Ocean Circulation on the Intel Paragon: Modeling and Implementation

Ka-Cheong Leung, Ishfaq Ahmad
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{kcleung, iahmad}@cs.ust.hk

Hsiao-Ming Hsu
National Center for Atmospheric Research
Mesoscale and Microscale Meteorology Division
Boulder, Colorado, U.S.A.
hsu@ash.mmm.ucar.edu

Abstract

In this paper, we present the modeling and implementation of a grand-challenge problem in the field of scientific computation: the Primitive-Equation Numerical Ocean Circulation Model. We present the mathematical formulation of the model and propose a scheme for its parallel implementation. Optimizations are made through collective communications and various partitioning schemes. In our experiments using up to 100 processors on the Intel Paragon parallel computer, the proposed strategy yields an encouraging speedup and exhibits a sustained scalability with increasing both the problem and machine sizes. We consider barotropic continental shelf waves in a periodic channel as a test problem. The model has numerous applications in environmental studies and ocean sciences.

1 Introduction

Massively parallel processors (MPP) are becoming a useful and practical tool for studying a wide range of previously unsolved problems. A class of these problems includes three-dimensional time-dependent numerical models for studying various physical world phenomena. Traditionally, these problems have been solved on vector supercomputers consisting of a few processors. However, the recent trend of scalable design for the modern MPPs has generated a new wave.

Scalability determines whether an application is feasible on a larger parallel computer without sacrificing efficiency. Practically, a program can be made large enough to operate efficiently on a scalable computer with a given granularity. However, scaling a problem may produce no better results because of the non-productive message-passing calls and synchronization among processors. The scalability property of distributed-memory multicomputers which belong to a class of MPPs has made it possible to study large-scale numerical models related to real-world problems, most of which are classified as the so-called *grand-challenge problems.* Although distributed-memory parallel computers are harder to program than their shared-memory counterparts, the advantage of scalability means that a sustained performance is possible while increasing both the problem and machine sizes. This provides new hopes for environmental problems such as prediction of weather, climate and global changes, and ocean sciences.

The objective of this paper is to present the state of our on-going project for studying the ocean circulation. For this purpose, we have developed and implemented an ocean circulation model on the Intel Paragon. The model is fully operational and is intended to be a vehicle for environmental studies.

This paper is organized as follows. Section 2 introduces the primitive-equation numerical ocean circulation model calculated through a second-order finite-difference approximation numerical scheme. Section 3 briefly describes the architecture and the programming environment of the Intel Paragon parallel computer. Section 4 presents our implementation scheme for the ocean model. Section 5 examines the experimental results of the proposed scheme and makes some constructive inferences about the effectiveness of the implementation. Section 6 gives some conclusions and discusses some possible extensions to our work.

2 The Model

Like other branches of geophysical fluid dynamics, analytical solutions to most ocean dynamics problems are not readily obtainable because the problems are nonlinear [7]. Historically, three-dimensional time-dependent numerical models for ocean circulation are based on the numerical approximations to the governing partial differential equations of the ocean dynamics and thermodynamics. Both horizontal and vertical gradients are explicitly represented by various finite-difference methods. The classical model of Bryan [3] is the very first of this kind. Over the last 30 years, this approach has been applied to various ocean numerical models for either basin-scale, regional-scale, or coastal oceans. Due to the explicit nature of the model, different physical processes can be included in a relatively straight-forward manner. Continuous improvements made by many researchers have demonstrated the flexibility and usefulness of this approach. In fact, most of the ocean circulation numerical models currently being used belong to this class.

Because of the long-term nature of the stable stratification in the large-scale ocean, theoreticians simplify the ocean into layers when they try to understand the fundamental mechanisms to drive the ocean circulation. Few numerical models have been built based on this analytic method; for example, see [2]. Even though this approach is conceptually easier than the previous one to be understood, methods to solve the technical difficulty in representing the mixed layer just beneath the ocean surface have only recently been tested [2]. Numerical models for coastal ocean employ mainly the geometric vertical coordinate with a vertical coordinate transformation to accommodate the bottom topography which is considered important and necessary. Some recent developments have been included in [13].

The numerical model used in this study is based on a combination of a finite-difference scheme in horizontal directions and a spectral scheme in the vertical direction. The basic model is described in Haidvogel *et al.* [12]. In fact, they discussed six different examples in channel, coastal, and basin-scale oceans. Obviously, the vertical specification of spectral modes determines the vertical resolution. A transformation algorithm [15] has been developed to overcome this restriction. The model has been applied to coastal studies, such as coastal trapped waves [18], shelf-break fronts [9], [10], eastern boundary current [14], bottom density front [6], coastal up-welling and down-welling [16]. Isolated topography in ocean is another interesting topic. Studies of flow over isolated sea-mount were conducted in [1], [4], [5], and [11]. An investigation of wind forcing over a circular bank was carried out in [8]. The water mass and circulation in the polar region have strong impacts in the change of our climate through the thermohaline circulation between polar, mid-latitude, and tropical oceans. The formations of the wintertime dense water and summertime halocline water have been reported in [15].

