A new parallelization algorithm of ocean model with explicit scheme

X D Fu
College of Information Science and Engineering, Ocean University of China, Qindao
266100, China
E-mail: fuxiaodan8087@163.com

Abstract. This paper will focus on the parallelization of ocean model with explicit scheme which is one of the most commonly used schemes in the discretization of governing equation of ocean model. The characteristic of explicit schema is that calculation is simple, and that the value of the given grid point of ocean model depends on the grid point at the previous time step, which means that one doesn’t need to solve sparse linear equations in the process of solving the governing equation of the ocean model. Aiming at characteristics of the explicit scheme, this paper designs a parallel algorithm named halo cells update with tiny modification of original ocean model and little change of space step and time step of the original ocean model, which can parallelize ocean model by designing transmission module between sub-domains. This paper takes the GRGO for an example to implement the parallelization of GRGO (Global Reduced Gravity Ocean model) with halo update. The result demonstrates that the higher speedup can be achieved at different problem size.

1. Introduction
A branch of modern natural science is oceanic and atmospheric science which involves a lot of ocean-atmosphere coupling models, such as POM (Princeton Ocean Model), ROMS (Regional Ocean Modeling System), FVCOM (Finite Volume Coastal Ocean Model) and CAM (Community Atmosphere Model). With the progress of computer hardware and software architecture, the high parallel performance computing technology has brought opportunities to improve the efficiency of these models; more and more parallel ocean models were developed, such as parallel versions of POM [1, 2], ROMS [3, 4], FVCOM [5] and CAM [6].

There are various software tools and library for developing parallel ocean model so far, such as HPF (high performance fortran), MPI (message passing interface), OpenMP (open multi-processing) and PVM (Parallel Virtual Machine). HPF is an extension of fortran, which has been chosen for the parallelization of FRAME (Fine Resolution AMmonia Exchange) [7]. MPI is a parallel software library and uses message transmission to communicate between parallel processes, which is one of the most commonly used parallel tools for the parallelization of ocean model, for example, the parallelization of FVCOM [5]. OpenMP is a directive-based parallel tool on shared memory platform; and the advantage of OpenMP is that one can quickly translate the serial ocean model to parallel version by means of inserting OpenMP parallel directive into the original serial ocean model [8, 9]. In order to make merit of MPI and OpenMP, Osthoff [10] applied hybrid MPI/OpenMP to parallelize the OLAM (Ocean-land-atmosphere model) model, the results demonstrated that the hybrid version decreases the number of output files and results in better performance for IO operations.
computing nodes of heterogeneous environment to parallel processors, which masks the differences between computing nodes, and its application in parallel ocean model was demonstrated in reference [11]. However, all these parallel techniques chosen for parallelization are based on developing environment, and are not based on the characteristic of the discretization of the governing equation of ocean models. This paper develops a common parallel algorithm for ocean model with explicit discretization of governing equation.

The time discretization of finite difference scheme of the governing equation of the ocean model is divided into explicit and implicit scheme. The selection of a difference scheme is related to the stability and convergence of the governing equations, and is also related to the selection of parallel technique in the parallelization of ocean model to a certain extent. When the governing equation is discretized with implicit scheme, large sparse linear system is produced, in which case, one can design parallel algorithm aiming at the characteristic of the linear system [12]. For example, when the governing equation is discretized with ADI scheme which produces tri-diagonal linear system, one can parallel design parallel algorithm to parallelize the tri-diagonal linear system [13, 14].

This paper will focus on the parallelization of ocean model with explicit scheme. Aiming at the characteristic of explicit scheme, this paper proposes a new parallel algorithm named halo cell update with tiny modification of the original ocean model. In order to test the performance of the halo cell update algorithm, this paper applies the parallel algorithm to parallelize the Global Reduced Gravity Ocean model under different problem size of ocean model.

2. Halo cell update

Applying the halo cell update to parallelize ocean model, the main concept of the parallel method is “divide and conquer”, which contains three steps as follows.

At the first stage, we divide the whole domain involved in the ocean model into sub-domains. Each sub-domain has its own boundary. In this paper, the inner grids which are close to the boundary are called inner boundary. Each sub-domain expands its boundary, which forms outer boundary called halo cells. The halo cells can be set boundary condition used to compute the inner grids.

At the second stage, at each integration step, the inner grids of each sub-domain can be computed according to boundary condition which are saved in halo cells, in which case, the sub-domain can be computed by each processor in a parallel manner, and the results of the inner grids that we get is correct, this is mainly because the inner grids is calculated with the correctly boundary condition saved in the halo cells.

