RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

Solving the Shallow Water Equations on the Cray X-MP/48 and the Connection Machine 2

Paul N. Swarztrauber and Richard K. Sato

December 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.48

NASA Cooperative Agreement Number NCC 2-387

Solving the Shallow Water Equations on the Cray X-MP/48 and the Connection Machine 2

Paul N. Swarztrauber and Richard K. Sato

Research Institute for Advanced Computer Science
NASA Ames Research Center - MS: 230-5
Moffett Field, CA 94035

RIACS Technical Report 89.48

December 1989

The Research Institute of Advanced Computer Science is operated by Universities Space Research Association, The American City Building, Suite 311, Columbia, MD 244, (301)730-2656

Work reported herein was supported in part by Cooperative Agreements NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA).

Solving the shallow water equations on the Cray X-MP/48 and the Connection Machine 2

by

Paul N. Swarztrauber [1,2] and Richard K. Sato [1]

December 1989

ABSTRACT

The shallow water equations in Cartesian coordinates and two dimensions are solved on the Connection Machine 2 (CM-2) using both the spectral and finite difference methods. A description of these implementations is presented together with a brief discussion of the CM-2 as it relates to these specific computations. The finite difference code was written both in C* and *LISP and the spectral code was written in *LISP. The performance of the codes is compared with a FORTRAN version that was optimized for the Cray X-MP/48.

1 National Center for Atmospheric Research, Boulder, Colorado 80307, which is sponsored by the National Science Foundation.

2 This work was supported by the NAS Systems Division via Cooperative Agreement NCC 2-387 between NASA and the University Space Research Association (USRA). It was performed while this author was visiting the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, CA 94035.

1. Introduction. The Connection Machine 1 (CM-1) is a single instruction multiple data (SIMD) computer with 64K one-bit processors [3] that has been of significant interest to the computer science community. However its performance is below that of other supercomputers and consequently it has not generated significant interest among institutions that require state-of-the-art performance. However, a departure from the "one-bit" philosophy of the CM-1 resulted in the CM-2 with a reported peak performance capability that is superior to most supercomputers and has therefore generated a broad base of interest in the supercomputing community.

Although the fully configured CM-2 is still commonly referred to as a 64K one-bit processor machine, its increased performance is due to the addition of 2K Weitek 32-bit floating point processors. Each Weitek is rated at 16 Mflops and therefore the peak rate of the CM-2 is presented as 32 billion floating point operations per second (32 Gflops). It therefore became necessary for the scientific computing community in general and the atmospheric science community in particular to examine the performance of the CM-2 on their particular type of computations. As we will show, the performance of the CM-2 on the shallow water equations is significantly less than 32 Gflops but it is nevertheless in a range that continues to make it interesting to scientific supercomputer users.

In a previous paper [8] we presented early results of the finite difference model which will be updated in this paper together with new results for a spectral model. The finite difference code is highly vectorizable and parallelizable and runs at near peak rates on most computers. Consequently it provides a optimistic view of the performance that is possible for problems in the atmospheric sciences.

The shallow water equations constitute a greatly simplified weather prediction code [7] that has already been used to benchmark a number of machines [4]. Their solution is a relatively small but necessary step in the process of determining the applicability of the SIMD architecture to climate and weather prediction. Ultimately these tests must be conducted on the sphere where harmonic transforms are computed for the spectral method. A compressible model should also be implemented. An explicit model with local communication should be straightforward compared to a implicit model which requires global communication. A list of problems that can be used to evaluate the suitability of massively parallel processing (MPP) is given in Table 1 below.

Table 1

PARALLEL GEODYNAMICS PROJECTS

Equations	Geometries	Methods
Shallow Water	Cartesian	Finite Difference
Incompressible	Spherical	Spectral
Compressible	Irregular	Semi-Lagrangian

The entries in Table 1 can be combined to provide a sequence of models and experiments that could assist in determining the suitability of massively parallel computing to the problems in the atmospheric and related sciences. For example, we chose first to solve the shallow water equations in Cartesian coordinates using finite differences on the Connection Machine. Our next step was to implement the spectral method for the shallow water equations in Cartesian coordinates. We will present the results of these experiments in the sections that follow; however, we begin with a brief description of the Connection Machine.

2. The Connection Machine 2. Consider now several components of the CM-2 and concepts that facilitate the implementation of scientific computations.

