in Parallel Supercomputing in Atmospheric Science, eds. G-R. Hoffman and T. Kauranne, World Scientific, 1993, 500–513.

The Message Passing Version of the Parallel Community Climate Model

J. B. Drake, R. E. Flanery, D. W. Walker, P. H. Worley
Oak Ridge National Laboratory[1]

I. T. Foster, J. G. Michalakes, R. L. Stevens
Argonne National Laboratory[2]

J. J. Hack, D. L. Williamson
National Center for Atmospheric Research[3]

Abstract

This paper is a brief overview of a parallel version of the NCAR Community Climate Model, CCM2, implemented for MIMD massively parallel computers using a message-passing programming paradigm. The parallel implementation was developed on an Intel iPSC/860 with 128 processors and on the Intel Delta with 512 processors, and the initial target platform for the production version of the code is the Intel Paragon with 2048 processors. Because the implementation uses a standard, portable message-passing library, the code can be easily ported to other multiprocessors supporting a message-passing programming paradigm, or run on machines distributed across a network.

The parallelization strategy used is to decompose the problem domain into geographical patches and assign each processor to do the computation associated with a distinct subset of the patches. With this decomposition, the physics calculations involve only grid points and data local to a processor and are performed in parallel. Using parallel algorithms developed for the semi-Lagrangian transport, the fast Fourier transform and the Legendre transform, both physics and dynamics are computed in parallel with minimal data movement and modest change to the original CCM2 source code.

[1]This research was supported by the Atmospheric and Climate Research Division, Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems Inc.

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

[3]The National Center for Atmospheric Research is sponsored by the National Science Foundation.

1 Introduction

The Computer Hardware, Advanced Mathematics, and Model Physics (CHAMMP) program [3] seeks to provide climate researchers with an advanced modeling capability for the study of global change issues. Current general circulation models are generally applied at coarse spatial resolution with minimal coupling between ocean, atmosphere and biosphere. As a first goal in the program, current state-of-the-art models are being implemented on massively parallel computers, allowing an increase in spatial resolution. Accomplishment of this task will lay the groundwork for the second goal, which is to couple oceanic and atmospheric models and produce an advanced climate model with improved process representation. To be useful for scientific investigations that might impact U.S. energy policy, this advanced climate model, running on a parallel computer, must also be able to perform simulations at a significantly improved speed over the current generation of CRAY class machines.

An initial step toward the realization of the program's first objective is the development of a parallel version of the Community Climate Model CCM2 for massively parallel MIMD distributed memory computers. By providing this implementation, a number of objectives of the CHAMMP program are furthered. The computational resources provided by parallel computing can be utilized by global climate researchers using the CCM2 to investigate better process modules and to develop more comprehensive couplings. This initial implementation also provides a program benchmark to indicate how well the current massively parallel computers can perform in comparison with machines of a more conventional architecture. Finally, this implementation defines the starting point of a development path for future climate models that will incorporate different physics, different numerical methods or be written in other parallel programming styles for computers with many thousands of processors. It is also hoped that the parallel version of the CCM2 will make an immediate contribution to climate modeling by making it feasible to increase the spatial resolution researchers customarily use.

This paper gives a brief overview of the parallel algorithms required to implement CCM2 using a message-passing programming paradigm, and describes issues that are important to this code for gaining good performance on distributed memory multiprocessors. While the target machine for the parallel code is the Intel Paragon, the code uses PICL, a portable instrumented communication library [5], [6], to implement interprocessor communication, providing a degree of portability and the ability to collect performance data. An experimental version of the code has also been implemented using PVM, the Parallel Virtual Machine, and has been used to run the code across a network of workstations. Validation of these models is currently in progress.