Table 1. List of symbols used in the Ocean Circulation Model.

f	Coriolis parameter
g	gravitational acceleration
h	bottom topographic height
m, n	scale factors in the horizontal curvilinear coordinate transformation time
t	time
q	horizontally averaged vorticity
R_u	total forcing terms in equation (1)
R_v	total forcing terms in equation (2)
u, v	horizontal components of velocity
w	vertical velocity in z-coordinate
x, y, z	Cartesian coordinate
ξ	horizontally transformed x-coordinate
η	horizontally transformed y-coordinate
σ	vertically transformed z-coordinate
ϕ	dynamic pressure (pressure/density)
ρ	density
ν	viscous coefficient for momentum
κ	diffusive coefficient for density
ψ	vertically averaging stream-function
Ω	vertical velocity in σ-coordinate

The model is based on a set of primitive equations in time and in the three-dimensional Euclidean space. Table 1 includes the list of symbols and their meanings used in describing the model. The equations of motion are derived from the principle of the conservation of momentum with approximation of the hydrostatic balance in the vertical direction and are given below.

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{hu}{mn}\right) = R_u \quad = \quad & -\left[\frac{\partial}{\partial \xi}\left(\frac{hu^2}{n}\right) + \frac{\partial}{\partial \eta}\left(\frac{huv}{m}\right) + \frac{\partial}{\partial \sigma}\left(\frac{hu\Omega}{mn}\right)\right] \\ & + \left\{\left(\frac{f}{mn}\right) + v\frac{\partial}{\partial \xi}\left(\frac{1}{n}\right) - u\frac{\partial}{\partial \eta}\left(\frac{1}{m}\right)\right\} hv \\ & - \left(\frac{h}{n}\right)\frac{\partial \phi}{\partial \xi} + (1-\sigma)\left(\frac{gh\rho}{2\rho_0 n}\right)\frac{\partial h}{\partial \xi} \\ & + h\nu\left[\frac{\partial}{\partial \xi}\left(\frac{m}{n}\frac{\partial u}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(\frac{n}{m}\frac{\partial u}{\partial \eta}\right)\right] \\ & + \left(\frac{4}{hmn}\right)\frac{\partial}{\partial \sigma}\left(\nu\frac{\partial u}{\partial \sigma}\right) \qquad (1)\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{hv}{mn}\right) = R_v \quad = \quad & -\left[\frac{\partial}{\partial \xi}\left(\frac{huv}{n}\right) + \frac{\partial}{\partial \eta}\left(\frac{hv^2}{m}\right) + \frac{\partial}{\partial \sigma}\left(\frac{hv\Omega}{mn}\right)\right] \\ & - \left\{\left(\frac{f}{mn}\right) + v\frac{\partial}{\partial \xi}\left(\frac{1}{n}\right) - u\frac{\partial}{\partial \eta}\left(\frac{1}{m}\right)\right\} hu \\ & - \left(\frac{h}{m}\right)\frac{\partial \phi}{\partial \eta} + (1-\sigma)\left(\frac{gh\rho}{2\rho_0 m}\right)\frac{\partial h}{\partial \eta} \\ & + h\nu\left[\frac{\partial}{\partial \xi}\left(\frac{m}{n}\frac{\partial v}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(\frac{n}{m}\frac{\partial v}{\partial \eta}\right)\right] \\ & + \left(\frac{4}{hmn}\right)\frac{\partial}{\partial \sigma}\left(\nu\frac{\partial u}{\partial \sigma}\right) \qquad (2)\end{aligned}$$

$$\frac{\partial \phi}{\partial \sigma} \quad = \quad -\left(\frac{gh\rho}{2\rho_0}\right) \qquad (3)$$

By the assumption of incompressibility, conservation of mass gives the continuity equation,

$$\frac{\partial}{\partial \xi}\left(\frac{hu}{n}\right) + \frac{\partial}{\partial \eta}\left(\frac{hv}{m}\right) + \frac{\partial}{\partial \sigma}\left(\frac{h\Omega}{mn}\right) \quad = \quad 0 \qquad (4)$$

The equation of density can be obtained from the conservation of thermal energy,

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{h\rho}{mn}\right) = RR \quad = \quad & -\left[\frac{\partial}{\partial \xi}\left(\frac{hu\rho}{n}\right) + \frac{\partial}{\partial \eta}\left(\frac{hv\rho}{m}\right) + \frac{\partial}{\partial \sigma}\left(\frac{h\Omega\rho}{mn}\right)\right] \\ & + h\kappa\left[\frac{\partial}{\partial \xi}\left(\frac{m}{n}\frac{\partial \rho}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(\frac{n}{m}\frac{\partial \rho}{\partial \eta}\right)\right] \\ & + \left(\frac{4}{hmn}\right)\frac{\partial}{\partial \sigma}\left(\kappa\frac{\partial \rho}{\partial \sigma}\right) \qquad (5)\end{aligned}$$

Because there is an unknown contribution to the depth-averaged component of the pressure field arising due to the rigid lid, the vertically averaged stream-function is introduced by equations (6) and (7)

$$\frac{\partial \psi}{\partial \eta} \quad = \quad -\overline{u}\left(\frac{h}{n}\right) \qquad (6)$$

$$\frac{\partial \psi}{\partial \xi} \quad = \quad -\overline{v}\left(\frac{h}{m}\right) \qquad (7)$$

With the aid of the vertically integrated continuity equation, equation (8) requires horizontal non-divergence.

$$\frac{\partial}{\partial \xi}\left(\frac{h\overline{u}}{n}\right) + \frac{\partial}{\partial \eta}\left(\frac{h\overline{v}}{m}\right) \quad = \quad 0 \qquad (8)$$

Moreover, a horizontal vorticity equation may be obtained by vertically integrating equations (1) and (2). The result is shown as

$$\frac{\partial}{\partial t}q = RZ \quad = \quad \left\{\frac{\partial}{\partial \xi}\left[\left(\frac{m}{h}\right)\overline{R_v}\right] - \frac{\partial}{\partial \eta}\left[\left(\frac{n}{h}\right)\overline{R_u}\right]\right\} \qquad (9)$$

where $\overline{R_u}$ and $\overline{R_v}$ are the vertical averages of R_u and R_v.