HOST

There is only one copy of the program which resides on the host or front end which is physically distinct from the CM-2 but controls its execution. A CM-2 can support up to four front end systems that must currently be either a DEC VAX 8000 series, Symbolics 3600 system or a Sun 4/280. Programs are developed, stored, compiled, loaded, and executed on the front end. Instructions to parallel variables are passed over an interface to the microcontroller where they are broadcast for execution by all processors simultaneously. Program steps

that do not involve operations on parallel variables are executed on the front end. Hence the CM-2 is essentially an extension to the front end which provides computational power for operations on parallel variables. Note that any serial portion of the code must therefore run at the speed of the host.

The CM-2 does not support multi-programming in the traditional sense. That is, several jobs cannot share the same processors but rather a job must complete execution before another job can be allowed access to the processors. However, a CM-2 can be configured as several smaller subsystems and each of these subsystems can be simultaneously running independent jobs each from a separate front end machine.

PROCESSORS

The CM-2 is a massively parallel SIMD machine. A fully configured CM-2 has 64K (65,536) single-bit processors that are packaged 16 per chip. For every two chips (32 processors), there is a Weitek 3132 floating point chip. The CM-2 operates at an 8.0 MHz clock speed and the Weitek processors can produce an add and a multiply per cycle for a peak computational rate of 16 million floating point operations per second (Mflops). The 2048 Weitek processors therefore combine for a total peak floating point computational rate of 32 billion floating point operations per second (Gflops). In practice the performance is considerably below this figure because of the communication between the CM chips and the Weitek 3132. The actual performance figures are presented in the sections that follow.

MEMORY

Each processor has 8K (8192) bytes of memory for a total of 512 megabytes of memory for a 64K processor system. Each processor can access data from its memory at a rate of 5 megabits per second. The total memory bandwidth is therefore over 300 gigabits per second. A processor can only operate on data that is in its own memory but the processors are logically interconnected so that data can be transferred between processors.

COMMUNICATIONS

The 4096 CM-2 chips are connected in a 12-dimensional hypercube network. Communication is bit-wise rather than packet-wise with a maximum of 5×10^{10} bits transmitted per second (aggregate). However, like peak computational rates, the peak communication rates are not achieved in practice, particularly by the router. General communication between processors is handled by a router that

can transmit messages of any length. The throughput of the router depends on the length of the messages and on the access patterns with typical bandwidths from .1 to 1 billion bits per second.

A second, faster mechanism for communications between chips is a programmable NEWS grid that can be configured to support multi-dimensional configurations. The NEWS grid allows processors to transmit data between processors according to a regular rectangular pattern. For example, in a 2-dimensional grid, each processor could simultaneously execute the same instruction to fetch data from its neighbor to the north, east, west or south. Use of the NEWS grid eliminates the overhead of the router. This feature will also be available in FORTRAN which is currently in beta test phase.

VIRTUAL PROCESSORS

The CM-2 is a data parallel computer. For variables that are declared to be parallel, each processor will have a single element of that variable in its memory. If this number exceeds the number of physical processors then virtual processors are created. When an operation is performed on a parallel variable, each processor performs the operation on its element. For example, if A, B, and C are declared as parallel variables, then each processor will have a single element from each of the three arrays and a parallel operation such as C = A + B will be executed by each processor using its A, B, and C variables.

Through the concept of virtual processors, the CM-2 can support applications that require more processors than are physically available. For example, parallel variables with 64K elements will have one value associated with each processor. If, however, parallel variables have 256K elements, then four virtual processors are created for each physical processor at the time the CM-2 is initialized for the application.

There are several advantages to an SIMD architecture:

a) It is not necessary to synchronize the processors which simplifies code development, debugging and fault isolation.

b) The issue of whether or not to parallelize is nonexistent. The nagging issue that persists throughout the multiprocessing community is whether or not to encourage users to multitask or to simply run separate job streams.

c) The host or front end contains the only copy of the code. Multiple copies of a weather code would require a substantial memory resource since these codes can occupy several Mbytes of memory. Indeed the code would occupy all of the memory if the CM-2 were configured as a multiple instruction multiple data (MIMD) machine that required multiple copies of the code. This aspect of an SIMD computer highlights the fundamental design philosophy of the CM; namely, that the electronics should be dedicated to items such as data and arithmetic where it is used frequently rather than memory for multiple copies of a code where its use is relatively infrequent. Memory for address space can also be minimal since arrays rather than their individual elements are referenced.

