SUPER COMPUTER
Contributions

Global spectral medium range weather forecasting model on PARAM

S.C. Purohit
T.V. Singh
P.S. Narayanan
Akshara Kaginalkar

Centre for Development of Advanced Computing, Pune University Campus, Ganesh Khind Road Pune 411007, India

Supercomputer **65**, XII-3

This paper describes the experience gained while parallelizing the global medium range weather forecasting application called T80 on a PARAM machine based on a distributed memory parallel architecture. The T80 code employs the spectral method for horizontal directions and finite differencing in the vertical direction and time marching. The parallel implementation takes care of the easy portability of parallel code across various platforms and environments. The parallel code is optimized using i860 based assembly routines. The validation of the parallel code has been accomplished by comparing the parallel T80 results with those of the Cray for 2nd February 1993 initial data.

1 Introduction

Numerical weather prediction is one of the important disciplines which requires a large amount of computer resources. There have been ongoing efforts to adapt existing meterological models to the existing range of supercomputers to get Tflop/s of performance so that predictions using much finer resolution than that currently used can be done. There are World-wide efforts to port weather forecasting codes on parallel computers. These include parallelisation of the ECMWF (European Centre for Medium Range Weather Forecasting) production code IFS (Integrated forecasting System) by GMD (German National Research Centre for Computer Science) and ECMWF and CHAMMP (Computer Hardware, Advanced Mathematics and Model Physics) program sponsored by the DOE (US Department of Energy), on various supercomputers like the Cray T3D, Intel Paragon, IBM SP2, Meiko etc. In order to become a self-reliant weather forecasting base in supercomputing, the Department of Science and Technology of Government of India decided of providing a cost-effective high-performance computing solution to the state-of-the-art models. A feasibility study of porting and developing a few operational/research codes on scalable parallel computers to compare with the performance of sequential computer was done. Thus a project to port the medium-range weather forecasting code T80 for such a purpose was initiated between National Centre for Medium Range Weather Forecasting (NCMRWF), Indian Institute of Technology (IIT), Delhi and Centre for

Pune, India. This required the T80 code to be ported and parallelized on a distributed memory scalable parallel computer PARAM8600 developed by C-DAC. The main aspect of this project is to find parallel techniques for the NCMRWF T80 code on a distributed memory parallel machine and to analyse and optimise the existing Cray code on PARAM. The T80 code is an unsteady 3-D simulation of the atmosphere around the earth which is divided into 128 latitudes, 256 longitudes, and 18 levels in a vertical direction above the surface of the earth. The code employs the pseudo spectral method in horizontal directions. In our implementaion the approach has been to understand the forecast model and the spectral algorithm used in the T80 code and specifically tune the code for PARAM8600, an Intel 860 based distributed memory parallel machine. The independent computation for FFT, Legendre transform and grid calculations are done across the latitudes distributed over the processors. In this paper we discuss the parallel strategy and issues regarding compute to communication ratio, load balancing and portability. In section 2, we present the spectral algorithm, the governing equations and time integration scheme, in section 3 we describe the sequential code structure and in section 4 the PARAM8600-machine hardware and software is described, followed by the parallel strategy in section 5. Validation of the results is done in section 6 and optimisation issues are discussed in section 7. The performance of the T80 code on PARAM 8600 is presented in section 8.

Governing equations and numerical algorithm

Governing equations for global weather model can be derived from the well known conservation laws of mass, momentum, and energy. The derivation of these equations are available elsewhere in the literature [1]. These mainly include vorticity, divergence, temperature, surface pressure and moisture equations. The derivatives in the vertical direction in these equations are approximated by using finite difference operators and a semi-implicit time integration scheme is applied to the coupled equations of divergence, temperature and surface pressure. This is done by applying a central difference scheme to the time derivative. Furthermore, the T80 model employs the spectral transform in horizontal direction [2,3] in which the crux of the algorithm lies in obtaining a spectral representation for all the terms. If it is a linear quantity, then the representations are straight forward. If it is a non-linear term, spectral transform methods have to be used. The computations for spectral algorithms are done in three discrete functional spaces: the grid point domain, Fourier domain and spectral domain. The time integration and the vertical integration are taking place in the spectral domain. Each time step of algorithm consists of the following steps

- Step Input spectral coefficients $u_n^m(Z_k)$ for all m, n and k.
- Step 2: Compute Fourier coefficients using inverse Legendre's transform

$$u^m(\mu_j, Z_k) = \sum_{n=|m|}^{M} u_n^m(Z_k) P_n^m(\mu_j)$$

for all m, j and k. P_n^m is the Legendre polynomial of degree n and order m.

