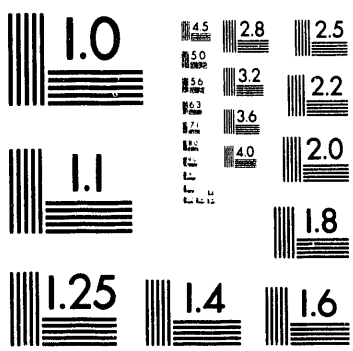

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1

1 of 1

Conf-9307111--2

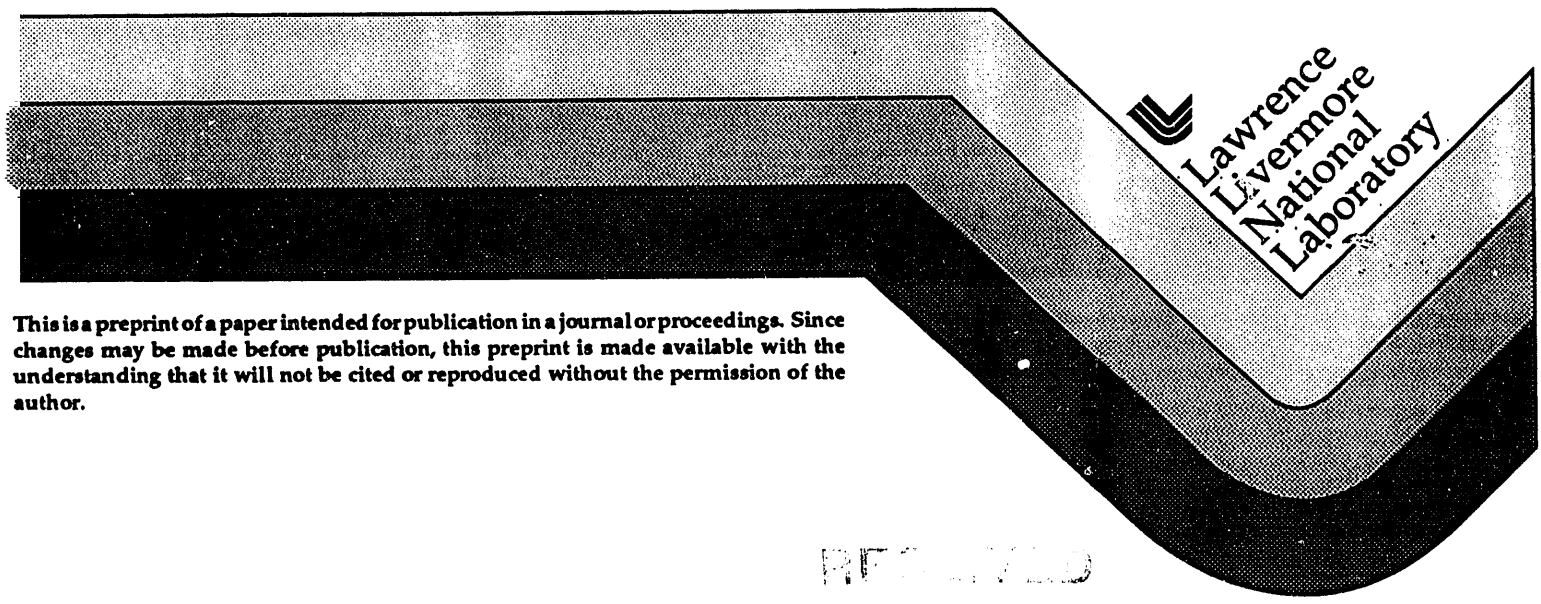
UCRL-JC-113039
PREPRINT

# Toward a High Performance Distributed Memory Climate Model

M.F. Wehner, J.J. Ambrosiano, J.C. Brown, W.P. Dannevik,
P.G. Eltgroth, A.A. Mirin, J.D. Farrara, C.C. Ma,
C.R. Mechoso, J.A. Spahr

This paper was prepared for submittal to the
Second International Symposium on High Performance Distributed Computing
Spokane, WA
July 21-23, 1993

February 15, 1993



# Toward a High Performance Distributed Memory Climate Model

M.F. Wehner, J.J. Ambrosiano, J.C. Brown, W.P. Dannevik, P.G. Eltgroth, A.A. Mirin
Lawrence Livermore National Laboratory
Livermore, California 94550
and
J.D. Farrara, C.C Ma, C.R. Mechoso, J.A. Spahr
Department of Atmospheric Sciences
University of California, Los Angeles 90024-1565

As part of a long range plan to develop a comprehensive climate systems modeling capability, we have taken the Atmospheric General Circulation Model originally developed by Arakawa and collaborators at UCLA and have recast it in a portable, parallel form. The code uses an explicit time-advance procedure on a staggered three-dimensional Eulerian mesh. We have implemented a two-dimensional latitude/longitude domain decomposition message passing strategy. Both dynamic memory management and interprocessor communication are handled with macro constructs that are preprocessed prior to compilation. The code can be moved about a variety of platforms, including massively parallel processors, workstation clusters, and vector processors, with a mere change of three parameters. Performance on the various platforms as well as issues associated with coupling different models for major components of the climate system are discussed.

## 1 Introduction

### 1.1 The Need to Understand and Predict Climate Change

Since the Industrial Revolution, humankind has inadvertantly mounted a grand-scale and largely uncontrolled experiment on the earth's global environmental systems. The growing consumption of fossil fuels, deforestation, and other human activities are increasing the atmospheric concentration of gases such as carbon dioxide, methane, nitrous oxide, and chlorofluorocarbons. Along with water vapor, these "greenhouse" gases are major factors in the radiation budget of the atmosphere, which in turn plays a critical role in controlling the earth's climate. Under a variety of reasonable growth scenarios for population and industrial activity, the increasing concentration of greenhouse gases over the next few decades could result in a radiative forcing equivalent to that produced by a doubling of preindustrial levels of carbon dioxide.

