Ocean Dynamics (2010) 60:331–339
DOI 10.1007/s10236-010-0274-6

Parallelization of a coupled wave-circulation model and its application

Guansuo Wang · Fangli Qiao · Changshui Xia

Received: 5 August 2009 / Accepted: 2 February 2010 / Published online: 9 March 2010
© Springer-Verlag 2010

Abstract Using the technique of Message Passing Interface, a parallelized version of a coupled wave-circulation model was set up. The tested model is a regional one for simulating the seas off China, which is comprised of 450,625 elements and 30 vertical sigma layers. The implementation efficiency was evaluated on two kinds of computers, the HP Integrity Superdome and SGI Altix 4700 multiprocessor. The numerical results show that the low-communication high-extra-computation scheme (LCHC) produces higher efficiency than the high-communication no-extra-computation scheme (HCNC) while the number of processors exceeds 24 for HP Integrity Superdome and eight for SGI Altix 4700, respectively. The experiments with both LCHC and HCNC scheme show super-linear speed-up when the number of processors is small. The model with the LCHC scheme is preferred as it achieves parallel efficiency in excess of 90% on the HP machines for all experiments with the number of processors no more than 100, while the efficiency decreases rapidly with the HCNC scheme after the number of processors increases to more than 24. Numerical results suggest that the parallelization of this coupled wave-circulation model is efficient and portable to a variety of parallel architectures.

Responsible Editor: Yasumasa Miyazawa

G. Wang (✉) · F. Qiao · C. Xia
The First Institute of Oceanography,
State Oceanic Administration (SOA),
Qingdao, China
e-mail: wesleyguansuowang@gmail.com

G. Wang · F. Qiao · C. Xia
Key Laboratory of Marine Science
and Numerical Modeling (MASNUM), SOA,
Qingdao, China

Keywords MPI · SPMD · Extra computation · Communication

1 Introduction

It is noted that the vertical mixing induced by surface wave motion plays a key role in the formation of the upper mixed layer in spring and summer (Qiao et al. 2004a, b). Matsuno et al. (2006) measured the vertical mixing in the East China Sea and found that the distribution of vertical mixing is in accordance with the wave-induced vertical mixing scheme suggested by Qiao et al. (2004a, b). A coupled wave-circulation model was developed in this study to verify and test the wave mixing scheme. The coupled model was comprised of the Princeton Ocean Model (POM) and the Key Laboratory of Marine Science and Numerical Modeling (MASNUM) wave number spectral model (Yuan et al. 1991; Yang et al. 2005). The MASNUM wave model has been parallelized by Wang et al. (2007).

Recently, an effort has been undertaken to apply this coupled wave-circulation model to study high-resolution circulations over decadal time scales and to develop a forecasting system for China's coastal regions. These applications will demand significant computational resources because of the high-grid density required to resolve fine-scale physical processes in the coastal area. However, operational integration of a forecasting system faces a time constraint. The runtime for these high-resolution models on modern desktop machines can be extremely long. A parallelization strategy for POM based on widely used message passing interface (MPI) technique is, therefore, needed.

Many global and regional ocean models, including Miami Isopycnic Coordinate Ocean Model (Bleck et al. 1995), Regional Ocean Model System (Wang et al. 2005),

the Modular Ocean Model (Beare and Stevens 1997), and more recently, the Finite Volume Coastal Ocean Model (Cowles 2008), support parallelization implementations. Several parallel versions of the POM model have already existed, e.g., pPOM (Giunta et al. 2007) and MP-POM (Oberpriller et al. 1998). Boukas et al. (1999) successfully tested and validated a parallel implementation of POM using the PVM message passing library. Sannino et al. (2001) compared the pure MPI version and the hybrid MPI+Open-MP one to demonstrate the potential performance of Parallel Hybrid-POM. The coupled parallel atmosphere model (5th-Generation Mesoscale Meteorological Model (MM5)) and ocean model (POM) were developed and verified using the Linux Belwulf system (Kim and Yamashita 2003); the parallel ocean model had an estimated speed-up effect of four using 16 processors. Although tremendous efforts have been made, the parallelization efficiency of POM remains unsatisfactory; for example, the efficiency is 25% with 16 CPUs (from Table 1 of Kim and Yamashita 2003) while it is 65% with four nodes (from Fig. 3 of Sannino et al. 2001). In this study, we adopt a new scheme, the low-communication high-extra-computation scheme (LCHC) (Sawdey et al. 1995; Beare and Stevens 1997) and find that the parallelization efficiency of POM can reach over 90%.

In this paper, POM is briefly presented in Section 2. The new technique used to parallelize the serial code is illustrated in Section 3, and model performance is reported in Section 4. The application of this coupled model in the seas off China is described in Section 5. Conclusion remarks are given in the last section.

2 POM model

The POM is a numerical ocean circulation model developed by Blumberg and Mellor (1987) for both coastal and open ocean studies. To reduce computation time, time splitting technique is used in its algorithms. The external mode of POM is two-dimensional and requires a small time step due to fast gravity waves, while the internal mode is three-dimensional and uses a larger time step. The calculation of the three-dimensional (internal mode) variables (such as temperature, salinity, and horizontal velocity components) is separated into a vertical diffusion time step and an advection plus horizontal diffusion time step. The former is implicit, whereas the latter is explicit.

