Parallel Computational Fluid Dynamics '92
R. Pelz, A. Ecer & J. Hauser, eds.
Elsevier Press, Netherlands

Direct Numerical Simulation of Turbulence on the Connection Machine

J. Blair Perot

Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA

Abstract

Detailed performance measurements of the direct numerical simulation of turbulence on the Connection Machine 2 are presented. These are compared to similar simulations being performed on the Cray Y-MP. The current and future utility of the Connection Machine as a tool in the direct numerical simulation of turbulence is discussed.

1. INTRODUCTION

The aim of this paper is to evaluate the performance of the Connection Machine 2 (CM-2) in the context of the direct numerical simulation (DNS) of turbulence. The study of turbulence through the use of direct computer simulations has, from its inception in the early 1980's, accounted for a significant percentage of scientific supercomputer usage. The fundamental questions that can be answered about turbulence are closely related to the current level of supercomputer performance. In fact, turbulence has been identified as one of the Grand Challenge problems [1] that could benefit significantly from increased supercomputer performance and, in particular, from massively parallel computers.

However, the issue of whether massively parallel machines can actually fulfill their performance expectations when it comes to turbulence simulation is still an open question. It is now fairly apparent that unlike vector supercomputers, the performance of massively parallel computers is not closely related to their peak performance. Instead, massively parallel computers are inevitably dominated by communication overhead. Their performance varies wildly from application to application, depending on how well the communication patterns of the application match the communication patterns implied by the particular architecture. For instance, the architectures of the CM-2 and Intel Hypercube are optimized for vastly different communication patterns. As a result, stencil type operations work very well on the Connection Machine, and Fast Fourier Transforms (FFT's) with large intermediate data rearrangement work well on the Intel Hypercube, but not vice versa.

Direct numerical simulation of anything more complicated than isotropic decaying turbulence typically involves a number of very different communication patterns and data structures. Whether these all can be mapped fairly efficiently to the Connection Machine is an interesting question. Any single bottleneck could dramatically effect the

performance of the simulation as a whole. Possible bottlenecks that will be investigated include the effect of implementing boundary conditions on a SIMD machine, and the impact of regular but long distance communication. So, despite the fact that turbulence simulation involves massively parallel data (on the order of 10^6 to 10^7 nodes), has no load balancing problems, and involves only regular communication, it is not a foregone conclusion that the CM-2 is the the supercomputer of choice. Only through the actual testing of an existing DNS code, and comparison to current vector supercomputer performance (as represented by the Cray Y-MP) can the potential of the CM-2 for turbulence simulations be evaluated.

2. NUMERICAL METHOD

The purpose of this section is not to present a new numerical method, although the method does differ in some fundamental ways from classical DNS methods. Instead, the purpose is to reveal the variety of solution algorithms, communication patterns and data structures that are used. An understanding of the basic numerical scheme will also help to put the various performance timings into the proper context.

2.1 Spatial Discretization

The spatial discretization of the incompressible Navier-Stokes equations is a primative variable, finite volume method on a staggered mesh. It is very much in the spirit of the discretization first introduced by Harlow and Welch [2]. The mesh is cartesian but not necessarily uniform, and in the results that follow one of the three directions will be non-uniform. A two dimensional representation of the spatial discretization is shown in Figure 1.

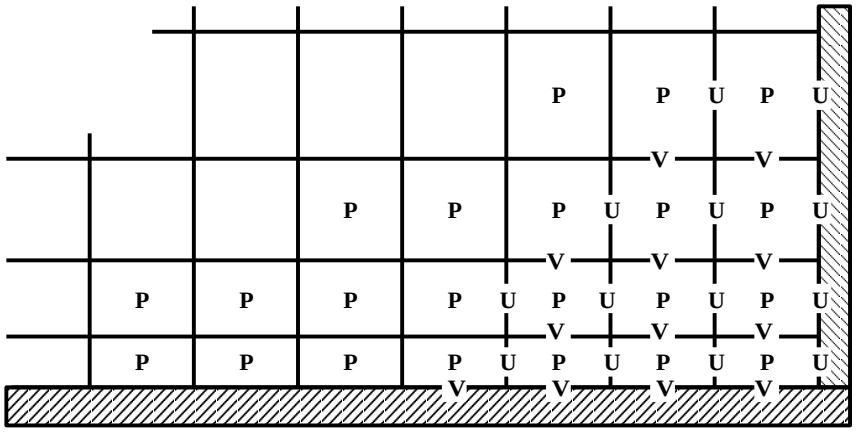

FIGURE 1. Locations for the discrete velocity and pressure variables on a 2-D staggered mesh.

