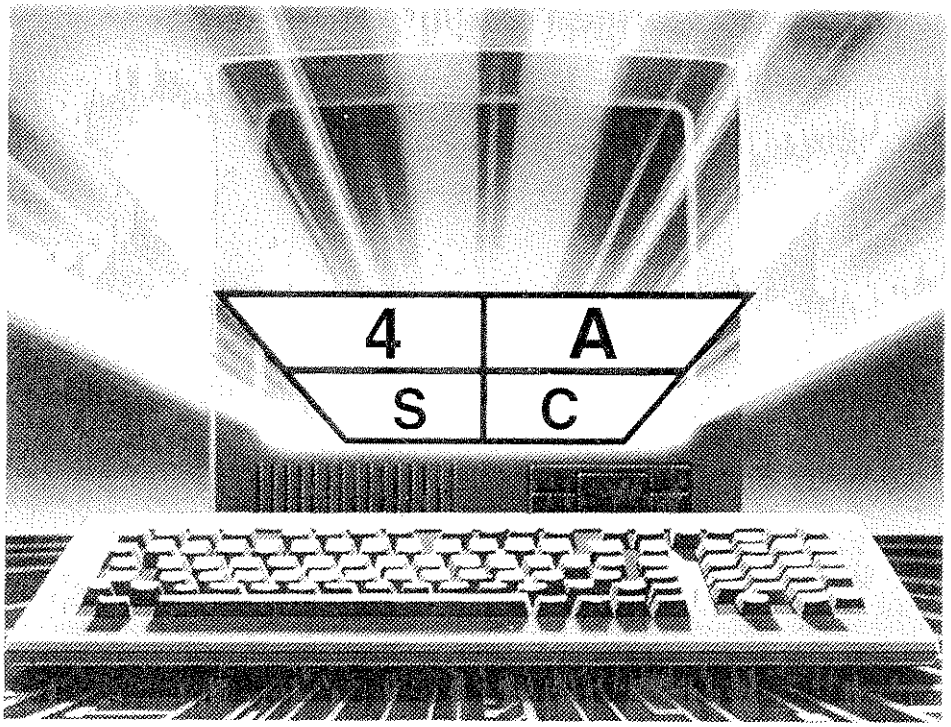

FOURTH AUSTRALIAN SUPERCOMPUTER CONFERENCE

"VISUALISE THE POSSIBILITIES"

UNIVERSITY PARK HOTEL - BOND UNIVERSITY - GOLD COAST
AUSTRALIA 2 - 5 DECEMBER 1991

CONFERENCE HANDBOOK AND PROCEEDINGS

Sponsored by

CRAY RESEARCH (AUSTRALIA)

CONVEX

Thinking Machines Corporation

digital

Hosted by

TECHQUAD

Promoted by

Australian Computer Society

Exploiting Parallelism in a Numerical Weather Model

Pau S. Chang
Greg K. Egan

Laboratory for Concurrent Computing Systems
Swinburne Institute of Technology
John St, Hawthorn 3122, Victoria

pau@stan.xx.swin.oz.au
Tel: 03 8198681
Fax: 03 8196443

Abstract

This paper reports our continued attempt in exploiting parallelism in the dynamics of an Australian region numerical weather prediction model. Reported in [2], parallelisation was performed for the original FORTRAN source, using an EPF (Encore Parallel FORTRAN) compiler, with good performance results after some manual fine tuning. Continuing that work, we also transliterated the FORTRAN source to SISAL, a functional language claimed to be suitable for exploiting program parallelism on shared memory multiple processor machines. Despite the presence of some known overheads, the initial results show that the SISAL code also performs competitively.

Introduction

Under a collaboration between the Laboratory for Concurrent Computing Systems and the Australian Bureau of Meteorology Research Centre, we are currently studying a number of numerical weather prediction (NWP) models in terms of their inherent parallelism. One of these is an Australian Region NWP model. Originally intended for an ETA-10 supercomputer, this model is under development at the Australian Bureau of Meteorology Research Centre in Melbourne by Leslie and Dietachmayer [1] for short-term forecasting (up to 36 hours) over the Australian region. Presently, the model consists of only the dynamics components.

