THE METEOROLOGICAL OFFICE UNIFIED MODEL FOR DATA ASSIMILATION, CLIMATE MODELLING AND NWP AND ITS IMPLEMENTATION ON A CRAY Y-MP

R.S.Bell and A.Dickinson
Meteorological Office
Bracknell, England

1. INTRODUCTION

The Met Office, as a national meteorological service, carries out operational weather forecasting using a variety of models of the atmosphere. These range from a global model with a horizontal resolution of 90Km down to a regional mesoscale model covering just the British Isles which uses a gridlength of 15Km. Because of the time critical nature of the work and the amount of data processing involved, such numerical weather prediction (NWP) models have always demanded the largest most powerful computers available. Nowadays, this means using a multi-processor supercomputer.

The Met Office is also at the forefront of the international effort in climate research, through the setting up of the Hadley Centre for Climate Prediction and Research. The growing international concern about the threat of global warming due to the steady increase in atmospheric concentrations of greenhouse gases has led to renewed efforts to predict the effects of climate change. This research requires coupled models of the complete atmosphere-ocean system to be integrated forwards a century or more ahead. Although these models use much lower spatial resolutions than forecast models, the cost of one such experiment equates to many thousands of hours on the most powerful computer systems.

Cray Y-MP 8/32 computer system has recently been installed at the Met Office and is currently being used to test the next generation NWP system. A second computer, with similar features and capability to the Y-MP 8/32, will be acquired early in 1991 to be used primarily for climate change prediction. A new unified climate-forecast model is being developed for the Cray Y-MP covering a wide range of ocean and atmosphere applications. Section 2 describes the scope of this model. In section 3 some general points on code design and multitasking strategy are discussed. The remaining sections describe more specific problems encountered whilst

designing a parallel processing strategy for the main components of the model, namely the data assimilation component, the atmosphere dynamics and physical parametrizations, and the ocean component.

2. THE UNIFIED MODEL

2.1 Scope

To a first approximation, forecast models and the atmospheric part of climate models may be viewed as just different resolution versions of the same program. In forecast mode the highest resolution that can be reasonably run in the available time is used. In climate studies there is a trade off between the accuracy in the representation of features and the length of time for which the model can be run. Therefore coarser resolution models are typically used. Table 1 gives examples of the resolutions being used at the present time. In addition to its flexibility with respect to resolution, the unified model allows users the option of including alternative packages if the need arises. For example, in climate configuration, more detailed physical parametrizations might be adopted, the impact of which are unimportant on the short timescales typical of NWP configurations.

Configuration	Levels	Points E-W	Points N-S	Resolution
Global short range	20	288	217	90km
Regional short range	20	229	132	50km
Monthly forecast	20	192	145	135km
Seasonal forecast	20	144	109	180km
Upper atmosphere	42	96	73	270km
Climate forecast	20	96	73	270km

Table 1. Dimensions of the atmospheric models used on the Cray Y-MP for various applications

The model can be run in three modes, atmosphere only, ocean only, or as a coupled atmosphere-ocean model. In each mode a run consists of an optional period of data assimilation followed by a forecast. The model may be global or limited in either horizontal or vertical extent. In the latter cases, boundary values of the prognostic variables must be given for the period of the integration.

At the control level of the model, large increases in complexity have been introduced by the need to build a single model with a large number of potential applications. Output fields are required to couple together the ocean and atmospheric components of the coupled model system. The global model runs must be capable of outputing lateral boundary information for a regional limited area model which may be located anywhere on the globe. The regional model is required to output boundary information for subsequent use by a separate mesoscale model run. All atmospheric configurations must be able to provide forcing information for an ocean wave and swell forecasting model.

The other aspect of output processing is the production of model prognostic, diagnostic and derived fields for subsequent perusal by the user, be it forecaster or research scientist. In excess of 1000 fields are provided to the forecaster from a global 6 day forecast. In this mode, many fields are required as frequently as 3 hourly during the run, whilst in climate mode output may only be required at 10 day intervals, although that 10 day field might be a mean of 10 component daily fields.

A final feature of the control system worth mentioning is its ability to automatically release other jobs at some predetermined interval. It would be entirely inappropriate if, during a long climate run of many model years, the model had to be stopped at intervals to process output.

