THIRD AUSTRALIAN SUPERCOMPUTER CONFERENCE

THE UNIVERSITY
of MELBOURNE

DECEMBER 3-6, 1990

ORGANISED BY:
STRATEGIC RESEARCH FOUNDATION

THE UNIVERSITY OF MELBOURNE

Parallelising an Australian Region NWP Model

P.S. Chang
pau@stan.xx.swin.oz(.au)

G.K. Egan
gke@stan.xx.swin.oz(.au)

Laboratory for Concurrent Computing Systems
School of Electrical Engineering
Swinburne Institute of Technology
John Street, Hawthorn 3122, Victoria, Australia

Abstract

In this paper, we report the progress of a research project in parallel computing for an Australian Region Numerical Weather Prediction (NWP) model, which is conducted under a collaboration between the Laboratory and the Australian Bureau of Meteorology Research Centre. Presently, work is conducted on a shared-memory Encore Multimax multiprocessor. The intention is to parallelise the model for its eventual efficient execution on multi-CPU Cray supercomputers. The parallelisation has been successful with encouraging proven results and with negligible parallelisation overhead.

Introduction

A number of weather models are being studied under a collaboration between the Laboratory and the Australian Bureau of Meteorology Research Centre. In this paper, parallelisation of an Australian Region Numerical Weather Prediction model will be described.

Originally intended for ETA-10, the model is under development at the Australian Bureau of Meteorology Research Centre in Melbourne, by Leslie and Dietachmayer[1]. It will replace the model developed by Leslie et al.[2] for short-term forecasting (up to 36 hours) over the Australian region. Presently, the model consists of only the dynamics components. The physics components will be added in further development.

The mathematical aspects of the model will be outlined briefly. The computation and parallelisation of the model in EPF[3], a UMAX f77 implementation of FORTRAN, will be described.

About the Model

As described in Equations 1 to 7, the model solves the basic NWP equations, which are those of Miyakoda[4], for wind velocity, temperature and surface pressure in a nested environment. First, we define the model variables.

Let	u and v	=	horizontal wind speeds,
	w	=	vertical wind speed,
	θ	=	temperature,
	ϕ	=	geopotential height,
	p	=	pressure,
	p_s	=	surface pressure, and
	m	=	map factor

in the Cartesian (x,y) and σ-coordinates, where σ is defined as

$$p = p_s \cdot \sigma$$

and $$w = \frac{d\sigma}{dt}$$

Incorporating surface pressure p_s, the continuity equation is

$$\frac{\delta p_s}{\delta t} + m^2 \left[\frac{\delta}{\delta x}\left(\frac{p_s u}{m}\right) + \frac{\delta}{\delta y}\left(\frac{p_s v}{m}\right) \right] + p_s \frac{\delta w}{\delta \sigma} = 0 \tag{1}$$

The equations of motion are

$$\frac{du}{dt} - v(f + f_*) = -m\left(\frac{\delta\phi}{\delta x}\right)_p \tag{2}$$

$$\frac{dv}{dt} - u(f + f_*) = -m\left(\frac{\delta\phi}{\delta y}\right)_p \tag{3}$$

where f and f_* are the Coriolis and the metric parameters, and

$$\frac{d}{dt} = \frac{\delta}{\delta t} + m\left(u\frac{\delta}{\delta x} + v\frac{\delta}{\delta y}\right)_\sigma + w\frac{\delta}{\delta\sigma} \tag{4}$$

and $$\left(\frac{\delta\phi}{\delta x}\right)_p = \left(\frac{\delta\phi}{\delta x}\right)_\sigma + \frac{RT}{p_s}\left(\frac{\delta p_s}{\delta x}\right)_\sigma \tag{5}$$

The thermodynamic equation is expressed as

$$\frac{d\theta}{dt} = \frac{\delta\theta}{\delta t} + m\left(u\frac{\delta\theta}{\delta x} + v\frac{\delta\theta}{\delta y}\right)_\sigma + w\frac{\delta\theta}{\delta\sigma} \tag{6}$$

where $\theta = T\left(\frac{p_0}{p}\right)^{\kappa}, \quad \kappa = \frac{R}{c_p}$

p_0 being 1000mb, T the absolute temperature, R the gas constant and c_p the specific heat at constant pressure. Equation 6 is transformed to

$$\frac{d\theta}{dt} = 0$$

Finally, the combination of the equation of state and the hydrostatic equation is expressed as

$$\frac{\delta\phi}{\delta\sigma} = -\frac{RT}{\sigma}$$

or $$\phi = RT + \frac{\delta(\phi\sigma)}{\delta\sigma} \qquad (7)$$

Model Computation

The flow chart in Figure 1 shows the simple structure of the computational model which is implemented in Arakawa-A non-staggering grids[5]. In the initialisation section, the field storage inputs for model variables are read in and initialised in `Fsin` while model constants are defined in `Const`.