The finite volume discretization is a departure from the traditional use of spectral methods in direct turbulence simulations. It is motivated by the desire to implement more complicated boundary conditions (and eventually more complicated geometries). Fortuitously, this also makes the method far more amenable to implementation on the CM-2 (as the timing results will show). To date, there is still no convincing evidence that the accuracy of spectral methods is far superior to that of second order methods, *when the grid spacing is at the limit of resolving the flow.* And in any case, this is a secondary issue in the context of this paper.

2.2 Temporal Discretization

The temporal discretization advances the nonlinear convective terms with a second order Adams-Bashforth method. An explicit method considerably simplifies the advancement of the non-linear terms, but also imposes a stability limit on the CFL number. The stability limit is not overly restrictive and corresponds roughly to the condition that the temporal accuracy match that of the spatial accuracy in a Taylor's hypothesis sense. The diffusive terms are advanced implicitly with the trapazoidal (Crank-Nicolson) method. An implicit method removes the very severe stability restrictions that would otherwise be imposed by this term, and computationally only requires a single matrix inversion, because the diffusive terms are linear in the velocity. Finally, the pressure is solved for by using a fractional step method [3]. Mathematically the time discretization can be written as,

$$\frac{\mathbf{v}^* - \mathbf{v}^n}{\Delta t} + \left(\frac{3}{2}(\mathbf{v}^n \cdot \nabla)\mathbf{v}^n - \frac{1}{2}(\mathbf{v}^{n-1} \cdot \nabla)\mathbf{v}^{n-1} \right) = \frac{1}{2Re}\nabla^2(\mathbf{v}^* + \mathbf{v}^n), \tag{1a}$$

$$\frac{\mathbf{v}^{n+1} - \mathbf{v}^*}{\Delta t} = -\nabla p^{n+1} \tag{1b}$$

where $\mathbf{v}^*$ is a temporary intermediate variable. The pressure is found from a Poisson equation obtained by taking the divergence of equation (1b). Ultimately the procedure breaks down into three fundamental parts,

$$\mathbf{r} = -\left(\frac{3}{2}(\mathbf{v}^n \cdot \nabla)\mathbf{v}^n - \frac{1}{2}(\mathbf{v}^{n-1} \cdot \nabla)\mathbf{v}^{n-1} \right) + \frac{1}{Re}\nabla^2(\mathbf{v}^n) \tag{2}$$

$$\left(1 - \frac{\Delta t}{2Re}\nabla^2 \right) \frac{\mathbf{v}^* - \mathbf{v}^n}{\Delta t} = \mathbf{r} \tag{3}$$

$$\Delta t \nabla^2 p = \nabla \cdot \mathbf{v}^* \qquad \text{and} \qquad \mathbf{v}^{n+1} = \mathbf{v}^* - \Delta t \nabla p. \tag{4}$$

The first part (Eqn. 2) involves stencil type, nearest neighbor communication to evaluate the derivatives. All the communication is local and involves only the six neighbors of a 3-D cartesian volume. In contrast, the second part (Eqn. 3) involves a matrix inversion, and will ultimately involve long distance communication. The matrix inversion can be

factored into a series of three tridiagonal inversions without any loss in the order of accuracy. The final part of the solution (Eqn. 4), the Poisson equation, is the most computationally intensive. For a simple domain it can be solved explicitly using a combination of discrete Fourier transforms, cosine transforms and tridiagonal matrix inversion. So, even though this is not a spectral method, an opportunity exists in this stage to evaluate the performance of FFT's on the CM-2.

