Parallelisation of the Unified Model Data Assimilation Scheme

Kenneth A. Hawick, [*] R. Stuart Bell, Alan Dickinson, [†]
Patrick D. Surry and Brian J. N. Wylie[*]

October 1992

Abstract

We describe the data assimilation scheme employed in the UK Meteorological Office's Unified Model and a number of schemes for its efficient implementation on massively parallel computer systems. A simple port to the Connection Machine CM-200 system was carried out and for which detailed timing and profiling figures are given. A number of alternative implementation algorithms were investigated and are also described.

An algorithmic inversion of the vectorised implementation allowed the processors to work on the parallel update of the influences of observation data on all the model grid points together rather than working sequentially through a vector of model grid points. A further improvement to the implementation required the packing of multiple observations into vectors of appropriate length to the number of processors. Finally, the model grid was statically divided into latitudinal sections to allow a better balance of processor work load.

Implementations of these schemes were carried out using High Performance Fortran style directives and Fortran 90 array constructs on the Connection Machine CM-200 system.

Profiling measurements were made for typical runs at both weather prediction and climate resolutions. A number of potentially better implementation schemes are suggested for other parallel computer systems. We report on implementations in progress for other parallel processing programming paradigms.

[*]Numerical Group, Edinburgh Parallel Computing Centre, James Clerk Maxwell Building, Mayfield Road, Edinburgh EH9 3JZ

[†]UK Meteorological Office, London Road, Bracknell

1 The Unified Model

This paper describes our work on implementing the data assimilation component of the UK Meteorological Office's (UKMO) Unified Model on a Connection Machine CM-200 massively parallel processor. This investigation was carried out to explore some of the issues and difficulties of parallel processing for meteorology in the context of a real operational weather and climate prediction code.

The Unified Model (UM) is a Fortran code employed by the UKMO for numerical weather prediction and for climate prediction. The UM allows various resolutions and options so that one suite of code can carry out all the required functions. Broadly speaking, a weather prediction code consists of three main components: the atmospheric dynamics; the physics calculations; and the assimilation of observational data. The dynamics of the UM is based around a regular rectangular grid based model and suggests a number of natural strategies for data decomposition onto a regular grid of processors in a parallel computing system. A number of authors have considered the basic parallel operations necessary for implementing the dynamics of weather models [4, 10, 5, 6, 7]. The physics calculations are believed to involve localised computations being characterised by nested conditional blocks of code [1].

The assimilation of observational data however is by no means trivial and it is difficult to identify a natural way to implement this on the massively parallel systems available now and in the foreseeable future. The data assimilation scheme is vital to the success of a numerical weather prediction (NWP) code and at present a one day assimilation costs twice as much in elapsed execution time as a single day forecast. This ratio is likely to increase in future given the expected trend in observation volumes [1].

It emerged from a feasibility study of the whole UM code [10] that, in consequence, the data assimilation scheme was a suitable target for serious investigation and possible implementations on parallel systems. If this component of NWP codes cannot be efficiently implemented on parallel systems, it will severely limit their utility over traditional vector systems for meteorological applications.

As indicated above, the dynamics component of the UM suggests that a regular data decomposition strategy will be most natural for a parallel implementation on a distributed memory parallel system [10]. EPCC has a number of distributed memory computer systems but a strong factor in choosing the system for preliminary studies was the availability of a data-parallel Fortran in the style of the proposed High Performance Fortran (HPF) draft standard [11]. The Connection Machine CM-200 system at EPCC was therefore a natural choice since the CM-Fortran language available on that machine already has many of the proposed HPF features and a similar programming style to the proposed HPF. Our work has therefore focussed around exploring the means of migrating the data assimilation component of the UM onto an HPF style computing environment using the

CM-200 as a development platform.

2 Data Assimilation Scheme

The data assimilation scheme employed in the UM model is fully described in [2, 3] and is known as the Analysis Correction (AC) scheme. As formulated at present this is an iterative analysis scheme with each iteration interleaved by a dynamics and a physics step. The assimilation scheme is being continually developed and different algorithms may be employed in future which make a more optimal use of observational data and which may be better suited to future operational observation systems. Nevertheless it is a useful exercise to investigate how the current scheme can be implemented on massively parallel computer systems as the results from this will feed directly into the planning for future systems.

This present work will not dwell on the numerical details of the scheme but will concentrate on the computational aspects for an efficient implementation. Furthermore, we restrict attention here to the analysis correction necessary for a single level of the atmosphere and focus on the horizontal analysis of surface pressure data. It is expected that a parallel implementation of the UM would only involve decomposing the model fields horizontally [10, 13] so that the serial loops over atmospheric levels would increase the computational load on individual processors in our implementations but would not affect the basic decompositional strategies described. For the purposes of obtaining quantitative performance figures we also restrict ourselves to the model field for pressure only. This makes our implementation experiments simpler at the cost of further reducing the computational load on individual processors.

The AC scheme essentially involves the interpolation of observed data onto the regular grids of the UM. The observed data will typically originate from: radiosondes; surface measurements; aircraft reports; satellite cloud vector data; satellite sounding data and satellite scatterometer data. The model fields of temperature, pressure, wind and moisture are influenced by a variety of these sources. The observational data are not at particular points on the model grids but are rather scattered across the surface of the globe. Furthermore, the area of influence of a particular observation is not trivial. The constraints of the AC scheme require that these areas of influence or "footprints" vary considerably in size and distribution. A vector of observations can therefore be assembled at each time step of the UM, but these vectors vary in length considerably and are not presented to the UM in a simple sorted order.

