Future Generation Computer Systems 9 (1993) 11–18
North-Holland

Implementing finite difference ocean circulation models on MIMD, distributed memory computers

A.R. Clare [a] and D.P. Stevens [b]

[a] *Computing Centre, University of East Anglia, Norwich NR4 7TJ, UK*
[b] *School of Mathematics, University of East Anglia, Norwich NR4 UK*

Abstract

Clare, A.R. and D.P. Stevens, Implementing finite difference ocean circulation models on MIMD, distributed memory computers, Future Generation Computer Systems 9 (1993) 11–18.

This paper considers the use of parallel computers for ocean modelling. After a brief review of the topic, the authors describe the experience of porting a simplified ocean model onto the Computing Surface. The parallel implementation is based on a straightforward domain decomposition. The use of CSTools in the code is briefly discussed. The performance of the parallel code, when run on Inmos T800 transputers and Intel i860's, is compared with the performance of a serial implementation when run on a range of commonly used serial computers (including a CRAY X-MP and an AMDAHL VP1200).

Keywords. High performance computing; parallelism; CFD; ocean circulation models.

1. Introduction

Modelling the ocean requires the solution of a set of four-dimensional (three spatial and one temporal) partial differential equations for a number of variables (typically velocity, temperature and salinity). The equations are solved using finite difference techniques. Small scale ocean eddies have a significant effect on the large scale circulation. Therefore large high resolution grids are required to resolve these features. Furthermore, the timescale for the adjustment of ocean circulation is long. Thus it is necessary to run the models for large numbers of timesteps in order to produce useful results.

Until recently oceanographers had to rely on crude parameterisations of the effects of eddies. It is only now becoming possible for oceanographers to run eddy resolving models and then only on the fastest computers. An example (with which one of the authors is involved) is the UK Fine Resolution Antarctic Model [13] which solves equations for over two million variables at every model timestep (40 minutes). Even though the FORTRAN 77 model code is highly vectorized it takes approximately 15 minutes (real time) to simulate a model day on a lightly loaded CRAY X-MP. Unfortunately this machine is usually shared with many other users. During the course of two years it has only been possible to integrate FRAM for 16 model years. Even so, many new and useful results have been obtained. Over the next few years it is hoped that a world ocean circulation model of similar or higher resolution to that in FRAM will be produced. The grand challenge in this field is to produce an accurate world ocean circulation model and to incorporate it into a climate modelling system. It has been estimated that an accurate eddy resolving ocean model would require a computer capable of 10^{15} Flops (the fastest present day supercomputers

Correspondence to: A.R. Clare, Computing Centre, University of East Anglia, Norwich NR4 7TJ, UK.

attain roughly 10^9 Flops) to perform useful experiments. At the present time it appears that such performance will only be attainable through the use of parallel architectures.

A number of researchers have implemented ocean models on contemporary parallel computers. Two significant examples, which deal with the implementation of global models, are Smith, Dukowicz and Malone [11] and Semtner and Chervin [10]. Smith et al. describe the use of a SIMD, distributed memory, 2048 processor CM-2 Connection Machine whereas Semtner and Chervin use a MIMD, shared memory, 4 processor CRAY X-MP/48. The implementation technique used by Smith et al. is domain decomposition (the ocean is divided up in longitude-latitude columns). The technique used by Semtner and Chervin may be viewed as a form of task farming where tasks are obtained by dividing the ocean into even sized longitude-depth slabs. By suitable utilization of the Cray's ability to read into and write from main memory concurrently with calculation work, processors are never delayed by the need to perform I/O operations. In practice Semtner and Chervin report attaining efficiencies of over 99%. Study of examples such as these reveals that in order to extract as much speed as possible from the underlying hardware, numerical techniques and programming styles must be aimed at specific architectural details.

In global ocean circulation modelling a major issue when designing a parallel program is how to handle continents, islands and ocean-floor topography, which combine to make the domain highly non-regular. A simple approach is to use a regular grid representation of the world combined with an array of 'start/stop indices' to mark the beginning and end of regions of water. These markers are then used to ensure that only points which correspond to areas of water are solved for. While being somewhat wasteful of store this technique is popular because it is relatively easy to program. On vector computers too much 'starting and stopping' can defeat the vectorization process. If the vector operation start-up time is long (machine specific) and the areas of land are small, it is more efficient to treat land as though it is ocean during calculations and to mask out the land's effect at the end of each iteration. This approach has been used, for example by Stevens [12], on CDC Cyber 205 and AMDAHL VP1200 vector supercomputers. Counter-intuitively, some times doing more calculations leads to a faster program!

