 CMES, vol.27, no.1, pp.1-23, 2008

HPC: Its application in Climate Modelling

Ravi S Nanjundiah[1]

Abstract: In this paper, application of high performance computing to climate modelling with specific reference to global General Circulation Models (GCM) is discussed. Methods of parallelization of global atmospheric models based on their numerical schemes is presented. It is seen that there is an interesting co-evolution of computer architecture and the type of numerical schemes used in general circulation models. A detailed survey of the Indian HPC scenario for meteorological computing is presented. Innovative and pioneering aspects of Indian efforts are highlighted.

Keyword: Climate Modelling, High Performance Computing

1 Introduction

Study of weather and climate with numerical models has been one of the most computationally challenging problems. The diversity of scales from the macro ($\sim 10^6$ m) to the micro ($\sim 10^{-3}$ m) involved in this phenomena and processes such as radiation, eddies at different scales and interaction between various sub-components such as the land-atmosphere interaction, ocean-atmosphere interaction makes this a fascinating subject. Traditionally, largest supercomputers have always been used for climate studies and numerical weather prediction.

A model for the atmosphere contains equations for conservation of mass, momentum, energy and species conservation. In the modelling of weather and climate, we consider air (or atmosphere) to be a fluid and hence try to model the flow of the fluid. The rotation of the earth and its impact on the flow through Coriolis acceleration is one of the important phenomenon which makes this flow different from conventional fluid mechanics. The driving force for all atmospheric motion is solar energy. The incoming solar energy varies over different longitudes and latitudes on daily and seasonal basis. Thus this differential heating of various parts of the globe drives atmospheric flow. Additionally factors such as interaction between clouds and radiation, aerosols (dust, pollutants, sea-salt etc) and radiation, also need to be considered. Energy from the earth-atmosphere is system is lost in the form of outgoing infrared radiation. Its interactions with various absorbing gases and clouds need to be considered. Another additional source of energy is the phase change from water vapour to liquid. Condensation releases latent heat which is transferred to the surrounding atmosphere. This process plays an important role in amplification of cyclones. We can see that thermodynamics and dynamics are closely coupled e.g. condensation of water heats the atmosphere and changes the pressure and this in turn changes the circulation. It is the convergence of moisture at a particular place caused by circulation patterns that causes rain. Therefore study of climate leads to set of coupled non-linear partial differential equations. Additionally when coupling between ocean and atmosphere is considered in an interactive fashion, then a further degree of non-linearity is added. We then also need to solve a similar set of equations for the oceans. Other interactions include those between land-surface and atmosphere in which vegetation (or the lack of it) plays an important role. The computations at the land-atmosphere interface is usually conducted in a separate sub-model called the land-surface model.

In atmosphere modelling, processes can be broadly classified into two categories: 'model dy-

[1] Centre for Atmospheric & Oceanic Sciences, Indian Institute of Science, Bangalore, 560 012, India; email:ravi@caos.iisc.ernet.in

namics' and 'model physics'. The part of the atmospheric model categorized as 'model dynamics' deals with the inviscid part of the equations for momentum, energy, mass and species conservation (excluding the sources and sinks). In this part one can use numerical discretization and thus error-bounds due to discretization can be estimated quite well. This is also sometimes called as the 'dynamical core' of the model. Various dynamical cores depending on the method of discretization of equations exist viz. finite-difference dynamical core, Eulerian (spectral) dynamical core, Semi-Lagrangian (spectral) dynamical core and finite volume dynamical core. Most of the dependencies in the horizontal direction are related to the choice of dynamical core and can have significant impact on the scalability of the model during its implementation on parallel computers.

'Model Physics' in contrast is largely empirical and deals with various forcing terms such as the computation of radiation, sub-grid scale phenomena and diffusion. Sub-grid scale phenomena are processes at scales which cannot be resolved at the resolution of the model but still are of considerable importance e.g. cumulus clouds. These clouds occur on scales of 1-10 km (the typical model resolution of a climate system model being about 100 km) but heating due to phase change from water vapour to liquid water has a major impact on the circulation patterns, especially in the tropics. These computations tend to have only vertical dependencies and are more scalable than the dynamical core.

The equations solved in an atmospheric model are:

Law of conservation of mass

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \vec{V} \right) = 0 \tag{1}$$

Where ρ is density and $\vec{V}$ is the velocity

Law of Conservation of Momentum

$$\frac{D\vec{V}}{Dt} + 2\vec{\Omega} \times \vec{V} = -\frac{\nabla p}{\rho} - \nabla \Phi + \vec{F} \tag{2}$$

where p is the pressure, Φ the geopotential, $\vec{\Omega}$ the earth's angular velocity vector and $\vec{F}$ represents the frictional forces.

Law of Conservation of Energy

$$\frac{D}{Dt} \ln \Theta = \frac{H_T}{C_p T} \tag{3}$$

where $\Theta = T(p_0/p)^{\kappa}$ is the potential temperature, $\kappa = R/C_p$ (C_p is the specific heat at constant pressure and R is the gas constant for air) and T the temperature. H_T represents the diabatic heating term (due to radiation, surface heating and phase change). p_0 is a reference pressure usually taken to be 1000 hPa i.e. the nominal pressure at earth's surface.

The Equation of State

$$p = \rho R T \tag{4}$$

Species Conservation Equations

The equation for species conservation (such as moisture, other trace gases and aerosols) can be written as:

$$\frac{Dq}{Dt} = S \tag{5}$$

where q is the specific humidity or concentration of any other species and S represents sources and sinks. For moisture the source term is evaporation and the sink term is precipitation.

These are appropriately scale analyzed for the class of models being developed. Generally for climate models with relatively coarse resolution ($\sim$ 100 km) i.e. models which can only resolve coarser scales such as the planetary and synoptic scales, simplification to the vertical momentum equation is obtained by the hydrostatic approximation.

$$\frac{\partial p}{\partial z} = -\rho g \tag{6}$$

This also allows the use of pressure as a vertical co-ordinate. To allow for terrain variation a non-dimensional vertical co-ordinate based on pressure is used (generally called as the sigma-co-ordinate). Some models also use a hybrid system of co-ordinates which begins as non-dimensional pressure co-ordinates at the surface and slowly

changes to pressure co-ordinate at the upper levels. In some very high resolution models, height 'z' is used as the co-ordinate. The advantage of using the hydrostatic approximation is that acoustic waves are filtered out and thus larger time-steps can be used in forecasting. This results in increased computational efficiency. However, there is a school of thought that considers the use of hydrostatic approximation to be inaccurate in the deep tropics and advocates the use of 'quasi-hydrostatic' approximation in coarse resolution models (White and Bromley, 1995). New very high resolution GCMs (with horizontal resolution of $\approx$ 10 km) are now experimenting with non-hydrostatic formulations.

2 Numerical Discretization of Equations

These set of coupled non-linear partial differential equations need to be solved numerically. Various numerical methods are in vogue to solve these equations. The most common techniques are:

1. Finite Difference Technique
2. Eulerian Spectral Technique
3. Finite Volume Semi-Lagrangian Technique

Of these finite difference and Eulerian spectral technique have been traditionally used. Currently finite volume semi-Lagrangian techniques seem to be gaining popularity in view of their better scalability and superior advection properties especially for advecting trace gases and aerosols (Lin, 2004).

2.1 Finite Difference Technique

Finite difference technique has been widely used in atmospheric modelling. For higher accuracy and better geostrophic adjustment, staggered grids (Arakawa and Lamb, 1981)) have been used. The 'B' and 'C' grids are widely used. Later we will see that a combination of D and C grids are used in finite volume techniques. For temporal discretization, leap-frog scheme (centre in time) is usually preferred. Removal of computational modes is done by using a three-time step filter. A semi-implicit scheme is generally preferred. The implicit part is used for pressure, temperature and divergence components of the equations as these give rise to high speed gravity waves, while non-linear and Coriolis terms are dealt with explicitly. Thus the use of a semi-implicit method combines the simplicity of an explicit scheme with the increased time-step of an implicit scheme without seriously distorting the modes of interest viz. the slower meteorological modes.

2.2 Eulerian Spectral Technique

This is one of the widely used techniques in weather and climate modelling. The spectral method came into vogue in the 80s which interestingly coincided with the development of vector computers. Vector computers are more efficient on operating on long arrays (vectors) than on single elements (scalars) Given that spectral methods had long arrays to be computed, it is not surprising that they could exploit the architecture of the vector computers very efficiently. However their scalability on massively parallel computers (which are built up with scalar processors) is limited. Additionally Gibbs oscillation causes erroneous results in regions of steep gradients. This causes problems in computation of variables such as specific humidity. These factors have caused their appeal to reduce somewhat in the field of climate modelling. However, their high degree of numerical accuracy makes them very attractive for weather forecasting. The Eulerian spectral technique is still being used very extensively for weather forecasts at the National Centers for Environmental Prediction, USA (NCEP) and also by other meteorological agencies such as the Japanese Meteorological Agency (JMA) and the (Indian) National Centre for Medium Range Forecasting (NCMRWF).