At the third stage, in order to compute correctly at the next time step each sub-domain sends its inner boundary and receives its halo cells from other sub-domains, which is called update step. It is obvious that after the update step the results of the parallel algorithm is identical to the results of the serial algorithm and we can apply the same solver to all sub-domains.

In order to understand the halo cell update correctly, let us consider the horizontal velocity $u$.

$$u_{i}^{n+1} = F(u_{i-1}^{n}, u_{i}^{n}, u_{i+1}^{n}), \quad i = 1, 2, ..., m, \quad u_{0}^{n+1}, u_{m+1}^{n+1} = g(t)$$

where $u$ is the horizontal velocity, $i$ is the number of elements in the total domain, $n$ represents the time step, $g(t)$ is the boundary condition and $F()$ is the discretization form of the governing equation of the ocean model. We divide the entire domain into two sub-domains, $u_{i}, (i=l, ..., p)$, and $u_{2,i}, (i=p+1, ..., m)$. Let us apply the halo cell update algorithm for these two sub-domains. These two sub-domains expand their boundaries, forming halo regions. We set the halo regions with the boundary condition: $u_{1,p+1} = u_{2,p+1}, u_{2,p} = u_{1,p}$. Now these two sub-domains have their own boundaries, and can be computed independently by different processors. That is

$$u_{1,i}^{n+1} = F(u_{i-1}^{n}, u_{1,i}^{n}, u_{i+1}^{n}), \quad i = 1, 2, ..., p, \quad u_{1,p+1}^{n+1} = u_{2,p+1}^{n+1}$$
$$u_{2,i}^{n+1} = F(u_{2,i-1}^{n}, u_{2,i}^{n}, u_{2,i+1}^{n}), \quad i = p + 1, ..., m, \quad u_{2,p}^{n+1} = u_{1,p}^{n}$$
At the update stage, the halo region of these two sub-domains must be refreshed; this is because we must use the updated halo region to compute the inner grids at the next time step. In this stage, the update function can be implemented by the basic message passing function:

\[ u_{1,p+1}^{n+1} = u_{2,p+1}^{n+1}, \quad u_{2,p}^{n+1} = u_{1,p}^{n+1} \]

The update function can be implemented by the use of basic MPI point-to-point communication operations, such as mpi_send and mpi_recv subroutines. The advantage of the halo cell update is as follows:

- This parallel algorithm is simple, i.e., only one update function written in forms of basic message communication subroutines is required to refresh halo cell, and it is clear that the result of the parallel algorithm is identical to the result of the serial algorithm.
- The parallel algorithm has certain extensibility, i.e. we can expand two halo cells to reduce the times of halo cell update, in which case, the times of halo cell update will reduce to a half at the cost of redundant computing of each sub-domain, as is shown in figure 1.

![Figure 1. The update of halo cells.](image)

### 3. Ocean model

The ocean model used in this paper is an extended 1.5 layer Global Reduced Gravity Ocean model (GRGO) [15, 16], which was first introduced by Cane to study ENSO. In this model, mass, momentum, and heat obey the conservation laws in the upper layer, and the lower layer is assumed to be infinitely deep and motionless to keep the kinetic energy finite, so the momentum, continuity, and thermodynamic equations for the upper can be written as:

\[
\frac{\partial U}{\partial t} + U \cdot \nabla U + f K \cdot U - \frac{1}{2} h \nabla b + \nabla (h b) = \frac{\tau}{h} + \mu \nabla^2 U - \frac{1}{h} w_e H(w_e) U
\]  
\[
\frac{\partial h}{\partial t} + \nabla \cdot (h U) = w_e
\]  
\[
\frac{\partial T}{\partial t} + U \cdot \nabla T = \frac{Q}{\rho_0 C_P h} + \kappa \nabla^2 T - \frac{1}{h} \frac{w_e H(w_e)}{h} (T - T_r)
\]

where \( U = U(x, y, t), h = h(x, y, t), T = T(x, y, t) \) represent the horizontal velocity vector, thickness of upper layer and surface temperature respectively; \( x, y \) represent horizontal coordinates; \( w_e \) represents entrainment velocity at the base of mixed layer; \( \tau \) represents wind stress vector; \( b \) represents buoyancy in the upper layer; \( T_r \) represents reference temperature, \( T \) represents the surface temperature; \( Q \) represents heat flux; \( \kappa \) represents the diffusivity coefficient; \( \rho_0 \) is the density; \( C_P \) represents thermal expansion coefficient; \( \mu \) represents horizontal viscosity coefficient.
The GRGO is about to solve $u, v, h, T$ by the integral of time according to the boundary condition, so in these equations, we mainly solve $\frac{\partial}{\partial t}$. The main idea of this paper is the domain decomposition in the horizontal directions; we are interested in the discretization of the terms containing $\frac{\partial}{\partial x} \frac{\partial^2}{\partial x^2} \frac{\partial}{\partial y} \frac{\partial^2}{\partial y^2}$ in the above equations.

