LABORATORY FOR CONCURRENT COMPUTING SYSTEMS

COMPUTER SYSTEMS ENGINEERING
School of Electrical Engineering
Swinburne Institute of Technology
John Street, Hawthorn 3122, Victoria, Australia.

Analysis of a Parallel Implementation of a Numerical Weather Model in the Functional Language SISAL

Technical Report 31-012

Pau S. Chang
Greg K. Egan

March 1990

Submitted to PARALLEL 90, 4-6 September 1990, Southampton, UK.

ABSTRACT

A one-level barotropic spectral Numerical Weather Prediction (NWP) model has been implemented [4] in the high-level parallel functional language SISAL [3]. In furthering that work, our attention has been focused on identifying the sources of the sequential sections in the concurrency profile for the timeloop section, and in parallelising a very significant serial code section in the initialisation section of the spectral model. The solutions of the encountered problems and their effects, the subsequent code refinements and performance, and the study of issues related to the effective use of current and next generation multiprocessors raised as a result of the analysis are discussed in this paper.

Analysis of a Parallel Implementation of a Numerical Weather Model in the Functional Language SISAL

Pau S. Chang and Gregory K. Egan

Laboratory for Concurrent Computing Systems, Computer Systems Engineering,
School of Electrical Engineering, Swinburne Institute of Technology,
P.O. Box 218 Hawthorn, Victoria 3122, Australia

ABSTRACT

A one-level barotropic spectral Numerical Weather Prediction (NWP) model has been implemented [4] in the high-level parallel functional language SISAL [3]. In furthering that work, our attention has been focused on identifying the sources of the sequential sections in the concurrency profile for the timeloop section, and in parallelising a very significant serial code section in the initialisation section of the spectral model. The solutions of the encountered problems and their effects, the subsequent code refinements and performance, and the study of issues related to the effective use of current and next generation multiprocessors raised as a result of the analysis are discussed in this paper.

INTRODUCTION

Numerical Weather Prediction (NWP) is acknowledged as being of vital importance to the Australian and world economies. The demand that NWP places on computing system performance has increased dramatically since the introduction of computer systems. As technological limits are approached in component performance, many computer manufacturers are turning to multiprocessor configurations to obtain increased performance. Although the underlying application may exhibit large amount of inherent parallelism, in many cases this has been lost in the formulation of sequential and vector systems; additionally there is a strong suggestion that existing language systems will prove inadequate for the new multiprocessors.

In our earlier feasibility study [4], we implemented a one-level barotropic spectral NWP model [1] using the high-level parallel functional language SISAL (Streams and Iteration in a Single Assignment Language) [3]. The analysis of the initial results which were obtained using an Optimising SISAL Compiler (OSC) [2] for a shared-memory ENCORE Multimax with 16 APC (32332/32081) processors leads to further analysis and results presented in [5] and the continued study of issues related to the effective use of current and next generation multiprocessors [6].

In furthering the earlier work, our attention has been focused on identifying the sources of the sequential sections on the concurrency profile of the timeloop section (Figure 2), and in parallelising a very significant serial code section which evaluates Legendre polynomials in the initialisation stage of the spectral model (Figure 1). The research has indicated that the current fixed global parallel execution cost estimator in OSC may lead to unsatisfactory results in some cases. The subsequent code refinements and the resulting model performance of the implementation will also be discussed.

1 MATHEMATICAL DESCRIPTION OF THE MODEL [1]

For the purpose of the feasibility study, the one-level spectral model is representative of a full multi-level spectral model. Inspection of the equations describing the one-level model suggests very high potential concurrency. In its primitive form, the model is expressed in terms of the vorticity and divergence of the horizontal wind field as shown in Equation 1 to 8.

Definitions of symbols used:

$\underline{V}$	=	wind vector (east U and north V)	D	=	horizontal divergence
ψ	=	stream function	J	=	wave number truncation (resolution)
∇	=	horizontal gradient operator	Ω	=	angular velocity of earth
χ	=	velocity potential	a	=	radius of earth
$\underline{k}$	=	vertical unit vector	ζ	=	vertical component of relative vorticity
Φ	=	geopotential height of the surface			
Φ'	=	time dependent perturbation field	Φ^*	=	global mean geopotential