In the Eulerian spectral technique, any variable (temperature, moisture and flow variables) is represented as a set of spectral co-efficients (which vary with time) along with a spherical harmonic basis (a combination of Fourier modes in the east-west direction and Legendre polynomials in the north-south direction). Since it is inefficient to compute the non-linear advection terms in the spectral space and impossible to compute

the model physics in this space, a transformation from grid to spectral space is used. Any variable, ζ, at latitude θ and longitude λ can be expressed as:

$$\zeta(\lambda,\mu) = \sum_{m=-m}^{M} \sum_{n=|m|}^{N(m)} \zeta_n^m P_n^m(\mu) e^{im\lambda} \tag{7}$$

where $\mu = \sin\theta$, where θ is the latitude, $P_n^m(\mu)$ is the normalized associated Legendre function. We can determine the spectral co-efficients by:

$$\begin{aligned} \zeta_n^m &= \int_{-1}^{1} \left[\frac{1}{2\pi} \int_0^{2\pi} \zeta(\lambda,\mu) e^{-im\lambda} d\lambda \right] P_n^m(\mu) d\mu \\ &\equiv \int_{-1}^{1} \zeta^m(\mu) P_n^m(\mu) d\mu \end{aligned} \tag{8}$$

where $\zeta^m(\mu)$ represents Fourier co-efficients over a latitude circle. The spherical harmonics $P_n^m(\mu)e^{im\lambda}$ are the eigensolution of the Laplacian on the sphere. Since in actual practice we cannot take an infinite series we truncate it to a wave number M in the zonal direction. In the meridional direction, $N(m)$ is the highest degree of the associated Legendre polynomial for this truncation.

From equation (8) we find that the transformation of $\zeta(\lambda,\mu)$ to ζ_n^m can be done in two steps:

1. A Fourier analysis step on every latitude circle to calculate the Fourier co-efficients $\zeta^m(\mu)$ (commonly known as the Fourier transform step).

2. An integration over all latitudes to obtain ζ_n^m (commonly known as the Legendre transform)

For conducting Fourier analysis efficient algorithms such as the Fast Fourier Transforms (FFT) are available. The integration over the latitudes (Legendre Transform) is replaced by a Gaussian quadrature.

$$\zeta_n^m = \sum_{j=1}^{J} \zeta^m(\mu_j) P_n^m(\mu_j) w_j \tag{9}$$

Here w_j are Gaussian weights which are determined by:

$$w_j = \frac{2(1-x_j^2)}{kP_{k-1}x_j^2}$$

Here x_j are latitude points (Gaussian points) which are zeros for the Legendre polynomial of order k. Use of this increases increases the order of accuracy of the method from k-1 to 2k-1. A detailed discussion of Gaussian quadrature is given in Durran (1999). The degree k and consequently the number of Gaussian latitudes is determined from constraints of alias-free computation of the quadratic (non-linear advection) terms of the equations. The actual value of the total number of Gaussian latitudes (usually denoted by J) depends on the type of truncation used and number of waves, M (truncation number). Commonly used truncations are rhomboidal and triangular truncation. If M is the truncation (i.e. the number of waves used in the zonal or east-west direction) then $J \geq \frac{3M+1}{2}$ for triangular truncation and $J \geq \frac{5M+1}{2}$ for rhomboidal truncation.

The inverse transform from spectral space to grid space is done in two steps:

1. An inverse Legendre transform to convert from spherical harmonic space to Fourier space

2. A Fourier transform to compute the grid point values from Fourier co-efficients.

The inverse Legendre transform is given by:

$$\zeta^m \mu = \sum_{n=m}^{N(m)} \zeta_n^m P_n^m(\mu)$$

The $N(m)$ in the above equation and equation (7) depends again on spectral truncation. For triangular truncation $N(m) = M$, for rhomboidal truncation $N(m) = |m| + M$. Rhomboidal truncation has better resolution than triangular in middle latitudes and if the emphasis is on higher latitudes, rhomboidal truncation may be preferred. For weather forecasting triangular truncation appears to be the universal choice.

The calculation of horizontal derivatives are straightforward and exact in spectral models. The zonal derivative can be expressed trivially for a Fourier co-efficient $\zeta^m(\mu)$ as

$$\frac{\partial(\zeta^m(\mu)e^{im\lambda})}{\partial x} = m\zeta^m \mu e^{im\lambda}$$

Similarly using the Rodrigues formula the meridional derivative can be obtained:

$$(1-\mu^2)\frac{dP_n^m}{d\mu} = -n\varepsilon_{n+1}^m P_{n+1}^m + \varepsilon_n^m P_{n-1}^m \qquad (10)$$

where $\varepsilon_n^m = \left(\frac{n^2-m^2}{4n^2-1}\right)^{\frac{1}{2}}$

Generally it is inefficient to compute the non-linear terms in the spectral space and hence these are computed on the physical space and then converted to spectral space using the above transformations. A detailed discussion of various aspects of application of spectral methods can be found in Durran (1999) and Krishnamurti, Bedi, and Hardiker (1998)

2.2.1 *Equations used in Eulerian Spectral Technique*

Using spectral methods it is more convenient to use vertical component of vorticity ζ and horizontal divergence D instead of the horizontal components of velocity u and v (as in equation(2)). In σ co-ordinate system (a non-dimensional terrain following system in vertical, this is generally used instead of distance or pressure co-ordinates) these equations can be written as (Haltiner and William, 1980):

The vorticity equation:

$$\frac{\partial\zeta}{\partial t} = -\nabla\cdot(\zeta+f)\vec{V} - k\cdot\nabla\times RT\nabla\left(\ln(p_s) + \dot{\sigma}\frac{\partial\vec{V}}{\partial\sigma}\right) \qquad (11)$$

where p_s is surface pressure, $\dot{\sigma}$ is the vertical velocity in sigma co-ordinates, f is the Coriolis parameter = $2\Omega sin\theta$ and $\vec{V}$ is the vector of horizontal velocities.

The divergence equation:

$$\frac{\partial D}{\partial t} = k\cdot\nabla\times(\zeta+f)\vec{V} - \nabla\cdot(RT\nabla(\ln(p_s)) + \dot{\sigma}\frac{\partial\vec{V}}{\partial\sigma} - \nabla^2(\phi+\vec{V}\cdot\vec{V}/2) \qquad (12)$$

The other equations i.e. for energy (equation 3) and species conservation (equation 5) remain largely the same. The spectral models are much more computationally efficient than finite-difference models of the same resolution and do not have major problems related to convergence of meridians at the poles (Durran, 1999). However to further increase the efficiency of these models, a 'reduced grid' is used at higher latitudes. In the 'reduced grid' there is a reduction in the highest wavenumber (i.e. Fourier coefficients above a particular wavenumber are assumed to be zero) and a corresponding decrease in the number of grids in the east-west direction.

From the above discussion we note that the coupling from the model dynamics viewpoint is quite tight in the horizontal direction, and from the model physics viewpoint, in the vertical direction. These factors make obtaining high scalability in parallel implementations a major issue.

2.3 *Finite Volume Semi-Lagrangian Technique*

The latest technique to be used in atmospheric modelling is the finite volume semi-Lagrangian technique commonly called as the Flux Form Semi-Lagrangian technique (FFSL) and is discussed in detail in Lin (2004).

The salient features of this technique are:

1. a pure explicit upwinding method
2. Time step splitting, smaller time steps to address fast gravity waves in the mass and momentum equations, larger time steps used for integrating species conservation equations.
3. use of finite volumes whose horizontal surfaces are Lagrangian bounding surfaces. The finite volumes are free vertically, they can float, compress or expand as dictated by the

hydrostatic dynamics. During computations in the Lagrangian mode there is no interaction in the vertical between the finite volumes.

4. computation of horizontal pressure gradient using a line integral on the latitude-vertical and longitude-vertical planes.
5. a mapping algorithm to convert from Lagrangian to the required Eulerian grid
6. solutions are largely free of spurious oscillations (a major problem with spectral models being the Gibbs oscillations)
7. it is found to be superior to spectral models for advection of trace gases and aerosols. Advection of trace gases and aerosols is gaining increasing importance with increased concerns about climate change.

The equations for these are written in flux form. We define a "pseudo-density" $\pi = \frac{\partial p}{\partial \zeta}$, where ζ is a generalized vertical co-ordinate and π is the vertical gradient of pressure in that co-ordinate. From hydrostatic balance (it is implicitly assumed that the scale of motion being studied is the larger scale and not the smaller cloud or meso-scale) the relationship between *pseudo-density* and true density is (ϕ is the geopotential height):

$$\pi = -\frac{\partial \phi}{\partial \zeta}\rho \tag{13}$$

The mass (and species) and momentum conservation laws are also appropriately used in the flux form. Detailed discussion of the flux form equations are given in Lin (2004).

This formulation uses a Lagrangian vertical co-ordinate and in FFSL, the Lagrangian surfaces are treated as bounding surfaces. This essentially reduces the three-dimensional problem to a two-dimensional one. The computation during the smaller timesteps for mass and momentum are done on the Arakawa-D Grid while the computations for species advection are done on the Arakawa C-Grid. For advection of species, accumulated winds from the smaller time-steps are used. The species integration step is larger by a about a factor of 2-3 vis-a-vis the momentum time-step. Meridional convergence of grids near the poles affects the size of the time step. To increase the timestep, higher frequencies are removed using a filter. The impact of the filter is minimized by applying it to variables on the C-grid and the tendency terms on the momentum D-grid.

FFSL uses dimensional splitting for computing meridional and zonal fluxes. Splitting errors are reduced by applying two orthogonal 1D flux-form operations in a directionally symmetric way. The inner operators i.e. zonal flux by meridional flux term and vice-versa are replaced by advective form operators. A piecewise parabolic method describes the variation of variables within the finite volume.

Finally it is necessary to ensure that Lagrangian surfaces do not get severely distorted during the process of integration. For this purpose the variables are mapped back periodically to the Eulerian vertical grid and computations recommenced with this Eulerian grid as the "initial" Lagrangian grid. All computations in FFSL are completely explicit and thus tailored for massively parallel computers.