Figure 1 illustrates the changes in the observation vector length and the maximum, average and minimum of the number of model grid points influenced for the typical operational data set we employed in this work. Our work focussed on a typical operational forecasting resolution which consists of a model grid of 288 by 217 points. Note that the anomalously low observation lengths at early time steps are

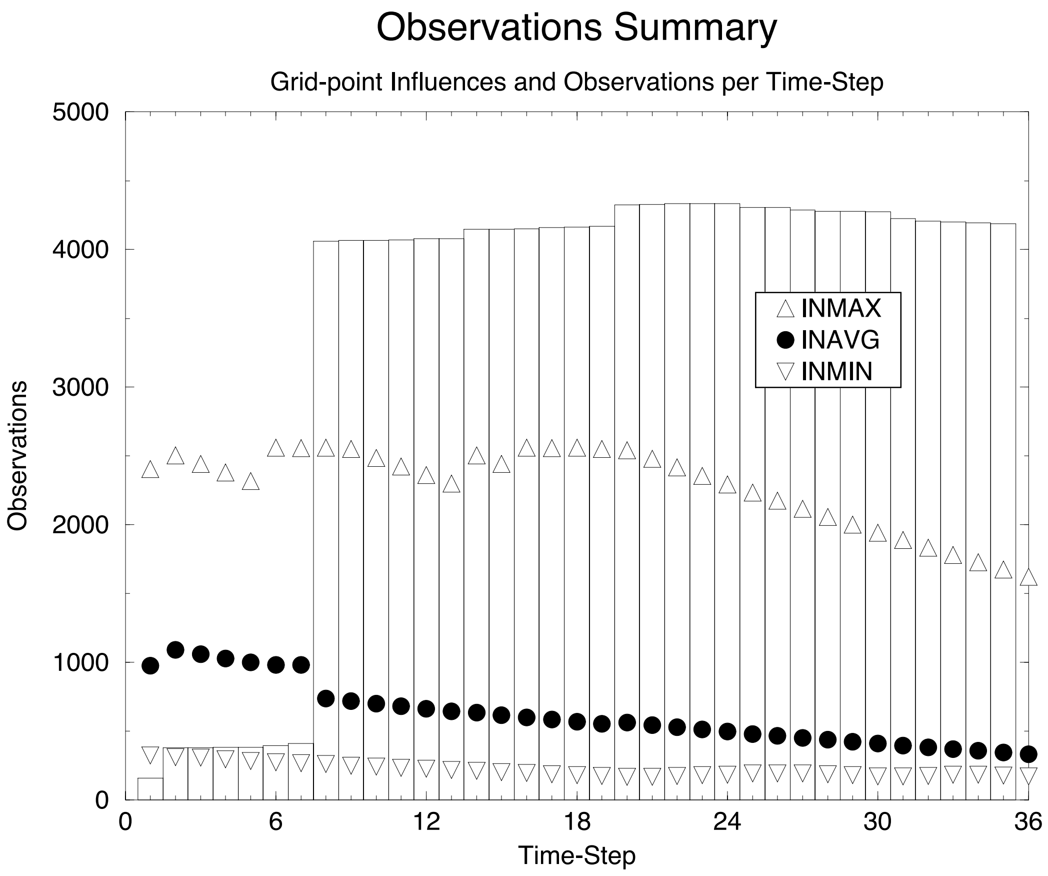

Figure 1: The histogram shows `LENOB`, the length of the vector of observations, at each time-step. Each observation influences a number of points on the analysis grid, and a summary of this is shown by `INAVG`, the average number of influenced points, between the limits `INMAX` and `INMIN`.

