Strategies for Parallel Implementation of a Global Spectral Atmospheric General Circulation Model

Ravi S Nanjundiah
Centre for Atmospheric and Oceanic Sciences,
Indian Institute of Science,
Bangalore, 560012 INDIA
email: ravi@caos.iisc.ernet.in

Abstract

In this paper we discuss the parallel implementation of a global spectral atmospheric general circulation model on a message passing platform. We also discuss strategies that needs to be employed to improve performance on parallel machines which will have multiprocessor nodes sharing an intra-node memory space. A brief discussion of the cause of load imbalances and simple methods to reduce the same are also presented.

1 Introduction

Application of parallel computing for atmospheric studies began at the Centre for Atmospheric Sciences in the late 80s (Nanjundiah, 1988, Sinha and Nanjundiah, 1996). A hierarchy of models used in weather forecasting and also in climate studies have been successfully implemented on a variety of parallel machines. Parallel climate models presently being used at the centre have helped in the study of varied phenomena such as the intraseasonal variations of the Indian summer monsoons ([9], [10]) to the study of climate change in the prehistoric periods [4]. Elsewhere many global General Circulation Models (GCMs) have been implemented on parallel computing platforms using both SMPs and MPPs ([1], [11] [5], [12]).

Models used in weather forecasting and climate modelling are computationally intensive. Atmospheric models used for weather forecasting use very high resolutions to resolve (to the extent possible) weather events of interest such as typhoons, fronts etc. Weather forecasting models can be either regional models (i.e whose domain is limited to the region of interest) with very high resolution (with multiple nests the resolution could be as high as 5 km) or global models (i.e. models which solve for atmospheric variables over the entire globe) with relatively coarser resolution (typically about 50 km or a spectral truncation of 210). Most regional models (with a few exceptions) are finite difference models while most global forecast models (again with a few exceptions the most notable of them being the UK Meteorological Office Unified Model, which uses the finite difference technique). Lately a third group of models have emerged viz. the Semi-Lagrangian models. These use the concept of 'movement of parcels' to compute the advection terms. Semi-Lagrangian models may also use spectral transformations and conduct temporal marching in the spectral space.

1.1 Load Decomposition for Atmospheric Models

Most parallel implementations (both SMP and MPP implementations, spectral or finite difference models) use the method of domain decomposition in one form or the other. With reference to use of parallel computing techniques, the major difference between finite difference grid point models and spectral models is that while the former generally require *nearest neighbour communication* (the actual number of gridpoints requiring overlap and communication depending on the order of the finite difference scheme), *spectral models* require a global communication stencil, as computation of spectral co-efficient requires data from every grid point (at least at the same horizontal level) in the physical domain. Thus a simplistic view of the pattern of communication in various categories of models

would be:

1. Spectral models with a global stencil of communication would be inherently suited to shared memory systems as the same data could be utilized by all processors to compute the spectral co-efficients

2. Finite difference models may be better suited to message passing systems as the communication stencil is generally more localized and restricted to nearest neighbours.

3. Semi-Lagrangian (SLT) models may not have a static communication stencil, as the data required by a processor could depend on the dynamic variables such as the horizontal velocities. Higher the velocities, the larger distance would a parcel travel within a timestep, leading to communication requirements from larger distances (possibly beyond the nearest neighbouring processor). The experience with SLT models has been limited. The CCM2/CCM3 (and their scalable parallel implementation [1] , the major difference between CCM2 and CCM3 is in the physics and not the numerics and hence the parallel implementation remains unaffected) use SLT for moisture transport. The experience in their scalable parallel implementation has been that near the poles especially near the pole experiencing winter, the number of grid points which a parcel may traverse could be substantial, sometimes beyond the nearest neighbour, if a two-dimensional decomposition in latitude and longitude is used. The cause of this large traverse is two-fold:

   (a) velocities near the poles are large, especially in the east-west or zonal direction.

   (b) the grid is uniform in angular space. This causes the physical distances especially near the poles to reduce vis-a-vis equator. This reduction occurs between grid point in the east-west direction only, grid distances do not reduce in the meridional direction.

