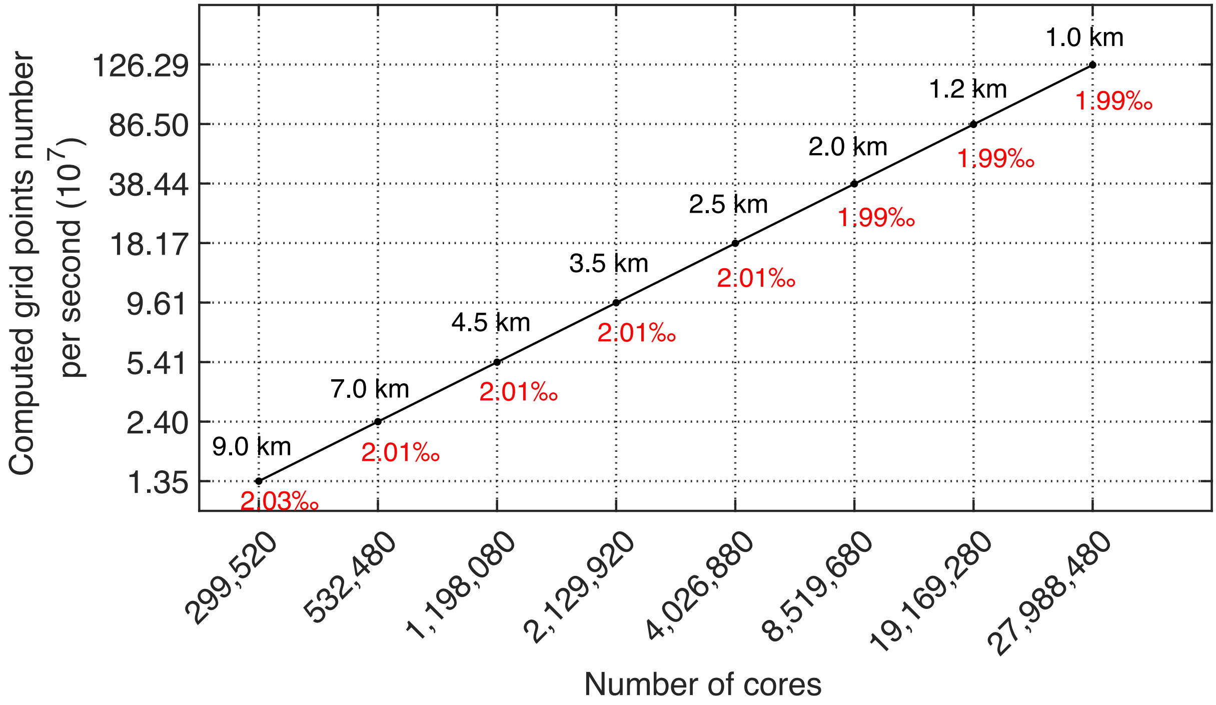

**Figure 13.** The weak scalability results, scaling from 299,520 to 27,988,480 cores with 8 different resolutions.

Additionally, as shown in Figure 15, 1.43 SYPD (Simulated Years Per Day) with a resolution of 1 km using 27,988,480 cores is achieved, which is 7.53 times better compared with 0.19 SYPD of the original version, exceeding the ideal performance upper bound.

## 5 Conclusions and Discussions

This paper presents a successful solution for global ocean simulations with ultrahigh resolution using NEMO on a new generation Sunway supercomputer. Three breakthroughs, including an adaptive four-level parallelization design, many-core optimization, and mixed-precision optimization, are designed and tested. The simulations achieve 71.48%, 83.40%, and 99.29% parallel efficiency with resolutions of 2 km, 1 km, and 500 m by using 27,988,480 cores, respectively.

Current resolutions of ocean models in ocean forecasting systems and climate models cannot well resolve the mesoscale and sub-mesoscale eddies in the real ocean, not to mention the other smaller-scale processes such as internal waves. However, these sub-mesoscales and small-scale processes are not only important to navigation safety in themselves, and but also have notable influences on the large-scale simulations of the global/regional ocean and circulations through interacting with different scales. Improving resolution is one of the best ways to resolve these processes and improve simulations, forecasts, and predictions.