Step 3: Compute Gaussian grid point values $u(\lambda_l, \mu_j, Z_k)$ using the inverse Fourier transform

$$u(\lambda_l, \mu_j, Z_k) = \sum_{m=-M}^{M} u(\mu_j, Z_k) e^{im\lambda_l}$$

for all l, j and k.

- Step 4: Compute non-linear terms and physics in grid point domain on Gaussian grid Update $u(\lambda_l, \mu_j, Z_k)$ for all i, j and k.

- Step 5: Compute Fourier coefficients $u^m(\mu_j, Z_k)$ using the direct Fourier transform

$$u^m(\mu_j, Z_k) = \frac{}{K_2} \sum_{l=0}^{K_2-1} u(\lambda_l, \mu_j, Z_k) e^{-im\lambda_l}$$

- Step 6: Compute spectral coefficients by direct Legendre's transform.

$$u_n^m(Z_k) = \sum_{j=1}^{K_1} \omega(\mu_j) u^m(\mu_j, Z_k) P_n^m(\mu_j)$$

for all m, n and k, where $\omega(\mu_j)$ are the Gaussian weights.

- Step 7: Perform calculations in spectral domain.

where

M	- truncation number (80)
K_0	- number of vertical levels (18)
K_1	- number of latitudes (128)
K_2	- number of longitudes (256)
$0 \leq m, n \leq M$	- spectral indices and their ranges
μ	- Gaussian latitude
$1 \leq j \leq K_1$	- latitude index and its range
λ	- longitude
$0 \leq l \leq K_2 -$	- longitude index and its range
Z	- vertical level
$1 \leq k \leq K_0$	- level index and its range

3 The sequential-code structure

The T80 code which is to be parallelised is an operational code running on a Cray machine; it employs triangular truncation and a spectral resolution of 80 waves. This code has 30,000 lines of FORTRAN code comprising 200 subroutines. The orginal T80 code supplied by NCMRWF has many Cray-dependent routines and its FFT part has been written in Cray assembly language. We have replaced this FFT by Temperton's [4] algorithm. Other Cray-specific routines were also replaced. In the FORTRAN source the global weather model, numerical algorithm and physics are integrated. Important components of the T80 code are the I/O, the forecast loop (GLOOPA), the radiation loop (GLOOPR), and the physics loop (GLOOPB), FFT and Legendre transform. The input and output data is in the form of sigma and surface data. The quantity called sigma is nondimensional parameter defined as "pressure at the point/surface pressure". The radiation routine is called once for every 24 hours simulation, whereas GLOOPA and GLOOPB are called in each iteration i.e. 96 times for a 24 hours forecast computation.

4 PARAM8600 – machine architecture and the environment

The PARAM 8600 is a back-end machine that can be connected to a PC or a Sparc Workstation and can be accessed from the front-end machine through a host interface card. PARAM can be connected to a maximum of 8 users through such cards.

The PARAM 8600 is based on RISC microprocessor Intel i860 XR operating at 33 MHz with a 4K data cache and a 8K instruction cache. The 64-bit calculations can be performed in double-precision mode on a i860 processor. In addition to the compute processor i860, PARAM 8600 has another microprocessor called the Transputer which operates at 20 MHz and are used for communication. Each Transputer has four 20 Mbit/s serial links. There is a 54 Mbyte/s high-speed bus called DRE (Data Restructuring Engine) between a Transputer and a i860 node. The PARAM 8600 node has one i860 processor with 32 MB memory and four transputers with 4 Mbyte memory each. Thus for any i860 node the bandwidth to the external world is 320 Mbits/sec. In all, the target machine for porting T80 has got 16 node cards comprising of 64 Transputers and 16 i860s.

The programming environment for PARAM 8600 is PARAS-MPV (Main Processor View). The PARAS microkernel is a message based operating system. This kernel is replicated on each node of the system and supports on each multiple tasks with a paged virtual space and message based interprocessor communication. Multiple jobs can run on PARAM by sharing nodes and multiple processes can be placed on the same node. The UNIX cross dependent tool for the i860s are used to develop an executable image of the user application. The application program is linked with the system library calls. The compiler used on PARAM8600 is PGI (Portland Group Inc.) compiler.

