The Parallelization of the Princeton Ocean Model

L.A. Boukas[1], N.Th. Mimikou[1], N.M. Missirlis[1]*, G.L. Mellor[2], A. Lascaratos[3], and G. Korres[3]

[1] Department of Informatics, University of Athens,
Panepistimiopolis 157 10, Athens, Greece.
[2] Princeton University,
Princeton, NJ 08544–0710, USA.
[3] Department of Applied Physics, University of Athens,
Panepistimiopolis 157 84, Athens, Greece.

Abstract. In this paper we present the parallel implementation of the Princeton Ocean Model (POM) using message passing. Domain decomposition techniques are used for the horizontal discretization whereas in the vertical direction each column per grid point is computed. The required interprocessor communication was implemented on the PVM message passing library. Three different data partitioning schemes are studied. It is found that the checkerboard partitioning produces the best results when the number of processors becomes larger than 15, otherwise row block striped partitioning is to be preferred.

Keywords: Princeton Ocean Model, domain decomposition, message passing

1 Introduction

The use of parallelism as a vehicle to advance Earth's simulations is considered as one of the most complex problems facing the scientific community today and is one of the Grand Challenges of the next century.

Climate models and in particular ocean models have followed a similar development with atmospheric models and in fact they have been coupled to simulate the Earth's global climate dynamics. However, there is a high demand for high–resolution and inclusion of more complex physical processes in order to make accurate predictions for future climate conditions. For meeting the required demands a huge amount of computations is needed that can be carried out in realistic time using parallel computers.

In previous work [1, 12, 5] we have parallelized the atmospheric model eta which is in operational use at the Greek National Meteorological Service. In the present paper we consider the parallelization of the Princeton Ocean Model (POM) [4]. Many other groups are developing parallel ocean models. A cluster

* Author for correspondence: Email: nmis@di.uoa.gr

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 1395–1402, 1999.

version of the Ocean Circulation Model (OCM) based on a model of the Geophysical Fluid Dynamics Laboratory (GFDL) for the CEDAR was developed in [6, 7, 8]. In [2] and [20] simple shallow–sea models were parallelized on different parallel machines. A parallel version of POP (Parallel Ocean Program) was developed in [19] for the CM2 and CM–5 machines. Additional work towards to implementation of POP on parallel computers with recent architectures and couple it with sea ice, atmospheric and land surface models is described in [11]. Another global model developed at GFDL was the Modular Ocean Model (MOM). The code of this model was structured for vector machines. A two dimensional version of MOM code was developed in [16] and is known as the Modular Ocean Model Array (MOMA) code. The Ocean Circulation and Climate Advanced Modelling (OCCAM) is a message passing implementation of the MOMA code [10]. The ocean general circulation model, OPYC which uses implicit time stepping was recently parallelized on the CRAY T3D [15]. HOPE is an ocean model which was also parallelized for distributed memory processors using the PARMACS message passing package [9]. In [3, 17, 18] the multipurpose isopycnic model MICOM was parallelized following two different approaches.

In this paper POM is briefly presented in section 2. In section 3 we discuss the design issues for the parallelization of POM using domain decomposition. Implementation is described in section 4, performance is presented in section 5 and finally future plans conserning continuation and optimizing of the current work is discussed in section 6.

2 The POM Model

The POM model was developed by A. Blumberg and G. Mellor [4]. The model is a three dimensional ocean model incorporating a turbulence closure model to provide a realistic parameterization of the vertical mixing processes [14]. The model uses a sigma coordinate system which simulates well the bottom topography. The prognostic variables are the three components of velocity, temperature, salinity, turbulence kinetic energy and turbulence macroscale.

The equations which form the basis of the circulation model are the traditional hydrodynamic equations for conservation of mass, momentum, temperature and salinity coupled by an equation of state. The governing equations together with their boundary conditions are solved by finite difference techniques. The horizontal grid uses curvilinear orthogonal coordinates on an Arakawa staggered C grid. The horizontal time differencing is explicit whereas the vertical differencing is implicit. The latter eliminates time constraints for the vertical coordinate and permits the use of fine vertical resolution in the surface and bottom layers. The model has a free surface and a mode splitting technique in time has been adopted for computational efficiency. The barotropic (external) mode portion of the primitive equations is two–dimensional and uses a short time step based on the CFL condition and the external wave speed. The baroclinic (internal) mode is three dimensional and uses a long time step based on the CFL condition and the internal wave speed.