2 NCAR CCM

Over the last decade, the NCAR Climate and Global Dynamics (CGD) Division has provided a comprehensive, three-dimensional global atmospheric model to university and NCAR scientists for use in the analysis and understanding of global climate. Because of its widespread use, the model was designated a Community Climate Model (CCM). The advantages of the community model concept, in which many scientists use the same basic model for a variety of scientific studies, were demonstrated in workshops held at NCAR in July 1985 [1], July 1987 [14], and July 1990 [15]. Fundamental strengths and weaknesses of the model have been identified at these workshops through the presentation of a diverse number of applications of the CCM. Much constructive dialogue has taken place between experts in several disciplines at these meetings, leading to continued improvements in the CCM with each release.

The most recent version of the CCM, CCM2, which was released to the NSF research community in October 1992, incorporates the most ambitious set of changes to date. The algorithms of the model are described in [7] and details of the implementation on the CRAY YMP are provided in a Users' guide [2]. The bulk of the effort in the NCAR Climate Modeling Section over the last several years has been to improve the physical representation of a wide range of key climate processes in the CCM, including clouds and radiation, moist convection, the planetary boundary layer, and transport. The resulting changes to the model have resulted in a significantly improved simulation and fundamentally better climate model. On the parameterized physics side, changes include the incorporation of a diurnal cycle, along with the required multilayer heat capacity soil model, and major improvements to the radiation scheme, including a δ-Eddington solar scheme (18 spectral bands), a new cloud albedo parameterization, a new cloud emissivity formulation using liquid water path length, a new cloud fraction parameterization, and a Voigt correction to infrared radiative cooling in the stratosphere. The moist adiabatic adjustment procedure has been replaced with a stability-dependent mass flux representation of moist convection, and an explicit planetary boundary layer parameterization is now included, along with a gravity-wave drag parameterization.

On the dynamics side, the spherical harmonic (spectral) transform method is still used for the horizontal discretization of vorticity, divergence, temperature and surface pressure, but a semi-Lagrangian transport scheme has been substituted for the advection of water vapor as well as for an arbitrary number of other scalar fields (e.g., cloud water variables, chemical constituents, etc.), and the vertical coordinate makes use of a hybrid (or generalized σ) formulation. The model has been developed for a standard horizontal spectral resolution of T42 (2.8 degrees by 2.8 degrees transform grid) with 18 vertical levels and a top at approximately 2.9 mb. The model code has also been entirely rewritten with three major objectives: much greater ease of use and modification, conformation to a plug-compatible physics interface, and the incorporation of single-job multitasking capabilities.

CCM2 will provide the basis for a large body of experimental and further developmental efforts by a large community of university and NCAR climate investigators, many of whom may not be directly involved in the CHAMMP initiative. Because of the community nature of the enterprise, new methods and process modules are continually emerging. The new methods will be incorporated in future releases and versions of the model as seems appropriate for computer efficiency and the requirement for increased capabilities.

3 Parallel Algorithms Overview

There are two major dynamics algorithms in the CCM2 code, the spectral transform method [4], [8], [9] and a semi-Lagrangian transport method [13]. Aside from the dynamics there are numerous calculations that incorporate other physical processes such as clouds and radiation, moist convection, the planetary boundary layer and surface processes. These other processes share the common feature that they are coupled horizontally only through the dynamics. We lump all these processes under the general term "physics" and note that the physics calculations are independent for each vertical column of grid cells in the model.

The independence of the physics calculations for each horizontal grid point is the primary source of parallelism in the parallel code, PCCM2. By partitioning the horizontal grid points into blocks and assigning them to the processors, a decomposition of the three dimensional data structures is defined. This decomposition allows each vertical column of grid points to be used in the radiation calculation without requiring information that another processor might have. Thus, the "physics" calculation is perfectly parallel, in that it requires no interprocessor communication. For a T42 grid, there are 128 horizontal grid points in the longitudinal direction and 64 grid points in the latitudinal direction. Thus there is potential for using 8192 processors for this calculation. Due to assumptions made in the parallel algorithms for the dynamics calculations, the current code can only use 1024 processors at T42, but all or part of this parallelism can be recovered if additional parallel inefficiency can be tolerated in the parallel dynamics algorithms.