5 Parallelization strategy

In deciding the parallelization strategy the crucial part is to identify the nature of the time-consuming parts in the sequential code and then the parallel overheads due to communication and extra computation required in parallelizing. Our strategy is based on data decomposition and consequently on data dependencies. There are many independent variables namely latitude,

longitude, vertical levels and spectral indices m and n which can be considered for data decomposition. Based on the dependencies of the computations with these variables, we decided for a data decomposition along the latitudes. Northern and southern latitudes are paired and placed on one processor due to the symmetry of the spectral transform.

In this strategy, data required by each FFT is available within the processor. Thus we avoid communication at this stage. We have several FFTs simultaneously running on different processors and a decomposition of the data for the parallel Legendre transform. As a first step, an equal number of latitudes are allocated to each processor and accordingly data is distributed among them. Then each processor computes Fourier coefficients using the inverse Legendre transform and the Gaussian grid point values using the inverse Fourier transform. Non-linear terms and physics in grid point domain calculation on the Gaussian grid are done simultaneously on each processor. Then each processor computes Fourier coefficients only for those latitudes which are assigned to it. During the Legendre transform for the spectral coefficients calculations, each processor finds the partial sum for the given bands. These partial sums are then circulated among all the processes to have global sum on each. This forms the communication structure for latitude-wise parallelisation. The parallel implementation of the above strategy has a master-worker programming model. The inital input is distributed among several workers for the time integration scheme. Each worker works independently until the global sum calculation during Legendre transform. Here all the workers communicate data to all. At the end all the data is collected by the master for zonal diagnostics and final output. All the communication calls are in communication library currently in a software layer on top of PARAS, but can also be replaced by public domain communication primitives like PVM/MPI.

A feature of the code is the reduced main memory requirement per node. This has been made possible due to the cutting of the physical arrays as shown in parallel version of GLOOPA. For T80's sequential version it requires 60 Mbyte while same parallel version of the code requires 23 Mbyte per node for 8 nodes. The I/O of parallel T80 and various other control parameters remains same as in sequential. Full T80 code is in the Fortran language and no assembly coding has been used.

Validation of the Results

The partial double-precision results of sequential T80 code on PARAM approximate the full double-precision results of T80 results on Cray. This verifies the correctness of modifications applied to the Cray code such as additional source modules written, reorganisation of files and parallelisation. For further verification, the sequential T80 code has been made system independent and has been run on different platforms. In order to verify parallel T80 forecasts we have compared our forecasts with Cray forecasts for both a one-day and five-days run. For the one-day run the difference between the parallel T80 forecast and the Cray forecast remain withina 5% limit. While for five days forecasts, there are differences which increase with the number of days. However, the results do not diverge for any variable and remain bounded within a few percent of acceptable accuracy. It is important here to mention that the one day sequential and parallel code on PARAM produces almost same results except for very minor differences of precision order. Therefore the reason for difference between sequential Cray and sequential PARAM can be attributed to different floating-point standards used in these machines.