2.3 Problem Configuration

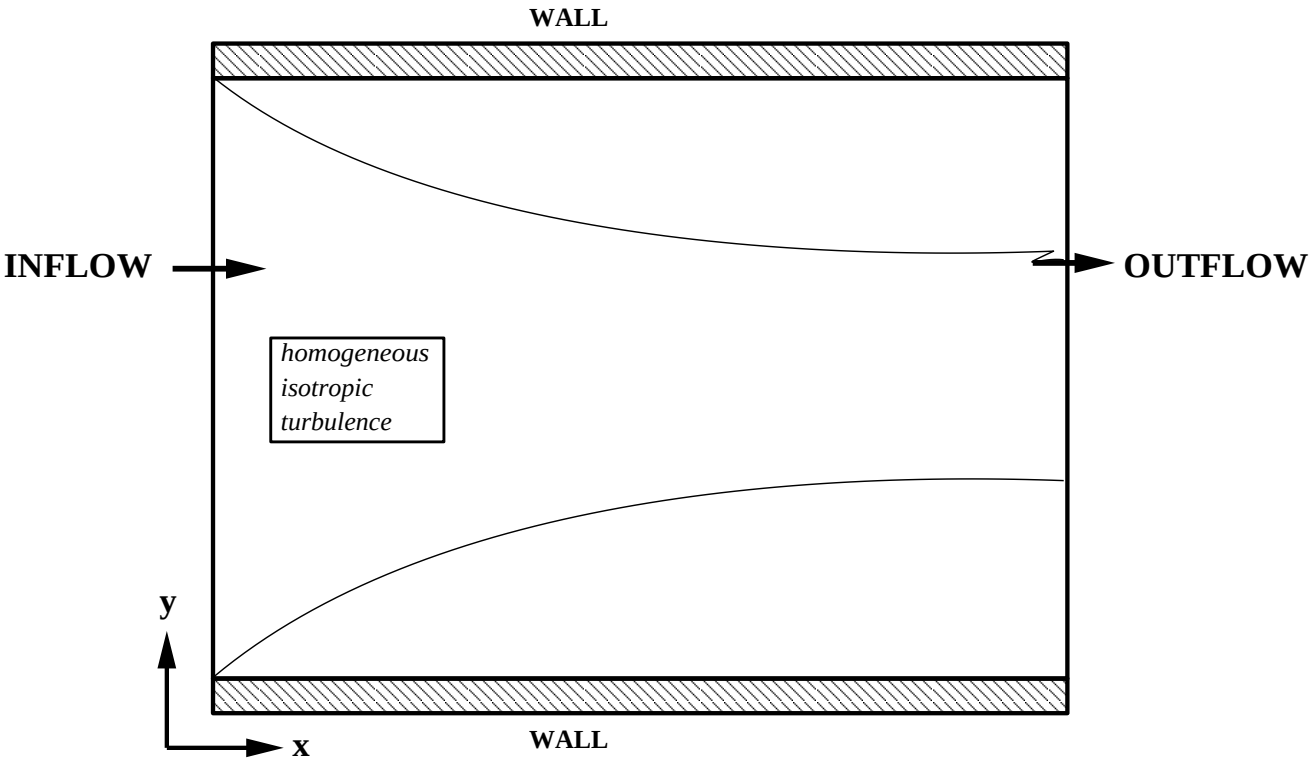

FIGURE 2. Problem configuration for the turbulent boundary layer with large free-stream turbulence.

The problem configuration used to test the computer performance is shown in Figure 2. It is a study of spatially decaying turbulence in the presence of a wall, or conversely, a boundary layer in the presence of high free-stream turbulence. The isotropic, homogeneous turbulence, that enters the domain at the left, interacts with the walls and decays as it is convected downstream. The domain is periodic in the spanwise (z) direction, and at the top and bottom faces of the domain no-slip boundary conditions are imposed. The inflow and outflow boundary conditions are topics unto themselves, and will not be discussed here or included in any of the timing results. The grid is stretched in the wall normal (y) direction in order to resolve the boundary layer. A fairly complicated configuration was chosen in view of the fact that future simulations will require at least this degree of versatility.

To confirm that the method is accurately resolving the turbulence a plot of the decay of turbulent kinetic energy and dissipation is shown in Figure 3. The slopes of the curves are nearly linear and very close to the accepted values of $(-1.2\ to - 1.4)$ for the kinetic energy, and $(-2.2\ to - 2.4)$ for the dissipation. This implies that both the large scales (responsible for the kinetic energy) and the small scales (responsible for the dissipation) are being adequately resolved.

