applied sciences

MDPI

Article

Deep Parallel Optimizations on an LASG/IAP Climate System Ocean Model and Its Large-Scale Parallelization

Huiqun Hao [1,2], Jinrong Jiang [1,*], Tianyi Wang [3], Hailong Liu [2,4], Pengfei Lin [2,4], Ziyang Zhang [5] and Beifang Niu [1,*]

1 Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
2 University of Chinese Academy of Sciences, Beijing 100049, China
3 School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
4 State Key Laboratory of Numerical Modeling for Atmospheric Sciences and Geophysical Fluid Dynamics (LASG), Institute of Atmospheric Physics (IAP), Chinese Academy of Sciences, Beijing 100029, China
5 State Grid Beijing Electric Power Company, Beijing 100031, China
* Correspondence: jjr@sccas.cn; (J.J.); bniu@sccas.cn (B.N.)

Abstract: This paper proposes a series of parallel optimizations on a high-resolution ocean model, the LASG/IAP Climate System Ocean Model (LICOM), which was independently developed by the Institute of Atmospheric Physics of the Chinese Academy of Sciences. The version of LICOM that we used was LICOM 2.1. In order to improve the parallel performance of LICOM, a series of parallel optimization methods were applied. We optimized the parallelization scheme to tackle the problem of load imbalance. Some communication optimizations were implemented, including data packing, the application of the least communication algorithm, and the replacement of communications with calculations. Furthermore, for the calculation procedures, we implemented some mature optimizations and expanded functions in a loop. Additionally, a hybrid of MPI and OpenMP, as well as an asynchronous parallel IO, was used. In this work, the optimized version of LICOM 2.1 was able to achieve a speedup of more than two times compared with the original code. The parallelization scheme optimization and the communication optimization produced considerable improvement in performance in the large-scale parallelization. Meanwhile, the newly optimized LICOM could scale up to 245,760 processor cores. However, for the original version, there was no speedup when scaled up to over 10,000 processor cores. Additionally, the problem of jumpy wall time during the time integration process was also tackled with this optimization. Finally, we conducted a practical simulation from 1993 to 2007 by using the optimized version of LICOM 2.1. The results showed that the mesoscale vortex was well simulated by the model.

Keywords: LICOM; meteorological model; parallel optimization

Citation: Hao, H.; Jiang, J.; Wang, T.; Liu, H.; Lin, P.; Zhang, Z.; Niu, B. Deep Parallel Optimizations on an LASG/IAP Climate System Ocean Model and Its Large-Scale Parallelization. *Appl. Sci.* **2023**, *13*, 2690. https://doi.org/10.3390/app13042690

Academic Editors: Antonio J. Nebro and Juan A. Gómez-Pulido

Received: 31 January 2023
Revised: 13 February 2023
Accepted: 16 February 2023
Published: 19 February 2023

1. Introduction

At present, changes in the global climate and ecological environment have become some of the most important scientific problems. The ocean, as a vital part of the global climate system, has become a heated topic of research for scientists. Ocean models have considerably developed and improved since 1969, when they were developed [1]. Now, there are various kinds of global ocean models. HYCOM [2], NEMO [3], MOM [4], and LICOM [5] are some global ocean models that are used for research and production. Among them, MOM, HYCOM, and NEMO represent ocean models that were developed in Europe and the USA. LICOM was developed by scientists from IAP-CAS [6]. LICOM is widely used in the area of climate simulation and prediction, as well as the area of the numerical simulation and prediction of air–sea coupling. Different editions of LICOM were used for the ocean components of three coupled air–sea models in the Sixth Coupled

Model Intercomparison Project (CMIP6) [7,8], which included the Flexible Global Ocean–Atmosphere–Land System model version 3 [9] with a finite-volume atmospheric model (FGOALS-f3) [10], the Flexible Global Ocean–Atmosphere–Land System model version 3 with a grid-point atmospheric model (CAS FGOALS-g3) [11], and the Chinese Academy of Sciences Earth System Model (CAS-ESM) [12]. LICOM is an important tool for investigating and forecasting ocean circulation and its mechanisms. Meanwhile, as a vital part of climate system models (CSMs) and Earth system model (ESMs) [13], LICOM's performance may considerably affect climate change simulations [14,15]. As the resolution increased, the enormous computing, communication, and input/output (IO) became significant scientific and engineering challenges for scientists [16]. It will take either more time or more computing resources to run models. Widely used in many areas, including material science [17], electrochemistry [18], and meteorology, high-performance computing has become a very powerful tool in scientific research [19]. Due to the time constraints of meteorological simulation software, parallel optimization in high-performance computing is considerably important for models. Most meteorological systems suffer from the issues of low simulation speed and the inability to utilize large-scale machines. Some work on optimizing meteorological simulating systems has been done by researchers to tackle these issues. Optimizations were conducted on NEMO [20]. The overall performance was improved by 31%. The Princeton ocean model (POM) [21] was transplanted into Sunway TaihuLight [22], a very powerful supercomputer. The new edition, swPOM, was 2.8 times faster than it was on conventional supercomputers. It could be scaled up to over 250,000 cores [23]. Meanwhile, in addition to oceanic models, researchers have performed optimizations on some atmospheric models, such as IAP-AGCM [24]. The optimized code scaled up to 196,608 CPU cores, attaining a speed of 11.1 simulation years per day (SYPD) at a high resolution of 25 km [25]. In this study, in order to improve the simulation speed of LICOM, we implemented a series of optimizations and achieved a considerable speedup in comparison with the original version. The fully optimized edition of LICOM was twice as fast as the original edition. The performance experiments were conducted on the "Era", "Tianhe II", and "Tianhe III" supercomputers. Additionally, although we performed optimizations on GPUs with OpenACC [26], CUDA [27], and HIP [28], the CPU version is still widely used. For instance, many machines, such as "Tianhe III", are still pure CPU machines. Moreover, since the CPU version of LICOM is used as the ocean component of various coupled Earth system models, it is necessary to deeply optimize the CPU version.

The rest of this paper is organized as follows. The following section introduces LICOM and its control flow, along with the parallel and communication algorithm. In Section 3, the detailed optimizations that we implemented on LICOM are illustrated. Section 4 describes the experimental setups and the performance of the optimized model. Finally, Section 5 draws conclusions concerning our optimization work on LICOM.

2. The LICOM Model

LICOM is used to solve the Navier–Stokes equations. An ocean circulation model can simulate ocean temperature, salinity, velocity, and sea surface height under certain initial and boundary conditions. Meanwhile, the results of simulations can be used as the lower boundary conditions of atmospheric models and sea ice models. In addition, they can provide boundary conditions for regional ocean models. In addition, the results are capable of providing information on marine environment variables with an equably distributed space and continuous time in order to cover the shortages in the currently uneven observation data. This is beneficial for understanding the physical mechanisms of oceanographic processes. According to the original N-S equations, LICOM uses a finite-difference discrete model equation to ensure the conservation of energy and volume that are transferred during a discrete process. Since there are considerable differences in the vertical direction during the processes of marine stratification and mixing, a method with different layers is used to solve the problem. LICOM uses sea surface fluctuation with a free

surface, including a surface gravity wave with a high speed and a Rossby wave with a low speed. In order to reduce the calculations, the model splits the surface wave mode and uses smaller time steps for integration, while it uses larger time steps for models that describe the vertical structure. During the process of integration, the interactions between the two time steps are kept. This method is called "the decomposition and interaction of models" [29]. The calculation of the barotropic model integral is considerably reduced through this decomposition. There are some procedures that cannot be captured by LICOM or by other ocean models, such as the procedure of turbulence. It is necessary to create processes of parameterization to describe these model-invisible procedures to realize their impacts. A low-resolution ocean model contains mesoscale vortex parameterization. However, a high-resolution ocean model (less than 10 km) can recognize mesoscale vortexes. Thus, there is no need for parameterization. Currently, LICOM has reached resolution levels of 10 km and even higher. It is capable of recognizing mesoscale vortexes and nicely simulating vortexes and their impacts. High-resolution ocean models use a corrected barotropic and baroclinic decomposition algorithm. Meanwhile, an improved double-adjustable and sticky disjunction diffusion scheme can be employed in the horizontal direction in the momentum equation and thermohaline equation. Therefore, mesoscale vortexes can be better simulated. The main control flow of LICOM is shown in Figure 1. The major processes in the integral loop include barotropic, baroclinic, and thermohaline processes, among others, and the Euler forward or leapfrog scheme is used.