In [2], we parallelised the model in the machine specific EPF FORTRAN [3], using an EPF compiler. Good performance results were achieved after some manual fine tuning. Now we have also transliterated the FORTRAN source to SISAL, a functional language claimed to be suitable for exploiting program parallelism on shared memory multiple processor machines [4]; the SISAL code can be ported to uniprocessor systems such as the SUN and IBM RS6000, as well as multiple processor machines such as Encore Multimax, Sequent Balance, Alliant, multiple processor Vax, CRAY supercomputers and dataflow machines. In our experiments, dummy datasets were used because the model was still not completely developed. In the SISAL version, therefore, we had to explicitly include an array bound check routine to make the code run smoothly. Nonetheless, the SISAL code still yielded competitive performance compared to the performance of the EPF FORTRAN code.

About the Weather Model

The mathematical and computational aspects of the model have been shown and described in [2]. Briefly, on the mathematical side, the model performs integrations of adiabatic equations using split-explicit method. It solves basic NWP equations, which are those of Miyakoda [5], for wind velocity, temperature and surface pressure field variables in a nested environment; the former two are three dimensional matrices representing horizontal x-y space and a vertical σ-coordinate (pressure levels). The model is implemented in Arakawa-A non-staggering grids [6].

On the computational side, the code spends most of its time in the time stepping section, which also is the focus of our parallelisation efforts. And the most computationally intensive parts of the model are in subroutine `Tstep` which computes a forward time step. It first performs semi-Lagrangian treatment of the horizontal advection of u, v and θ using polynomial interpolation of arbitrary order. Following that, it computes the new values of the field variables in an adjustment step. The adjustment step is basically forward or backward with some modifications. The new values are then blended with the external nesting data. In other features of the model, the Coriolis term may be optionally forward, centred or backward, the vertical advective flux across the layer interfaces is calculated using first-order upwinding, and spatial derivatives are evaluated to fourth-order accuracy. The integrated diagnostic quantities such as kinetic energy, potential energy, total energy, mean vertical motion and mass are evaluated to ensure their conservation. Also, noise accumulates at high frequencies as integrations proceed; low pass filtering is therefore performed, once in every few time loop iterations, to eliminate any resultant aliasing. On the whole, the model has a simple computational structure.

Parallelisation Aspects

Parallelisation has been attempted previously at the EPF FORTRAN level [2], and now at the SISAL level.

At the FORTRAN level, the code was first automatically parallelised using an EPF (Encore Parallel FORTRAN) compiler; the compiler is capable of converting a standard FORTRAN program into a source annotated with parallel primitives. However, as with other automatic parallel annotators, the EPF annotator also requires user intervention to produce more efficient codes. In this case, we had to reorganise the code parallelism in the EPF FORTRAN version generated by the compiler. Fortunately, this fine tuning effort was already usefully reduced by the compiler's annotation pass. However, the parallelised EPF FORTRAN code has a drawback in contrast to the SISAL source, in that it is machine specific to Encore Multimax multiprocessors.

In the SISAL implementation, as we did in the EPF FORTRAN implementation, we targeted exploiting the most common source of parallelism, which is the `for` loop expression. The code was first transliterated from the original FORTRAN version, then all loops were transformed to SISAL's parallel loop form wherever possible. Our previous experience in the parallel implementation of a spectral barotropic weather model in SISAL [7] was valuable, in that we had learnt to cope with SISAL's single assignment rule, its unique way of exposing inherent parallelism, which at times causes clumsiness in programming. Constructing SISAL's parallel loops was simple using the loop transformations method described in [8], and even more so in this grid model. At compile time, SISAL's compiler for multiprocessors, OSC (Optimizing SISAL Compiler) [9], evaluates the parallel costs of these loops, and will slice/parallelise them accordingly if the costs are desirable.

However, as this model is not yet complete in development, dummy data sets have been used for the main field variables in our experiments. In the time stepping section, there are some parts which computationally depend on these field variables, and as a result inevitably access out of bound elements in some arrays. While this is allowed in FORTRAN, SISAL's OSC compiler is unforgiving; so in our current SISAL implementation, we have deliberately included a routine to check array bounds so that the SISAL program could run smoothly, but

this unfortunately has also resulted in quite significant performance overheads. Nonetheless, this problem will become trivial when appropriate data sets are used in a fully developed model in the future.

