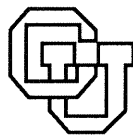


**Scaling Performance of the Shallow Water
Equations on the Suprenum-1 Supercomputer**

Oliver A. McBryan

CU-CS-637-92

December 1992



University of Colorado at Boulder

DEPARTMENT OF COMPUTER SCIENCE

SCALING PERFORMANCE OF THE SHALLOW WATER EQUATIONS ON THE SUPRENUM-1 SUPERCOMPUTER

Oliver A. McBryan*

Department of Computer Science
University of Colorado
Boulder, CO 80309

Abstract.

We describe the implementation of a fluid dynamical benchmark on the 256 node SUPRENUM-1 parallel computer. The benchmark, the Shallow Water Equations, is frequently used as a model for both oceanographic and atmospheric circulation. We describe the steps involved in implementing the algorithm on the SUPRENUM-1 and we provide details of performance.

Optimal SUPRENUM performance requires algorithms that may be compiled into vector instructions with long vector length, and as with many other MIMD systems, relatively few communication operations. For such algorithms the system delivers a very impressive fraction of its theoretical peak rate. SUPRENUM software is excellent, including communication facilities and a fully vectorizing compiler for Fortran 77 which was used in this study.

We have measured 5.33 Mflops (64-bit arithmetic) for single node performance, and 1280 Mflops aggregate performance with 256 nodes, at efficiencies up to 95%. This compares well with vector and MIMD supercomputers and shows that SUPRENUM was among the fastest MIMD computers during 1992. Performance of 1530 Mflops was measured for the same algorithm on the CRAY YMP/8, and 543 Mflops was measured on the 128-node Intel iPSC/860. The SIMD Thinking Machines CM-200 delivers 5.25 Gflops (64-bit) and 8.09 Gflops (32-bit) for the benchmark. We also discuss the influence of physical cluster interconnection topology and asynchronous communication on SUPRENUM performance.

* Research supported by the Air Force Office of Scientific Research, under grant AFOSR-89-0422 and by NSF Grand Challenge Application Group grant ASC-9217394.

1. INTRODUCTION

The Shallow Water Equations are a standard model for atmospheric and oceanographic processes. Implementations of the algorithm have been used as benchmarks for vector and parallel supercomputer performance for many years¹⁻⁵. The Shallow Water algorithm is very memory intensive, involving 14 variables per grid point, and accesses these using nine-point stencils, non-linear expressions and essential divisions. The combined effect provides a decidedly non-trivial test of any computer system. We have recently implemented the benchmark on the 256 node SUPRENUM-1 MIMD parallel supercomputer and report on the results in this paper - see also⁵⁻⁷.

The tests were run on the SUPRENUM-1 hardware at the GMD in St-Augustin, Germany, which was running the Peace 3.0 operating system software. The Shallow Water code ran on a single node using 64-bit arithmetic at 5.33 Mflops and on a 256 node system at up to 1280 Mflops. Performance of 5.33 Mflops per node was quite impressive, especially as the code was not explicitly vectorized in any way.

In fact this single-node performance exceeds the typical performance we have seen for the same algorithm on the Intel iPSC/860 systems, despite the fact that the latter system's nodes have several times the peak performance of SUPRENUM's. We conclude that the SUPRENUM compiler is doing an excellent job of locating vectorizable statements, and in generating efficient pipelined vector instructions to implement such statements. Numerical results agreed to high precision with those from other machines. We expect that even higher per-node performance could be achieved by utilizing explicit optimizations, and by coding computationally intensive segments using SUPRENUM Fortran's array extensions (Fortran 90).