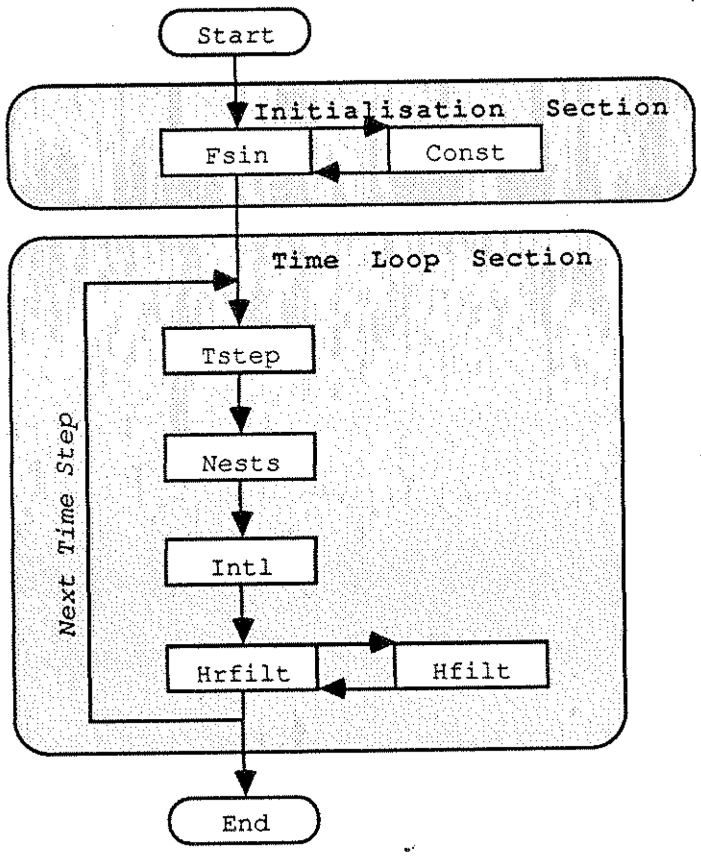

Figure 1: Flow chart showing the mechanism of the model

In the time looping section, the adiabatic equations are integrated in Tstep using split-explicit method. There are two parts in Tstep. The first part performs semi-Lagrangian treatment of the horizontal advection of u, v and θ using polynomial interpolation of arbitrary order. The second part is an adjustment step which computes the new values of the model variables. The adjustment step is basically forward or backward with some modifications. 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. Tstep thus forms the most computationally intensive part of the model. The new values of the model variables are then blended with the external nesting data in Nests. The integrated diagnostic quantities such as kinetic energy, potential energy, total energy, mean vertical motion and mass are evaluated in Intl to ensure their conservation. Finally, noise accumulates at high frequencies as integrations proceed. Therefore, low pass filtering must be performed in Hrfilt and Hfilt, every few number of time loop iterations to eliminate aliasing.

Parallelisation Aspects

The field variables of the model, except surface pressure, are three dimensional i.e. horizontal x-y space and a vertical σ-coordinate. The codes have been slightly restructured so that parallelism is readily visualised. One may parallelise at any of the three dimensions in most cases. In some specific cases, however, where summations are performed along the σ layer interfaces, the easiest path of parallelisation is at the σ dimension although one can still base on the other dimensions with some effort.

The model was originally written in FORTRAN, the language commonly used by weather modelers at the Research Centre. In order to sustain and ease software maintenance of the models in long term by these researchers, it was decided that parallelisation should be performed in FORTRAN rather than re-translating the codes to other languages such as C and SISAL. While unnecessary code restructuring was avoided, an important guide-line has been that code restructuring for parallelisation should be minimal, so that future model updates would be easy for the researchers.

EPF

EPF is the UMAX f77 implementation of FORTRAN-77, enhanced with parallel programming primitives to provide a parallel environment in a program and to synchronise the explicitly intended parallel executions. These primitives are PARALLEL, DOALL, LOCK WAIT, LOCK SEND, BARRIER, EVENT and CRITICAL SECTION.