Experimental Results

The codes have been run on an Encore Multimax multiprocessor consisting of six XPC-processors, running UMAX 4.3, at Swinburne Institute of Technology, and a four CPU CRAY XMP/48 at Lawrence Livermore National Laboratory. The various runtime and speedup results, as tabulated in Table 1 and Table 2, are for the model size of 65 by 40 grid (approximately 150km between grid points) by 12 σ-levels, and for one time step.

Encore Multimax

# CPUs	Original FORTRAN	EPF FORTRAN				SISAL		
P	Time(sec)	Time(sec)	Speedup S	Speedup Sco	Efficiency (S/P)	Time(sec)	Speedup Ss	Efficiency (Ss/P)
1	57.7	58.4	1.00	0.99	100%	58.5	1.00	100%
2	-	29.5	1.98	1.96	99%	32.6	1.79	90%
3	-	20.9	2.79	2.76	93%	23.2	2.52	84%
4	-	16.2	3.60	3.56	90%	18.1	3.23	81%
5	-	15.4	3.79	3.75	76%	18.0	3.24	65%
6	-	12.7	4.60	4.54	77%	14.9	3.92	65%

Table 1: Results obtained from an Encore Multimax multiprocessor

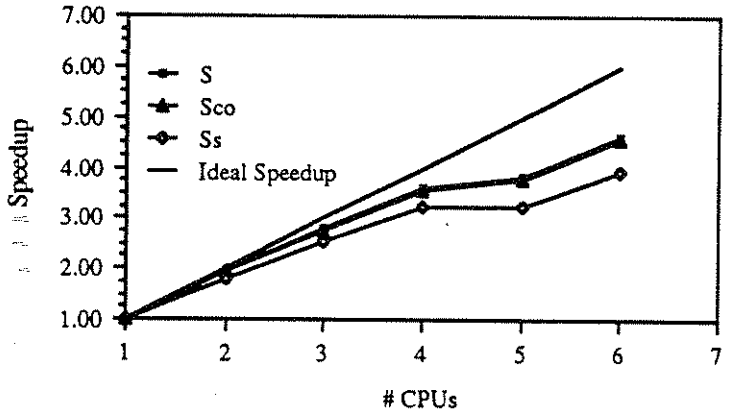

Figure 1: Various speedup curves

The results of executing the codes on the Encore Multimax machine are tabulated in Table 1. On a single processor, the execution time for the EPF FORTRAN version is only fractionally longer than the original non-parallel FORTRAN version (both fully optimised). This shows that the overall overheads contributed by the parallelisation in EPF is negligible. When multiple processors are used simultaneously to share the work load, the execution times are close to the ideal.

The values of Sco (the speedup of the EPF FORTRAN version over the standard FORTRAN version of the model) show that the speedup gained in runtime resulting from the parallel implementation is desirable. On the other hand, the values of S (the speedup performance of the EPF version) indicate that the EPF compiler gives very good parallel processing support to the implementation. The curves of Sco and S in Figure 1 almost coincide, indicating an efficient and encouraging result with good system utilisation (efficiency) in a multiple processor environment.

The SISAL compiler used for this experiment is OSC Version 8.5. Although the speedup performance of the current SISAL code is not as good as that of EPF FORTRAN, their single processor runtimes are, nonetheless, close. The results indicate that the functional language SISAL, and its compiler, can also provide very good parallel processing support. There are two known major overheads in this case. One is the deliberately added array bound check routine which was mentioned above. The other is due to the OSC generated code which allocates and deallocates arrays. The compiler currently does not optimise this code, which automatically reclaims memory between cycles of execution. For our SISAL implementation, these repetitive operations are unnecessarily costly. The allocation of storage could take place once, before the loop begins execution, and storage could be freed after the loop completes; hence an *aggregate preconstruction* optimiser is desperately needed [10]. This problem is not new [7], and is being attacked in the hope of achieving substantial improvements in both sequential and parallel performance of SISAL programs [10].