ϕ and λ are spherical coordinates

$$\zeta = \underline{k} \cdot \nabla \times \underline{V} = \nabla^2 \psi \tag{1}$$

$$\Phi = \Phi^* + \Phi' \tag{2}$$

$$D = \nabla \cdot \underline{V} = \nabla^2 \chi \tag{3}$$

$$\frac{\delta(\nabla^2\psi)}{\delta t} = \frac{-1}{a \cos^2\phi} \left[\frac{\delta(U\nabla^2\psi)}{\delta\lambda} + \cos\phi \frac{\delta(V\nabla^2\psi)}{\delta\phi} \right] - 2\Omega \left(\sin\phi \nabla^2\chi + \frac{V}{a}\right) \tag{4}$$

$$\frac{\delta(\nabla^2\chi)}{\delta t} = \frac{1}{a \cos^2\phi} \left[\frac{\delta(V\nabla^2\psi)}{\delta\lambda} - \cos\phi \frac{\delta(U\nabla^2\psi)}{\delta\phi} \right] + 2\Omega \left(\sin\phi \nabla^2\chi - \frac{U}{a}\right) - \nabla^2 \left[\frac{U^2 + V^2}{2 \cos^2\phi} + \Phi' \right] \tag{5}$$

$$\frac{\delta\Phi'}{\delta t} = \frac{-1}{a \cos^2\phi} \left[\frac{\delta U\Phi'}{\delta\lambda} + \cos\phi \frac{\delta V\Phi'}{\delta\phi} \right] - \Phi^* D \tag{6}$$

$$U = \frac{-\cos\phi}{a} \frac{\delta\psi}{\delta t} + \frac{1}{a} \frac{\delta\chi}{\delta\lambda} \tag{7}$$

$$V = \frac{1}{a} \frac{\delta\psi}{\delta\lambda} + \frac{\cos\phi}{a} \frac{\delta\chi}{\delta\phi} \tag{8}$$

Integration of the primitive equations is facilitated by a spectral grid transform technique which arises in evaluation of the nonlinear products. The first step of this technique is to obtain the truncated expansions for approximating the stream function, geopotential height and two derived wind fields U and V illustrated in Equations 10 to 15.

$$\psi = a^2 \sum_{m=-J}^{+J} \sum_{r=|m|}^{|m|+J} \psi_r^m \, Y_r^m \quad (9) \qquad \Phi = a^2 \sum_{m=-J}^{+J} \sum_{r=|m|}^{|m|+J} \Phi_r^m \, Y_r^m \quad (10)$$

$$U = a \sum_{m=-J}^{+J} \sum_{r=|m|}^{|m|+J+1} U_r^m \, Y_r^m \quad (11) \qquad V = a \sum_{m=-J}^{+J} \sum_{r=|m|}^{|m|+J+1} V_r^m \, Y_r^m \quad (12)$$

where $Y_r^m = P_r^m (\sin\phi) \, e^{im\lambda}$

$P_r^m (\sin\phi) = alp_r^m$ = Normalised Legendre Polynomial

$$\int_{-\pi/2}^{\pi/2} P_r^m (\sin\phi) \, P_r^m (\sin\phi) \, \cos\phi \, d\phi = 1 \quad (13)$$

and $$U_r^m = (r-1) \, \rho_r^m \, \psi_{r-1}^m - (r+2) \, \rho_{r+1}^m \, \psi_{r+1}^m + im\chi_r^m \quad (14)$$

$$V_r^m = -(r-1) \, \rho_r^m \, \chi_{r-1}^m + (r+2) \, \rho_{r+1}^m \, \chi_{r+1}^m + im\psi_r^m \quad (15)$$

where $$\rho_r^m = \sqrt{\frac{(r^2 - m^2)}{(4r^2 - 1)}}$$

This is followed by a Fast Fourier Transformation (FFT) of these fields to the Gaussian latitude-longitude grid on the globe. The nonlinear products of Equations 4 to 8 are those on the left hand sides of Equations 16 to 20. They can now be obtained by direct multiplications in the grid domain.

$$U\nabla^2\psi = a \sum_{m=-J}^{+J} A_m \, e^{im\lambda} \quad (16) \qquad V\nabla^2\psi = a \sum_{m=-J}^{+J} B_m \, e^{im\lambda} \quad (17)$$

$$U\Phi' = a^3 \sum_{m=-J}^{+J} C_m \, e^{im\lambda} \quad (18) \qquad V\Phi' = a^3 \sum_{m=-J}^{+J} D_m \, e^{im\lambda} \quad (19)$$

$$\frac{U^2 + V^2}{2} = a^2 \sum_{m=-J}^{+J} E_m \, e^{im\lambda} \quad (20)$$

The next step is an inverse FFT of these terms as described in Equations 16 to 20 and the final step is to transform these fields back to the spectral domain. The final spectral forms of the model are described in Equations 21 to 24.