The dynamics calculations make use of the spectral transform method for the approximation of all terms of the equations except the advective term in the moisture transport equation. The spectral transform involves two stages or two separate transforms, the fast Fourier transform (FFT) and the Legendre transform. The Fourier transform integrates information along each east-west gridline in the longitudinal direction. In the spectral transform from grid space to spectral space, this is followed by a Legendre transform integrating the results of the FFT in the north-south, or latitudinal, direction. Thus, the spectral transform operates on data "globally" in that information from each horizontal grid point, and from each processor, contributes to each value in the final result. Earlier research on the spectral transform method established that efficient Legendre transforms could be performed using at least a couple of methods. For example, a highly efficient ring-pipeline algorithm was proposed [16] that overlaps communication and computation, giving a "scalable" parallel algorithm. For the initial implementation of PCCM2, a vector sum algorithm (described below) is used to calculate the Legendre transform. Unlike the ring-pipeline algorithm, the vector sum algorithm, as implemented, does not permit the overlapping of communication and computation. But it is still fast and is more easily implemented in the context of CCM2 than is the ring-pipeline algorithm. The FFT can also be performed effectively in parallel [12] by exploiting the fact that there are multiple gridlines to be transformed at any one time, making it possible to hide some of the communication cost by overlapping the communication in one FFT with the computation in another. Notably, both the Intel Paragon and the Thinking Machines CM-5 have hardware that supports the overlapping of communication with computation. The parallelization of the spectral transforms in CCM2 has driven most of the design decisions adopted for the organization of the data structures.

The advective terms in the moisture equation are updated using a semi-Lagrangian transport method. The method updates the value of the moisture field at a grid point (the arrival point, A) by first establishing a trajectory through which the particle arriving at A has moved during the current timestep. From this trajectory the departure point, D, is calculated and the moisture field is interpolated at D using shape preserving interpolation. All the calculations involve physical space (grid point) data, and are decomposed over the processors with the same mesh decomposition used to parallelize the physics and the spectral transform. The parallel implementation of this algorithm uses the fact that timestep constraints imposed by the Eulerian dynamics limit the distance between the departure and arrival points in the latitude direction. By extending the arrays in each processor, thus "overlapping" regions assigned to neighboring processors, and updating the overlapped portion of the array prior to each timestep via interprocessor communication, the calculations in the different processors can proceed independently.

The rest of this section describes in more detail the data decomposition and parallel algorithms for the Legendre transform and the semi-Lagrangian transport. (A detailed description of the parallel FFT can be found in [12].) To motivate this discussion, important attributes of the spectral transform method are also reviewed.

3.1 The Spectral Transform Algorithm

The spectral transform method is based on a dual representation of the scalar fields in terms of a truncated series of spherical harmonic functions and in terms of values on a rectangular tensor-product grid whose axes represent longitude and latitude.

Representations of the state variables in spectral space are the coefficients of an expansion in terms of complex exponentials and associated Legendre polynomials,

$$\xi(\lambda,\mu) = \sum_{m=-M}^{M} \sum_{n=|m|}^{N(m)} \xi_n^m P_n^m(\mu) e^{i\cdot m\lambda}, \tag{1}$$

where $P_n^m(\mu)$ is the (normalized) associated Legendre function [10] and $i = \sqrt{-1}$. The spectral coefficients are then determined by the equation

$$\xi_n^m = \int_{-1}^{1} \left[\frac{1}{2\pi} \int_0^{2\pi} \xi(\lambda,\mu) e^{-i\cdot m\lambda} d\lambda \right] P_n^m(\mu) d\mu \equiv \int_{-1}^{1} \xi^m(\mu) P_n^m(\mu) d\mu \tag{2}$$

since the spherical harmonics $P_n^m(\mu)e^{i\cdot m\lambda}$ form an orthonormal basis for square integrable functions on the sphere. In the truncated expansion, M is the highest Fourier mode and $N(m)$ is the highest degree of the associated Legendre function in the north-south representation. Since the physical quantities are real, ξ_n^{-m} is the complex conjugate of ξ_n^m, and only spectral coefficients for nonnegative modes need to be calculated.