   The combination of the above mentioned effects cause large changes in the communication stencil. However, the effect of dynamics on the communication stencil is asymmetric i.e. velocities in the north-south (meridional direction) are much smaller than in the east-west (zonal direction). Thus the traverse of parcels in the north-south direction is much smaller than in the east-west direction and therefore communication needs between processors in the meridional direction is essentially of the nearest neighbour-type. If one were considering a one-dimensional decomposition in latitude alone (i.e. a set of latitude circles assigned to a processor), the communication stencil essentially collapses to that of the finite difference scheme or the communication requirements could even be less with SLT (vis-a-vis finited difference models), if for the same resolution a higher order finite difference scheme is used.

With the advent of machines in which the node in a message passing cluster comprises of multiple processors sharing a common memory (i.e. within the node processors share a common memory space) it is necessary to look at alternate strategies to exploit the various levels of parallelism that could be available with this class of parallel machines. We present here some of the implementation strategies which are currently being implemented/considered for the National Centre for Medium Range Weather Forecasting Model (NCMRWF model).

The strategies considered here are for the moderately parallel range (4-32 processors) of machines. This is because in the foreseeable feature, this is the range of processors that would be available to us in the developing countries for conducting research in atmospheric sciences. Most parallel implementation conducted elsewhere have stressed on scalability with larger array of processors. Our experience has shown that obtaining a good performance with the NCMRWF model (even on the moderate range) is a non-trivial task. The details of the model given below makes the cause of this much clearer.

2 The NCMRWF Model

The present study uses a version of the NCMRWF model. This is a global model (i.e., its domain of integration is global in the horizontal direction) and in the vertical its domain extends from the surface (nominally at a pressure of 1000 hPa) to the top level at 21 hPa. The main prognostic variables of this model are vorticity, divergence, temperature and specific hu-

midity. Winds are computed from vorticity and divergence. Other prognostic variables are surface temperature, precipitation (both snow and rainfall) and surface wetness.

The model is spectral i.e, time integration is done in the wave number space. However, some of the computations for the model need to be done in the physical (latitude-longitude) space e.g.. computation of radiation, surface effects, clouding and precipitation, evaporation, etc (generally called as the physics of the model). The process of advection, horizontal diffusion and time incrementation is generally referred to as the 'models dynamics' and the computations for these are conducted in spectral space.

The tendencies of variables (such as the rate of change of temperature due to radiative processes etc) calculated in physical space need to be transformed to spectral space and variables in spectral space need to be transformed to physical space (for the purpose of conducting model physics computations). The model's initial state is in spectral space. These are done using Fourier and Legendre Transforms.

The model uses a semi-implicit scheme (with leap-frog technique) for temporal integration. It has a horizontal resolution of T80 (i.e. 80 waves are considered to represent a variable). In the latitudinal direction (i.e, north-south direction) it has 128 Gaussian latitudinal circles. On each of these circles it has 256 grid points in the zonal direction (east-west direction). A comprehensive documentation of the model's dynamics is given in [6]

We have conducted most of the simulations at the T-80 resolution. A validated simulation with the parallel version of the model at T-126 has been successfully conducted. The two versions of the models are different in more than resolution alone, there being differences in physics (such as computation of surface budgets, radiation etc). Due to these substantial differences, the two versions of the model do not share a common code.

Typically the computational flow of the model's integration is as follows:

1. Transform from spectral space to Fourier-latitude Space
2. Transform from Fourier-latitude space to physical grid space
3. Compute in physical space the non non-linear terms and the model physics
4. Convert to Fourier-latitude space and compute zonal derivatives
5. Transform to spectral space, compute meridional derivatives and march in time.
6. Transform from spectral to Fourier-latitude space
7. Transform from Fourier-latitude space to physical grid space
8. compute model physics
9. Transform to Fourier-latitude space
10. Transform to spectral space
11. Add the physical tendencies to variables and advance in time
12. repeat the above steps

Comparing this model with other models such as the CCM2 we note that there are major differences. The most significant being:

1. The NCMRWF model uses two transformation between physical and spectral spaces for a timestep while CCM uses a single transformation between the two spaces for a single timestep.
2. For computations in the physical space, the NCM-RWF model converts data from spectral space to grid space for a latitude, computes the required quantities and converts data to Fourier-latitude space before commencing computations for the next latitude circle. On the contrary, CCM, computes all physical quantities in the physical space after transformation to physical space for the entire grid and returns to spectral space only after the computation has been completed for the entire physical grid.