A further major issue in writing efficient ocean modelling code is dealing with the sheer volume of data. In the FRAM project for example as much as 100 Megabytes of information had to be dumped every 10 model days plus 1 Megabyte every model day. Obviously a fast I/O system, which will not unduly delay the main calculation work, is required.

The remainder of this paper considers ocean modelling in the context of the Meiko Computing Surface. A simplified ocean model is described in Section 2 and its implementation on a Computing Surface in Section 3. Section 4 presents some experimental results and Section 5 gives our conclusions. The reader should note that this paper is an extension of work described in Clare and Stevens [4].

2. The sequential ocean modelling code

2.1 Three-dimensional models

Many of the three-dimensional ocean circulation models used over the past twenty years originate from the pioneering work of Bryan [3]. Over the years a number of enhancements to the model physics, the numerical scheme and the computer code have been made. A good description of the basic model and code structure can be found in Cox [5]. Semtner [9] describes the history of model development to that date.

2.2 A simplified model

Given the complexity of three-dimensional ocean circulation models such as that described by Cox [5], it was decided to produce a reduced physics version to test on the Meiko Computing Surface. The vertical structure, which is represented by a number of different levels in the original model, is replaced by a single active layer overlaying a much deeper passive layer. All the usual horizontal processes were retained. This well-known simplification, which removes the vertical dimension from the problem, results in a one layer reduced gravity model.

The model equations are

$$\frac{\partial u}{\partial t} + \Gamma(u) - fv = -\frac{g'}{a\cos\phi}\frac{\partial \eta}{\partial \lambda} + \frac{\tau^x}{\rho\eta} + A_m\left(\nabla^2 u + \frac{(1-\tan^2\phi)u}{a^2} - \frac{2\sin\phi}{a^2\cos^2\phi}\frac{\partial v}{\partial \lambda}\right), \quad (1)$$

$$\frac{\partial v}{\partial t} + \Gamma(v) + fu = -\frac{g'}{a}\frac{\partial \eta}{\partial \phi} + \frac{\tau^y}{\rho\eta} + A_m\left(\nabla^2 v + \frac{(1-\tan^2\phi)v}{a^2} + \frac{2\sin\phi}{a^2\cos^2\phi}\frac{\partial u}{\partial \lambda}\right), \quad (2)$$

$$\frac{\partial \eta}{\partial t} + \Gamma(\eta) = 0, \quad (3)$$

where

$$\Gamma(\mu) = \frac{1}{a\cos\phi}\left[\frac{\partial}{\partial\lambda}(u\mu) + \frac{\partial}{\partial\phi}(v\mu\cos\phi)\right]$$

and

$$\nabla^2(\mu) = \frac{1}{a^2\cos^2\phi}\left[\frac{\partial^2\mu}{\partial\lambda^2} + \frac{\partial}{\partial\phi}\left(\frac{\partial\mu}{\partial\phi}\cos\phi\right)\right].$$

The variables λ, ϕ, u, v, η, τ^λ, τ^ϕ represent longitude, latitude, zonal velocity, meridional velocity, layer thickness, zonal wind stress and meridional wind stress respectively. The radius of the Earth is a, the reduced gravity is g', the density of the active upper layer is ρ and $f = 2\Omega \sin\phi$ is the Coriolis parameter where Ω is the speed of angular rotation of the Earth. The coefficient for horizontal momentum mixing is A_m.

The model is discretized in the same way as Cox [5], using a staggered spatial grid. The finite difference scheme uses centred differencing in space and leapfrogging in time, which is energetically consistent. This results in three explicit prognostic equations for u, v and η, which can be simply stepped forward in time. A Robert time filter [1] is used to remove the computational mode associated with leapfrogging.