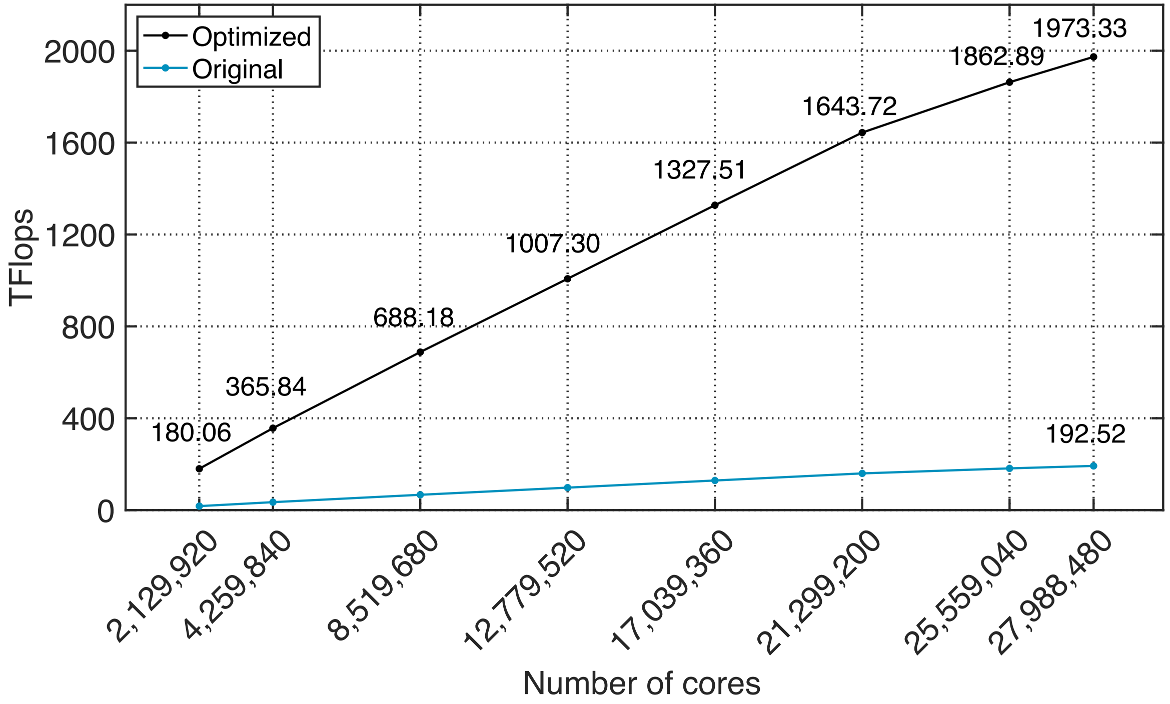

**Figure 14.** The peak performance results for simulations with 1 km resolution. Numbers along the graph line are the peak performances for simulations with different cores. The peak performance with 27,988,480 cores reaches 1.973 PFlops.

The highest resolution used in this study is 500 m, and this resolution can well-resolve the sub-mesoscale eddies and partly resolve the internal waves, which are very important for the safety of offshore structures. Therefore, the breakthroughs made in this study make the direct simulations of these important sub-mesoscales and small-scale processes at the global scale possible. This will substantially improve the forecast and prediction accuracy of the ocean and climate, while strongly supporting the key outcome of a predicted ocean of the UN Ocean Decade.

This study is conducted in the new generation of Sunway supercomputers. The breakthroughs in this paper, such as the four-level parallelization design and the method for efficient data transportation inside CPEs, provide novel ideas for other applications in this series of Sunway supercomputers. Moreover, the proposed new optimization approaches, such as a four-level parallel framework with longitude-latitude-depth decomposition, a multi-level mixed-precision optimization method that uses half-, single-, and double-precision, are the methods of general applicability. We test these optimization approaches in the NEMO, but these can be incorporated into other global/regional ocean general circulation models (e.g., MOM, POP, ROMS, etc.). Moreover, the optimizations on the stencil computation can be applied to any model with stencil computations.