2.2 Formulation

2.2.1 Data assimilation

The data assimilation scheme adjusts the model atmosphere towards observations, providing both initial fields for the operational forecasts and analyses from which global climatologies may be generated. At the Met. Office, data assimilation is achieved by an iterative or "nudging" procedure (Lorenc et al, 1991). In this procedure, as with any dataassimilation scheme, we seek to extract information from the observations and add it to our current knowledge of the atmosphere as represented by the model. At each timestep, we calculate differences of the observations from model values. The analysis increments, which are required to nudge the model state towards the observations, are a linear combination of weighted observation increments, where the weights are given by a rather complicated function of displacement in space and time, and also of observation density and error.

2.2.2 Atmosphere model

The atmosphere part of the model integrates the hydrostatic primitive equations of motion, using a split-explicit finite difference method on an Arakawa 'B' grid (Cullen et al, 1990). For all configurations of the model a spherical polar coordinate system is used. When the model is run with a limited horizontal extent, a coordinate pole must be chosen as far removed from the area of interest as possible. Hybrid vertical coordinates are used, which can be specified as pressure or sigma or a combination of the two. The integration scheme conserves mass, mass-weighted potential temperature and moisture, and angular momentum. The gravity wave terms are integrated using a forward-backward scheme which is second order accurate in space and time. The horizontal advection uses a two-step Heun scheme with fourth-order accuracy at low wind speeds. The scheme is automatically reduced to second-order accuracy where the winds are strong in order to ensure stability.

Computational stability is maintained in global versions of the model by Fourier filtering at high latitudes. The filtering performed is automatically adjusted according to the local wind speed. In addition, horizontal diffusion is applied with a coefficient depending on wind speed, and the vertical loss of total energy from the atmosphere arising from diffusion terms is diagnosed and returned in the form of a globally uniform heat source.

The model includes multilayer soil temperatures and a soil moisture prediction scheme. Different soil types are specified, and used to determine the surface albedo. A model of the vegetation canopy is included. Moisture can be retained in the canopy or transferred to the soil or atmosphere. Different vegetation types can be specified. Snow depth is predicted and used in the calculation of albedo. Vertical turbulent transport in the boundary layer depends on the local Richardson number. The presence or absence of cloud is taken into account in calculating the transport coefficients.

Large-scale clouds are represented by their liquid water (or ice) content. The total optical thickness of the clouds is taken into account in the radiation calculations. Large-scale precipitation is calculated in terms of the water or ice content of the cloud. For liquid cloud, the scheme

represents the coalescence and accretion processes as precipitation falls through the cloud; frozen cloud starts precipitating as soon as it forms. Cooling of the atmosphere due to evaporation of precipitation is included. Sub-gridscale convective processes are modelled using a simple cloud model (Gregory et al, 1990); convection affects the large-scale atmosphere through compensating subsidence, detrainment, and the evaporation of falling precipitation.

The radiation calculation uses six bands in the long wave and four in the solar calculation and allows for water vapour, ozone, carbon dioxide, and the layer and convective cloud distributions. Cloud radiative properties depend on cloud water content.

The effects of the drag caused by sub-grid-scale gravity waves is estimated using the sub-grid variance of the orography and the known absorption properties of gravity waves in a given atmospheric profile.

2.2.3 Ocean model

The ocean model is based on code developed at the Geophysical Fluid Dynamics Laboratory by Bryan and Cox (Cox, 1984). The model uses a leap frog integration scheme with a second order accurate finite difference representation on a spherical grid. The formulation allows irregular domains to be used with realistic coastlines and bottom topography. The mesh is variable and is constructed so that the zonal grid spacing is a function of longitude and the meridional grid spacing a function of latitude. Physical parametrizations include a mixed layer model, a treatment of penetrative solar radiation, convective adjustment and isopycnal diffusion.

