Implementation and performance of a parallel σ-coordinate ocean model

Ragnhild Blikberg
Department of Informatics
University of Bergen
Thormøhlens gt. 55
N-5020 Bergen

Jarle Berntsen
Department of Mathematics
University of Bergen
Johs. Bruns gt. 12
N-5008 Bergen

Institute of Marine Research
Nordnesparken 2
N-5024 Bergen-Nordnes

Abstract. This report is a documentation of the techniques applied for making a parallel version of the sigma-coordinate ocean model due to Berntsen, Skogen and Espelid [2], with recent modifications given by Berntsen [1]. Results are given for a quadratic and constant depth closed basin and the North Sea, both with 20 km and 4 km horizontal resolution, and the speedups obtained are discussed.

Contents

1 Introduction

In present studies of oceanic processes with numerical models the space and time resolution is typically smaller than necessary to resolve major processes. The choices of grid sizes and time steps are typically limited by the size of the available computers rather than by the needs for resolution. Processes on sub-grid scales are parameterized and the qualities of such parametrizations are questionable.

To be able to produce numerical results from ocean models that are affected by smaller scale processes, it is necessary to develop codes that may take advantage of todays parallel computers. This report is a documentation of the techniques applied for making a parallel version of the sigma-coordinate ocean model due to Berntsen, Skogen and Espelid [2], with recent modifications given by Berntsen [1]. Results and discussions of the obtained speedups are given.

The code is run on a 128 processor CRAY Origin 2000.

1.1 Applications

Regional ocean model like the present have been used in numerous oceanographic studies including:

- Studies of the circulation and geophysical processes in the North Sea, the Skagerrak, the Barents Sea and other regions of the ocean
- Studies of the effects of physical processes on the primary production in the ocean
- Studies of the effects of antropogenic inputs of nutrients on the primary production in the ocean
- Studies of effects of the ocean circulation on the distributions of fish eggs and larvae of major fish species like cod and herring
- Studies of the year to year variations of inflow of Atlantic water to regional seas like the North Sea and the Barents Sea
- Studies of the effects of major forcing functions like wind and fresh water runoff on the physical and biological variables
- Studies of possible effects of man made changes of the forcing fields
- Can the harvestable productivity of the ocean be substantially increased by artificial upwelling of nutrient rich deep water?

2 The σ-coordinate model

2.1 The basic variables and equations

The symbols used in the description of the model are given in Appendix A. The model assumes that the weight of the fluid identically balances the pressure (hydrostatic assumption), and that density differences are neglected unless the differences are multiplied by gravity (Boussinesq approximation). In the present study the following equations are used to describe the variables as functions of the cartesian coordinates x, y, z.

The continuity equation is

$$\nabla \cdot \vec{U} + \frac{\partial W}{\partial z} = 0, \tag{1}$$

and the Reynolds momentum equations are

$$\frac{\partial U}{\partial t} + \vec{U} \cdot \nabla U + W\frac{\partial U}{\partial z} - fV = -\frac{1}{\rho_0}\frac{\partial P}{\partial x} + \frac{\partial}{\partial z}(K_M\frac{\partial U}{\partial z}) + F_x, \tag{2}$$

$$\frac{\partial V}{\partial t} + \vec{U} \cdot \nabla V + W\frac{\partial V}{\partial z} + fU = -\frac{1}{\rho_0}\frac{\partial P}{\partial y} + \frac{\partial}{\partial z}(K_M\frac{\partial V}{\partial z}) + F_y, \tag{3}$$

$$\rho g = -\frac{\partial P}{\partial z}. \tag{4}$$

The pressure at depth z may be obtained by integrating equation (4) vertically

$$P = g\rho_0\eta + g\int_z^0 \rho(\acute{z})d\acute{z}. \tag{5}$$

The conservation equations for temperature and salinity are

$$\frac{\partial \rho}{\partial t} + \vec{U} \cdot \nabla\rho + W\frac{\partial \rho}{\partial z} = \frac{\partial}{\partial z}(K_H\frac{\partial \rho}{\partial z}) + F_\rho. \tag{6}$$

The horizontal diffusive terms F_x, F_y and F_ρ are all written in the form:

$$F = \frac{\partial}{\partial x}(A\frac{\partial F}{\partial x}) + \frac{\partial}{\partial y}(A\frac{\partial F}{\partial y}). \tag{7}$$

The algorithms used to define the vertical viscosity, K_M, and the vertical diffusivity, K_H, are typically problem specific. The same applies to the horizontal parameters A_M and A_H.

2.2 The σ-coordinate system

The basic equations have been transformed into a bottom following sigma coordinate system, see Phillips [7]. The independent variables (x, y, z, t) are transformed to (x^*, y^*, σ, t^*), where

$$x^* = x \quad y^* = y \quad \sigma = \frac{z - \eta}{H + \eta} \quad t^* = t. \tag{8}$$

σ ranges from $\sigma = 0$ at $z = \eta$ to $\sigma = -1$ at $z = -H(x, y)$. The basic equations after the transformation are given in Berntsen, Skogen and Espelid [2].

2.3 The numerical σ-coordinate model

The governing equations form a set of simultaneous partial differential equations which cannot be solved using known analytic methods. Therefore the equations have been discretized using finite difference methods. The horizontal finite difference scheme is staggered, and the Arakawa C-grid [6] has been used.

The model is written in Fortran 90 and the discrete versions of the state variables and parameters are gathered in a module, STATE, that may be addressed by all subroutines. The equations are stepped forward in time using the same time step for all equations. The method of fractional steps is applied. That is a sequence of subroutines is called to perform specific subtasks and update the corresponding variables in MODULE STATE in each timestep. After all subroutines are called the effects of all terms in the governing equations are included.