$$-r(r+1) \frac{\delta \psi_r^m}{\delta t} = -\int_{-\pi/2}^{\pi/2} \frac{1}{a \cos^2\phi} \left[imA_m P_r^m(\sin\phi) - B_m \cos\phi \frac{\delta P_r^m(\sin\phi)}{\delta\phi} \right] \cos\phi \, d\phi$$

$$+ 2\Omega \left[r(r-1) \, \rho_r^m \, \chi_{r-1}^m + (r+1)(r+2) \, \rho_{r+1}^m \, \chi_{r+1}^m - V_r^m \right] \quad (21)$$

$$-r\,(r+1)\frac{\delta\chi_r^m}{\delta t} = \int_{-\pi/2}^{\pi/2} \frac{1}{\cos^2\phi}\left[\, imB_m P_r^m(\sin\phi) + A_m \cos\phi \frac{\delta P(\sin\phi)}{\delta\phi}\right]\cos\phi\, d\phi$$

$$- 2\Omega\left[\, r(r-1)\,\rho_r^m\,\psi_{r-1}^m + (r+1)(r+2)\,\rho_{r+1}^m\,\psi_{r+1}^m + U_r^m\,\right]$$

$$+ r(r+1)(E_r^m + \Phi_r^m) \qquad (22)$$

$$\frac{\delta\Phi_r^m}{\delta t} = -\int_{-\pi/2}^{\pi/2} \frac{1}{\cos^2\phi}\left[\, imC_m P_r^m(\sin\phi) - D_m \cos\phi \frac{\delta P_r^m(\sin\phi)}{\delta\phi}\right]\cos\phi\, d\phi$$

$$+ \Phi^*\, r(r+1)\,\chi_{r+1}^m \qquad (23)$$

$$E_r^m = \int_{-\pi/2}^{\pi/2} \frac{E_m}{\cos^2\phi} P_r^m(\sin\phi)\cos\phi\, d\phi \qquad (24)$$

The detailed mathematical expressions and descriptions of this NWP model can be found in [1] and [5]. In this paper, the model size is expressed in terms of its *resolution number* J. The *spectral truncation limits* jx, jxx and mx are related to J where jx = mx = J + 1 and jxx = J + 2. The *number of latitudes* of the globe and the *number of longitudinal* points on each latitude are also related to J by ilat = (5 * J + 1) / 2 and ilong = 3 * J + 1 respectively.

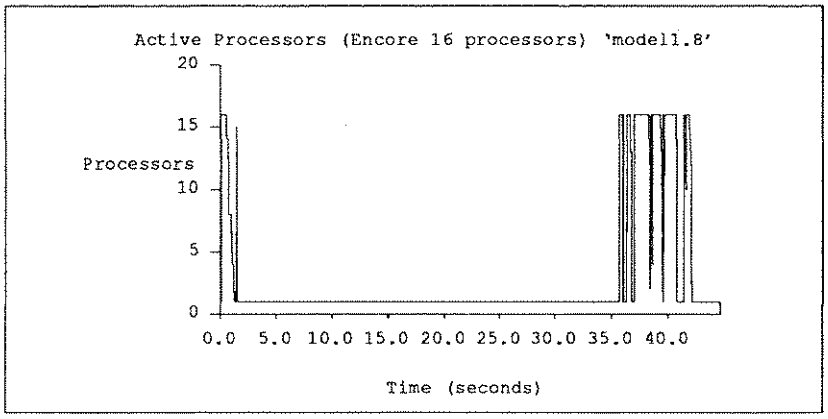

Figure 1: The concurrency profile of the full model from the implementation in [4]

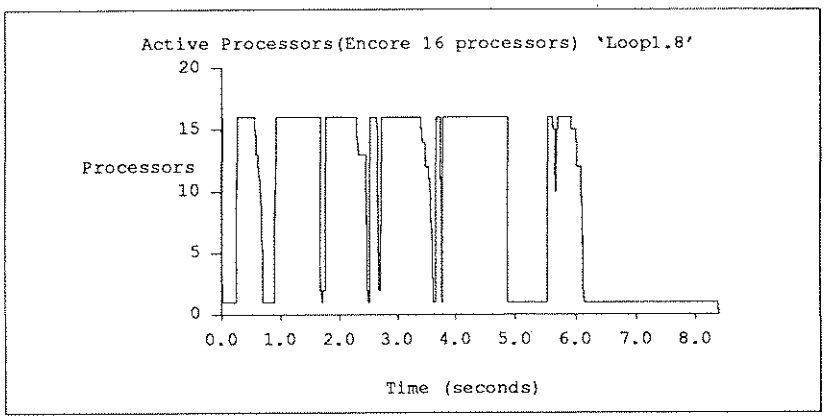

Figure 2: The concurrency profile of the timeloop extracted from the model

2 SEQUENTIAL CODE SECTIONS IN THE TIMELOOP

The parallelism profile of the initial SISAL implementation for a model size of J = 30 with 16 processors sharing the workload (Figures 2) indicates that the inter-function sequential or Amdahl [7] notches in the timeloop caused by data dependency have a second order effect on potential speedup. There still remain however three significant serial sections which consume approximately 13% of the total execution time and grow with problem size.