There are also some disadvantages to SIMD.

a) The scalar part of any code will perform at the speed of the host. This simply emphasizes what is already known from Amdahl's rule, namely that scalar code must be kept at a minimum for efficient multiprocessing. For certain computations such as the physical parameterizations in atmospheric models, this may require a creative reformulation of the computations.

b) Computations on smaller arrays take the same time as computations on larger arrays. The computing time depends only on the length of the formula and is independent of the size or dimensions of the array. If the formulas for the boundary and interior of a domain have about the same arithmetic operations per point then they will require about the same amount of computing time.

c) Nested gridpoint conditionals are expensive since all possible branches must be computed. The appropriate alternative is then selected via a mask. However this increase in computation is bounded once the model is fixed unlike the computations as a function of grid size.

3. The finite difference model of the shallow water equations. The benchmark code for the CM-2 is a simple atmospheric dynamics model based on the shallow water equations [7] which represent a primitive but useful model of the dynamics of the atmosphere. Its performance provides a rough estimate of the performance of more complex atmospheric models on various computer

systems. This estimate is generally of the high side since state-of-the-art models would have a higher percentage of scalar computations that would run on the host or front end system.

The shallow water equations consist of the following system of three time dependent, partial differential equations. Two additional equations define relationships for the vorticity and for the height field.

$$\frac{\partial u}{\partial t} - \zeta v + \frac{\partial H}{\partial x} = 0$$

$$\frac{\partial v}{\partial t} + \zeta u + \frac{\partial H}{\partial y} = 0$$

$$\frac{\partial P}{\partial t} + \frac{\partial}{\partial x}(Pu) + \frac{\partial}{\partial y}(Pv) = 0$$

where u and v are velocities in the x and y direction respectively; P is pressure; $H = P + \frac{1}{2}(u^2 + v^2)$ is the height field and ζ is the vorticity given by

$$\zeta = \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}.$$

This form of the shallow water equations was used by Sadourney [7] because it yielded a conservative finite difference approximation. Using these equations and his numerical formulation, a solution was obtained on a rectangular domain in Cartesian coordinates with periodic boundary conditions in both directions. The leapfrog time differencing scheme was used and second order centered finite difference approximations were used for spatial derivatives.

The model is in-memory with variables declared as two-dimensional arrays that are the size of the grid plus 1 (an extra row and column are defined for storage of periodic boundary values). For a grid of 256×256 points, the memory requirement on the Cray X-MP/48 is about one million words. Secondary storage is not used and hence there was no requirement for I/O to external devices. The computationally intensive portions of the code consist of three doubly-nested DO-loops in which various quantities are computed over the two-dimensional grid.

Each of the loops is evaluated in a separate subroutine and the computations in the loops account for more than 90% of the floating point operations in the code. The inner loops are easily vectorized and for systems like the CM-2 which support parallel execution on multiple processors, the computations that correspond to both the inner and outer loops can be executed in parallel.

The FORTRAN code appeared in [4] where it was used to benchmark several supercomputers other than the CM-2 for meteorological modeling. The FORTRAN code was converted to *LISP and on TMC's System V the LISP version ran at 1.4 Gflops. This figure was extrapolated from a speed of .18 Gflops on the NCAR 8K system. Both a C* and *LISP version were coded for the CM-2 and these codes appear in [8]. The benchmark results are presented in Table 2. All cases were run on an 8K processor system using 32-bit precision floating point arithmetic. Performance figures on the CM are usually reported for a full system but as extrapolations from figures that are obtained from a smaller system. This is justified by the observation that the additional processors simply add additional flops if the size of the grid is increased without an increase in communication or the scalar part of the computation.

The first three rows of Table 2 give the performance achieved by an unoptimized C* version of the model. Calls to a lower level language (PARIS) subroutine library were required to access data from neighboring processors. For example, the line

```
CM_get_from_west_always(&pW, &p, FLEN)
```

fetches the value of the variable "p" from the processor to the west and stores it in the local varfable "pW". The grid size was specified as a power of two in order to efficiently map it onto the the CM-2 and to obtain the highest computational rates. For Case A, a model grid of 64×128 was specified to correspond to the 8K processors of the CM-2 and hence there was a physical processor for each point of the model grid giving a VP ratio of 1.