Under the definition of the vertically averaged vorticity,

$$q = \frac{\partial}{\partial \xi}\left(\frac{\overline{v}}{n}\right) - \frac{\partial}{\partial \eta}\left(\frac{\overline{u}}{m}\right) \tag{10}$$

a non-separable elliptic partial differential equation for the stream-function can be derived

$$q = \left(\frac{m}{hn}\right)\frac{\partial^2 \psi}{\partial \xi^2} + \left(\frac{n}{hm}\right)\frac{\partial^2 \psi}{\partial \eta^2} + \frac{\partial}{\partial \xi}\left(\frac{m}{hn}\right)\frac{\partial \psi}{\partial \xi} + \frac{\partial}{\partial \eta}\left(\frac{n}{hm}\right)\frac{\partial \psi}{\partial \eta} \tag{11}$$

There are a total of seven partial differential equations (PDEs), equation (1), (2), (3), (4), (5), (9) and (11), which form a complete set of governing equations. We name Equation (1), (2), (3), (4), (5), (9) and (11) as *u*-Equation, *v*-Equation, *phi*-Equation, *w*-Equation, *rho*-Equation, *q*-Equation, and *psi*-Equation, respectively. Four of these equations, *u*-Equation, *v*-Equation, *w*-Equation and *rho*-Equation, are mixed hyperbolic-parabolic PDEs, while *psi*-Equation is an elliptic PDE and the rest are vertical ordinary differential equations. The dependent variables to be solved are the three components of velocity (u, v, and Ω), density (ρ), dynamic pressure (ϕ), vertically averaged vorticity (q), and vertically averaged stream-function (ψ). For this version of the model, the lateral boundary is assumed to be periodic, and both top and bottom boundaries are rigid. In particular, model can be forced by either imposed wind stresses or thermal fluxes. Initial conditions are easily supplied prior to run.

The PDEs of the model are approximated by various numerical schemes. In the vertical direction, the scheme is pseudo-spectral and the basis set is a modified set of *Chebyshev polynomials*. Not only are the Chebyshev polynomials accurate in the computational sense, but they also give high resolution near both top and bottom boundaries.

For the model equations in the horizontal plane, a method based on the traditional second-order finite-difference method with the Arakawa-C grid is employed [7]. For each partial differential term, say ∂m, where m is a function of x, y and z, if we want to differentiate it along x-direction, we take the difference between $m(x \pm \Delta x, y, z)$ and $m(x, y, z)$ as a result of ∂m at (x, y, z). Note that Δx is a constant value. It depends on how we set the grid on the function. Usually, Δx is equal to the width of each fringe. Lastly, time-stepping is carried out using a predictor-corrector technique based on the leapfrog-trapezoidal scheme.

3 Overview of the Intel Paragon

The Paragon XP/S from Intel Corporation is a distributed-memory multiple instruction stream multiple data stream (MIMD) machine in which the nodes are connected through a fast 2-dimensional mesh network. Each node operating at a clock speed of 50 MHz is a self-contained computer board with a 75-MFLOPS Intel i860/XP processor and 32 MB of main memory. The system consists of three types of nodes: compute nodes, which are used for the execution of parallel programs; service nodes, which offer capabilities of a UNIX system, including compilers and program development tools; and I/O nodes, which provide interfaces to mass storage and LANs.

We can think of the nodes of the Intel Paragon as physically separate computers, but all the nodes function identically. Each node can also run different programs. To obtain a better performance from the system, users can program several nodes to cooperate on a single application.

Paragon Mesh Routing Chips (MRCs), connected by high-speed channels, are the basis of the communication network, where nodes may be attached. There are two independent channels – one for each direction – between any two neighboring nodes. The channels are 16 bits wide and have a bandwidth of 175 Mbps. The MRCs can route messages autonomously and are independent of the attached nodes. In order to avoid deadlocks, communication uses deterministic wormhole routing. Messages are sent first in the horizontal direction and then in the vertical direction. The pipelined nature of the wormhole routing allows the usable bandwidth to be nearly independent of the distance between any two communicating nodes.

The Intel Paragon provides a flexible programming environment for developing parallel programs [17]. The Paragon's operating system is called Paragon OSF/1, which provides an OSF/1-compatible application interface. The NX library is designed for message-passing among cooperating nodes. While support is provided for both synchronous and asynchronous messages as well as interrupt-producing messages, global operations such as global sum etc. are also available. For our experiments, we have used a 140-node Paragon at the Hong Kong University of Science and Technology.

4 Parallel Implementation

In this section, we describe our implementation scheme for the ocean circulation model. The computational flow and dependencies among the equations of the ocean model are shown in Figure 1. Here each solid box represents a major computational function of various equations; equation numbers are described in Section 2. Some of the solid boxes show calculations for only a part of the equation, and others for the entire equation. RHS and LHS denote the *right-hand-side* and *the left-hand-side* of each equation, respectively. PAR and SEQ indicate the corresponding computation to be executed in parallel and sequential fashion, respectively. Each dashed box indicates that data communication and synchronization need to be performed to satisfy the data dependencies between computational phases.