The speedups gained in the SISAL version, Ss, and the EPF version, S, indicate that there is a large amount of inherent parallelism to be exploited in the dynamics of this weather model, and that scalable speedups could be expected if more processors are added.

CRAY XMP/48

Having carried out our experiments on an Encore Multimax machine, it is useful to test the same codes on a multiple CPU supercomputer. Unfortunately, we could only manage some very limited access to a CRAY XMP/48, running UNICOS 5.1, located at the Lawrence Livermore National Laboratory; and we only managed to obtain results for the SISAL version (Table 2), and not for the standard FORTRAN version.

# CPUs	CPU Time (sec)	WallClock Time (sec)	Percentage CPU Utilization	Speedup (CPU Time)
1	3.8867	4.5575	85.3%	1.0
4	1.3931	6.8203	20.4%	2.79
	1.3740	6.8203	20.1%	
	1.2554	6.8167	18.4%	
	1.2885	6.8207	18.9%	

Table 2: Results of the SISAL version on a CRAY XMP/48

The XMP system was heavily loaded at the time the code was run, as indicated by the low figures of percentage machine utilisation. However, with 4 CPUs, the parallelism in the SISAL code offers a speedup (based on CPU times) of nearly 3, or 70% of the ideal. These results show that the model can potentially achieve good parallel performance on a multiple CPU CRAY machine through exploiting parallelism in SISAL.

Conclusions

A large amount of parallelism in the dynamics of this grid weather model has been realised. While the exploitation of parallelism in the machine dependent EPF FORTRAN is efficient, SISAL as a functional language also allows parallelism to be effectively expressed to achieve good speedup in addition to code portability. The parallel performance of our SISAL implementation of the weather model so far has been encouraging and competitive compared to the implementations in the original FORTRAN and the EPF FORTRAN, and is expected to improve substantially when an *aggregate preconstruction* optimiser is included in OSC.

Acknowledgements

The original standard FORTRAN version of the weather model was developed and provided by Dr. Lance Leslie and Dr. Gary Dietachmayer of the Australian Bureau of Meteorology Research Centre. The runtimes of the SISAL code on a CRAY machine were obtained from the CRAY XMP/48 at Lawrence Livermore National Laboratory with the invaluable assistance of Dave Cann. This project is conducted under a collaboration between the Laboratory for Concurrent Computing Systems at Swinburne Institute of Technology, and the Australian Bureau of Meteorology Research Centre.

References

[1] L.M. Leslie et al., "A High Resolution Primitive Equations NWP Model for Operations and Research", pp. 11-35, Australian Meteorological Magazine, No. 33, March 1985.

[2] P.S. Chang and G.K. Egan, "Parallelising an Australian Region Numerical Weather Prediction Model", Third Australian Supercomputer Conference, December 1990.

[3] Encore Computer Corporation, Multimax Technical Summary.

[4] J. McGraw et al., "SISAL: Streams and Iteration in a Single Assignment Language, Language Reference Manual Version 1.2", Memo 146, Lawrence Livermore National Laboratory, March 1985.

[5] K. Miyakoda, "Cumulative Results of Testing a Meteorological-Mathematical Model", pp. 99-130, Royal Irish Academy Proceedings, July 1973.

[6] A. Arakawa and V. R. Lamb, "Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model", pp. 174-265, 337, Methods of Computational Physics, Vol. 17, Academic Press, 1977.

[7] P.S. Chang and G.K. Egan, "An Implementation of a Barotropic Numerical Weather Prediction Model in the Functional Language SISAL", pp. 109-117, Proceedings of the Second ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, SIGPLAN Notices, Vol. 25, No. 3, March 1990.

[8] P.S. Chang, "Implementation of a Numerical Weather Prediction Model in SISAL", *Master's Thesis*, Technical Report 31-017, Laboratory for Concurrent Computing Systems, Swinburne Institute of Technology, June 1990.

[9] D. Cann, "Compilation Techniques for High Performance Applicative Computation", Technical Report CS-89-108, Colorado State University, May 1989.

[10] D. Cann, "Retire FORTRAN? A Debate Rekindled", Technical Report UCRL-JC-107018, Lawrence Livermore National Laboratory, April 1991.