The EPF compiler consists of analysis and transformation tools, a code generator, a parallelising compiler and a parallel runtime library. Thus it can be used to convert a standard FORTRAN program into a source annotated with parallel primitives. In the compilation process, it first evaluates the possible concurrency of program parts in .LST files. It then annotates those parts and generates the corresponding EPF programs (.E files).

As often do other automatic parallel annotators, the EPF annotator also requires user intervention to produce the most efficient codes for large programs. One may need to reorganise the code parallelism in the .E files generated by the EPF compiler; however, as in the case of this project, the parallelisation effort has already been usefully reduced by the compiler's annotation pass.

The form of annotation used by EPF, although similar to other parallelising FORTRAN compilers, are specific to the Encore Multimax multiprocessors on Unix BSD 4.3 (UMAX 4.3) and System V (UMAX V). This makes the EPF codes non-portable. However, the parallel structures of the model have been established. Thus if the codes were to be executed on other shared memory multi-CPU machines, only translation to the syntax of the new parallel environment is required.

A Simple Example

The following is an example of parallelising a standard FORTRAN code to EPF's .E form. Extracted from `Tstep`, this code attempts to compute an adjustment step for u and v described by the following equations:

$$\frac{du}{dt} = v(f + f_*) - w\,\frac{\delta u}{\delta\sigma} - m\,[\,\frac{\delta\phi}{\delta x} + RT\,\frac{\delta}{\delta y}\,(\ln p_s)\,]$$

$$\frac{dv}{dt} = u(f + f_*) - w\,\frac{\delta v}{\delta\sigma} - m\,[\,\frac{\delta\phi}{\delta y} + RT\,\frac{\delta}{\delta x}\,(\ln p_s)\,]$$

This example has been specifically chosen because it clearly shows how a code can be parallelised and the form of the annotation used by EPF. Most codes in the model are actually much more complicated and require significant tuning to achieve the fastest execution time.

In the standard FORTRAN form, the code may be programmed as:

```
      DO 620 K = 1,KMAX
         DO 620 I = I2+NPNS,ILM-NPNS
            DO 620 J = J2+NPNS,JLM-NPNS
C  Vertical advection  $w\frac{\delta u}{\delta\sigma}$  and  $w\frac{\delta v}{\delta\sigma}$
        UADV = ( WADU(K+1,I,J) - WADU(K,I,J) - U(K,I,J)*(W(K+1,I,J)-W(K,I,J)) )*DQI(K)
        VADV = ( WADV(K+1,I,J) - WADV(K,I,J) - V(K,I,J)*(W(K+1,I,J)-W(K,I,J)) )*DQI(K)

C  Coriolis and map-factor term $f$ and $f_*$.

        UCORMF = (1.0-CORCNT)*( F(I,J)+DUM1(K,I,J) )*V(K,I,J)
        VCORMF =-(1.0-CORCNT)*( F(I,J)+DUM1(K,I,J) )*U(K,I,J)
C  Pressure gradient terms  $\frac{\delta\phi}{\delta x}, \frac{\delta\phi}{\delta y}, \frac{\delta}{\delta y}(\ln p_s), \frac{\delta}{\delta x}(\ln p_s)$.

        DPHIDX = -TWDSI*PHI(K,I+2,J) + THDSI*PHI(K,I+1,J)
     x                               -THDSI*PHI(K,I-1,J) + TWDSI*PHI(K,I-2,J)
        DPHIDY = -TWDSI*PHI(K,I,J+2) + THDSI*PHI(K,I,J+1)
     x                               -THDSI*PHI(K,I,J-1) + TWDSI*PHI(K,I,J-2)
        DLPSDX = -TWDSI*DUM2(1,I+2,J) + THDSI*DUM2(1,I+1,J)
     x                                -THDSI*DUM2(1,I-1,J) + TWDSI*DUM2(1,I-2,J)
        DLPSDY = -TWDSI*DUM2(1,I,J+2) + THDSI*DUM2(1,I,J+1)
     x                                -THDSI*DUM2(1,I,J-1) + TWDSI*DUM2(1,I,J-2)

C  RT == RGAS*TP
        DPTDX = DPHIDX + RGAS*TP(K,I,J)*DLPSDX
        DPTDY = DPHIDY + RGAS*TP(K,I,J)*DLPSDY
C  Calculate the right hand side of the momentum equations.
        RU = U(K,I,J) + DTA*( -UADV + UCORMF - EM(I,J)*DPTDX )
        RV = V(K,I,J) + DTA*( -VADV + VCORMF - EM(I,J)*DPTDY )
C  Update U and V.
        FACT = CORCNT*DTA*( F(I,J) +DUM1(K,I,J) )
        UP(K,I,J) = ( RU + FACT*RV )/( 1 + FACT**2 )
        VP(K,I,J) = ( RV - FACT*RU )/( 1 + FACT**2 )

   620 CONTINUE
```