While it is likely that this process will eventually affect global climate, we are far from being able to predict accurately the timing, magnitude, and regional pattern of such changes. Attempts to alter the present global system of energy production to effectively counter a possible world-wide temperature rise could incur enormous social and economic costs. However, the effects of global climate change resulting from an unchecked temperature increase could prove equally costly. For these reasons it is vitally important to improve the scientific basis for understanding and predicting the response of the earth's environmental systems to substantial perturbations, both natural and human-induced.

### 1.2 The Need for Models and High-Performance Computing

The inadvertant experiment described above is difficult to interpret for at least two reasons. First, there is no "control" experiment. The record of the unperturbed climate system is too sparse and unreliable to provide a comparison or reference dataset, especially as other factors also affect climate. Second, our knowledge of the magnitude and spatial pattern of perturbing influences is inadequate. For example, the time histories of the concentrations of greenhouse gases, the amount of volcanic aerosols aloft, and the record of solar radiation are all poorly known. In addition, only about half of the anthropogenic emissions of carbon dioxide increase its concentration in the atmosphere.

We know little about what happens to the remaining half, particularly with respect to how the global carbon dioxide sink is partitioned between the ocean and terrestrial biosphere.

All these uncertainties, combined with the practical impossibility of comprehensive laboratory-based climate experiments, have created a unique and vital role for computer-based climate simulation experiments. By enabling climate scientists to quantify their models and to test them in detail against a wide array of observational datasets, such experiments can provide an objective basis for understanding the climate system and for estimating future changes.

The bewildering diversity of physical processes that together determine a given climate regime occurs over a wide range of space and time scales that are activated by nonlinear couplings. Models that attempt to encapsulate these processes are necessarily complex and overtax the capabilities even of current supercomputers.

This disparity between computer requirements and capabilities will become more acute with the next generation of models. Current estimates of climatic change are often based on atmospheric models run with limited chemical reactions, simplified oceans, and essentially no representations of terrestrial and marine biosystems. Such limitations, together with the delayed response currently evident in the global temperature record, point clearly to the need for more comprehensive and better-tested models. Future models of the climate system will need increased spatial resolution, enhanced representation of processes, and full coupling of additional major components of the earth system, including dynamic representation of the oceans, ecosystems, and interactive biogeochemistry.

It is clear that substantially faster algorithms and increased throughput are imperative if we are to move forward to advanced simulations of the climate system during this decade. Parallel processing offers significant potential for attaining increased performance if the basic algorithms can be recast in a suitable parallel form.

## 2 The UCLA Atmospheric General Circulation Model

The general circulation of the atmosphere is one of the major component subsystems of the earth's climate system. Atmospheric General Circulation Models (AGCMs) solve a set of primitive equations to simulate the long-term (climatic) behavior of the atmosphere. They generally fall into two categories defined according to the algorithms used to solve the hydrodynamical equations of motion. The more conventional of the two is characterized by a grid point discretization. An alternative method of solution is provided by a spectral transform method using a spherical harmonic approximation to calculate certain horizontal derivatives. At currently attainable horizontal grid resolutions, spectral methods have an edge over grid point methods on traditional vector supercomputers. It is felt by many that at the desired higher resolutions this situation will reverse. Additionally, spectral methods require numerous Fourier transforms per time step. Hence on distributed memory computer systems, large amounts of data must be communicated between processors. As is well known, parallel algorithms generally are limited by the time spent communicating data relative to that spent actually calculating data. For these and additional reasons, we have chosen to implement a grid-point-based AGCM.

The AGCM developed by Arakawa and coworkers at UCLA beginning in the mid-1960's [1] is such a grid point model. The current version of the UCLA AGCM incorporates advanced finite differencing techniques [1, 3], state of the art parameterizations of physical processes, and has been extensively tested and used to study a variety of climate problems [12, 13]. In a separate study, the UCLA AGCM has been coupled to the NOAA Geophysical Fluid Dynamics Laboratory / Princeton Universtity Modular Ocean Model (MOM) for investigations on the seasonal cycle and interannual variability of the atmosphere-ocean system [15]. Also underway at UCLA is addtional study of the distribution of the coupled AGCM/MOM across high speed networks [14]. We have recast an existing vectorized version in a portable, parallel form. We have also extended the model to include a number of additional atmospheric processes and have imbedded the AGCM into a framework intended to allow coupling to the other major sub-models (ocean, biosphere, atmospheric chemistry, etc.) defining an earth system model (ESM).

Atmospheric general circulation models typically consist of two parts. The first part is the solution of the equations of hydrodynamics suitably approximated to the regimes encountered by the atmosphere. This typically involves a straightforward derivation beginning with the Navier-Stokes equations. The second part is a collection of physically based parameterizations of the other processes important to the simulation of the atmosphere. These process parameterizations are generally considerably less connected

to physical "first principles" than the hydrodynamics due to the complexity of the processes involved. In some cases, a lack of understanding of the details of the relevant physical mechanisms forces these parameterizations to be oversimplified.

The hydrodynamics portion of the UCLA AGCM is a grid point based explicit finite difference model. The horizontal differencing scheme is designed to conserve the potential enstrophy, the global integral of the potential vorticity, as well as the total energy [2, 4]. The solution of the inertial terms of the momentum conservation equation is fourth order. Other terms of this equation and the continuity equation are solved to second order in the spatial dimensions. The horizontal advection of the scalar quantities is also fourth order accurate in the spatial dimensions.

The vertical differencing scheme under the hydrostatic approximation reduces the problem from fully three-dimensional equations to a set of coupled two-dimensional equations closely related to the shallow water equations. From these, the velocity components and the pressures are determined. All differencing schemes in the vertical dimension are second order accurate.

The thermodynamic energy equation, formulated in terms of the potential temperature, describes the variation of the internal energy in both the vertical and horizontal directions. Advection of moisture (formulated in terms of specific humidity) and ozone concentration are performed. The ideal gas law provides a closure to solve for the density, which also varies in each of the directions. This then allows the pressure gradient force to vary in the vertical dimension even within a finite difference cell.

The finite difference mesh is staggered in both the vertical and horizontal directions. In the horizontal direction, the Arakawa C-mesh in latitude/longitude coordinates is used because of its superior properties for large scale atmospheric motion [1]. In the vertical direction, the thermodynamic variables and the (derived) vertical velocity are staggered. The net result in three dimensions is a cell shaped as a cube in spherical geometry with the velocity components centered on each of the faces and the thermodynamic variables (potential temperature, pressure, specific humidity, ozone, etc.) at the cell center.