For Case B, the model grid was specified as 64×4096 which has 32 times more points than the number of physical processors giving a VP ratio of 32. Case C was a run with a 256×256 model grid. Again there are more points than the total number of physical processors, and int this configuration, each physical processor represents a 4×2 array of virtual processors for a VP ratio of eight. For both Case B and Case C, the performance achieved was 110 Mflops on the 8K

system which corresponds to 880 Mflops on a full 64K system. Although the performance improved when the VP ratios increased from one to eight and from one to 32, performance did not improve when the VP ratio increased from eight to 32. This is evidently because the communications overhead is less efficient for the 64x4096 configuration than for the 256×256 configuration.

Optimized C* Version

The optimized C* version and the unoptimized C* version are compared in the next three entries of Table 2. In the optimized version, additional subroutines were added to the lower-level subroutines library and invoked in the C* program. These additional routines allowed more efficient use of the processors by overlapping communications with computations. For example, if a value was fetched from a neighboring processor and was to be added to another variable a "fetch and add" instruction was issued rather than a "fetch" instruction followed by an "add" instruction. The line

```
CM_get_from_west_with_f_add_always (&cu, &cu, FLOAT)
```

fetches the value of "cu" from the west neighbor and adds it to the local value of "cu". The optimized C* version resulted in Mflops rate improvements of 35%, 49%, and 66% over the unoptimized version for the three cases that were run with a top rate of 1464 Mflops for the 256×256 grid.

*LISP Version

The *LISP version resulted in the highest performance rates although the corresponding model configuration 512×512 was not run for the optimized C* version. The *LISP version ran at 215 Mflops on the 8K processor system. Assuming a linear scale up in performance, a 64K system would execute at a rate of 1.7 Gflops if a VP ratio of 32 is maintained. The *LISP compiler is considered to be more highly optimized than the C* compiler.

Cray X-MP Version

The Cray X-MP version is written in FORTRAN which is highly vectorized and a listing of the code is provided in [8]. The 256×256 grid requires one million words of memory and executes at 148 Mflops on a one processor Cray X-MP. The 512×512 case requires 3.9 million words of memory and also executes at 148 Mflops. A microtasked version runs at 560 Mflops on the full Cray X-MP/48. The Cray X-MP floating point operations are performed with 64-bit precision compared with 32-bit precision on the CM-2.

Table 2
Mflops for Shallow Water Code on Thinking Machine CM-2

Case	Language	CM Size	Model Grid	VP ratio	Mflops (measured)	Mflops (for full 64K sys)
A	Unopt C*	8192	64 × 128	1	77	616
B	Unopt C*	8192	64 × 4096	32	102	816
C	Unopt C*	8192	256 × 256	8	102	816
D	Opt C*	8192	64 × 128	1	112	896
E	Opt C*	8192	64 × 4096	32	164	1312
F	Opt C*	8192	256 × 256	8	183	1464
G	*LISP	8192	512 × 512	32	215	1720

Observations

As with all computational models of multidimensional phenomena, the benchmark includes loops of lower dimension. For example, periodic boundary conditions are implemented in the FORTRAN version by one dimensional loops in which the left boundary is copied to the right as well as the top to the bottom. These loops are usually implemented on a serial or vector computer without much consideration since it is known that their contribution to the computing time is an order of magnitude below that of the two dimensional loops. However, when implemented in this manner on the Connection Machine, the total computing time (for the *LISP version) was increased by a factor of 2.5 to 4 depending on the VP ratio. The largest increase corresponded to the highest VP ratios for which the least amount of interprocessor communications is required and hence the highest computational rates are achieved. Although the boundary

computations are performed on a subset of the grid, they require the same amount of time as computations on the interior when the amount of arithmetic per point is the same. In addition, the communications that were explicitly specified by the code are not directly supported by the NEWS communication.

These reduced rates were not representative of the rates that could be obtained via alternate algorithmic and coding efforts and therefore the periodic boundary conditions were implemented using the periodic communications that are available using NEWS communication. This was achieved by simply branching around the code that explicitly implemented the boundary conditions and increasing the number of grid points to the nearest power of two.

This modification in the computation is representative of the efforts that will be necessary to implement existing codes on a parallel processor. The periodic boundary conditions could be implemented using more efficient communications methods such as the scan instruction or perhaps in microcode using the hypercube communications. These efforts are seen as typical for any significant change in computer architecture and not unlike the efforts that were required to adapt codes to early vector architectures.