Descriptions of how different physical effects are included are given in Berntsen, Skogen and Espelid [2] and Berntsen [1].

3 Adjustment to equilibrium of a density driven flow

The model is set up for a quadratic and constant depth closed basin. The model domain is $0 \leq x \leq L_x$, $0 \leq y \leq L_y$ with $L_x = L_y = 200000m$. There are vertical walls at at all sides. The depth of the basin is equal to $-200m$ for all x and y.

3.1 Initial values

Let r define the metric $r = \sqrt{(x-100000)^2 + (y-100000)^2 + (1000z)^2}$. Initially we have a core of light(fresh) water in the center of the basin defined according to

$$\rho(x,y,z) = \rho_f + \Delta\rho \times tanh\left(\frac{r}{\Delta r}\right)$$

where $\rho_f = 1020kgm^{-3}$, $\Delta\rho = 5kgm^{-3}$ and $\Delta r = \sqrt{3} \times 50000m$. This means that the lengthscales of variation from the fresher water with density $1020kgm^{-3}$ water masses with density approaching $\rho_0 = 1025kgm^{-3}$ are approximately $50000m$ in the horizontal and $50m$ in the vertical.

Initially U and V are derived from the thermal wind relations

$$f\frac{\partial U}{\partial z} = \frac{g}{\rho_0}\frac{\partial \rho}{\partial y},$$

$$f\frac{\partial V}{\partial z} = -\frac{g}{\rho_0}\frac{\partial \rho}{\partial x},$$

with $U = V = 0$ at the bottom.

The vertical velocities, W, are initially equal to 0.

The initial free surface is determined from

$$\frac{\partial \tilde{\eta}}{\partial x} = -\frac{1}{\rho_0}\int_H^0 \frac{\partial \rho}{\partial x} dz,$$

$$\frac{\partial \tilde{\eta}}{\partial y} = -\frac{1}{\rho_0}\int_H^0 \frac{\partial \rho}{\partial y} dz,$$

assuming $\tilde{\eta}(0,0) = 0m$. The average $\hat{\eta}$ is then computed from

$$\hat{\eta} = \frac{1}{L_x \times L_y}\int_0^{L_x}\int_0^{L_y} \tilde{\eta} dx dy,$$

and finally an initial free surface which averaged over the basin will be zero is computed from $\eta = \tilde{\eta} - \hat{\eta}$.

3.2 Boundary conditions

At the free surface no wind stress

$$K_M \frac{\partial \vec{U}}{\partial z} = 0, \qquad z = \eta$$

is assumed. A quadratic law for the bottom stress

$$K_M \frac{\partial \vec{U}}{\partial z} = c_d (U_b^2 + V_b^2)^{1/2} \vec{U}_b, \qquad z = H(x, y),$$

where

$$c_d = \max[0.0025, \frac{\kappa^2}{(ln(z_b/z_0))^2}]$$

and z_b is the distance of the nearest grid point to the bottom, is applied. The von Karman constant $\kappa = 0.4$. The bottom roughness parameter $z_0 = 0.01m$.

For the closed horizontal boundaries no normal flux is assumed. No friction (slip) at the lateral boundaries is assumed.

3.3 Numerical results

The flow in the present study is density driven and because of the presence of diffusivity and viscosity terms, $A_M = A_H = 10000m^2s^{-1}$ and $K_M = K_H = 0.01m^2s^{-1}$, and the bottom friction, the flow will gradually decrease and because there are no external forcing the steady state solution will be that there is no flow. In Figure 1 the transports through a vertical section going from $(x, y) = (L_x/2, L_y/2)$ to $(L_x, L_y/2)$ are given. To propagate the numerical solution 150 hours in time with $80 \times 80 \times 55$ grid cells and timesteps of 360 seconds in 3D and 22.5 seconds in 2D takes approximately 4.5 hours on one of the processors on the Cray Origin 2000, and we still want to solve the problem with a higher resolution in space and time.