3 Parallel Implementation of Atmospheric Models

Numerical techniques (discussed above) for solving the equations for atmospheric state need differing strategies for parallel implementation. Finite Difference (and finite volume) technique tend to have communications that are of the nearest neighbour type. While the communication stencil for finite difference models is largely static, for semi-Lagrangian finite volume techniques this could be variable, depending on the velocities (discussed in greater detail in subsection 3.3). Higher the velocity, longer the distance over which a parcel could travel and hence larger the communication stencil. Spectral techniques in contrast have communications which are more global in nature due to the transformation from grid-space to spectral space and vice-versa. In this section we briefly review some of the efforts at parallelizing various models. We begin with a

brief survey of parallel implementation of finite difference General Circulation Models followed by implementation of spectral and finite volume models.

3.1 Implementation of Finite Difference Models

Very few global climate models in the world use finite difference techniques. In contrast, regional meso-scale models (models for a small part of the globe with very high resolution, typically a few km, generally used for predicting weather on the local scale or extreme weather events such as cyclones), usually use finite difference techniques. We review two parallel implementations of finite difference models viz. implementation of SKYHI model (Jones, Kerr, and Hemler, 1995) and the implementation of UCLA GCM (Lau and Farrara, 1996) .

Implementation of SKYHI GCM

The parallel implementation of SKYHI supported both shared and distributed memory systems and had support for data parallel, message passing and work sharing programming models. It used the unstaggered grid, generally known as the "Arakawa-A grid". Its latitude-longitude grid has constant spacing in angular dimension i.e. given the sphericity of the earth, distances between grid points in the east-west direction reduce as one moves towards the poles. The numerical scheme in the model was explicit leap-frog (centered difference in time), this being done to exploit higher degree of efficiency on parallel computers. The explicit scheme made communications nearest-neighbour. The more traditionally used method for finite difference model is the semi-implicit leap-frog scheme. In this scheme, pressure-perturbation related terms are treated implicitly while advection, Coriolis and physical parameterization terms are treated explicitly. This kind of 'splitting' is usually done to increase the size of the time-step. The terms related to pressure perturbations give rise to fast-moving gravity modes while the other terms are related to slower moving meteorological modes. However use of a semi-implicit solver would need global communication to solve the Helmholtz equation for pressure perturbations. Global communication is a larger overhead as all the processors would need to exchange data leading to significant wait-states and corresponding reduction in computational efficiency. The disadvantage of an explicit scheme however is that significantly smaller time-steps are required due the Courant condition. This becomes a major issue in spherical grids as physical distances in the east-west direction reduce as one approaches the poles. Usually to prevent this, a Fourier filter is used to remove shorter waves.

Parallel computation is usually done by distributing the geographical domain over different processors – the domain decomposition technique. The decomposition is generally applied over dimensions over which the coupling is weak. In atmospheric models, the coupling in the vertical direction is very strong, especially in the computation of radiation and rain. In radiation computations, the infra-red radiation emitted by the surface and any layer of the atmosphere could be absorbed by other layers and re-emitted. This causes the coupling to be very strong. Similarly, rainfall calculation needs computation of deep-penetrative convection, in which conditions in the lower levels have significant impact on conditions at the upper parts of the troposphere. Therefore decomposition in the vertical (at least for the physics modules) is avoided. However, an exception to this is the implementation of a spectral GCM at NCEP (Sela, 1995) and the scalable parallel implementation of CAM3. These are discussed in sections 3.2.2 and 3.3 respectively. SKYHI implementation used two-dimensional decomposition in the latitudinal and latitudinal directions. In the data parallel implementation, prognostic and work arrays utilized a global address space and operations were carried simultaneously through the use of array syntax. In the message-passing implementation, local addressing was used and 'ghost-cells' or 'halos' were built around the decomposed sub-domains which contained additional data from neighbouring sub-domains. These data were required for computation of the derivate, first derivative for advection and second derivative for diffusion terms.

In work-share decomposition option, which was used on shared-memory systems, individual processors were assigned to work on separate parts or "chunks" in the spatial domain of the prognostic spatial loop. Statically-allocated prognostic and work arrays were declared as private to prevent processors from accessing the data used by other processors. The communication calls were isolated from the main body of the code into routines which were machine dependant and appropriate routines for a given machine were then chose at compile time. The scalability of the parallel implementation was quite satisfactory, with a speedup of about 10 on 16 processors on Cray-90. On a CM-5 the speedup from 64 to 256 processors was about 2.2.

UCLA GCM

Detailed report on implementation of UCLA GCM is available in Lau and Farrara (1996) UCLA GCM also uses an explicit scheme. It uses a staggered "Arakawa C-Grid". Usually staggered grids are preferred as they have better geostrophic adjustment i.e. any gravity wave disturbance (generally 'noise' for a meteorological model) moves out quickly leaving behind quasi-balance between Coriolis and pressure- gradient components of the horizontal momentum equation (Haltiner and William, 1980). Their implementation also uses a two-dimensional horizontal domain decomposition. They identified two bottlenecks in scalability of their implementation: (i) Fourier filtering and (ii) load imbalances in model physics.

Fourier filtering is required at higher latitudes to remove shorter waves. The physical distance between gridpoints reduces near the poles. If we do not filter the shorter wavelengths then smaller timesteps would be required. Fourier filtering requires conversion of all data on a latitude circle to Fourier co-efficients, removal of shorter wavelengths and reconstruction of filtered data in physical (grid) space. This does not pose a problem in one-dimensional decomposition in which most of the decomposition is done such that a set of latitude circles are assigned to a processor. However in two-dimensional decomposition, data across processors is needed. There are two ways of doing this:

1. Use of parallel FFT.
2. Transpose of data, use of sequential FFT and reverse transpose of data.

General experience is that transpose-sequential FFT is superior to parallel FFT (Drake, Foster, Michalakes, Toonen, and Woreley, 1995). Lau and Farrara (1996) also have used the same technique. This requires additional communications for transposing an array of data on to a processor. This is followed by application of sequential FFT. Consequent to this step, the filtered data is transposed back to the original processors. Since this filtering is to be applied at every vertical level and also on the wind components, each vector of such variables can be constructed on a separate processor and these computations can further be conducted in parallel.

The second major issue is load imbalance. Computational load is never usually the same across processors. In two-dimensional decompositions, processors handling day and night processors can suffer considerable load imbalance Michalakes and Nanjundiah (1994). The processors handling day sub-domains need to conduct additional computations for the incoming solar radiation. This can cause significant load-imbalance. Additional imbalances can occur in computation of rainfall. Processors handling tropical regions have a higher computational load (Nanjundiah, 2000).

Lau and Farrara (1996) have developed an algorithm for redistributing the load dynamically. First the load on each processor is calculated. Based on the load, each processor is ranked, the highest rank being given to the processor with largest load followed by processors in decreasing order of load. After ranking the processors, load is moved between processors with highest and lowest ranks, followed by processors with next ranking at the two ends and so on. Such a method of re-ordering of load ensures a more equitable load on all the processors and since the communications are between a pair of processors, the communication can be in parallel. The data is repeatedly shuffled in iterations till the load is almost

equitable. They compared this with other methods of load balancing and found it to be the most cost-effective. On a 64 processor Cray T3D, they show that after two iterations, the load imbalance in the physics part of the code is down to about 6% from about 37% in the beginning of the first iteration.

3.2 Implementation of Spectral Models

Spectral models are less suited for massively parallel implementation than grid-point or finite volume models. This is due the transformation of data from physical (grid) space to spectral space and vice-versa. There are many spectral GCMs with each having its own parallel implementation. We discuss here two major parallel implementations viz. Drake, Foster, Michalakes, Toonen, and Woreley (1995) and Sela (1995) as they provide contrasting methods of parallelization. While Drake, Foster, Michalakes, Toonen, and Woreley (1995) discusses the scalable parallel implementation of the NCAR's (National Center for Atmospheric Research) CCM2 (Community Climate Model version 2), Sela (1995) discusses the implementation of NCEP's (National Centers for Environmental Prediction, then known as NMC) MRF (Medium Range Forecast) model. Though both are spectral models there are some significant differences that are noteworthy:

1. NCAR used a single transformation from grid to spectral space and vice-versa in a single timestep

2. NCEP used two such transformations, one each for physics and dynamics part of the code in a regular timestep and a third transformation during the timesteps when additional computations for radiation are conducted. These radiation computations are typically invoked once every three hours and may be conducted on a smaller grid (i.e. on a lower resolution).

3. NCAR model stored data in both spectral and grid spaces for atmospheric variables. While this increased storage requirements, it was also more convenient for calculations.

4. NCEP model stored most of its atmospheric state variables in spectral space.

5. NCAR model used semi-Lagrangian moisture transport to circumvent Gibbs oscillation. These oscillations are an artifact of spectral transformation and could lead to overshoot (i.e. spurious rainfall) and undershoot (i.e. negative moisture over regions of low moisture).

It is interesting to note that while many changes have been made to NCAR's models, the numerical scheme remained largely unchanged from CCM2 to CAM3 till the inclusion of the optional semi-Lagrangian spectral dynamical core and the finite volume dynamical core. The Eulerian spectral dynamical core used in CAM3 is still the default option and is largely similar to the dynamical core used by Drake, Foster, Michalakes, Toonen, and Woreley (1995) over twelve years ago. Latest versions of NCEP still uses the same dynamical core. However substantial changes have been made to model physics and also to the code to improve portability and transparency.

3.2.1 Scalable Parallel Implementation of CCM2

While many versions of NCAR model have been developed after CCM2, no similar scalable parallel implementation has been attempted with them. Hence we discuss the implementation of CCM2 on a distributed memory system. A detailed discussion of this implementation is available in Drake, Foster, Michalakes, Toonen, and Woreley (1995). This implementation targeted massively parallel computing system that were then available such as the Intel Paragon (1024 processors) and the Touchstone Delta (512 processors).