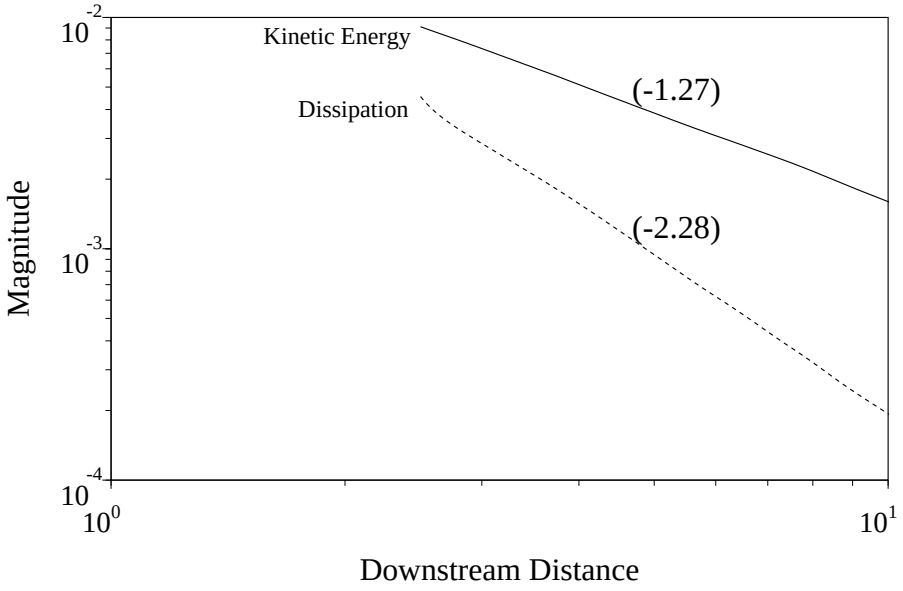

FIGURE 3. The decay of kinetic energy and dissipation as a function of downstream location.

3. GENERAL PERFORMANCE RESULTS

A good measure of performance for turbulence simulations is the normalized CPU-time, or CPU-time per time-step per computational node. In this way, simulations with vastly different numbers of grid points can be reasonably compared. Figure 4 shows a plot of the normalized CPU-time as a function of the the problem size (or number of nodes), for the Cray Y-MP and three different sized CM-2 configurations. For a 32^3 grid ($2\text{x}10^4$ nodes) the single processor Y-MP takes 5 μs per time-step per node, which corresponds to 145 Mflops. As the problem size increases, the performance of the Y-MP increases slightly due to the increased vector lengths. At problem sizes corresponding to $6\text{x}10^5$ nodes it has reached 175 Mflops or almost 4 μs per time-step per node. However, as the problem size increases further there is an abrupt decrease in performance. This is because the Y-MP runs out of core memory and data must now be swapped in and out of memory from disk. The CM-2 does not have this problem, because its core memory is so much cheaper it can provide on the order of 100 times more memory to the user.

The outstanding feature of the CM-2 timings is the dramatic increase in performance with increasing problem size, almost an order of magnitude for the 32k CM-2 when the problem size is increased from 32^3 to 256^3. This is in contrast to the Cray Y-