3 Parallelization of POM

There are many tools to parallelize a serial model like POM. For instance, there is a tool for parallelism using additional zones (Oberpriller et al. 1998) and the PVM message passing library (Boukas et al. 1999). Recently, Giunta et al. (2007) developed a parallel implementation for the POM with a nested-domain feature using the run-time system library and the Fortran loop index converter. Among these tools, OpenMP Application Program Interface (OpenMP API) and MPI are two common and convenient parallel schemes.

By taking advantage of parallelism within algorithms to do many calculations simultaneously, MPI and OpenMP API are powerful at reducing computing time. MPI is a language-independent communications protocol and is often used as a computer language on its own. It is portable on distributed and shared memory machines, but its program is difficult to develop and debug. In contrast, OpenMP is an extension to a compiler, not a computer language. To use OpenMP, programmers only add comments to the program, which are used as hints by the compiler. So, OpenMP is easy to implement. However, it could be only run on shared memory machines. For a parallel coupled model to run efficiently on a large variety of parallel machines, MPI is a good choice.

Sannino et al. (2001) applied the Single Program Multiple Data (SPMD) method in POM based on the hybrid MPI+OpenMP. Here, we describe how parallelization issues have been addressed by using pure MPI in detail. Note that two new parallel schemes were designed for a parallel version of POM to obtain higher performance and be more portable than the previous scheme. We chose the technique of SPMD to write the parallel codes of POM. The basic idea of SPMD is to decompose the computational domain into subdomains and assign each subdomain to a different processor. Each processor integrates the numerical model by a separate computing process using the original code. In this way, only one code needs to be maintained for both sequential and parallel computing platforms. In addition, the sequential code is reused entirely in the parallelization.

3.1 Domain decomposition

According to the SPMD method, we first implement domain decomposition, which is an effective way to divide computational work among processors. This parallel version of POM is structured by employing a two (or one)-dimensional geometric decomposition for the ocean computational grid. In particular, the grid is horizontally partitioned into rectangular blocks evenly across the latitudinal and/or longitudinal dimensions, leaving the vertical dimension unchanged. It is noted that computation in the vertical dimension is difficult to parallelize for the presence of an implicit scheme in the code. In addition, the model ocean (like the real ocean) is thin compared with its horizontal grids. This makes it reasonable to ignore the vertical dimension when parallelizing the code.

The aforementioned partitioning scheme is called a checkerboard or block-striped approach. Each processor is assigned a continuous set of latitudes or longitudes in the block-striped partitioning. Because a checkerboard partitioning splits both latitude and longitude of the domain, no processor is assigned any complete latitude or longitude. A checkerboard-partitioned rectangular mesh maps naturally onto a two-dimensional mesh of processors. POM was designed to run on vector machines; hence, it is easy to maintain load balance on each processor using this method of partitioning.

3.2 Communication

The finite-difference method used for the first-order derivatives in POM requires gradients determined from values stored in neighboring grid points. Each subdomain is then allocated to a single processor, which runs a complete copy of the code, i.e., it is responsible for stepping forward the dynamic ocean variables on the subdomain. Problems occur when stepping forward variables at the perimeter of the subdomain, which requires some variables from the neighboring processors using out-of-bound array subscripts. All these grid points, which are over-dimensioned in the horizontal extent of the local subdomain, form an artificial boundary.

To guarantee consistency of parallel computation with its sequential computation, inter-processor communication is needed. The declaration of arrays can be modified to add extra fake zones, or called halo points. These zones, generally known as overlap areas, contain copies of the artificial boundary values stored on neighboring subdomains (Kernel points) in the grid topology. During model integral, communication among different processors is carried out by means of MPI, a widely used message passing library. Two types of inter-processor communications are used in the model: point-to-point communication to update halo values among processors and global communication to check the stability (CFL) condition and to gather model outputs.

3.3 Parallel scheme

When testing on a cluster of workstations or a supercomputer, it is noticed that a number of bottlenecks contribute to poor parallel efficiency. The frequency of inter-processor communications and extra computations are implemented to improve the performance of the parallel code. The parallel schemes tend to be relatively straightforward; they aim to avoid over-complicating the code with the technical aspects of parallelism. The following describes the parallel schemes used in the parallel version of POM.

3.3.1 High-Communication No-Extra Computation

All subdomain variables were overlapped using a slice of one-grid points. These slices represent halo points containing the copies of the values stored at neighboring kernel points. This means that the variable dimensions were expanded for one-grid point, e.g., the local valid bound of a computation subdomain is larger than or equal to I1 and less than or equal to I2 in longitude, and from J1 to J2 in latitude, where I1, I2, J1, and J2 are valid local indices (I1/I2, J1/J2) for the subdomain. The dimensions of variables were extended one point grid from (I1/I2, J1/J2), and were in the ranges of (I1−1/I2+1) and (J1−1/J2+1). The ranges of computation in each subdomain were not overlapped and are composed a complete domain on a processor.