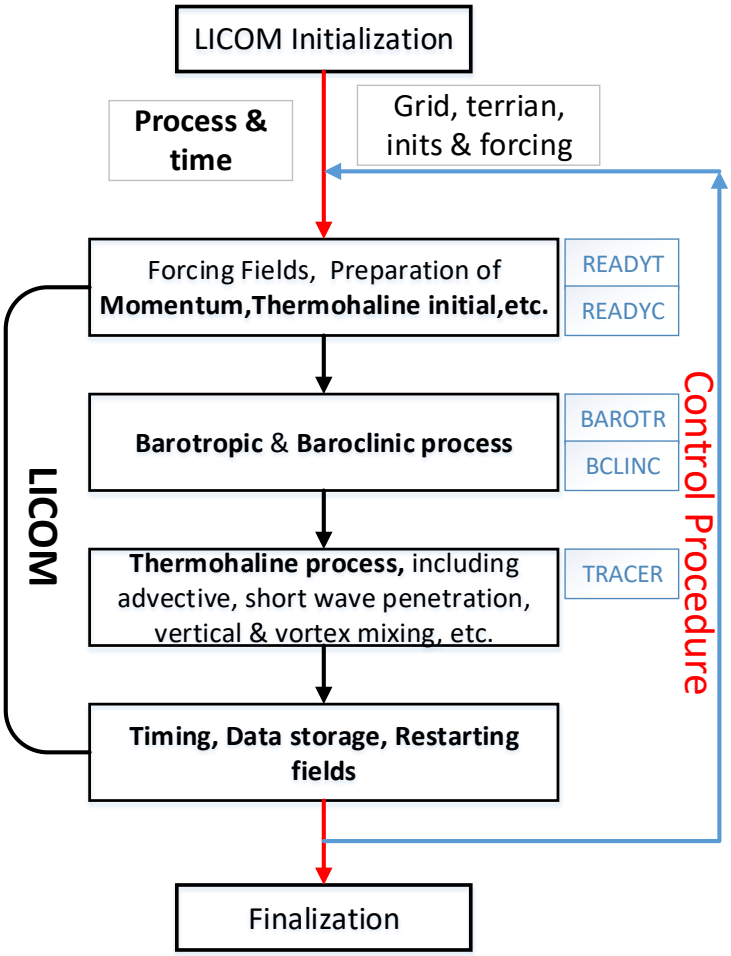

Figure 1. The control flow of LICOM.

LICOM has a variety of choices of output format, including binary files and netcdf. A binary file can be converted into a netcdf file. Thus, the results can be easily handled and investigated by using various types of professional software.

LICOM 's Parallel Scheme

The ocean is discretized into two-dimensional grid points with staggered latitude and longitude coordinates in the horizontal direction. The latitudinal direction is divided into *imt_global* grid points, and the longitudinal direction is divided into *jmt_global* grid points. The vertical direction is divided into km layers with the ocean depth as the coordinate. Thus, the ocean is decomposed into a three-dimensional structure. On this basis, the ocean grid points are organized into grid blocks. Each grid block contains $BLCKX \times BLCKY \times km$ grid points. There is barotropic-mode communication between grid blocks, and there are mutual pressure and vertical velocity calculations between layers. As shown in Figure 2, the ocean area is decomposed into $\lceil imt\_global/BLCKX \rceil \times \lceil jmt\_global/BLCKY \rceil$ grid blocks. In order to achieve a higher efficiency with the difference method, LICOM uses MPI to parallelize and speed up the calculation. $NBLOCKS\_CLINIC$ grid blocks are allocated to each process and sequentially calculated inside one grid block.

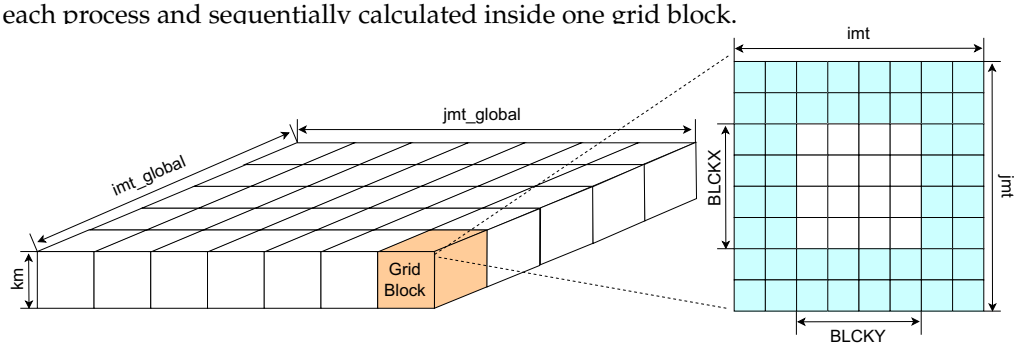

Figure 2. Ocean grid.

There is an independent array for each grid block. The size of the array decreases as the degree of parallelism increases. Since differential computation requires boundary values of adjacent grid blocks, frequent boundary communication is necessary. The array for each grid block is updated according to the array boundaries from two-dimensional logically adjacent processes. Therefore, the array of each grid block stores the boundary data of the adjacent grid block, in addition to its own data. The boundary data from other grid blocks are called the ghost boundary, while its own boundary data are called the real boundary. LICOM employs an adjustable ghost boundary strategy. The size of the ghost boundary depends on a specific issue. The blue squares in Figure 2 show the situation when the ghost equals 2. Setting the ghost to 2 instead of 1 will effectively eliminate the communication–calculation ratio, since communication is only needed for every two calculation steps.

Figure 3 illustrates the calculation procedure of a grid point in one iteration step. First, the two stripes of the ghost boundaries of each of the grid blocks are updated. The real boundary in grid blocks 1, c, and d is transferred to b and a of grid block 2. Meanwhile, the real boundary in grid blocks 2, c, and d is transferred to b and a of grid block 1. Then, the real boundaries are updated. In grid block 1, the grid points are updated from left to right until b. Meanwhile, in grid block 2, the grid points are updated from right to left until b. Finally, both grid blocks 1 and 2 update to c.

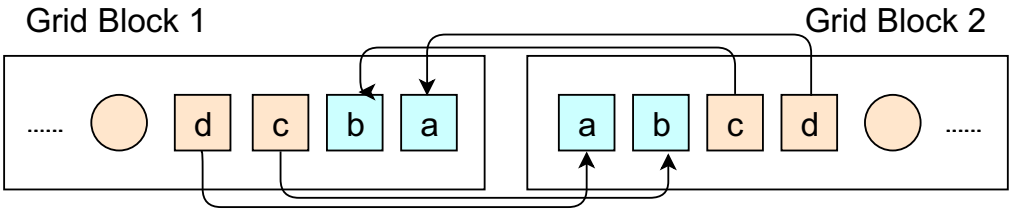

Figure 3. Ocean grid boundary communication.

3. Optimization of Parallelization

This section illustrates the series of optimizations that were implemented to speed up the running of LICOM. The original LICOM was parallelized by using only MPI. We utilized a hybrid of MPI and OpenMP in an optimized version of LICOM. Algorithm 1 shows an example of the implementation of OpenMP. However, due to the architecture of the supercomputers, the hybrid of MPI and OpenMP did not achieve good performance because the utilization of the MPI and OpenMP hybrid had no advantages at small scales. Thus, this optimization of the hybrid of MPI and OpenMP is only discussed, but is not included in the description of the performance tests in the following section. In the test on Tianhe III with a large number of PEs, we used the hybrid of MPI and OpenMP in order to utilize more PEs.

Algorithm 1: OpenMP implementation

```
OMP PARALLEL DO PRIVATE (J,I) for J←JSM to JEM do
    for I←2 to IMM do
        Calculate WKA(I,J,5) and WKA(I,J,6) ;
    end
end
```

3.1. Optimization A: Improving the Parallelization Scheme

We optimized the parallelization scheme of LICOM, as shown in Figure 4. N grid points were to be allocated to np processors. If N was divisible by np, the best decomposition was to allocate n, which equalled N divided by np, to each processor. However, in most cases, N was not divisible by np. Thus, in the original scheme, $n+1$ grid points were allocated to each processor from processor No. 0. Here, we set n to $\lfloor N/np \rfloor$. For the last processor, $N-(n+1)*(np-1)$ grid points were allocated. In the optimized scheme, n grid points were allocated to each processor, while the rest of the $N-n*np$ grid points were evenly allocated to some processors. In this way, the scalability of the model was considerably improved.

We redesigned the software structure of LICOM to improve the modernization of the software. Thus, LICOM was more convenient to use. For example, we replaced an alterable array with a fixed array, used structured data, and extracted important parameters. Therefore, there is no need to recompile or alter the code for different experiments or processor distributions.