The fundamental principle behind an efficient implementation of such a model is to find the portion of the code which can be parallelized and partition it into smaller execution pieces. The parallel part may be executed with the smaller amount of program execution times by using more processors, implying a finer granularity of the problem. The data communication can play an important role in affecting

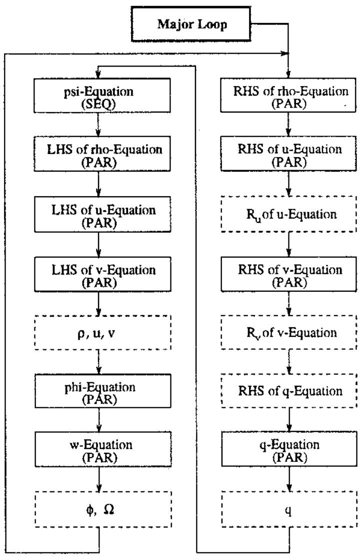


Figure 1. Computational flow of the parallelized ocean model.

the performance of the program – especially when the granularity is small – since all partial results need to be sent to the shared variables during each time step. Clearly when the problem granularity is coarse, communication overhead is smaller but the degree of parallelism is reduced.

Theoretically, any of the three spatial dimensions can be partitioned into sub-domains, each of which can be executed independently by one of the processors. However, because of the usage of the pseudo-spectral scheme in the vertical direction, partitioning in the vertical direction becomes highly cumbersome. Thus, the entire computation domain has been partitioned in the both horizontal directions. Given a data block with dimensions l by m by n along x, y, and z directions respectively, indices start from 1 to l along x-direction, from 1 to m along y-direction, from 0 to n - 1 along z-direction, respectively. Under the proposed scheme, data can be partitioned along either x or y direction, or both.

The majority of the sequential code comes from the calculation of the horizontal stream-function solved by a non-separable elliptic PDE, *psi*-Equation, as mentioned in Section 2. This elliptic solver was provided by Dr. John Adams of the National Center for Atmospheric Research, and is not yet readily parallelizable.

The general algorithm for the parallelization of each single block of the computational function for all equations is shown as follows:

Parallel algorithm for a single computational block

```
for $k$ from 0 to $n - 1$
    for $j$ from $y\_start$ to $y\_stop$
        for $i$ from $x\_start$ to $x\_stop$
            perform numerical computations
        end-for $i$
    end-for $j$
end-for $k$
```

According to the algorithm, each node requires to store only a subset of the whole data with dimension from *x_start* to *x_stop* along x-direction, from *y_start* to *y_stop* along y-direction, from 0 to n - 1 along z-direction, where each node has its own set of *x_start*, *x_stop*, *y_start*, and *y_stop*.

Since most of the computations require data from the boundary of neighboring processors in the entire three-dimensional domain, such kind of data partitioning involves substantial amount of data communication of different types among processors for computing the data at the domain boundaries.

To minimize the communication overheads during computation, we examined the portion of data required during different computations. Our implementation strategy also includes collective communication, by grouping different types of data into combined messages as much as possible. This means the updates are done once during each time step and therefore communication overhead is drastically reduced.

The peripheral data around a local partition may be required for calculation, but this data is stored into the local memories of the neighboring processors. Hence, data communication is required to receive the relevant data from the neighboring nodes in order to perform the correct computation. To further reduce the amount of communication, we examined the computational path in details to decide which part of the data communication could be eliminated. According to our analysis, there are only five major variables, u, v, ρ, ϕ and Ω, that are modified in each iteration using some data not present locally. Figure 2 shows, for each such variable, the portion of the data within the boundaries to be required for computation and updated for each iteration.

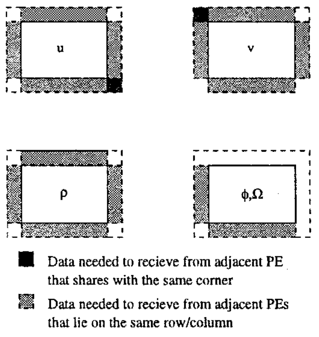

u
v
ρ
ϕ,Ω
Data needed to recieve from adjacent PE that shares with the same corner
Data needed to recieve from adjacent PEs that lie on the same row/column

Figure 2. Portion of data needed to receive from neighboring processors.

In the ocean model, as mentioned above, the majority of the sequential code comes from the calculation of the elliptical PDE stream-function. At present, it is not clear how to execute this equation in parallel. Also, referring to Figure 1 and the equations in Section 2, solving equations (1), (2) and (5) in parallel will not cause any conflict. Figure 3 depicts the task graph, indicating the dependencies and parallelism among various equations, for the ocean circulation model. Thus, it is possible to exploit functional parallelism by solving multiple equations simultaneously and by using data parallelism within each equation. However, in order to extract meaningful improvements from such a scheme, a further careful analysis of the communication overhead is required. Moreover, a parallelization strategy for the serial elliptical solver would be required. Currently, we are exploring these possibilities.