The multi-node performance compares well with the iPSC/860 hypercube where an optimized Fortran version of the Shallow Water Equations runs at 543 Mflops (64-bit precision) on 128 processors⁵, with the CRAY XMP which solves the equations at a rate of 560 Mflops on 4 processors, and with the CRAY YMP where 1530 Mflops has been attained using 8 processors. The SIMD Thinking Machines CM-200 however is substantially faster, delivering 5.25 Gflops (64-bit) and 8.09 Gflops (32-bit)⁸. Some single-node SUPRENUM measurements reported here are slightly different from those reported a year ago⁵. This is because of variations due to compiler or operating system changes. In that paper measurements were restricted to only 16 nodes as only a SUPRENUM prototype consisting of a single cluster was available. The main content of this paper is the measurement of performance on up to 256 nodes and the demonstration of good scaling behavior of the complete system.

2. THE SUPRENUM-1 SUPERCOMPUTER

The German SUPRENUM-1 computer couples up to 16 processor clusters with a network of 200 Mbit/sec busses. The busses were intended to be arranged as a rectangular grid with 4 horizontal and 4 vertical busses, although other configurations have also been employed (see below). Each cluster consists of 16 processors connected by a fast bus, along with I/O devices for communication to the global bus grid and to disk and host computers. There can be a dedicated disk for each cluster. Individual processors can deliver up to 20 Mflops (64-bit chained) or 10 Mflops (64-bit unchained) of computing power and support 8 Mbytes of memory, upgradable to 32 Mbytes. The high bandwidth of the bus network makes this an interesting machine for a wide range of applications, including those requiring long-range communication. No more than four communication steps are ever required between remote nodes (with four steps needed only if both a horizontal and a vertical bus must be traversed).

While SUPRENUM clusters are well defined by their interconnection bus, the connectivity between clusters is modifiable by rewiring the connections appropriately. In principle this is simple, although in practice it turns out to be a major undertaking because there are severe physical constraints on the length of the buses involved, plus the fact that each bus must actually connect to form a ring. Each ring must visit from 4 to 6 clusters. During 1991, the SUPRENUM-1 clusters were connected in a simple ring (actually four parallel rings, although it was not possible to fully utilize the parallelism). In January 1992 the SUPRENUM-1 was re-configured as a full double matrix of busses. In June 1992 the SUPRENUM-1 topology was changed to provide a topology where each cluster has a direct connection to every other one so that no all communication operations required at most three steps.. The results reported on in this paper deal primarily with the latter interconnection network as this provided the best overall performance from the three interconnection schemes which were studied.

SUPRENUM software is characterized by the best support for MIMD scientific applications to be found among the various distributed memory MIMD vendors. The effort invested in development of libraries of high-level grid and communication primitives greatly eases the effort of moving applications to the computer, and also provides substantial high-level portability to other systems, since the communication library can be implemented in terms of low level primitives on any distributed system.

The first 16-processor prototype system was delivered in 1989 and the first operational 256 processor system became available in November 1991. The full system has a 5 Gflops peak rating and should have high realizable efficiency in appropriate applications, namely those where communication is relatively infrequent and where long vector lengths predominate.

3. THE SHALLOW WATER EQUATIONS BENCHMARK

As an example of the current capabilities of the SUPRENUM system we describe the implementation of a standard two-dimensional atmospheric model - the Shallow Water Equations - on the SUPRENUM-1. These equations provide a primitive but useful model of the dynamics of the atmosphere. Because the model is simple, yet captures features typical of more complex codes, the model is frequently used in the atmospheric sciences community to benchmark computers^{1,2}. Furthermore, the model has been extensively analyzed mathematically and numerically^{9,10}.

The Shallow Water Equations, without a Coriolis force term, take the form

$$\begin{aligned}\frac{\partial u}{\partial t} - \zeta v + \frac{\partial H}{\partial x} &= 0, \\ \frac{\partial v}{\partial t} - \zeta u + \frac{\partial H}{\partial y} &= 0, \\ \frac{\partial P}{\partial t} + \frac{\partial Pu}{\partial x} + \frac{\partial Pv}{\partial y} &= 0,\end{aligned}$$

where u and v are the velocity components in the x and y directions, P is pressure, ζ is the vorticity: $\zeta = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$ and H , related to the height field, is given by: $H = P + (u^2 + v^2)/2$. It is required to solve these equations in a rectangle $a \leq x \leq b, c \leq y \leq d$. Periodic boundary conditions are imposed on u, v , and P , each of which satisfies $f(x+b, y) = f(x+a, y)$, $f(x, y+d) = f(x, y+c)$.