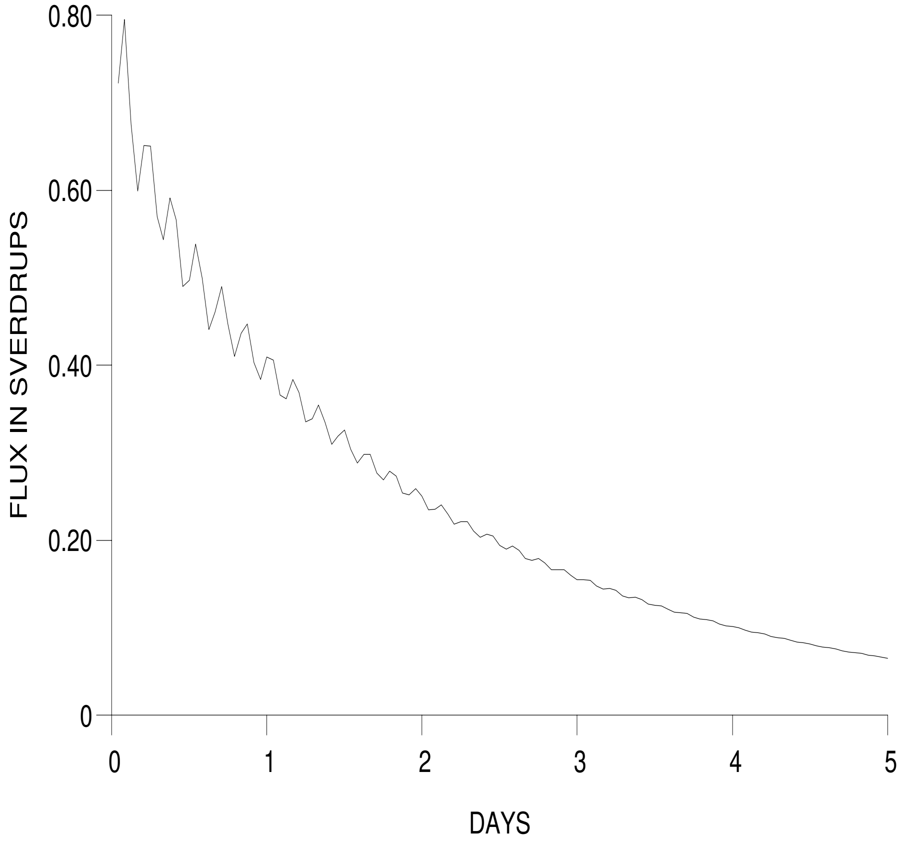

Figure 1. The transports in Sverdrups ($1\mathrm{Sv} = 1000000 m^3 s^{-1}$) through a vertical section from $(x, y) = (L_x/2, L_y/2)$ to $(L_x, L_y/2)$ towards the x-axis.

4 The North Sea experiment

A number of applied studies for regional oceans like the North Sea, the Barents Sea or the Norwegian Sea are still performed with approximately 20km horizontal resolution and from 10 to 30 vertical layers, see for instance Smith *et al.* [8]. A new time-split σ-coordinate model is presented in Berntsen [1] and applied in a sensitivity study for an extended North Sea. Horizontally the model domain is discretized in a 20×20 km grid with 80×70 gridpoints, see Figure 2. In the present study the focus is on the potential for parallelization of the code and verticallt 21 layers is applied. The model is run with boundary conditions and initial values from Engedahl *et al.* [4]. It is forced with atmospheric forcing provided by DNMI [3]. Four components of the tide, M_2, S_2, O_1 and K_1, are forced through the open boundaries with tidal fields provided by the Norwegian Meteorological Institute. The inflow from the Baltic is implemented after an algorithm due to Stigebrandt [9]. The model is run with river runoff from the main rivers. A typical output from the model is given in Figure 3. See Berntsen [1] for details on the model, the model set up and model results.

The model is recently also implemented for a $310 \times 310 \times 21$ grid with 4 km horizontal resolution, see Figure 4.

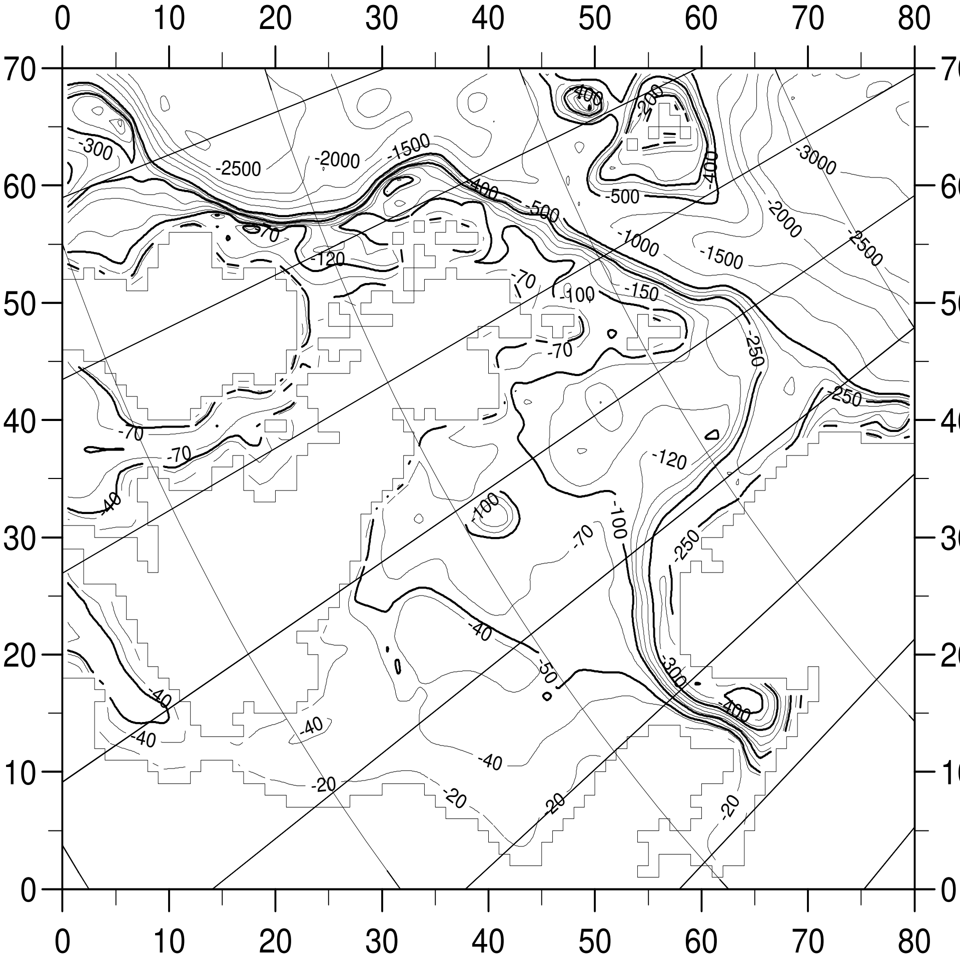

Figure 2. Bottom topography of the North Sea model area.