In a complete time step, when a variable that affords a value for the neighboring halo (artificial boundary) is locally updated, a communication call should be carried out to update the variable halo before it is used. As illustrated in Fig. 1, in the initial implementation, each processor calculates new values for all variables in its own local domain, which is bounded by the halo S0. For each time step, data exchange between adjacent processors must occur. This message passing involves that a processor sends values that have been updated and stored in S0, to adjacent processors, then receives some values from adjacent processors, and stores them in the halo R0.

As indicated in Fig. 2, the 2-D external mode (Loop 8000 in original POM code) and the three-dimensional internal mode (Loop 9000 in original POM code) are the most time-consuming part of POM. The external mode calculation results in update of ocean surface elevation (EL) and the vertically averaged velocities (UA and VA). From the contents in the right box of Fig. 2, the parameters of LHalo and L1 are set to zero in the high-communication no-extra-computation scheme (HCNC) scheme. It is obvious that the value in R0, the halo zones of the intermediate variables (FLUXUA and FLUXVA), must be updated by calling communication subroutine before Loop 410 for producing correct results of ELF (the ocean surface elevation used in the external mode) in Loop 410, because in Loop 405, the intermediate variables are updated within a valid range (I1/I2, J1/J2). The halo elements of the intermediate variables are not refreshed; they are relied upon to calculate the variable of ELF at the perimeter of the subdomain in Loop 410. So a communication event is requested prior to Loop 410. Care should be taken to relate UAF and VAF, which is done in the external mode by six synchronizations. In POM, 50 communications and 28 synchronizations are required to complete a time-step computation. Therefore, this scheme is suitable only for the low-computation high-communication computer system.

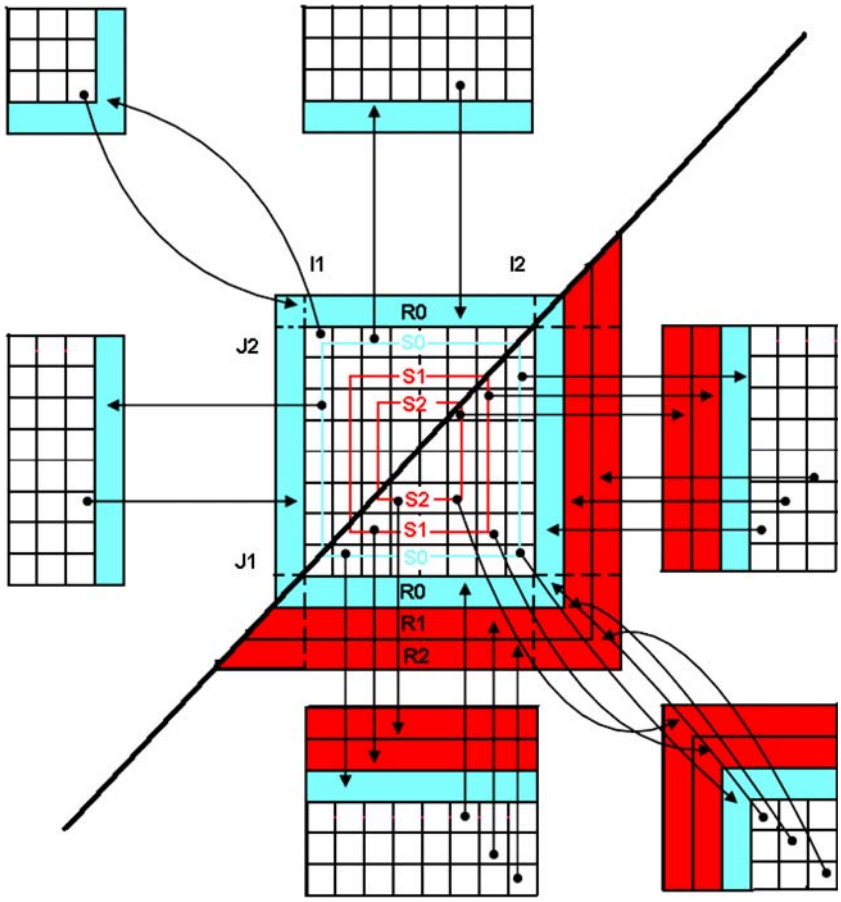

Fig. 1 Schematic diagram of communication among subdomains using an evenly checkerboard decomposition. The *top-left figure* shows the program of communication in HCNC scheme. For calculating the variable at the perimeter of the subdomain without extra computation, an extra halo, R0 (*cyan region*), must be declared for receiving data from the neighboring inner halo, S0 (*region included by cyan rectangle*). The remains clearly explain the communication of the LCHC scheme. In order to reduce the frequency of communication, two additional outer halos, R1 and R2 (*red region*), are declared to store the values that have been sent from the boundary halos, S1 and S2 (*region included by read rectangle*), of adjacent domains. The valid range of longitude in the local domain is from I1 to I2, while J1 and J2 represent the minimal and maximal indices of latitude

3.3.2 Low-Communication High-Extra Computation

On many machines, barrier synchronizations are expensive due to barrier execution overhead and high latency of communication events. Especially on a high-computation performance platform, it is clear that the frequency of synchronization and communication events has a devastating effect on the overall performance of the model. To resolve this problem, a scheme is adopted to reduce communication frequency, trading it for a small amount of extra computation and the cost of some redundant storage (Sawdey et al. 1995; Beare and Stevens 1997), which is called an additional number of fake halos.