The two transformations and a memory window of a single latitude circle (instead of the entire grid space) reduce the memory requirements considerably. However, communication overheads increase enormously, as each transform requires additional communication. It is interesting to note that in spite of such major differences in the computational flow of the two models, the

numerical scheme used by them models is largely identical i.e a semi-implicit scheme in which some terms involving pressure gradients and divergence in the divergence, surface pressure and thermodynamic equations (called as the linear part of these equations) are calculated using a implicit technique while all other terms are computed explicitly.

The parallel version of the NCMRWF model has been developed on the 32 node IBM SP2 located at the Supercomputer Education and Research Centre (SERC) of the Indian Institute of Science, as part of a joint study project involving scientists from IISc, IBM and National Aerospace Laboratory, Bangalore. The parallel code is generic in nature and uses MPI constructs. The parallel model can be run with little or no changes on any parallel platform supporting MPI.

We now discuss some of the changes performed to implement the parallel implementation. Most of the modifications are for computing the Gaussian quadrature to complete the transformation from Fourier-latitude space to spectral space.

3 The Modifications to the Legendre Transform

We have used the method of the domain decomposition to distribute load between processors. The global domain was split into smaller sub-domains, i.e., slices of latitude-circles assigned to a processor, (in a conventional computer, the entire would be assigned to a single processor, here the domain is split and each resultant sub-domain is assigned to a processor). Symmetry about the equator for computing the spectral coefficients for pairs of latitude-circles is exploited and each such latitude pair is assigned to a processor (i.e. a processor computing for 30°N would also compute 30°S. This method of decomposition requires communication to complete the summation of spectral coefficients using the Gaussian quadrature ([12], [13]). The communications are global in nature i.e., calculation of spectral coefficients requires data from every point on the sphere. This stencil of communication greatly constrains the scalability (i e decrease in computational time with increasing number of processors). We have used the method of binary tree summation to partially reduce this bottleneck.

Alternative techniques of decomposing the computations are currently being examined. We present two of the techniques currently being considered:

1. Partial decomposition in spectral space for Gaussian Quadrature summation
2. Total decomposition in the spectral space.

3.1 Partial Decomposition in Spectral Space

This can be briefly described as follows:

1. Formation of clusters (or subsets) of processors. Each cluster typically contains four processors.
2. Local Gaussian Quadrature summation for the sub-domain of the processor.
3. Transpose of the partially summed spectral coefficients. This allows all the partial sums (from other sub-domains of the cluster) for the same spectral wave number to lie within a processor.
4. Summation of the partially summed spectral coefficients (over the domain of the entire cluster)
5. Swap the cluster-summed spectral co-efficients with processors which handle the same wave number from other clusters.
6. Sum to obtain globally summed spectral coefficients.
7. Swap within clusters to have the data for the entire spectral domain on every processor.

In comparison with a binary tree summation for the entire spectral domain, this method offers a reduction in communication size (data). The performance using this method is presently being studied. This method will be more advantageous with distributed shared memory (DSM) machines (with clusters of shared memory processors linked to other clusters through message passing links). The higher bandwidths available with shared memory architecture could be used for the transposing of the data, while smaller requirements across clusters could be handled by the message passing links. We consider this technique of implementation to be very useful in the long term, as DSMs are likely to be the parallel computers of the future.

3.2 Total Decomposition in Spectral Space

Using partial spectral decomposition techniques we can decompose computations only in the physical space and not in the spectral space (except for the Gaussian summation). By decomposing in the spectral space, we can exploit the parallelism inherent in the spectral space also.

Also in the above discussed techniques it is difficult to hide communications behind computations. Hiding communication involves starting up a communication much before the variables being communicated are used in computations. This involves using asynchronous and non-blocking communications (in contrast to blocking and synchronous communications used by MPI/MPL summation routines such MPI_ and MP_REDUCE). With a view to incorporate this feature into our parallel model, we re-examined the parallelizing technique and communication stencil.