3.2. Optimization B: Communication Optimization

In order to reduce the time consumed, we optimized the communication procedure. The method of data packing was employed to improve the communication time. In addition, the algorithms were improved by replacing communication with calculation.

(I) For a high-latitude area, a one-dimensional horizontal smoothing algorithm needs to conduct smoothing more than once. Since the times of smoothing on different latitudes are not the same, the original algorithm conducts smoothing once on each latitude, while the new algorithm involves just one smoothing and two-dimensional communication at all latitudes. Algorithms 2 and 3 show the representative code alterations. Algorithm 2 provides the original code, and Algorithm 3 describes the optimized version.

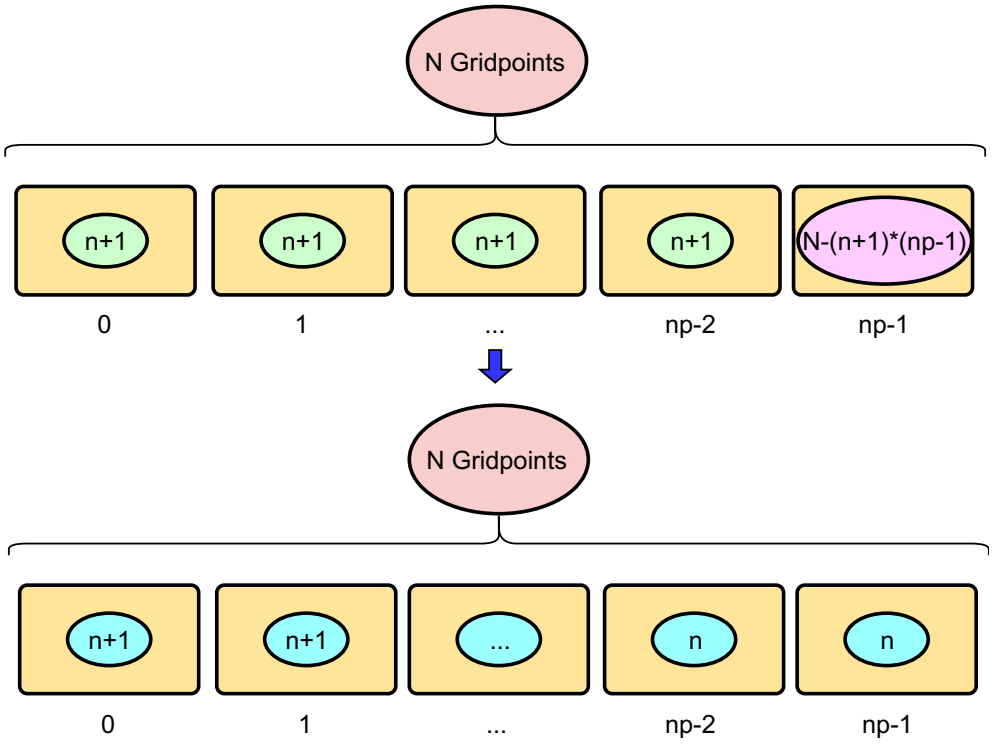

Figure 4. Optimization of the parallelization scheme.

Algorithm 2: Communication Optimization (I-A)

```
for J←JST to JMT do
    for JJ←1 to NN(j) do
        for K←1 to KK do
            for I←1 to IMT do
                Calculate XS(I);
            end
            for I←2 to IMM do
                Calculate X(I,J,K);
            end
        end
        Boundary Exchange 1D;
    end
end
```

(II) Based on previous test results when using Intel Vtune, the hotspots of LICOM are the communication functions. Therefore, the optimization of communications is of great importance. During the process of integration, the least communication algorithm was employed. Some communications were replaced by adding boundary calculations. It was beneficial to eliminate communication by adding some calculations. Because of the large number of PEs used, the total calculation job was divided by the number of PEs, while the cost of communication was multiplied by the number of PEs. Algorithms 4 and 5 show the representative code alterations. The original code in Algorithm 4 was optimized into the code in Algorithm 5 to eliminate the cost of communication.

Algorithm 3: Communication Optimization (I-B)

```
for NCY←1 to MAX_NN do
    for j←jst to jmt do
        if NN(j)≥NCY then
            for K←1 to KK do
                for I←1 to IMT do
                    Calculate XS(I);
                end
                for I←2 to IMM do
                    Calculate X(I,J,K);
                end
            end
        end
        Boundary Exchange 2D;
    end
end
```

Algorithm 4: Communication Optimization (II-A)

```
for J←JSM to JEM do
    for I←2 to IMM do
        Calculate UB(I,J), VB(I,J), and H0(I,J) ;
    end
end
Exchange boundary ub, vb, and h0 ;
for J←JSM to JEM do
    for I←2 to IMM do
        Calculate WKA(I,J,1), WKA(I,J,2), and WORK(I,J) ;
    end
end
```

Algorithm 5: Communication Optimization (II-B)

```
for J←JST to JET do
    for I←1 to IMT do
        Calculate UB(I,J), VB(I,J), and H0(I,J) ;
    end
end
for J←JST to JET do
    for I←1 to IMT do
        Calculate WKA(I,J,1), WKA(I,J,2), and WORK(I,J) ;
    end
end
```

(III) The grid matching information was obtained via communication in the original code in Listing 1. In contrast, in the optimized algorithm, it was obtained by conducting a calculation in Listing 2. Therefore, point-to-point communication was reduced by a considerable amount.

Listing 1. Communication Optimization (III-A)

```
if (mytid == 0) then
  i_start(1)= i_global(1)
  j_start(1)= j_global(1)
  do n=1,nproc-1
    call mpi_recv(j_start(n+1),1,mpi_integer,n,tag_1d,&
                  mpi_comm_ocn,status,ierr)
    call mpi_recv(i_start(n+1),1,mpi_integer,n,tag_2d,&
                  mpi_comm_ocn,status,ierr)
  end do
else
  j_start(1) =j_global(1)
  i_start(1) =i_global(1)
  call mpi_send(j_start(1),1,mpi_integer,0,tag_1d,&
                mpi_comm_ocn,ierr)
  call mpi_send(i_start(1),1,mpi_integer,0,tag_2d,&
                mpi_comm_ocn,ierr)
end if
```

Listing 2. Communication Optimization (III-B)

```
do i=1,nproc
  iix=mod(i-1,nx_proc)
  iiy=(i-1-iix)/nx_proc
  i_start(i)=iix*(imt-num_overlap)+1
  j_start(i)=3
  if(iiy/=0) then
    do j=1,iiy
      j_start(i)=j_start(i)+jmt-num_overlap
    end do
  endif
end do
```

3.3. Optimization C: Calculation Optimization

We carried out a series of mature optimizing methods, including vectorization, memory access optimization, instruction optimization, cache optimization, and runtime optimization. Thus, we could fully discover the potential performance of stencil calculation on the CPU cluster, such as by expanding the functions in a loop, as shown in Algorithms 6 and 7. The original DENS function in Algorithm 6 was expanded in the optimized version in Algorithm 7 in order to utilize vectorization.

Algorithm 6: Function Expansion A

```
for K←1 to KMM1 do
    for J←JST to JET do
        for I←2 to IMM do
            Calculate TUP, SUP, and RHOUP ;
            Calling DENS() function during calculation of RHOUP ;
        end
    end
end
```

Algorithm 7: Function Expansion B

```
for K←1 to KMM1 do
    for J←JST to JET do
        for I←2 to IMM do
            Calculate TUP, SUP, and RHOUP ;
            Instead of calling the DENS() function, the calculation procedure is
              written out in detail ;
        end
    end
end
```

3.4. Optimization D: Parallel IO

We also designed an asynchronous parallel IO method. As shown in Figure 5, an independent node was employed to conduct the IO procedures. A separate communication group was created for the IO work. When the computing nodes needed to carry out IO procedures after calculation procedures, the communication group of the computing nodes collected the data and sent it to the IO node. After receiving all of the data, the IO node began IO procedures. Meanwhile, the nodes from the computing group could continue with their calculation procedures instead of waiting for the completion of the IO procedures. Therefore, the elapsed time for IO was hidden by overlapping the calculations and IO.