2.3 Overview of the sequential program

The basic structure of the more complex code was retained. The ocean is represented by the three arrays of real numbers: `U`, `V` and `ETA`. `U` and `V` represent the eastwards and northwards velocities respectively and `ETA` represents the surface layer thickness. Each of these arrays is three-dimensional. The first two dimensions represent the surface layer of the ocean as a rectangular grid of points. These dimensions are of size `IMT` by `JMT` (which are reconfigurable, program constants). The third dimension in `U`, `V` and `ETA` is needed so that the state of the ocean can be recorded for consecutive timesteps.

The program is built around a single main loop, each iteration of which simulates a timestep. Within this loop the calculation proceeds from the southernmost row (latitude line) of ocean points to the nothernmost. For each row, the subroutines `VELOC` and `THICK` are called. These subroutines calculate new values of the velocity and thickness for all the points in the current row. The calculation for a given row depends upon the current values in the row itself and on the values in the neighbouring north and south rows over the previous two timesteps. Hence, the arrays `U`, `V` and `ETA` are of size 3 in their third, temporal dimension.

Below is the fragment of code that implements the calculations involved in a single timestep.

```
      DO 380 J=2 , JMT-1
        IF (J .NE. JMT-1) CALL VELOC(J)
        CALL THICK(J)
380   CONTINUE
```

Rows 1 and `JMT` are always land and therefore not updated. Since the finite difference grid is staggered, row `JMT` – 1 of `U` and `V` also represents land and therefore `VELOC` is not called for this row.

This row update strategy is inherited from the implementation of the full three-dimensional model. Due to computer memory limitations it is often impossible to hold the full three-dimensional model grid in core. Hence the code has been written in a style which enables rows to be cycled through memory as required. Note also that by operating on complete rows at a time, `VELOC` and `THICK`, could be optimized for the original target, vector processing architectures.

3. Parallel implementation on the Computing Surface

Parallelism is achieved by using the well documented and obviously appropriate technique of

geometric decomposition. The structure of the parallel program is similar in many ways to that described by Barbieri, Bray and Garret [2], for example, which describes the implementation of a weather prediction program on the Computing Surface.

The arrays U, V and ETA are sliced up in their north-south, JMT dimension (so that the cuts run from west to east) and balanced across the available processors. The processors are simply connected in a ring. Each processor runs a copy of the same program but works with a different section of the ocean. Where necessary, different actions are programmed for different processors by branching on a processor's unique, numerical identifier.

A one-dimensional decomposition was chosen rather than a two-dimensional decomposition because it offered better potential performance. This derives from the transputer's ability to overlap communication with calculation [8]. Given an appropriate application and careful programming, the cost of any communication can be reduced to a constant (the set up time). In such circumstances it is better to have 2 neighbours (1D scheme) rather than 4 (2D scheme) in order to minimize the overall ratio of communication to calculation. Since it was possible to exploit this feature of the transputer for the ocean modelling application a 1D decomposition was chosen. A similar performance model to that described by Barbieri et al. [2] can be applied to the parallel program described here.

The decision to perform a one-dimensional decomposition simplified the coding work. For example, the main two prognostic sub-routines, THICK and VELOC, which had been written to operate on complete west-east rows of the ocean

Table 1
The results of the North Atlantic experiment.

System	Processors	Time (secs)	Mflops	Speedup	Efficiency
Amdahl VP1200 (Vector Mode)	1	63	122.48		
Amdahl VP1200 (Scalar Mode)	1	740	10.50		
CRAY X-MP	1	67	116.01		
DEC Station 5000	1	1141	6.76		
Sun SPARC 1	1	3784	2.04		
Sun SPARC 2	1	1585	4.87		
Inmos T800	1	31372	0.25	1.00	1.00
	2	16822	0.46	1.86	0.93
	3	11151	0.69	2.81	0.94
	4	8465	0.91	3.71	0.93
	5	6789	1.14	4.62	0.92
	9	3770	2.05	8.32	0.92
	11	3101	2.49	10.12	0.92
	20	1764	4.37	17.78	0.89
	25	1434	5.38	21.88	0.88
	33	1139	6.77	27.54	0.83
	50	835	9.24	37.57	0.75
	99	513	15.04	61.15	0.62
Intel i860	1	1480	5.21	1.00	1.00
	2	654	11.78	2.26	1.13
	3	501	15.40	2.96	0.99
	4	402	19.18	3.68	0.92
	5	345	22.33	4.29	0.86
	9	239	32.27	6.19	0.69
	11	215	35.77	6.87	0.62
	20	188	41.01	7.87	0.39
	25	177	43.39	8.33	0.33
	33	174	44.30	8.50	0.26
	50	178	43.29	8.31	0.17

at a time, were left untouched by the port to the parallel machine.