The modified version of the corresponding .E code generated by EPF is:

```
        PARALLEL
        INTEGER  I,J,k
        REAL   UADV,VADV,UCORMF,VCORMF,DPHIDX,DPHIDY,DLPSDX,DLPSDY
      x        DPTDX,DPTDY,RU,RV,FACT

         PRIVATE   I,J,K,FACT,UADV,VADV,UCORMF,VCORMF,DPHIDX,DPHIDY,DLPSDX,
      x            DLPSDY,DPTDX,DPTDY,RU,RV

        DOALL(K=1:KMAX)
      DO 620 I = I2+NPNS,ILM-NPNS
      DO 620 J = J2+NPNS,JLM-NPNS

C Vertical advection.
      UADV = ( WADU(K+1,I,J) - WADU(K,I,J) - U(K,I,J)*(W(K+1,I,J)-W(K,I,J)) )*DQI(K)
      VADV = ( WADV(K+1,I,J) - WADV(K,I,J) - V(K,I,J)*(W(K+1,I,J)-W(K,I,J)) )*DQI(K)

C Coriolis and map-factor term.
      UCORMF = (1.0-CORCNT)*( F(I,J)+DUM1(K,I,J) )*V(K,I,J)
      VCORMF =-(1.0-CORCNT)*( F(I,J)+DUM1(K,I,J) )*U(K,I,J)

C Pressure gradient terms.
      DPHIDX = -TWDSI*PHI(K,I+2,J) + THDSI*PHI(K,I+1,J)
     x                 -THDSI*PHI(K,I-1,J) + TWDSI*PHI(K,I-2,J)
      DPHIDY = -TWDSI*PHI(K,I,J+2) + THDSI*PHI(K,I,J+1)
     x                 -THDSI*PHI(K,I,J-1) + TWDSI*PHI(K,I,J-2)
      DLPSDX = -TWDSI*DUM2(1,I+2,J) + THDSI*DUM2(1,I+1,J)
     x                 -THDSI*DUM2(1,I-1,J) + TWDSI*DUM2(1,I-2,J)
      DLPSDY = -TWDSI*DUM2(1,I,J+2) + THDSI*DUM2(1,I,J+1)
     x                 -THDSI*DUM2(1,I,J-1) + TWDSI*DUM2(1,I,J-2)
      DPTDX = DPHIDX + RGAS*TP(K,I,J)*DLPSDX
      DPTDY = DPHIDY + RGAS*TP(K,I,J)*DLPSDY

C Calculate the right hand side of the momentum equations.
      RU = U(K,I,J) + DTA*( -UADV + UCORMF - EM(I,J)*DPTDX )
      RV = V(K,I,J) + DTA*( -VADV + VCORMF - EM(I,J)*DPTDY )

C Update U and V.
      FACT = CORCNT*DTA*( F(I,J) +DUM1(K,I,J) )
      UP(K,I,J) = ( RU + FACT*RV )/( 1 + FACT**2 )
      VP(K,I,J) = ( RV - FACT*RU )/( 1 + FACT**2 )

  620 CONTINUE

       END DOALL
       END PARALLEL
```

In the EPF version, the parallel environment is bounded between PARALLEL and END PARALLEL statements. The parallelised section, in this case, is the loop driven by DOALL(...) and closed by END DOALL statements. Loop slicing is performed along the range of K (σ-coordinate), the number of loop slices being dependent on the number of processors available.

Execution Performance

A four XPC-processor Encore Multimax running under UMAX 4.3 has been used to investigate the execution performance of the parallelised model in EPF. The various runtime results as tabulated in Table 1 are for the model size of 65 by 40 grid (approximately 150km between grid points) by 12 σ-levels. On a single processor, the execution time for the EPF version is only fractionally longer than the original unparallelised version (both fully optimised). This shows that the overall overhead contributed by parallelisation in EPF is negligible. When multiple processors are used simultaneously to share the work load, Figure 2 shows that the execution time curve is close to the ideal.