A scaling of the equations results in a slightly simpler format. Introduce mass fluxes $U = Pu$ and $V = Pv$ and the potential velocity $Z = \zeta/P$, in terms of which the equations reduce to:

$$\begin{aligned}\frac{\partial u}{\partial t} - ZV + \frac{\partial H}{\partial x} &= 0, \\ \frac{\partial v}{\partial t} + ZU + \frac{\partial H}{\partial y} &= 0, \\ \frac{\partial P}{\partial t} + \frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} &= 0.\end{aligned}$$

4. DISCRETIZATION

We have discretized the above equations on a rectangular staggered grid with periodic boundary conditions. The variables P and H have integer subscripts, Z has half-integer subscripts, U has integer and half-integer subscripts, and V has half-integer and integer subscripts respectively.

Initial conditions are chosen to satisfy $\vec{\nabla} \cdot \vec{v} = 0$ at all times. We time difference using the Leap-frog method. We then apply a time filter to avoid weak instabilities inherent in the Leap-frog scheme:

$$F^{(n)} = f^{(n)} + \alpha (f^{(n+1)} - 2f^{(n)} + f^{(n-1)}),$$

where α is a filtering parameter. The filtered values of the variables at the previous time-step are used in computing new values at the next time-step. For a complete description of the discretization we refer to¹.

5. SERIAL FORTRAN IMPLEMENTATION

The Fortran code implementing the above algorithm involves a 2D rectangular grid with variables: $u(i,j)$, $v(i,j)$, $p(i,j)$, $z(i,j)$, $psi(i,j)$, $h(i,j)$. There are three main loops, two corresponding to the Leap-frog time propagation of various quantities, and one for the filtering step. Execution of these three loops completes a single time step, which is then repeated until the desired temporal simulation interval has been achieved. A typical code sequence, used in the updating of the U , V and P variables, is:

```
do 20 j=1,n
do 20 i=1,m
  unew(i+1,j) = uold(i+1,j)
    + tdt8*(z(i+1,j+1)+z(i+1,j))*(cv(i+1,j+1)+cv(i,j+1)+cv(i,j)+cv(i+1,j))
    - tdt8*(h(i+1,j)-h(i,j))
  vnew(i,j+1) = vold(i,j+1)
    - tdt8*(z(i+1,j+1)+z(i,j+1))*(cu(i+1,j+1)+cu(i,j+1)+cu(i,j)+cu(i+1,j))
    - tdt8*(h(i,j+1)-h(i,j))
  pnew(i,j) = pold(i,j) - tdt8*(cu(i+1,j)-cu(i,j)) - tdt8*(cv(i,j+1)-cv(i,j))
20 continue
```

Each such loop is followed by code to implement the periodic boundary conditions. In the above case, the corresponding boundary code takes the form:

```
do 21 j=1,n
  unew(1,j) = unew(m+1,j)
  vnew(m+1,j+1) = vnew(1,j+1)
  pnew(m+1,j) = pnew(1,j)
21 continue
```

Note that there are such loops for both the horizontal and vertical boundaries, and in addition some corner values are copied as single items.

Excluding the boundary computations, the three major loops in a time step involve 65 arithmetic operations per grid point. Furthermore 14 physical variables must be stored per grid point, which significantly limits the largest grid size that can be accommodated in a single node.

6. SUPRENUM IMPLEMENTATION

To speed the implementation effort we decided to test the idea of porting a generic MIMD parallel version of the Shallow Water Equations to the SUPRENUM-1. The work was based on a parallel code developed by McBryan and Pozo^{8,11}. The code was developed for a generic class of MIMD parallel computers, based on the assumption of a single process per node model. The code was developed and tested using a simulator for the generic model developed previously^{12,13}. The simulator supports versions of the Intel iPSC communication protocols.