Their implementation involved parallelization only in the horizontal space. They did not address parallelization in the vertical due to constraints of additional communications for model physics. Their parallel implementation involved parallelization of the two major steps in spectral computations viz. Fourier transform and the Legendre transform.

A one-dimensional decomposition by allocating chunks of latitude-circles would limit the scalability of transform e.g. for a typical T-42 resolution the number of latitude circles is 64. But co-latitudes (i.e. similar latitudes in the two hemispheres) are usually allocated to the same processor as there is an overlap in spectral calculations and hence the maximum number of processors that can be used by such a decomposition would be 32. Such a decomposition would leave the Fourier (and physics) computations untouched and parallel communications would be needed for completing the Legendre transform (equation 9). Various options would be available for this purpose:

1. partial sums of equation 9, followed by communication of these to other processors using a binary tree for higher efficiency.

2. transpose of vectors such that the entire set of co-efficients are available within a processor, followed a global sum on these – A transpose algorithm.

The first method needs smaller communications. However results on different processor would not be bit-identical, as the order of summation would differ in them.

For obtaining more sub-domains, Drake, Foster, Michalakes, Toonen, and Woreley (1995) used a latitude-longitude decomposition in the horizontal. However such a decomposition needs parallelization for both Fourier and Legendre transforms. For parallelization of Fourier transform two alternatives are available (a) parallel FFT and (b) transpose-sequential FFT. They find that transpose-sequential FFT is more efficient than parallel FFT. In addition since many variables (such as divergence, vorticity, temperature, surface pressure and some of their derivatives) need to be computed, many arrays are available for conducting Fourier transforms simulatanesouly on multiple processors. By transposing such that the entire array length is available within a single processor a sequential FFT can be used on it.

Further they show that the processor configuration can also be considered as a two dimensional array. For further parallelization of the Legendre transform they suggest that processors in the a column of such a two-dimensional grid should handle the same spectral co-efficients. Additionally they ensured that for the same wavenumber, all the spectral co-efficients were within the same processor column. Then each column of processors can communicate with each other in parallel and complete the Legendre transform. Furthermore, computations in the spectral space which do not have further interactions between wavenumbers can also be completed in parallel. Since care is taken that for a zonal wavenumber 'm' the entire data lies on the same column of processors, no further communication is needed during the inverse Legendre transform.

For inverse FFT, all the wavenumber data is available within the processors at the end of inverse Legendre transform. After using an inverse FFT, a transpose communication is conducted so that the updated data returns to the original physical grid configuration

CCM2 uses triangular truncation and thus the number of spectral co-efficients reduces with increase in zonal wavenumber. This could lead to a significant load imbalance if a set of wavenumbers were consecutively allocated to a processor. Therefore they used a re-ordering algorithm so that the computational load was equitably shared. The two-dimensional domain decomposition also caused significant variation between processors handling day and night domains. Processors handling day domains would handle additional computations for the incoming solar radiation (Michalakes and Nanjundiah, 1994). To overcome this load imbalance, gridpoints along a longitude were shuffled such that two grid points 180^o apart were together and the load imbalance removed.

The scalability for this implementation was reasonable even at 1024 processors and did not show any saturation. Though this implementation was not further carried to later versions of NCAR model, a version with this parallelized dynamical and CCM3 physics was used in the development of a coupled ocean-atmosphere model called FOAM (Tobis, Schafer, Foster, Jacob, and Anderson, 1997) which is still being used for studying

various aspects of climate.

The CCM2 also used semi-Lagrangian advection for moisture transport to reduce the impact of Gibbs oscillation. The semi-Lagrangian moisture advection is completely on the physical grid. 'Haloes', i.e. extra data points on all sides of the actual grid were added and data required for this exchanged. The size of these 'Haloes' depended on the velocities i.e. higher the velocity, larger the distance a parcel would travel and hence larger the halo. It was seen that instead of using square grids, rectangular grids with more points in the longitudnal direction were more computationally efficient.

3.2.2 NCEP Model Implementation

For parallel implementation of the NCEP (then National Meteorological Center, NMC) model a different strategy was adopted. When we examine the spectral method (equation 7, we notice that that for a spectral-to- grid transform and vice-versa, no vertical information is required. The NCEP model implementation (Sela, 1995), which mostly stores all its data in the spectral space, tries to exploit the fact that if all the data for a given level is available, then no further communication is needed. Therefore they have chosen vertical as the preferred dimension for domain decomposition. Further decomposition is done in the latitudinal direction if the number of processors available is larger than the number of levels. For physics computations a transpose is required. While the dynamics in this implementation shows good scalability, they say that the scalability of the physics component is less than satisfactory. This strategy is still being used by NCEP (Kanamitsu, Kumar, Juang, Yang, Schemm, Hong, Peng, Chen, and Ji (2002)).

3.3 Implementation of Finite Volume Semi-Lagrangian Models

Sawyer and Mirin (2007) discuss in detail the scalable parallel implementation of the CAM3 finite-volume model. Here we present a brief overview of the same.

The horizontal data dependency for the FFSL (Flux Form Semi-Lagrangian model) is determined by the dimensionless Courant numbers C^{λ} (for zonal i.e. east-west flow) and C^{ϕ} (for meridional i.e. north-south flow) given by:

$$C^{\lambda} = \frac{u\Delta t}{R\Delta\lambda cos\theta}$$

and

$$C^{\phi} = \frac{v\Delta t}{R\Delta\theta}$$

Generally the experience is that 'v' the wind in meridional direction is smaller than 'u', the wind in zonal (east-west) direction. In the zonal direction, winds are not only higher but as one approaches the poles, the physical distance for the same $\Delta\lambda$ reduces. A quasi-Lagrangian approach is used in this formulation. The longitudinal Courant number (in FFSL) at the edge of a cell between *i* and *i-1*, $C^{\lambda}_{i-\frac{1}{2}}$ can be greater than 1 and thus written as:

$$C^{\lambda}_{i-\frac{1}{2}} = K_{i-\frac{1}{2}} + c_{i-\frac{1}{2}} \tag{14}$$

where $K_{i-\frac{1}{2}}$ is the integer part of the Courant number and $c_{i-\frac{1}{2}} = mod(C^{\lambda}_{i-\frac{1}{2},K})$ is the fractional Courant number. Similarly the flux can be divided into an integer and fractional part. The zonal flux at the left edge of a finite volume can be written as:

$$\chi_{i-\frac{1}{2}} = (integerflux)_{i-\frac{1}{2}} + (fractionalflux)_{i-\frac{1}{2}} \tag{15}$$

This implies that there could be data-dependence on departure points which could be far off for a given Δt. Thus a logical way of decomposing this is to assign a set of latitude circles to a given processor. The data required for computing the zonal fluxes would be completely available within a processor and for meridional fluxes the data communication is of the nearest-neighbour type. Additionally all the data required in the vertical for 'model physics' would be available within the processor.

However the number of processors that can be used would be limited by the number of latitude

circles. Thus to exploit the power of large number of processors it is necessary to decompose in more than one dimension. Examining the FFSL (section 2.3) we notice that during the computation of dynamics, the horizontal surfaces of finite volumes are 'material' surfaces i.e. there is no flow in the 'vertical direction' and hence the algorithm is essentially 2-dimensional. Thus the computations in the dynamics can be parallelized in the vertical direction. Additionally as seen above, data dependency is much less in the meridional direction. Thus for the dynamics, a decomposition in latitude and height i.e. chunks of latitudes circles with part of the vertical domain are assigned to a processor. However, for calculation of geo-potential, a global sum in the vertical was required. For this they initially used a transpose-sum-transpose algorithm (as discussed in section 3.2.1). This was replaced by partial sums within the processors which were exchanged between processors to compute the global sum. However, the partial sum technique is not bit-equivalent to the sequential code as the order of summation is not maintained.

Eulerian re-mapping and physics computation are tightly coupled in the vertical. Hence for this, a decomposition using of chunks of latitudes and longitudes but with entire data in the vertical on a single processor are used. Sawyer and Mirin (2007) have implemented an efficient transpose technique to convert data from one method of decomposition to the other.

They have tested their parallel implementation on a variety of machines and also used MPI-2 which allows one-sided communication. Further they incorporated multi-threading and found significant improvements. For this, the one-way communication is decomposed into several blocks and the processes of communication (MPI_PUT) is multi-threaded. They have run their application on large systems using upto 2944 processors at 0.5x0.625*26L (latxlonxlevs). They find good scalability. The amount of time spent in transpose increases from about 7 % at 32 processors to 32 % at 2944 processors.

The scalability of FFSL being good, it is not surprising that development of quite a few models, notably the next generation of NCAR's model, use this as the default dynamical core.

3.4 Co-Evolution of GCMs and Supercomputers: issues of efficiency and scalability

In the above sections we have seen an interesting evolution in the development of atmospheric models. The initial atmospheric models used finite difference techniques. Then spectral models became popular and now the current trend appears to be towards finite volume techniques. If we trace the evolution of computers, we notice that real breakthrough in computing speeds occurred with the development of the vector computers such as the Cray during the mid 70s. Vector computers could perform more efficiently on large vector-lengths. The on-board memory on computers was expensive Thus the spectral models which had long vector lengths for manipulation and which had lower memory requirement (if most of the data were stored only in spectral space), were inherently better suited to exploit this. The superior computationally efficiency of a spectral model vis-a-vis a finite difference model in the form of a larger time-step and higher accuracy of the numerical method also contributed to its increased popularity. In the late 80s, microprocessor based massively parallel started gaining popularity. The experience was that spectral GCMs had more problems in scalability than their finite-difference counterparts. Also as seen above, spectral models had serious limitations in advecting chemical tracers and aerosols whose importance increased with growing concerns of climate change. Thus the better scalability of finite- volume techniques coupled with their superior properties of advecting tracers in conjunction with the evolution of massively parallel processor has led to a heightened interest in finite-volume techniques.