4. Discretization
This section will deal with the discretization of governing equations of the ocean model. We discretize the governing equations with explicit scheme which is relatively simple to set up and to program in order to find the numerical solutions of the equations. In order to keep the numerical solution stable and avoid generating oscillations, we must choose appropriate time step and spatial step to meet the CFL (Courant–Friedrichs–Lewy) condition. In computational fluid dynamics, the CFL is a necessary condition for the stable while solving certain partial differential equation. In this paper, we choose $\frac{dt}{dx} < 0.1$ (This paper focuses on a two dimensional system). Take the momentum equation (1) and the continuity equation (2) for examples, and the finite difference equations are as follows:

$$\frac{v^{n+1}_{i,j} - v^n_{i,j}}{\Delta t} + u^n_{i,j} \frac{u^n_{i,j} - u^n_{i-1,j}}{\Delta x} + v^n_{i,j} \frac{u^n_{i,j} - u^n_{i,j-1}}{\Delta y} - f v^n_{i,j} = \frac{1}{\tau_x} h^n_{i,j} \frac{b^n_{i,j} - b^n_{i-1,j}}{\Delta x} + \frac{\partial}{\partial y} \left( \frac{\partial}{\partial x} \frac{\partial}{\partial x} \frac{\partial}{\partial y} \right)$$

For example, from equation (4) and figure 2(a), when we compute the $u$ at $n + 1$ time, we must use $u$ and the four points around $u$ of $n$ time. From equation (5) and figure 2(b), when we compute the $h$ at $n + 1$ time, we use the $h_{i,j}$, $h_{i,j-1}$ and $h_{i+1,j}$ of $n$ time. We can decide the schema of the domain decomposition according to the relationships of the grids.

5. Result and analysis
The aim of the parallelization of the ocean model is to reduce the runtime and increase the efficiency.
of the ocean model. This paper tests the parallel version of GRGO in the following environments:

Platform 1, the number of CPU cores is 4, the memory size is 4 GB, and the operating system is Windows 7.

Platform 2, the number of CPU cores is 8, the memory size is 8 GB, and the operating system is Windows Server.

Platform 3, the number of CPU cores is 16, the memory size is 16 GB, and the operating system is Linux.

This paper uses different kinds of resolution of the ocean model to test the performance of the parallel algorithm in the above parallel environment, which means the problem size is different.

The running time of U equation at different problem sizes and different platforms is shown in following tables 1-3.

| Table 1. The running time of U equation at different problem size on platform 1. |
|----------------------------------|----------------------------------|
| CPU cores | Different problem sizes (units: s) |
| {} | 1350*600 | 2700*1200 |
| 1 | 41.32 | 164.95 |
| 2 | 21.15 | 84.23 |
| 3 | 14.77 | 58.04 |
| 4 | 11.52 | 44.63 |

| Table 2. The running time of U equation at different problem size on platform 2. |
|----------------------------------|----------------------------------|
| CPU cores | Different problem sizes (units: s) |
| {} | 1440*480 | 3000*1000 |
| 1 | 61.63 | 268.27 |
| 2 | 32.00 | 147.40 |
| 4 | 25.53 | 103.00 |
| 6 | 19.24 | 80.30 |
| 8 | 16.43 | 69.20 |

| Table 3. The running time of U equation at different problem size on platform 3. |
|----------------------------------|----------------------------------|
| CPU cores | Different problem sizes (units: s) |
| {} | 1350*600 | 3000*1000 | 3600*1600 |
| 1 | 25.23 | 94.28 | 183.17 |
| 2 | 12.86 | 47.89 | 94.94 |
| 4 | 6.72 | 24.07 | 47.23 |
| 10 | 4.97 | 17.40 | 32.51 |
| 16 | 3.56 | 12.12 | 22.34 |

The speedup is an important indicator measuring the performance of parallel algorithm, and is defined as the ratio of serial execution time and parallel execution time. In order to analyse the performance of parallel algorithm, this paper calculates the speedup according to the running time as shown in above tables.

From tables 1-3, the results reveal that the running time reduces as the CPU cores increase on the condition that the problem size is fixed and that the speedup increases, which is shown in figures 3-5.
2700*1200 are 3.59 and 3.70. On platform 2, when the number of CPU cores is 8, the speedups of problem size 1440*480 and 300*1000 are 3.75 and 3.88. On platform 3, when the number of CPU cores is 16, the speedups of problem size 1350*600, 3000*1000 and 3600*1600 are 7.09, 7.78 and 8.20.