SUPRENUM supports a library interface allowing both Intel iPSC1 and iPSC2 communication interfaces to be utilized. It suffices to declare the main program of both the host and node processes to be SUPRENUM **tasks**, while the rest of each program may remain as a pure Intel iPSC program. This approach greatly eased code modifications that would have been required to develop a complete SUPRENUM-1 implementation from scratch. In fact the code was ported and fully working within hours. The program ran immediately and gave correct results on the first try. This demonstrates the advantages of developing MIMD codes initially using simulators, and transferring to hardware only when the simulations are running correctly.

Since the code involves rectangular grid arrays, and a nine-point stencil, the parallelization of the code is straightforward. A logical mapping of the processors to a two dimensional array is selected. Thus if $P = p_x p_y$, is a factorization of the number of processors P , then we regard the processors as arranged in a $p_x \times p_y$ logical grid. The large arrays representing physical variables (u, v , etc.) are then decomposed into equal sized blocks, with one block assigned to each processor. For simplicity we assume that the x and y grid dimensions are exact multiples of the corresponding processor numbers p_x and p_y . Each

such block is then stored in an array of the same shape, but which has an extra boundary row or column provided on each of the four sides. These extra boundary points are used to maintain copies of the true (i.e. interior) boundary points of the four neighboring processors. The three main loops of the time step are decomposed into equivalent loops performed by each processor on the interior points of the block assigned to that processor. Prior to each loop, the boundary values are updated by exchanging appropriate values between neighboring processors, following a synchronization to ensure that all neighbors have completed changes. Such exchanging generally requires communication which was implemented by communicating large packets for each of the four sides of a block.

There is an essential simplification that occurs in the case that either p_x or p_y is 1 - in which case the logical rectangular processor array reduces to a line of processors. In this case two of the four communications required within each main loop are not needed, reducing substantially the communication overhead. As mentioned previously, the Shallow Water code uses periodic boundary conditions in each dimension. Normally periodic boundary conditions require copying data between processors at opposite edges of the processor array. In the case that one or other of p_x or p_y is 1, the periodic boundary condition in the corresponding dimension may be implemented by in-memory copying, rather than by communication.

A final optimization of the communication structure was required to get the peak performance. Before each of the main loops in the algorithm, the boundary data for the various physical variables (P, U, V, Z, H) used in that loop need to be copied from neighboring processors. Typically two or three variables are needed from a specific direction, although the number needed may depend on the direction. Because of the high communication startup cost of SUPRENUM-1 (at least 2 msecs), it is essential to limit the number of individual communication requests. This was accomplished by packaging several communications of different physical variables in a single direction into one large communication package. For some steps this reduced startup overhead by a factor of three. In the final implementation we also replaced the Intel iPSC communication calls for this one exchange operation by explicit calls to SUPRENUM Fortran equivalents, thereby saving an extra copying of each data array to a communication buffer. SUPRENUM Fortran supports explicit communication operations using a standard Fortran I/O control list syntax.

There is potential in the Shallow Water Equations to overlap communication with computation, provided the underlying hardware supports asynchronous communication modes. In this case one would begin each major loop by an asynchronous exchange of boundary data. Following this one executes the main body of the loop, however iterating only over the "interior points" of the subgrid. It is then necessary to await completion of the exchange operation, after which the the loop iteration may be completed on the outermost rows and columns. In principle such an approach can yield 100% computational efficiency - i.e. communication effects become negligible. We implemented such an algorithm on SUPRENUM-1. However due to inherent design aspects of the PEACE

operating system we were unable to effectively use asynchronous communication in the current version of PEACE.

7. PERFORMANCE RESULTS: SUPRENUM-1

All measurements were performed on a 256-processor SUPRENUM-1 system at the GMD, in Schloss Berlinghoven, Germany. The Shallow Water Code was exactly the standard sequential code, modified only to take account of communication. No attempt was made to introduce Fortran 90 vectorization constructs, or to otherwise adapt the code to known features of the SUPRENUM compiler. The code was compiled with both the vectorizer and optimizer switches on.