To evaluate the spectral coefficients numerically, a fast Fourier transform (FFT) is used to find $\xi^m(\mu)$ for any given μ. The Legendre transform is approximated using a Gaussian quadrature rule. Denoting the Gauss points in $[-1,1]$ by μ_j and the Gauss weights by w_j,

$$\xi_n^m = \sum_{j=1}^{J} \xi^m(\mu_j) P_n^m(\mu_j) w_j. \tag{3}$$

Here J is the number of Gauss points. (For simplicity, we will henceforth refer to (3) as the forward Legendre transform.) The point values are recovered from the spectral coefficients by computing

$$\xi^m(\mu) = \sum_{n=|m|}^{N(m)} \xi_n^m P_n^m(\mu) \tag{4}$$

for each m (which we will refer to as the inverse Legendre transform), followed by FFTs to calculate $\xi(\lambda,\mu)$.

The tensor-product grid in physical space is rectangular with I grid lines evenly spaced along the longitude axis and J grid lines along the latitude axis placed at the Gaussian quadrature points used in the forward Legendre transform. To allow exact, unaliased transforms of quadratic terms the following relations are sufficient: $J \geq (3M+1)/2$, $I = 2J$, and $N(m) = M$ [9]. Using $N(m) = M$ is called a triangular truncation because the (m,n) indices of the spectral coefficients make up a triangular array. The examples in the rest of this section will assume a triangular truncation is used.

3.2 Data Decompositions

In the spectral transform algorithm, computations are performed in both the physical and spherical harmonic (or *spectral*) domains, and transforming from one domain to the other involves passing through the *Fourier domain*, whose coordinates are Fourier wavenumber and latitude coordinates. Thus, we must be concerned with the distribution of data in three domains.

In specifying the domain decompositions, the multiprocessor is viewed as a logical $P \times Q$ two dimensional processor grid. (P and Q are currently compile-time parameters for PCCM2.) For the physical domain, the latitudinal dimension is partitioned into $2Q$ intervals, each containing $J/2Q$ consecutive grid lines along the latitude axis. Each processor row is assigned two of these intervals, one from the northern hemisphere, and the reflected latitudes in the southern hemisphere. This assignment allows symmetry to be exploited in the Legendre transform. The assignment also restricts Q, the number of processor rows, to be no larger than $J/2$.

The longitudinal dimension is partitioned into P equal intervals, with each interval being assigned to a different processor column. The resulting "block" decomposition of the physical domain is illustrated in Fig. 1 for a small example.

The Fourier domain can be regarded as a wavenumber-latitude grid, so, like the physical domain, the Fourier domain is two-dimensional. However, a different decomposition is used. The differences arise because of the way in which the FFT algorithm permutes the ordering of the output Fourier coefficients [12]. But, modulo this reordering, the wavenumber "dimension" is partitioned into P sets of consecutive wavenumbers, with each set being assigned to a different processor column. The partitioning function in the latitude direction is the same as in the physical domain. See Fig. 1 for an example decomposition.

The spectral domain can also be regarded as two dimensional. For example, for a triangular truncation, the domain is a triangular grid whose axes are wavenumber and degree of associated Legendre polynomial (n). The wavenumber "dimension" is partitioned and assigned to processors exactly as for the Fourier domain, i.e. the wavenumbers are reordered, partitioned into consecutive blocks, and assigned to the processor columns. But, unlike the physical and Fourier domains, the remaining dimension in the spectral domain is not partitioned. Instead, all spectral coefficients associated with a given wavenumber are duplicated across all processors in the processor column to which that wavenumber was assigned. It is this duplication that allows the vector sum algorithm described below to be used. Again, see Fig. 1 for an example decomposition.