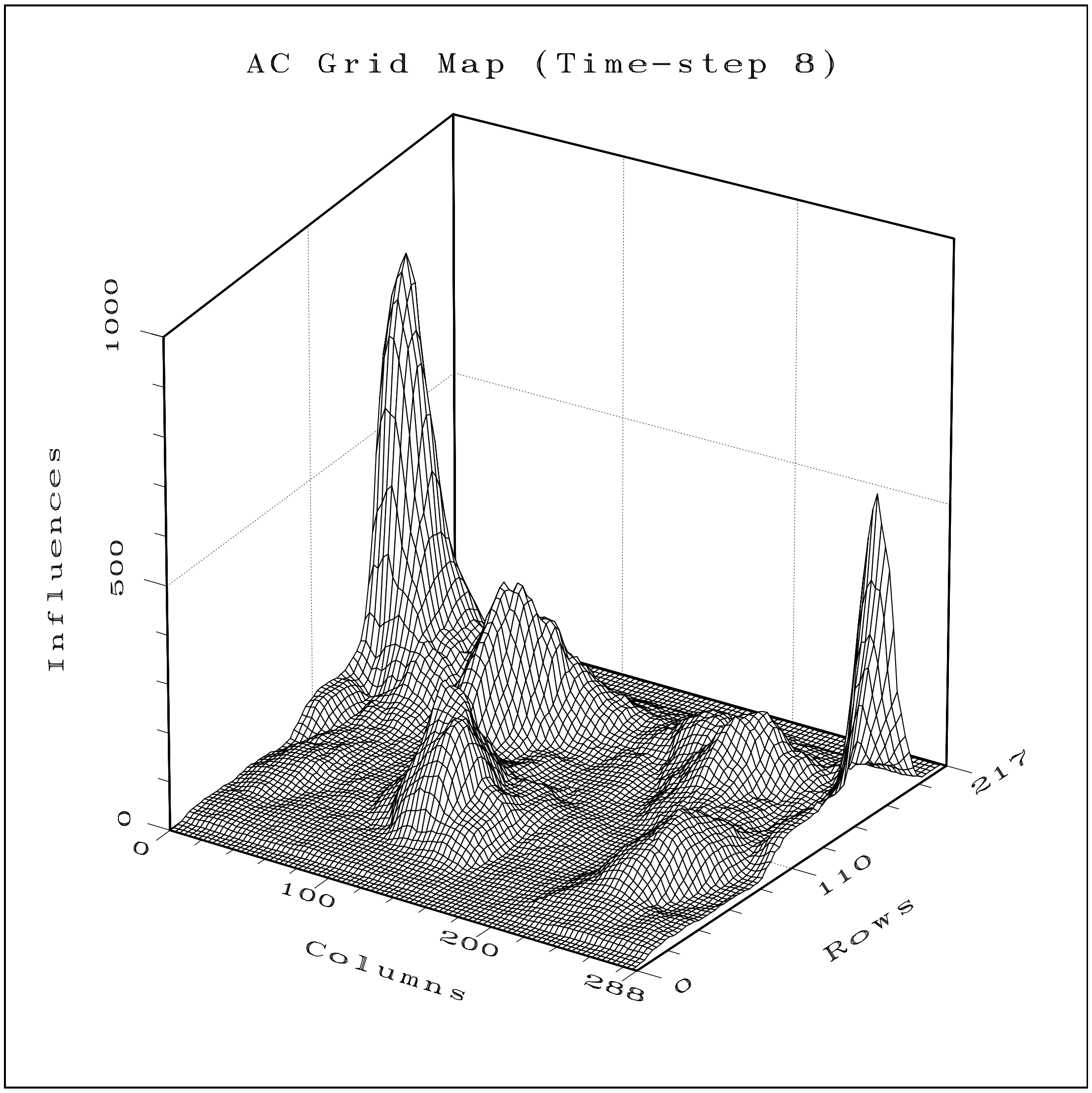

Figure 2: Inhomogeneity of observation influences per grid point in space.

Sec #	Serial SS1 s	Parallel CM-'q' s	Vector Y-MP/1 s	Gflop total	Perf. Rel. to Y-MP/1	Section description
1	19	1.3	—	—	—	Initialisation
2	2047	1302	23.2	0.97	0.018	Create index list
3	3704	54.6	36.3	6.94	0.665	Horizontal geometry calc.
4	21	1.2	—	—	—	Loop over slabs
5	4311	43.6	32.2	6.09	0.571	Slab horizontal influence
6	544	6.3	–^–	–^–	–^–	+ Setup rotation matrix
7	515	6.5	–^–	–^–	–^–	+ Loop over levels
8	19	1.2	—	—	—	Accumulation
9	2375	774	2.6	0.13	0.003	Indexing fields
X	—	28.6	—	—	—	Copy data to/from CM

Table 1: For each of the code sections identified in the HORINF2 routine (including an additional 'parallelisation overhead' section for the CM), cumulative execution times (for 72 calls) are compared for a range of machines: a (loaded) serial workstation (Sun Sparcstation 1), the Version 1 straightforward data-parallel port (with minimal optimisation) to a 'quart'-sized Connection Machine (CM-200) and a vector optimised Cray Y-MP (single CPU).

a feature of the cut down data set we are using. These low values are not relevant to operational predictions.

Figure 2 illustrates the wide variation in the number of observational influences per model grid point. The continents can be readily discerned from this data and the highest peak is surely around Reading and Bracknell.

2.1 Serial and Existing Vector Implementations

It is difficult to obtain a realistic assessment of the performance of a parallel system without implementing the full operational code and measuring wall clock time. To understand more fully the different components of our partial implementation of the AC scheme, we have profiled the AC code and broken it down into a number of real computational components and book-keeping parts. The execution time on a number of systems is shown in table 1. Note that the Connection Machine CM-200 system can be operated in a number of configurations with different numbers of processors. These are colloquially referred to as 'pint' (8192 processors), 'quart' (16384), 'half-gallon' (32768) and full gallon (65536). These are denoted by 'p' and 'q' in the performance tables which follow.

Figure 3 shows the breakdown of execution time for each of the sections identified in table 1. These figures are for a serial implementation on a SUN workstation. They show the wide variation between the computational and book-keeping components of the AC scheme, and help explain some of the performance figures presented below.