4 The Indian Experience

It would not be out of place to highlight the fact that Indians were one of the pioneers in applying parallel computing to fluid dynamical problems. Thus it also not surprising that one of the earliest software to be implemented by Indians on

a parallel computer was a climate model (Nanjundiah, 1988). In contrast to efforts in other parts of the world, the efforts were modest and tailored towards solving a particular problem (in this case understanding the monsoon variability) rather than looking for scalability for a large number of processors. This was in consonance with the hardware development efforts of the times when modest sized parallel computers were being fabricated by Indian researchers.

4.1 Implementation of Finite Difference Models on Flosolver

One earliest parallel computer to be developed in India was the Flosolver Mk1 at National Aerospace Laboratories. This parallel computer used intel 8086 as the CPU and the interprocessor communication was through MultiBus-I – a device which was not considered suitable for parallel computing by its own makers, as it was likely to reach saturation levels very quickly. Problems of bus-contention were skillfully solved using random numbers to allocate the use of communication bus to a processor. For details of this pioneering development see Sinha, Deshpande, and Sarasamma (1988). The interprocessor communication occurred through a global memory to which a processor would write its data and the same would be read by another processer (which had requested the data). Initially a Laplace solver and Transonic small perturbation software were parallelized on the Flosolver. The developers of Flosolver felt emboldened enough to look for a climate model for implementation on the Flosolver. The same was done in collaboration with Centre for Atmospheric Sciences (now Centre for Atmospheric & Oceanic Sciences, CAS), Indian Institute of Science, Bangalore. For this implementation, Flosolver Mk2, which again used MultiBus-I but 80386 processor was used. The researchers at IISc were facing a complementary problem. Having developed a climate model to study the intraseasonal variability of monsoons, they were hungering for computational power which was unavailable to them due to the technology denial regime in force in those times. It is also interesting to note that we know of no published effort from this period which discusses a successful parallel implementation of a climate model. Thus perhaps it could be said that this was one of the earliest (if not the first) parallel implementation of a climate model on a parallel computer.

The first model to be implemented was a zonally-symmetric two-level model i.e. a model which ignored variations in the east-west direction. It used a centered differencing scheme in the horizontal and a leap frog scheme in time. The communication in this was largely of the nearest neighbour type. Only the shifting sea-ice boundaries which changed with time needed global communication. This was done by assembling the required variables on a single processor and sending the relevant information about these boundaries to the other processors. Experiments on Flosolver helped CAS scientists to address one of the key concerns of the model developers i.e. whether the poleward propagations of cloudbands simulated by their model were sensitive to horizontal resolution. Running the model at higher resolution than could be done at CAS at that time, they ascertained the robustness of their results. Some of the results obtained from these experiments on Flosolver helped to understand the mechanism of intraseasonal variation of the monsoons and were reported in Nanjundiah, Srinivasan, and Gadgil (1992)

The second climate model to be implemented at Flosolver was a three-dimensional model (Nanjundiah and Sinha, 1992). This was implemented on Flosolver Mk3. Mk3 used the 80860 RISC processor connected by MultiBus (I & II). This model used the Arakawa B-grid and the Arakawa cumulus convection scheme. The numerical scheme was pure explicit - a combination of predictor-corrector and leap-frog schemes. Most of the communications were of the nearest neighbour type. Domain was decomposed in the latitudinal direction. Checking of moisture minima (to prevent humidity from going negative) however needed processing along a longitude. This needed global communication. The scalability was however low due to the fact that while the processing power had increased from 80386 to 80860, the communication hardware had

remained the same viz. Multibus.

5 Implementation of Spectral Models on Indian Parallel Processing Systems

The benchmark for testing Indian parallel processing system has been the NCMRWF model at T-80 resolution. Almost all systems have been benchmarked for this code. The reason for this was the project sponsored by Department of Science & Technology (Govt of India) on usability of Indian parallel processing systems for weather forecasting. Many institutions notably National Aerospace Laboratories (NAL) Bangalore, Centre for Development of Advanced Computing (CDAC) and Bhabha Atomic Research Centre (BARC) with active support from scientists of NCMRWF took part in this effort. These efforts have been documented and widely debated in the Indian atmospheric science community.

The NCMRWF model is a spectral model and for the purpose of benchmarking was used at T-80 resolution (and hence commonly if somewhat erroneously known as the T-80 model). In its dynamics and most of physics, it closely resembles the NCEP model (Sela, 1995)

5.1 NCMRWF Model Parallelization at NAL

Initial effort for parallelization of the NCMRWF T-80 model was begun on the Flosolver Mk3 computer which used intel 80860 processor (commonly known as i860) and MultiBus for communication. Since the number of processors used on this system was small (about 8) only one-dimensional domain decomposition in the latitudinal direction was used. This involved parallelization of the Legendre transform. However given the weak communication interface, only a sequential ring summation could be implemented. This proved a bottleneck for scalability.

Basu (1998) came out with a criticism of all Indian parallel processing efforts and argued that all these efforts had shown poor scalability. In response, Sinha and Nanjundiah (1998) analysed the efforts at NAL and elsewhere and showed that the scalability obtained by the Indian efforts were at par with those obtained for the NCEP model on Cray C90 and for the ECMWF model on the Fujitsu system. A further indepth analysis of the bottlenecks in parallelizing the NCMRWF code was conducted by Venkatesh, Rajalakshmy, Sarasamma, and Sinha (1998). They showed that scalability could also be affected by the parts left for sequential computations in the parallel software, on the premise that they consumed small amounts of computational time in the sequential code. One of these was the linear part of computations in the spectral domain. They showed that for a configuration larger than 4 processors, it was necessary to parallelize these parts of the software also and this would lead to much better scalability.

5.2 Re-engineering of the model

While the model was being analysed for its shortcomings, it was also felt that portability of the model was a major issue. In addition to improving the scalability of the existing code, it was also decided to re-engineer the code to make it platform independent (Nanjundiah and Sinha, 1998). This re-engineering exercise, done over a period of about three years, fructified with the model transforming from a fixed resolution, platform-dependent model to one which was platform independent and could be used for a variety of purposes including class room teaching (which could be done by running the model at extremely low resolutions) While the original model had Cray-specific constructs, the new model used Fortran 90 features and was completely platform independent and could even run on PC's using MS-windows as the operating system. Current Science, the popular Indian science magazine, hailed this move in its editorial and called it as a move from a closed style of computing to an open 'bazaar style'.

This re-engineering has spurred further research into modifying the model. Not only the software aspects have been modified but also new schemes for boundary layer based on weakly forced convection, and a new radiation scheme with higher resolutions at the lower layers have been incorporated.

5.3 Hardware innovations tailored to meteorological requirements

The criticism of Indian parallel processing systems also forced some hard-thinking among the developers and designers of Flosolver. Detailed analysis of the NCMRWF model's performance showed up the bottleneck to be in inter-node communication (Venkatesh, Sinha, and Nanjundiah, 2001). This was related to the weak interprocessor link used in indigenous parallel processing systems. Since Legendre transform (as discussed above) has global summation, different types of communication stencils were analysed viz. gather-sum-broadcast, ring-summation, binary-tree summation. The estimates were made for each of these methods of summation. For a given number of processors n_p, if t_{cp} is the time required to communicate N words between two processors and t_{sum} is the time required to add two arrays of length N then the total time, T_{total} required for this summation using various methods are:

1. for gather-sum-broadcast

$$T_{total} = 2 \times (n_p - 1) \times t_{cp} + (n_p - 1) \times t_{sum} \tag{16}$$

2. for ring summation:

$$T_{total} = (n_p - 1) \times (t_{cp} + t_{sum}) + (n_p - 1) \times t_{cp} \tag{17}$$

3. For binary tree-summation (which involves parallel exchange of data between pairs of processors):

$$T_{total} = log_2 n_p \times (t_{cp} + t_{sum}) + log_2 n_p \times t_{cp} \tag{18}$$

Comparing equations (16, 17 and 18) we note that time for gather-sum-broadcast and ring are identical. For binary tree the times are much less. This however assumes that messages between two pairs of processors were perfectly parallel (which was not the case with the locally produced parallel computers). The tree summation was further analysed and shown that there were stages where individual processors would wait for intermediate sums and that the largest factor t_{cp}, depended on the processor-to-switch communication bandwidth and the switch bandwidth. Additionally for bit-reproducible results, binary tree-summation could not be used.

This led to thinking about customizing communication hardware especially for weather/climate models which have global reduction operations and a radically new design of a communication switch emerged. Switches hitherto were passive, merely communicating messages between the nodes of a parallel computer. In the new design, its role was to be that of an active element in global reduction operations. A simple algorithm for such a switch would be:

1. all processors communicate their data (in case of spectral models, spectral coefficients) to the switch,
2. the summation (or any other global reduction operation) is conducted on the switch
3. the summed (or reduced) data is sent back to the individual nodes.

The procedure is shown schematically in figure 1.