Because SUPRENUM nodes are vector processors, there is a substantial advantage to arranging the subgrids in each node such that the grid columns are as long as possible. In practice, Fortran columns longer than about 1024 words are not an advantage. This is because the vector registers are limited to a total of 7K words, and Shallow Water requires 7 registers for efficient code generation.

In order to maximize computational efficiency (by minimizing communication words sent per Mflop), it is desirable to solve as large a problem as will fit in each node. This turns out to be a problem with 32K grid points which consumes approximately 6 MBytes of node memory. All measurements presented here utilize subgrids of maximal size, although their rectangular shape may vary. We maximize both vector performance and computational efficiency on a node by using a 32×1024 subgrid in each processor. To indicate the importance of preserving a long vector length we note that performance on a single node goes from 2.69 Mflops on a 128×256 grid to 5.33 Mflops on a 32×1024 grid, essentially a factor 2 improvement (see Table 1 below).

As discussed earlier, the number of communications per node can be reduced by a factor of two by choosing a one-dimensional processor grid, which may be aligned with either the X or Y axes. If the processors are in a line in the X direction, then the communication packets will be of size 1024 words (Y dimension of the subgrids) per variable, while if aligned along the Y axis, only 32 words are communicated per physical variable.

More generally we can expect lower performance as the subgrids tend towards a square shape, such as 128×256 , due to the shorter vector lengths. Also using fully two-dimensional processor grids such as a 16×16 grid will double the number of communications per node, resulting in poorer performance. All of these phenomena are illustrated in the measured results.

The final effect which we have studied is the influence of cluster interconnection topology on performance. The SUPRENUM-1 has been interconnected in three different ways as described in section 2 - ring, full matrix and full interconnect, and we have measured Shallow Water Equations performance in all cases. There is a significant dependence

of performance on the topology used. For example the worst-case efficiency measured with the double matrix topology was 39% while with the full interconnect topology, the worst case efficiency is 70%. On the other hand for the most efficient (linear) cases, performance with the full connection topology is slightly worse, dropping from 96% to 94% efficiency. Clearly the advantages of the full interconnect topology outweigh the disadvantages. We give only the measured data for the full interconnect topology.

We present the measured results in Tables 1-4. The tables indicate the number of processors P , their arrangement as a logical $P_x \times P_y$ rectangular processor array, the computational domain size $M_x \times M_y$, the resulting computational efficiency and the Mflops generated. The computational efficiency in all cases is defined as

$$E = T_{best}(1) / (P T(P)),$$

where $T(P)$ is the solution time with P processors and $T_{best}(1)$ is the best possible single-node performance with a subgrid of the same size but optimal shape.

Table 1 presents the effect of varying the grid shape in a single node. This demonstrates clearly the importance of maximizing vector length. Indeed the almost square 256×128 grid provides only 77% of the performance of the elongated 32×1024 grid with the same number of grid points. At the other extreme, the 1024×32 grid delivers only 43% of the performance of the 32×1024 grid.

P	P_x	P_y	M_x	M_y	Efficiency	Mflops
1	1	1	1024	32	0.430	2.29
1	1	1	256	128	0.768	4.09
1	1	1	128	256	0.883	4.71
1	1	1	64	512	0.955	5.09
1	1	1	32	1024	1.000	5.33

Table 2 describes the performance of Shallow Water on grids of optimal shape for the system. Each node contains an optimal 32×1024 grid and the processors are arranged in a line parallel to the Y direction in order to minimize communication.

P	Px	Py	Mx	My	Efficiency	Mflops
1	1	1	32	1024	1.000	5.33
2	1	2	32	2048	0.949	10.11
4	1	4	32	4096	0.948	20.21
8	1	8	32	8192	0.948	40.41
16	1	16	32	16384	0.946	80.61
32	1	32	32	32768	0.945	161.10
64	1	64	32	65536	0.947	322.74
128	1	128	32	131072	0.945	644.87
256	1	256	32	262144	0.940	1280.37

Table 3 is similar except that the processors are arranged in a line parallel to the X axis, resulting in more square grids, and slightly increased communication cost.