High-resolution ocean simulations are crucial for navigation safety, weather forecasting, and global climate change prediction. We believe that these approaches are also suitable for other OGCMs and supercomputers. In this paper, three innovative

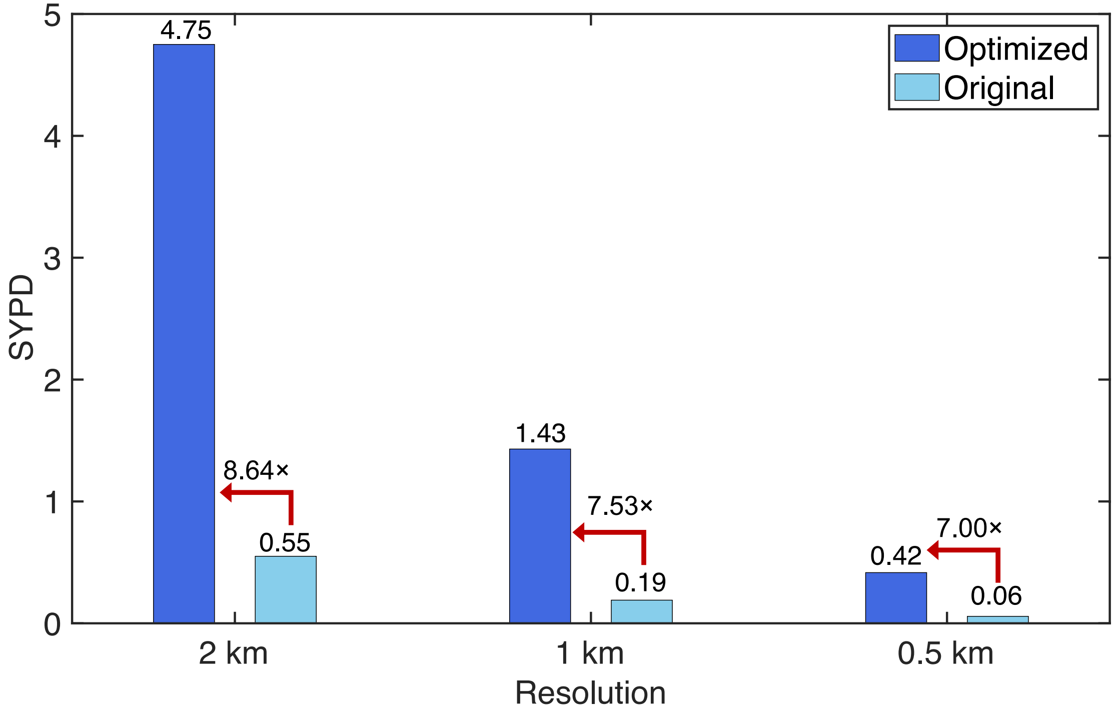

**Figure 15.** The computing throughput results by using 27,988,480 cores. SYPD (Simulated Years Per Day) increases from 0.19 to 1.43 with the 1 km resolution. The optimized performance is 7.53 times better than that of the original version.

algorithms proposed based on the new generation of Sunway supercomputers provide an important reference for ultrahigh resolution ocean circulation forecasting. However, only benchmarks are tested in the work, and real application data has not been used. In the future, we will build an ultrahigh-resolution ocean circulation forecasting model under real scenarios and conduct in-depth research to provide efficient solutions to predict ocean circulation and climate change accurately.

From the view of software and hardware co-design, the following should be focused on in the future. The first is the decomposition and load balance. For the model design, we should find the proper decomposing scheme to utilize the computer architecture fully. Besides the time dimension, solving an ocean general circulation model is a 3-dimension problem, with longitude, latitude, and depth. Usually, only the longitude-latitude domain is decomposed. In our work, driven by the RMA technology, we achieved the longitude-latitude-depth domain decomposed, which enables better large-scale scalability. Meanwhile, keeping a good load balance is also important for scalability. For the computer design, the RMA technology is a good example, which enables the longitude-latitude-depth domain decomposing. In other words, the high communication bandwidth between different cores or nodes will help achieve large-scale scalability. The second is communications. With the increasing processes used for model simulation, the ratio of communications time to computational time will increase. For the model design, the first thing is to avoid the global operator, such as ALLREDUCE, and BCAST, which will take more time with increasing the processes. Otherwise, it will be the crucial bottleneck. Meanwhile, we also should pack the exchanged data between different processes as much as possible. For the computer design, the low latency will help in saving the communications time. The third is reduced-precision. Our work's results demonstrate that there is a great potential to save computational time by incorporating the mixed double-, single-, and half-precision into the model. For the model design, we should understand the minimum computational precision requirements essential for successful ocean simulations, and then revise or develop arithmetic. For the computer design, the support for half-precision should be considered in the future.