4. The spectral model of the finite difference equations. A natural next step in the progression of models toward weather and climate prediction on the CM-2 is one in which the spectral method is used to solve the shallow water equations. The techniques that are used to ensure stability of the computations for the spectral method are different from those used for the finite difference method and therefore the conservative form of the equations that were used above in Section 3. for the finite difference method can be replaced by the more traditional form of the shallow water equations.

$$\frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} + v\,\frac{\partial u}{\partial y} - fv = -g\,\frac{\partial h}{\partial x}$$

$$\frac{\partial v}{\partial t} + u\,\frac{\partial v}{\partial x} + v\,\frac{\partial v}{\partial y} + fu = -g\,\frac{\partial h}{\partial y}$$

$$\frac{\partial h}{\partial t} + u\,\frac{\partial h}{\partial x} + v\,\frac{\partial h}{\partial y} + h\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0$$

where $g = 9.8\ m/sec^2$, $f = 10^{-4}/sec$, and $\bar{h} \approx 1 \times 10^4$.

These equations are solved subject to periodic boundary conditions using the computational approach that is outlined below.

1. The spectral representations of u, v, and h are computed using a multiple FFT program developed by TMC [5]
2. The upper third of the coefficients are truncated to significantly reduce (but not completely eliminate) aliasing. This approach was also used in [1].
3. The spatial derivatives u_x, u_y, v_x, v_y, h_x, and h_y in physical space are computed by formal differentiation of the spectral representations computed in 1. followed by the inverse FFT.
4. The time derivatives u_t, v_t, and h_t are computed by substituting the spatial derivatives into the right hand side of the shallow water equations given above.
5. The solution is advanced to the next time level using "leap frog" time differencing.

This approach is in fact called the pseudo-spectral method which is a significant variant of the spectral method. A comparison of three different methods for solving differential equations is given in [1]. The steps 1 through 5 were implemented in FORTRAN/CAL on the Cray X-MP/48 and in *LISP by Oliver McBryan on the CM-2. The codes were optimized for the particular machine rather than line for line translations from one machine to the other. The results are given below in Table 3.

Table 3
Spectral Shallow Water Equations Model
Cray X-MP/48 and Thinking Machine CM-2

Resolution	Mflops			Time/Grid point		
	X-MP (1 proc)	X-MP (4 proc)	CM2 (65K proc)	X-MP (1 proc)	X-MP (4 proc)	CM2 (65K proc)
256 × 256	125	397	601	2.41	.75	.77
512 × 512	129	476	806	2.59	.70	.64
2048 × 2048	-	-	1162	-	-	.53

NOTE: The X-MP/48 time is for 64 bit precision and the CM-2 time is for 32 bit precision.

Observations

a) Most of the computing time is in subroutine FFT991 which is a fast Fourier transform (FFT) written in Cray machine language by Clive Temperton at ECMWF. Because of the difficulty in converting this code to a multitasked version, the use of four processors was made possible by separating the

rectangle into four subdomains and creating four separate tasks which called FFT991 on these subdomains. There is some overhead in this operation since the length of the vectors is divided by four.

b) The time for an ordered transform on the CM-2 was prohibitive because of the cost of the global communication required for bit reversal. An ordered transform on the CM-2 takes two to three times as long as an unordered transform [6]. However, since the order of the FFT in spectral space is not particularly relevant to the solution which is presented in physical space, it is not necessary to order the FFT. The spectral representations of the derivatives were computed by multiplying the bit reversed coefficients by wave numbers that were also in bit reversed order. The inverse FFT then accepts the bit reversed coefficients and provides the derivatives on the grid which is ordered in physical space.

c) The spectral code makes extensive use of the FFT and indeed the performance on the shallow water equations is about the same as the performance on the FFT. The vectorization and parallelization of the FFT is nontrivial [9], [10], particularly when compared with the finite difference method. Nevertheless methods have been developed over the years that enable the FFT to perform at near peak rates and FFT codes have been repeatedly groomed for optimum performance on the Cray X-MP. This fact should be kept in mind when comparing the performance of the computers since the FFT on the CM-2 is both a new code and a new implementation of the FFT. It was written by the staff at Thinking Machines Corporation [5] under the supervision of Lennart Johnsson and was provided to NCAR on a beta test basis.