P	Px	Py	Mx	My	Efficiency	Mflops
1	1	1	32	1024	1.000	5.33
2	2	1	64	1024	0.940	10.01
4	4	1	128	1024	0.934	19.89
8	8	1	256	1024	0.932	39.68
16	16	1	512	1024	0.931	79.29
32	32	1	1024	1024	0.919	156.76
64	64	1	2048	1024	0.920	313.24
128	128	1	4096	1024	0.912	621.83
256	256	1	8192	1024	0.905	1234.73

In Table 4, we compare the effect of varying the shape of the processor grid for 256 node computations. Each node is maintained at the optimal 32×1024 grid. The almost square 4096×2048 grid on a 128×2 processor array is seen to deliver 1117 Mflops. The alternative of creating a near square global grid from 256 near square subgrids would have yielded about 25% less performance as indicated by Table 1.

TABLE 4: 256-NODE PERFORMANCE AS FUNCTION OF PROCESSOR GRID

P	Px	Py	Mx	My	Efficiency	Mflops
256	256	1	8192	1024	0.905	1234.73
256	128	2	4096	2048	0.820	1117.40
256	64	4	2048	4096	0.709	967.34
256	32	8	1024	8192	0.815	1110.72
256	16	16	512	16384	0.738	1007.87
256	8	32	256	32768	0.866	1180.16
256	4	64	128	65536	0.883	1202.52
256	2	128	64	131072	0.885	1206.58
256	1	256	32	262144	0.940	1280.37

8. A COMPARISON OF CRAY, CM-200, iPSC/860 AND SUPRENUM-1

We have compared the SUPRENUM performance with that on the CRAY XMP and YMP computers, on the Thinking Machines CM-200 and on the Intel iPSC/860 hypercube. Results are presented in Table 5.

The performance on a single processor of a CRAY-XMP was 148 Mflops. The CRAY-XMP4/8 executed the Shallow Water Equations at 560 Mflops using 4 processors on a 512×512 grid, the largest that could be handled directly (i.e without SSD coding). The CRAY-YMP with 8 processors runs the Shallow Water Equations at 1,530 Mflops on a 512×512 grid.[†] The Connection Machine CM-200 results are described in more detail in⁸, while the Intel iPSC/860 results are reported in more detail in^{8,11}.

TABLE 5: COMPARISON TO OTHER ARCHITECTURES

Machine	Processors	Grid Size	Mflops
CM-200 (64-bit)	2048	16M	5249
CM-200 (32-bit)	2048	32M	8086
CRAY-XMP	4	256K	560
CRAY-YMP	8	256K	1530
Intel iPSC/860	128	2M	543
SUPRENUM-1	256	8M	1280

[†] CRAY measurements were performed by Dr. R. Sato, National Center for Atmospheric Research, Boulder, CO.

To relate SUPRENUM-1 to other systems, it is fair to say that for most of 1992 this system was among the most powerful available MIMD systems. However the SIMD CM-200 is far more powerful. With the arrival of vector nodes for the Thinking Machines CM-5 computer, SUPRENUM will no longer be in this position in late 1992. The performance measurements also should be qualified by the cost per Mflops of the different systems, which we have not considered in detail. However it does appear that SUPRENUM loses much of its performance advantage relative to the iPSC/860 if pricing is considered. This is due to the fact that the SUPRENUM nodes involve essentially more complex hardware (e.g. vector nodes) than the iPSC/860. The SUPRENUM node design was formulated long before the much cheaper i860 processor appeared.