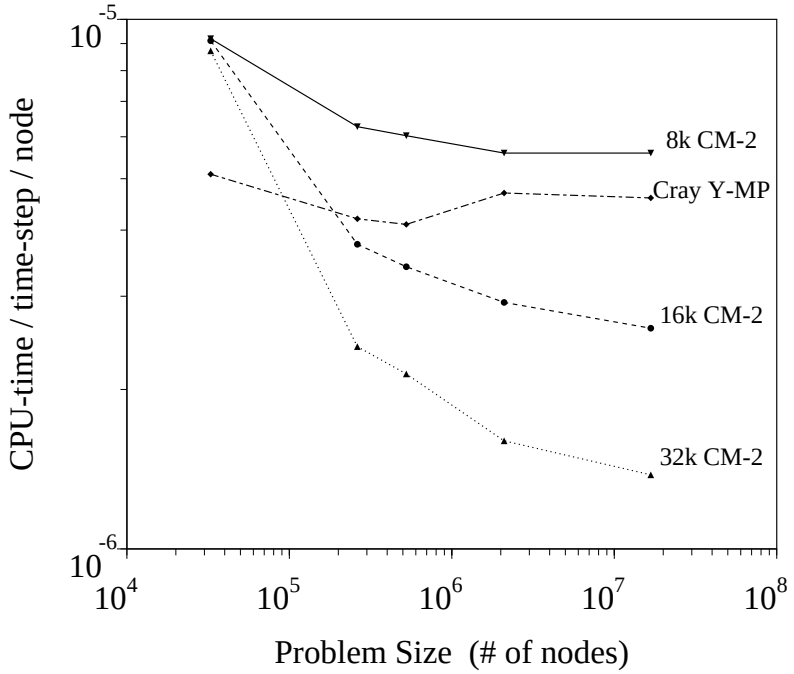

FIGURE 4. CPU-time per time-step per node as a function of the problem size for the Cray Y-MP and sections of the CM-2.

MP which is fairly independent of the grid size. The reason for this behavior is because the CM-2 performance is communication limited. Only with very large problems does the amount of computation begin to amortize the communication overhead. The effect of communication can also be seen in the fact that the Cray obtains, at its worst, 44% of its peak processor speed, but the CM-2 never obtains better than 22% of its peak.

In figure 5 the Mflops ratings of the Y-MP have been used to calculate a Y-MP equivalent Mflops rating for the CM-2. The dependence on problem size is still clear. For larger problems (greater than 128^3 nodes) the 16k CM-2 is roughly equivalent to one processor of the Y-MP. For smaller problem sizes, however, it is no longer clear that the CM-2 has great performance advantages over the Y-MP. This conclusion may be important to other types of turbulence simulation, such as large eddy simulation, which tend to use smaller grid sizes.

Finally, in figure 6 the normalized CPU-time is given as a function of the machine size. The slope (given in parentheses) of each line represents the speedup obtained for various problem sizes. Only the larger grid sizes manage to overwhelm the communication overhead and obtain reasonable speedups. With the smallest grid of 32^3 almost no speedup was obtained by adding more processors. Note that while the 128^3 simulation only takes about ten seconds per time-step, a single simulation takes on the order of a hundred time-steps, and reasonably converged turbulence statistics require on the order of a hundred simulations. So, the 128^3 problem actually requires about 30 CPU-hours of computer time.

It is important to mention some of the factors that may effect the CM-2 timings

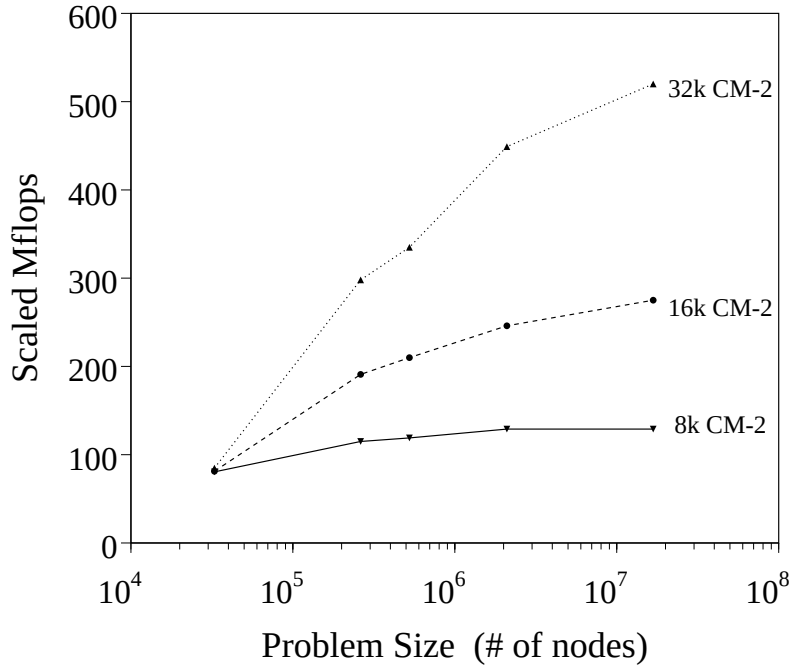

FIGURE 5. Cray Y-MP equivalent Mflops as a function of the problem size for various sections of the CM-2.