The vertical domain is divided into three distinct regions. These are the stratosphere, the troposphere and the planetary boundary layer (PBL). The vertical coordinate is a modified $\sigma$ coordinate system with $\sigma = -1$ at the top of the atmosphere, $\sigma = 0$ at the tropopause, $\sigma = 1$ at the troposphere-PBL boundary, and $\sigma = 2$ at the earth's surface [5]. The PBL is represented by the lowest model layer, and the free atmosphere-PBL boundary is considered as a material boundary surface with no mass or momentum advected through it via the hydrodynamic equations. The only exchange between these layers occurs via parameterized source terms depending on model specific physics. The top and bottom of the model are also considered as material surfaces, with no exchanges across them except for parameterized fluxes of momentum, heat and moisture. The horizontal boundary condition is that of periodicity in the longitudinal coordinate. In the latitudinal coordinate, a regularity condition is imposed at each of the poles.

The second portion of the UCLA AGCM, the parameterized physical processes, consists of several subprocesses that are operator split. These are a layer cloud instability calculation at the troposphere-PBL boundary[5], cumulus convective transport of energy, momentum and moisture [6], moist convective instability adjustment [5], long wave (infrared) radiation transport [7], short wave (visible and UV) radiation transport, surface fluxes of momentum, moisture and energy to the PBL [5], the evolution of the ground temperature and snow cover [8], and simplified ozone photochemistry[9]. All of these subprocesses involve quantities only from a single horizontal mesh cell. However, because of the staggered mesh, these quantities are not all at the same physical location. Instead, quantities relating to the thermodynamic state are cell-centered and those relating to the hydrodynamic state are face-centered. Typically, no further information for these parameterized processes is needed in the horizontal dimensions other than the horizontal velocity components located on each of the four faces of the cell. In the vertical direction, however, some of these processes require an implicit solution. Hence, these subprocesses are collectively described as the "column physics" modules to denote their dependence on a vertical column of air.

The time differencing of the AGCM equations is mostly explicit. Because of the substantial computational burden of the column physics, the hydrodynamics is subcycled with respect to it. The time step governing the hydrodynamics is considerably more restrictive than that required by the more slowly varying column physics processes. In explicit hydrodynamics schemes, the time step is usually governed by the Courant-Friedrich-Levy (CFL) condition in order to ensure stability. In a spherical coordinate system, the longitudinal spacing between cell centers converges as points approach the polar singularity. Hence, in a

grid point model with a fixed time step scheme, the time step is usually determined by the cells nearest the poles. For a simulation of the atmospheric dynamics, this extra resolution near the poles is physically unnecessary since the accuracy of the calculation is typically determined by the cells near the equator. Various attempts have been made to overcome this restrictive time step. In the UCLA AGCM, Arakawa and Lamb [1] developed a set of discrete Fourier filters specifically designed to damp unstable modes which may arise when violating the CFL condition in the vicinity of the poles. These filters contain a latitudinal dependence but are applied over the complete longitudinal domain. This allows a choice of hydrodynamic time step in violation of the CFL condition in the higher latitudes but consistent with it near the equator. A typical operational time step in a 4° by 5° calculation is one hour for the column physics and 7.5 minutes for the hydrodynamics. It is desirable to reduce the column physics time step somewhat since some of these processes vary on a faster time scale. However, as the horizontal resolution is increased, this time step may not need to be decreased further since the scaling of these processes is essentially determined by the vertical structure of the model. This is not the case for the hydrodynamics, as the CFL condition still determines the time step.

## 3 Parallelization Strategy

### 3.1 Domain Decomposition Methodology

The parallel AGCM is designed for a distributed memory multiple-instruction-multiple-data (MIMD) computing environment. A two-dimensional latitude/longitude domain decomposition method is used, so that each subdomain actually consists of a number of contiguous vertical columns extending from the earth's surface into the upper atmosphere (see Fig. 1). The choice to decompose in only two dimensions is based on the fact that column processes strongly couple the elements within the column and do not naturally parallelize along the column. Additionally, the number of meshpoints in the vertical direction is usually small. We choose the two dimensional decomposition over a one dimensional decomposition because the latter does not provide the necessary concurrency for mesh sizes of interest and has an asymptotically larger communications cost[10].

Subdomains are assigned to processors in a deterministic manner and data are transmitted between subdomains in the form of messages. In order to calculate the fourth order solution of the horizontal part of the hydrodynamics [4], two points on each of the four sides of a cell are required. For cells in the interior of a subdomain, this data is readily available in local memory. For cells bordering on the edge of a subdomain, some of this data is contained in a neighboring subdomain and hence, on a different processor's local memory. This is the data that must be communicated.

The structure of a subdomain is rectangular (in logical space) with an inner rectangle consisting of the cells to be calculated by the assigned processor and a border or frame of cells into which data will be transmitted from the neighboring subdomains. These "phony" cells do not have their solutions advanced in the hydrodynamics portion of the calculation. Rather, they may be thought of as the implements of interface conditions between subdomains. At the physical boundaries, these phony cells provide the imposed physical boundary conditions instead. For the fourth order scheme, this border is two cells wide in order that the last "real" cell may be calculated. We have made a conscious effort to maintain a uniformity of the subdomains. Since the subdomain interface and imposed boundary conditions are used at the same point in the calculation, we have written a subroutine that determines which of these two cases must be invoked. In the subroutines that difference the equations, no knowledge of the physical location of the subdomain is necessary. In this manner, we have been able to use the same code on multiple and single subdomain decompositions, allowing us access to traditional single processors computers for purposes of debugging and comparison.

Communication of any particular quantity is required only once per time step due to the explicit time differencing of the algorithm. Communication is accomplished in two steps. Data is first exchanged between subdomains bordering each other at a given latitude. This is followed by an exchange between subdomains bordering each other at a given longitude. Diagonal communication of data at the extreme corners of the subdomain frame is thus implicitly accomplished without further messages.