9. CONCLUSIONS

The SUPRENUM-1 system is shown to deliver excellent performance per node for problems which are vectorizable and which also have a long vector length. This performance scales well to large systems. While communication overheads are greater than on many competing systems (e.g. Intel iPSC/860), this is more than counterbalanced by the higher achievable node performance. With the successful demonstration of the 256-node SUPRENUM-1, the SUPRENUM project may be regarded as a scientific success. Based on this success of the initial SUPRENUM-1 prototype, it is unfortunate that a successor system based on newer technology is not being developed.

ACKNOWLEDGEMENTS

We would like to thank the GMD for providing access to the SUPRENUM-1, We thank H. Bast of SUPRENUM GmbH for helpful comments.

References

1. G.-R. Hoffmann, P.N. Swarztrauber, and R.A. Sweet, "Aspects of using multiprocessors for meteorological modeling," in *Multiprocessing in Meteorological Models*, ed. D. Snelling, pp. 126-195, Springer-Verlag, Berlin, 1988.
2. O. McBryan, "New Architectures: Performance Highlights and New Algorithms," *Parallel Computing*, vol. 7, pp. 477-499, North-Holland, 1988.
3. O. McBryan and R. Pozo, "Performance Evaluation of the Myrias SPS-2 Computer," CS Dept Technical Report CU-CS-505-90 (to appear in *Concurrency: Practice and Experience*), University of Colorado, Boulder, 1990.

4. O. McBryan and R. Pozo, "Performance Evaluation of the Evans and Sutherland ES-1 Computer," CS Dept Technical Report CU-CS-506-90, University of Colorado, Boulder, 1990.
5. O. McBryan, "A Comparison of the Intel iPSC860 and SUPRENUM-1 Parallel Computers," *University of Colorado Tech. Report CU-CS-499-90 and Supercomputer*, vol. 41, no. 1, pp. 6-17, 1991.
6. O. McBryan, "Performance of the Shallow Water Benchmark on the SUPRENUM-1 Parallel Supercomputer," in *Leistungsmessungen für technisch-wissenschaftliche Anwendungen auf dem SUPRENUM-System*, ed. H. Mierendorff and U. Trottenberg, Arbeitspapiere der GMD 624, GMD, Sankt-Augustin, Germany, March 1992.
7. O. McBryan, "Performance of the Shallow Water Benchmark on the SUPRENUM-1 Parallel Supercomputer with the Full Bus Interconnection Network," in *Ergänzende Leistungsmessungen für technisch-wissenschaftliche Anwendungen auf dem SUPRENUM-System*, ed. H. Mierendorff and U. Trottenberg, Arbeitspapiere der GMD 669, GMD, Sankt-Augustin, Germany, July 1992.
8. O. McBryan and R. Pozo, "Performance of the Shallow Water Equations on the Intel iPSC/860 Computer," CS Dept Technical Report, University of Colorado, Boulder, 1991.
9. R. Sadourny, "The dynamics of finite difference models of the shallow water equations," *JAS*, vol. 32, pp. 680-689, 1975.
10. G.L. Browning and H.-O. Kreiss, "Reduced systems for the shallow water equations," *JAS*, vol. 44, 1987.
11. R. Pozo, "Performance Modeling of Parallel Architectures for Scientific Computing," PhD Thesis, Department of Computer Science, University of Colorado at Boulder, 1991.
12. O. McBryan and E. Van de Velde, *Hypercube Algorithms and Implementations*, SIAM J. Sci. Stat. Comput., 8, pp. 227-287, 1987.
13. O. McBryan, "Software Issues at the User Interface," in *Frontiers of Supercomputing II: A National Reassessment*, ed. W.L. Thompson, University of Colorado CS Dept. Tech Report CU-CS-527-91 and MIT Press, 1992, to appear.

**ANY OPINIONS, FINDINGS, AND CONCLUSIONS OR RECOMMENDATIONS
EXPRESSED IN THIS PUBLICATION ARE THOSE OF THE AUTHOR(S) AND DO NOT
NECESSARILY REFLECT THE VIEWS OF THE AGENCIES NAMED IN THE
ACKNOWLEDGMENTS SECTION.**