Configuration	Levels	Points E-W	Points N-S	Resolution
Global climate	20	96	73	270km
N. Atlantic (FOAM)	40	140	112	100km
Tropical Indian	16	96	92	30-150km
Tropical Pacific	16	134	94	30-150km

Table 2. Dimensions of the ocean models used on the Cray Y-MP for various applications

Several different ocean applications are being developed for the Cray Y-MP. These are shown in Table 2. The FOAM (Forecast Ocean Atmosphere) model covers the North Atlantic and is expected to become operational within the next 4 years using a gridlength of around 30 km. The tropical models are run as part of the TOGA (Tropical Ocean Global Atmosphere) programme, focussing on phenomena such as El Niño and the Indian Monsoon.

3. CODE DESIGN AND MULTITASKING STRATEGY

3.1 Background

On previous Met Office computer systems production code was generally written in a low level language in order to extract the maximum performance from the hardware. On the Cyber 205, for example, Q8 calls (allowing direct access to hardware instructions) were used extensively in climate and operational forecast applications. Although this approach led to fast efficient code, it made it difficult to maintain or modify. It was therefore decided to use Fortran when writing the unified model and leave the job of producing efficient, optimised code to the Cray compiling system. It is felt that the performance gains that might be obtained by judicious use of a low level language are now outweighed by the need to make code modifications as easy as possible. Apart from at the control level, all sections which perform meteorological calculations are designed to be 'plug compatible', and communicate information with the rest of the model purely through their argument list. All routines correspond to the standard for exchanging meteorological code proposed by Kalnay et al (1989).

The Cray Y-MP in use at the Met Office has eight CPUs each with a 6 nanosecond clock period, 32 Mwords of main memory and 128 Mwords of SSD memory. Under the UNICOS operating system the eight processors can operate independently on separate programs or concurrently on a single problem in multitasking mode. The dedicated resources of all 8 CPUs on the Y-MP will be used for production runs of the forecast models. Climate modelling will also require some of the speed gains offered by multitasking, although in practice a small number of separate climate integrations may be run in parallel.

In discussing the performance of the unified model in detail, it is convenient to consider the major components separately. This will be done in subsequent sections. In this overview, we note from Table 3, the

relative costs of the main components of an atmospheric model run in both data assimilation mode and forecast mode at the highest proposed resolution. We see that in assimilation mode, the assimilation component is most costly both in terms of work and time. In forecast mode, we see that the dynamics entails much more work than the physics but the costs are almost the same, indicating clearly that dynamics code is more efficient. The trend is towards further decreases in the relative cost of the dynamics as observing systems become more extensive and parametrization schemes become more complex.

Model component	% of total work (flopcount)	% of total cost (elapsed secs)
ASSIMILATION MODE		
Assimilation	44	49
Physics	17	23
Dynamics	39	28
FORECAST MODE		
Physics	30	45
Dynamics	70	55

Table 3 Contribution of various components to total cost of model

3.2 Using Autotasking

Three alternative ways of multitasking a piece of code are available on a Cray Y-MP; macrotasking, microtasking and Autotasking. Macrotasking allows a program to be partitioned into two or more tasks at the subroutine level. The programmer inserts library routine calls into the source code to initiate and synchronise tasks scheduled by the operating system. Microtasking, on the other hand, allows parallelism to be exploited at the DO-LOOP level and is therefore suitable for problems that exhibit a finer granularity. The programmer identifies parallel regions in his program and inserts directives accordingly. Code is then generated which allows the program to hand out loop iterations to each CPU as it becomes available for work using a master-slave relationship. If no extra CPUs are available, the code will continue to execute on a single CPU. Autotasking is functionally equivalent to microtasking with the added advantage of automatic detection of parallelism and automatic scoping of variables. It attempts to identify and exploit parallel regions in a program through the use of a

preprocessor.

Because of its ease of use, Autotasking was chosen to multitask the unified model. It is a particularly attractive option because no machine-dependent code needs to be inserted. If programmer intervention is desired to further optimise performance in routines where the Autotasking preprocessor fails to detect an opportunity for parallelism, it is done by means of directives.

Autotasking consists of a three phase compiling system: dependence analysis (fpp), translation (fmp) and code generation (cft77). The dependence analysis phase looks for parallelism in the program and inserts directives which are then interpreted by the translation phase. The translation phase converts the directives into extra code which controls the parallel execution. Directives to override how a particular parallel block is handled by the compiling system may be inserted by the programmer at either the fpp or fmp step. More details on the features of Autotasking may be found in Furtney (1990).