Overall, the above are only several examples for further improving the performance of ocean modeling from perspectives of model development and computer design. Furthermore, other aspects such as I/O efficiency, and the trade-off between precision and energy consumption should also be considered. Moreover it should be noted that these suggestions are from different aspects of the model and computer development and need to be considered based on the software and hardware co-design ideology.

*Code availability.* The swNEMO_v4.0 source codes with user manual are available at https://doi.org/10.5281/zenodo.5976033 (Ye et al., 2022).

## Appendix A: Supplementary code in Fortran format

The following five chunks of code correspond to the five kernels stated in our previous experimental section. The codes listed below are exactly from NEMO for experimental validation.

**Kernel 1**

```
DO $jk = 2, jpkm1$
  DO $jj = 2, jpjm1$
    DO $ji = fs\_2, fs\_jpim1$
      $zmsku = wmask(ji,jj,jk)/MAX(umask(ji,jj,jk-1) + umask(ji-1,jj,jk)$
      $\& + umask(ji-1,jj,jk-1) + umask(ji,jj,jk), 1.0)$
      $zmskv = wmask(ji,jj,jk)/MAX(vmask(ji,jj,jk-1) + vmask(ji,jj-1,jk)$
      $\& + vmask(ji,jj-1,jk-1) + vmask(ji,jj,jk), 1.0)$
      $zahu\_w = (pahu(ji,jj,jk-1) + pahu(ji-1,jj,jk)$
      $\& + pahu(ji-1,jj,jk-1) + pahu(ji,jj,jk)) * zmsku$
      $zahv\_w = (pahv(ji,jj,jk-1) + pahv(ji,jj-1,jk)$
      $\& + pahv(ji,jj-1,jk-1) + pahv(ji,jj,jk)) * zmskv$
      $ah\_wslp2(ji,jj,jk) = zahu\_w * wslpi(ji,jj,jk) * wslpi(ji,jj,jk)$
      $\& + zahv\_w * wslpj(ji,jj,jk) * wslpj(ji,jj,jk)$
    END DO
  END DO
END DO
```

**Kernel 2**

```
  DO $jj = 2, jpjm1$
    DO $ji = fs_2, fs_jpim1$
      $pta(ji, jj, 1) = e3t_b(ji, jj, 1) * ptb(ji, jj, 1) + p2dt * e3t_n(ji, jj, 1) * pta(ji, jj, 1)$
    END DO
  END DO
  DO $jk = 2, jpkm1$
    DO $jj = 2, jpjm1$
      DO $ji = fs_2, fs_jpim1$
        $zrhs = e3t_b(ji, jj, jk) * ptb(ji, jj, jk, jn) + p2dt * e3t_n(ji, jj, jk) * pta(ji, jj, jk)$
        $pta(ji, jj, jk) = zrhs - zwi(ji, jj, jk)/zwt(ji, jj, jk - 1) * pta(ji, jj, jk - 1)$
      END DO
    END DO
  END DO
```

**Kernel 3**

```
  DO $jj = 2, jpjm1$
    DO $ji = fs_2, fs_jpim1$
      $zwt(ji, jj, 1) = zwd(ji, jj, 1)$
    END DO
  END DO
  DO $jk = 2, jpkm1$
    DO $jj = 2, jpjm1$
      DO $ji = fs_2, fs_jpim1$
        $zwt(ji, jj, jk) = zwd(ji, jj, jk) - zwi(ji, jj, jk) * zws(ji, jj, jk - 1)/zwt(ji, jj, jk - 1)$
      END DO
    END DO
  END DO
  DO $jk = 1, jpkm1$
    DO $jj = 2, jpjm1$
      DO $ji = fs_2, fs_jpim1$
        $zwi(ji, jj, jk) = -p2dt * zwt(ji, jj, jk)/e3w_n(ji, jj, jk)$
        $zws(ji, jj, jk) = -p2dt * zwt(ji, jj, jk + 1)/e3w_n(ji, jj, jk + 1)$
        $zwd(ji, jj, jk) = e3t_a(ji, jj, jk) - zwi(ji, jj, jk) - zws(ji, jj, jk)$
        END DO
    END DO
END DO
```