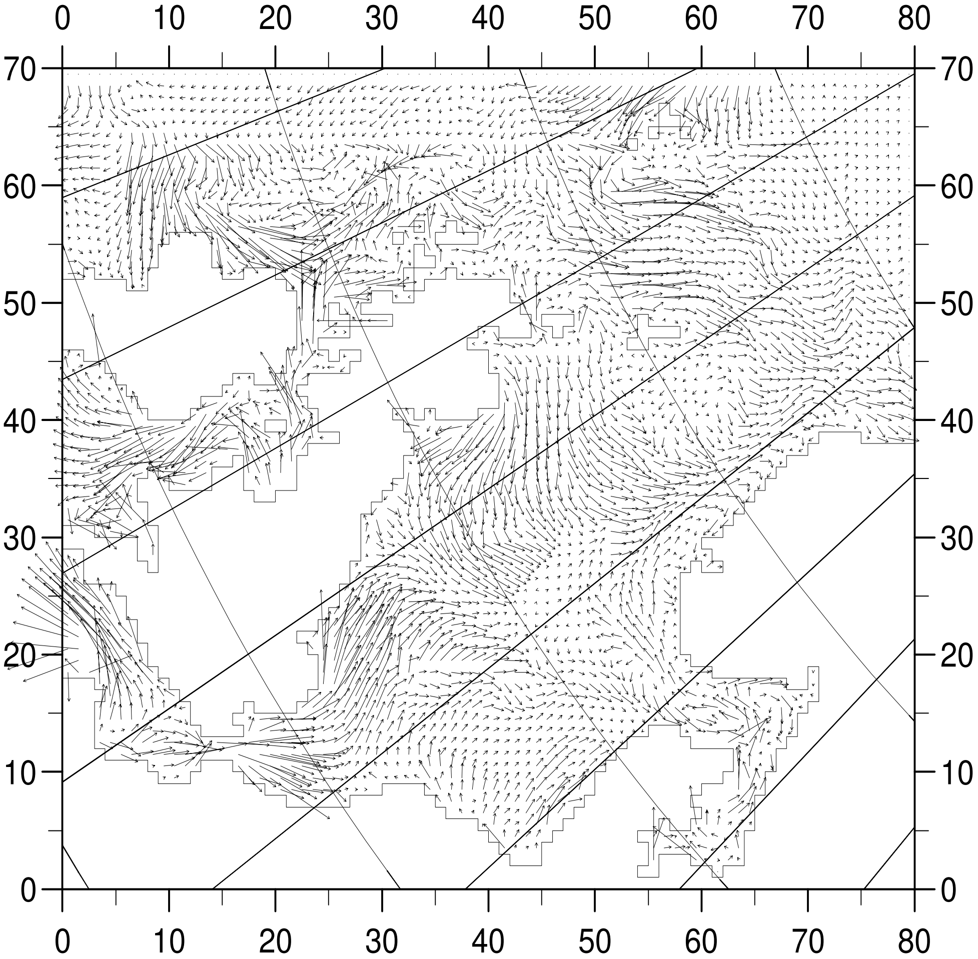

Figure 3. Typical flow pattern in a tidally driven North Sea.

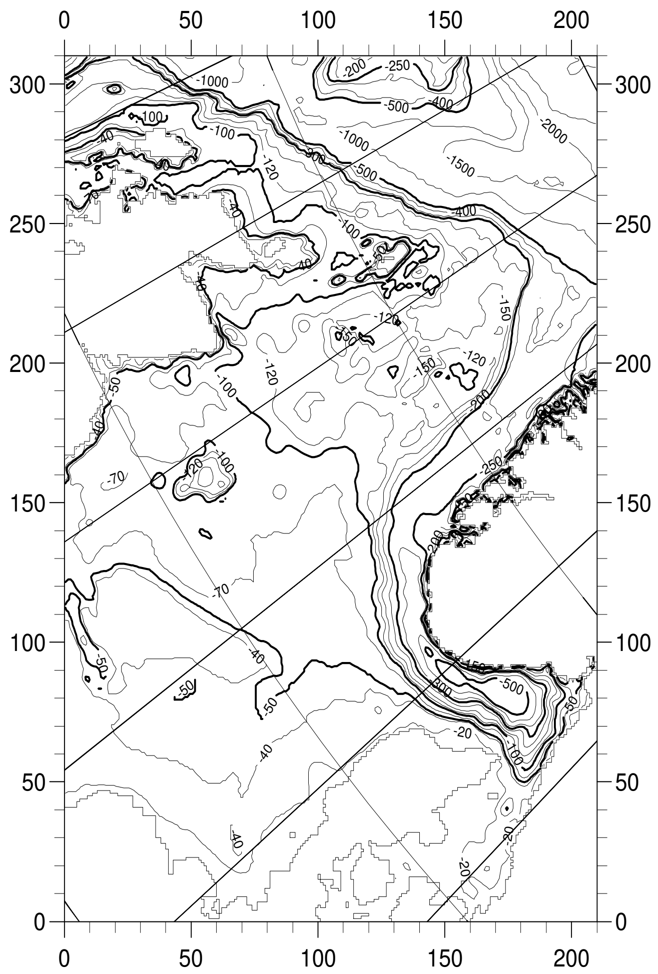

Figure 4. Flow pattern in a tidally driven North Sea.

5 Parallelization

In this chapter the method used for parallelizing the σ-coordinate model on a 128 processor CRAY Origin 2000 will be described, and the results of this parallelization will be given and discussed.