The sequential sections are due to three specific functions, all of which are involved in building single dimensional arrays from singly nested parallel **FOR** loops which contain small loop bodies. A number of subsequent experiments have shown that the OSC's slicing and parallelisation of small body singly nested parallel **FOR** loops which produce single dimensional arrays are globally determined by the compile time routine which estimates the parallel execution costs, and these loops are not sliced due to the failure of the OSC cost estimator to recognise the critical path significance of these functions. Discussions of a similar issue, but for a dataflow architecture, on the relationship between the body of a parallel **FOR** loop and the exploitable concurrency residing in it can also be found in [6].

2.1 SINGLY NESTED PARALLEL LOOP WITH A SIMPLE LOOP BODY

SISAL presently does not implicitly support the data structure for complex numbers. Hence a complex number is represented by a **record** of two numbers, *Repart* and *Impart* in our implementation. Figure 3a shows one of these functions which explicitly converts four arrays of real numbers, each having an array size of jx * mx * 2, to four corresponding arrays of complex numbers of array size jx * mx each. Regardless of the number of processors used, the total length of the serial code on the concurrency profile undesirably increases with the size of the loop bound.

Nevertheless, this code can be locally compiled with the maximum slicing of the parallel **FOR** loop enforced by using the -H1 pragma of OSC. The Local Maximum Slicing or LMS curves in Figures 4a and 4b illustrate the desired improvement to this code as a result. However, the present OSC does not support the linkage to a separately compiled routine, and therefore the amount of slicing of parallel **FOR** loops can only be specified as a globally effective OSC pragma at compile time. Unfortunately a pragma value of -H1 results in slicing of the routines of interest but leads to over-parallelisation of the rest of the program [2]. This in turn results in an execution time of 30 seconds compared with the original 8 seconds for the model size J = 30.

```
ctC, eC, ptC, ztC :=
  FOR complex_index IN 1, jxmx
  index := complex_index * 2
  RETURNS  ARRAY of RECORD CplexReal[Repart : ct[index - 1]; Impart : ct[index]]
           ARRAY of RECORD CplexReal[Repart : e[index - 1]; Impart : e[index]]
           ARRAY of RECORD CplexReal[Repart : pt[index - 1]; Impart : pt[index]]
           ARRAY of RECORD CplexReal[Repart : zt[index - 1]; Impart : zt[index]]
  END FOR
```

*Figure 3a. A singly nested parallel **FOR** loop with a small loop body*

```
ctC, eC, ptC, ztC :=
  FOR m IN 1, mx
    ctC, eC, ptC, ztC :=
    FOR j IN 1, jx
    complex_index := jx * (m - 1) + j;
    index := complex_index * 2
    RETURNS    ARRAY of RECORD CplexReal[Repart : ct[index - 1]; Impart : ct[index]]
               ARRAY of RECORD CplexReal[Repart : e[index - 1]; Impart : e[index]]
               ARRAY of RECORD CplexReal[Repart : pt[index - 1]; Impart : pt[index]]
               ARRAY of RECORD CplexReal[Repart : zt[index - 1]; Impart : zt[index]]
    END FOR
 RETURNS  VALUE of CATENATE ctC
          VALUE of CATENATE eC
          VALUE of CATENATE ptC
          VALUE of CATENATE ztC
 END FOR
```

Figure 3b: A quasi doubly nested loop

A possible solution to this problem is to augment the cost estimation of OSC by providing the number of processors available on the target machine as an additional pragma. The likely effect of this solution can be demonstrated by explicitly slicing these loops using a *quasi doubly nested* (QDN) technique, which returns in this case the desired single dimensional arrays. The technique may be used effectively to force appropriate decisions from the current OSC cost estimator.

2.2 *QUASI DOUBLY NESTED* TECHNIQUE

Using the QDN technique, as shown in Figure 3b, the inner parallel **FOR** loop of loop bound jx computes the correct array indices. This loop resides inside an outer parallel **FOR** loop, of loop bound mx, which concatenates every temporary array it produces. The loop body of this outer loop hence becomes larger and an order of complexity higher. This technique produces a slightly larger code size but the overhead is felt only when one processor is employed. Furthermore, the execution time and concurrency profiles produced using this technique are the same as those produced when the original code is locally compiled with maximum slicing, as already discussed in the previous section. Figures 4a and 4b illustrate the more efficient exploitation of concurrency and an execution time of 3.5 times faster for this code relative to the original unparallelised code.