Note that in a triangular truncation, the number of spectral coefficients associated with a given Fourier wavenumber decreases as the wavenumber increases. Without the reordering of the wavenumbers caused by the FFT, this would cause a noticeable load imbalance, with processor columns associated with larger wavenumbers having

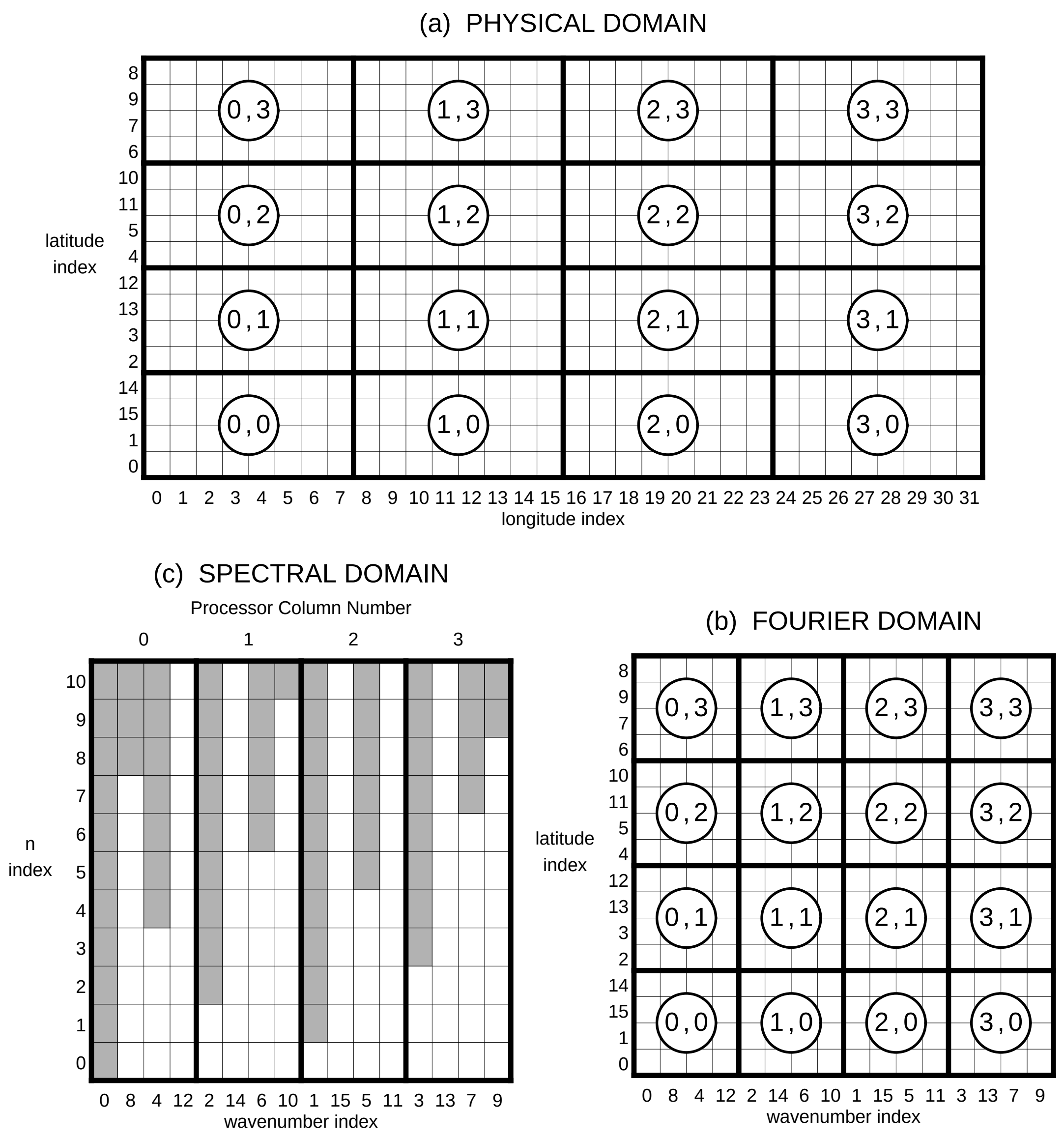

Figure 1: The decomposition of (a) the physical, (b) the spectral, and (c) the Fourier domains over a 4×4 grid of processors. For figures (a) and (b), each small cell represents a data item. The thicker lines show the boundaries between processors. The circles contain the processor coordinates. The shaded cells in figure (c) represent the spectral coefficients to be included in the spectral transform, and how these are decomposed over processor columns.

very few spectral coefficients. The reordering of the wavenumbers leads to a much better load balance.