All test runs are done on the 7.2.1.1 Fortran 90 compiler and the compiler options `-O3 -mips4 -OPT:roundoff=0` have been used for sequential optimization. The `-O3` option does aggressive optimization, the `-mips4` option specifies the MIPS4 instruction set, while the `-OPT:roundoff=0` option takes away the compiler's permission, through `-O3`, to do roundoff-altering optimizations.

All CPU-times listed in the tables are measured at the master thread.

5.1 Power Fortran Accelerator, PFA

The tool used for parallelization of the ocean model is PFA (Power Fortran Accelerator), see SGI [5]. This is a preprocessor that analyses the data dependencies in loops, and parallelizes automatically loops containing no data dependencies. When it encounters problems with data dependencies, it tells the user what is causing this. The user can then, where possible, modify the code or insert directives to assist the compiler in parallelizing the code. The parallelization can also be done manually. This kind of parallelism is called Shared Memory Parallelism (SMP).

PFA is activated by compiling and linking by `-pfa`.

5.2 Automatically parallelization

For the codes referred to in this paper, PFA parallelized all parallelizable loops automatically. This was thanks to the new version of PFA. For older versions, it has been necessary to insert directives.

For the quadratic and constant depth closed basin code 99.4% for the $42 \times 42 \times 29$ grid code and 99.8% of the $82 \times 82 \times 55$ grid code was parallelized. 95% of the North Sea 20×20 km code and 97% of the North Sea 4×4 km code was parallelized. The reason for the difference in parallelisme between the basin case and the two North Sea cases is that the North Sea cases apply some routines that the basin case does not need, and these routines are impossible to parallelize. These routines are typically applied in more realistic simulations to read input data on atmospheric and tidal forcing, to produce model outputs or to update the numerical solutions at the open boundaries.

5.3 Results

For grid-size $IM \times JM \times KB$, the most time-consuming loops in this model are at the form:

```
DO J = 2,JM-1
   DO I = 2,IM
      IF (water) THEN
         DO K = 1, KB-1
            < ...  >
         END DO
      END IF
   END DO
END DO
```

The following subsections give the result of parallelizing the quadratic and constant depth closed basin experiment, and the two different North Sea experiments, all for different grid-sizes.

Explanation of the tables:
NT = Number of threads
Time = CPU-time (seconds) for the master thread
Speedup = (Time for NT = 1) / (Time for NT)
Eff (%) = Efficiencies in percent = $(Speedup/NT) \times 100$

5.3.1 Quadratic and constant depth closed basin

In the tables below the results for the quadratic basin are shown for the $42 \times 42 \times 29$ grid and the $82 \times 82 \times 55$ grid.

NT	Time	Speedup	Eff (%)
1	100	1.0	100
2	66	1.5	75
4	45	2.2	55
8	35	2.9	36
12	37	2.7	23
16	34	2.9	18
20	36	2.8	14

Table 1. Basin, $42 \times 42 \times 29$ grid.

NT	Time	Speedup	Eff (%)
1	107	1.0	100
2	55	1.9	97
4	30	3.6	90
8	17	6.5	81
12	13	8.5	71
16	10	10.7	67
20	9	11.9	60

Table 2. Basin, $82 \times 82 \times 55$ grid.

For this code, the work-balance between the loop-iterations is almost perfect. For each loop-iteration the IF-construction is fulfilled the same number of times. Therefore the default scheduling (SGI [5]) of loop-iterations, SIMPLE, which divides the loop-iterations among threads by dividing them into contiguous pieces and assigning one piece to each thread, is the best possible way of scheduling. This explains the good speedup, at least for the finest mesh. As we can see, the scaling gets better as the mesh gets finer. When the mesh is refined, the work in each outer loop-iteration increases. Then the proportional time spent in parallel administration gets smaller compared to the time spent on computation.

5.3.2 North Sea 20×20 km grid

In more realistic cases the model area is irregular like for the North Sea 20×20 km model, see Figure 2. In Table 3, the results are shown for the $80 \times 70 \times 21$ grid.

NT	Time	Speedup	Eff (%)
1	51.5	1.0	100
2	34.5	1.5	75
4	21.4	2.4	60
8	14.1	3.7	46
12	12.3	4.2	35
16	11.6	4.4	28
20	10.6	4.9	25

Table 3. North Sea 20×20 km, $80 \times 70 \times 21$ grid.

For this code, the work-balance between the loop-iterations is far from perfect. For each loop-iteration the IF-construction is fulfilled a variable number of times. Therefore the default scheduling, SIMPLE, is not necessarily the best way of scheduling.

In the GSS scheduling method the piece size of loop-iterations varies depending on the number of iterations remaining. By parcelling out relatively large pieces to start with and relatively small pieces towards the end, the system can achieve good load-balancing while reducing the number of entries into the critical section.

After some experimenting, it turned out that the GSS scheduling was beneficial for the SUPERBEEF-routine, which is one of the time-consuming routines (11%). The other scheduling methods that have been tried did not improve the speedups. No scheduling method improved the speedups for the remaining subroutines. The reasons for this will be discussed in section 5.4.3.

The results for the whole code, when GSS scheduling is applied to the SUPERBEEF routine, is shown in Table 4. Comparing the results in Table 3 and 4, it can be seen that using the scheduling GSS instead of the default scheduling, SIMPLE, did improve the speedups slightly for upto 12 threads.

NT	Time	Speedup	Eff (%)
1	51.5	1.0	100
2	32.8	1.6	80
4	20.3	2.5	63
8	13.5	3.8	48
12	12.5	4.1	34
16	11.6	4.4	28
20	11.3	4.6	23

Table 4. North Sea 20×20 km, $80 \times 70 \times 21$ grid, GSS scheduling in SUPERBEEF.