The communications described thus far are necessary to implement calculation of difference approximations to horizontal derivatives for the hydrodynamics These communications require that messages be sent only to each of a subdomain's four nearest neighbors. There are two other parts of Arakawa's algorithm that require interprocessor data communication. Both of these are global in the longitudinal direction. The

simpler of the two is a calculation of the vorticity on the grid points nearest to the polar singularity. Because of the staggered mesh, this is accomplished via a circulation theorem that requires a knowledge of the latitudinal velocity component at each of the points encircling the pole. Since this is needed at only a single latitude line for each hemisphere, the amount of data communicated is not large. A larger and more complex communication procedure is required to implement the discrete Fourier filters used to increase the time step. Calculation of these filter functions involves the convolution of two vectors about the entire longitudinal domain. The filtering of a given variable takes the form

$$\phi'(i) = \sum_{i'=1}^{M} S(i-i')\phi(i') \ . \qquad (1)$$

Here, $S$ is a fixed set of coefficients, $M$ is the number of longitudinal cells and $\phi$ is the vector to be smoothed. For each grid point, a sum involving some combination of each of the elements of both of these vectors is required. For the fixed coefficient vector, we copy every longitudinal element for the range of latitude lines contained in the subdomain into the local memory of that subdomain's processor at problem generation. To calculate the actual summations, partial sums of the inner product of $S$ and the portion of $\phi$ contained in the subdomain are first calculated. These partial sums are then communicated via messages to the appropriate subdomain and the summation is completed. The resulting equation is

$$\phi'(i) = \sum_{n=1}^{n_p}\Big(\sum_{i'=L_n}^{U_n} S(i-i')\phi(i')\Big) \ . \qquad (2)$$

Here, $L_n$ and $U_n$ are the lower and upper longitudinal elements of the subdomain $n$. For each subdomain, the inner sum is calculated for all values of $i = 1, ..., M$ and hence uses all the elements of the fixed coefficient vector, $S$. The partial sums are communicated by messages from each subdomain to all the other subdomains (at that latitude), requiring of order $N$ messages where $N$ is the number of longitudinal subdomains. Alternatively, a tree-based communications pattern would allow a reduction to of order $\log N$ messages. We are also investigating equivalent filter functions having a limited rather than global extent to further reduce communication cost.

The filtering procedure can seriously affect overall performance on distributed memory computers. There are several issues. The filters are constructed to be latitude dependent. This means that there is a load imbalance between subdomains containing high latitudes and those near the equator. This load imbalance is caused by both the communication and the extra computation required to perform the summations. This is further accentuated by the use of two types of filters, "strong" and "weak". Under the operational conditions described above, the weak filter operates on only about 10% of the cells in the vicinity of the poles, while the strong filter operates on about half. It may be possible to alleviate some of this load imbalance by tailoring the size of the subdomains. This would involve making the latitudinal size smaller in the near-polar subdomains than in the near-equatorial subdomains. Unfortunately, this strategy runs opposite to that dictated by the column physics, where increased tropical cumulus convective activity leads to another load imbalance. An implicit barrier communication between these two sections of the model further complicates matters.

Another performance issue regarding filters is specific to the UCLA AGCM. The functions that are altered by the filters are not the prognostic variables themselves but rather the terms that comprise the hydrodynamical operator functions. This procedure is demanded by conservation requirements. The impact on performance is that communication must be interspersed throughout a portion of the algorithm rather than being performed at one specific point. We have designed the code to perform the border communication of all of the prognostic variables at the end of a time step. However, the calculation of the filtered quantities involves three stages that cannot be performed at this point. The first stage is the calculation of the unfiltered quantities at the interior cells of the subdomain using the previously communicated prognostic variables. This is followed by the filtering procedure and its requisite global communication. Finally, a border communication of the same type as required by the prognostic variables must be performed. In fact, all of these communications cannot be performed simultaneously either, since the strongly filtered quantities depend on the weakly filtered quantities as well. Our chosen method is to calculate the filtered quantities as close to the beginning of the current time step as possible so that the filtering occurs reasonably near the prognostic variable border communication of the previous time step.

### 3.2 Data Structures

Data structures in the parallel AGCM are designed to be uniform across the subdomains. A single processor is assigned the role of initializing global data. It

also calculates the subdomain decomposition and divides the global data appropriately among the subdomains. This processor may optionally be responsible for i/o and aggregation facilties. The processors assigned to specific subdomains are directed to receive in a message from the global domain processor certain scalar information prior to any subsequent operations. Once the subdomain array bounds have been determined, dynamic allocation of arrays and other initialization procedures may be completed.

We have designed the array structures used by both the global domain and the local subdomain processors to have an identical structure. That is, these arrays are contained in FORTRAN header files with the same array names and same variable dimension names. However, the values of these dimensions are different between subdomains and the global domain. In this manner, data may be communicated easily between the global and subdomain processors. We do this by copying the relevant portions of a global array in the global processsor's local memory into a scratch array to be sent to a subdomain via a message. The subdomain processor then receives this data directly into an array with the original (and proper) name. In addition to providing concise coding, this technique has the added advantage of allowing us to skip this communication procedure entirely in a single processor configuration. In this case, the global domain and the single subdomain are identical.

# 4 Portability

The rapid evolution in computing technology has produced a multiplicity of platforms from a variety of vendors. Because we wish to carry out scientific studies in a reliable production environment while developing advanced computational versions for high performance systems, it is of paramount importance to maintain a portable source code. We have addressed two issues that affect portability: dynamic memory management and interprocess communication.

## 4.1 Dynamic Memory Management

Allocating memory at run time, rather than at compile time, allows memory to be used more efficiently and makes it much easier to resize domains dynamically (for example, when attempting to balance the load between processors). Because the language constructs for dynamic memory management are nonstandard, we are using the M4 preprocessor to handle macro constructs that are processed prior to compilation. This practice confines nonportable constructs to the bodies of the macros, allowing the code to be easily modified to accommodate new architectures.