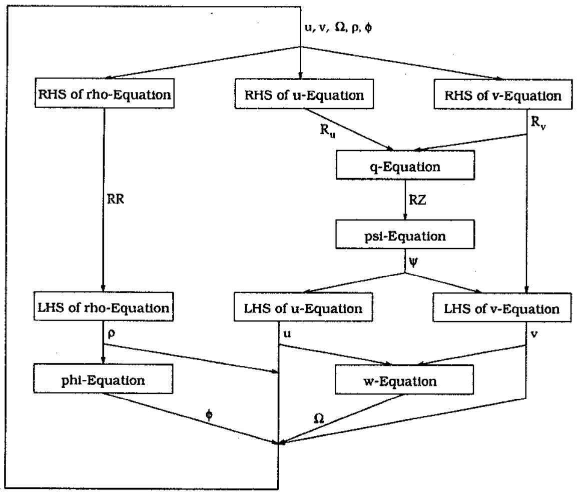


Figure 3. Task graph for the Ocean Circulation Model.

5 Experimental Results

In our experiments, we considered barotropic continental shelf waves in a periodic channel as a test problem. Such waves are solutions of a simplified linear version of the nonlinear primitive-equation ocean circulation model described in Section 2. To obtain the numerical solution, the imposed wave amplitude was small so that the nonlinear terms became negligible. The vertical structure was represented by the barotropic mode and the first four vertical baroclinic modes in the Chebyshev wavenumber space, although only the barotropic mode was actually necessary. The time step was chosen to be 0.01 of the wave period, and the bottom topography varied exponentially across the channel. Furthermore, the total number of time steps for each run was chosen to be 50.

The horizontal grid sizes of the numerical model for both x and y directions were chosen to be (l x m), where $l, m \in \{21, 41, 61\}$. The vertical dimension of the grid was kept fixed as 5 points. Hence, there were altogether 6 combinations for the problem grid. The parallel code was run on the Intel Paragon with 1, 2, 4, 8, 16, 32, 64 and 100 processors, corresponding to 48 test cases using the 6 grids. For determining the effect of data partitioning in the two horizontal dimensions, the processors were used in various combinations of two-dimensional meshes. For those experiments, a total of 36 processor configurations were used, and therefore, using 6 grids, the number of test cases corresponded to 196.

In order to collect timing data about the program execution, software timers were placed within several parts of the code to measure the elapsed time for each relevant section of the program code. To raise the accuracy of the result obtained, the same set of experiment was performed a total of 12 times.

The results are provided in four sets. The first set provides the measured total elapsed times, speedups, the serial times, and the average communication overhead, for each of the 48 test cases (6 grids and 8 processor configurations). The second set includes the results showing the fraction of time spent on various equations. The third set consists of the plots showing the impact of data partitioning in the two horizontal directions for 196 test cases (6 grid and 36 processor configurations). The fourth set includes a template showing the states of the ocean at different times.

Table 2. Elapsed times (in seconds) for various grids.

Grid size	Number of nodes							
	1	2	4	8	16	32	64	100
21x21	55.82	29.69	15.82	9.20	5.83	4.79	4.35	4.61
41x21	112.81	58.59	31.15	17.57	10.82	7.68	6.78	6.69
41x41	229.16	119.34	62.17	34.71	20.38	14.00	11.05	10.27
61x21	170.67	88.53	47.40	26.68	16.61	12.06	10.23	9.70
61x41	345.58	182.63	94.89	53.21	31.65	22.50	17.61	16.21
61x61	521.25	272.92	143.00	79.53	47.33	33.39	25.97	23.01

We first examine the first set of experiments. Table 2 shows the total elapsed times for the 6 grid sizes using 1, 2, 4, 8, 16, 32, 64 and 100 processors. The processor grid configurations in these cases were 1x1, 2x1, 2x2, 4x2, 4x4, 8x4, 8x8, and 10x10. As can be seen, for each grid size, the total elapsed time was steadily decreased with an increase in the number of processors. One can also be noticed – comparing the top and bottom entries in the first column – that the serial time was increased 9 times when the problem size was increased 9 times (from 21x21 to 61x61). However, this ratio starts decreasing as the number of processors was increased gradually. For example, this ratio was approximately 8 when the number of processors was 16. This ratio dropped to 7, 6 and then 5 when the number of processors was 32, 64 and 100, respectively.

Table 3. Speedups for various grids.

Grid size	Number of nodes							
	1	2	4	8	16	32	64	100
21x21	1.00	1.88	3.53	6.07	9.57	11.66	12.84	12.11
41x21	1.00	1.93	3.62	6.42	10.43	14.69	16.64	16.87
41x41	1.00	1.92	3.69	6.60	11.24	16.37	20.74	22.31
61x21	1.00	1.93	3.60	6.40	10.28	14.15	16.68	17.59
61x41	1.00	1.89	3.64	6.49	10.92	15.36	19.62	21.32
61x61	1.00	1.91	3.65	6.55	11.01	15.61	20.07	22.65

The corresponding values of speedup for the same set of experiments are shown in Table 3. These values are quite encouraging and one can notice an increasing trend in speedup with increasing number of processors. The increase in speedup was very rapid up to 64 processors after which the amount of improvement became smaller. This is despite the fact that, while the parallel times for the other components became smaller, there was a substantial amount of serial computations in the elliptical solver which was proved to be the bottleneck.

Table 4. Serial times (in sec) for various grids.