5.3.3 North Sea 4×4 km grid

For the regular area the speedups depended on the grid-size. In Table 5 the results for the North Sea 4×4 km grid with $210 \times 310 \times 21$ grid cells are given.

NT	Time	Speedup	Eff (%)
1	589	1.0	100
2	373	1.6	80
4	205	2.9	73
8	124	4.8	60
12	97	6.1	51
16	84	7.0	44
20	67	8.8	44

Table 5. North Sea 4×4 km, $210 \times 310 \times 21$ grid.

Also for the North Sea the grid-size seems to matter. Compared to the results in Table 3, the speedups here are much better.

The GSS scheduling was then added to study whether SUPERBEEF-routine to see if the speedup could be improved. The results are:

NT	Time	Speedup	Eff (%)
1	589	1.0	100
2	354	1.7	85
4	192	3.1	78
8	120	4.9	61
12	89	6.6	55
16	74	8.0	50
20	65	9.1	46

Table 6. North Sea 4×4 km, $210 \times 310 \times 21$ grid, GSS scheduling in SUPERBEEF.

Also in this case, like in the North Sea 20×20 km grid experiment, the GSS-scheduling used on SUPERBEEF seemed to improve the results slightly. Other types of scheduling were also tried without success.

5.4 Discussion

In this section we will try to explain why the obtained speedup is less than ideal speedup.

5.4.1 Amdahls law

Taking into account that some parts of the code are not parallelized, we find that the maximum theoretical speedup that can be achieved, not taking data-layout into account, is

$$S = \frac{1}{1 - p + \frac{p}{NT}} \tag{9}$$

where S = *speedup*, p = *fraction of code which is parallelized*, and NT = *number of threads*.

For ideal speedup, $NT = \frac{T_{seq}}{T_{par}}$, with sequential CPU-time given by T_{seq} and the parallel by T_{par}, Amdahls law (9) can now be written as

$$S = \frac{1}{1 - p + p\frac{T_{par}}{T_{seq}}} \tag{10}$$

5.4.2 Work-balance among the loop-iterations

In the basin code, each iteration in the parallelized loops uses about the same CPU-time i.e. it is work-balance among the loop-iterations. In the following formulas it is assumed that each iteration uses exactly the same CPU-time.

If the number of iterations cannot be divided equally among the threads

$$\frac{T_{par}}{T_{seq}} \sim \frac{M}{It} \tag{11}$$

where $M = \lceil \frac{It}{NT} \rceil$ and *It=total number of iterations in the parallelized loop*. Inserting this into (10), gives maximum theoretical speedup corrected for Amdahls law and load imbalance:

$$S = \frac{1}{1 - p + p\frac{M}{It}} \tag{12}$$

For the $82 \times 82 \times 55$ basin case the percentage of code which is parallelized is 99.4%, giving $p = 0.994$. This is estimated by profiling. The most time-consuming loops are parallelized in the Y-direction, this means that $It = 80$ (check the typical loop in section 5.3). In Figure 4, the ideal speedup, the speedup due to Amdahls law, speedup corrected for Amdahls law and load-imbalance, and the obtained speedup are shown graphically. Note that the speedup due to Amdahls law and speedup corrected for Amdahls law and load-imbalance do not differ in the points where NT is a factor of It.

Then what is causing this gap between the maximum theoretical possible obtainable speedup due to Amdahls law corrected for load-imbalance and the obtained speedup? Most likely it is due to the parallel overhead, which is the time the runtime system needs to administrate and synchronize multiple threads.

5.4.3 Work-imbalance among the loop-iterations

If the different loop-iterations do not use the same CPU-time, it is work-imbalance among the loop-iterations. For this phenomenon, the North Sea 20×20 km grid case is picked as an example. Divide a typical parallelized loop, like the first big loop in SUPERBEEF, in two pieces, one from J = 2 , 35 and another one from J = 36 , 69. Then count the number of times the IF-construction is fulfilled (hits) for each part, measure the CPU-times for the two pieces, and you will get the result showed in Table 7.

Area	Hits	Time
J = 2,35	1192	1.94
J = 36,69	2106	3.39

Table 7. The area counted on, number of hits for this area (one loop) and CPU-time (seconds) for this area (total run).