3.3 Data organisation

At the present time the atmosphere and ocean components use different data organisations. The atmosphere model stores data by fields whilst the ocean model stores data by latitude rows. Both have an outer loop over the number of levels. In its default mode, Autotasking allocates successive iterations of the outer loop to separate processors. This can be particularly effective if the number of processors is an exact divisor of the number of levels. Alternatively, the innermost vector loop may be stripmined by partitioning it into shorter strips and allocating each strip to a separate processor. This approach removes the need to consider any horizontal (or vertical) dependencies inherent in the integration scheme, but it is only successful as a general multitasking strategy if the inner loop is long. It is therefore only a viable option if the data is stored by fields.

Because the atmosphere model uses an explicit timestepping scheme, only one copy of the prognostic fields needs to be stored. This allows all the data to be retained in memory, even at the highest proposed resolution. The ease with which data can be manipulated in a single shared memory greatly simplifies the data assimilation problem as well as the generation of output fields, particularly when processing complicated configurations such

as the lateral boundary information for an embedded regional model.

4. PARALLEL PROCESSING THE DATA ASSIMILATION COMPONENT

Observations describing the state of the atmosphere are distributed inhomogeneously in space and time. A brief summary of the currently available observations is given in Table 4, together with an estimate of future trends.

Observing system	Type of report	Number/6hours	
		now	future
Radiosonde	T,V,q (multilevel)	1000	1000
Surface	P,T,V,q (1 level)	5000	5000
Aircraft	T,V (1 level)	1000	10000
Satellite cloud vector	V (1 level)	2000	2000
Satellite soundings	T (multilevel)	5000	100000
Satellite scatterometer	V (sea surface)	0	50000
Satellite Windlidar	V (multilevel)	0	50000

Table 4 Observation characteristics and data volumes
(T=temperature,P=pressure,V=wind and q=moisture)

We note from this table the diversity of the global observing system, with some components observing wind only, some temperature only and others giving a more complete measurement; some components giving a profile of measurements through a depth of the atmosphere and others restricting their observation to a single level; some components giving a rather limited global coverage whilst those satellite based systems giving a more complete coverage. All of these features of the observing system conspire to make the problem of multitasking the data assimilation component, to say the least, rather messy.

The data assimilation code is greatly simplified by considering each component of the observing system in turn and adapt the algorithms to suit. Further simplification is achieved by partitioning the problem into a vertical part and a horizontal part. The result from the vertical stage is a column of increments on model levels at the observation location. The vertical stage is not very costly. Vectorization is over the number of observations and Autotasking is used to partition these loops over the available processors.

The horizontal stage seeks to spread the increment (Y) from the observation (k) to nearby gridpoints (i), as indicated by equation (1).

$$\Delta x_i = \Sigma_k W_{ik} Y_k \qquad (1)$$

Each observation has a sphere of influence which can exceed 1000km. Thus at the highest resolutions this might involve as many as 500 gridpoints. The code for equation (1) gives nested DO-LOOPS over observations and influenced gridpoints. At the higher resolutions it appears not to matter which way the loops are nested, since both the observation counts given in Table 4 and the number of influenced gridpoints given above are sufficiently large for efficient vectorization on a Cray Y-MP. For lower resolutions the number of influenced gridpoints scales down substantially and an inner loop over observations seems more appropriate. However this strategy leads to a potential store hazard where near-coincident observations which are adjacent in the observation vector attempt to update the same location in the output increment vector on the model grid. For this reason, the alternative vectorization strategy with an outer loop over observations has been adopted.

Autotasking directives may be inserted by the programmer to achieve multitasking of the outer loop, forcing single iterations (observations) of the outer loop onto different processors as they become available. The complexity of the inner loop appears to rule out automatic multitasking without programmer intervention. A similar store hazard to the one discussed above needs safeguarding against when this code is multitasked, since the increment vector of model gridpoints is an array in shared memory and potentially there is a problem if individual processors simultaneously update this shared array. The solution to this problem is achieved by inserting further directives (GUARDS) around the critical area of code which performs the gathering from/addition to/and scattering back of the partial analysis increments array, thus preventing simultaneous access by the processors. Fortunately the amount of work in the GUARDed area is small relative to the total work in the loop so that with 8 processors the waiting time at the GUARD is minimal.