**Kernel 4**

```
DO $jk = 1, jpkm1$
    DO $jj = 1, jpjm1$
        DO $ji = 1, fs_jpim1$
            $zdit(ji,jj,jk) = (ptb(ji+1,jj,jk) - ptb(ji,jj,jk)) * umask(ji,jj,jk)$
            $zdjt(ji,jj,jk) = (ptb(ji,jj+1,jk) - ptb(ji,jj,jk)) * vmask(ji,jj,jk)$
        END DO
    END DO
END DO
DO $jk = 1, jpkm1$
    $zdk1t(:,:) = (ptb(:,:,jk) - ptb(:,:,jk+1)) * wmask(:,:,jk+1)$
    IF $jk == 1$ THEN $zdkt(:,:) = zdk1t(:,:)$
    ELSE
        DO $jj = 1, jpj$
            DO $ji = 1, jpi$
                $zdkt(ji,jj) = (ptb(ji,jj,jk-1) - ptb(ji,jj,jk)) * wmask(ji,jj,jk)$
            END DO
        END DO
    END IF
    DO $jj = 1, jpjm1$
        DO $ji = 1, fs_jpim1$
            $zabe1 = pahu(ji,jj,jk) * e2_e1u(ji,jj) * e3u_n(ji,jj,jk)$
            $zabe2 = pahv(ji,jj,jk) * e1_e2v(ji,jj) * e3v_n(ji,jj,jk)$
            $zmsku = 1./MAX(wmask(ji+1,jj,jk) + wmask(ji,jj,jk+1)$
            $\& + wmask(ji+1,jj,jk+1) + wmask(ji,jj,jk), 1.)$
            $zmskv = 1./MAX(wmask(ji,jj+1,jk) + wmask(ji,jj,jk+1)$
            $\& + wmask(ji,jj+1,jk+1) + wmask(ji,jj,jk), 1.)$
            $zcof1 = -pahu(ji,jj,jk) * e2u(ji,jj) * uslp(ji,jj,jk) * zmsku$
            $zcof2 = -pahv(ji,jj,jk) * e1v(ji,jj) * vslp(ji,jj,jk) * zmskv$
            $zftu(ji,jj,jk) = (zabe1 * zdit(ji,jj,jk)$
            $\& + zcof1 * (zdkt(ji+1,jj) + zdk1t(ji,jj)$
            $\& + zdk1t(ji+1,jj) + zdkt(ji,jj))) * umask(ji,jj,jk)$
            $zftv(ji,jj,jk) = (zabe2 * zdjt(ji,jj,jk)$
            $\& + zcof2 * (zdkt(ji,jj+1) + zdk1t(ji,jj)$
            $\& + zdk1t(ji,jj+1) + zdkt(ji,jj))) * vmask(ji,jj,jk)$
         END DO
      END DO
      DO $jj = 2, jpjm1$
         DO $ji = fs_2, fs_j pim1$
            $pta(ji,jj,jk,jn) = pta(ji,jj,jk,jn) + zsign * (zftu(ji,jj,jk) - zftu(ji-1,jj,jk)$
            $\& + zftv(ji,jj,jk) - zftv(ji,jj-1,jk))$
            $\& * r1_e 1e2t(ji,jj)/e3t_n(ji,jj,jk)$
         END DO
      END DO
   END DO
```

