Design and Parallel Implementation of Two Numerical Methods for Modeling the Atmospheric Circulation

I-Liang Chern
Ian Foster

CRPC-TR91239
October 1991

Center for Research on Parallel Computation
Rice University
P.O. Box 1892
Houston, TX 77251-1892

To appear in Proceedings on "Parallel Computational Fluid Dynamics," Stuttgart, Germany, June 10-12, 1991, Editors: K. G. Reinsch, et al., Elsevier Science Publishers B. V., 1992, pp. 83-96. Also preprint MCS-P264-0991.

To appear in Proceedings of the Conference on "Parallel Computational Fluid Dynamics", Stuttgart, Germany, 10-12 June 1991, Editors: K. G. Reinsch, et al.

Design and Parallel Implementation of Two Numerical Methods for Modeling the Atmospheric Circulation[1]

I-Liang Chern and Ian Foster

Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA

Abstract

We propose two numerical methods suitable for simulating on parallel supercomputers the time evolution of the primitive equations used in atmospheric circulation models. The first is a control volume method on an icosahedral-hexagonal grid on the sphere. This method has a number of computational advantages compared to other methods used for the same purpose, including a nearly uniform resolution on the sphere, no pole problem, and reduced global data communication requirements. The second method is a composite-mesh, second-order Godunov method. This also avoids global communication and the pole problem and, in addition, is able to handle discontinuities. We perform several simulations to demonstrate the convergence of the two methods. We outline the techniques used to achieve parallel implementations and report on computational experiments that demonstrate the methods' suitability for parallel execution.

1 Introduction

The core of a modern climate modeling system is a general circulation model (GCM), a numerical model for simulation of the large-scale flow motions of the atmosphere [14]. A GCM usually consists of two parts: the dynamics and the physical parametrization. The dynamics component employs a numerical algorithm to integrate the governing evolution equations of the atmosphere. The numerical grid scale associated with the algorithm is usually large (hundreds to thousands of kilometers). Sub-grid scale physical processes (e.g., cloud, radiation) are then formulated as parameterized functions of large-scale flow variables. This formulation, called physical parametrization, is more accurate if the grid scale is small. Thus, accurate simulation of both dynamics and sub-grid scale physical processes requires that simulations be performed at high resolutions.

As the performance of massively parallel computers overtakes that of conventional supercomputers, a need arises for new numerical methods and implementation techniques capable of taking advantage of these new computer architectures. This need is particularly acute in climate modeling: parallel computation promises to deliver the order-of-magnitude increases in throughput required for long-term, high-resolution climate studies, but the numerical methods traditionally used in climate models are not obviously well-suited to parallel implementation [3, 9].

[1]This research was supported by the Atmospheric and Climate Research Division and the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Most existing climate models are based on the spectral transform method [1]. The spectral method is popular because of its spectral accuracy and its avoidance of the pole problem. However, the method also has a number of serious problems, including non-local effects, high computational requirements at fine resolutions, and high communication requirements on parallel computers.

Explicit grid-point methods are attractive from the point of view of parallel execution as they require little global communication. Besides, it is much easier to implement the adaptive mesh refinement technique to grid-point methods. However, most grid-point models suffer from the pole problem due to the use of a spherical coordinate system and a uniform latitude-longitude grid.

In this paper, we propose two grid-point methods that avoid the pole problem while retaining the "local communication" property of explicit methods. They are thus suitable for parallel computers. The first is a conservative control volume method on an icosahedral-hexagonal grid. This method avoids the pole problem by the use of the primitive form of the evolution equations in the 3-D Cartesian coordinate system and the quasi-homogeneity of the icosahedral grid on the sphere. The second method is a composite-mesh finite difference method. The pole problem is avoided by the use of the primitive equations in the north/south polar stereographic coordinates and by a shape-preserving interpolation used near the equator. For simplicity, we consider the application of the methods to the shallow water equations on the sphere; however, the methods are directly applicable to the primitive equations used in GCMs. The control volume method is used to perform a Rossby-Haurwitz wave simulation, while the composite mesh method is used to perform a rigid body rotation of a cosine hump over the pole. A detailed description of the control volume method, a brief history of the approach, and several convergence tests are provided in [5].

Parallel implementations of both methods have been developed and ported to a number of parallel computers, including Intel iPSC/860 and Touchstone DELTA, Symult s2010, and Sequent Symmetry. Good parallel performance is achieved on all machines. The parallel codes permit empirical study of parallel performance and scalability issues that will be important when designing future parallel climate models. The parallel codes were developed using PCN, a portable parallel programming system developed at Argonne National Laboratory and Caltech. The PCN system includes a parallel programming language, support for integration of Fortran and C code, and sophisticated profiling tools.