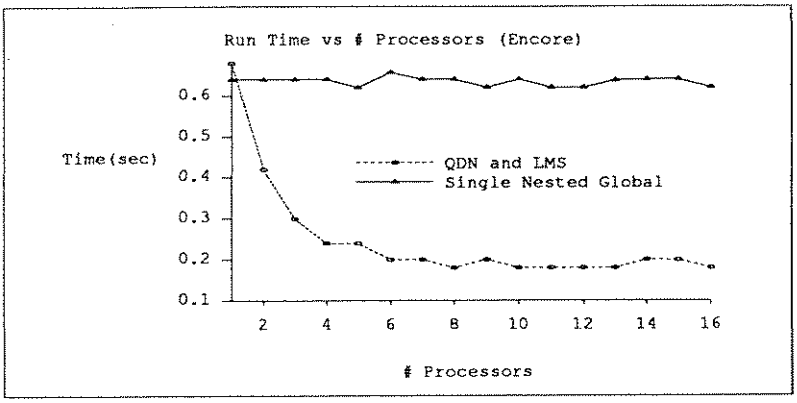

Figure 4a: The comparison of execution time as a function of the number of processors (J = 30)

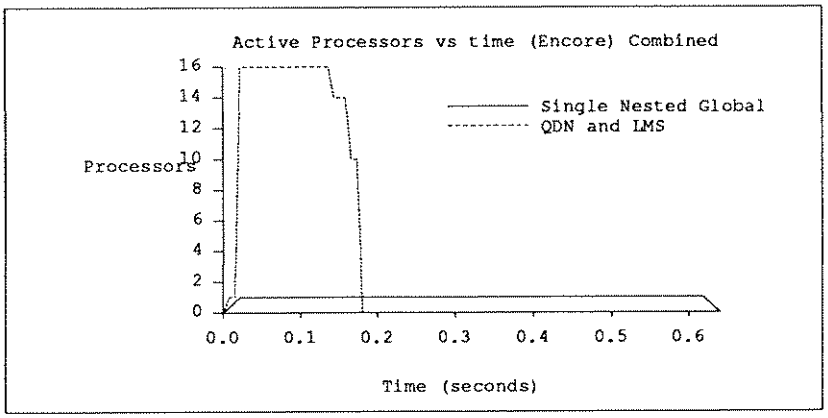

Figure 4b: The comparison of concurrency profiles as a function of the number of processors (J = 30)

3 CONCURRENCY IN THE COMPUTATION OF SPHERICAL HARMONICS

The serial computations for the Legendre polynomial of the first kind which produce the spherical harmonics of the globe [1] dominate the initialisation stages of both the FORTRAN and the initial SISAL models. These sequential computations when parallelised significantly speed up the initialisation stage.

3.1 SERIAL IMPLEMENTATION

The model groups the latitudes of the globe into ilat / 2 number of North-South latitude pairs. For each latitude pair, the function **LEGENDRE** computes jxx * mx number of spherical harmonics. The conventional implementation is sequentially conceived where both the computations for each of the harmonics on the same latitude pair and for each frame of latitude pairs are executed sequentially. In other words, there are a total of ilat / 2 * jxx * mx harmonics produced and hence the same number of corresponding serial array updates performed. The SISAL equivalent of the FORTRAN version, from direct transliteration, is shown in Figure 5a.

```
alp := FOR INITIAL
       WORKlgn := ARRAY_fill(1, jxxmx, 0.0d0);
       lat_level := 1;
       alp_LGN := LEGENDRE(ir, irmax2, jxxmx, coaiy[1], siaiy[1], deltaiy[1],
                           WORKlgn);
       WHILE lat_level < ilat / 2 REPEAT
       lat_level := old lat_level + 1;
       alp_LGN := LEGENDRE(ir, irmax2, jxxmx, coaiy[lat_level], siaiy[lat_level],
                           deltaiy[lat_level], old alp_LGN);
       RETURNS ARRAY of alp_LGN
       END FOR
```

Figure 5a: A sequential computation of the spherical harmonics

```
alp := FOR lat_level IN 1, ilat / 2
       alp_LGN := LEGENDRE(ir, irmax2, jxxmx, coaiy[lat_level], siaiy[lat_level],
                           deltaiy[lat_level]);
       RETURNS ARRAY of alp_LGN
       END FOR
```

Figure 5b: A parallel computation of the spherical harmonics

3.2 PARALLEL IMPLEMENTATION

The above SISAL routine can be conveniently parallelised using the SISAL forall construct (Figure 5b). In this version, all frames of latitude pairs (**LEGENDRE**) are computed concurrently because they are found to be data independent of each other. Figure 6a and 6b show the dramatic improvement in the run time and concurrency of the initialisation stage of the model for J = 30.

The improved performance over the initial implementation as illustrated in Figure 6b suggests that all processors have been kept busy throughout. **LEGENDRE** could be coded as a "wavefront" algorithm leading to additional improvement however, due to the satisfactory gains already obtained, this is not presently being pursued.