Table 1: Variation of array allocation syntax with architecture. The symbol pntr_a stands for the pointer to the array a (when relevant).

| Architecture | Syntax |
|---|---|
| Sun-4 | pntr_a = malloc(nbytes) |
| Cray-C90 | call hpalloc(pntr_a,nwords,err_flg,0) |
| BBN-TC2000 | allocate a |

For example, the variation of array allocation syntax with architecture is shown in Table 1. On the Sun-4 the storage in bytes must be specified, on the Cray-C90 the unit of storage is words, and the BBN-TC2000 computes the memory to be allocated from the array bounds information specified in the array declaration statement. The memory allocation macro has the syntax:

```
ALLOCAT(A(I1:J1,I2:J2,...,IN:JN), NBYTES),
```

where on the Cray-C90 the second argument is divided by 8, and on the BBN-TC2000 it is not used. In all of the above cases the array bounds information is not explicitly needed in the allocation command. However because the Fortran-90 standard requires that information to be present in the allocation syntax, it is included in the macro call for flexibility and future portability.

We have designed macros to handle storage allocation and deallocation, equivalencing, and real and integer declarations (both local and common). An optional error handling procedure is provided.

## 4.2 Interprocess Communication

Message passing constructs also vary from platform to platform. Each parallel architecture has its own message passing library, and message passing software designed to accommodate a variety of platforms is available. Here too, we have chosen to construct macro commands for sending and receiving messages.

These macros take the form:

```
MSEND(DESTINATION PROCESS I.D., MESSAGE I.D.,
```

```
      DATA TYPE, DATA, LENGTH, BLOCKING FLAG)

MRECV(SENDING PROCESS I.D., MESSAGE I.D.,
      DATA TYPE, DATA, LENGTH, BLOCKING FLAG)
```

Support is provided for packages in which the receiving process specifies either both message identifier and sending process or message identifier only. In the latter case one can invoke an optional message identifier conversion in which unique message identifiers are generated to handle the case where multiple processes spawn messages having the same identifier. Another feature of these macros is a data type flag to facilitate data conversion for heterogeneous systems. As with the memory management macros, an optional error handling procedure is provided.

Macros have been constructed to handle Parallel Virtual Machine (PVM) 2.4.1, the Argonne P4 package, the Livermore Message Passing System (LMPS), Thinking Machines' CMMD 2.0, and Parasoft's EXPRESS.

These macro constructs, along with CPP precompile directives, allow movement between the Cray-C90, the BBN-TC2000, the TMC CM-5, and a Sun or IBM workstation cluster with a simple change of only three parameters.

# 5 Performance

## 5.1 Performance Scaling Model

The parallel performance of any code depends on a number of factors. First, it is important to balance the computational load so that some processors do not idle while others are doing most of the work. It is also important to minimize interprocessor communications costs, again to avoid significant processor idle time. While some systems will allow message traffic simultaneously with computation, it is nevertheless informative to estimate communications costs by assuming that the two do not overlap.

One can define the speedup obtained from parallelization as the wall clock time for a serial calculation divided by the wall clock time for the same calculation in parallel. A perfectly efficient calculation will have a parallel speedup equal to the number of processors applied to the problem. Assuming the arithmetic time to be inversely proportional to the number of active processors, the speedup becomes a function of the communications time relative to the time to carry out arithmetic computations.

For the two-dimensional domain decomposition strategy used in the atmospheric model, the speedup $S$ takes the form

$$S(n_p, n_g) = n_p \left[1 + c\left(\frac{n_p}{n_g}\right)^{\frac{1}{2}}\right]^{-1} \qquad (3)$$

where $n_p$ is the processor count, $n_g$ is the number of horizontal zones, and $c$ is a machine and algorithm dependent constant. This formula assumes a load balanced computation having subdomains of nontrivial size and reflects well known scaling of communications overhead with subdomain surface to volume ratio for domain decomposition of algorithms explicit in time. This is approximately correct for the explicit time advance scheme used for the hydrodynamic fields of the UCLA AGCM.

The longitudinal global filtering operation that is applied to certain terms in the momentum equation modifies the speedup scaling. The filtering operation is designed to eliminate unstable computational modes that originate from the convergence of meridians in polar regions. This introduces the need for non-nearest neighbor subdomain communications. As a result, the parameter $c$ is modified to the form[11]

$$c' = c\frac{1 + \alpha\sqrt{n_g}}{1 + \beta\sqrt{n_g}}, \qquad (4)$$

where $\alpha$ and $\beta$ are constants that depend on the form of filter arithmetic and the number of processors $n_p$. It should also be noted that $c$ is inversely proportional to the amount of arithmetic operations required to implement the column physics parameterizations, which require no interprocessor communications.

## 5.2 Performance on BBN-TC2000

We have carried out a series of timing runs with the parallel code in order to validate the preceeding scalings and determine absolute performance levels. All of these runs were performed with the 9 layer 4° by 5° version of the AGCM with square (N by N) subdomain decompositions. In the standard operational mode of this configuration of the model, the total AGCM time step is taken as one hour. The hydrodynamics is subcycled over this time step eight times resulting in a hydrodynamic time step of 7.5 minutes. Since the initialization and termination phases of the calculation involve serial operation of the code, we have measured the performance of only the time advance segments of the model. Furthermore, because of a large penalty due to the page faulting process on the BBN TC-200 which occurred as code and data were loaded into appropriate locations upon these phases, we computed

speedups only over the middle three time steps of a five hour simulation. The amount of calculation required per time step is variable in the AGCM, being dependent on the state of the atmosphere at the given moment. This is mainly due to branching in the convective parts of the model. However, from a set of longer serial runs, we have inferred that these runs are representative of the typical calcululation.

Measured and predicted parallel speedups are shown in Fig. 2 for the hydrodynamics module of the code, with filtering and column physics turned off. The predicted scaling is from Eq.(3) with $c = 4.5$. Note that the time step must be artifically lowered in this case in order to ensure stability of the calculation.

The addition of filtering should decrease the parallel performance, since $\alpha$ in Eq.(3) is usually larger than $\beta$. This trend is confirmed in Fig. 2. The agreement is better than expected, especially since the surface-to-volume ratio is not small at the larger processor count, whereas the scaling estimates assume that this ratio is small.