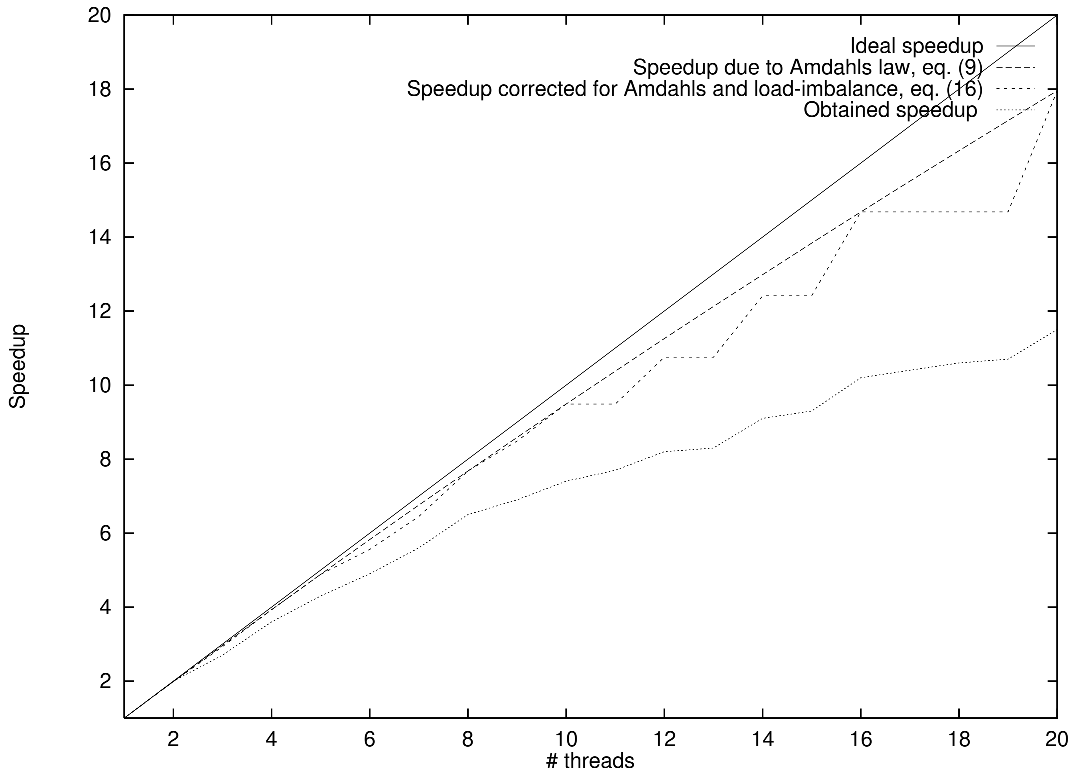

Figure 4. Ideal speedup, speedup due to Amdahls law, speedup corrected for Amdahls law and load-imbalance, and obtained speedup for the $82 \times 82 \times 55$-grid, basin experiment.

Examining the hits- and time-fractions gives

$$\frac{1192}{2106} = 0.566 \tag{13}$$

$$\frac{1.94}{3.39} = 0.572 \tag{14}$$

These numbers show that the time spent in a loop is proportional to the number of times the IF-construction is entered. Therefore, in the following examination, it is sufficient to know the number of hits.

It can also be seen from Table 7 that the maximum theoretical possible obtainable speedup for this loop with the SIMPLE scheduling is

$$\frac{1192 + 2106}{2106} = \frac{3298}{2106} = 1.57 \tag{15}$$

This formula can be extended for more than two threads:

$$S = \frac{sum(hits)}{max(hits)} \tag{16}$$

where S = *speedup for NT threads*, and *hits = number of times the IF-construction is entered*

Assume that 100% of the code contains parallelized loops like this. If the same counting is done for dividing in 3 to 20 pieces, $p = 0.95$, $M = \text{Max(Hits)}$ and $It = 3298$ are inserted in formula (12), we get the result showed in Figure 5.

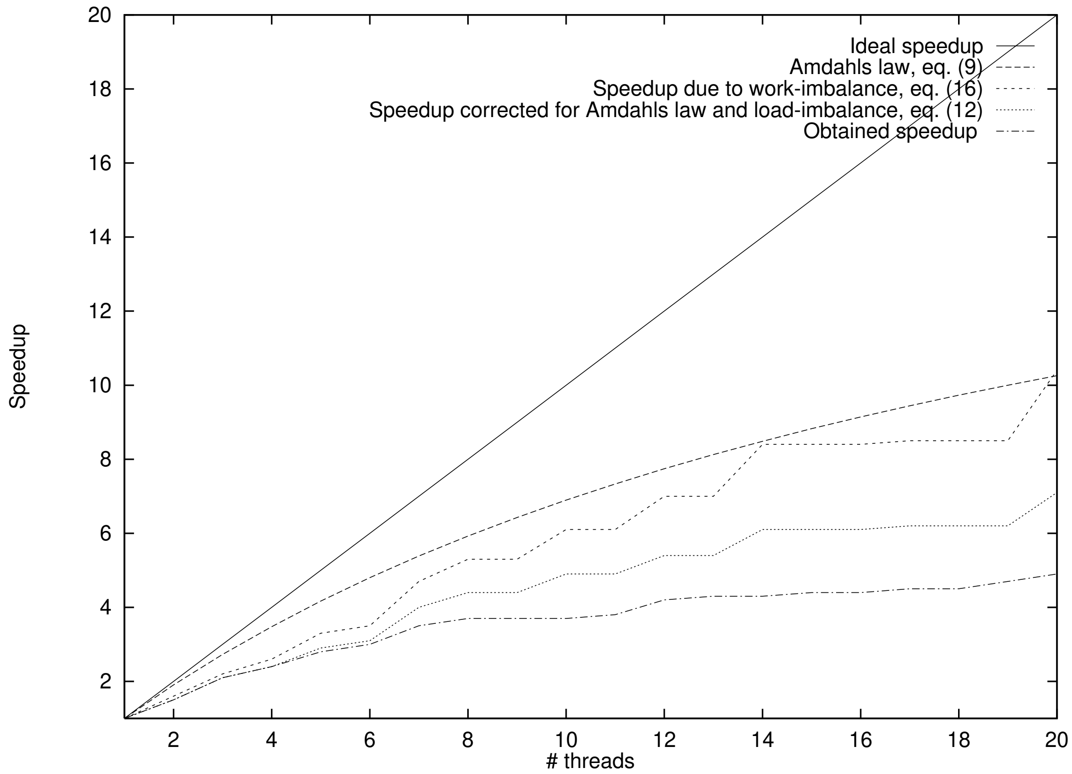

Figure 5. Ideal speedup, speedup due to Amdahls law, speedup due to the work-imbalance between the loop-iterations and speedup corrected for Amdahls law and load-imbalance for the North Sea 20 × 20 km code with gridsize 80 × 70 × 21 and SIMPLE scheduling if 95% of the code is parallelized.