Optimisation

Before optimising the code, we first studied the most time-comsuming portion in the latitude loop, i.e. Legendre, inverse Legendre transform and FFT. In the typical subroutine `FL222K_ori` shows a Legendre transform code for the `QLN` with output in array `FLN` using `FP` and `FM` arrays as coefficients. If one observes this routine carefully then one can see that there is no need of the extra array `S(LNT2)`. Also, because of loop unrolling `FM` and `FP` arrays were being fetched again and again into the limited cash size of i860 processor. In order to avoid this we first made this code simple by removing intermediate array `S` and putting output in `FLN` array directly. Then it was clear that the `FM` and `FP` arrays were being used for computation again and again and should be available in the cache during the computation of Legendre transform. In order to put these array in cache, which is 4Kb in case of i860 processor, we created an array for vector registers vreg(164,6). For this purpose we used streamin function provided by the PGI compiler. Also we replaced the core time-consuming portion of the loop by some i860 assembly functions supplied by PGI like zxpy and streamin (see code `FL222K_mod`). These routines are easily integrated in the Fortran code without changing the structure of the loop. A considerable time improvement was possible due to this exercise.

```
      SUBROUTINE FL222K_ori (FP,FM,FLN,QLN,N)
      INTEGER
                TWOJ1

       PARAMETER(
                 JCAP=80,
                 JCAP1=JCAP+1,
                 JCAP2=JCAP+2,
                 TWOJ1=2*JCAP1,
                 LNT=JCAP2*JCAP1/2
                 LNT2=2*LNT)

      SAVE
C-----------------------------
      DIMENSION FP(TWOJ1,N), FM(TWOJ1,N)
     1 QLN(LNT2), FLN(LNT2,N)
      DIMENSION       S(LNT2)
CC
      NPAIR = (JCAP1-3)/2
CMIC$ DO ALL PRIVATE(S) AUTOSCOPE
      DO 2 K=1,N
      DO 220 I=1,TWOJ1
      S(I) = FP(I,K) * QLN(I)
  220 CONTINUE
      LEN = TWOJ1 - 2
      DO 230 I=1,LEN
      S(I+TWOJ1) = FM(I,K) * QLN(I+TWOJ1)
  230 CONTINUE
      IPLUS = TWOJ1*2 - 2
```

```
      LEN    = TWOJ1 - 4

      DO 260 J=1,NPAIR
      DO 240 I=1,LEN
      S(I+IPLUS) = FP(I,K) * QLN(I+IPLUS)
  240 CONTINUE
      IPLUS = IPLUS + LEN
      LEN = LEN - 2

      DO 250 I=1,LEN
      S(I+IPLUS) = FM(I,K) * QLN(I+IPLUS)
  250 CONTINUE
      IPLUS = IPLUS + LEN
      LEN = LEN - 2
  260 CONTINUE

      DO 270 I=1,LEN
      S(I+IPLUS) = FP(I,K) * QLN(I+IPLUS)
  270 CONTINUE

      DO 280 I=1,LNT2
      FLN(I,K) = FLN(I,K) + S(I
  280 CONTINUE
    2 CONTINUE
      RETURN
      END
```

The routine after optimisation

```
      SUBROUTINE FL222K_mod(FP,FM,FLN,QLN,N)
      INTEGER
                TWOJ1

       PARAMETER(
                 JCAP=80,
                 JCAP1=JCAP+1,
                 JCAP2=JCAP+2,
                 TWOJ1=2*JCAP1,
                 LNT=JCAP2*JCAP1/2,
                 LNT2=2*LNT)

      SAVE
```

```
      DIMENSION FP(TWOJ1,N), FM(TWOJ1,N)
     1 QLN(LNT2), FLN(LNT2,N)
        common /optimise/vreg(164,6)
```

```
      NPAIR = (JCAP1-3)/2
      DO 2 K=1,N
CC
         iplus = 0
         len = twoj1
         maxlen = len
        call __streamin4(fp(1,k),vreg(1,1),%val(maxlen),%val(1))
        call __streamin4(fm(1,k),vreg(1,2),%val(maxlen),%val(1))

      DO 260 J=1,(NPAIR+1

        call  .streamin4(fln(1+iplus,k),vreg(1,3),%val(len),%val(1))
        call  .streamin4(qln(1+iplus),vreg(1,4),%val(len),%val(1))
        call  .zxpy4(fln(1+iplus,k),vreg(1,3),vreg(1,1),vreg(1,4),%va

      IPLUS = IPLUS + LEN
      LEN = LEN - 2
CC
        call  .streamin4(fln(1+iplus,k),vreg(1,3),%val(len),%val(1))
        call  .streamin4(qln(1+iplus),vreg(1,4),%val(len),%val(1))
        call  .zxpy4(fln(1+iplus,k),vreg(1,3),vreg(1,2),vreg(1,4),%va

      IPLUS = IPLUS + LEN
      LEN = LEN - 2
  260 CONTINUE
CC
      DO 270 I=1,LEN
      fln(I+IPLUS,k) = fln(i+iplus,k)+FP(I,K) * QLN(I+IPLUS
  270 CONTINUE
CC
    2 CONTINUE
      RETURN
      END
```

This exercise was first checked on a small module of T80 code and then integrated in the full code. Validity of the results was checked at each stage. This way all the time-consuming Legendre and inverse Legendre transform routines were inserted in GLOOPA, GLOOPB, and in GLOOPR and then integrated with the main T80 code. The performance was improved by 40 % with this modification.

Performance

The sequential performance depends on the architecture of the processor and the parallel performance depends on the architecture of the system. The parallel T80 code that has been developed is portable and is user friendly in the sense that the general structure of original T80 is not changed, making it easy for atmospheric scientists to enhance and modify the code. Table 1 gives

the performance of parallel T80 on different numbers of nodes.