From Eq.(3), we would expect that the addition of column physics would increase parallel speedup, because column physics increases the gridpoint computational intensity while requiring minimal additional communication. This trend is also confirmed in Fig. 2. In the largest decomposition (10 by 10), we see that the parallel speedup for the complete model is about 45. Examination of the timing statistics for the individual portions of the model reveals a load imbalance caused by unequally sized subdomains. This is a result of a mismatch between the subdomain decomposition and the number of zones in each direction. A more suitably matched decomposition to the 4° by 5° horizontal resolution is a 9 by 11 subdomain configuration. In this case, the overall calculation time is about 9% less than in the 10 by 10 subdomain configuration with most of the savings occurring in the hydrodynamics part of the calculation. In the column physics, the overall load is more evenly distributed in this case but the cost for the slowest subdomain remains about the same.

From the scaling estimates, it can be seen that increasing the horizontal resolution should increase parallel efficiency. In the near future, horizontal resolutions on the order of 1° by 1° will be performed, increasing the number of horizontal zones by a factor of 20. Also, it is hoped that improvements to the filtering algorithm should substantially reduce the communications penalty without adversely affecting the model predictions.

### 5.3 Performance on Workstation Clusters

The AGCM has been run on a cluster of small computers, using IBM RS/6000-550 workstations and ethernet communications under the PVM 2.4.1 system. Timings were performed for up to 16 processors and were taken on a dedicated system, with a minimum set of other processes and communicators. The times reported by the operating system inferred initial efficiencies near 100 percent for small numbers of processors, dropping to 50 percent at approximately 16 processors. However, a disparity was noted between the times reported by the system and the actual wall clock times for configurations which needed increased communication at the poles. This discrepancy correlated well with a large and increasing number of voluntary context switches reported by the individual CPUs. A voluntary context switch typically occurs when an unavailable service has been requested by the user. When additional communication measurements were taken, it was observed that the average communicating workstation bandwidth never exceeded 3.7 Megabits/s. on the cluster. The data transfer rate for the high communication configurations was typically in the range 2.0 - 3.0 Megabits/s. Peak ethernet bandwidth is specified to be 10 Megabits/s. Further experimentation using both public and private ethernet capability and using both UDP-IP and TCP-IP protocols did not materially change the observations. Similar testing on the BBN system revealed that, although the communication needs increased for these configurations, the BBN communication system yielded minimal performance degradation, reaching a peak observed throughput of 14.2 Megabits/s.

The message passage performance monitoring revealed that message lengths were distributed in a bimodal fashion for all domain decompositions. Messages clustered at lengths of 1 kilobyte for one group and near 100 kilobytes for the second group. The first group of messages was more numerous and contained most of the traffic. The second group of messages appeared to deal with data transfers at problem initiation and termination. The median message length ranged between 1.2 and 2.4 kilobytes. When varying domain decompositions with a fixed number of processors, the number and length of messages increased dramatically as the number of longitude segments increased. As the number of longitude segments increased from 1 to 16, the number of messages went up by a factor of five, and the total data transferred went up by a factor of seven.

Although at first glance the performance degradation of the IBM cluster under heavy communication

load would seem to be the fault of the message passing implementation, it is likely that the operating system attributes, combined with the basic structure of PVM 2.4.1 are the major contributors to inefficiency. While it is very useful at our present stage of code development to have explicit accounting and control of data transfer, there is inadequate operating system support on the cluster to permit separate monitoring of such factors as operator system switching times and PVM daemon overhead. It is likely that future enhancements in PVM structure and implementation will lead to efficiency improvements.

A second set of performance tests have been made using a cluster of SUN workstations using local ethernet communications. In this series of tests, the system was not made available to other users. Two message passing systems were tested, P4 and PVM v.2.4.1 in configurations up to 16 subdomain processors. In Fig. 3, it is seen that PVM v.2.4.1 is slightly more efficient than P4 on the SUN workstation. Also in Fig. 3, we show the results for the IBM cluster using the reported system times. Note that since the individual IBM nodes are faster than the SUN nodes, the parallel speedup obtained is somewhat lower.

# 6 Coupling of Climate System Components

In the near future, the AGCM model will be coupled with a variety of other applications models, such as an ocean GCM (OGCM) , a land surface and terrestial ecosystem model, an atmospheric chemistry model and a ocean biogeochemistry model. These packages must be coupled in a flexible framework that will allow efficient concurrent execution under a variety of control and synchronization strategies, and that supports intermodular exchange of a wide variety of data structures.

## 6.1 Inter-Module Data Communication

A well established practice that contributes to the modularity of complex computer codes is the art of data hiding, attaining its highest form in the methods of object-oriented programming. The principle of data hiding states that the internal representation of data in an object should be of no concern to the ultimate user of that data and so should be hidden. In principle, well-crafted objects simply provide data on request. In practice, the line between major science packages may not be so clean.

Rather than being written in an object-oriented language, the major portions of our physics models are written in FORTRAN, the *lingua franca* of scientific programming. Therefore, we need some additional tools to accomplish data hiding.

We have constructed a set of services in a small library called the Run-Time Data Base (RTDB). The RTDB maintains a set of global arrays that have been created by the various packages in the model and links a global name or alias with each such array. The data base stores the dimensionality, data type, and memory location of the array in structures that are accessible by hash table lookup of the alias. One can copy the entire array or subarray into a user-provided buffer, or dispense a pointer. Both FORTRAN and C application interfaces are provided.

Using the Run-Time Data Base, we can make data available for interchange between packages directly whenever the physical representations match. That is, whenever the data type, units, and data point locations of one model match those of another we can allow the models to share the same data structure through the RTDB. For instance two packages that each need temperature in degrees Kelvin on the same grid can share the same array, referring to it as say "temperature" when accessing the RTDB and using any other convenient variable names internally.

For a more general and robust interface, we must provide a higher degree of data hiding by building interface routines. We must assume that in general the data type, units, and locations (time or space) do not match in going from one package to the next. In addition, the quantities needed by one package may not be the same as those of another, but rather are derived from them. For convenience, the raw data going into the package interface routine as well as the transformed data destined for the next package can be posted to the Run-Time Data Base.