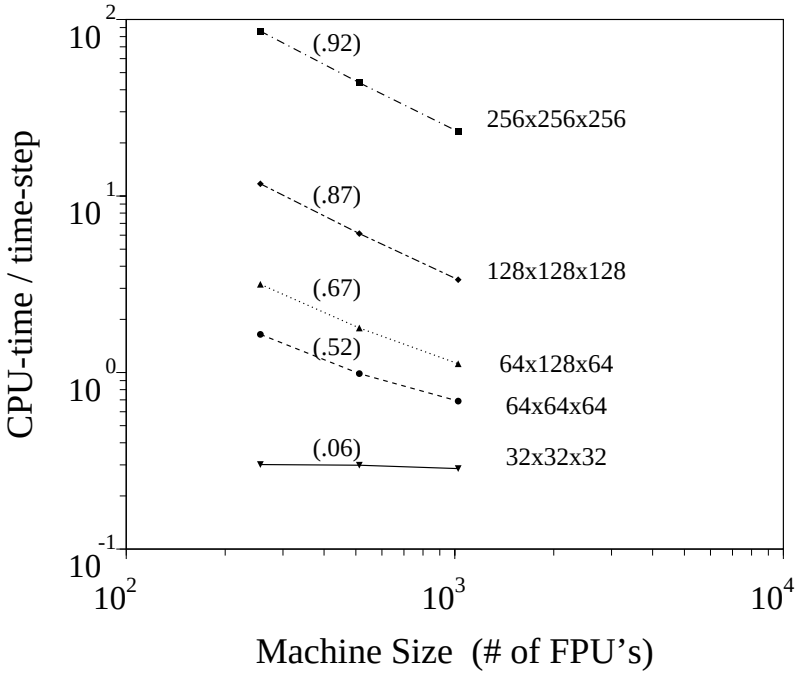

FIGURE 6. The speedup of the CM-2 for various problem sizes.

that are presented above. The number of grid points along each axis are a power of two. Many CM-2 subroutines run more efficiently with power of two axes. 32-bit floating point was used, as opposed to 64-bit on the Cray Y-MP, since 32-bit was deemed to be sufficiently accurate. The CM-2 actually has a 64-bit floating point unit, so 32-bit only saves about 25% on the computation time. Comparing 64-bit and 32-bit simulations is legitimate because the ultimate concern is the time it takes to arrive at a sufficiently accurate solution. Finally, the timings were performed on an unloaded Sun workstation front end. The loading and type of front end were found to effect performance by as much as 50%.

4. DETAILED TIMINGS

The code divides naturally into three parts each of which are fairly typical of DNS, (and computational fluid dynamics, in general). By analyzing each of these parts in detail, it will become clear where bottlenecks lie, and how well individual algorithms map to the CM-2.

4.1 First Stage

The first stage of the algorithm involves explicit stencil type operations to calculate the derivatives of the convective and diffusive terms. The operation is given by,

$$\mathbf{r} = -\left(\frac{3}{2}(\mathbf{v}^n \cdot \nabla)\mathbf{v}^n - \frac{1}{2}(\mathbf{v}^{n-1} \cdot \nabla)\mathbf{v}^{n-1}\right) + \frac{1}{Re}\nabla^2(\mathbf{v}^n). \tag{5}$$

This involves nearest neighbor communication which is very efficiently implemented on the Connection Machine provided NEWS communication grid. For a 128^3 grid only 25% of the total time was spent in this portion of the code (as opposed to 30% for the Y-MP), and a speedup of 96% was obtained (as opposed to 89% for the overall code) when the grid was doubled. This excellent performance is not due to a lack of communication in this portion of the code, fully 50% of the total time is spent in communication (within the CSHIFT function). It is due to the fact the CM-2 is well suited to cartesian nearest neighbor communication.