The single tasked speed of this horizontal part of the assimilation is roughly 110 megaflops. This compares unfavourably with the sort of peak

rates achieved by straightforward dynamics code because of the complexity of the coding, for instance, in the spherical geometry calculations required for (observation-gridpoint) distances. However, with the coding strategy adopted, the code exhibits a large granularity and performance improvements of roughly 7.8 are obtained with 8 CPUs.

Referring back to Table 4, we give details of future trends in data volumes during the coming decade. This increase in data volumes is potentially enormous and clearly more computer power will be required to cope with the explosion in satellite observations. One redeeming feature of the future observing systems is that the data are on a more uniform grid of a density not too dissimilar to those of the models, so we might envisage rather simpler algorithms for mapping observation increments onto the model grid.

Before leaving the data assimilation problem, it is worthwhile stressing the advantages of a large shared memory for this particular problem. We have had some experience of using local memory parallel processors (ETA-10) and large data transfers were necessary to obtain a good load balance with this algorithm. If model fields are held in local memory, then the observations appropriate to that same sub-domain must be held in the same memory. The uneven geographical distribution of observations leads to load balancing problems. In addition, as has already been mentioned, the horizontal influence of an observation is quite large and in the horizontal processing step this leads to increments being generated in one processor which update model fields held in the memory of other processors.

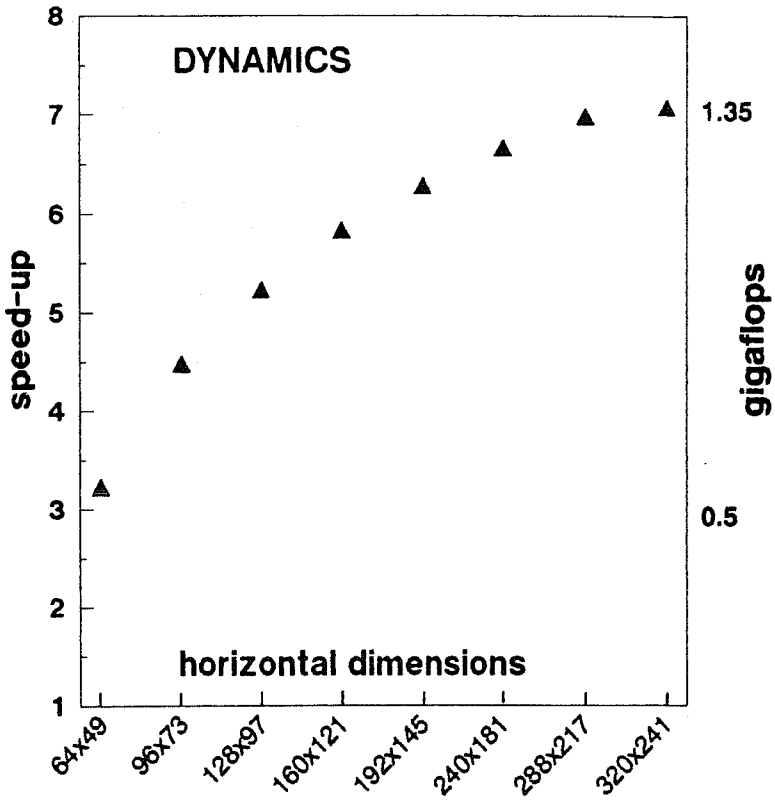
5. PARALLEL PROCESSING THE ATMOSPHERE MODEL

The forecast program has been designed to store data as horizontal fields, so that inner loops are typically over all points along a model level. This arrangement maximises vector efficiency while allowing a range of multitasking options. The multitasking strategy currently adopted uses Autotasking to partition the innermost loops across all the available processors. A feature of this approach is that provided the vectors are long enough, load balancing is assured by the dynamic scheduling algorithm used in Autotasking.