3 Parallelization of POM

In this section we describe how we have implemented POM using message passing. The most common technique to introduce parallelism in environmental models using finite differences for the discretization of the involved system of partial differential equations is domain decomposition. The basic idea is to decompose the computational domain into subdomains and assign each subdomain to a different processor. Each processor solves by a separate computing process the problem using the original code. In this way, there is only a single code to be maintained for both sequential and parallel computing platforms. In addition, the sequential code is reused entirely in the parallelization. To keep the computations consistent with the sequential code inter–processor communication is needed.

The computations in the horizontal mesh use explicit finite differences. This means that each grid point in the next time level is computed using only values from grid points in the previous time levels. So, the computations for each grid point in the advanced time level are independent and can be carried out in parallel by assigning a group of these points to each of the available processors. As the computations in the vertical direction use implicit schemes the previous approach cannont be applied since in the present case the values of the grid points in the next time level are computed via the solution of a linear system using direct methods (e.g. Gaussian elimination). Although there exist methods for the parallel solution of such systems, we postponed this approach for future investigation and decided not to introduce parallelism in the vertical direction.

3.1 Mapping Schemes and Communication Patterns

The discretization of the involved partial differential equations defines a rectangular mesh of grid points, where the value of each point is computed independently. There are a number of different ways to partition a rectangular domain [13]. However, for simplicity of algorithm implementation, the domain is usually divided into subdomains with simple and regular geometries.

In the present study we consider basicaly two ways to partition a rectangular two dimensional domain: block striped and checkerboard partitioning. In the block striped partitioning each processor is assigned a continuous set of rows or columns. In a rowwise block striping of an $N \times M$ mesh on p processors (labeled $P_0,\ P_1,\ \ldots,\ P_{p-1}$) processor P_i contains rows with indices $(N/p)i,\ (N/p)i + 1,\ \ldots,\ (N/p)(i+1)-1$. In the checkerboard partitioning the mesh is partitioned into smaller rectangular blocks of equal size that are distributed among the processors.

A checkerboard partitioning splits both rows and columns of the mesh, so no processor is assigned any complete row or column. A checkerboard–partitioned rectangular mesh maps naturally onto a two–dimensional rectangular mesh of processors. The $N \times M$ mesh maps onto an $r \times q = p$ processor mesh by dividing it into blocks of size $N/r \times M/q$. The computations of each grid column are independent of each other and as we do not consider any parallelization in the

vertical direction they do not cause any additional implications to the aforementioned partitioning techniques. Due to the structure of the computational stencil processors have to communicate their boundary points to the eight neighbouring processors after each scan of the domain. Therefore, each processor contains all the grid points corresponding to its domain, as well as those grid points which form its artificial boundary (in our case two rows/columns in each side). The communication pattern is carried out concurrently in all local processors. To avoid communication between diagonal processors which contain only a common corner, the communication is carried out in two phases. In the first phase the column boundaries are exchanged with the left and right neighbour and in the second phase the row boundaries are exchanged with the upper and lower neighbour. Note, that the aforementioned phases may be performed in any order.

4 Parallel Implementation

The main objective during the parallelization of the POM model was to produce a portable, easy to use optimised code that would execute on both shared and distributed memory architectures via interfaces that support message passing. This was achieved by gathering all the machine and interface dependent routines and grouping optimised versions of them in a number of files that were finally used to form a library serving for all parts of the parallel program discussed in this section, from the initial data partitioning phase to the communication patterns.

During the execution of the parallel POM model two types of communication arise:

- *local communication* used to update the halo points whenever neccessary.
- *global communication* at each external time step to check whether the CFL condition is satisfied.