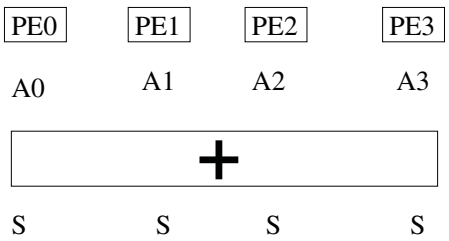

Figure 1: Schematic of summation on the Floswitch

If we analyze the above procedure, we see that steps (1) and (3) can be parallel and only during (2) would the processors wait. However, the wait in this case would not be sequential (as in the previous algorithms). The t_{total} for this case would

be:

$$T_{total} = 2 \times (t_{cp}/2) + t^{\star}_{sum} \tag{19}$$

where $t^{\star}_{sum}$ is the time taken for summing n_p arrays on the switch it would be $t^{\star}_{sum} = (n_p - 1)t^{\star}_s$ if added in sequence and $t^{\star}_{sum} = \log_2 n_p t^{\star}_s$ if added in a binary tree.

A comparison of equations(18, 19) for binary tree-summation and the designed switch (called the Floswitch) shows that $log_2 n_p$ term multiplies t_{sum} in Floswitch. Since summation is much faster than communication, the scalabilty would be higher. Picture of an early version of the switch is shown in figure (2). This switch in essence is:

- Scalable
- Global communication is symmetric
- Order of summation (or any other reduction operation) can be preserved

This switch has now been augmented into an optical switch and currently acts as an interconnect for upto 128 nodes in the latest version of Flosolver. The software has also undergone augmentation, again completely re-written in C and extensively modified and is now called the Varsha GCM. It is being used for in-house research on forecasting of monsoons on different scales.

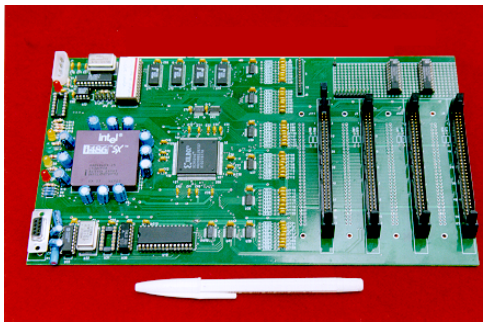

Figure 2: An early version the Floswitch

5.4 Atmospheric/Climate Modelling at CDAC

Centre for Development of Advanced Computing (CDAC) was established in the late eighties to foster research on development of high performance computers and their application to various fields including weather/climate modelling. Over the years it has gathered enough expertise in these fields. They also implemented the NCMRWF model on their Param range of computers (Purohit, Singh, Narayanan, and Kaginalkar, 1996). The earliest implementation was on the Param-8600, a computer built around the i860 processor. The i860 was linked to transputers, which in turn were used as communication devices and the interconnect between transputer and i860 was a bus called the Data Restructuring Engine (DRE). They also used domain decomposition in the latitudinal direction with communications for the parallelized Legendre transform. Another salient feature of their implementation was efficient use of cache by using vector registers for frequently used variables. Some of the model's codes were also replaced with system calls. This resulted in considerable improvement in throughput timings. Kaginalkar and Purohit (1996) further discuss the benchmarking of the NCMRWF model on the Param 8600. Instead of using a tree -algorithm for summation, they used a hypercube algorithm. This was implemented on their 8 node dual-CPU SMP cluster. They intercompared different communication interfaces for their system and found that Myrinet performed better than other interfaces. CDAC has developed a general purpose high-speed switch, PARAMNET, which is used for developing computers at CDAC such as the Param-Padma.

CDAC is now also working on meso-scale models such as WRF and MM5. In addition to it, a high resolution version of NCEP model at T-170 ($\approx$ 70 km) is being used in the climate mode for seasonal forecasts and a version of the French LMDZ model for aerosol transport studies over the Indian region.

5.5 Parallel implementation of a spectral model model at IIT-Delhi

Dash, Selvakumar, and Jha (1995) describe the parallel implementation of a spectral model at T21 and T42 resolutions on a transputer based parallel computer developed by Centre for De-

velopment of Telematics (C-DOT). The resolution of this model has been increased and used extensively for climate studies including the effect of Eurasian snow-cover on Indian monsoon (Dash, Singh, Shekhar, and Vernekar, 2003), and orography (Dash, Singh, Shekhar, and Vernekar, 2005). One of the noteworthy experiments conducted by them was the parallelization in the longitudinal direction i.e. a set of longitudes are given to a processor. By this process all computations for Legendre transform are in-processor and do not require communications. Communications are however required for Fourier transforms. Additionally this could lead to load imbalances due to diurnal cycle of radiation between processors.

Other centres such as BARC have also implemented the NCMRWF model on their parallel computers. The parallel machine built by BARC has been used for operational forecasting by NCMRWF (Jagadeesh, Rajesh, Phoolchand, Dhekne, and Kaura, 2000).

At IISc parallel versions of NCMRWF model, CCM2, CAM3 and climate system model, CCSM2/3 have been used to address a large number of problems including impact of numerical methods on tropical circulation, role of black carbon aerosols on the strength of the Indian summer monsoon, relationship between African orography and the Indian summer monsoon and other topics related to the monsoons and its variability. Latest work at IISc on improving scalability and efficiency of atmospheric models and climate system models is discussed next.

5.6 Improvement of Scalability through message compression

Most of work on scalability involves improvement of algorithms. We know of very few studies where for a given communication stencil and a given parallel machine, the impact of compressing messages has been studied. A study has been recently conducted at IISc on the role of message compression on the scalability of an atmospheric model (Kumar, Nanjundiah, Thazuthaveetil, and Govindarajan, 2008). The atmospheric model used was Community Atmospheric Model (CAM3) and the computer used was the IBM Power5 cluster based in the Supercomputing Education & Research Centre (SERC) of IISc. It is a SMP cluster with 4 processors per node and uses a Gigabit-ethernet interconnect for inter-node communication. The default parallel implementation of CAM3 was used for this purpose . Two methods of message-compression were used viz. lossless and lossy. Lossless compression uses the fact that data between iterations of the model does not change significantly and only those bits that change are sent. On the receive-side this data is used for perfect reconstruction of the message. Lossy compression, involves reduction in the number of bits transmitted. However, lossy compression involves a threshold of acceptability which could be application-specific. Combination of lossy and lossless techniques were also tried. The results for lossless compression are shown in 1 and those for lossy compression are shown in figure3. With lossless compression, improvements upto 15 % were obtained. With lossy compression improvements of upto 32 % were observed.

Such message-compression techniques could be very useful in grid-computing, where interfaces are even slower and any reduction in message size could lead to significant improvements in throughput. Work on grid-enabling climate system models is currently underway at IISc and CDAC.

5.7 Climate System Modelling in India

In contrast to atmospheric models which have been in use for about 15 years in India, use of climate system models is still in its infancy. Climate system modelling involves the simultaneous evolution of coupled states of atmosphere, ocean and land. Hence models for atmosphere, ocean, land and sea-ice need to be executed interactively. Thus the complexity of the models increases fourfold and computational load is much higher. A schematic view of climate system model is shown in figure 4. To the best of our knowledge the first implementation of a climate system model within the country and its use for climate studies was done by scientists at CDAC and IISc (Janakiraman, Nanjundiah, and Vinayachandran, 2005).

Table 1: CAM Performance T42: Speedup for Lossless compression scheme

Number of Processes	Original Execution				With Intra step Lossless Execution			
	Total Execution Time (secs)	Speedup	Physics Execution Time (secs)	Dynamics Execution Time (secs)	Total Execution Time (secs)	Speedup	Physics Execution Time (secs)	Dynamics Execution Time (secs)
1	34044	1.00	29496	3467	–	–	–	–
16	3447	9.89	2180	1180	3034	11.22	2185	758
32	2516	13.53	1127	1317	2130	16.	1166	895