We realized that conducting communications in the Fourier Latitude space instead of spectral space could effectively help us to hide communications behind computations. The model calculates Fourier coefficients for a latitude at a time. The co-efficients are really required only during the time of Gaussian Quadrature (which could be done after the co-efficients for all latitudes have been computed). Thus we could start up the communication for a latitude (for which the Fourier Computations have been completed) while the next latitude is being computed. Additionally, the size of the messages reduce with increasing number of processors (thus aiding in scalability subject to the communication links being fast enough). We should be able to hide communications to a very large extent by this method of decomposition and thus improve the scalability of the parallel implementation. Additionally, the computations in the spectral domain can be parallelized in a simple fashion and this should further help in improving the scalability (as the parallelizable part increases, the sequential computations reduce and thus according to Amdahl's law, the efficiency of parallel implementation should improve). Experiments are presently continuing with this method of decomposition and we hope that the scalability at larger processor configurations (such as 32 processors and 64 processors) will improve considerably. Its impact might not be noticeable at lower processor configurations.

4 Modifications to Fourier Transform

In the previous section we have discussed the strategies that are being implemented to improve performance through changes to Legendre Transform. The other major transform that is used is the Fourier transform. This transform is conducted for computing Fourier coefficients for variables in the grid-point space (on a given latitude circle). A Fourier transform for a given variable requires the data from all grid points (for that given variable) on that latitude circle. Thus a two-dimensional domain decomposition (in the latitude-longitude plane) requires communication for the completion of the Fourier transform. In [1] it is shown that a transpose- sequential Fourier Transform-transpose technique is more efficient than a distributed FFT. They have discussed in the context of message passing machines. All their experiments were conducted with uniprocessor nodes (i.e. one processor on a node). However with multi-processor nodes (sharing a common memory) the scenario could change. A typical decomposition could be

1. decomposition over longitudes over the processors on a single node
2. decomposition over latitudes over nodes.

Now on such machines, if all the data for a variable were to be available in the shared memory space of the processors on a single node, then even a transform might not be required, or at most it might be a process of 'copying' shared data from the shared space to the local memory of a processor and therefore conducting Fourier transform would be a much easier task than over conventional message passing machines. Following the Fourier transforms the coefficients could be moved to shared space or to local memory of the processors to complete the transpose and further computation of the Legendre Transform. A local Gaussian quadrature could be performed to sum over the set of latitudes assigned to the node and further communication between nodes completed for a global summation. However it has sometimes been suggested that the order of summation for Gaussian quadrature (i.e. sum from pole to equator) be maintained regardless of the number of processors used, to make the results reproducible and independent of the number of processors. Such a constraint is more required for climate simulations than

for weather forecasting applications [2]. This makes communication processes more stringent. For satisfying this condition, we could communicate the Fourier co-efficients for each latitude and then follow it up with Gaussian quadrature in the strict order of latitudes. For improving scalability, a partial set of co-efficients (but for all the latitudes) could be assembled on a node and the quadrature performed. This procedure is very similar to the 'total decomposition in spectral domain' discussed above for one dimensional domain decomposition.

5 Load Balancing Issues

It is difficult to completely balance computational loads between the nodes of a parallel machine. The load imbalances for a two-dimensional decomposition of the parallel CCM2 has been studied in [7]. They concluded that the major cause of imbalance was the diurnal variation of solar radiation in the longitudinal direction. The grid points in the daylight zone need to perform additional computations for shortwave radiation which introduces significant load imbalance. To offset this [3] suggested that daylight and nighttime points be alternately swapped (i.e. moved to 180 longitudinally) so that these points alternate and the load imbalance is evened out.

We conducted a load study on a one dimensional decomposition of the NCMRWF model and found that the major cause of imbalance (to the extent of 15%) was caused by the computation of cumulus convection. This imbalance has a continuous gradient with minimum load on the processor computing the polar regions and maximum load on the processors computing the equator and the tropics. The higher precipitation (and related computations in the cumulus convection scheme) in the tropics vis-a-vis the other regions was the cause of this imbalance. A load simple load balancing strategy was introduced to overcome this problem. Instead of assigning a continuous set of latitudes (or co-latitudes) to a processor, a staggered set was assigned. For example, in a four processor configuration, the first processor would be assigned latitudes 1, 5 ,9,... while the second processor would compute latitudes 2, 6, 10,..., and the third processor's domain would comprise of 3,7,11,..., and so on . Thus all processors had a more equitable distribution of both polar and tropical latitudes and the load imbalance due to cumulus convection was largely removed. A combination of both strategies i.e swapping of day and night points to overcome the effects of diurnal variation and staggered set of latitudes to overcome the imbalances due to cumulus convection is being considered for the two-dimensional parallel implementation of the NCMRWF model.