5.1 Dynamics

In the dynamics the same operations are generally carried out at every point in the integration domain. This makes these algorithms easy to

(a)



(b)

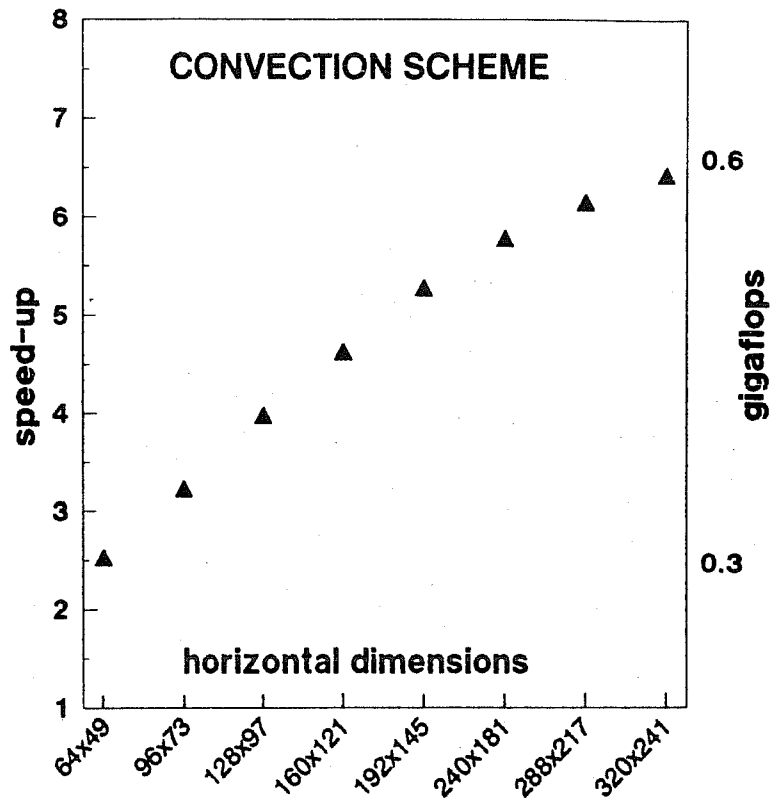


Figure 1. The gigaflop rate and speed-up over single tasked code recorded for 8-CPU Autotasked versions of (a) the global model dynamics and (b) the convection scheme using a range of horizontal dimensions.

vectorize, but there are horizontal and vertical dependencies resulting from the discretization process which limit the range of effective macrotasking options (Dickinson, 1990). By adopting a multitasking approach which stripmines inner loops, these dependences can be ignored. But the success of this strategy depends on having large do loops containing long vectors, since each loop incurs an overhead related to the cost of the single task code inserted by the preprocessor to control the multiprocessing. The performance of the dynamics routines, including the FFT code used for stability filtering, is shown in Figure 1(a). At the forecast resolution a 6.9 times speed up over the non-Autotasked, single CPU, code is recorded with a respectable performance of 1.3 gigaflops, while the climate resolution shows a speedup factor of only 4.5 and a performance level of 0.8 gigaflops.

An estimate of the multitasking overhead may be obtained by computing the difference between the cost of an Autotasked run limited to 1 CPU and the cost of the same run when Autotasking is not used. If this is inserted into Amdahl's Law, as shown below, it gives an indication of the maximum speed up that can be obtained.

$$\text{mt\_overhead} = \text{1 CPU Autotasked time} - \text{single\_tasked\_time} \qquad (2)$$

$$\text{max\_speed\_up} = \frac{\text{single\_tasked\_time}}{\text{mt\_overhead} + \text{single\_tasked\_time/number\_of\_cpus}} \qquad (3)$$

Timings of the dynamics code show that the multitasking overhead is 10% at climate resolution and 2% at forecast resolution. On substituting this into Amdahl's Law (3) with number_of_cpus=8, maximum speed-ups of 4.5 and 6.9 respectively are obtained: the same as given in Figure 1(a). Since the multitasking overhead is proportional to the number of parallel loops in the code, any further improvement in performance can only be obtained, logic permitting, by combining together loops which use the same vector length. Any lack of load balancing appears to be insignificant, underlining the effectiveness of the dynamic scheduling algorithm used within Autotasking.