In determining the halo width, there must be a tradeoff between communication and computation. If a large halo width is used, the parallel regions are larger, more computation occurs between synchronization and communication events, fewer synchronization are necessary—the messages are larger but fewer are sent, and increased redundant computations will be performed. Thus, increasing the halo width can decrease the number of communication and synchronization events, which, in turn, can increase the efficiency of the parallel program.

We now consider the case of adding two extra halos, R1 and R2 (red region), based on the extra halo R0, to enlarge the parallel region and reduce the frequency of communication (Fig. 1). When message passing occurs, data from halos S0–S3 are packed and sent to adjacent processors, while the corresponding message from adjacent processors is received and stored in the halos of R0–R2. The most important requirement for producing correct results from a parallel program with large halo areas is the correctness of reading data from the overlap areas. For the parallel program to obtain correct results, data read from R0–R2 must be identical to the data storing in corresponding S0–S2 on the neighboring processors. If we allow halo elements to be filled or refreshed by communication only with neighboring processors, then we simply need to ensure that the correct communication events are generated, which limits the size of the parallel regions and increases communication. Allowing overlap elements to be refreshed by computation could avoid these potential problems.

By the description of the right box in Fig. 2, once all used variables adding extra halo zones (R1 and R2) are refreshed by communication at the beginning of the external mode, each processor could calculate the intermediate variables (FLUXUA, FLUXVA) within the domain bounded by R1, namely the calculating range of longitude from (I1−2) to (I2+2) and of latitude between (J1−2) and (J2+2). Hence, the elements in the R1 and R0 halos are already up to date and no more message passing is required

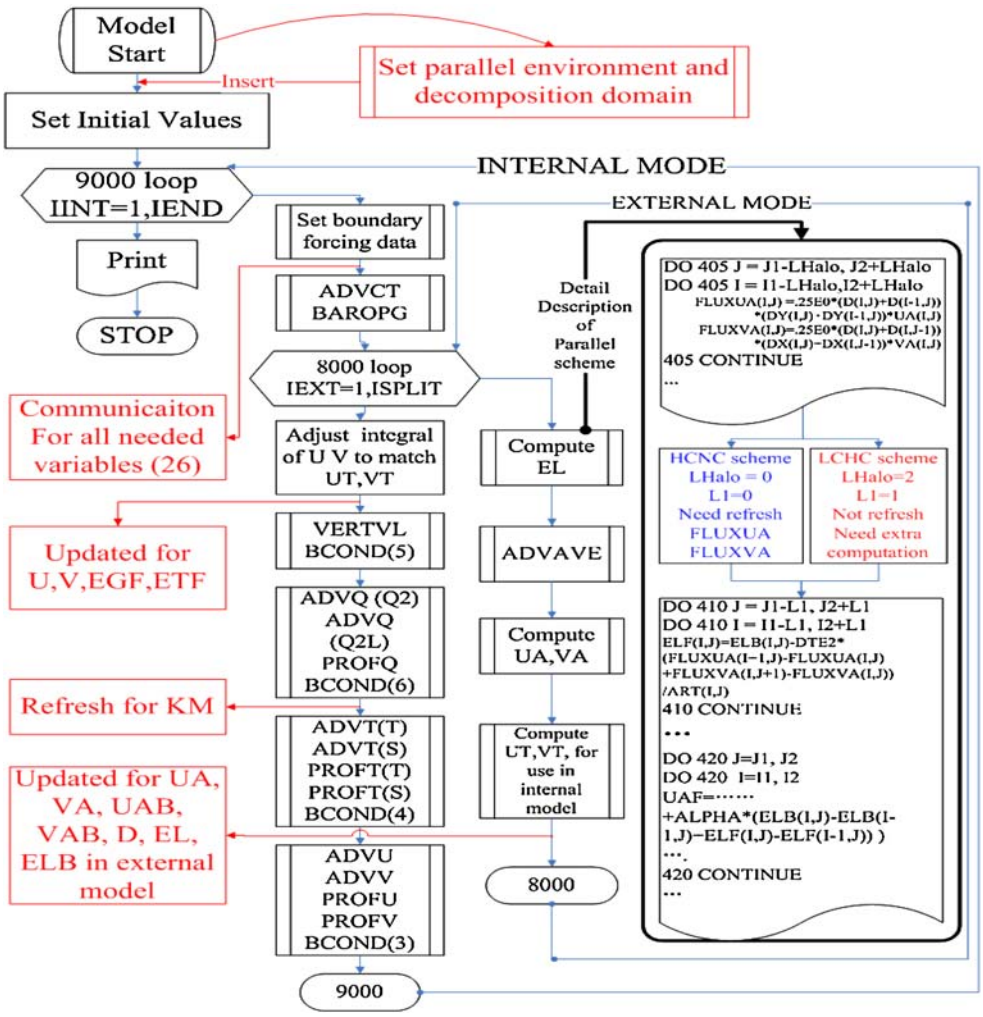

Fig. 2 Flow structure of the parallel POM code. The *black boxes*, except for the content in the *right box*, show the main flow diagram of the serial POM code. The *boxes with sidebars* contain the names of subroutines. The *red boxes* show schematically where the parallel sections (communication and set parallel environment and decomposition functions) are inserted in the LCHC scheme. The *big black box on the right* is a detailed description of the LCHC and HCNC schemes