2 Control Volume Method

Finite difference methods on an icosahedral-hexagonal grid have been developed by Sadourny *et al.* [12], Williamson [15], and Masuda and Ohnishi [10]. The new elements introduced here include a symmetrized icosahedral grid, the use of the primitive form of the equations in the 3-D Cartesian coordinate system, a projective bilinear interpolation, and a high-order Gaussian quadrature integration formula. The method uses only a local stencil.

2.1 A Symmetrized Icosahedral Grid

An icosahedral-hexagonal grid is constructed on a sphere from a spherical icosahedron that has 12 nodes and 20 equilateral triangular surfaces. Each spherical triangle is further partitioned into N^2 smaller triangles based on some geodesic arguments. A symmetrization procedure is adopted to make the grid more homogeneous [5]. All points in this constructed grid have six surrounding triangles, except for the 12 principal nodes of the icosahedron, which have only five. At each grid point, we connect by geodesic curves the centroids of these neighboring triangles to give hexagons or pentagons. These hexagons and pentagons are our control area elements for numerical integration.

For convenience in implementation, each triangle is joined with one of its neighbors to form a rhombus; each of these 10 rhombi then contains an $N \times N$ mesh. The two polar points are located in two separate "polar rhombi".

2.2 The Shallow-Water Equations

We avoid the pole problem by expressing the shallow-water equations on the unit sphere in the 3-D Cartesian coordinate system:

$$\mathbf{U}_t + \vec{\nabla} \cdot \mathbf{F} = \mathbf{H_1} + \mathbf{H_2}, \; \vec{x} \in S^2, \tag{2.1}$$

where

$$\mathbf{U} = \begin{bmatrix} \phi \\ \phi\vec{V} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \phi\vec{V} \\ \phi\vec{V}\vec{V} + pI \end{bmatrix}, \tag{2.2}$$

$$\mathbf{H_1} = \begin{bmatrix} 0 \\ -f\vec{x} \times \phi\vec{V} \end{bmatrix}, \mathbf{H_2} = \begin{bmatrix} 0 \\ -\phi|\vec{V}|^2\vec{x} \end{bmatrix}. \tag{2.3}$$

Here $\vec{x} = (x_1, x_2, x_3)$ is the position vector on the unit sphere S^2; ϕ is the geopotential; $p = \phi^2/2$; $\vec{V}$ is the 3-D wind which is tangent to the sphere; and $f = 2\Omega \sin\theta$ is the Coriolis parameter, where Ω is the angular velocity of the earth times the radius of the earth. The term $-\phi|\vec{V}|^2 x$ is the centripetal force to keep the flow on the sphere.

2.3 A Control-Volume Method

Let us consider a control area element A centered at P_0 and surrounded by the geodesics C_i, $i = 1, L$. We first integrate (2.1) without the centripetal force over a control volume element A and apply Stokes's theorem to obtain

$$\iint_A \widehat{\mathbf{U}}_t \, dA = \sum_{i=1}^{L} \int_{C_i} \mathbf{F} \cdot \vec{n} \, ds + \iint_A \mathbf{H}_1 \, dA. \tag{2.4}$$

Here, $\vec{n}$ is the inner normal of the geodesic curve C_i on the sphere, and $\widehat{\mathbf{U}} = (\phi, \hat{\vec{V}})^t$ is an intermediate state. Then, the intermediate velocity $\hat{\vec{V}}$ is projected onto the tangent space of the sphere. This takes care of the contribution of the centripetal force.

Next, we discretize (2.4) using the leap-frog method in the temporal direction and using the midpoint rule for the line integrals in the spatial direction. Let us denote by

$\mathbf{U}^n(P_0)$ the average state of $\mathbf{U}$ at time step n in the control area element A centered at P_0; by M_i the midpoint of the geodesic curve C_i; and by Δt, Δs_i, and ΔA, respectively, the temporal mesh size, the arc length of the geodesic curve C_i and the surface area of A. Then the intermediate state $\widehat{\mathbf{U}}^{n+1} = (\phi^{n+1}, \hat{\vec{V}}^{n+1})^t$ is defined by

$$\begin{aligned}\widehat{\mathbf{U}}^{n+1}(P_0) &= \mathbf{U}^{n-1}(P_0) \\ &\quad + \frac{2\Delta t}{\Delta A}\sum_{i=1}^{L} \mathbf{F}(\overline{\mathbf{U}}^n(M_i)) \cdot \vec{n}|_{M_i}\, \Delta s_i \\ &\quad + 2\Delta t\, \mathbf{H}_1(\mathbf{U}^n(P_0)). \end{aligned} \tag{2.5}$$

The new velocity is obtained by

$$\vec{V}^{n+1} = \hat{\vec{V}}^{n+1} - (\hat{\vec{V}}^{n+1} \cdot \vec{x})\,\vec{x}. \tag{2.6}$$

The mid-states $\overline{\mathbf{U}}^n(M_i)$ are obtained by a 'projected' bilinear interpolation from U^n at the four points P_0, P_i, and $P_{i\pm1}$, where P_i, $i = 1, L$ are the neighboring grid points of P_0 [5].