The interface routines A/B, A/C, that make data from package A available to packages B and C respectively, are considered part of the communications wrapper of A. Writing these routines is like providing a set of labeled output jacks on an electronic component module. To facilitate this process, we may in some cases write export functions for a particular package. These are functions that provide data values at specified temporal and spatial points on request, interpolating from the values stored in the internal representation of the package. Thus, instead of working with an internal array that stores temperature at the grid points of package A, the interface routine calls an export function, say Temp_A(x,y,z,t), to obtain the

data. This allows the internal representation of temperature in A to be changed easily (e.g., converting from a structured to an unstructured grid) without a major rewrite of the interface routines. The symmetric analog to an export routine is an import routine that provides further insulation for the receiving package. We plan to place a variety of interpolation functions in a portable library to enable these export and import functions to be written easily, perhaps by a contributing author.

At the moment, the data exchanged between packages via the RTDB is assumed to reside in local processor memory, or if not, to be made available on that processor by the programmer, for example by explicit message passing. We plan to remove this restriction by designing a distributed data base for communication. The distributed data base would have a global representation for the data to be shared between packages, together with information on how the data is distributed. A package requesting a subset of that data would not have to know the details of how the data is distributed among the processors. Data not in the local memory of the requesting processor would be gathered via messages from wherever it resides.

### 6.2 Time Integration and Synchronization Issues

The AGCM program comprises a rather large and complex parallel code. However, it forms only a portion of a much more complex modeling system that includes different modules for calculating various physical aspects of the earth system. Each module can be an independent parallel program, typically using domain decomposition to implement parallelism within a module. It is desirable to run the entire suite of programs in parallel, thus exploiting high level functional decomposition as well as data decomposition.

We have chosen to build a general work distribution code section above the existing AGCM code in order to accommodate other physics modules such as ocean circulation, chemistry, biological processes, and more. The code section, written in standard Fortran, is executed by each processor taking part in the overall model. In the following discussion, we use the term package to denote a collection of procedures that model a related set of physical processes. The AGCM code is one such package. We have identified at least six packages that will be needed for the entire earth system model. At the work distribution level for packages, it is assumed that each package can be characterized by a single time. This implies that any parallelism within a package has been controlled at least to the extent that all data subdomains in the package are synchronized in time before any subdomain processor returns control to the scheduling phase. All communication between or among packages is carried out explicitly, typically using messages or file entries.

Earth system model packages are assumed to be written in Fortran or C and to use messages with blocking receives in order to transfer data. In this scheme, there is no facility to interpret the content of a message before accepting it. Therefore, a known scheduling must be used to direct each package. In order to make the task of the package builder easy, we have assumed that each package has some schedule of times at which it must prepare (and possibly send or receive) data. This allows a package to serve as a stand alone code, provided needed information can be gotten from files or internal means. Issues of consistency and differing resolutions are dealt with at the level of intermodule communication, not within the packages themselves. Clearly, the scheduling algorithm must be known within both the package and the work delivery code sections.

The actual scheduling of a package is accomplished by determining the next time at which a message must be received by that package from each other package. If all other packages that must transmit information to the original package (to allow it to advance beyond its present state) have done so, then the earliest next message reception time is used and the package is advanced from its present time to the earliest next message reception time. Although the package has been scheduled to be advanced using knowledge of event times of senders, the scheduling process does not guarantee that messages are received. Thus the package may encounter a wait while attempting to receive information from others. Two options are available for dealing with this synchronization issue. For direct package to package communication, the wait can be absorbed within the receiving package. If the communication is to take place via some intermediate process, such as a run time data base entry, then the scheduling process itself calls a check-in procedure that receives tokens from other processes as their data placement is completed. When all needed information is in place, then the actual advance can occur.

It is assumed that the actual preparation of data for communication to other packages takes place within the sending package. When a package exits from its own advance, the scheduling process assumes that all needed data has been placed into messages or made available to an intermediate process. In the latter case, the scheduler sends tokens to appropriate receivers to

maintain synchronization. An error message will be generated if the scheduling process finds no package available for scheduling. It is allowable to have non-symmetric information transfers, such as, for example, one package sending data to another at a different rate than it receives data from that same package. At present, the "coupling times" for information transfers are specified by the user at run time. A system of priorities has been placed into the scheduling so that, if two or more packages could be advanced at the same time, the higher priority package is scheduled first. A simple priority reversal scheme has been instituted to allow fairness, if desired. Leapfrog time advance schemes are also supported, in order to allow the types of time centering between packages that are customary in many physics codes.

The work scheduling module is an attempt to allow data flow style parallel processing within the restrictions of an imperative language. It allows the exploration of a variety of coupling schemes for earth system modeling without restriction to specific non-portable approaches. It is undergoing continuing development as the number and complexity of physics modules in the earth system model increase.

## 7 Outlook

Our work with parallelization of the AGCM, as well as similar work with an oceanic general circulation model, suggest that the domain decomposition/message passing (DDMP) method represents a flexible, robust paradigm with favorable scaling properties for a wide range of anticipated climate modeling applications. Many of the additional physics process modules which will be added in the future (e.g., atmospheric chemistry and terrestrial ecosystem models) have large gridpoint computational intensity with only modest horizontal connectivity. In addition, future applications will often feature increased horizontal resolution in many of the modules. Both of these trends will lead to increased parallel efficiency in a predictable fashion. In addition, the increased options for concurrent execution of multiple modules, each parallelized via the DDMP approach, will lead to a rich environment for implementation of load balancing strategies. We believe that one of the next major issues to be addressed will relate to the question of how to understand exactly what is being produced by the next-generation climate model, i.e., the challenge of high-performance scientific visualization and efficient manipulation of truly enormous data volumes on distributed-memory platforms.

## Acknowledgement

Particular thanks are expressed to Professor Akio Arakawa at UCLA. His research on atmospheric dynamics has provided us a most suitable algorithm for the beginnings of a high performance distributed memory climate model. Work performed under the auspices of the U. S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48. Additional support has been provided by the Department of Energy CHAMMP program.