![Figure 3](image)

**Figure 3.** The speedup of different problem size on platform 1.

On platform 1, when the number of CPU cores is fixed, we can get higher speedup as the problem size increases, which is shown in figure 3, i.e., the speedup of problem size 2700*1200 is higher than that of problem size 1350*600. On platform 2 and platform 3, we can get similar results that the higher speedup is gained at the bigger problem size when the number of CPU cores is fixed, as shown in figures 4 and 5, i.e., the speedup of problem size 3000*1000 is higher than that of problem size 1440*480; the speedup of problem size 3600*1600 is the highest one in all of the problem sizes 3600*1600, 300*1000 and 1350*600. The reason of above results is as follows: as the number of CPU cores is fixed, the increased amount of calculation is greater than increased amount of communication when the problem size increases, which demonstrate that the halo cell update has scalability.
From tables 1-3 and figures 3-5, the results reveal that the speedup increases as the number of CPU cores increase and the higher speedup is gained at bigger problem size, which demonstrates that the halo cell update algorithm can get better performance in the parallelization of ocean model.

![Figure 5. The speedup of different problem size on platform 3.](image)

6. Conclusion
Aiming at the characteristic of the explicit scheme of the ocean model, this paper develops a parallel algorithm used to parallelize ocean circulation model with explicit scheme. In order to measure the performance of the parallel algorithm, this paper applies this parallel algorithm to parallelize GRGO model. The process of parallelizing ocean model is simple, which means that one needs a bit of modification of original ocean model; in addition, the parallel version of ocean model is scalable. This paper tests parallel version of GRGO in three platforms; and the result reveals that the running time reduces as the number of CPU cores increases, and that with fixed number of CPU cores the speedup increases when the problem size increases, which demonstrates that halo cell update is scalability.

References
[1] Giunta G, Mariani P, Montella R, et al 2007 pPOM: A nested, scalable, parallel and fortran 90 implementation of the princeton ocean model Environ. Modell. Softw. 22 117-22
[2] Jordi A and Wang D P 2012 sPOM: A parallel implementation of princeton ocean model Environ. Modell. Softw. 38 59-61
[3] Wang P, Song Y T, Chao Y, et al 2005 Parallel computation of the regional ocean modeling system Int. J. High Perform. C 19 375-85
[4] Govett M, Hart L, Henderson T, et al 2003 The scalable modeling system: Directive-based code parallelization for distributed and shared memory computers Parallel Comput. 29 995-1020
[5] Cowles G W 2008 Parallelization of the FVCOM coastal ocean model Int. J. High Perform. C 22 177-93
[6] Tseng Y H and Ding C 2008 Efficient parallel i/o in Community Atmosphere Model (CAM) Int. J. High Perform. C 22 206-18
[7] Pais V A, Fournier N, Sutton M A, et al 2004 Using high performance fortran to parallelise a multilayer atmospheric transport model Parallel Comput. 30 21-33
[8] Malfetti P 2001 Application of OpenMP to weather, wave and ocean codes Scientific Programming 9 99-107
[9] Wallcraft A J 2000 SPMD OpenMP versus MPI for ocean models Concurrency: Practice and Experience 12 1155-64
[10] Osthoff C, Grunmann P, Boito F, et al 2011 Improving performance on atmospheric models
through a hybrid OpenMP/MPI implementation 2011 IEEE 9th Int. Symp. On Parallel and Distributed Processing with Applications (Busan, South Korea: IEEE) pp 69-74

[11] Beare M I and Stevens D P 1997 Optimization of a parallel ocean general circulation model Annales Geophysicae vol 15 (Berlin/Heidelberg: Springer-Verlag) pp 1369-77

[12] Jos D K, Aad V D S, Hakan O and Henk D 2002 A fully implicit parallel ocean model using MUMPS J. Supercomput. 23 167-83

[13] Mendiratta K and Polizzi E 2011 A threaded SPIKE algorithm for solving general banded systems Parallel Comput. 37 733-41

[14] Mattor N, Williams T J and Hewett D W 1995 Algorithm for solving tridiagonal matrix problems in parallel Parallel Comput 21 1769-82

[15] Fang Y 2006 A coupled model study of the remote influence of ENSO on tropical Atlantic SST variability (Texas A & M University)

[16] Cane M A 1979 The response of an equatorial ocean to simple wind stress patterns: I- Model formulation and analytic results; II- Numerical results J. Mar. Res. 37 233-99