4.1 Data Partitioning

Prior to the execution of the parallel program, a data partitioning scheme, as described in section 3, is used to allocate the data to the processors while maintaining a reasonably even distribution of the work to be performed. In this part, all the data that will be used as input to the parallel program are decomposed with consecutive calls to the library routines *par_split_2d* and *par_split_3d* for (x, y)–decomposition of 2d and 3d matrices, respectively. The z–direction is allocated to the same processor. For each processor that takes part in the data mapping, special routines are called in order to decide its position on the mesh, its neighbours and the amount of kernel and hallo points that it will hold.

For the checkerboard partitioning, the structure of the computational stencil requires two rows/columns artificial boundary for the computation of the points on the perimeter of each subdomain.

4.2 Local Communication

For the purposes of message passing, a subroutine *par_exchange* handles all the steps involved in the communication. Each message is formed by internal elements of the sending processor, then it is packed in a vector array and is finally transmited to the receiving processor, which will unpack the message and use it to update its halo points. According to the processor the message will be sent to, and considering the fact that there is an overlapping of message so as to avoid communication with the diagonal processors, we have the following four variants of the Send subroutine:

Send_up, Send_down, Send_left, Send_right

Similarly, according to the processor from which the message has been sent, we have the following variants of the Receive subroutine:

Recv_down, Recv_up, Recv_right, Recv_left

Considering the fact that the proposed communication scheme consists of four Send calls, that can be performed in pairs concurrently, and four blocking Receive calls, the body of subroutine Exchange is either: call Send_up, call Send_down, call Recv_down, call Recv_up, call Send_left, call Send_right, call Recv_right, call Recv_left or call Send_left, call Send_right, call Recv_right, call Recv_left, call Send_up, call Send_down, call Recv_down, call Recv_up with additional two alternative forms.

Communication, expressed in calls of subroutine *par_exchange*, is chosen to be performed at the beginning of the subprogram that requires it so as to avoid repeated and unnecessary updating of the same halo points within loops, unless this is critical to the correctness of the results.

Moreover, communication is restricted to 2d and 3d arrays and is handled differently by each processor, in the sense that the amount of halo points exchanged by each processor depends on its position in the 2d–mesh, as processors laying on the perimeter of the mesh have fewer "valid" halo points than the internal ones. With the term "valid", we refer to those halo points that are essential for the execution, in contrast to the "dummy" halo points assigned to the processors from the side that they lack a neighbour, for reasons of consistency as well as security regarding pointer referencings.

5 Performance

The aforementioned data partitioning and message passing techniques have been successfully tested and validated in both shared and distributed memory architectures. For the purposes of this paper, runs were made on the distributed memory Parsytec CC platform with 2, 4, 6, 8, 10, 12, 14 and 16 processors which was the platform used for purposes of development and testing of the results. The 128MB RAM restriction discouraged the execution of the sequential

program. Thus, all results assume time for two processors to be the basis for the scalability study.

The programming model that was selected for the implementation was PVM as being our initial development environment while an optimised MPI version is currently being tested and refined.

The data sets used for our experiments simulate the Mediterranean basin and contain 30 layers of 360×120 of grid points each, that is a grid spacing of 0.125^o. Figures 1 and 2 present our numerical results of the model as they were estimated for a 30–days 3–d calculation run taking 1440 steps. Times are measured in seconds in all cases. From the results of Fig. 1 and 2 we note that the best