Nodes	Total time (Sec.)	Compute time	communication time	Remarks
1	26160	26160		Without optimisation
1	16800	16800		With PGI compiler and use of cache for Legendre Transform
1+4	6275	5054	1079	-
1+8	4371	3217	1010	-
1+13	3703	2314	1246	-
1+15	3694	2152	1396	-

Table Performance of the portable T80 code on different number of node.

9 Conclusion

In this paper we have presented the preliminary work done towards parallelisation of the spectral method based weather forecast code on an MIMD type distributed memory supercomputer. The partial double-precision results of parallel T80 code on PARAM scientifically approximates the full double-precision results of T80 on Cray. The parallel T80 is portable and can be made available on different platforms. It is scalable over the number of processors and over different size models. When we were benchmarking the Cray code on the PARAM, we found that many critical modules in T80 have been tuned for the Cray. If we want to perform FFT on each 64 independent data sets, then Cray performs a vector operation on all data sets and a major portion of this is written in Cray Assembly Language. We had to exploited the proper features of the i860 for the computation-intensive parts and to get better performance and by using fast global communication routines, communication overheads can be reduced to get better performance. Further work is going to port this parallel T80 code on different platforms using public domain communication primitives like PVM/MPI.

Acknowledgements

The authors wish to thank to Dr. P. Rama Rao, Secretary, Department of Science and Technology (DST) and Dr. S.K. Mishra, Director, National Centre for Medium Range Weather Forecasting (NCMRWF) for timely suggestions and help while this project was successfully completed at C-DAC.

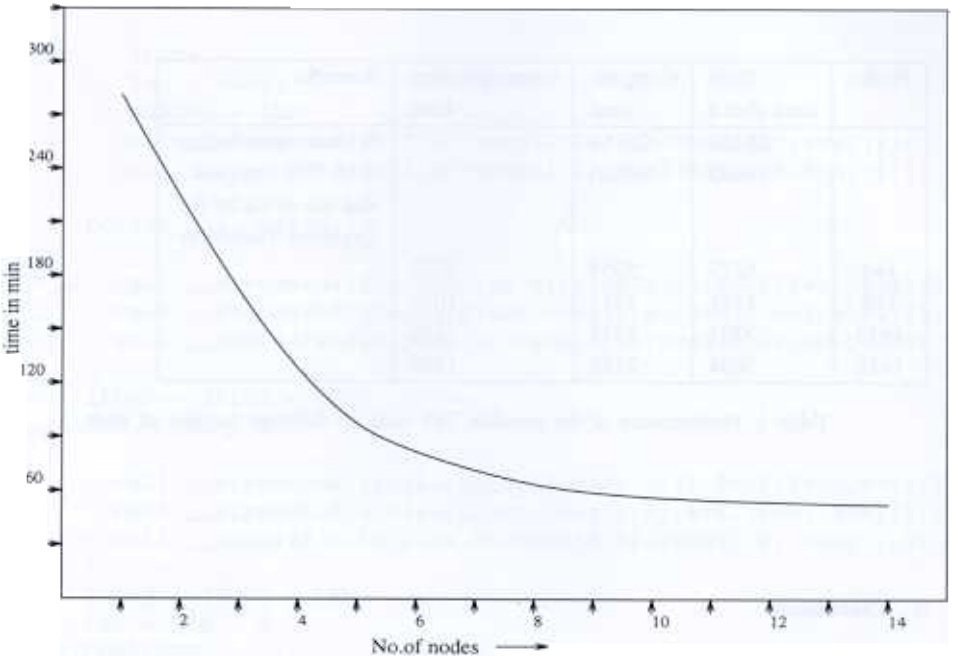

Figure Performance of the T80 on the PARAM 8600.

References

[1] Research version of Medium range forecast model (1988), Documentation - Volume 1; Hydrodynamics, Physical parametrization and user's guide.

[2] Bourke, W., et al., *Global Modelling of Atmospheric Flow by Spectral Methods in Computational Physics, Vol.7: General circulation models of Atmosphere*, ed J. Chang, Academic Press, pp.267-324.

[3] Purohit, S.C., P.S. Narayanan, T.V. Singh and Akshara Kaginalkar, *Development and Implementation of Parallel Numerical Weather Prediction Algorithm on PARAM – Progress report*, 1994.

[4] Temperton, C., *Self-Sorting mixed radix Fast Fourier Transform*, Journal of Computational Physics **52**, 1-23, 1983.