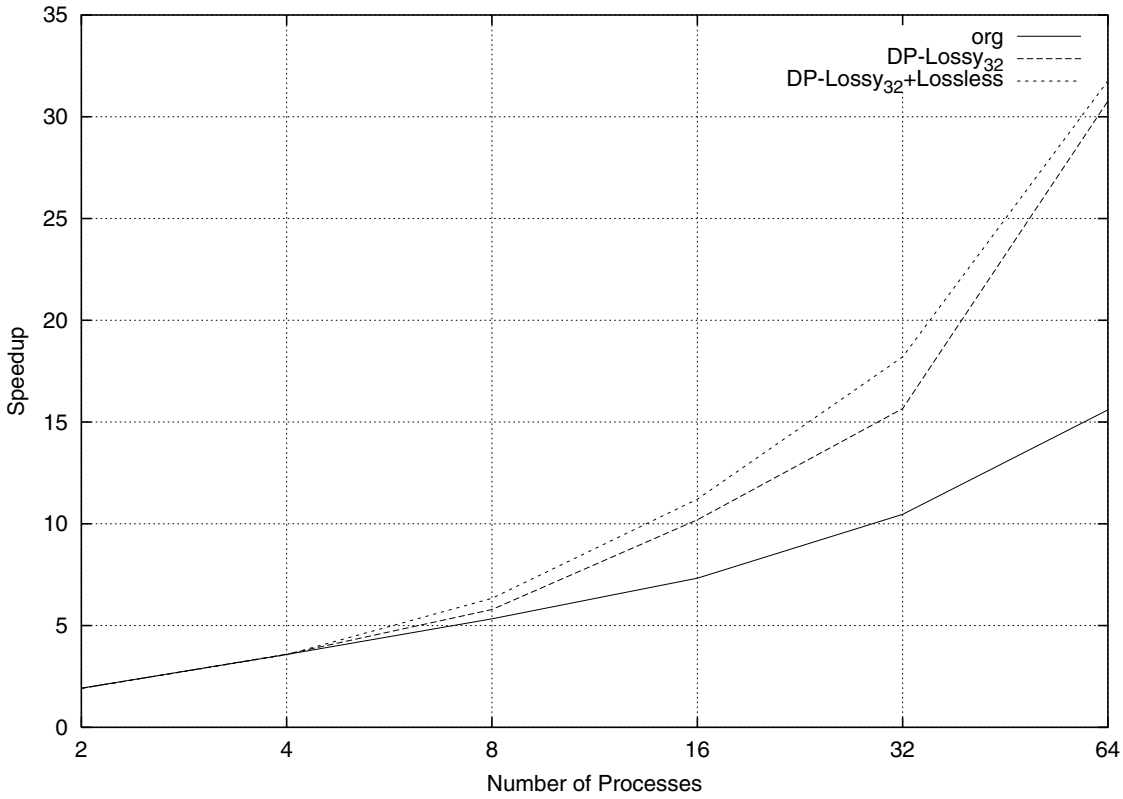

Figure 3: Speedups for CAM3 T-42 on a Power5 cluster with lossy compression

The Community Climate System Model version 2 was implemented on Param-Padma, the first computer from India to feature on the Top500 list, with a peak computing capacity of about 1 Teraflop. This was a MPMD implementation with five executables for atmosphere (CAM2), ocean (Parallel Ocean Program, POP), land (Community Land Model) and a sea-ice model (Community Sea Ice Model). It was run on 104 processors of Param Padma. A 100 year simulation was conducted and is perhaps the largest numerical experiment conducted within the country in the field of climate modelling. The MPMD implementation involved heterogeneous implementations, while CAM2 used both MPI and OpenMP constructs, POP was implemented only with MPI, coupler with only OpenMP, CLM with both MPI and OpenMP and CSM with MPI.

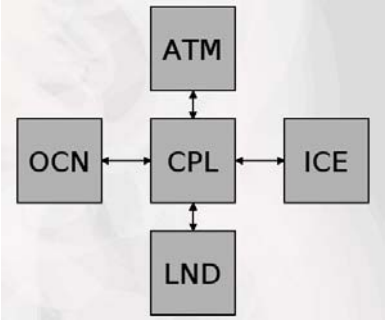

Figure 4: Schematic of a climate system model showing interlinking of four separate executables through a coupler

5.8 Load Balancing for the Climate System Model

A later version of the Climate system model, CCSM3, has been implemented on IISc's computers. This like its predecessor, is also a MPMD implementation. During the process of implementation it was noticed that significant load imbalances occurred periodically in the atmospheric component of the model. It was also noticed that during these periods of peak loads, the other components of the climate system model were idle. Further analysis of this peak load showed that this was related to long wave computations, specifically the computations of absorptivity and emissivity. These co-efficients being very computationally intensive are calculated only once in 6-12 hours. The peak load is more than 4 times the base load. Thus it was felt necessary to reduce the load imbalance and improve the throughput of the climate system model. Most other previous studies of load imbalance in a climate system model have looked at intra-component load balancing (Carr, Carpenter, Cordery, Drake, Ham, Hoffman, and Worley, 2005). Conventionally, a "rule of thumb" is used to balance loads across components. This involves allocating highest number of processors (more than half) to atmosphere, the most computationally intensive component, followed by ocean, land and sea-ice (Sundari, Vadhiyar, and Nanjundiah, 2007). However, no attempts have been attempted at inter-component load-balancing. This, to our knowledge, is the first such attempt.

The balancing algorithm involves moving a part of the emissivity and absorptivity calculations from processors handling atmospheric computations to processors handling other components. While doing this, it is necessary to ascertain whether other processors are free to conduct these computations. Otherwise it could lead to more idling time. For this purpose data about computational loads is gathered dynamically during run-time. During this period of data-gathering the load-balancing algorithm is switched off (this is done periodically, around once in 15 days of simulation). Using this data-on-the-fly, it is ascertained whether loads can be shifted to other components. Also depending on the computational loads on the processors, the amount of computational load that can be shifted are determined. This load balancing algorithm gives significant improvements, especially at low processor configurations (figure 5). Analysis and further work is currently underway to test these schemes on various machines including shared memory machines, clusters and distributed shared memory systems.

5.9 Accuracy of Computations

Accuracy of computations in atmospheric modelling has always been a major issue. It is well-known that no two computers give bit-identical forecasts. Even on the same computer, it is has been noticed that changing the compiler options can lead to different results. Internal rounding-off of floating point data can lead to such results. Conditional branching on the values of floating variables (if statements) and a small difference in values computed around the branch value can lead to very differing paths of integration (e.g. if the conditional value is 1 and if one computer calculates this as 0.999 and the other calculates this as 1.001 then the two will take different branches on the conditionality and the results could be significantly different). Rosinski and Williamson (1997) have looked at this issue and have come up with a technique to ascertain whether port to a computer is acceptable or not. This involves comparison of error growth with respect to results from a com-

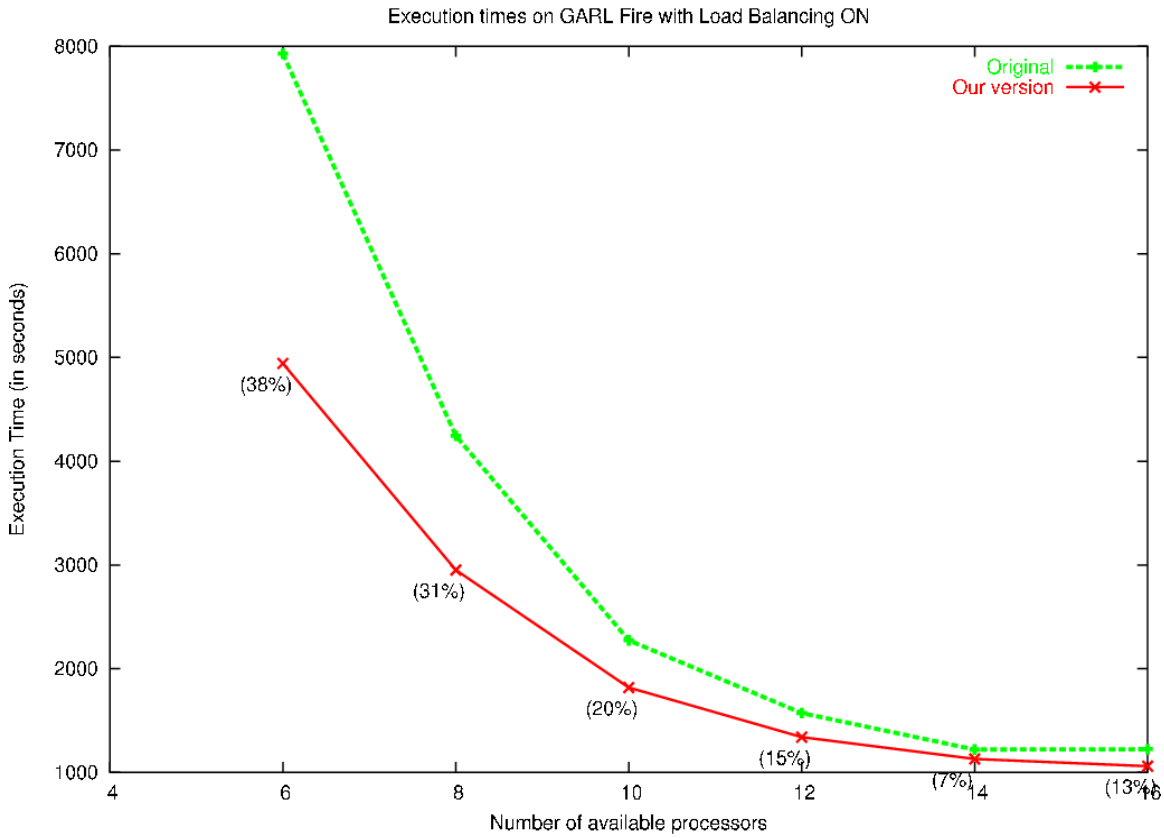

Figure 5: Timings for the Climate System Model on a cluster. Numbers in brackets indicate the percentage improvement over the original code.

puter with a 'valid' port and the error growth on the 'valid' port computer when the input data is perturbed at the least significant bit. If the error growth on the new machine is less than the perturbation error growth then it is considered a valid port. This comparison can at-best be done for short integrations. For long climate-mode integrations, the comparison is generally qualitative. In parallel computing, the results could change when number of processors are changed. This being due to change in order of summation (in Legendre transform) leading to differing results. These are related to the roundoff errors in the computations. Janakiraman, Ratnam, and Kaginalkar (2000) showed that rounding off towards positive infinity or negative infinity even when using the IEEE 754 standard introduces greater differences in results than changes in compiler options or different computing platforms. Therefore these method of perturbations provide more reasonable estimate of errors.

Recently Venkatesh and Sinha (2007) have also looked at this issue. They also note that even for sequential runs with different types of processors the results can change and these changes become noticeable beyond twelve days (in rainfall patterns). To overcome this problem they suggest the use of multi-precision i.e. the number of digits to which the variables are stored be increased. Storing 64 digits (not bits) or more during computations, they show that the threshold of repeatability for both sequential and parallel simulations (with different number of processors) increases to 30 days. To further determine whether the chaotic behaviour of the Lorenzian system is sensitive to precision, they have conducted simulations for the Lorenzian model with variables stored upto 2048 digits. They found that the time at which two solutions with same initial perturbation diverge by an order of magnitude does not depend on precision.

6 Conclusions

In this paper we have surveyed the application of HPC to climate/climate system modelling with specific reference to atmospheric modelling. We have looked at the Indian and international scenarios. The survey is by no means complete or exhaustive. Only those studies which we considered pioneering or novel have been considered (and thus could be subjective).