Given the simplicity of this particular ocean model and the associated geometric decomposition scheme, plus the fact the solving partial differential equations by finite differences must constitute one of the main uses of parallel computers at the present time, the coding work was felt to be unduly difficult. Two major factors led to difficulties:

(1) The authors lack of previous experience with parallel programming and in particular with using CSTools.
(2) It was felt that the facilities provided by CSTools stopped short of what was required to implement this application with ease.

For example, operations such as 'exchange data with neighbour', which occur often in parallel finite difference programs, had to be coded by hand. This required mastery of the complicated CSTools routines for handling asynchronous communications. A further example is that a special scheme had to be devised so processors would write their results (in the form of large binary data files) in a coordinated fashion. This is because CSTools has no built-in facilities to neatly handle the situation where many processors must each contribute a portion of a program's overall results. The scheme finally adopted for this latter problem, makes each processor write its results to a separate file and then uses additional, serial, utility programs to rework these files into whatever other form might be required. Finding efficient and elegant solutions to these and similar problems consumed a significant amount of time. In view of these experiences, the authors imagine that CSTools compares badly with some other parallel programming environments, such as Express [6] for example which appears to provide a higher level of support for the programmer.

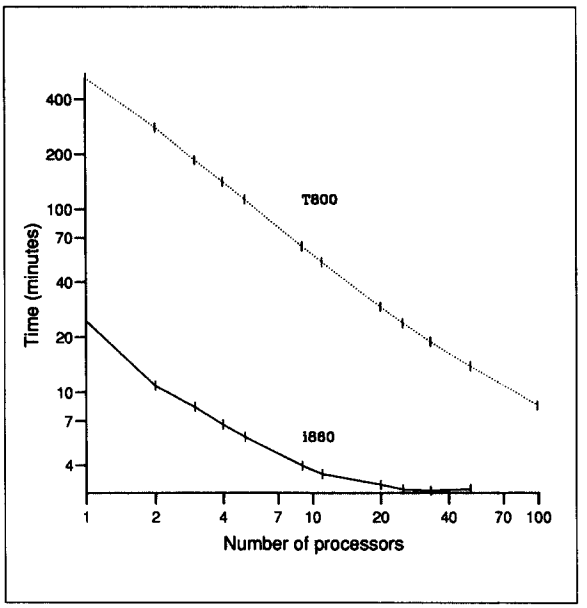


Fig. 1. The number of processors versus the time taken to complete a one year run of the North Atlantic model (log scales). The dotted line is for Inmos T800 processors and the solid line is for Intel i860 processors.

4. Performance experiments

4.1 The North Atlantic experiment

A one year simulation of an idealized North Atlantic model was used by Clare and Stevens [4] to compare the parallel and serial implementations. Since that time the authors have been allowed access to a larger array of Intel i860 processors. Furthermore some of the machines have had compiler upgrades, which has improved their preformance. A revised set of results are illustrated in *Table 1* and *Fig. 1*.

To summarize: only selected extracts from the multi-processor results are tabulated; the geometric decomposition consisted of 99 'grains' of work and therefore 99 processors was the maximum that could be fruitfully used on the problem; the timing figures given for a single transputer and a single i860 processor were obtained by using the original, sequential version of the program whereas the figures given for two or more processors were obtained using the parallel version; the results show that the two supercomputers (the CRAY and the Amdahl) far out-perform the other machines; the super-linear speedup exhibited by two i860's is probably caused by the single processor's performance being limited by the speed at which it can access its local main memory, rather than the rate at which it can perform floating-point calculations; the rapid drop in efficiency, as more i860's are used, is due to their performance being communication bound for this model.

The North Atlantic model, which is based on a 71 by 101 grid, was the author's first attempt at producing a parallel ocean modelling code. It was designed with transputers in mind and at a time when the authors did not envisage having access to i860's. To more fully explore the potential of the i860's a second, more computationally demanding model has been developed, as described below.