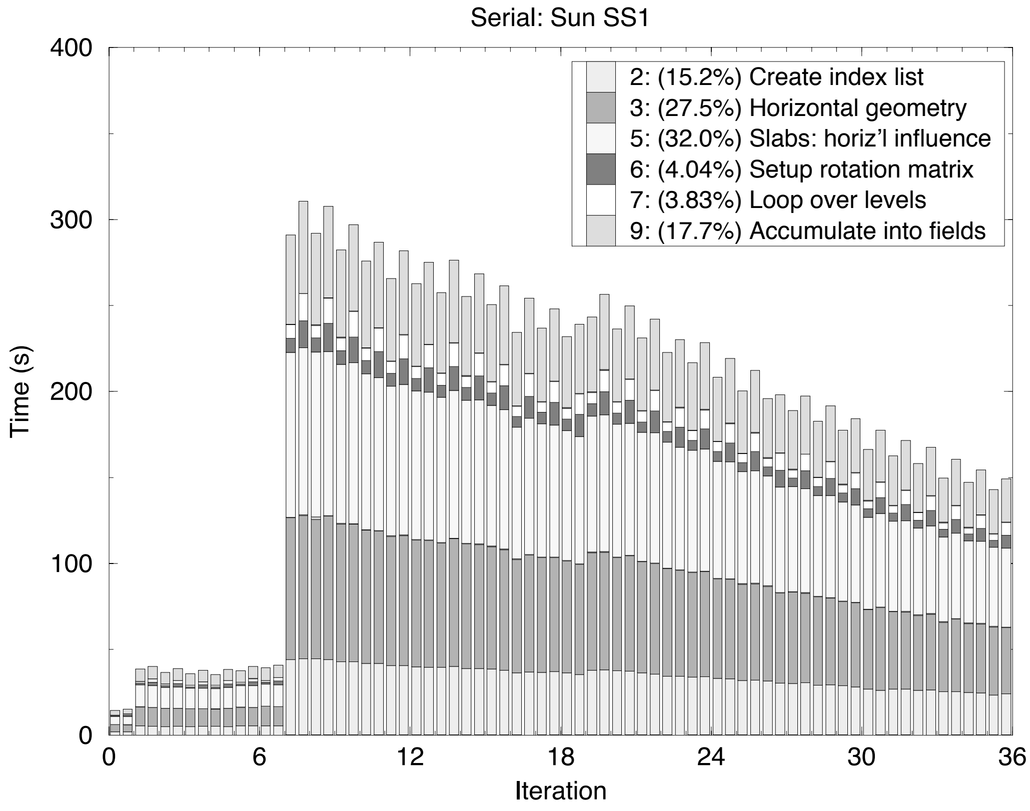

Figure 3: Breakdown of HORINF2 performance by iteration and section for the serial code.

3 Parallel Implementations

As presented above, the AC scheme essentially involves the interpolation of randomly distributed observational data influences onto a regular grid. We have identified a number of possible algorithms for implementing this scheme on a data-parallel computing system. These are illustrated in figure 4 and are described in detail in the following sections.

3.1 Simple Indirection

This approach is that currently adopted for the CRAY Y-MP implementation. A single observation vector is considered at a time and gather-scatter operations are employed to interpolate its influences onto the model grid. There is some inefficiency here since no attempt is made to make the observation vector length