**Kernel 5**

```
   DO $jk = 2, jpkm1$
      DO $jj = 2, jpjm1$
         DO $ji = fs_2, fs_j pim1$
            $zmsku = wmask(ji,jj,jk)/MAX(umask(ji,jj,jk-1) + umask(ji-1,jj,jk)$
            $\& + umask(ji-1,jj,jk-1) + umask(ji,jj,jk), 1.0)$
            $zmskv = wmask(ji,jj,jk)/MAX(vmask(ji,jj,jk-1) + vmask(ji,jj-1,jk)$
            $\& + vmask(ji,jj-1,jk-1) + vmask(ji,jj,jk), 1.0)$
            $zahu_w = (pahu(ji,jj,jk-1) + pahu(ji-1,jj,jk)$
            $\& + pahu(ji-1,jj,jk-1) + pahu(ji,jj,jk)) * zmsku$
            $zahv_w = (pahv(ji,jj,jk-1) + pahv(ji,jj-1,jk)$
            $\& + pahv(ji,jj-1,jk-1) + pahv(ji,jj,jk)) * zmskv$
            $zcoef3 = -zahu_w * e2t(ji,jj) * zmsku * wslpi(ji,jj,jk)$
            $zcoef4 = -zahv_w * e1t(ji,jj) * zmskv * wslpj(ji,jj,jk)$
            $ztfw(ji,jj,jk) = zcoef3 * (zdit(ji,jj,jk-1) + zdit(ji-1,jj,jk)$
            $\& + zdit(ji-1,jj,jk-1) + zdit(ji,jj,jk))$
            $\& + zcoef4 * (zdjt(ji,jj,jk-1) + zdjt(ji,jj-1,jk)$
            $\& + zdjt(ji,jj-1,jk-1) + zdjt(ji,jj,jk))$
            $ztfw(ji,jj,jk) = ztfw(ji,jj,jk) + e1e2t(ji,jj)/e3w_n(ji,jj,jk) * wmask(ji,jj,jk)$
            $\& * (ah_w slp2(ji,jj,jk) - akz(ji,jj,jk))$
            $\& * (ptb(ji,jj,jk-1) - ptb(ji,jj,jk)$
         END DO
```

```
    END DO
  END DO
```

*Author contributions.* Y. Ye, Z. Song, and S. Zhou: Methodology, coding, numerical experiment, analysis, and writing–original draft preparation; Y. Liu, Q. Shu, and B. Wang: Algorithm description, and validation; Y. Liu and W. Liu: Algorithm and numerical experiments' suggestion; F. Qiao, and L. Wang: Conceptualization, supervision, funding acquisition, and writing–review and editing; All authors discussed, read, edited, and approved the article. All authors have read and agreed to the published version of the manuscript.

*Competing interests.* The authors have declared that neither they nor their coauthors have any competing interests.

*Acknowledgements.* This work is supported by the Marine S&T Fund of Shandong Province for Pilot National Laboratory for Marine Science and Technology (Qingdao) (2022QNLM010202), the National Natural Science Foundation of China (U1806205, 42022042, and 41821004), the China-Korea Cooperation Project on Northwest Pacific Marine Ecosystem Simulation under the Climate Change, and CAS Interdisciplinary Innovation Team (JCTD-2020-12).

## References

Afzal, A., Ansari, Z., Faizabadi, A. R. and Ramis, M.: Parallelization strategies for computational fluid dynamics software: state of the art review, Archives of Computational Methods in Engineering, vol. 24, no. 2, pp. 337-363, 2017.

Allison, H. Baker, et al: "Evaluating statistical consistency in the ocean model component of the Community Earth System Model (pyCECT v2.0)", Geoscientific Model Development, 2016.

Bryan, K. and Cox, M. D.: A numerical investigation of the oceanic general circulation, Tellus, vol. 19, no. 1, pp. 54-80, 1967. [Online]. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.2153- 3490.1967.tb01459.x

Bryan, K.: A numerical method for the study of the circulation of the world ocean, Journal of Computational Physics, vol. 4, no. 3, pp. 347-376, 1969. [Online]. https://www.science-direct.com/science/article/pii/0021999169900047

Chassignet, E. P., LeSommer, J. and Wallcraft, A. J.: Generalcirculation models, Encyclopedia of Ocean Sciences, 3rd edn., edited by: Cochran, KJ, Bokuniewicz, HJ, and Yager, PL, vol. 5, pp. 486-490, 2019.

Dawson, A. and Du ben, P. D.: rpe v5: an emulator for reduced floating- point precision in large numerical simulations, Geoscientific Model Development, vol. 10, no. 6, pp. 2221-2230, 2017.