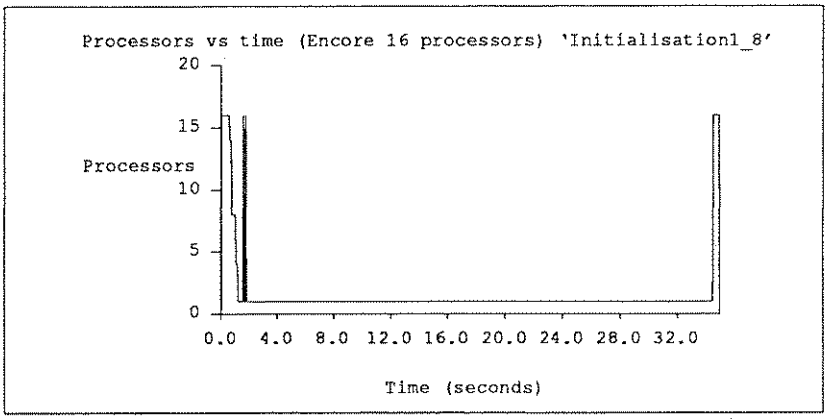

Figure 6a: The concurrency profile of the sequential SISAL implementation

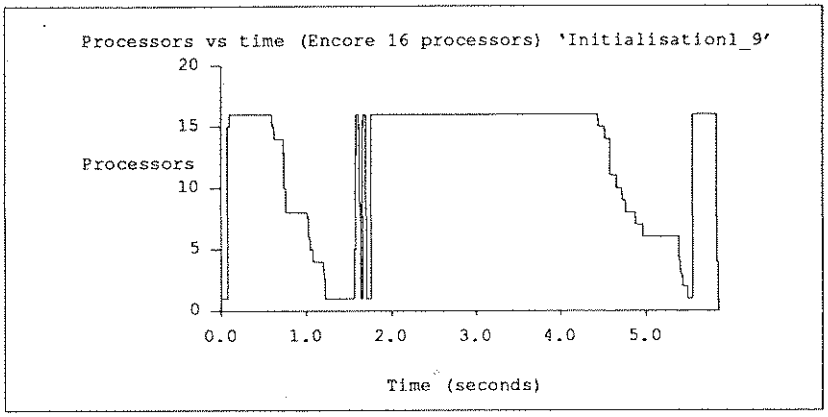

Figure 6b: The concurrency profile of the parallel SISAL implementation

4 MODEL PERFORMANCE FROM CODE REFINEMENTS

The available dataset only allows the model size to be increased up to J = 30. In order to show the capability of the new SISAL implementation in the exploitation of concurrency of larger model sizes, and its potential in providing substantial speedup over the sequential FORTRAN version, J has been extended beyond 30 by building additional dummy datasets which still perform the same amount of computation.

Figures 7a and 7b illustrate the performance of the refined implementation. The execution time profile (Figure 7a) indicates that the runtime of a small model size saturates quickly with increasing number of processors because there is not enough available parallelism to be exploited. However, when the model size grows larger, the increased available concurrency lowers the rate of saturation. The "Ideal" line shown assumes three unrealistic conditions that the SISAL code is perfectly parallel, the compiler is overhead free and the ENCORE Multimax architecture is fully capable of exploiting this parallelism. It is nonetheless included to show that the actual run time for model size J = 30 approaches ideal.