Grid size	Number of nodes							
	1	2	4	8	16	32	64	100
21x21	0.82	0.81	0.82	0.82	0.82	0.81	0.83	0.85
41x21	2.22	2.22	2.22	2.23	2.26	2.27	2.24	2.28
41x41	4.41	4.41	4.42	4.42	4.42	4.47	4.45	4.42
61x21	4.36	4.36	4.37	4.37	4.38	4.46	4.46	4.54
61x41	8.67	8.83	8.67	8.68	8.69	8.84	8.85	9.00
61x61	12.89	12.87	12.87	12.88	13.12	13.12	13.36	13.35

We also calculated separately the times for the serial part. These times are provided in Table 4. The serial times were almost constant for a fixed grid size. However, one can also observe that the fraction of the serial component became much larger for large grids – for example, the serial time increased by a factor ranging from 15.7 to 16.2 when the grid size was increased from 21x21 to 61x61. While the parallel part steadily decreased, this notorious serial part remained the main hurdle in improving the speedup further. For example, using 100 processors, the serial fraction constituted more than 58% of the total running time.

Table 5. Average communication overheads (in sec) for various grids.

Grid size	Number of nodes						
	2	4	8	16	32	64	100
21x21	0.20	0.64	0.60	0.68	1.30	1.77	2.18
41x21	0.21	0.83	0.74	0.76	0.99	1.84	2.24
41x41	0.35	1.07	1.00	0.83	1.15	1.77	2.20
61x21	0.23	1.06	0.85	1.03	1.43	2.22	2.44
61x41	1.27	1.38	1.18	1.00	1.88	2.26	2.54
61x61	0.58	1.76	1.36	1.41	2.80	3.32	3.11

The times spent on communication are given in Table 5. These times represent the sum of all communication overheads incurred by the processors performing the parallel computation. The values in this table were determined by taking the averages across all processors except the processor 0 which was performing the serial part. Clearly, the communication times increased with an increase in the problem size and the number of processors. Comparing these value in this table with the total elapsed times shown earlier in Table 2, we can notice that the fraction of communication overheads out of the total elapsed times was as large as 0.47 (for the 21x21 grid using 100 processors) but decreased when the problem size is large (61x61 grid using 100 processors).

The results for the next set of experiments are given in Figure 4, Figure 5, and Figure 6. In each of these figures, we provide four plots. The first three plots show the fraction of the time spent on the three most computationally intensive equations, using a variable number of processors. These equations are *u*-Equation, *v*-Equation and *rho*-Equation. The fourth plot shows the fraction of the time spent on all equations (serial as well as parallel). This time was taken across the processor 0 which executed the parallel computation – like all other nodes – and also executed the serial elliptical PDE solver. In other words, this fraction includes the time for computing *u*-Equation, *v*-Equation, *rho*-Equation, and all other equations. The fraction of the communication and synchronization time is thus simply 1 minus this fraction. An inspection of Figure 4 reveals that the fraction for all three equations as well as the cumulative Equation component dropped rapidly as the number of processors was increased.

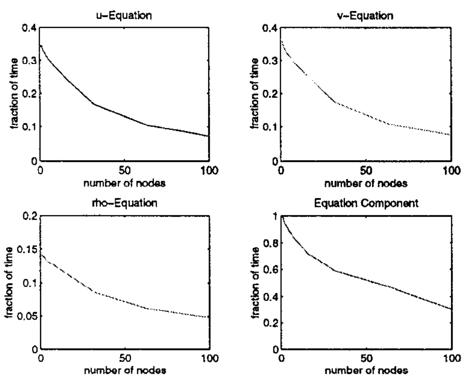


Figure 4. Distributions of elapsed times on the Paragon for 21x21 grid.

One reason for the increased communication and synchronization overhead was that the problem size was fixed and therefore granularity decreased with increasing number of processors. The excessive amount of this overhead, however, was largely due to the fact that all nodes needed to send the relevant data to a single processor before the execution of the serial equation. Similarly, the processor 0 needed to send all of the relevant computed results back to all other processors upon the computation of that equation. Since this indeed induced a contention on the processor 0 as well as a large amount of communication data, the performance of the parallel program deteriorated rapidly.

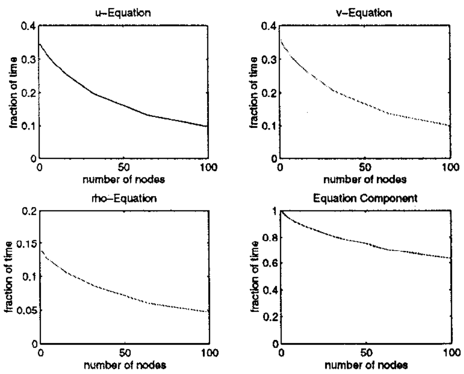


Figure 5. Distributions of elapsed times on the Paragon for 41x41 grid.

From Figure 5 and Figure 6, where we used 41x41 and 61x61 grid sizes, we can notice a similar trend. In particular, we can observe that the three compute-intensive equations yielded the same fraction of time as in the case of Figure 4 (when the data size was 21x21). This implies that these equations were properly parallelized. We can also notice that the fraction of the cumulative Equation Component was larger when the grid size was increased. This confirms with the earlier results which indicated that the larger problem sizes exhibited better speedup and smaller communication and synchronization overheads.