The following artificial dissipation and time filter are introduced to stablize the scheme. The artificial dissipation $\mu(U^{n-1}(P_i) - U^{n-1}(P_0)) \otimes \vec{n}\,|_{M_i}$, is added to $F(\overline{\mathbf{U}}^n(M_i))$. The time filter is defined by the following formula:

$$\tilde{\mathbf{U}}^n = \mathbf{U}^n + \alpha(\tilde{\mathbf{U}}^{n-1} - 2\mathbf{U}^n + \mathbf{U}^{n+1}). \tag{2.7}$$

Here, $\tilde{\mathbf{U}}$ is the filtered state. The coefficients μ and α are chosen to be 2 and 0.01, respectively.

The method described above is formally second order. However, it becomes first order near the edges of the 20 equilateral triangles due to a slight inhomogeneity of the grid. To obtain a second order scheme on the whole globe while retaining a stencil that uses only nearest neighboring data, we numerically integrate the flux integral $\int_{C_i} \mathbf{F} \cdot \vec{n}\, ds$ by using a two-point Gaussian quadrature rule:

$$\left(w^1 \mathbf{F}(\overline{\mathbf{U}}^n(Q_i^1)) \cdot \vec{n}|_{Q_i^1} + w^2 \mathbf{F}(\overline{\mathbf{U}}^n(Q_i^2)) \cdot \vec{n}|_{Q_i^2}\right) \Delta s_i$$

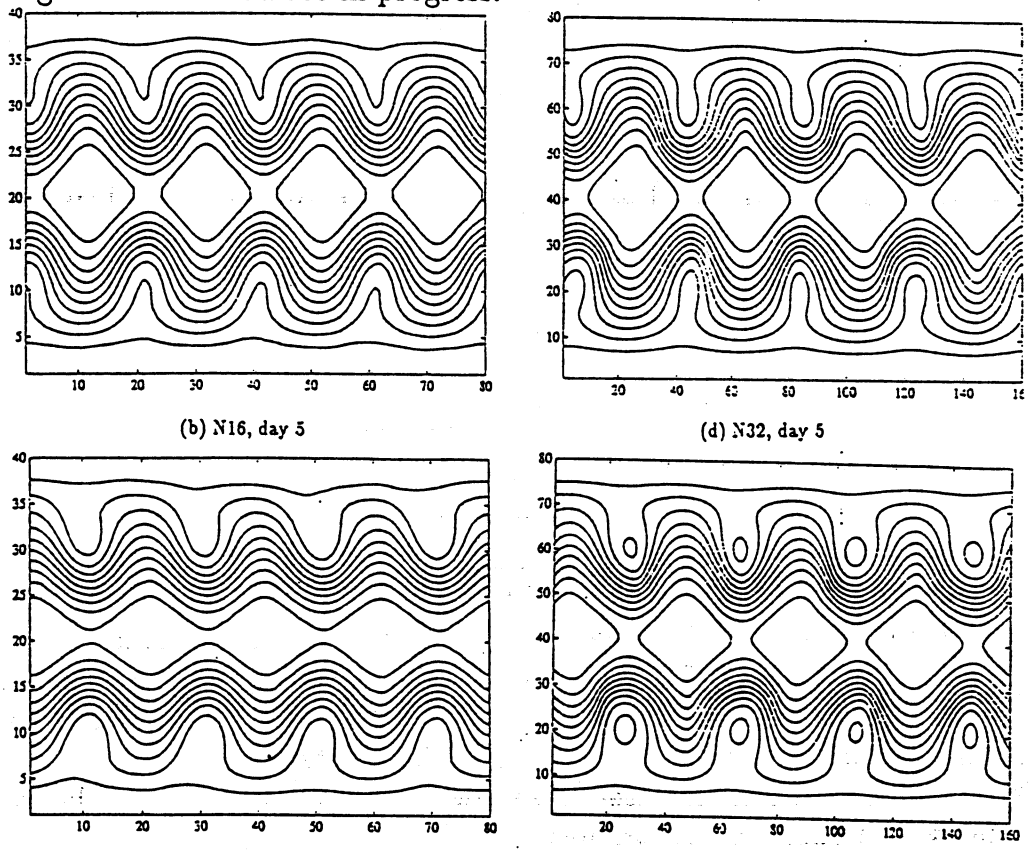
where Q_i^1, Q_i^2 are the two quadrature nodes on the geodesic curve C_i and the weights $w^1 = w^2 = 1/2$. The states $\overline{\mathbf{U}}^n(Q_i^1)$ and $\overline{\mathbf{U}}^n(Q_i^2)$ are obtained by the above projective bilinear interpolation.