#Processors	Time (seconds) Original FORTRAN	Time (seconds) EPF FORTRAN	Time (seconds) EPF Ideal	Speedup S	Speedup Sco
1	57.7	58.6	58.6	1.00	0.98
2	-	30.5	29.3	1.92	1.89
3	-	20.9	19.5	2.80	2.76
4	-	16.2	14.6	3.62	3.56

Table 1: Execution performance of the model in a Multimax multiprocessor environment

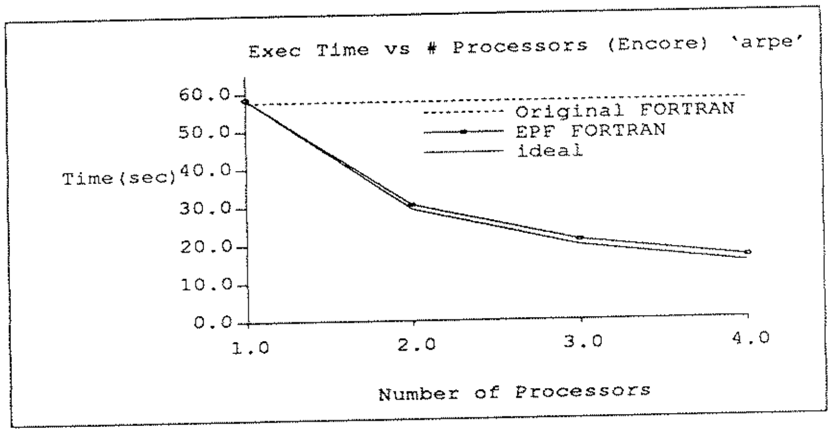

Figure 2: Reduced execution time in a multiprocessor environment

`Sco` is defined as the speedup of the EPF version over the standard FORTRAN version of the model while `S` is the speedup of the EPF version over the single processor run time of the same version. In the speedup plots in Figure 3, the `S` curve indicates that EPF gives very good parallel processing support to the implementation and the `Sco` curve shows that the speedup gained in runtime resulting from the parallel implementation is also as desirable. The two curves are almost coincident, indicating an efficient and encouraging result with 91% system utilisation in a four processor environment.

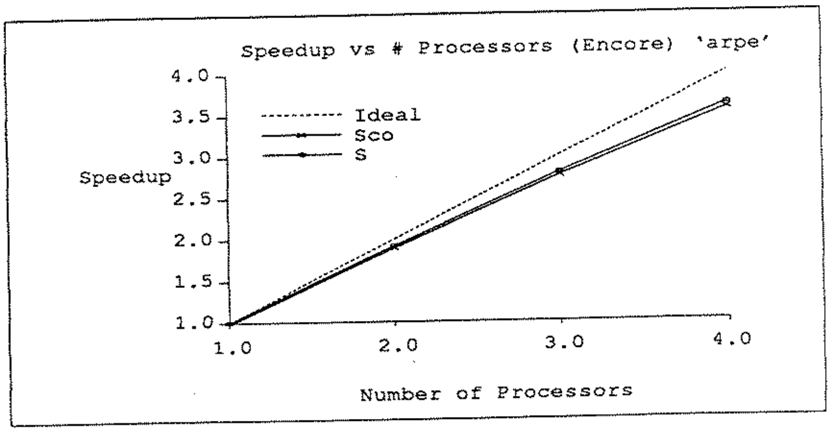

Figure 3: Speedup curves

Future Work

Work is underway at the Bureau to add the physical components into the model. Their impact to the model's parallel performance will be investigated once the full version is available.

Our research in a parallel implementation of a spectral barotropic NWP model[6] resulted in various recommendations of performance critical changes[7] to the functional parallel programming language SISAL[8] and the Optimising SISAL Compiler OSC[9]. These recommendations are being implemented by the SISAL committee and the developer of OSC for the new version of SISAL namely SISAL 2.0. When the new compiler is available, the Australian Region NWP model will be used to investigate the performance of SISAL 2.0 and the new compiler.

Conclusions

The encouraging results obtained with the Australian Region NWP model in EPF, a UMAX f77 implementation of parallel FORTRAN, have shown that the model has been successfully parallelised. The results show that EPF introduces little or negligible parallelisation overhead.

Acknowledgements

We are indebted to Dr. Lance Leslie and Dr. Gary Dietachmayer of the Australian Bureau of Meteorology Research Centre for their invaluable support to this project. They have built and supplied the original FORTRAN version of the model, and have been helpful in explaining its mathematical aspects and functions. 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. Leslie L.M. and Dietachmayer G., personal communication, September 1990.

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

3. Encore Computer Corporation, "Encore Parallel Fortran Manual", December 1988.

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

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

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

7. Chang P.S., "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.

8. McGraw J. 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.

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