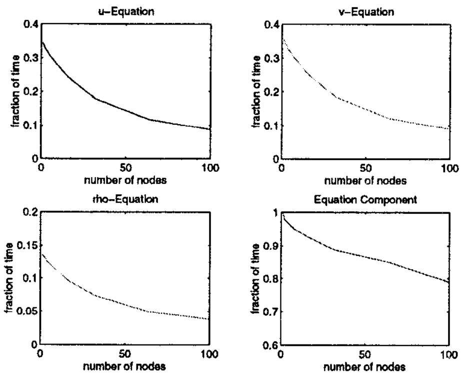
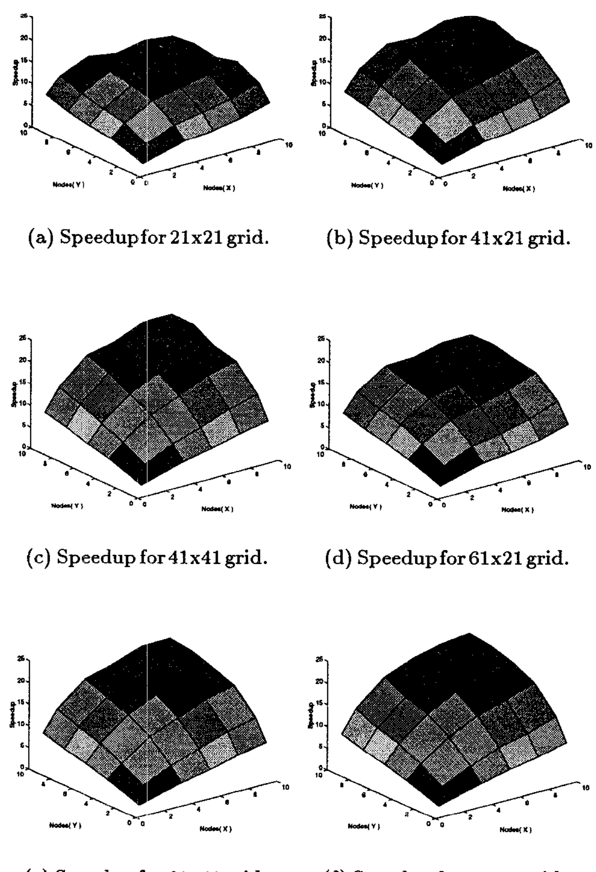


Figure 6. Distributions of elapsed times on the Paragon for 61x61 grid.

The third set of results show the effect of partitioning in the two horizontal dimensions. For this purpose, as mentioned earlier, we used 36 configurations for processor meshes, ranging from linear arrays to rectangles - such as 4x2, 2x4, 6x2, 2x6, 8x2, 2x8, 10x2 and 2x10 - to perfect squares - such as 2x2, 4x4, 6x6, 8x8 and 10x10. These results for 6 different grid sizes are illustrated through 3-dimensional plots in Figure 7. From this figure, we can see that speedups for partitioning in y-direction in general outperformed those in x-direction. This is particularly true when the grid was square. However, when the grid was large in the x-dimension such as 61x21, this was not always the case. This is perhaps due to the nature of some of the computationally-intensive equations which require less communication in the y-dimension. However, in almost all cases, we observed that partitioning in both dimensions was better than partitioning in just one direction. For example, when the grid size was 61x61, the speedup using a 4x4 processor grid was 11.013. On the other hand, the speedup was 10.294 using 8x2 processor grid and was 10.604 using 2x8 processor grid. Similar observations can be made about other cases.

Our fourth set of result (Figure 8) is a template showing the state of the ocean at different times. The state of the ocean is described by observing the horizontal velocity components (u, v) and the vertically averaging stream-function (ψ). The grid size in this case was 61x61. In order to observe these variables, we set the vertical velocity (Ω) and density (ρ) and dynamic pressure (ϕ) equal to 0. This was done to remove the effect of the height. Observations were made at time steps 0, 10, 20, 30, 40 and 50. This figure indicates the smooth movements of variables at different times.



(a) Speedup for 21x21 grid. (b) Speedup for 41x21 grid.

(c) Speedup for 41x41 grid. (d) Speedup for 61x21 grid.

(e) Speedup for 61x41 grid. (f) Speedup for 61x61 grid.

Figure 7. Speedup plots for various grids.

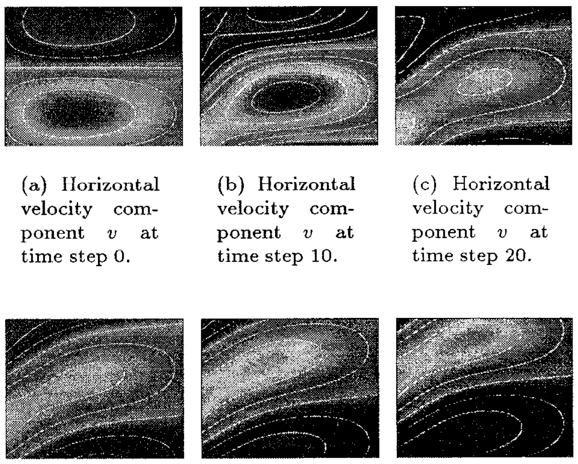
6 Conclusions

In this paper, we described the parallel implementation of a grand-challenge problem, the Primitive-Equation Numerical Ocean Circulation Model, on the Intel Paragon. In our experiments, we considered barotropic continental shelf waves in a periodic channel as a test problem. Results show that the problem scaled very well and yielded a good speedup despite a large fraction of the serial computation.