for calculating the new values of ELF in the domain bounded by R0 (I1−1/I2+1, J1−1/J2+1). Intermediate variables (FLUXUA, FLUXVA) have updated their halo data by this extra computation. By computing in Loop 410, the halo (R0) elements of ELF are refreshed. When calculating the UAF and VAF in Loops 420, 425, 430, and 435, the parallel program can obtain correct results without extra communication, because the halo values of ELF updated by extra computing are identical to the values that the halo elements shadow on the neighboring processors. At the end of the external mode, communications are called to update the halo variables and to prepare for the next step (Fig. 2). This scheme decreases the number of synchronizations in the external mode to one every time step and reduces the relatively expensive latency time. The number of synchronizations was decreased from 28 to four.

Because each halo element corresponds to a valid element on a different processor, refreshing halo elements by computation leads to redundant computation. When a processor computes a value to fill a halo element, the same computation is being done by another processor to fill the corresponding valid element. If the cost of computing the halo element locally is less than that of getting the value from the valid element on a different processor, then

Table 1 Computation time of four experiments

Elapsed time (s)	Number of processors							
	1	8	16	24	32	64	80	100
Exp. 1	18,795	3,075	1,892	1,598	1,218	668	566	
Exp. 2	18,795	3,039	1,816	1,486	1,091	516	414	
Exp. 3	31,497	3,760	2,018	1,386	1,065	646	540	471
Exp. 4	31,497	3,997	2,051	1,395	1,033	517	405	335

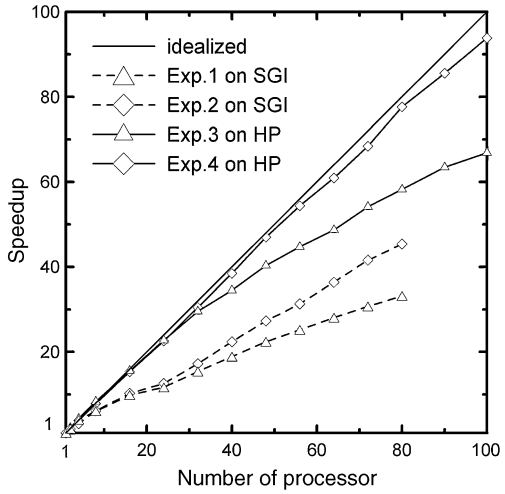

Fig. 3 Speed-up rates versus the number of processors for the parallel codes. Results using the HCNC scheme (*diamonds*; used in Exps. 2 and 4) and LCHC scheme (*triangles*; used in Exps. 1 and 3). Experiments 1 and 2 (*dashed lines*) were loaded on the SGI computer; Exps. 3 and 4 (*solid lines*) were completed on the HP computer system. The speed-up has been computed as T_1/T_p, where T_1 is the total computing time using one processor, and T_p, using multiple processors. As a reference, the ideal speed-up relation is also plotted (*black line with no symbols*)

refreshing the halo by computation will increase the computational performance.

4 Performance evaluation

The aforementioned domain decomposition and parallel schemes have been successfully tested and validated in shared memory architectures. Performance results presented in this paper were obtained on two computer systems: HP 9000 Superdome Server and SGI Altix 4700 Server. The HP 9000 Superdome Server is an ideal steppingstone to HP Integrity servers based on the 64-bit 1.6 GHz Intel Itanium2 Processor. It provides 128 Montecito cores with 18 MB L3 cache and has 256 GB of shared memory. The SGI Altix 4700 Server is powered by a total of 42 Dual-Core Intel Itanium Processors (64-bit 1.6 GHz, with 8 MB L3 Cache) and 84 GB shared memory. The main difference between these two types of servers is that HP 9000 Server has larger L3 cache than that of SGI Altix 4700 Server. For performance studies, we used the Intel Fortran compiler and MPICH on SGI. On the HP machine, an HP Fortran compiler and HP MPI were used. The Fortran compiler and MPI code installed in SGI and HP machines are the other main differences.

The model used here as a testing bed was a regional model for seas off China, containing 30 vertical sigma layers and with horizontal grid points of 721×625. The horizontal resolution is (1/24)×(1/24)°. The external and internal time steps were 4.14 and 124.2 s, respectively. Four experiments were carried out to estimate the parallel performance on different platforms (Exps. 1 and 2 ran on SGI, while Exps. 3 and 4 ran on HP). Exps. 1 and 3 used the HCNC scheme, while Exps. 2 and 4 used the LCHC scheme.