3.3 Parallel Legendre Transform

The forward and inverse Legendre transforms are

$$\xi_n^m = \sum_{j=1}^{J} \xi^m(\mu_j) P_n^m(\mu_j) w_j$$

and

$$\xi^m(\mu_j) = \sum_{n=|m|}^{N(m)} \xi_n^m P_n^m(\mu_j)$$

respectively. For the forward Legendre transform, each ξ_n^m depends only on data associated with the same wavenumber m, and so depends only on data assigned to a single processor column. Each processor in that column can calculate independently its contribution to ξ_n^m, using data associated with the latitudes assigned to that processor. To finish the calculation, these P contributions need to be summed, and the result needs to be rebroadcast to all P processors, since spectral coefficients are duplicated within the processor column. To minimize communication costs, local contributions to all spectral coefficients can be calculated first, leaving a P-way vector sum (made up of the local contributions to all of the spectral coefficients assigned to this processor column) and rebroadcast to be calculated. This motivates naming this approach the *vector sum* algorithm.

The column-wise vector sum is a separate step in the algorithm, and the communication is not overlapped with computation. But there are sophisticated techniques for calculating the vector sum that do a good job of minimizing both the communication cost and the associated parallel computation cost. Currently we use a variant of the recursive halving algorithm [11].

For the inverse transform, calculation of $\xi^m(\mu_j)$ requires only spectral coefficients associated with wavenumber m, all of which are local to every processor in the corresponding processor column. Thus, no interprocessor communication is required in the inverse transform.

In summary, using the vector sum algorithm to compute the Legendre transforms incurs no additional computational cost, is perfectly parallel with good load balance within a processor column, and requires interprocessor communication in only the forward transform. Moreover, this communication can be implemented very efficiently. Finally, few modifications to CCM2 were required to implement this algorithm in PCCM2.

The disadvantages of the vector sum algorithm are that all computations *within* the spectral domain must be calculated redundantly (in the processor column), the communication in the forward Legendre transform can not be overlapped with communication, and additional storage is required to hold the duplicated spectral coefficients. Since relatively little work is done in the spectral domain in CCM2, this redundant work has not proved to be an issue, and the vector sum has proved to be a viable parallel algorithm for PCCM2.

3.4 Semi-Lagrangian Transport

As mentioned previously, the advection of moisture is done in CCM2 using a semi-Lagrangian transport (SLT) method in conjunction with shape preserving interpolation [13]. The method updates the value of the moisture field at a grid point (the arrival point, A) by first establishing a trajectory through which the particle arriving at A has moved during the current timestep ($2\Delta t$). This trajectory is found iteratively using the interpolated velocity field at the mid-point, M, of the trajectory. From this mid-point the departure point, D, is calculated and the moisture field is interpolated at D using shape preserving interpolation. All the calculations involve physical space (grid point) data and are decomposed over the processors with the same mesh decomposition described above.

The modifications made for the parallel implementation involved a redefinition of the extended grid arrays already implemented for the SLT. Extended grids are necessary since cubic interpolation requires two additional points outside the region being interpolated. Extending the grids even further leads to regions of overlap among the processors, but it can be guaranteed that with enough extension the departure point and subsequent interpolation of the moisture field can be done with the data on the extended grid, and thus local to the processor. The amount of the extension is controlled by separate parameters for the latitudinal and longitudinal directions.

The overlap regions on each processor must be updated each timestep. Communication is blocked in such a way to allow the possibility of overlap with more than one processor. This can occur, for example, when a large number of processors are used and each processor has only two latitudes. The setting of the extended grid at the poles also requires communication between processors. In particular, the pole point, which occupies an entire latitude line in the extended grid, is assigned a value based on the zonal average of nearby latitude lines. A sum across the pole processors is required for this to be calculated. Since the pole processors lie on the first row of the processor mesh, a separate procedure is used for these processors.