Examining the Indian scenario we notice that the implementations have been in the moderately parallel range. However this does not decrease the novelty of these studies. Some of the earliest parallel implementations were done in India. The development of a customized switch tailored to the needs of meteorological computing is unique. These efforts show that Indian climate modellers and computational scientists can 'think-out-of-the-box' to solve their problems.

The international scenario is particularly relevant as India is witnessing a massive expansion of computational capacities. However it should also be noted that the problem related to forecast of monsoons on various scales is a very tough one and needs to be solved in a synergistic fashion involving computational scientists, climate modellers and weather forecasters. One also needs to look at emerging trends in computing such as possible use of Gaming Processor Units (GPUs) and cell-processors, improvements in numerical techniques and developments in data assimilation techniques to solve problems related to monsoon forecasting in an innovative fashion.

Acknowledgement: The author gratefully acknowledges the help of his colleague Dr Satish Vadhiyar (VSS) and his student Ms Sundari in conducting the load balancing calculations. He is also thankful to VSS for his useful comments and suggestions on an earlier version of the manuscript. He also thanks Drs T. N. Venkatesh (NAL) and Akshara Kaginalkar (CDAC) for their useful inputs. This work is partally supported by Ministry of Information Technology, Govt of India, under project no. DIT/R&D/C-DAC/2(10)/2006/DT.30/04/07. res

References

Arakawa, A.; Lamb, V. (1981): A Potential entsrophy and energy conserving scheme for shallow water equations. *Mon. Wea. Rev.*, vol. 109, pp. 18–36.

Basu, B. K. (1998): Usability of parallel processing computers in numerical weather prediction. *Current Science*, vol. 74, pp. 518–516.

Carr, G. R.; Carpenter, I. L.; Cordery, M. J.; Drake, J. B.; Ham, M. W.; Hoffman, F. M.; Worley, P. H. (2005): Porting and performance of the community climate system model (ccsm3) on the cray x1. In *Proceedings of the 11th Workshop on High Performance Computing in Meteorology, Reading, UK, October 25-29-19, 2004*. World Scientific.

Dash, S. K.; Selvakumar, S.; Jha, B. (1995): Climate Modelling using Parallel Processors. *Atmos. Env.*, vol. 29, pp. 2001–2007.

Dash, S. K.; Singh, G. P.; Shekhar, M. S.; Vernekar, A. D. (2003): Influence of Eurasian Snow Depth Anomaly on Indian Monsoon Circulation . *Mausam*, vol. 54, no. 2, pp. 427–442.

Dash, S. K.; Singh, G. P.; Shekhar, M. S.; Vernekar, A. D. (2005): Response of the Indian Summer Monsoon circulation and rainfall to seasonal snow depth anomaly over Eurasia . *Climate Dynamics*, vol. 24, no. 1, pp. 1–13.

Drake, J.; Foster, I. T.; Michalakes, J.; Toonen, B.; Woreley, P. (1995): Design and Performance of a Scalable Parallel Community Climate Model . *Par. Comp*, vol. 21, no. 10, pp. 1571–1591.

Durran, D. R. (1999): *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics*. Springer.

Haltiner, G. J.; William, R. T. (1980): *Numerical Prediction and Dynamic Meteorology*. John Wiley & Sons.

Jagadeesh, B.; Rajesh, K.; Phoolchand, R. S.; Dhekne, P. S.; Kaura, H. S. (2000): Anupam-alpha parallel computer for operational weather

forecasting. In *Fourth International Conference on High Performance Computing in the Asia-Pacific Region*, pp. 1140–1145. IEEE.

Janakiraman, S.; Nanjundiah, R. S.; Vinayachandran, P. N. (2005): Simulations of Monsoons with coupled ocean atmosphere model on Param Padma. *Current Science*, vol. 89, pp. 1555–1562.

Janakiraman, S.; Ratnam, J. V.; Kaginalkar, A. (2000): Study of machine round-off response on weather forecasting simulations using high performance computing systems. In *HPC Asia Beijing, China*.

Jones, P. W.; Kerr, C. L.; Hemler, R. S. (1995): Practical considerations in development of a parallel SKYHI general circulation model. *Par. Comp*, vol. 21, no. 10, pp. 1677–1694.

Kaginalkar, A.; Purohit, S. C. (1996): Benchmarking of medium-range forecast model on param: A parallel machine. In *Making its Mark: Proceedings of the seventh Ecmwf Workshop on the Use of High Performance Computing in Meteorology*. World Scientific.

Kanamitsu, M.; Kumar, A.; Juang, H. M.; Yang, F.; Schemm, J.; Hong, S. Y.; Peng, P.; Chen, W.; Ji, M. (2002): NCEP Dynamical Seasonal Forecast System 2000. *Bull. Amer. Met. Soc*, vol. 83, pp. 1019–1037.

Krishnamurti, T. N.; Bedi, H. S.; Hardiker, V. M. (1998): *An Introduction to Global Spectral Modelling*. Oxford University Press.

Kumar, V. S.; Nanjundiah, R. S.; Thazuthaveetil, M. J.; Govindarajan, R. (2008): Impact of Message Compression on the Scalability of an Atmospheric Model. Application on Clusters. *Par. Comp.*, vol. 34, no. 1, pp. 1–16.

Lau, J. Z.; Farrara, J. D. (1996): Performance Analysis and Optimization on the UCLA Parallel Atmospheric General Circulation Model Code. *Supecomputing, 1996. Proceedings of the 1996 ACM/IEEE Conference on Supercomputing*.

Lin, S.-J. (2004): A "Vertically Lagrangian" Finite-Volume Dynamical Core for Global Models. *Mon. Wea. Rev.*, vol. 132, pp. 2293–2307.

Michalakes, J. G.; Nanjundiah, R. S. (1994): Computational Load in Model Physics of the Parallel NCAR Community Model. Technical Report MCS-TM-186, Mathematics and Computer Science Division, Argonne National Laboratory, 1994.

Nanjundiah, R. S. (1988): Simulation of Monsoon on Flosolver. Technical Report PDAE8803, National Aerospace Laboratories, Bangalore, 1988.

Nanjundiah, R. S. (2000): Seasonal Simulation of the Monsoon with the NCMRWF Model. *Current Science*, vol. 78, pp. 869–875.

Nanjundiah, R. S.; Sinha, U. N. (1992): Implementation of A Three-Dimensional Global Climate Model on Flosolver Mk3 Parallel Computer. Technical Report 92-AS-3, CAS, Indian Institute of Science, 1992.

Nanjundiah, R. S.; Sinha, U. N. (1998): Impact of modern software engineering practices on the capabilities of an atmospheric general circulation model. *Current Science*, vol. 76, pp. 1114–1116.

Nanjundiah, R. S.; Srinivasan, J.; Gadgil, S. (1992): Intraseasonal Variation of the Indian Summer Monsoon. II: Theoritical Aspects. *Jour. of Jap. Met. Soc.*, vol. 70, no. 2, pp. 529–550.

Purohit, S. C.; Singh, T. V.; Narayanan, P. S.; Kaginalkar, A. (1996): Glogal spectal medium range weather forecasting model on Param. *Supercomputer.*, vol. 13.

Rosinski, J. M.; Williamson, D. L. (1997): The accumulation of Rounding Errors and Port Validation for Global Atmsopheric Models. *SIAM J. Sci. Comput*, vol. 18, no. 2, pp. 552–564.

Sawyer, W. B.; Mirin, A. A. (2007): The implementation of the finite-volume dynamical core in the community atmosphere model. *J. Comput. Appl. Math.*, vol. 203, no. 2, pp. 387–396.

Sela, J. (1995): Weather Forecasting on Parallel Architectures . *Par. Comp*, vol. 21, no. 10, pp. 1630–1654.

Sinha, U. N.; Deshpande, M. D.; Sarasamma, V. R. (1988): Flosolver: A Parallel Computer for Fluid Dynamics. *Current Science*, vol. 57, no. 2, pp. 1277–1285.

Sinha, U. N.; Nanjundiah, R. S. (1998): Usability of parallel processing computer in numerical weather prediction. *Current Science*, vol. 74, pp. 1045–1048.

Sundari, M. S.; Vadhiyar, S. S.; Nanjundiah, R. S. (2007): Load Balancing Long Wave Radiation Calculations in a Climate System Model. Technical Report 2007-AS, CAOS, Indian Institute of Science, Bangalore 560012, India, 2007.

Tobis, M.; Schafer, C.; Foster, I.; Jacob, R.; Anderson, J. (1997): Foam: expanding the horizons of climate modeling. In *Supercomputing '97: Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)*, pp. 1–15, New York, NY, USA. ACM Press.

Venkatesh, T. N.; Rajalakshmy, S.; Sarasamma, V. R.; Sinha, U. N. (1998): Scalability of the parallel GCM-T80 Code. *Current Science*, vol. 75, no. 7, pp. 709–712.

Venkatesh, T. N.; Sinha, U. N. (2007): Role of precision in meteorological computing: A study using the nmitli varsha gcm. In *Use of High Performance Computing In Meteorology Proceedings of the Twelfth ECMWF Workshop, Reading, UK 30 October - 3 November 2006*. World Scientific.

Venkatesh, T. N.; Sinha, U. N.; Nanjundiah, R. S. (2001): Building a scalable parallel architecture for spectral gcms. In *Developments in Teracomputing: Proceedings of the Ninth Ecmwf Workshop on the Use of High Performance Computing in Meteorology*, pp. 62–72. World Scientific.

White, A. A.; Bromley, R. A. (1995): Dynamically consistent, quasi-hydrostatic equations for global models with a complete representation of the Coriolis force. *Quarterly Journal of the Royal Meteorological Society*, vol. 121, pp. 399–418.