4.2 The Tropical Pacific experiment

As with the North Atlantic, modelling the Tropical Pacific is a classic oceanographic problem. Gill [7] provides a good introduction to the subject. The ocean basin extends from 32°S to 32°N and from 0°E to 100°E (note that while the latitudinal position of the ocean basin is crucial the longitudinal position is arbitrary). The horizontal resolution is 0.25°, which is approximately 27 km. The size of the arrays for the prognostic variables is 401×258. This provides 256 'grains' of work.

A constant easterly wind stress of 0.05 Nm^{-2} is specified at the ocean surface. The parameters A_m and g' have the values 2×10^3 $m^2 s^{-1}$ and 9.806×0.003 ms^{-1}. The ocean is initially at rest with the thickness of the upper layer set to 300 m. The model is then integrated forward in time for 7 weeks using 1 hour timesteps.

The results from this experiment are displayed in *Table 2* and *Fig. 2*. It is immediately apparent that this model is less communication bound when run on i860's. That is, the efficiency does not decay so rapidly as the number of processors is increased. Further investigation has revealed that a significant proportion of the reported i860 time, when the number of processors is large, is due to I/O. For example, the time spent writing results by the slowest processor in the experiment with 32 i860's is 31.2 seconds. This compares with less than one tenth of a second on both supercomputers, which are writing out 32 times more data than each i860. If the 32 i860's had a comparable I/O system, their overall Mflop rate might be expected to exceed 140 Mflops.

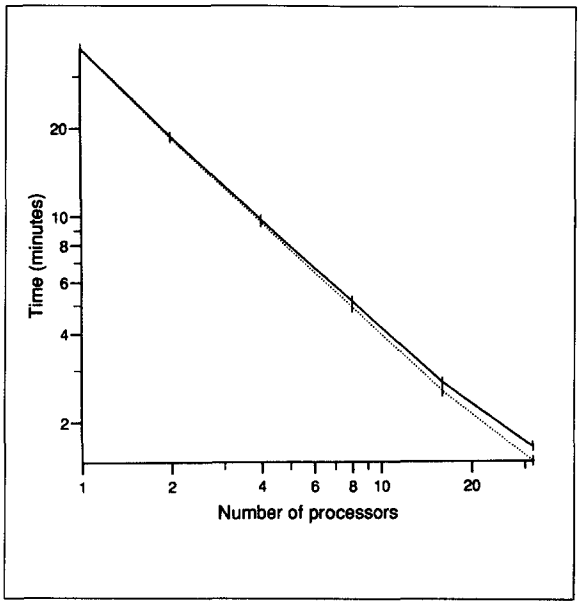


Fig. 2. The number of i860 processors versus the time taken to complete a seven week run of the Tropical Pacific model (solid line). The dotted line shows the time taken excluding I/O. Both axes are logarithmic.

Table 2
The results of the Tropical Pacific Experiment

System	Processors	Time (secs)	Mflops	Speedup	Efficiency
Amdahl VP1200 (Vector Mode)	1	49	267.42		
CRAY X-MP	1	92	142.43		
DEC Station 5000	1	2771	4.73		
Sun SPARC 1	1	6574	1.99		
Sun SPARC 2	1	2800	4.68		
Intel i860	1	2244	5.84	1.00	1.00
	2	1128	11.61	1.99	0.99
	4	595	22.02	3.77	0.94
	8	332	39.46	6.76	0.84
	16	179	73.18	12.54	0.78
	32	123	106.45	18.24	0.57

A surprising feature of the results is that the Mflop rate achieved by the Amdahl, when compared with the result from the North Atlantic experiment, is more than double and the total time required to solve this more complex problem is less. This is because the performance of the Amdahl is improved when vector lengths are longer. In addition there are less vector startup operations required in the Tropical Pacific model because it is run for fewer timesteps. The CRAY does not exhibit similar behaviour because its architecture is based on fixed length vector registers (64 words long) rather than arbitrary length vector pipelines.

5. Conclusions

This paper has considered the Computing Surface as a platform for ocean modelling work. The raw speed of the i860 processors is impressive and comparable with that of contemporary supercomputers. Unfortunately, the I/O system in the Computing Surface is not of a similar high standard and this significantly reduces overall performance for this application.