5.2 Physics

In contrast to the dynamics, the physics are data dependent and localised,

being characterised by nested conditional blocks of code. There are no horizontal dependencies and so a straightforward macrotasking approach where the integration domain is split into 8 regions, each integrated in parallel by a separate processor, is a valid alternative strategy to Autotasking inner loops. If the conditional constructs are converted to use the Cray vector merge functions such as CVMGT, then gigaflop rates in excess of those obtained for the dynamics can be produced. This is a false measure of performance, however, since most of the physical processes apply only at sub-areas of the grid and a mask-merge approach leads to calculations being carried out at all gridpoints irrespective of where the final results need to be applied.

In the gravity wave drag parametrization, for example, vertically propagating mountain waves are modelled and so the effects need only be computed over land points. Since just one third of the Earth's surface is covered by land, the cost of this routine can be significantly reduced by first focussing in on this set of points. This is achieved by using explicit gathers in the code based on index lists produced from simple tests. The other physics routines use this approach in varying degrees. For example, the cost of the convection scheme is shown in Figure 1(b). Note that the gigaflop rate is less than half that of the dynamics, but the wall clock time of the code is significantly less than can be obtained with the mask-merge approach. Tests show that further improvements in wall clock time, particularly at climate resolution, can be obtained by using the regional macrotasking approach described above, but with the addition of Autotasking of inner loops to provide load balancing.

Even if a macrotasking layer were added to the physics, the performance of the climate resolution version of the model would still be rather poor and it would be unacceptable to run this in a stand alone mode. However, at least four parallel streams of work are identifiable in climate prediction: a control run, various increased CO_2 senarios, formulation changes and runs from different initial data. Also, the size of the problem is such that four copies may easily fit into the available memory without the need to roll out a job.

A test of the throughput of the 8-CPU Autotasked code is shown in Table 5. It should be noted that these test jobs contain no I/O and no significant pieces of single tasked code. It can be seen that only a small improvement

is obtained as the number of jobs in the system is increased. This is because Autotasking grabs all available CPUs (as controlled by GETCPUS and the environment variable NCPUS), even if it cannot use them efficiently, and appears unable to share resources with another Autotasked job which also wants to use all the available CPUs. An alternative strategy is to limit the number of CPUs allocated to each job. As shown in Table 6, an acceptable level of throughput may be obtained with 4 parallel jobs each Autotasked over 2 CPUs.

No of copies of job in system	Avg time per job in secs
1	28.3
2	27.0
3	26.1
4	26.4

Table 5 Throughput test of 100 timestep run of climate resolution atmosphere model using 8-CPU Autotasked code.

	Wall clock time in seconds	No of CPUs actually utilised	No of CPUs allocated to work
Single tasked	114	4	4
Autotasked on 2 CPUs	64	7.1	8
Autotasked on 8 CPUs	105	4.4	8

Table 6. Throughput test of four 100-timestep runs of climate resolution atmosphere model.

6. PARALLEL PROCESSING THE OCEAN COMPONENT

The ocean model is organised by latitude rows, giving the option of an out of memory solution. For ease of vectorization values are computed over both land and sea points. At each timestep boundary conditions are applied along the coasts. Evidence from running the code on a Cyber 205 suggests that there is little advantage in first gathering together sea points in the manner described in §5.2 for the atmospheric physics, since savings in processing time are balanced by the extra costs of the gathers and scatters.

At the time of writing, conversion of the ocean model for the Cray Y-MP has only just begun. The initial multitasking strategy will be to apply Autotasking to the loops over levels. From Table 2, it can be seen that the number of levels divides by 4 in the case of the climate model and 8 for the tropical ocean and FOAM models. In practise, as in the case of the atmosphere climate model, the climate and tropical versions of the ocean model will be restricted to use just 2 CPUs.