6 Conclusions

In this paper we have discussed various strategies that have been implemented to improve the performance of the NCMRWF model. We have proposed strategies for architectures in which the node of a parallel machine is no longer a single processor but a set of processors sharing a common memory. We also have discussed the cause of load imbalances in parallel implementations of two GCMs and simple techniques by which these imbalances can be removed. The combination of these decomposition and load balancing techniques will yield improved throughput on message passing platforms. Further, on parallel machines with multi-processor nodes, the strategies discussed above could be used to improve efficiency in spite of differing speeds of intra-node and inter-node communication.

Acknowledgments

The author thank SERC-IISc for providing computational support on the IBM-SP2. The parallel implementation of the NCMRWF model has been partially supported by IBM, USA, under the joint-study agreement #5385 and the same is gratefully acknowledged.

References

[1] Drake, J., Foster I., Michalakes, J., Toonen, B., and Worley P., 1995: Design and Performance of a Scalable Parallel Community Climate Model. *J of Parallel Computing,* **21,** 1571-1591.

[2] Burton, and Dickenson, 1995: Parallelising the Unified Model for the Cray T3E. *In Proceedings of Seventh ECMWF Workshop on the Use of Parallel Computers in Meteorology,* Dec 2-6, 1996, ECMWF, Reading, UK.

[3] Foster I., and Toonen, B., 1994: Load Balancing Algorithms for Climate Models. *In Proc. 1994 Scalable High Performance Computing Conference,* IEEE Computer Science Press.

[4] Jagadeesha, D., Nanjundiah, R.S., and Ramesh, R., 1998: Orbital Forcing of Monsoonal Climates in NCAR

CCM2 with two Horizontal Resolutions. *Submitted to Paleoclimatology - Data and Modelling.*

[5] Jones, P., Kerr, C., and Hemler R, 1995: Practical Considerations in Development of a Parallel SKYHI General Circulation Model *J of Parallel Computing,* **21,** 1677-1694.

[6] Kalnay E., Sela, J., Campana, K., Basu, B.K., Schwarzkopf, M., Long, P., Caplan, M., and Alpert, J., 1988: Documentation of the Research Version of the NMC Medium Range Forecast Model.

[7] Michalakes, J and Nanjundiah, R.S., 1994: Computational Load in Model Physics of the Parallel NCAR Community Climate Model. **ANL/MCS-TM-186**, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., USA.

[8] Nanjundiah 1988: Simulation of Monsoon on Flosolver, **PD AE 8813,** National Aerospace Laboratories, Bangalore, India.

[9] Nanjundiah, R. S., Srinivasan J., and Gadgil S., 1992: Intraseasonal Variation of the Indian Summer Monsoon. II : Theoretical Aspects. *J of Jap Met. Soc,* **70,** 529-550.

[10] Raju A. and Nanjundiah R.S, 1997: Development of the Climate Version of the NCMRWF Model, presented at *Tropmet 97 Conference,* February 2-8 1997, Indian Institute of Science, India.

[11] Sela, J , 1995: Weather Forecasting on Parallel Architectures, *J of Parallel Computing,* **21,** 1639-1654.

[12] Sinha U, and Nanjundiah R.S., 1996: A Decade of Parallel Meteorological Computing on the Flosolver, in *Proceedings of Seventh ECMWF Workshop on Use of Parallel Computers in Meteorology,* Dec 1-6 1996, ECMWF, Reading, U.K.

[13] Sinha U.N., Sarasamma, V., Rajalakshmy, S., Subramanium, K., Bhardwaj, P., Chandrashekhar, V, Venkatesh, T., Sunder, R., Basu, B.K., Gadgil S and Raju A., 1994: Monsoon Forecasting on Parallel Computers *Current Science,* **67,** 178-184.