d) Tempertons FFT on the Cray X-MP/48 is for real sequences and requires about half the time required for a complex transform. On the other hand the TMC FFT is for complex sequences. This apparent disadvantage was for all practical purposes overcome by combining two real variables into a single complex variable which could then be transformed concurrently [6]. Although the complex transform would have to be postprocessed to recover the real transforms, it was not necessary for the reasons already mentioned in b) above. It was simply sufficient to multiply each coefficient by its corresponding wave number followed by an inverse transform which yielded the derivatives of two distinct variables as the real and imaginary parts respectively. This technique works best if the number of variables is even

which unfortunately is not the case here were the variables are u, v, , and P. However it is possible to extract two different derivatives, say with respect to both x and y with one complex transform [6].

e) CM FORTRAN is currently under beta test at several institutions and is expected to make the CM-2 more attractive to the scientific computing community.

5. Conclusions. A status report has been provided on a continuing project; namely, to determine the suitability of massively parallel processing to the computational needs of the atmospheric science community. As of this report, two data points have been obtained. The first is 1.7 Gflops for an elementary weather model in Cartesian coordinates using the finite difference method and the second is about 1 Gflop for the same model using the spectral method. These results are encouraging and suggest the need for further investigation into the suitability of massively parallel processing and in particular the suitability of SIMD architectures. Optimum codes for the CM-2 were compared with codes that were optimized for the Cray X-MP/48 rather than a comparison of Gflops for the two machines on the same code.

The Gflops that were obtained on the CM-2 were not obtained on the first runs and indeed more than a modest effort was required by individuals with considerable knowledge about parallel computing and the machine itself. This effort must be weighed against any increase in flops/dollars. A substantial increase in performance is necessary to justify the effort that is required to convert a major production code to a new architecture. This study is a beginning and much work remains to be done before a definitive statement can be made regarding the future relationship of SIMD and computing in the atmospheric and related geosceinces. In particular more data points should be obtained for more complex problems, perhaps selected from Table 1 in section 1.

ACKNOWLEDGEMENTS

We are grateful to the staff at Thinking Machines Corporation for their support of this project and in particular we wish to thank Dr. Lennart Johnsson and his staff for their counsel and for providing the CM FFT under beta test. We would also like to thank Dr. John Richards (TMC) for developing the C* version of the

finite difference model and Dr. Oliver McBryan at the University of Colorado for his extensive support of this project including the development of *LISP codes for both the spectral and finite difference methods.

REFERENCES

[1] G.L. Browning, J.J. Hack and P.N. Swarztrauber, *A comparison of three numerical methods for solving differential equations on the sphere, The Monthly Weather Review,* 117(1989), pp. 1058-1075.

[2] *Connection Machine Model CM-2 Technical Summary,* Thinking Machines Technical Report HA87-4, April, 1987.

[3] D. Hillis, *The Connection Machine,* MIT Press, Cambridge, MA., 1985.

[4] G.-R. Hoffman, P. N. Swarztrauber, and R. A. Sweet, *Aspects of using multiprocessors for meteorological modeling,* in: Multiprocessing in Meteorological Models, G.-R. Hoffmann and D. F. Snelling, eds., Springer-Verlag, New York, 1988.

[5] L. Johnsson, R. Krawitz, and R. Frye, *Computing radix-2 FFT on the Connection Machine,* Technical Report, Thinking Machines Corp., Cambridge, MA, 02142, 1989.

[6] O.A. McBryan, *Connection Machine application performance,* Rpt. CU-CS-434-89, Department of Computer Science, University of Colorado at Boulder, Boulder, Colorado, 80309, 1989.

[7] R. Sadourney, *The Dynamics of finite-difference models of the shallow-water equations,* Journal of Atmospheric Sciences, 32, (1975), p680-689.

[8] R.K. Sato, and P.N. Swarztrauber, *Benchmarking the Connection Machine,* Proceedings of the IEEE Supercomputer Conference, Orlando Florida, November 14-18, 1988, pp. 304-309.

[9] P.N. Swarztrauber, *Vectorizing the FFT's,* In: *Parallel Computations,* G. Rodrigue, ed., Academic Press, New York, 1982.

[10] P.N. Swarztrauber, *Multiprocessor FFTs, Parallel Computing.* 5 (1987), pp. 197-210.