Table 1 presents some detailed results of the regional model for 3-day integration, taking 2,261 time steps

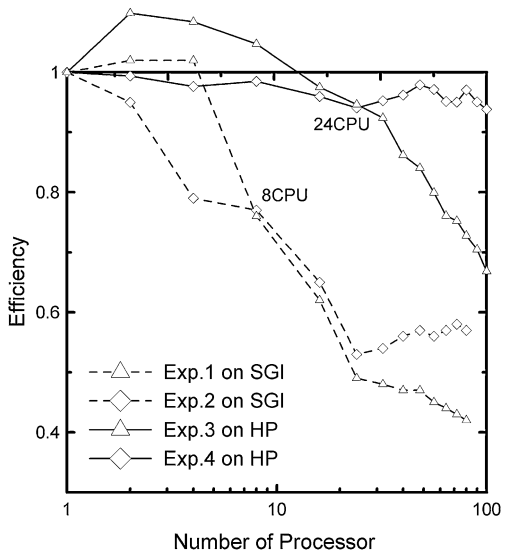

Fig. 4 Efficiency versus the number of processors for the parallel codes. Results using the HCNC scheme (*diamonds*; used in Exps. 2 and 4) and LCHC scheme (*triangles*; used in Exps. 1 and 3). Experiments 1 and 2 (*dashed lines*) were loaded on the SGI computer; Exps. 3 and 4 (*solid lines*) were completed on the HP computer system

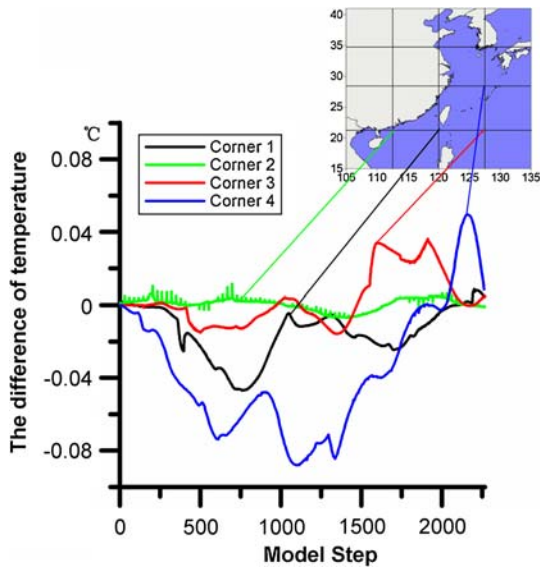

Fig. 5 Difference evolution between serial and parallel (adopting the HCNC scheme) results of SST at the four different overlap corner points during 2,264 model time steps. *Black line* is the difference result of the *lower-left corner point*. *Red*, *blue*, and *green lines*, respectively, show the error evolution of the *lower-right*, *top-right*, and *top-left corner points*. The *inset* describes the decomposition of the model domain