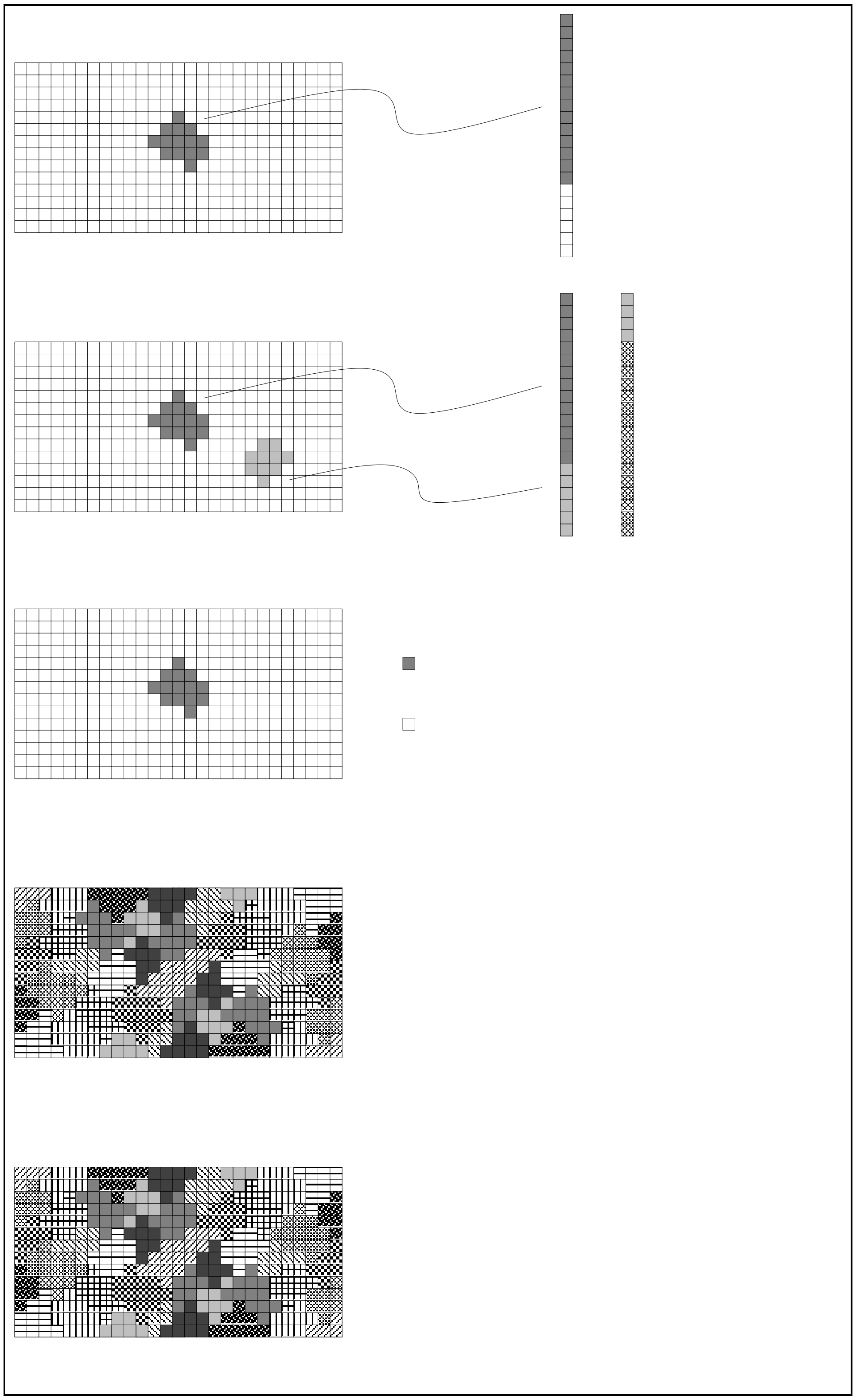

Figure 4: Strategies for the parallelisation of the UM Analysis Correction Scheme.

conform with the natural vector length (or number of processors) of the computer system.

3.2 Packed Indirection

This scheme allows some increased efficiency — in principle. A number of observation vectors are packed together to make up vectors close to or slightly less than full machine length vectors. Unfortunately the extra book-keeping involved in this scheme results in no appreciable improved performance for our test data set.

3.3 Full Grid (Data Parallel)

This method involves an inversion of the loops of code in the AC scheme. Whereas before, the loops were arranged so that the effect of each observation vector in turn was computed, and so the observations were decomposed across processors, here we consider the model grid to be decomposed across processors. This scheme is extremely inefficient, since a typical observation influence is small compared to the whole grid. Consequently there are a number of idle processors — the ones whose grid points are not influenced by the influence vector under consideration.

3.4 Packed Full Grid (PFG)

This scheme makes more effective use of the processors by packing a number of partial observations into a buffer and computing their effect simultaneously. The buffer arrangement is illustrated in figure 5. In order to ensure a reasonably well packed buffer of observations it was found convenient to order the observations randomly. This avoids any spatial ordering which may exist in the list of observations presented to the AC scheme as input data. It is important that as little overlap occur as possible to ensure the best packing. However it is desirable to minimise any book-keeping overhead in enabling the optimal packing. A pseudo-random order was the computationally cheapest way of doing this.

3.5 Efficient Full Grid (Scattered Spatial Decomposition - SSD)

This scheme is the most efficient we have been able to devise for the AC scheme on distributed memory hardware. The observation packing efficiency we can obtain for the packed full grid method is not optimal even with pseudo-random ordering. If the observation distribution is known at the start of an operational forecast run it will be possible to carry out an appropriate scattered spatial decomposition of the model grid points onto the processors so as to balance the typical load as evenly as possible. This is illustrated in figure 6. A simple algorithm for making

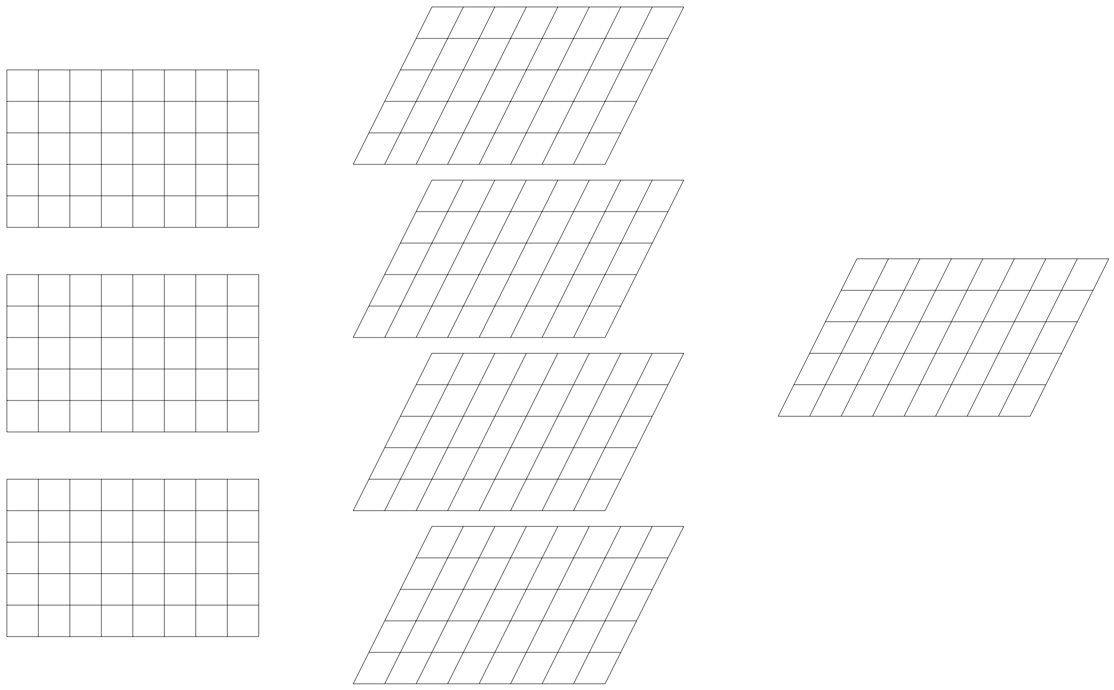

Figure 5: The technique for forming the histogram of observation footprints is illustrated after three jobs have been added to the buffer.