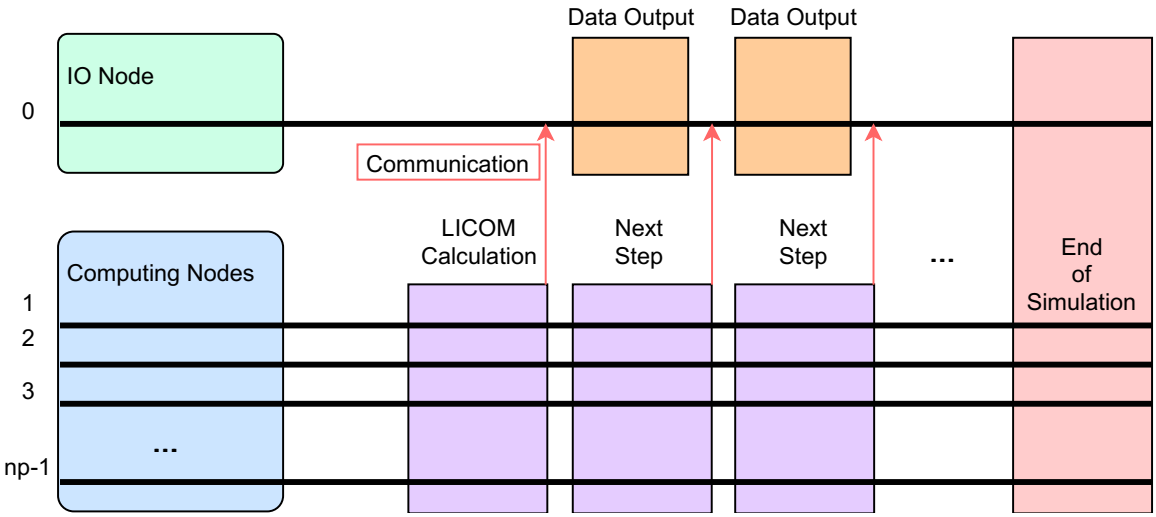

Figure 5. Asynchronous parallel IO.

4. Parallel Performance and Application in Actual Scenarios

Overall, we ran three sets of tests. The setup of the tests is shown in Table 1.

Table 1. Setup of Tests.

No.	Description	Machine
1	Performance test with three versions of code	Era and Tianhe II
2	Scalability test	Tianhe III
3	Real scenario test	Era

For the first test, we used three versions of the code for comparison. The code versions were (a) the original LICOM, (b) semi-optimized LICOM, in which optimization methods B(I), B(III), and C were employed, and (c) fully optimized LICOM. The hardware environments were the "Era" supercomputer and the "Tianhe II" supercomputer. For the second test, the fully optimized code was run on the "Tianhe III" supercomputer. For the third test, the fully optimized code was run on the "Era" supercomputer for a long term in a real scenario. For all of the tests on the three different hardware platforms, the same test case with a global resolution of 10 km $\times$ 10 km was used. The simulation started from 1993. The detailed configurations are listed in Table 2. These parameters controlled the run of LICOM and described the real scenario in our test. An elapsed time of one model day was used as an indicator for comparison. We ran each particular test at least five times to calculate the average value. The simulation year per day, as a widely used indicator of running speed, was calculated from the elapsed time of one model day.

Table 2. Configurations of the test cases.

Module	LICOM2.1
Horizontal resolution	0.1° (lat) $\times$ 0.1°(lon)
Vertical levels	55 levels
Grid point	3600 $\times$ 1683
Advection scheme	Shape-preserving [30]
Vertical mixing	Canuto [31]
Mesoscale eddy	Gent and McWilliams [32]
Horizontal viscosity	3×10^3 $m^2 s^{-1}$
Forcing formula and dataset	Large and Yeager [33]; COREs

4.1. Performance on Era

The CPU on the computing nodes of Era was an Intel(R) CPU E5-2680V3:2.5GHz. The operating system was Linux CentOS release 6.4 (Final). The compilers were Intel composer_xe_2013_sp1.0.080 and Intelmpi 4.1.3.049. Based on the above environment, we conducted tests using up to 4800 processor cores. Since LICOM is so complicated, the elapsed time of LICOM when simulating one model day is used as an indicator of performance. Figure 6 shows the elapsed time and running speed of the three editions of LICOM, while Figure 7 shows the speedups. The simulation year per day (SYPD) is usually used to measure the computational performance of models. The speedups for the three editions of code were calculated separately. The elapsed times for each edition of code on 1200 PEs were chosen as references. For instance, the speedup of the original code on 4800 PEs over 1200 PEs was the elapsed time of the original code on 1200 PEs divided by the elapsed time of the original code on 4800 PEs. The speedup of the fully optimized code on 4800 PEs over 1200 PEs was the elapsed time of the fully optimized code on 1200 PEs divided by the elapsed time of the fully optimized code on 4800 PEs. The reason for why the elapsed times of 1200 PEs were chosen as references was that, with a resolution of 10 km $\times$ 10 km, LICOM needed a very large memory space. Thus, we needed more PEs so that after decomposition, the limited memory space on each PE could meet the demands

of LICOM. As shown in Figures 6 and 7, the semi-optimized LICOM was much faster than the original LICOM, but it still suffered from the problem of scalability. However, the fully optimized LICOM with the optimization of the decomposition scheme showed considerably good scalability when 4800 processor cores were used. The computing speed reached 9 model years per day. Additionally, the elapsed times of both the original and semi-optimized code on 3600 PEs were longer than those on 2400 PEs. This was a non-intuitive phenomenon. However, for the fully optimized code, there was no similar non-intuitive phenomenon. We can infer that the problem that caused the non-intuitive phenomenon was tackled by the optimizations in the fully optimized code. Therefore, the possible reasons that caused this non-intuitive phenomenon could have been the communication overhead and load imbalance.

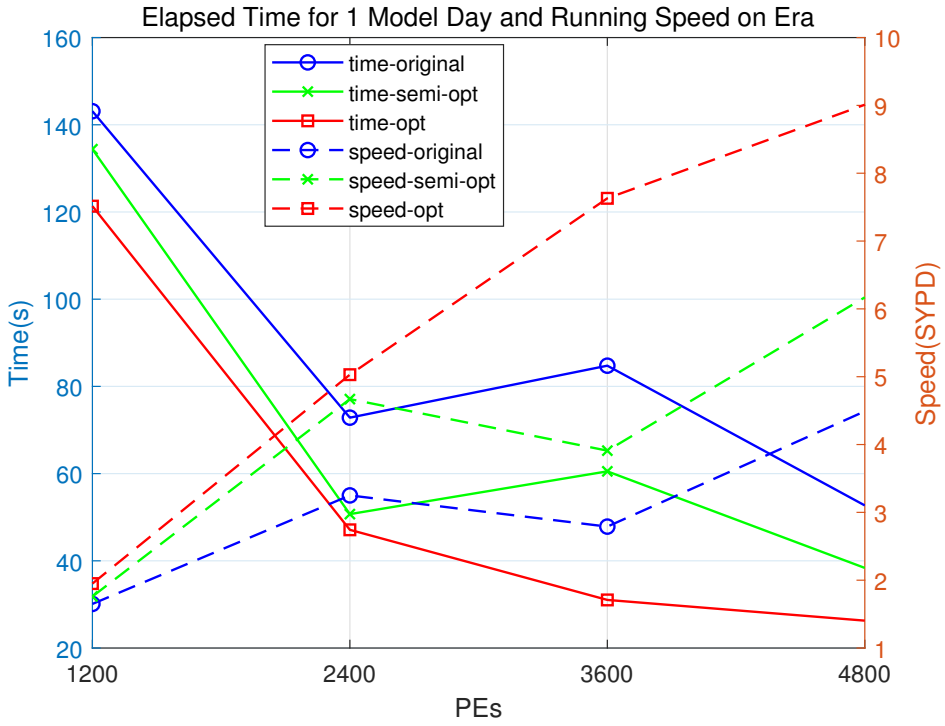

Figure 6. Elapsed time and speed of LICOM on Era.

4.2. Performance on Tianhe II

Based on the test results on Era, we decided to conduct tests on a larger scale. The hardware environment was Tianhe II. The processor was an Intel Ivy Bridge-E Xeon E5-2692V2:2.2GHz. The operating system edition was Red Hat Enterprise Linux Server release 6.5 (Santiago). The compilers were Intel composer_xe_2013_sp1.2.144 and MPICH 3.1.3. In the above hardware environment, we conducted tests on the same three code editions. As shown in Figures 8 and 9, the fully optimized LICOM achieves a good speedup when 9600 and 19,200 processor cores were used. The computing speed reached 12.6 model years per day, which was twice the speed of the original LICOM. However, as shown in Figure 9, a non-intuitive phenomenon occurred. The speedup for the semi-optimized code on 9600 PEs was smaller than that on 4800 PEs. Similarly to the phenomenon on Era, this may have been due to the communication overhead and load imbalance. In contrast, the fully optimized code achieved good speedups on 9600 and 19,200 PEs. This showed that our optimization worked well in improving the scalability of the code.