## References

[1] A. Arakawa and V. Lamb, *Methods in Comp. Phys.* **17**(1977) 173-265.

[2] A. Arakawa and V. Lamb, *Mon. Weath. Rev.* **109**(1981) 18-36.

[3] A. Arakawa and M. Suarez, *Mon. Weath. Rev.* **111**(1983) 34-45.

[4] K. Takano and M. Wurtele, *Air Force Geophysics Laboratory Report AFGL-TR-82-0205*(1982)

[5] M. Suarez, A. Arakawa and D. Randall, *Mon. Weath. Rev.* **111**(1983) 2224-2243.

[6] A. Arakawa and W.H. Schubert, *J. Atmos. Sci.* **31**(1974) 674-701.

[7] Harshvarden, *et. al.*, *J. Geophys. Res.* **92**(1987) 1009-1016.

[8] A. Arakawa, "Design of the UCLA general circulation model", (1972) Technical Report No. 7, Dept. of Meteorology, University of California, Los Angeles.

[9] M.E. Schlesinger and Y. Mintz, *J. Atmos. Sci.* **36**(1979) 1325-1361.

[10] R. Procassini, S. Whitman, and W. Dannevik, "Porting a Global Ocean Circulation Model Onto a Shared-Memory Multiprocessor", to appear in *J. Supercomputing.*

[11] W. Dannevik, "Geophysical Fluid Dynamics, Large Eddy Simulations and Massively Parallel Computing", in B. Galperin and S.Orszag, eds., *Large Eddy Simulations of Complex Engineering and Geophysical Flows*, Springer Verlag, New York (in press).

[12] C.R. Mechoso, M.J. Suarez, K. Yamazaki, A. Kitoh, A. Arakawa, *Adv. Geophys.*29(1986) 375-413.

[13] C.R. Mechoso, S.W. Lyons and J.A. Spahr, *J. Climate* 3(1990) 812-826.

[14] C.R. Mechoso, C.-C. Ma, J.D. Farrara, J.A. Spahr, R.W. Moore, "Parallelization and distribution of a coupled atmosphere-ocean general circulation mode.", *Mon. Weath. Rev.*, in press.

[15] C.-C. Ma, Y. Chao, C.R. Mechoso, W.M. Weibel and D. Halpern, "Comparison of vertical mixing schemes for ocean general circulation models.", Preprints of the Fifth Conference on Climate Variations, Denver, CO, Amer. Meteor. Soc. (1991) 388-391.

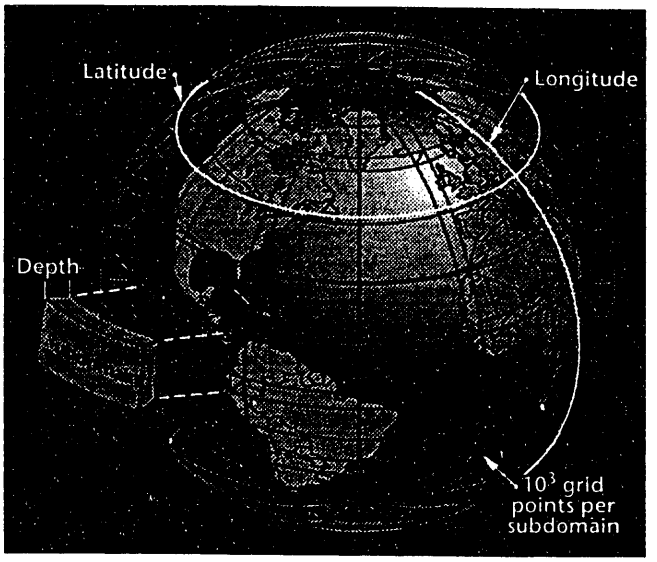



Figure 1. Two-dimensional domain decomposition used for parallelizing the atmospheric general circulation model.

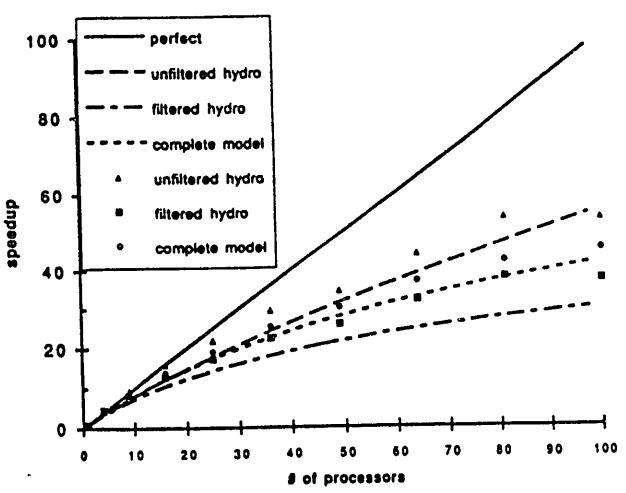



Figure 2. Parallel performance of the AGCM on the BBN-TC2000. Speedup as defined in eq.(3). vs. number of processors.

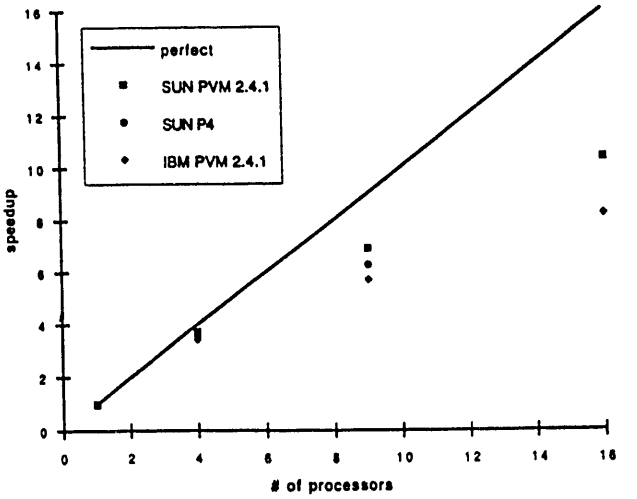



Figure 3. Parallel speedup of the AGCM on clusters of workstations.

DATE
FILMED
10/12/93
END