2.4 Convergence Tests

The Rossby-Haurwitz waves (R-H waves) are perturbations of a zonal flow by spherical harmonics. Their simulation is one of the standard tests used to validate the correctness of numerical methods for the shallow-water equations on the sphere. The Rossby-Haurwitz waves are given by the stream function [11]

$$\psi = -a^2\omega \sin\theta + a^2 K \cos^R\theta \sin\theta \cos R\lambda, \tag{2.8}$$

where ω, K, and a (radius of the earth) are constants; and $R = 2, 3, \cdots$ is the wave number. The corresponding geopotential ϕ can be found by integrating the equations given above and is given by (38) in Phillips [11]. These waves are stable for $R \leq 5$ and unstable for $R > 5$. We have performed 10-day simulations for the low-order control volume method for $R = 4$ at the two resolutions $N = 16$ and $N = 32$. The Courant number is set to be 0.6, which yields $\Delta t = 10.6$ minutes for $N = 16$ and 5.3 minutes for $N = 32$ for these data. Some of our results are illustrated in Figure 1 (a)-(d); we find that our method is more stable than previous schemes using a similar icosahedral grid (c.f. [17, Figs. 3 and 4], and [16, Figs. 1 and 2]. For the conservative properties of the method, we find that, at day 10, mass is conserved but 1.3% (resp. 0.8%) of total energy and 8.4% (resp. 6.7 %) of total enstrophy are lost with N = 16 (resp. N = 32). More detailed convergence tests for the low-order method are reported in [5]. Convergence tests for the high-order method are in progress.



(b) N16, day 5 (d) N32, day 5

(c) N16, day 10 (e) N32, day 10

Figure 1: Simulated geopotential, Rossby-Haurwitz wavenumber 4

3 Composite-Mesh Method

Composite mesh finite difference methods for climate modeling have been studied by Phillip [11] Stoker and Isaacson [13], and Browning [2]. The method proposed here is an extension of Browning's. The new elements introduced here are a high-order, shape-preserving interpolation and a second-order Godunov finite difference method for the

purpose of handling sharp gradients.

3.1 The Coordinates and the Equations

We use the following polar stereographic coordinate systems:

$$\begin{aligned} X &= m\cos\theta\cos\lambda, \\ Y &= m\cos\theta\sin\lambda, \end{aligned}$$

where λ is the longitude, θ is the latitude, and $m = 2(1+q\sin\theta)^{-1}$ is the map factor. For the northern hemisphere projection $q = 1$, while for the southern hemisphere projection $q = -1$.

The shallow water equations in these coordinate systems in the flux form are

$$\frac{\partial}{\partial t}(m^{-2}\mathbf{U}) + \frac{\partial}{\partial X}(m^{-1}\mathbf{F}) + \frac{\partial}{\partial Y}(m^{-1}\mathbf{G}) = m^{-2}\mathbf{H}$$

$$\mathbf{U} = \begin{bmatrix} \phi \\ \phi U \\ \phi V \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \phi U \\ \phi U^2 + p \\ \phi UV \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \phi V \\ \phi UV \\ \phi V^2 + p \end{bmatrix},$$

$$\mathbf{H} = qf\phi \begin{bmatrix} 0 \\ V \\ -U \end{bmatrix} - \phi(XV - YU)/2 \begin{bmatrix} 0 \\ V \\ -U \end{bmatrix} - p/2 \begin{bmatrix} 0 \\ X \\ Y \end{bmatrix}$$

Here, (U, V) are velocity components in the stereographic coordinate, and $p = \phi^2/2$.

3.2 Grid, Interpolation, and Method

We use a uniform grid on the planes tangent to the sphere at the north and south poles respectively and with the unit sphere projected onto these planes by the polar stereographic projections. The actual computational grid on each plane extends slightly beyond the equator for stability reasons. The boundary data on each plane are obtained by interpolation of data from the other plane. We use a two-dimensional, shape-preserving, Hermite cubic interpolant [18], where the shape-preserving derivative constraint is the one proposed by de Boor and Swartz [7], and a tensor product formula is used to extend a one-dimensional interpolant to a two-dimensional interpolant.

For the finite difference method, we use Strang's splitting formula by solving the following two one-dimensional problems

$$\frac{\partial}{\partial t}(m^{-2}\mathbf{U}) + \frac{\partial}{\partial X}(m^{-1}\mathbf{F}) = m^{-2}\mathbf{H}_1 \tag{3.9}$$

$$\frac{\partial}{\partial t}(m^{-2}\mathbf{U}) + \frac{\partial}{\partial Y}(m^{-1}\mathbf{G}) = m^{-2}\mathbf{H}_2 \tag{3.10}$$

A second-order Godunov method is adopted to solve the above equations. The ingredients of this method in our application are a piecewise linear reconstruction, the von Leer limiter [6], and a generalized Riemann solver. The generalized Riemann solver provides an approximate solution to (3.9) or (3.10) with a piecewise linear initial data. A detailed description of this solver will be reported in a future paper.