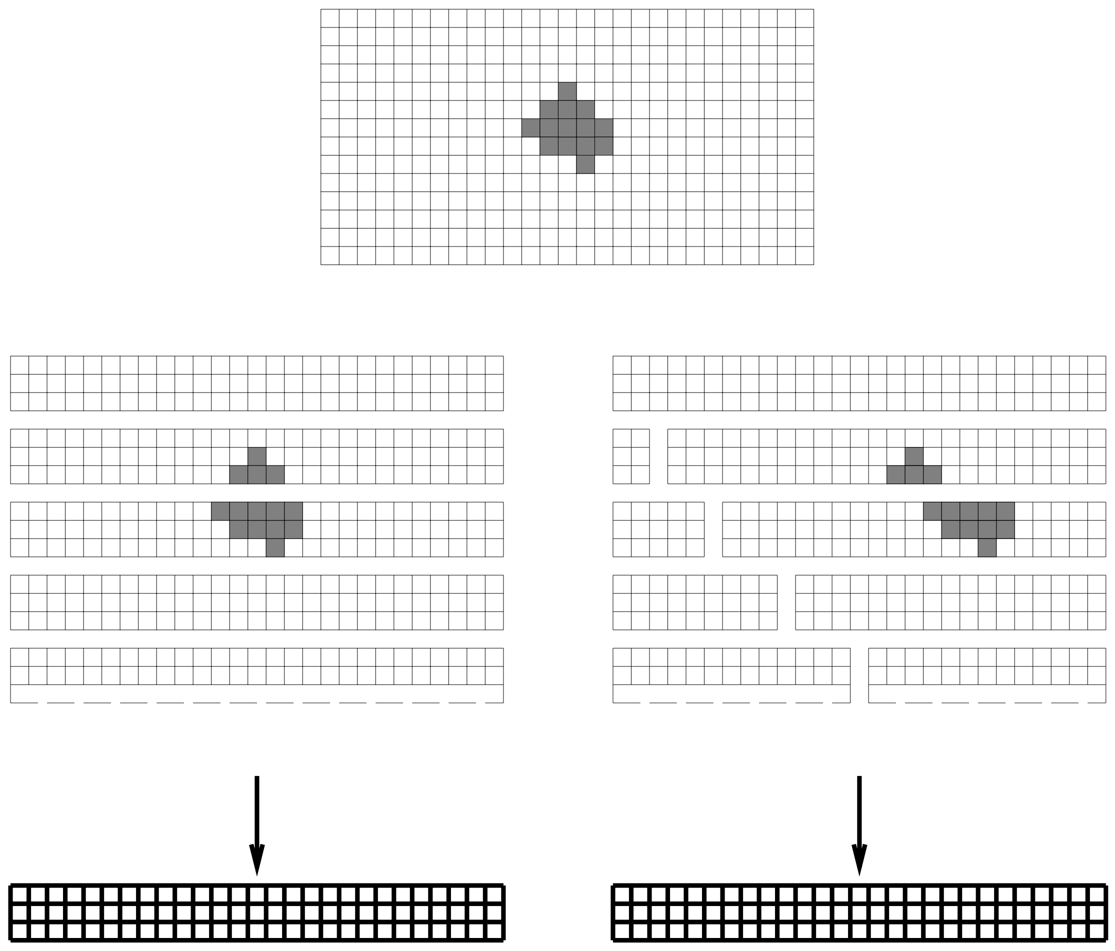

Figure 6: The grid is explicitly divided into sections which are offset from each other to simulate a scattered decomposition of the grid onto the processors

the decomposition was to split the model grid into sections by latitude and shifted by a regular number of grid points. This turns out to be a better algorithm for minimising the footprint overlaps in the buffer and gave the best performance. The effect is the creation of a fairly regular scattered spatial decomposition of the observational influences, which matches the communications requirements of the algorithm with that available in the architecture.

4 Performance Results and Conclusions

A selection of performance results are given in table 2. This shows the breakdown of the individual components of the AC scheme for implementations of: the simple indirection; the full grid; the packed full grid and the scattered spatial decomposition algorithms. The percentage time breakdowns are shown for each implementation as well as absolute timings.

Table 3 shows the results of running the various algorithms on a larger Connection Machine at Cambridge USA [1].

This shows that in general the scattered spatial decomposition (SSD) scheme achieves the best absolute performance, but also indicates some interesting anomalies for different numbers of processors. We expect the scaleability from the climate resolution of 96 by 73 grid points to be poorer as there is less computational work to do and consequently the compute to communications ratio drags the performance down.

The results indicate that the SSD algorithm continues to give the best absolute performance while the packed full grid algorithm appears to scale better for the particular observational data set we used.

In absolute terms our performance figures compare reasonably well with the CRAY Y-MP system. The SSD algorithm on a quart sized CM-200 is faster that a single procesor CRAY Y-MP in the compute intensive part of the calculation where efficient parallelism has been achieved. However in the present experiment the additional book-keeping required is still expensive although it is anticipated that a more realistic three dimensional configuration will yield a more favourable comparison. The performance of this algorithm will improve in efficiency with increased number of observation data. NWP operational requirements suggests that this is likely to be necessary as more satellite data becomes available. Increased grid point resolution is also likely to improve the efficiency of our algorithm. In conclusion, it appears there may be scope for a massively parallel processor to be used for operational forecasting in a scalable way.

Alternative approachs to parallel implementations of the AC scheme are possible.