While reasonable speedup was obtained by partitioning the domain in either direction, it is better to have an evenly-partitioned processor mesh in order to minimize the relative wastage in both mesh directions. This is because for a sufficiently large problem size, it is possible to obtain a further improvements in speedup by partitioning in both dimensions when the number of nodes is very large, say, 100.

There are several possible extensions to our work some of which are listed below:

- devise a scheme to parallelize the elliptical PDE solver;



(a) Horizontal velocity component v at time step 0.
(b) Horizontal velocity component v at time step 10.
(c) Horizontal velocity component v at time step 20.
(d) Horizontal velocity component v at time step 30.
(e) Horizontal velocity component v at time step 40.
(f) Horizontal velocity component v at time step 50.

Figure 8. States of the horizontal velocity component v at different times.

- exploit control parallelism in addition to the data parallelism;
- heterogeneous scheduling techniques may be employed to run the serial part on a fast workstation while the fine-grained computations are executed on the parallel processors;
- obtain further improvement by possibly partitioning the domain in the z-direction.

The model has numerous applications. The water mass and circulation in the polar region have strong impact in the change of our climate through the circulation between polar, mid-latitude, and tropical oceans. The model can also be used for various other studies, such as coastal trapped waves, shelf-break fronts, eastern boundary current, bottom density front, coastal up-welling and down-welling, and isolated topography.

References

[1] A. Beckmann and D. B. Haidvogel. Numerical Simulation of Flow Around a Tall Isolated Seamount. Part I: Problem Formulation and Model Accuracy. *Journal of Physical Oceanography*, 23:1736–1753, 1993.

[2] R. Bleck and D. B. Boudra. Wind-Driven Spin-Up in Eddy-Resolving Ocean Models Formulated in Isopycnic and Isobaric Coordinates. *Journal of Geophysical Research*, 91:7611–7621, 1986.

[3] K. Bryan. A Numerical Investigation of Nonlinear Model of a Wind-Driven Ocean. *Journal of Atmospheric Sciences*, 20:594–606, 1963.

[4] D. C. Chapman and D. B. Haidvogel. Formation of Taylor Caps over a Tall Isolated Seamount in a Stratified Ocean. *Geophysical and Astrophysical Fluid Dynamics*, 64:31–65, 1992.

[5] D. C. Chapman and D. B. Haidvogel. Generation of Internal Lee Waves Trapped over a Tall Seamount. *Geophysical and Astrophysical Fluid Dynamics*, 69:33–54, 1993.

[6] D. C. Chapman and S. Lentz. Trapping of a Coastal Density Front by the Bottom Boundary Layer. *Journal of Physical Oceanography*, 24:1464–1479, 1994.

[7] T. L. Freeman and C. Phillips. *Parallel Numerical Algorithms*. Prentice Hall, 1992.

[8] G. Gawarkiewicz. Steady Wind Forcing of a Density Front over a Circular Bank. *Journal of Marine Research*, 51:109–134, 1993.

[9] G. Gawarkiewicz and D. C. Chapman. Formation and Maintenance of Shelfbreak Fronts in an Unstratified Flow. *Journal of Physical Oceanography*, 21:1225–1239, 1991.

[10] G. Gawarkiewicz and D. C. Chapman. The Role of Stratification in the Formation and Maintenance of Shelf-Break Front. *Journal of Physical Oceanography*, 22:753–772, 1992.

[11] D. B. Haidvogel, A. Beckmann, D. C. Chapman, and R. Q. Lin. Numerical Simulation of Flow Around a Tall Isolated Seamount. Part II: Resonant Generation of Trapped Waves. *Journal of Physical Oceanography*, 23:2373–2391, 1993.

[12] D. B. Haidvogel, J. L. Wilkin, and R. Young. A Semi-Spectral Primitive Equation Ocean Circulation Model using Vertical Sigma and Orthogonal Curvilinear Horizontal Coordinates. *Journal of Computational Physics*, 94:151–185, 1991.

[13] N. E. Heaps. Three-Dimensional Coastal Ocean Models. *American Geophysical Union*, 208, 1987.

[14] E. E. Hoffmann, K. S. Hedstrom, J. R. Moisan, D. B. Haidvogel, and D. L. Mackas. Use If Simulated Drifter Tracks to Investigate General Transport Patterns and Residence Times in the Coastal Transition Zone. *Journal of Geophysical Research*, 96:15041–15052, 1991.

[15] H. Hsu. Numerical Studies of Arctic Shelf Circulation for Wintertime Dense Water and Summertime Light Water (Invited). In *Arctic System Science (ARCSS) – Ocean Atmosphere Ice Interaction (OAII) Modelling Workshop*, Monterey, CA, July 13-14, 1992.

[16] H. Hsu, R. C. Beardsley, and J. F. Price. Including Surface and Bottom Boundary Layer Physics in a Primitive-Equation Model to Study Coastal Circulations. In *AGU 1992 Ocean Science Meeting*, New Orleans, LA, January 27-31, 1992.

[17] Intel Supercomputer Systems Division. *Paragon User's Guide*. Intel Corporation, June 1994.

[18] J. L. Wilkin and D. C. Chapman. Scattering of Coastal-Trapped Waves by Irregularitics in Coastline and Topography. *Journal of Physical Oceanography*, 20:396–421, 1990.