3.3 Test of a Rigid Body Rotation

To study the convergence of this method, a preliminary test is the following rigid body rotation test [19]. We consider the linear advection equation

$$u_t + \vec{V} \cdot \nabla u = 0$$

where $\vec{V}$ is a tangent vector field on the sphere generated by a rigid body rotation. The initial data is a cosine hump on the equator. Figure 2 shows the plots of the hump at the initial time and at the time after one cycle rotation. We see almost no error in the shape and location and only a small amount of dissipation. A more detailed convergence test is under way.

Figure 2: Rigid body rotation of a cosine hump, (a) initial time, (b) after one cycle rotation

4 Parallel Implementations

We have devoted considerable effort to implementing the two numerical methods on parallel computers, both because we feel that the development of the necessary parallel algorithms is of interest in its own right, and because we needed a basis for empirical evaluation of the parallel scalability of the two methods.

Out parallel implementations are designed to execute on parallel computers in which processors communicate by sending messages through a communication network (multicomputers). We choose to work with such computers both because they appear to be more scalable than shared memory computers, and because the message-passing model is easily emulated on shared memory machines.

Both the control volume and composite mesh methods are explicit methods: that is, the computation performed to update the state of a grid point requires only the value of that grid point and a small number of "near neighbors". In general, parallel algorithms for explicit methods are obtained by partitioning the data domain into disjoint subdomains (the partition being chosen to minimize communication requirements), organizing communication between subdomains to communicate data from "near neighbors", and

mapping the subdomains to processors in a parallel computer in a way that maximizes parallel efficiency.

We apply this general strategy when developing parallel algorithms for the two methods. Partitioning is straightforward in both cases, but communication and mapping are complicated by inhomogenieties in the problem domains and, in the composite mesh method, interpolation between the north and south mesh. We find that some care is needed if good performance is to be achieved.

The parallel performance of explicit codes is often constrained by the need to perform a global reduction operation at each step in order to compute a time step that does not violate the CFL condition. However, as the maximum speed of an atmospheric flow tends not to vary greatly over time, it is common in atmospheric modeling to adopt a fixed time step and then verify that this is valid at each step. This approach is adopted here.

A novel feature of our approach is our use of the high-level parallel programming system PCN [4] to implement the parallel algorithms, rather than low-level send/receive primitives as is common in scientific programming. We report on this aspect of our work in some detail, as it may be of interest to others developing similar codes.

PCN provides a number of features which simplify the task of developing high performance parallel programs. These include: explicit support for parallelization of existing code written in C or Fortran; integrated profiling tools; a high-level concurrent language for specification of concurrent algorithms; separate specification of partition and mapping; and portability across a range of different machines [8]. The portability of PCN programs is particularly useful: for example, it proved possible to develop and refine our parallel codes on a Sun workstation and then to execute the resulting codes on a variety of parallel machines without modification.

4.1 Control Volume Method

Partition. Recall that the icosahedral mesh can be considered as consisting of ten equi-sized rhombi and two polar points. A natural partition decomposes each non-polar rhombus into a number (say C^2) of subrhombi, giving a total of $10C^2 + 2$ subdomains, $10C^2$ containing $(N/C)^2$ points (where N^2 is the total number of points in a rhombus) and the two polar rhombi containing one point each.

To give an indication of how PCN is used to specify the concurrent aspects of the control volume code, we sketch in Figure 3 the PCN code that implements this decomposition. The PCN procedure `sphere` comprises a set of arguments (`c,n,...`) representing the size of the problem, etc., and a parallel block (represented as {|| `p1, ..., pn`}) indicating that procedures `mesh(1,c,n,...)`, ..., `pole(11,n,...)` are to be executed concurrently. When executed, the `sphere` procedure creates twelve concurrently executing processes representing the ten rhombi and two poles. In a similar fashion, the `mesh` procedure creates C^2 subrhombi processes to handle computation within a single rhombus.

Communication. All communication in this code is nearest neighbor and results from the use of a six-point stencil in the numerical method. Inhomogeneities at the poles and between adjacent sub-rhombi from different rhombi lead to irregularities in this communication. However, it turns out that each subrhombi requires values from either five or six

```
sphere(c,n,...)
{|| mesh(0,c,n,...),
    mesh(1,c,n,...),
    mesh(2,c,n,...),
    mesh(3,c,n,...),
    mesh(4,c,n,...),
    mesh(5,c,n,...),
    mesh(6,c,n,...),
    mesh(7,c,n,...),
    mesh(8,c,n,...),
    mesh(9,c,n,...),
    pole(10,n,...),
    pole(11,n,...)
}

mesh(me,c,n,...)
{|| i in 1..c, j in 1..c :  submesh(me,i,j,n,...)}
```

Figure 3: Top-level PCN code

other subrhombi. The communication channels used to perform the necessary exchanges are specified by introducing shared variables into the code in Figure 3.