4 Performance Issues

The performance of CCM2 on a particular machine is largely determined by four factors: the individual processor speed, the interprocessor communication speed and latency, the I/O bandwidth, and the load imbalance. In this section each of these will be discussed along with the strategy being followed to optimize performance on distributed memory, message-passing multicomputers. First, however, we will briefly outline the performance of the code on the CRAY Y-MP. Reference will be made to some specific aspects of the model code and parameters. Rather than describing them in detail here, the reader is referred to the Users' Guide [2].

The hardware performance monitor on the CRAY Y-MP reports that the CCM2 running on a single processor averages 154 Mflops and can complete 0.25 timesteps per CPU second. When multitasked on 8 processors, it completes 1.86 timesteps per wallclock second. These averages were taken from a 24 hour simulation at T42 resolution in which a history tape and restart file were written once. They include the varying amounts of time required by differing types of timesteps for the calculation of absorptivities and emissivities associated with the radiation calculation. (More specifically, IRAD = 3 and IRADAE = 36 with DTIME = 1200.) A table of the top 12 routines in CPU usage, Fig. 2, shows the computation rate and the percentage of total time the code spends in the routine. These routines account for 60% of the

Routine	Mflops	%Total
RADABS	216	13.7
RADDED	152	10.0
LINEMS	169	5.2
RADCSW	144	5.2
GRCALC	154	5.1
RADCLW	54	4.5
GRMULT	180	3.6
HERXIN	191	2.8
SPEGRD	167	2.7
CMFMCA	93	2.5
QUAD	156	2.5
OUTFLD	41	2.2

Figure 2: Top 12 most intensive routines on the CRAY Y-MP

total execution time of the model.

The individual processor speed on the Intel iPSC/860 is highly dependent on the optimization done by the compiler and on the characteristics of the code being compiled. Thus, the comparison with the CRAY results must be taken as comparing a mature compiler with one under development. The average performance of a single processor executing the PCCM2 code is 5.3 Mflops. This number was obtained by making the identical run as above on a small number of processors so that the communication time was minimal. This number represents 7% of the manufacturer quoted peak speed of 80 Mflops for single precision floating point calculations. The computationally intensive routines listed above will be examined and modified to better take advantage of the i860 chip. Such improvements are expected to have a dramatic effect on the performance of the PCCM2.

Through parallel execution the speed of the simulation can be increased dramatically. Executing on 512 processors of the Intel DELTA, the average computational rate is 750 Mflops. At this rate an average of 1.2 timesteps were taken each second. With 512 processors, only 1.4 Mflops per processor were achieved, implying that the communication costs and the load imbalance impacted the performance of the code. Of the listed routines, only LINEMS, GRCALC and SPEGRD involve communication. The other routines execute in parallel without communication. Other routines that do not appear on the top 12 list involve communication and may be significant in terms of communication costs. For example, the SLT routine, SLTINI, communicates the overlap regions each timestep to initialize the extended grid. This routine was insignificant on the CRAY but can be expected to have some impact in the parallel message-passing code.

Efforts are currently underway to quantify the load imbalance of the parallel code and to identify communication tasks that present performance bottlenecks. For the 512 processor simulation on the Intel DELTA, 28% of the total time was spent in the spectral dynamics calculation, while 25% was spent in the semi-Lagrangian transport section of the dynamics calculation. The physics calculations used 41% of the total time and roughly 10% of this was idle time due to load imbalance. A finer division, by routine, will give a clearer picture and indicate where performance improvements can be made.

The parallel implementation optionally keeps the scratch files NRA1 and NRB1 incore. If the out-of-core option is chosen, each processor opens a separate file so that parallel I/O can be performed. On the Intel computers, this creates contention for the I/O node services, since individual processors must perform I/O through the I/O nodes. Runs on the Intel iPSC/860 indicate that the out-of-core option degrades performance significantly. The runs reported above were performed with the incore option enabled.