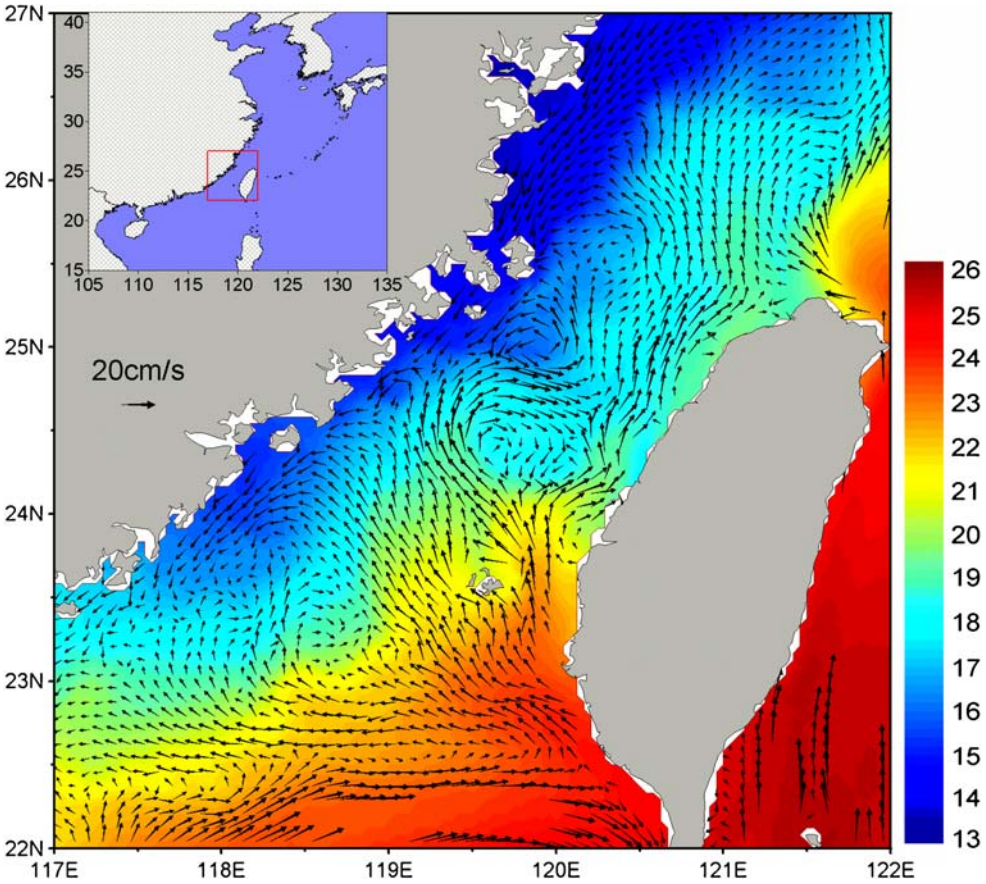

Fig. 6 Simulated sea-surface currents (cm/s) and SST (°C) for February 4, 2008 in the Taiwan Strait from the high-resolution coupled model. The *small inset* represents the model domain of (15–41° N, 105–135° E). The *red rectangle* is the enlarged area

without IO operation. Times were measured in seconds in all cases. The sequential code on the SGI completed the 3-day integration in 18,795 s of elapsed time; while the sequential model running on HP took 31,497 s. The sequential code was, therefore, found to execute 1.68 times faster on the SGI than that on the HP. The Intel Compiler may contribute to the higher computing speed of SGI. It was found that the elapsed time decreased with the number of processors. The HCNC scheme (Exp. 1) produced higher parallel performance than the LCHC scheme (Exp. 2) when the number of processors was less than 8. The same was true for Exps. 3 and 4. The elapsed time shows that the LCHC scheme (Exp. 4) had a significant advantage when the number of processors was more than 24 (Table 1); otherwise, the HCNC scheme was preferred. In Exp. 4, the number of processors was increased to 100, and the 3-day run was completed in 335 s of wall clock time, just 70% of the elapsed time of Exp. 3. The wall clock elapsed time for Exp. 2 was only 73% of that for Exp. 1 as the number of processors reached 80.

Figures 3 and 4, respectively, display the speed-up rate and parallel efficiency of the four experiments. As can be seen from Fig. 4 (Exps. 1 and 3), efficiency exhibits a super-linear behavior (more than 100%) when the number of CPUs is not large. This kind of behavior is common in cache-based machines. As the number of processors increases, the size of the subdomain mapped to each processor decreases, and more subdomains fit the cache. Figure 3 shows that the speed-up of the computation (Exp. 4) remains nearly linear, with a maximum speed-up of 93.8 for 100 processors. The speed-up of Exp. 3 remains linear when the number of processors is less than 24; its speed-up quickly departures from the realized line when the number exceeds 24. Figure 4 also shows that with 100 processors, the efficiency of the HCNC scheme on HP quickly drops to 66.9%, while the efficiency of the LCHC scheme on HP remains more than 93.0%. It is obvious that the LCHC scheme is to be preferred when the number of processors becomes more than 24 on a HP system. We note that the speedup curves of Exps. 1 and 2 present same shape and quickly departure the ideal line (Fig. 3). It is obvious that the LCHC scheme gains an advantage over the HCNC scheme on an SGI system while the number of the processors is larger than eight. The efficiency of the LCHC scheme remains 50% higher with the increase of the processors. In contrast, the efficiency of HCNC quickly dropped to 42% as the number of processors reaches 80. Therefore, the LCHC scheme is the preferred one when the number of the processors is large.

For comparison with the efficiency reported by Sannino et al. (2001), we chose the elapsed time of 16 processors as a reference time. The parallel efficiency of the LCHC scheme is 87% and 93% on SGL and HP machines,

respectively, while the efficiency of the parallel code using MPI and OpenMP standard by Sannino et al. (2001) is about 65%, even less than the parallel efficiency of our HCNC scheme (>70%).

5 Application

The hindcast and forecast computations using the super-high-resolution model were feasible by the success of the code parallelization effort. A fine-resolution regional hindcast system for the seas off China has been completed using the parallelized coupled wave-circulation model. The wave and ocean circulation models used in this hindcast system are the parallel version of the MASNUM wave model (Yuan et al. 1991; Yang et al. 2005; Wang et al. 2007) and the parallelized POM. It should be mentioned that there are still tiny differences between the serial model and the parallel model in the overlap regions (four corners of each local domain); for example, the difference of sea surface temperature (SST) is less than 0.1°C (Fig. 5). This most likely is due to the truncation error in the overlap region during communication. The lateral boundary conditions (of temperature, salinity, sea level, and velocities) for this regional model come from the global 0.5×0.5° model results. The regional model was integrated for 6 year, driven by wind stress and heat fluxes from the comprehensive ocean–atmosphere dataset. Then, 1-year simulation is implemented by using surface winds and heat fluxes from a configuration of the MM5. Nudging-based data assimilation techniques are used to incorporate satellite data of SST. With the use of 32 processors of the SGI Altix 4700, a 1-year hindcast of this model without IO operation requires around 42 h of wall clock time. SST and sea-surface current vectors for February 2008 are shown in Fig. 6.