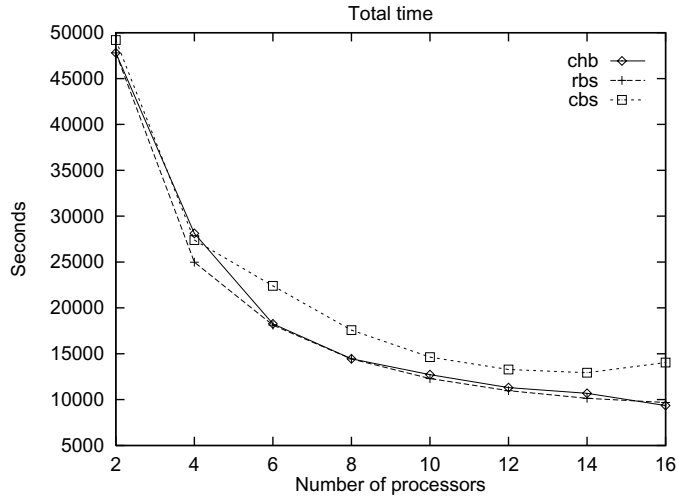

Fig. 1. Parallel run time

performance for the POM model is achieved for the checkerboard partitioning when the number of processors is greater than 15, otherwise block row striped partitioning is slightly better. This serves as an indication towards selecting, when real–time problems are studied, of that grid size–number of processors combination which would offer the

The achieved results 5.1 and 0.64 for the relative speedup (estimated true speedup 10.2) and efficiency, respectively, for 16 processors show also a satisfactory scalability behaviour of the model.

6 Future Work

Even though for a first approach the results are quite encouraging, they are amenable to improvements. More work has to be done, covering multiple areas not only of parallelization but also in conjuction to the parallel POM model with

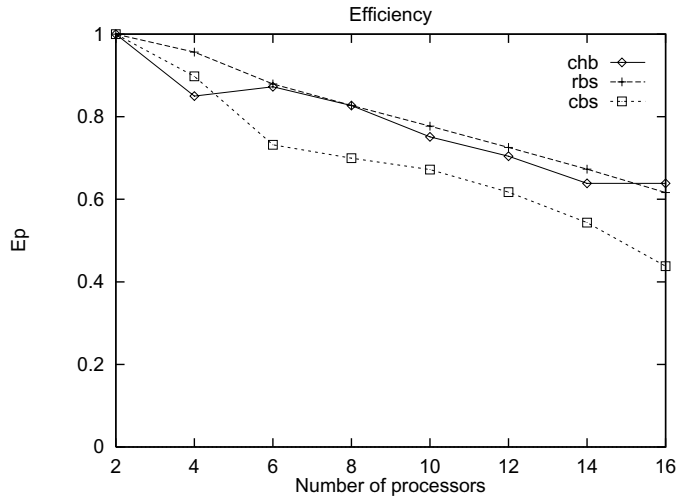

Fig. 2. Efficiency

other relative applications in order to provide a wider package of applications concerning enviroment.

References

[1] Argyropoulos, N.E., Boukas, L.A., Mimikou, N.Th., Missirlis, N.M., Papageorgiou, J.G.: A Distributed Implementation of the Numerical Weather Prediction Eta Model. Presented at the IASTED International Conference Parallel and Distributed Systems, Euro–PDS'97, June 9–11, 1997, Barcelona, Spain and appeared in the Proceedings of the IASTED Conference on Parallel and Distributed Computing and Networks, IASTED/Acta Press (also accepted, after selection, to be published in extended form in the IASTED Journal for Parallel and Distributed Systems) (1997) 301–304 .

[2] Ashworth, M.: Parallel Processing in Environmental Modeling. Parallel Supercomputing in Atmospheric Sciences, G–R. Hoffmann and T. Kauranne eds., World Scientific, Singapore (1993) 1–25.

[3] Bleck, R., Dean, S., O'Keefe, M., Sawdey, A.: A comparison of data parallel and message–passing versions of the Miami Isopycnic Coordinate Ocean Model (MICOM). Parallel Computing, **21** no. 10 (1995) 1695–1720.

[4] Blumberg, A.F., Mellor, G.L.: A coastal ocean numerical model. Mathematical Modelling of Estharine Physics, Proc. Int. Symp., Hamburg, August 24–26, 1978, edited by Sundermann and K.–P. Holz, Springer–Verlag, Berlin (1980) 203–214.

[5] Boukas, L.A., Mimikou, N.Th., Missirlis, N.M.: A parallel implementation of the Eta model. Abstracts of the Symposium on Regional Weather Prediction On Parallel Computer Enviroments Athens, Greece (1997).