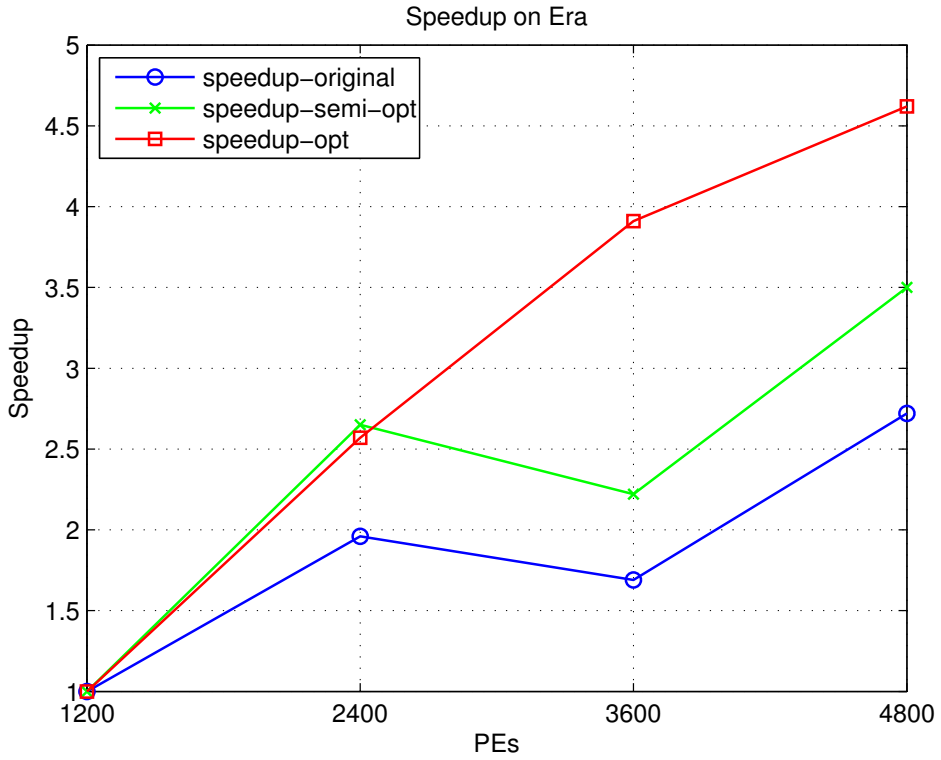

Figure 7. Speedup of LICOM on Era.

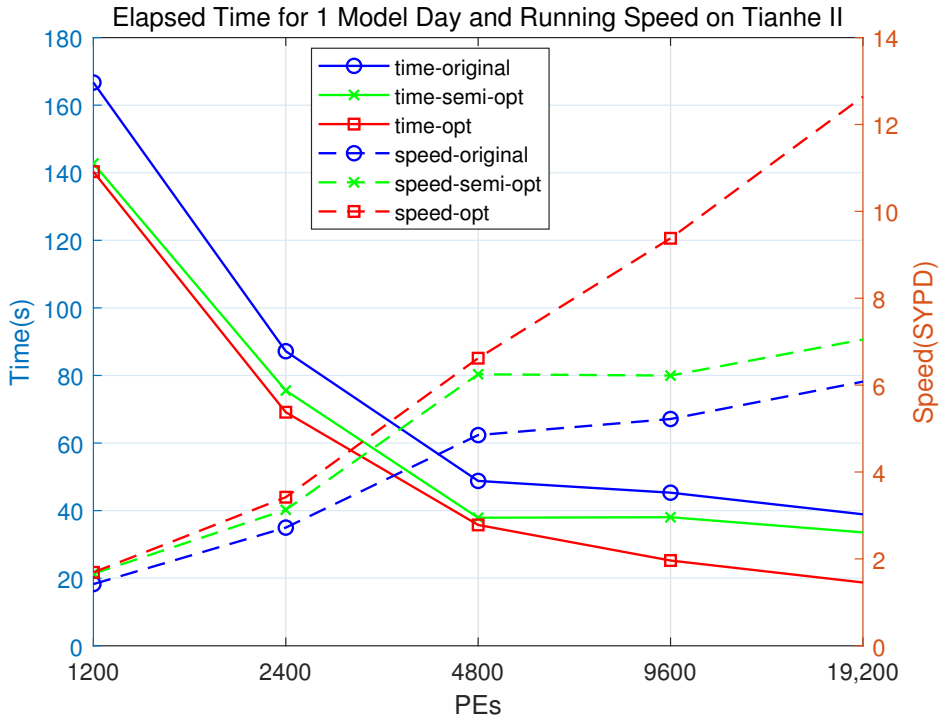

Figure 8. Elapsed time and speed of LICOM on Tianhe II.

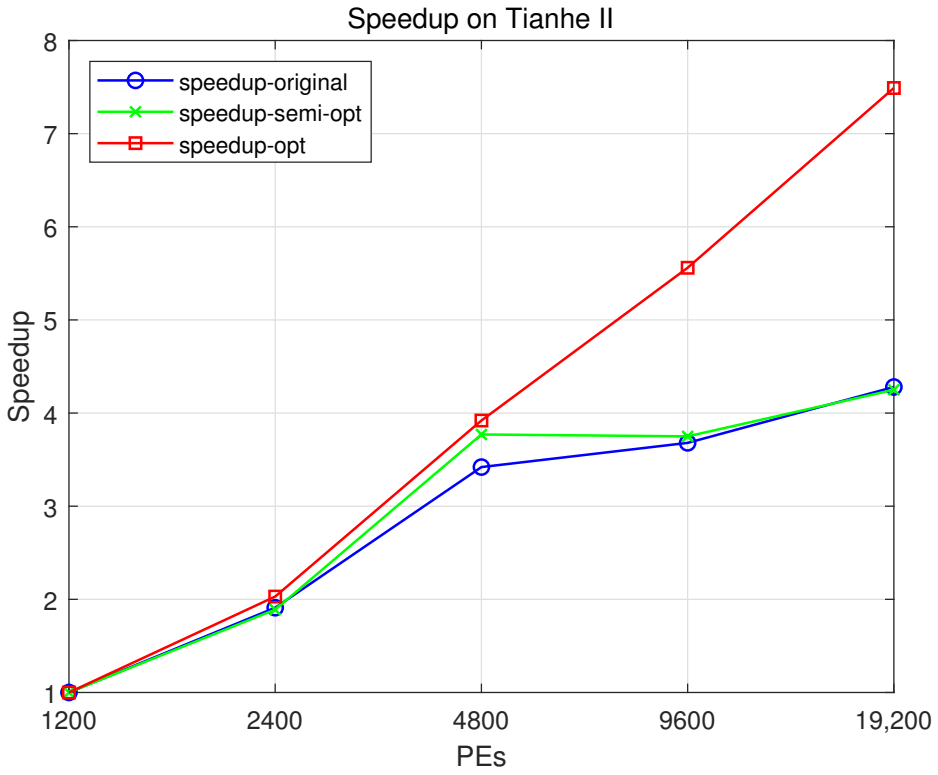

Figure 9. Speedup of LICOM on Tianhe II.

4.3. Performance on Tianhe III

We tested the fully optimized LICOM on the prototype system of the Tianhe III supercomputer. We reached up to 245,760 PEs, which was the summation of the CPU cores and the Matrix-2000 cores. Figure 10 shows the performance of LICOM on various numbers of PEs. The speedup in Figure 11 is the running time on each PE count divided by the running time on 1920 PEs. The reason for why 1920 PEs were chosen as the reference was that the elapsed time on 960 PEs was too long, which may lead to an abnormal speedup diagram. The speedup diagram in Figure 11 shows that there is still potential for optimization, since the speedup fell when more than 61,400 cores were used.

For the speedups on the three supercomputers, we can see that on Era and Tianhe II, there were still good speedups when 4800 and 19,200 PEs were used. The reason for why we did not test on more PEs was that we did not get access due to the policy for the supercomputer. However, on Tianhe III, the speedup was lower when more PEs were used. This might have been due to the communication overhead. Additionally, on a similar number of PEs, Era achieved the best speedup. For instance, on 4800 PEs (7680 on Tianhe III), the speedup for the fully optimized code on the three platforms was about 4.6, 3.9, and 3, respectively. A possible reason might be that Era had the CPU with the best performance.

4.4. Experiment on the Application of LICOM in a Real Scenario

Moreover, we conducted a set of application tests. LICOM was used with 10 km $\times$ 10 km as the resolution. CORE-II was employed as a forcing field. The simulation period was from 1993 to 2007. The output included temperature, salinity, sea surface height, and the 3D current field. Figure 12 shows the abnormal sea surface height value (difference from the average value) on 31 December 2007. As can be seen, an eddy was apparent. Additionally, there was a special difference. The middle image in Figure 12 shows the average number of eddies on every grid point. West and east boundary eddies often occurred in the southern ocean. There were at least 50 eddies that occurred in some areas. Therewasis a considerable difference between the structures of the eddies of anticyclones and cyclones, as shown in the bottom image in Figure 12. An anticyclone eddy

had a sunken center with a temperature increment of 2°, while a cyclone eddy had a raised center with a temperature decline of 2°.