Dong, J., Fox-Kemper, B., Zhang, H., and Dong, C.: The seasonality of submesoscale energy production, content, and cascade, Geophysical Research Letters, vol. 47, no. 6, p. e2020GL087388, 2020.

Eyring, V., Bony, S.,Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J. and Taylor, K. E.: Overview of the coupled model intercomparison project phase 6 (cmip6) experimental design and organization, Geo- scientific Model Development, vol. 9, no. 5, pp. 1937-1958, 2016.

Fu, H., Liao, J., Yang, J., Wang, L., Song, Z., Huang, X., Yang, C., Xue, W., Liu, F., Qiao, F. et al.: The sunway taihulight supercomputer: system and applications, Science China Information Sciences, vol. 59, no. 7, pp. 1-16, 2016.

Fu, H., Liao, J., Xue, W., Wang, L., Chen, D., Gu, L., Xu, J., Ding, N., Wang, X., He, C., et al.: Refactoring and optimizing the community atmosphere model (cam) on the sunway taihulight supercomputer, in SC16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, pp. 969-980, 2016.

Gu, J., Feng, F., Hao, X., Fang, T., Zhao, C., An, H., et al.: Establishing a non-hydrostatic global atmospheric modeling system at 3-km horizontal resolution with aerosol feedbacks on the Sunway supercomputer of China, Science Bulletin, vol. 67, pp.1170-1181, 2022.

Hu, Y., Huang, X., Wang, X.,Fu, H., Xu, S., Ruan, H., Xue, W. and Yang, G.: A scalable barotropic mode solver for the parallel ocean program, in European Conference on Parallel Processing. Springer, pp. 739-750, 2013.

Jiang, J., Lin, P., Wang, J., Liu, H., Chi, X., Hao, H., Wang, Y., Wang, W. and Zhang, L.: Porting lasg/iap climate system ocean model to gpus using openacc, IEEE Access, vol. 7, pp. 154490-154501, 2019.

Lellouche, J. M., Greiner, E., Galloudec, O. L., Garric, G., Regnier, C., Drevillon, M., Benkiran, M., Testut, C. E., Bourdalle-Badie, R., Gasparin, F. et al.: Recent updates to the copernicus marine service global ocean monitoring and forecasting real-time 1/12 high-resolution system, Ocean Science, vol. 14, no. 5, pp. 1093-1126, 2018.

Liao, X., Xiao, L., Yang, C. and Lu, Y.: Milkyway-2 supercomputer: system and application, Frontiers of Computer Science, vol. 8, no. 3, pp. 345-356, 2014.

Wan, L.: The high resolution global ocean forecasting system in the nmefc and its intercomparison with the godae oceanview iv-tt class 4 metrics, Website, 2020, https://www.godae.org/ godae-data/OceanView/Events/DA-OSEval-TT-2017/2.3-NMEFC-High-Resolution-Global-Ocean-Forecasting-System-and-Vali-dation v3.pdf.

Madec, G. and Team, N. S. NEMO ocean engine, 2016.

Aumont, Oliver, et al: "PISCES-v2: an ocean biogeochemical model for carbon and ecosystem studies." Geoscientific Model Development, vol. 8, no. 8, pp. 2465-2513, 2015.

Prims, O. T., Acosta, M. C., Moore, A. M., Castrillo, M., Serradell, K., Cortés, A. and Doblas-Reyes, F. J.: How to use mixed precision in ocean models: exploring a potential reduction of numerical precision in nemo 4.0 and roms 3.6, Geoscientific Model Development, vol. 12, no. 7, pp. 3135-3148, 2019.

Putnam, A., Caulfield, A. M., Chung, E. S., Chiou, D., Constantinides, K., Demme, J., Esmaeilzadeh, H., Fowers, J., Gopal, G. P., Gray, J. et al.: A reconfigurable fabric for accelerating large-scale datacenter services, in 2014 ACM/IEEE: 41st International Symposium on Computer Architecture (ISCA). IEEE, pp. 13-24, 2014.

Qiao, F., Zhao, W., Yin, X., Huang, X., Liu, X., Shu, Q., Wang, G., Song, Z., Li, X., Liu, H. et al.: A highly effective global surface wave numerical simulation with ultra-high resolution, in SC16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, pp. 46-56, 2016.