[6] DeRose, L., Gallivan K., Gallopoulos, E.: 3–d land avoidance and load balancing in regional ocean simulation. Proc. 1996 Int'l. Conf. Parallel Processing **2** (1996) 158–165.

[7] DeRose, L., Gallivan K., Gallopoulos, E.: Status Report: Parallel Ocean Circulation Modeling on CEDAR. Parallel Supercomputing in Atmospheric Science. G.–R. Hoffmann and T. Kauranne eds., World Scientific,Singapore (1993) 157–172.

[8] DeRose, L, Gallivan, K., Gallopoulos, E.: Experiments with an ocean circulation model on CEDAR. Proc. 1992 ACM International Conference on Supercomputing (1992) 397–408.

[9] Gülzow, V., Kleese, K.: About the parallelization of the HOPE Ocean Model. Proceedings of the Sixth ECMWF Workshop on the Use of Parallel Processors in Meteorology, Reading, England, November 1994. Proceedings published by World Scientific Publishers, Coming of Age, edited by G–R. Hoffmann and N. Kreitz (1995) 505–511.

[10] Gwilliam, C.S.: The OCCAM Global Ocean Model. Proceedings of the Sixth ECMWF Workshop on the Use of Parallel Processors in Meteorology, Reading, England, November 1994. Proceedings published by World Scientific Publishers, Coming of Age, edited by G–R. Hoffmann and N. Kreitz (1995) 446–454.

[11] Jones, P.W.: The Los Alamos parallel ocean program (POP) and coupled model on MPP and clustered SMP architectures. Making its Mark, G–R. Hoffmann and N. Kreitz, World Scientific, Singapore (1997) 226–238.

[12] Kallos, G., Nickovic, S., Jovic, D., Kakaliagou, O., Papadopoulos, A., Missirlis, N., Boukas, L., Mimikou, N.: The Eta Model Operational Forecasting System and its Parallel Implementation. Proceedings of the First Workshop on Large–Scale Scientific Computations, eds. M. Griebel, O. Iliev, S.D. Margenov, P.S. Vassilevski, Varna, Bulgaria (1997) 176–188.

[13] Kumar, V., et al: Introduction to Parallel Computing: Design and Analysis of Algorithms. The Benjamin/Cummings Publ. (1994).

[14] Mellor, G.L., Yamada, T.: Development of a turbulence closure model for geophysical fluid problems. Rev. Geophys. Space Phys. **10** No. 4 (1982) 851–875.

[15] Oberhuber, J.M., Ketelsen, K.: Parallelization of an OGCM on the CRAY T3D. Proceedings of the Sixth ECMWF Workshop on the Use of Parallel Processors in Meteorology, Reading, England, November 1994. Proceedings published by World Scientific Publishers, Coming of Age, edited by G–R. Hoffmann and N. Kreitz (1995) 494–504.

[16] Pacanowski, R.C., Dixon, K., Rosati, A.: The GFDL Modular Ocean Model Users Guide. Technical Report **2** GFDL Ocean Group (1990).

[17] Sawdey, A., O'Keefe, M., Bleck, R., Numrich, R.W.: The design, implementation and performance of a parallel ocean circulation Model. Proceedings of the Sixth ECMWF Workshop on the Use of Parallel Processors in Meteorology, Reading, England, November 1994. Proceedings published by World Scientific Publishers, Coming of Age, edited by G–R. Hoffmann and N. Kreitz (1995) 523–550.

[18] Sawdey, A.C., O'Keefe, M.T., Jones, W.B.: A General Programming Model for Developing Scalable Ocean Circulation Applications. Making its Mark, G–R. Hoffmann and N. Kreitz, World Scientific, Singapore (1997) 209–225.

[19] Smith, R.D., Dukowicz, J.K., Malone, R.C.: Physica D **60** (1992) 38.

[20] Wait, R., Harding, T.J.: Numerical Software for 3D Hydrodynamic Modeling using Transputer Array. Parallel Supercomputing in Atmospheric Sciences, G–R. Hoffmann and T. Kauranne eds., World Scientific, Singapore (1993) 453–464.