[1]It is a pleasure to thank Bruce Boghosian and TMC for their co-operation in making this resource available to us

EPCC CM-200 ‘pint’								
Section	SSD		PFG		Full Grid		Indirection	
#	Time	%	Time	%	Time	%	Time	%
1	6.555	53.7	19.784	55.7	15.208	29.9	0.015	0.0
2	2.420	19.8	1.369	3.9	—	—	20.686	59.6
3	0.760	6.2	5.329	15.0	16.271	32.0	0.903	2.6
4	0.005	0.0	0.003	0.0	0.014	0.0	0.014	0.0
5	0.575	4.7	3.114	8.8	13.484	26.5	0.724	2.1
6	0.054	0.4	0.725	2.0	0.811	1.6	0.086	0.2
7	0.055	0.5	1.002	2.8	0.868	1.7	0.088	0.3
8	0.005	0.0	0.003	0.0	0.014	0.0	0.014	0.0
9	0.077	0.6	0.352	1.0	1.752	3.4	11.757	33.9
X	1.699	13.9	2.616	7.4	2.382	4.7	0.431	1.2
Total	12.204		*34.296 (35.5)		*50.803 (52.8)		*34.717 (40.1)	

EPCC CM-200 ‘quart’								
Section	SSD		PFG		Full Grid		Indirection	
#	Time	%	Time	%	Time	%	Time	%
1	4.555	51.4	12.555	56.1	8.917	29.4	0.018	0.1
2	1.849	20.9	0.901	4.0	—	—	18.089	58.7
3	0.432	4.9	3.238	14.5	9.534	31.6	0.758	2.5
4	0.005	0.1	0.003	0.0	0.014	0.1	0.016	0.1
5	0.331	3.7	1.955	8.7	7.904	26.2	0.605	2.0
6	0.036	0.4	0.487	2.2	0.508	1.7	0.088	0.3
7	0.036	0.4	0.682	3.0	0.525	1.7	0.090	0.3
8	0.005	0.1	0.003	0.0	0.014	0.1	0.017	0.1
9	0.048	0.5	0.202	0.9	1.010	3.3	10.744	34.9
X	1.557	17.6	2.367	10.6	1.789	5.9	0.397	1.3
Total	8.855		22.393		30.216		30.823	

Table 2: The performance of each HORINF2 implementation on the EPCC CM-200 in ‘pint’ and ‘quart’ sizes is compared. The times given are cpseconds per call. While the ‘quart’ performance is identical for Full Grid and Indirection, the scaleability is superior for Full Grid. PFG retains the scaleability of Full Grid within the ‘core computation’ (#2–9), yet is 3 times faster within this region, at the expense of a slightly more complex initialisation section (#1). SSD is much faster in all sections, with the exception of section #2 which has additional table calculations. The delivered factor of 3 improvement in total execution speed includes far more significant speedup in certain calculation sections, and still incorporates a scaleability of 1.38. [The ‘pint’ entries marked with asterisks ‘*’ are from partial runs, and estimated full run execution times are included in brackets.]