Mapping. The mapping problem is defined as follows: allocate subdomains identified by the partition to processors in a way that achieves minimal run-time. This minimization involves avoiding load imbalances and communication costs. In the present instance, determination of an optimal mapping is complicated by the spherical problem domain and the irregular communication patterns. However, we have found that even simple, non-optimal mappings give reasonable parallel efficiencies on multicomputers such as the Symult s2010 and the Intel Touchstone Delta, due to the high communication performance of these machines and their use of cut-through routing.

PCN allows us to separate the tasks of (a) mapping our application to an abstract process structure and (b) mapping this process structure to a parallel computer. The first, application-specific mapping is expressed in terms of an abstract, machine-independent structure consisting of "rhombi" and "poles"; the mapping of this abstract structure to a parallel computer is specified by an application-independent (but machine-dependent) library. We illustrate this approach by showing in Figure 4 an annotated version of the code fragment given above. The annotations (`@rhombus(0)`, etc.) alter the mapping and hence the parallel performance of the code but not the result computed.

The use of an abstract mapping structure provides portability: the parallel code can be ported to a different architecture (e.g., a hypercube instead of a mesh) simply by providing a library that embeds the abstract structure in the new architecture. It also

```
sphere(c,n,...)
{|| mesh(0,c,n,...)@rhombus(0),
   mesh(1,c,n,...)@rhombus(1),
   mesh(2,c,n,...)@rhombus(2),
   mesh(3,c,n,...)@rhombus(3),
   mesh(4,c,n,...)@rhombus(4),
   mesh(5,c,n,...)@rhombus(5),
   mesh(6,c,n,...)@rhombus(6),
   mesh(7,c,n,...)@rhombus(7),
   mesh(8,c,n,...)@rhombus(8),
   mesh(9,c,n,...)@rhombus(9),
   pole(10,n,...)@north_pole,
   pole(11,n,...)@south_pole
}

mesh(me,c,n,...)
{|| i in 1..c, j in 1..c :  submesh(me,i,j,n,...)@{i,j} }
```

Figure 4: Top-level code with mapping annotations

simplifies the exploration of alternative mappings. For example, it is likely that on larger meshs performance can be enhanced by a careful mapping of communicating subrhombi to nearby processors. This can be achieved by changing a library, not the application code.

Performance. The parallel code has been executed on a Sun-4 workstation, a 24-node Sequent Symmetry shared memory computer, a 192-node Symult s2010 mesh, a 64-node Intel iPSC/860 hypercube, and the 520-node Intel Touchstone DELTA system. Only preliminary performance results are available at the time of writing; evaluation and tuning are ongoing.

More-or-less perfect scaled speedups are achieved on each of the parallel machines. That is, execution time per step stays approximately constant if the number of grid points is scaled linearly with the number of processors. This is to be expected, as the communication/computation ratio stays constant if the problem size is scaled with the number of processors.

Preliminary non-scaled results for the parallel code are presented in Figure 5. This shows performance as a function of number of processors, expressed in terms of time per step, for a problem size of N = 56 (approx. 150 km resolution). Reasonable parallel efficiencies are achieved on all four parallel machines. On the DELTA, parallel efficiency is about 70% on 492 processors (C=7).

The PCN system includes a sophisticated set of profiling tools. An integrated profiling module automatically collects performance information each time a program executes.

Graphical tools allow this data to be examined on a global, per-node, and per-procedure basis. These facilities allow us to gain an understanding of the issues determining parallel performance and to detect performance "bugs" that might otherwise be difficult to locate.

An example display is shown in Figure 6. This is a summary picture of a run on 492 nodes of the Intel Touchstone DELTA, in which each horizontal line is a processor and shading distinguishs time spent idle (light), busy (dark), and communicating (white). We immediately see that much of the 30% parallel inefficiency observed on the DELTA is due to load imbalance: the processors handling location (0,0) in each rhombus are spending more time computing than other processors. The reason for this imbalance has not yet been determined. One possibility is that some operations in this region are generating denormalized numbers; the i860 processor used in the DELTA traps operations on such values to software handlers, at a cost of many tens of instructions.