Figure 5 indicates that Amdahls law and load-imbalance explain most of the difference between the ideal speedup and the obtained speedup. The remaining difference is probably du to parallel overhead.

It should be mentioned that for some ocean models, the IF-construction in section 5.3 is removed, and the same computations are performed for land as well as for water. For these ocean models, the work-balance between the iterations will be almost perfect, and the speedups will be much better than those we got for the North Sea experiments.

However, if the IF-construction was removed for our ocean model, the CPU-time would still be the same because the computations would then be performed also in land cells. If the CPU-times for all the threads were added, the total CPU-time for this version would be much larger than for our old version.

In contrast to the version not differing between land and water, the ocean model applied in the present studies has a potential when it comes to reducing the CPU-time for parallel execution. If the code is changed to give better work-balance among the loop-iterations, or if a good way of scheduling the loop-iterations is found, it may be possible to increase the speedup and decrease the CPU-time.

Then why did not a different scheduling improve the result for the North Sea cases? This was tried for different loops and was only giving profit for the SUPERBEEF routine, see Table 4 and 6.

If nothing else is specified, the data are distributed after the "first touch" method, meaning that the data are stored at the thread that first uses it. From Figure 5 it can be observed that the difference between the maximum theoretical possible obtainable speedup, not taking data-layout into account, and obtained speedup is not very large. This indicates that the data already have a good data-layout. Therefore, changing the scheduling could make the data-layout worse.

Besides, in scheduling, as everywhere else, there is no "free lunch". While providing a better load-balance, the GSS scheduling is not without cost. Book-keeping and administration increase, and unless the computational work of each iteration is large enough, we can not expect an overall improvement.

5.4.4 Data distribution

If data are stored at the node that the thread is using, data-transfer is minimized. But we do not know where the data are stored. However, the data can be distributed by using directives. Tests of this did not give improve the speedups. There may be two reasons:

1. The data is already well distributed by default.

2. The code is not data intensive. Every time a data is fetched from memory, many operations are performed with the same data before a new data is needed.

It has already been mentioned that the data-layout probably is acceptable by default. This is probably the reason why the data distribution directives did not help either.

6 Conclusions

In this paper the technique applied for parallelizing the σ-coordinate ocean model on a CRAY Origin 2000 ha been described, and the results obtained are discussed.

There are various techniques that may be applied in order to port numerical codes to parallel computers. In the present study it has been shown that by using a very simple techniques, such as compiling by `-pfa`, it was possible to obtain a speedup close to 12 on 20 processors on the CRAY Origin 2000.

The results showed that, due to the work-balance between the iterations, the speedup was much better for a regular area compared to an irregular area. Changing the scheduling methods had only marginal effects on the speedups. Data distribution directives did not help at all, most probably because the data already was well distributed.

The results also showed that the speedups increased with a finer mesh both for a regular and an irregular area.

For ocean modelling this opens the possibility to study major problems using a much finer mesh and thus including the effects of smaller scale processors in our studies.

A List of symbols

$\vec{U} = (U, V)$	Horizontal velocities in $x-$ and y-direction respectively
W	Vertical velocity in the z-coordinate system
η	Surface elevation
H	Bottom static depth
P	Pressure
ρ	In situ density
K_M	Vertical eddy viscosity
A_M	Horizontal eddy viscosity
C_M	Dimensionless horizontal eddy viscosity coefficient
K_H	Vertical eddy diffusivity
A_H	Horizontal eddy diffusivity
ρ_0	Reference density
g	Gravity
f	The Coriolis parameter
$\vec{U_b} = (U_b, V_b)$	Horizontal velocities at the bottom
C_D	Bottom drag coefficient
κ	The von Karman constant
z_0	Bottom roughness parameter

References

[1] J. Berntsen. A sensitivity study of a baroclinic model for the North Sea with focus on the Utsira-Orkenys transect, 1998.

[2] J. Berntsen, M.D. Skogen, and T.O. Espelid. Description of a σ-coordinate ocean model. Technical Report Fisken og Havet Nr. 12, Institute of Marine Research, 1996.

[3] L.I. Eide, M. Reistad, and J. Guddal. Database av beregnede vind og bølgeparametre for Nordsjøen, Norskehavet og barentshavet, 1985. DNMI rapport.

[4] H. Engedahl, B. Ådlandsvik, and E.A. Martinsen. Production of monthly mean climatological archives of salinity, temperature, current and sea level for the Nordic Seas. *J. Mar. Syst.*, 14:1–26, 1998.

[5] Silicon Graphics Inc. MIPS Fortran77 Programmer's Guide. 1996.

[6] F. Mesinger and A. Arakawa. Numerical methods used in atmospheric models, volume i, 1976. Garp Publication Series No. 17.

[7] N.A. Phillips. A coordinate system having some special advantages for numerical forecasting. *J. of Meteorology*, 14:184–185, 1957.

[8] J.A. Smith, P.E. Damm, M.D. Skogen, R.A. Flather, and J. Pätsch. An investigation into the variability of circulation and transport on the North-West European shelf using three hydrodynamic models. *Deutsche Hydrographische Zeitschrift*, 48:325–348, 1996.

[9] A. Stigebrandt. Barotropic and baroclinic response of a semi-enclosed basin to barotropic forcing of the sea. In H.J. Freeland, D.M. Farmer, and C.D. Levings, editors, *Proceeding of the NATO Conference on Fjord Oceanography*. Plenum Press, New York, 1980.