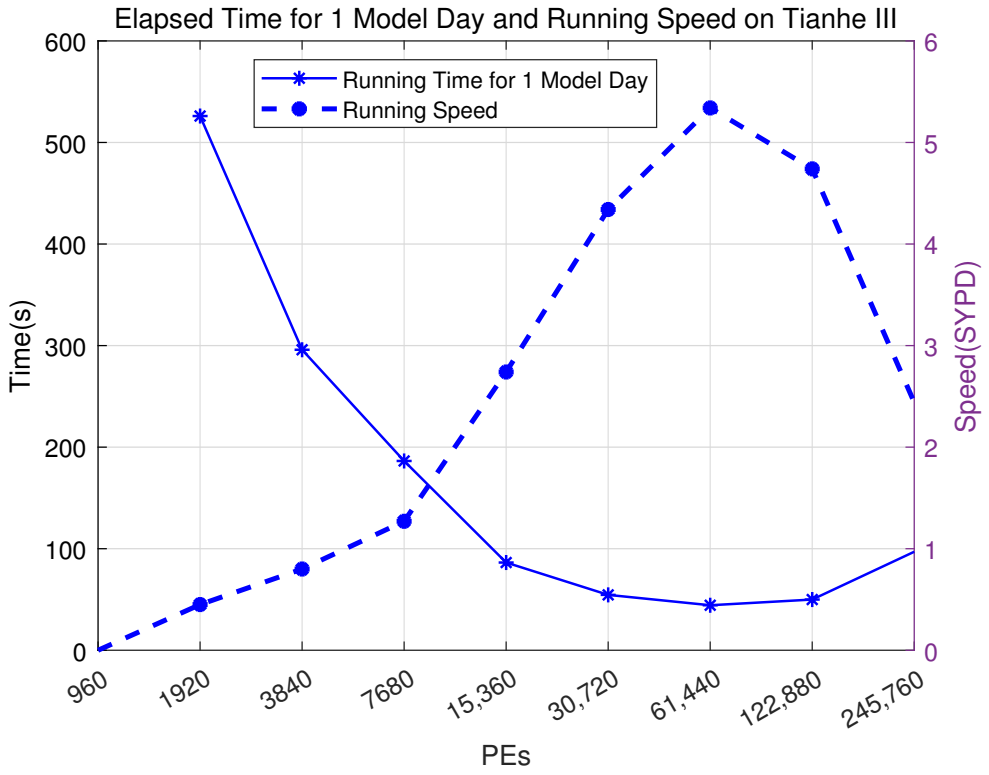

Figure 10. Elapsed time and speed of LICOM on Tianhe III.

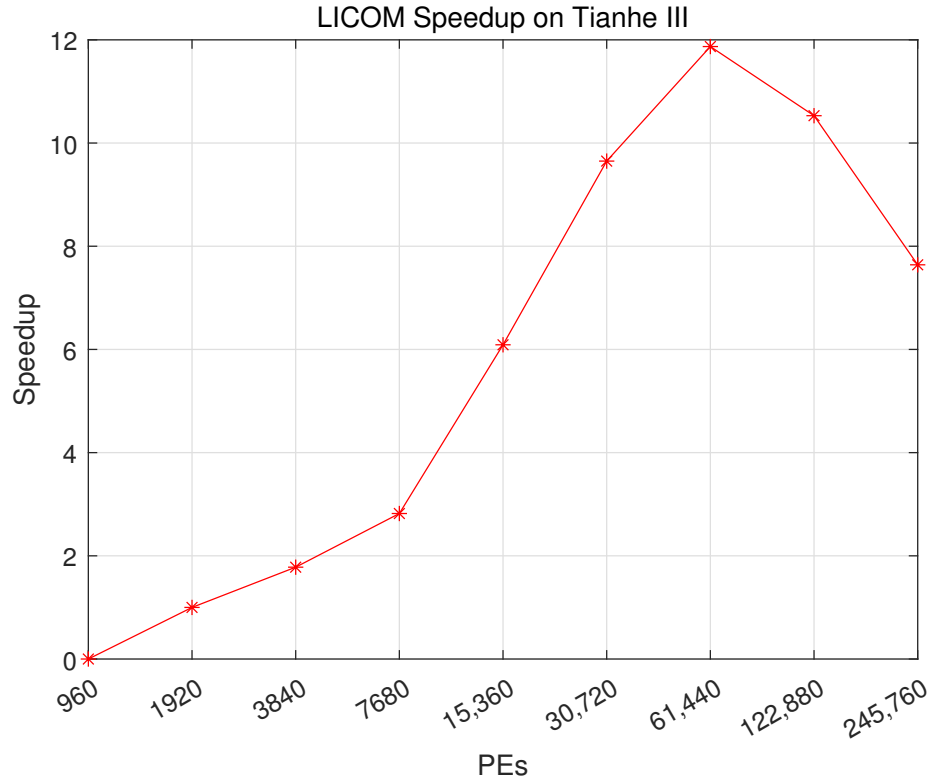

Figure 11. Speedup of LICOM on Tianhe III.

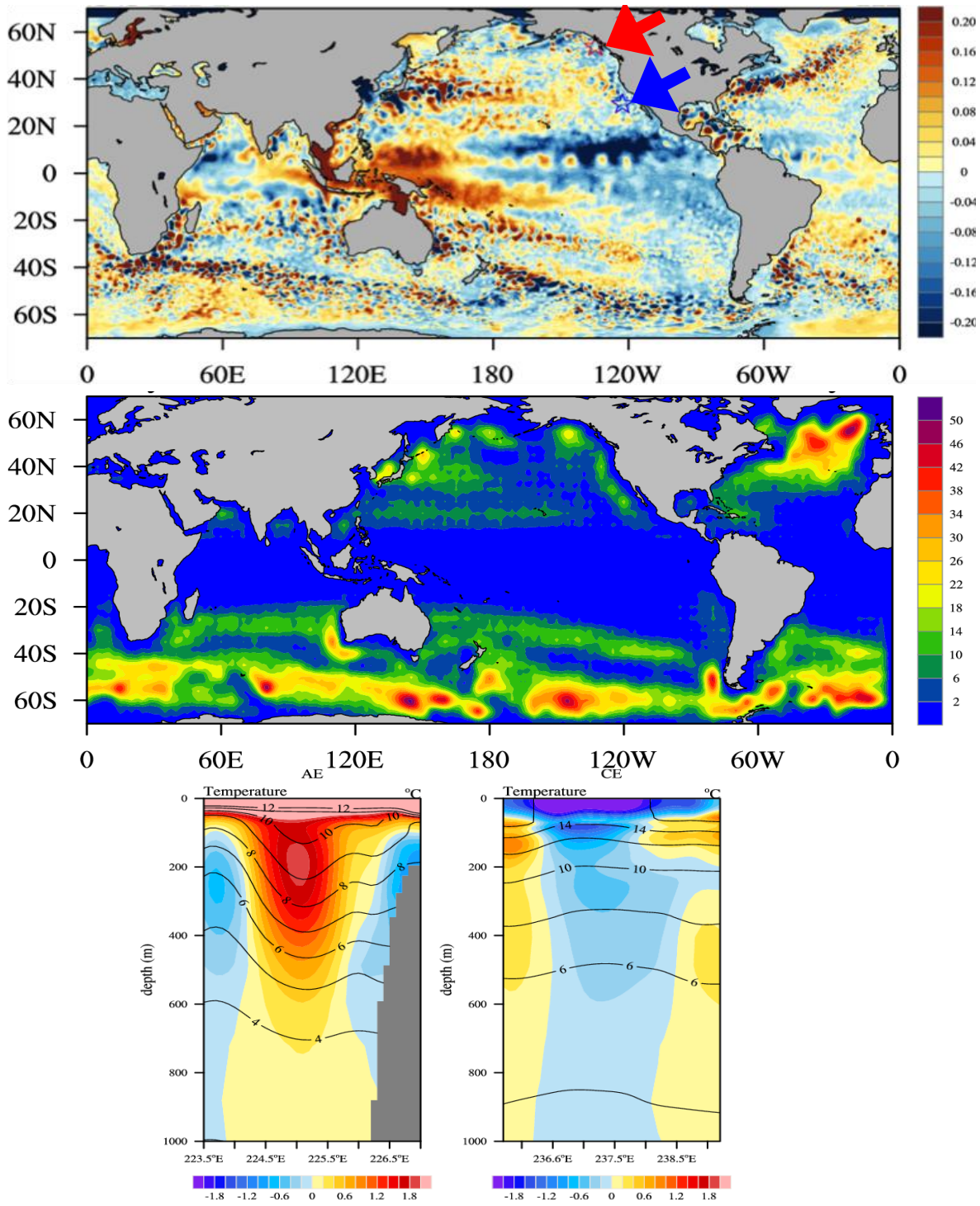

Figure 12. Output of the LICOM application.