A forecast system for the seas off China is developed and run since October 2007 based on the paralleled coupled wave-circulation model. This forecast system completes each 3-day forecast run in 2,440 s of elapsed time with output at every 6 h, by using 24 processors of the SGI Altix 3700.

6 Conclusion

All previous efforts, especially those by Sannino et al. (2001) and Kim and Yamashita (2003) using MPI and OpenMP standards, are quite important for the speed-up of POM. Here, two new parallel schemes of LCHC and HCNC are designed to improve the performance of parallel POM for a large variety of parallel machines.

Two efficient parallelization schemes were designed and implemented for the coupled wave-circulation model. Our numerical experiments show that the HCNC scheme is preferred on these two computer systems when the number of processors is less than 24 (HP) and eight (SGI). In principle, the parallel code is able to implement super-high-resolution simulation with high efficiency. The efficiency, measured on two modern multiprocessor computer systems, shows good performance for a representative application. The parallel code is suitable to run efficiently on a large variety of parallel machines based on the MPI standard interface. In the quite near future, such as within 2 months, we would like to release the code with the wave-induced vertical mixing scheme and parallelization.

Acknowledgments We would like to acknowledge the developers of MPICH and HP-MPI code (MPICH web: http://www.mcs.anl.gov/research/projects/mpi; The HP-MPI web: http://h21007.www2.hp.com/portal/site/dspp). The lead author would like to thank his English tutor, Mr. Salmon, for revising the manuscript. This study was supported by the key project of National Natural Science Foundation of China (grant no. 40730842), the Foundation of the Social Commonwealth Study of the First Institute of Oceanography (GY02-2008G37 and GY02-2009T04).

References

Beare MI, Stevens DP (1997) Optimisation of a parallel ocean general circulation model. Ann Geophysicae 15:1369–1377

Bleck R, Dean S, O'Keefe M, Sawdey A (1995) A comparison of data-parallel and message-passing versions of the Miami isopycnic coordinate ocean model. Parallel Comput 21:1695–1720

Blumberg AF, Mellor GL (1987) A description of a three-dimensional coastal ocean circulation model. In: Heaps N (ed) Three-dimensional coastal ocean models, vol 4. AGU, Washington DC, pp 1–16

Boukas LA, Mimikou NT, Missirlis NM, Mellor GL, Lascaratos A, Korres G (1999) The parallelization of the Princeton ocean model. Lect Notes Comput Sci 1685:1395–1402

Cowles GW (2008) Parallelization of the FVCOM coastal ocean model. Int J High Perform Comput Appl 22:177–193

Giunta G, Mariani P, Montella R, Riccio A (2007) pPOM: A nested, scalable, parallel and Fortran 90 implementation of the Princeton Ocean Model. Environ Model Softw 22:117–122

Kim K, Yamashita T (2003) Parallel computation of coupled atmosphere and ocean model—the case study of typhoon 9918 in the Yatsushiro sea. Disaster Prevention Research Institute Annuals, Kyoto University 46B:627–636

Matsuno T, Lee JS, Shimizu M, Kim S, Chan P (2006) Measurements of the turbulent energy dissipation rate e and an evaluation of the dispersion process of the Changjiang Diluted Water in the East China Sea. J Geophys Res 111:C11S09. doi:10.1029/2005JC003196

Oberpriller WD, Sawdey AC, O'Keefe MT, Gao S (1998) Paralleling the princeton ocean model using TOPAZ. In: Parallel Computer. Sys. Lab., Dept. Elec. Comp. Eng., University of Minnesota, Tech. Report, pp 21

Qiao F, Ma J, Yang Y, Yuan Y (2004a) Simulation of the temperature and salinity along 36°N in the Yellow Sea with a wave-current coupled model. J Korean Soc Oceanogr 39:35–45

Qiao F, Xia C, Shi J, Ma J, Ge R, Yuan Y (2004b) Seasonal variability of thermocline in the Yellow Sea. Chin J Oceanol Limnol 22:299–305

Sannino G, Artale V, Lanucara P (2001) A hybrid OpenMP-MPI parallelization of the Princeton Ocean Model. In: Parallel computing, advance and current issues, pp 222–229

Sawdey A, O'Keefe M, Bleck R, Numrich RW (1995) The design, implementation and performance of a parallel ocean circulation model. In: Hoffman G, Kreitz N (eds) Proceedings of the Sixth ECMWF Workshop on the use of Parallel Processing in Meteorology, London, pp 523–548

Wang P, Song Y, Chao Y, Zhang H (2005) Parallel computation of the regional ocean modeling system. Int J High Perform Comput Appl 19:375–385

Wang G, Qiao F, Yang Y (2007) Study on parallel algorithm for MPI-based LAGFD-WAM numerical wave model. Advance in Marine Science 25:401–407 (In Chinese)

Yang Y, Qiao F, Zhao W, Teng Y, Yuan Y (2005) MASNUM ocean wave numerical model in spherical coordinates and its application. Acta Oceanol Sin 22:1–7

Yuan Y, Pan Z, Hua F, Sun L (1991) LAGDF-WAM numerical wave model. Acta Oceanol Sin 10:483–488