Qiu, B., Chen, S., Klein, P., Wang, J., Torres, H., Fu, L. L. and Menemenlis, D.: Seasonality in transition scale from balanced to unbalanced motions in the world ocean, Journal of Physical Oceanography, vol. 48, no. 3, pp. 591-605, 2018.

Qiu, B., Chen, S., Klein, P., Torres, H., Wang, J., Fu, L. L. and Menemenlis, D.: Reconstructing upper-ocean vertical velocity field from sea surface height in the presence of unbalanced motion, Journal of Physical Oceanography, vol. 50, no. 1, pp. 55-79, 2020.

Ruston, B.: Validation test report for the dtic, Website, https://apps.dtic.mil/sti/pdfs/AD1090615.pdf, 2019.

Rocha, C. B., Gille, S. T., Chereskin, T. K. and Menemenlis, D.: Seasonality of submesoscale dynamics in the kuroshio extension, Geophysical Research Letters, vol. 43, no. 21, pp. 11-304, 2016.

Smith, R., Jones, P., Briegleb, B.,Bryan, F., Danabasoglu, G., Dennis, J., Dukowicz, J., Eden, C., Fox-Kemper, B., Gent, P. et al.: The parallel ocean program (pop) reference manual ocean component of the community climate system model (ccsm) and community earth system model (cesm), LAUR-01853, vol. 141, pp. 1-140, 2010.

Vazhkudai, S. S., deSupinski, B. R.,Bland, A. S., Geist, A., Sexton, J., Kahle, J.,Zimmer, C. J., Atchley, S., Oral, S., Maxwell, D. E. et al.: The design, deployment, and evaluation of the coral pre-exascale systems, in SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, pp. 661-672, 2018.

Viglione, G. A., Thompson, A. F., Flexas, M. M., Sprintall, J. and Swart, S.: Abrupt transitions in submesoscale structure in southern drake passage: Glider observations and model results, Journal of Physical Oceanography, vol. 48, no. 9, pp. 2011-2027, 2018.

Wang, P., Jiang, J., Lin, P., Ding, M., Wei, J., Zhang, F., Zhao, L., Li, Y., Yu, Z., Zheng, W., et al.: The gpu version of licom3 under hip framework and its large-scale application, Geoscientific Model Development Discussions, pp. 1-34, 2020.

Xu, S., Huang, X., Oey, L.-Y., Xu, F., Fu, H., Zhang, Y. and Yang, G.: Pom.gpu-v1.0: a gpu-based princeton ocean model, Geoscientific Model Development, vol. 8, no. 9, pp. 2815-2827, 2015. [Online]. Available: https://gmd.copernicus.org/articles/8/2815/2015/

Yang, X., Zhou, S., Zhou, S., Song, Z. and Liu, W.: A barotropic solver for high-resolution ocean general circulation models, Journal of Marine Science and Engineering, vol. 9, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2077-1312/9/4/421

Yuan, Y., Shi, F., Kirby, J. T. and Yu, Fu.: Funwave-gpu: Multiple-gpu acceleration of a boussinesq-type wave model, Journal of Advances in Modeling Earth Systems, vol. 12, no. 5, p. e2019MS001957, 2020.

Zeng, Y., Wang, L., Zhang, J., Zhu, G., Zhuang, Y. and Guo, Q.: Redistributing and optimizing high-resolution ocean model pop2 to million sunway cores, in International Conference on Algorithms and Architectures for Parallel Processing. Springer, pp. 2750289, 2020.

Zhang, S., Fu, H., Wu, L., Li, Y., Wang, H., Zeng, Y., Duan, X., Wan, W., Wang, L., Zhuang, Y. et al.: Optimizing high-resolution community earth system model on a heterogeneous many-core supercomputing platform, Geoscientific Model Development, vol. 13, no. 10, pp. 4809-4829, 2020.

Ye, Y., Song, Z., Zhou S. et al.:swNEMO(4.0), Zenodo[code], https://doi.org/10.5281/zenodo.5976033, 2022.

Zhao, W., Song, Z., Qiao, F. and Yin, X.: High efficient parallel numerical surface wave model based on an irregular quasi-rectangular domain decomposition scheme, Science China Earth Sciences, vol. 57, no. 8, pp. 1869-1878, 2014.