In addition, we compared the results produced by the original code and the fully optimized code to see whether there were any differences. Figure 13 shows the results of the sea surface temperature and the differences between the two editions.

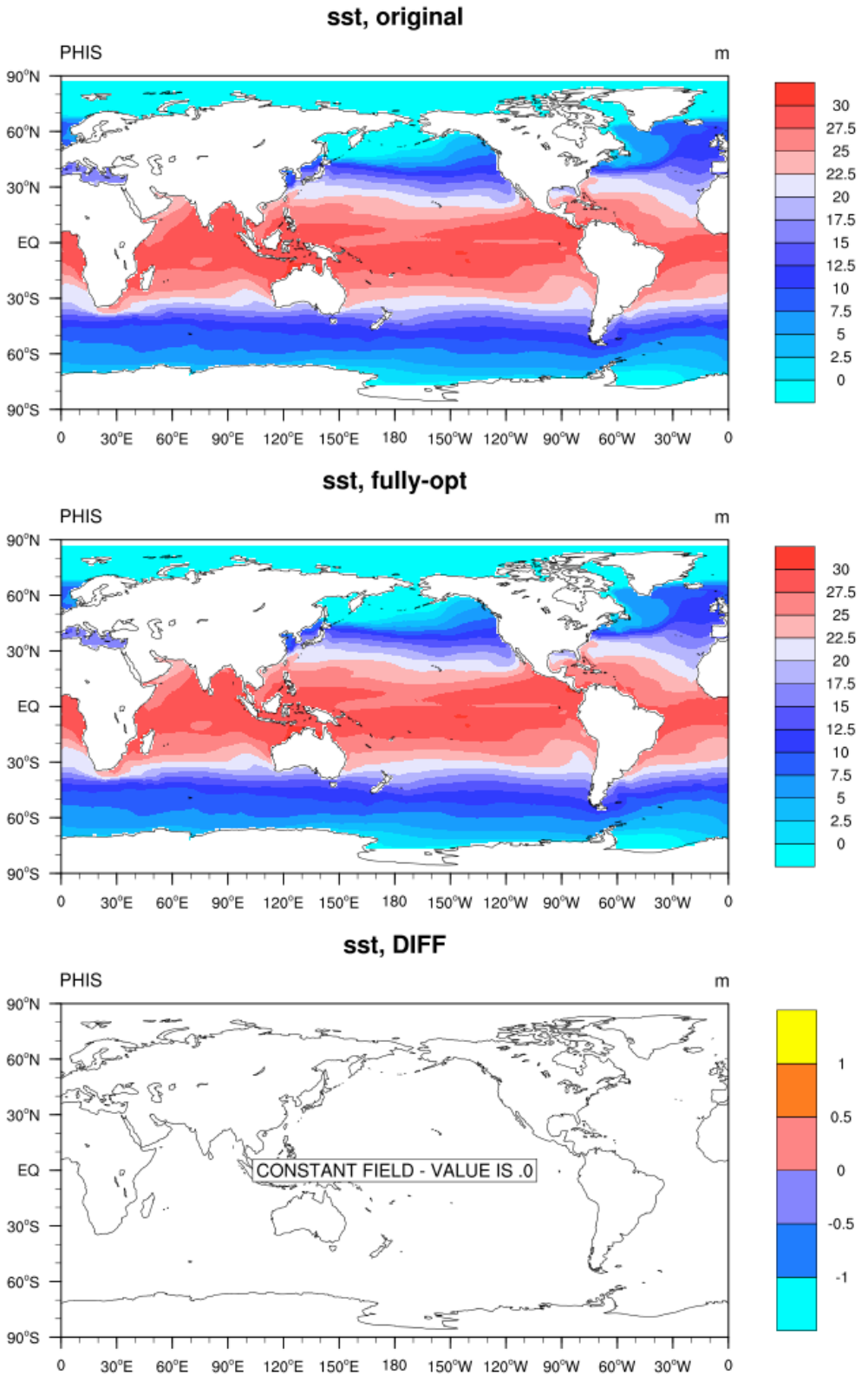

Figure 13. Correctness of simulations.

5. Conclusions

High-resolution general ocean circulation models are also called "eddy-resolving" ocean circulation models, and they are usually models with resolutions higher than 10 km. This kind of model is capable of simulating the characteristics of mesoscale eddies and

their climate effects. Moreover, an "eddy-resolving" ocean model can describe submarine topography, the land–sea distribution, and the spatial and temporal structure of an ocean's west boundary circulation well. Therefore, the development of global "eddy-resolving" ocean circulation models has been drawing people's attention in the field of physical oceanography and climate research.

In this work, we applied several parallel optimization methods on LICOM, including improvements in the parallelization scheme, communication optimization, the floating-point performance, the asynchronous IO, hybrid programming of MPI and OpenMP, and the redesign of the software structure. The performance of the distributed cluster was fully utilized. The computing speed of the optimized version of LICOM reached 12.6 model years per day when 19,200 processor cores were used, which was twice that of the original LICOM. The optimized LICOM could scale up to 245,760 processor cores. However, for the old version, there would not be much of a speedup when more than 19,200 processor cores were used. This is a vital improvement thanks to the optimization in this work. As mentioned in Section 1, swPOM can be scaled up to 250,000 cores. Although it is not appropriate to simply compare the scalability of different systems on different machines, the results of our work are around the same level as that of other researchers' work. We found that the optimization of communications and the tackling of load imbalance have considerable benefits in improving the performance of LICOM according to our test results.

In addition, we conducted simulations of a real scenario from 1993 to 2007 by using the optimized LICOM. The results showed that mesoscale vortexes were well simulated by the model. In conclusion, our optimization work considerably improved the performance of LICOM in terms of computing speed and scalability.

Author Contributions: Conceptualization, H.L., P.L., B.N. and J.J. ; methodology, J.J.; software, T.W., H.H. and J.J.; validation, H.H. and T.W.; formal analysis, B.N.; investigation, H.H.; resources, J.J.; data curation, H.H. and T.W.; writing—original draft preparation, H.H.; writing—review and editing, J.J.; visualization, P.L. and Z.Z.; supervision, J.J.; project administration, J.J.; funding acquisition, J.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (Grant No. 41931183) and the National Key Scientific and Technological Infrastructure project, "Earth System Science Numerical Simulator Facility" (EarthLab).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

IO	Input and Output
LICOM	LASG/IAP Climate System Ocean Model
LASG	State Key Laboratory of Numerical Modeling for Atmospheric Sciences Geophysical Fluid Dynamics
IAP	Institute of Atmospheric Physics
HYCOM	Hybrid Coordinate Ocean Model
NEMO	Nucleus for the European Modeling of the Ocean
MOM	Modular Ocean Model
CMIP6	The Sixth Coupled Model Intercomparison Project

FGOALS-f3	Flexible Global Ocean–Atmosphere–Land System model version 3 with a finite-volume atmospheric model
FGOALS-g3	Flexible Global Ocean–Atmosphere–Land System model version 3 with a grid-point atmospheric model
CAS-ESM	Chinese Academy of Sciences Earth System Model
CSM	Climate System Model
ESM	Earth System Model
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
HIP	Heterogeneous Compute Interface for Portability