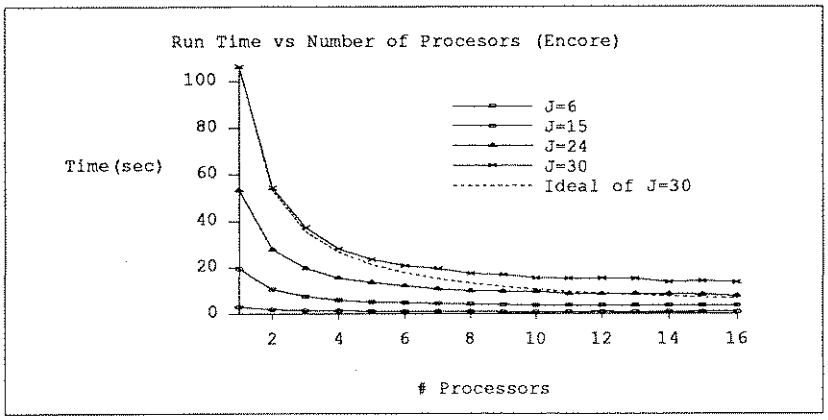

Figure 7a: The model execution time profile

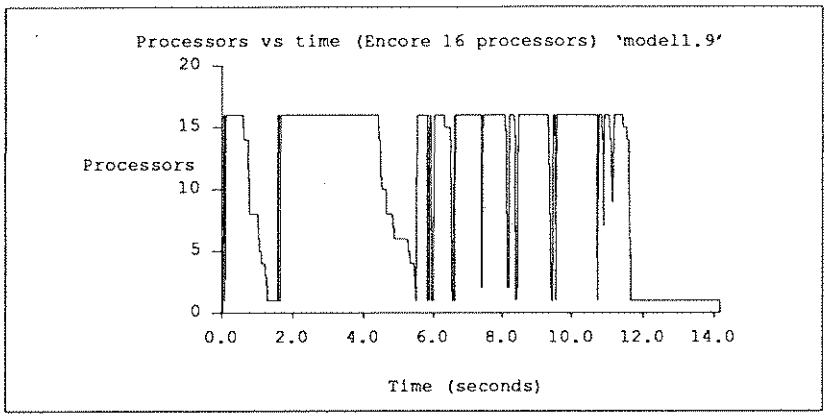

Figure 7b: The concurrency profile of the model for J = 30

The FORTRAN and the new SISAL Barotropic model implementations generally consist of an initialisation section in which all lookup tables and the initial values of weather variables are sorted, and a timeloop section in which the future weather states are computed. For a 24 hour forecast using the model size J = 30, the models will need to iterate 48 times in the timeloop, which therefore dominates the computation. Hence, the performance of the timeloop is particularly important.

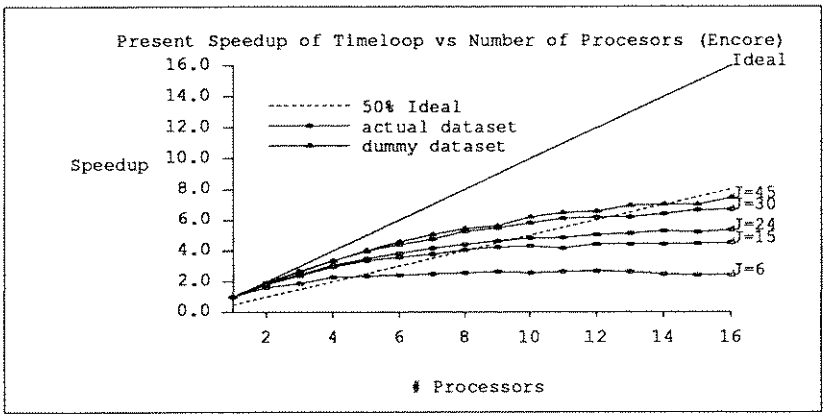

Figure 8a: The speedup profile of the timeloop for the present implementation

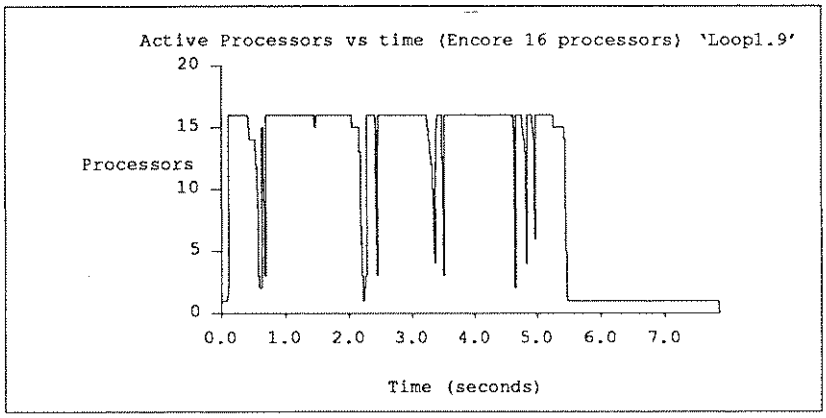

Figure 8b: The concurrency profile of the timeloop (J = 30) for the present implementation

Figures 8a and 8b illustrate the performance of the timeloop with varying model sizes for the present implementation. The importance of these results has produced further analysis in the eager memory deallocation routine of the OSC run time system [5]. That analysis proposes a better static analysis to determine that the array sizes are invariant through loop iterations and may be re-allocated in addition to a "lazy" deallocation of memory structures in parallel with the main computation only when necessary. The product is the removal of a significant section of serial code at the end of each time step in the timeloop, in the form of a long 'tail' (Figure 8b).

Figures 9a and 9b shows the effect that more efficient memory deallocation could have [5]. In this example we removed the deallocation code entirely. The result is a loop iteration with better scalability and significantly improved speedup characteristics.