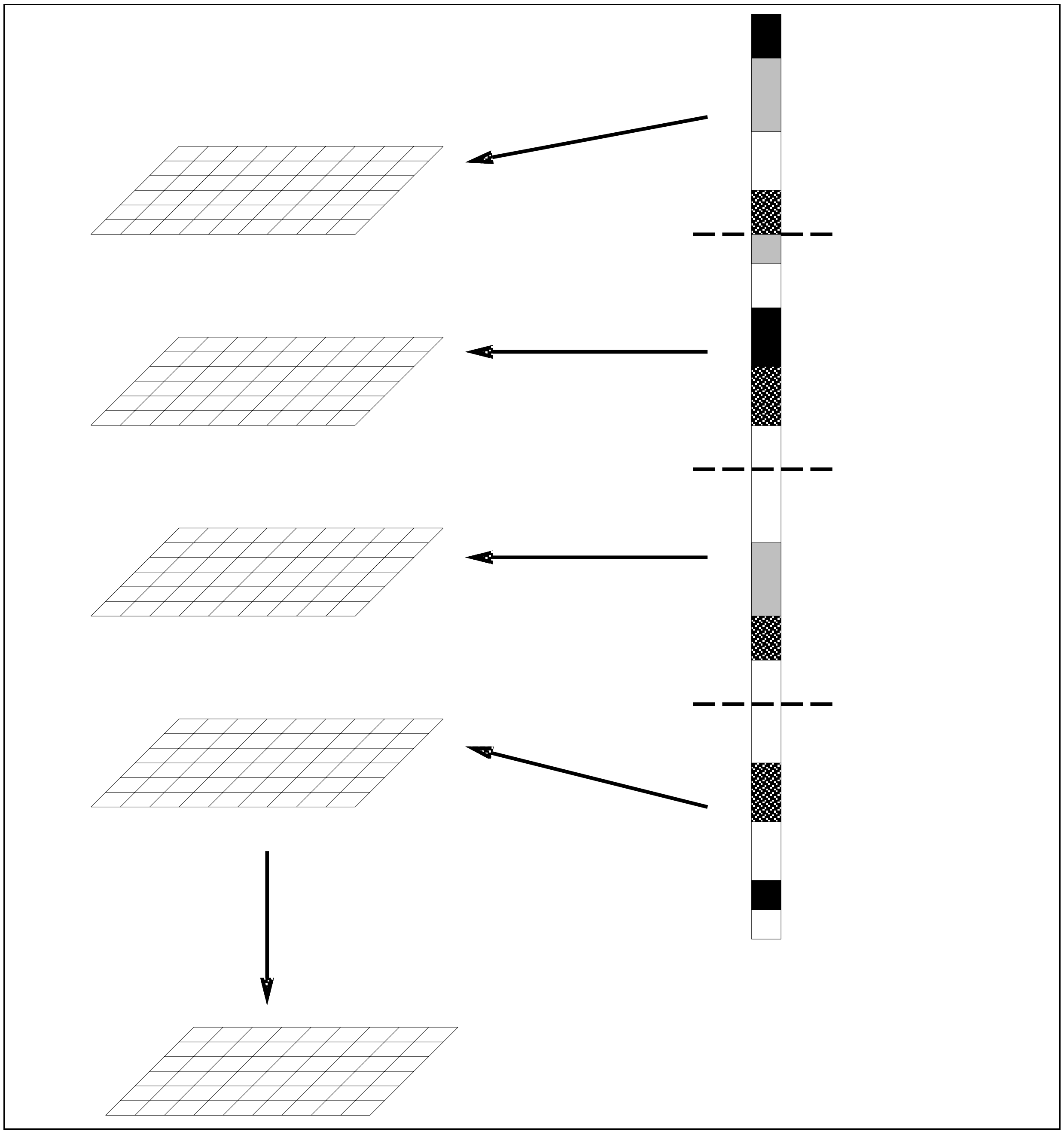

Figure 7: Possible task farming strategy for scaling to many processors.

CM Host		Forecast 'hi' resolution				Climate 'lo' resolution			
Site	Size	Run	Busy	CM%	Scal.	Run	Busy	CM%	Scal.
Version 2									
CMNS	'pint'	5421	5126	94.6	—	1056	783	74.1	—
CMNS	'quart'	3430	2923	85.2	1.75	769	474	61.6	1.65
CMNS	'half'	1837	1579	86.0	3.25	507	320	63.1	2.45
Version 3									
CMNS	'pint'	4835	4228	87.4	—	1536	846	55.1	—
CMNS	'quart'	2767	2497	90.2	1.69	1282	590	46.0	1.43
CMNS	'half'	1873	1463	78.1	2.89	973	463	47.6	1.83
Version 4									
CMNS	'pint'	1607	1121	69.8	—	1339	451	33.7	—
CMNS	'quart'	1026	793	77.3	1.41	993	382	38.5	1.18
CMNS	'half'	1228	652	53.1	1.72	1088	428	39.3	1.05

Table 3: Run execution times and CM 'busy' times (in seconds) are shown for the two simulation resolutions and three versions of the code on different sizes of Connection Machine. Percentage CM utilisation and scaleability figures are also calculated.

We are currently investigating a task-farming strategy that may be more appropriate to a more loosely-coupled distributed memory system. This algorithm is illustrated in figure 7. Providing there are sufficiently many observations, they can be partitioned across a number of autonomous processors, each of which has its own copy of the model grid. Each processor then computes the influences of its 'bag' of observations and interpolates them onto its copy of the grid. The grid copies are then combined together as a global gathering operation.

This scheme as it stands is likely to be limited by the amount of memory available on each processor, but a compound scheme might involve partitioning the model grid up as well as the observation data so that each processor has some regular or scattered spatial part of the model grid as well as its own subset of observations.

5 Acknowledgements

Edinburgh Parallel Computing Centre is supported by major grants and contracts from the Advisory Board for the Research Councils, the Department of Trade and Industry, the Information Systems Committee of the Universities Funding Council, the Science and Engineering Research Council, Scottish Enterprise Software Group, and industry. The work reported here was also funded directly by the UK Meteorological Office.

References

[1] R. S. Bell and A. Dickinson, 1990, The Meteorological Office Unified Model for Data Assimilation, Climate Modelling and NWP and its Implementation on a CRAY Y-MP, in Fourth Workshop on Use of parallel processors in meteorology, ECMWF, 1990

[2] A. C. Lorenc, R. S. Bell and B. Macpherson, 1991, The Meteorological Office analysis correction data assimilation scheme, in Q.J.R.Meteol. Soc. 117, pp59-89

[3] Unified Model Documentation paper 30, UK Meteorological Office, 1991.

[4] M. J. Suarez, Atmospheric Modelling on a SIMD Computer, in Multiprocessing in Meteorological Models edited by G. -R. Hoffmann and D. F. Snelling, Published by Springer-Verlag, 1988.

[5] Saulo R. M. Barres and Tuomo Kauranne, 1990, Spectral and Multigrid Spherical Helmholtz Equation Solvers on Distributed Memory Parallel Computers, in Fourth Workshop on Use of parallel processors in meteorology, ECMWF, 1990

[6] David Dent, 1990, The ECMWF Model on the CRAY Y-MP8, in Fourth Workshop on Use of parallel processors in meteorology, ECMWF, 1990

[7] S. F. B. Tett, A Massively Parallel Algorithm for the Spectral Method, in Fourth Workshop on Use of parallel processors in meteorology, ECMWF, 1990

[8] Unified Model Documentation paper 10, UK Meteorological Office, 1991.

[9] ACexpt Report: Profiling, Porting, Performance and Parallelisation. EPCC Technical Report, 1992.

[10] Unified Weather/Climate Model Parallel Implementation Feasibility Study. K. A. Hawick, EPCC Report, 1991.

[11] Draft High Performance Fortran Standard

[12] Connection Machine Series CM-200 Technical Summary, Thinking Machines Corporation.

[13] PhD Thesis, S. F. B. Tett, University of Edinburgh, 1992.