The simple technique of geometric decomposition is obviously appropriate for parallel ocean modelling when the method of finite differences is being used. In practice, coding work is complicated by the need to tailor code to fit specific computer architectures in order to maximize performance. For example, on the Computing Surface this typically requires exploiting the processors' ability to overlap communication with calculation. The author's look forward to the day when standard programming interfaces will simplify coding work while sustaining the high performance associated with hand-tuned programs.

Acknowledgements

The authors would like to thank: Andrew Grant and the Manchester Computing Centre for the use of their i860's; the Edinburgh Parallel Computer Centre for the use of the Edinburgh Concurrent Supercomputer (which was used to run code on 33 or more transputers); David Wallace, Ken Bowler and the EPCC again for allowing us time on the Maxwell Grand Challenge machine (which was used to test the code on more than 2 i860's). Anthony Clare is supported by the ISC HPDS initiative. Dave Stevens is working for the FRAM project under NERC grant GST/02/408.

References

[1] R. Asselin, 1972: Frequency filter for time integrations, *Mth. Wea. Rev.* 100 (1972) 487–490.

[2] A. Barbieri, A. Bray and P. Garret, Weather prediction using the computer surface, *Surface Noise* 6 (1992) 30–37.

[3] K. Bryan, A numerical method for the study of the circulation of the world ocean. *J. Comp. Phy.* 4, (1969) 347–376.

[4] A.R. Clare and D.P. Stevens, Porting a finite difference ocean circulation model to the Meiko computing surface, *Proc. Internat. Conf. Parallel Computing 91*, D.J. Evans, G.R. Joubert and H. Liddell, eds. (North-Holland, Amsterdam, 1992) 585–592.

[5] M.D. Cox, A primitive equation, 3-Dimensional model of the ocean, GFDL Ocean Group Tech. Rep. No. 1, 1984.

[6] J. Flower and A. Kolawa, Parallel programming with EXPRESS, *Surface Noise* 6 (1992) 18–27.

[7] A.E. Gill, *Atmosphere-Ocean Dynamics* (Academic Press, New York, 1992).

[8] INMOS, *The Transputer Databook* (Edition 2) (1989).

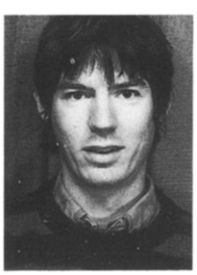
[9] A.J. Semtner, History and methodology of modelling the circulation of the world ocean, *Proc. NATO Advanced Study Institute on Advanced Physical Oceanographic Numerical Modelling*, ed. J.J. O'Brien (Reidel, Dordrecht, 1986) 23–32.

[10] A.J. Semtner and R.M. Chervin, A simulation of the global ocean circulation with resolved eddies, *J. Geophys. Res.* 93 (C12), (1988) 15502–15222.

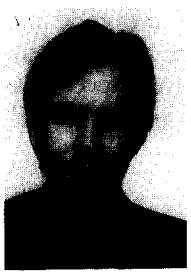
[11] R.D. Smith, J.K. Dukowicz and R.C. Malone, Parallel ocean general circulation modelling, submitted to *Physica D*, 1992.

[12] D.P. Stevens, A numerical ocean circulation model of the Norwegian and Greenland seas, *Progress in Oceanography* 27 (1991) 365–402.

[13] The FRAM Group, Initial results from a fine resolution model of the Southern Ocean, *EOS Trans. Amer. Geophys. Union* 72 (1991) 169, 174–175.



Dr. Anthony R. Clare was born in London, 1963. He received his B.Sc. (Hons.) in Computer Science from the University of East Anglia in 1984, and his Ph.D. (Evaluating Declarative Programming Languages) from the UEA in 1988. He has been Research Associate at the UEA from 1988-1990, and a Parallel Processing Support Officer at UEA since 1990.



Dr. David Stevens was born in Hastings, 1962. He has his B.Sc. (Hons.) in Mathematics from UEA in 1983, his M.Sc. (Theoretical mechanics) from UEA in 1984, his Ph.D. in Dynamical Oceanography from UEA in 1988. He is now a lecturer in Mathematics at UEA.