Aside from the scratch datasets, the code also writes a history file every few timesteps and a restart dataset. These files can be written in parallel but are currently written by collecting the data on one processor and writing in a sequential mode. Intel's parallel file system (PFS) will be used to support the production use of the PCCM2 and the I/O will be optimized when PFS becomes available.

5 Conclusions

The development of a distributed memory, message-passing version of the Community Climate Model, CCM2, required the development of parallel algorithms for the spectral transform and for the semi-Lagrangian transport, as has been described in this paper. An unoptimized implementation has been completed and is currently being tested and validated on the Intel iPSC/860 and DELTA. The code will be optimized for parallel execution, communication and parallel I/O on the Intel Paragon.

References

[1] R. Anthes, *1986 summary of workshop on the NCAR Community Climate/Forecast models*, Bull. Amer. Meteor. Soc., 67 (1986), pp. 194–198.

[2] L. J. Bath, J. Rosinski, and J. Olson, *Users' guide to NCAR CCM2*, NCAR Tech. Note NCAR/TN–379+IA, National Center for Atmospheric Research, Boulder, Colo., 1992.

[3] Department of Energy, *Building an advanced climate model: Progress plan for the CHAMMP climate modeling program*, DOE Tech. Report DOE/ER–0479T, U.S. Department of Energy, Washington, D.C., December 1990.

[4] E. Eliasen, B. Machenhauer, and E. Rasmussen, *On a numerical method for integration of the hydrodynamical equations with a spectral representation of the horizontal fields*, Rep. No. 2, Institut for Teoretisk Meteorologi, Kobenhavns Universitet, Denmark, 1970.

[5] G. A. Geist, M. T. Heath, B. W. Peyton, and P. H. Worley, *PICL: a portable instrumented communication library, C reference manual*, Tech. Report ORNL/TM-11130, Oak Ridge National Laboratory, Oak Ridge, TN, July 1990.

[6] ——, *A users' guide to PICL: a portable instrumented communication library*, Tech. Report ORNL/TM-11616, Oak Ridge National Laboratory, Oak Ridge, TN, August 1990.

[7] J. J. Hack, B. A. Boville, B. P. Briegleb, J. T. Kiehl, P. J. Rasch, and D. L. Williamson, *Description of the NCAR Community Climate Model (CCM2)*, NCAR Tech. Note NCAR/TN–382+STR, National Center for Atmospheric Research, Boulder, Colo., 1992.

[8] B. Machenhauer, *The spectral method*, in Numerical Methods Used in Atmospheric Models, vol. II of GARP Pub. Ser. No. 17. JOC, World Meteorological Organization, Geneva, Switzerland, 1979, ch. 3, pp. 121–275.

[9] S. A. Orszag, *Transform method for calculation of vector-coupled sums: Application to the spectral form of the vorticity equation*, J. Atmos. Sci., 27 (1970), pp. 890–895.

[10] I. A. Stegun, *Legendre functions*, in Handbook of Mathematical Functions, M. Abramowitz and I. A. Stegun, eds., Dover Publications, New York, 1972, ch. 8, pp. 332–353.

[11] R. A. van de Geijn, *On global combine operations*, LAPACK Working Note 29, Computer Science Department, University of Tennessee, Knoxville, TN 37996, April 1991.

[12] D. W. Walker, P. H. Worley, and J. B. Drake, *Parallelizing the spectral transform method. Part II*, Concurrency: Practice and Experience, 4 (1992), pp. 509–531.

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

[14] D. L. Williamson, Ed., *Report of the second workshop on the community climate model*, NCAR Tech Note NCAR/TN-310+PROC, National Center for Atmospheric Research, Boulder, CO 80307, 1988.

[15] ——, *CCM progress report - July 1990*, NCAR Tech Note NCAR/TN–351+PPR, National Center for Atmospheric Research, Boulder, CO, July 1990.

[16] P. H. Worley and J. B. Drake, *Parallelizing the spectral transform method*, Concurrency: Practice and Experience, 4 (1992), pp. 269–291.