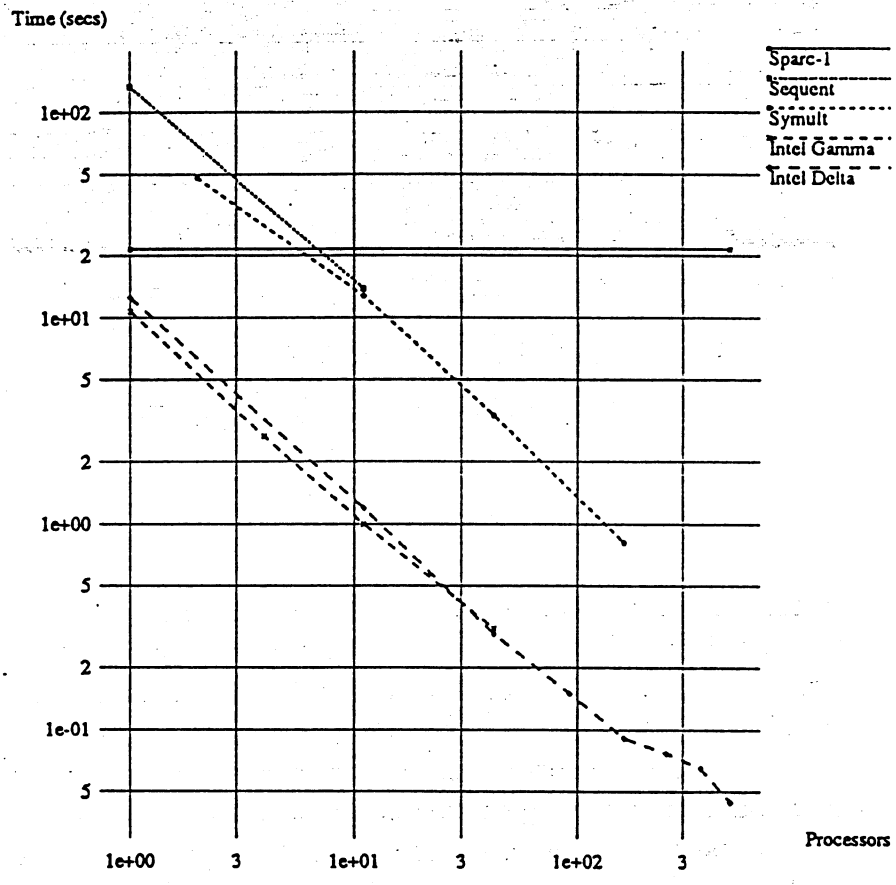


Figure 5: Parallel Performance

4.2 Composite Mesh Method

The issues that must be addressed when developing a parallel algorithm for the composite mesh method are similar to those encountered in the control volume code; hence, we will not discuss them in detail. We partition the computation by using a straightforward domain decomposition: the north and south grids are each divided into C^2 disjoint, equi-sized subgrids or *charts*. Communication requirements are a combination of boundary data exchanges between neighboring charts within the north and south meshs and

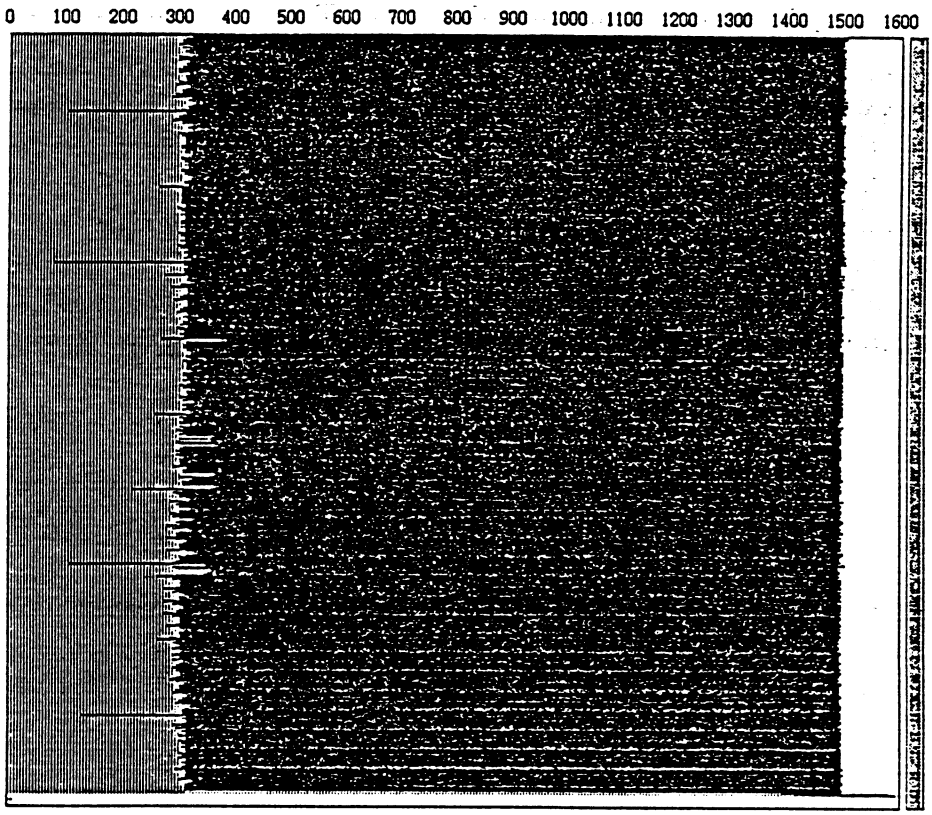


Figure 6: PCN performance display: time breakdown

interpolation data exchanges between charts in different meshs.

Mapping is complicated by the fact that the amount of computation in a chart depends on the proportion of its area covered with computational points. Hence, is is sometimes advantageous to map several charts to a single processor.