References

1. Zhang, X.; Yu, Y.; Liu, H. The development and application of ocean circulation model I. Global general ocean circulation model. *Chin. J. Atmos. Sci.* **2003**, *27*, 607–617.
2. Wallcraft, A.J.; Metzger, E.J.; Carroll, S.N. Software Design Description for the HYbrid Coordinate Ocean Model (HYCOM), version 2.2. 2009. Available online: https://apps.dtic.mil/sti/citations/ADA494779 (accessed on 10 August 2022)
3. Madec, G.; Bourdallé-Badie, R.; Bouttier, P.A.; Bricaud, C.; Bruciaferri, D.; Calvert, D.; Chanut, J.; Clementi, E.; Coward, A.; Delrosso, D.; et al. NEMO ocean engine. In *Scientific Notes of IPSL Climate Modelling Center (v4.2, Number 27)*; Zenodo: Honolulu, HW, USA, 2022. [CrossRef]
4. Pacanowski, R.C.; Dixon, K.W.; Rosati, A. GFDL Modular Ocean Model, Users Guide Version 1.0. *Gfdl. Tech. Rep.* **1991**, *2*, 142.
5. Liu, H.; Lin, P.; Yu, Y.; Zhang, X. The Baseline Evaluation of LASG/IAP Climate System Ocean Model (LICOM) Version 2. *Acta Meteorol. Sin.* **2012**, *26*, 318–329. [CrossRef]
6. Liu, H.; Yu, Y.; Liu, X.; Zhang, X. The Development of LASG/IAP Climate System Ocean Circulation Model(LICOM)(Abstract). In Proceedings of the Chinese Meteorological Society Annual Conference, Beijing, China, 8 December 2003.
7. Lin, P.; Yu, Z.; Liu, H.; Yu, Y.; Li, Y.; Jiang, J.; Xue, W.; Chen, K.; Yang, Q.; Zhao, B.; et al. LICOM Model Datasets for the CMIP6 Ocean Model Intercomparison Project. *Adv. Atmos. Sci.* **2020**, *37*, 239–249. [CrossRef]
8. Griffies, S.; Danabasoglu, G.; Durack, P.; Adcroft, A.; Balaji, V.; Böning, C.; Chassignet, E.; Curchitser, E.; Deshayes, J.; Drange, H.; et al. OMIP contribution to CMIP6: Experimental and diagnostic protocol for the physical component of the Ocean Model Intercomparison Project. *Geosci. Model Dev.* **2016**, *9*, 3231–3296. [CrossRef]
9. Li, Y.W.; Liu, H.L.; Ding, M.R.; Lin, P.F.; Yu, Z.P.; Meng, Y.; Li, Y.L.; Jian, X.; Jiang, J.; Chen, K.; et al. Eddy-resolving Simulation of CAS-LICOM3 for Phase 2 of the Ocean Model Intercomparison Project. *Adv. Atmos. Sci.* **2020**, *37*, 1067–1080. 10.1007/s00376-020-0057-z. [CrossRef]
10. He, B.; Yu, Y.; Bao, Q.; Lin, P.F.; Liu, H.L.; Li, J.X.; Wang, L.; Liu, Y.M.; Wu, G.; Chen, K.; et al. CAS FGOALS-f3-L model dataset descriptions for CMIP6 DECK experiments. *Atmos. Ocean. Sci. Lett.* **2020**, *13*, 582–588. [CrossRef]
11. Li, L.; Yu, Y.; Tang, Y.; Lin, P.; Xie, J.; Song, M.; Dong, L.; Zhou, T.; Liu, L.; Wang, L.; et al. The Flexible Global Ocean Atmosphere Land System Model Grid Point Version 3 (FGOALS-g3): Description and Evaluation. *J. Adv. Model. Earth Syst.* **2020**, *12*, 9. [CrossRef]
12. Zhang, H.; Zhang, M.; Jin, J.; Fei, K.; Ji, D.; Wu, C.; Zhu, J.; He, J.; Chai, Z.; Xie, J.; et al. CAS-ESM 2: Description and climate simulation performance of the Chinese Academy of Sciences (CAS) Earth System Model (ESM) version 2. *J. Adv. Model. Earth Syst.* **2020**, *12*, 12. [CrossRef]
13. Craig, A.P.; Vertenstein, M.; Jacob, R. A new flexible coupler for earth system modeling developed for CCSM4 and CESM1. *Int. J. High Perform. Comput. Appl.* **2012**, *26*, 31–42. [CrossRef]
14. Jiang, J.; Wang, T.; Chi, X.; Hao, H.; Wang, Y.; Chen, Y.; Zhang, H. SC-ESAP: A Parallel Application Platform for Earth System Model. *Procedia Comput. Sci.* **2016**, *80*, 1612–1623. [CrossRef]
15. Liu, H.; Lin, P.; Zheng, W.; Luan, Y.; Ma, J.; Ding, M.; Mo, H.; Wan, L.; Ling, T. A global eddy-resolving ocean forecast system in China—LICOM forecast system (LFS). *J. Oper. Oceanogr.* **2021**, 1–13. [CrossRef]
16. Palmer, T. Climate forecasting: Build high-resolution global climate models. *Nat. News* **2014**, *515*, 338–339. 10.1038/515338a. [CrossRef]
17. Bahadur, A.; Iqbal, S.; Shoaib, M.; Saeed, A. Electrochemical study of specially designed graphene-Fe3O4-polyaniline nanocomposite as a high-performance anode for lithium-ion battery. *Dalton Trans. Int. J. Inorg. Chem.* **2018**, *47*, 15031–15037. [CrossRef]
18. Ditta, N.A.; Yaqub, M.; Nadeem, S.; Jamil, S.; Hassan, S.U.; Iqbal, S.; Javed, M.; Elkaeed, E.B.; Alshammari, F.H.; Alwadai, N. Electrochemical Studies of LbL Films With Dawson Type Heteropolyanion Glassy Carbon Electrode Sensor Modified for Methyl Parathion Detection. *Front. Mater.* **2022**, *9*, 877683. [CrossRef]
19. Chi, X.; Hu, Y. The Current Supercomputing Development of China. *Res. World* **2013**, *8*, 56–60.
20. Zhou, S.; Liu, W.; Song, Z.; Yang, X. Code modernization optimization of ocean general circulation model NEMO. *Adv. Mar. Sci.* **2021**, *39*, 62–67. [CrossRef]
21. Mellor, G.L. *User's Guide for a Three Dimentional, Primitive Equation, Numerical Ocean Model*; Program in Atmospheric and Oceanic Sciences: Princeton, NJ, USA, 1998.

22. 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. *Sci. China Inf. Sci.* **2016**, *59*, 072001. [CrossRef]
23. Wu, Q.; Ni, Y.; Huang, X. Regional Ocean Model Parallel Optimization in "Sunway TaihuLight". *J. Comput. Res. Dev.* **2019**, *56*, 1556–1566. [CrossRef]
24. Zhang, H.; Lin, Z.; Zeng, Q. The computational scheme and the test for dynamical framework of IAP AGCM-4. *Chinese J. Atmos. Sci.* **2009**, *33*, 1267–1285.
25. Cao, H.; Yuan, L.; Zhang, H.; Zhang, Y.; Wu, B.; Li, K.; Li, S.; Zhang, M.; Lu, P.; Xiao, J. AGCM-3DLF: Accelerating Atmospheric General Circulation Model via 3D Parallelization and Leap-Format. *Distrib. Parallel Clust. Comput.* **2021**, *14*, 8. [CrossRef]
26. Jiang, J.; Lin, P.; Wang, J.; Liu, H.; Chi, X.; Hao, H.; Wang, Y.; Wang, W.; Zhang, L. Porting LASG/IAP Climate System Ocean Model to Gpus Using OpenAcc. *IEEE Access* **2019**, *7*, 154490–154501. [CrossRef]
27. Wei, J.; Jiang, J.; Liu, H.; Zhang, F.; Lin, P.; Wang, P.; Yu, Y.; Chi, X.; Zhao, L.; Ding, M. LICOM3-CUDA: A GPU version of LASG/IAP climate system ocean model version 3 based on CUDA. *J. Supercomput.* **2023**, 1–31. [CrossRef]
28. 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 LASG/IAP Climate System Ocean Model version 3 (LICOM3) under the heterogeneous-compute interface for portability (HIP) framework and its large-scale application. *Geosci. Model Dev.* **2021**, *14*, 2781–2799. [CrossRef]
29. Blumberg, A.F.; Mellor, G.L. Three-Dimensional Coastal Ocean Models. *Coast. Estuar. Sci.* **1987**, *32*, 1–16.
30. Yu, R. A two-step shape-preserving advection scheme. *Adv. Atmos. Sci.* **1994**, *11*, 479–490.
31. Canuto, V.M.; Howard, A.; Cheng, Y.; Dubovikov, M.S. Ocean turbulence. Part I: One-point closure modelmomentum and heat vertical diffusivities. *J. Phys. Oceanogr.* **2001**, *31*, 1413–1426. [CrossRef]
32. Gent, P.; Mcwilliams, J.C. Isopycnal mixing in ocean circulation models. *J. Phys. Oceanogr.* **1990**, *20*, 150–155. [CrossRef]
33. Large, W.; Yeager, S. *Diurnal to Decadal Global Forcing for Ocean and Sea-Ice Models: The Data Sets and Flux Climatologies*; NCAR/TN-460+STR; OpenSky Press: Austin, TX, USA, 2004; pp. 1–105.