Some indication of the performance of the ocean code may be obtained from a multitasking exercise undertaken by Cray UK on the FRAM Antarctic modelling code run by the Institude of Oceanographic Sciences. The model uses 32 levels on a 722x221 horizontal grid and is similar in formulation and design to the Met Office model. Applying Autotasking on an 8 CPU Y-MP, the code recorded 870 Mflops which was 6.1 times faster than single tasked code. When limited to just 4 CPUs, 550 Mflops and a 3.6 times speed up were observed. All of the code was Autotasked over levels, except for the solution of a Laplacian resulting from the upper boundary condition, which was stripmined across the 722 points E-W. This was the most expensive routine, accounting for 30% of the execution time. From Table 2 it can be seen that lower horizontal resolutions are planned by the Met Office (Table 2) leading to potentially less parallelism. However, a computationally more efficient formulation of the upper boundary condition is planned, which will allow this part of the code to be Autotasked over levels.

7. CONCLUSIONS

The Met Office's atmosphere and ocean modelling codes are being rewritten for the Cray Y-MP using standard Fortran. Care has been taken to apply a consistent coding style and common interface to all components of the system, unifying the code for climate and forecast applications. It is expected that this code will form a platform for our modelling activities over the next 10 years and more.

Optimization is handled by the Autotasking preprocessor. This allows the speed benefits of multitasking to be applied to all modelling activities, particularly high resolution numerical weather prediction, in a user friendly, almost transparent, way.

The atmosphere component of the unified model stores its data as horizontal

fields. By Autotasking inner loops, horizontal and vertical dependencies are avoided which usually need to be considered when using the more traditional macrotasking approach to parallelise grid point models. Acceptable levels of parallelism can be achieved with large problem sizes, although the combined use of macrotasking and Autotasking (to give better load balancing) may well give the best results, particularly for the physics routines. Further work still needs to be done to determine the optimal combination which minimises wall clock time.

At climate resolution, the ocean and atmosphere models are less parallel and so it is not appropriate to run these integrations in stand alone mode. Batch throughput tests of the atmosphere code at climate resolution suggest that running more than one 8-CPU Autotasked job in the system at the same time does not lead to a significant improvement in overall performance. Essentially, batch processing will only make an inefficient 8-CPU Autotasked job relatively more efficient if other jobs in the system are restricted to use less than 8 CPUs. Acceptable levels of performance appear to be possible if climate runs are limited to use 2 CPUs each.

Retention of model data in a single shared memory has greatly simplified those parts of the model which are non-local in character with significant horizontal or vertical dependencies. In particular this strategy is well suited to the data assimilation problem and also the generation of certain output fields such as lateral boundary information for other models.

8. ACKNOWLEGMENTS

The authors wish to thank Deborah Salmond and Mike O'Neill of Cray Research UK for providing the ocean modelling timings discussed in §6.

9. REFERENCES

Cox, M.D., 1984: A Three Dimensional Model of the Global Ocean. GFDL Ocean Group Tech. Note 1. GFDL, Princeton, U.S.A.

Cullen, M.J.P., Davies,T. and Mawson M.H.,1990: Conservative Finite Difference Schemes for a Unified Forecast Climate Model. Unified Model Documentation Paper No 10, Short Range Forecasting Research Division, Met Office, Bracknell.

Dickinson, A., 1990: Multitasking the Meteorological Office Forecast Model

on an ETA-10. The Dawn of Massively Parallel Processing in Meteorology, Eds G-R Hoffman and D. K. Maretis, Springer-Verlag.

Furtney, M., 1990: Parallel Processing at Cray Research Inc. The Dawn of Massively Parallel Processing in Meteorology, Eds G-R Hoffman and D. K. Maretis, Springer-Verlag.

Gregory, D., and Rowntree, P.R., 1990: A Mass Flux Convection Scheme with Representation of Cloud Ensemble Characteristics and Stability Dependent Closure, Mon. Weather Review, June 1990.

Lorenc, A.C., Bell, R.S. and Macpherson, B., 1991: The Meteorological Office Analysis Correction Data Assimilation Scheme. Quart. J. R. Meteorol. Soc

Kalnay, E., Kanamitsu, M., Pfaendtner, J., Sela, J., Suarez, M., Stackpole, J., Tuccillio, J., Umscheid, L. and Williamson D., 1989: Rules for Interchange of Physical Parametrizations. Bulletin of American Met Soc, Vol 70, No 6.