Preliminary experiments show almost linear speedups on a moderate number of processors. A load balancing strategy is employed to balance the amount of computation performed per processor: we generate more charts than processors and use a bin-packing algorithm to allocate charts to processors. This strategy improves performance by between five and ten per cent, depending on problem size and number of processors.

5 Comparison with Spectral Transform

Our investigation of alternative numerical methods is motivated in part by the elevated computational and communication requirements of the spectral transform method, especially at high resolutions [9]. Hence, we are interested in understanding how the icosahedral, composite mesh, and spectral transform methods compare from the point of view of parallel performance. We outline here the results of a preliminary comparison of the

spectral transform and icosahedral methods.

In comparing the two methods, we assume that the semi-implicit time stepping employed in spectral transform methods allows a time step four times longer than in the fully explicit icosahedral method. We adjust computational requirements to allow for the greater uniformity of the icosahedral grid, but do not take into account the benefits of spectral accuracy.

Analytic and empirical studies (of the kind reported in [9]) suggest that the icosahedral method is consistently faster for all problem sizes on medium-grained parallel computers (e.g., Intel iPSC/860 and Touchstone DELTA). At low resolutions, the spectral method suffers from high communication costs; at high resolutions, its n^3 computation costs dominate execution time. In addition, the icosahedral method does proportionally better if either the number of processors or communication costs are increased.

These results are encouraging. However, although the icosahedral method is always better, it is only *much* better on very large problems or computers. The reason that it is not more competitive in other situations is the excessively small time step enforced by its purely local communication. Semi-Lagrangian methods may be a solution to this problem.

References

[1] Bourke, W., An efficient, one-level, primitive-equation spectral model, *Mon. Wea. Rev.*, **102**, 687–701, 1972.

[2] Browning, G., Hack, J., and Swarztrauber, P., A comparison of three numerical methods for solving differential equations on the sphere, *Mon. Wea. Rev.*, **117**(5), 1058–1075, 1989.

[3] U.S. Department of Energy, Building an Advanced Climate Model: Program Plan for the CHAMMP Climate Modeling Program, Publication DOE/ER-0479T, 1990 (Available from National Technical Information Service).

[4] Chandy, K.M., and Taylor, S., *An Introduction to Parallel Programming*, Jones and Bartlett, 1991.

[5] Chern, I., A control volume method on an icosahedral grid for numerical integration of the shallow-water equations on the sphere, MCS Preprint MCS-P214-0291, Argonne National Laboratory, 1991.

[6] Colella, P. and Woodward, P., The piecewise parabolic method (PPM) for gas-dynamical simulations, *J. Comp. Phys.*, **54**, 174–201, 1984.

[7] de Boor, C., and Swartz, B., Piecewise monotone interpolation, *J. Approx. Theory*, **21**, 411–416, 1977.

[8] Foster, I., Kesselman, C., and Taylor, S., Concurrency: Simple concepts and powerful tools, *The Computer Journal*, Dec. 1990.

[9] Foster, I., Gropp, W., and Stevens, R., The parallel scalability of the spectral transform method, *Mon. Wea. Rev.*, (to appear).

[10] Masuda Y., and Ohnishi, H., An integration scheme of the primitive equation model with an icosahedral-hexagonal grid system and its application to the shallow-water equations, *Short- and Medium-Range Numerical Weather Prediction, WHO/IUGG NWP,* Tokyo, 317–326, 1986.

[11] Phillips, N., Numerical integration of the primitive equations on the hemisphere, *Mon. Wea. Rev.,* **87**(9), 333–345, 1959.

[12] Sadourny, R., Arakawa, A., and Minitz, Y., Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere, *Mon. Wea. Rev.*, **96**(6), 351–445, 1968.

[13] Stoker, J., and Isaacson, E., Final report 1, Courant Institute of Mathematical Sciences, IMM 407, New York University (1975), 73 pp.

[14] Washington, W., and Parkinson, C., *An Introduction to Three-Dimensional Climate Modeling*, University Science Books, 1986.

[15] Williamson, D., Integration of the barotropic vorticity equation on a spherical geodesic grid, *Tellus,* **10**, 642–653, 1968.

[16] Williamson, D., Integration of the primitive barotropic model over a spherical geodesic grid, *Mon. Wea. Rev.,* **98**, 512–520, 1969.

[17] Williamson, D., A comparison of first- and second-order difference approximations over a spherical geodesic grid, *J. Comp. Phys.*, **7**(2), 301–309, 1971.

[18] Williamson, D., and Rasch, P., Two-dimensional semi-Lagrangian transport with shape-preserving interpolation , *Mon. Wea. Rev.* **117**(1), 102–129, 1989.

[19] Williamson, D., Drake, J., Hack, J., Jacob, R., and Swarztrauber, P., A standard test set for numerical approximations to the shallow water equations on the sphere, Tech. Report, NCAR, 1991.