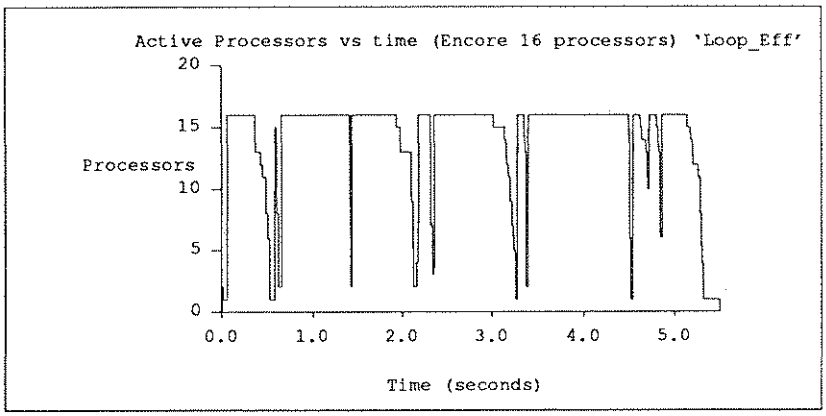

Figure 9a: The would be concurrency profile (J = 30) of the timeloop with an efficient implementation

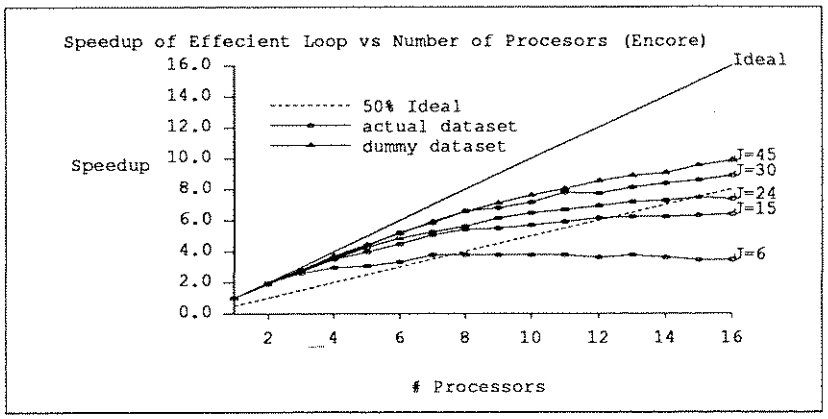

Figure 9b: The would be speedup of the timeloop with an efficient implementation

6 CONCLUSIONS

In this paper, we have described how a significant serial code section in the initialisation section of the adopted weather model was parallelised. We have also identified that the current global parallel execution cost estimation routine of OSC runtime system can lead to unsatisfactory results in some circumstances. We have suggested solutions and demonstrated the effect these solutions have on execution times. We have shown that the resulting improved SISAL implementation exhibits good scalability. This research will lead to further work by ourselves and the OSC developers in the implementation of a more appropriate memory deallocation scheme and parallel execution cost estimation function.

ACKNOWLEDGEMENTS

We would like to thank Drs Ian Simmonds and Ian Smith of the Department of Meteorology at the University of Melbourne for providing access to, and interpretation of, the original FORTRAN implementation of the barotropic spectral model. We thank Warwick Heath for his work on instrumentation. Our gratitude is also extended to all members of the Project team for their contributions to the discussions on memory allocation routine. This research was funded in part by the Royal Melbourne Institute of Technology and the Commonwealth Scientific and Industrial Research Organisation.

REFERENCES

1. W. Bourke An Efficient, One-Level, Primitive Equation Spectral Model, pp 683-689, Monthly Weather Review, Vol. 100, No. 9, September 1972.

2. D. C. Cann High Performance Parallel Applicative Computation, Technical Report CS 89-104, Colorado State University, February 1989.

3. McGraw et al SISAL: Streams and Iteration in a Single Assignment Language, Language Reference Manual, M146, Lawrence Livermore National Laboratories.

4. P. S. Chang and G. K. Egan A Parallel Implementation of a Barotropic Spectral Numerical Weather Prediction Model in the Functional Dataflow Language SISAL, Technical Report 31-016, Laboratory for Concurrent Computing Systems, School of Electrical Engineering, Swinburne Institute of Technology, August 1989.

5. P. S. Chang and G. K. Egan An Implementation of a Barotropic Spectral Numerical Weather Prediction Model in the Functional Language SISAL, pp. 109-117, SIGPLAN Notices, Vol. 25, No. 3, March 1990, Proceedings of the Second ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), Seattle, Washington, March 15-16, 1990.

6. G. K. Egan, N. J. Webb and W. Bohm Numerical Examples on the RMIT/CSIRO Dataflow Machine, Technical Report TR 118 082 R, Joint RMIT/CSIRO Parallel Systems Architecture Project, RMIT, May 1989.

7. D. Abramson and G. K. Egan Design Considerations for a High Performance Dataflow Multiprocessor, ACM Computer Architecture Symposium Workshop, Eilat, 1989, Prentice-Hall, in print.

8. G. Amdahl Validity of the Single-Processor Approach to Achieving Large-Scale Computer Capabilities, pp. 483-485, AFIPS Conference Proceeding, 1967.