In addition, this portion of the code is where the majority of boundary conditions are implemented. Boundary conditions on a SIMD machine require that all interior processors lie idle while the boundary nodes do something special. This can impact the overall performance significantly. For this stage of the algorithm it was found that the four domain faces that require boundary conditions took on the order of 30% of the time. For this reason, algorithms that can naturally incorporate boundary conditions into the solution procedure are very desirable on the CM-2.

4.2 Second Stage

The second stage of the algorithm requires the solution of a Helmholtz equation,

$$\left(1 - \frac{\Delta t}{2Re}\nabla^2\right)\frac{\mathbf{v}^* - \mathbf{v}^n}{\Delta t} = \mathbf{r}. \tag{6}$$

There are a number of techniques for the solution of this matrix equation. On the Cray, the matrix is factored into a set of three tridiagonal matrices and then Guass elimination is used to solve the tridiagonals. On the CM-2 the same procedure can be used but a parallel tridiagonal algorithm such as cyclic reduction [4] must be substituted for serial Guass elimination. Or, on the CM-2, the matrix can be left unfactored and simply solved using an iterative technique such as conjugate gradients [5]. In either case, this portion of the code takes about 25% of the overall time, compared to about 30% for the Y-MP.

Cyclic reduction involves long distance communication. For the 128^3 grid, 85% of the cyclic reduction algorithm was spent in communication and a speedup of only 88% was obtained. In contrast, the conjugate gradient solution technique involves only nearest neighbor communication, and had a very good speedup. However, the conjugate gradient (CG) algorithm involves iteration. It was found that because the matrix is highly diagonally dominant, only 2-4 iterations are required, and the conjugate gradient algorithm is competitive with the more complicated matrix factorization and cyclic reduction technique. This is a useful fact for those cases, such as unstructured meshes, where the matrix can not be factored.

4.3 Third Stage

The final stage of the algorithm involves the solution of a Poisson equation for the pressure,

$$\Delta t \nabla^2 p = \nabla \cdot \mathbf{v}^*. \tag{7}$$

This is the most computationally intensive portion of the algorithm. For the simple geometry of this flow, this equation can be solved explicitly through the use of a Fourier transform in the z-direction, a cosine transform in the x-direction, and a tridiagonal matrix inversion in the y-direction [6]. Both the Fourier transform and cosine transform involve calls to the Connection Machine FFT subroutines.

FFT's involve regular (butterfly), but long distance communication. The FFT's have linear speedup and scale linearly in the number of data points but are very slow. Fully 50% of this portion of the code (and 25% of the total time) is spent in four FFT calls. A simple diagonally preconditioned conjugate gradient solution of the Poisson equation was found to be only three times to ten times slower than the transform method. This is because 15 iterations of the CG method can be accomplished in about the time it takes to do a single FFT. With better preconditioning, the CG method would be very competitive with the transform method.

5. ADDITIONAL ISSUES

In an evaluation of the CM-2 as a tool for turbulence research there are other issues besides performance. For instance, the issue of memory has already been mentioned. Being able to perform simulations entirely in core memory not only saves time swapping out to disk, but considerably simplifies the programming burden.

In terms of programming, the CM-2 is very similar to the Y-MP. The SIMD architecture allows a serial programing style, and similar programming environment. This

is a great advantage of SIMD machines that should not be overlooked by the application programmer when evaluating the utility of parallel architectures. Both machines compile Fortran 90, and the CM-2 extensions to Fortran 90 are a very useful addition. Optimization on the Y-MP is much easier than on the CM-2, because the Connection Machine fortran compiler is still fairly unpredictable. But the debugging environment of the CM-2 tends to make up for this deficiency. And as we have seen, the lack of versatility of the CM-2 is made up for by the fact that it performs many brute force algorithms very efficiently.

Another useful measure besides performance is performance per dollar, or Mflops per Mdollar. A Cray Y-MP, such as the one used in this study costs approximately 25 million dollars (3.125 million per processor), and averages about 160 Mflops, or about 50 Mflops/Mdollar. The CM-2 costs about 5 million dollars. If a performance of 300 to 500 Cray equivalent Mflops is assumed (remembering, of course, that CM-2 performance is very problem dependent), then the CM-2 gets from 60 Mflops/Mdollar to 100 Mflops/Mdollar. These numbers are very approximate, and probably only accurate to within a factor of two. Nonetheless, they do indicate that the Connection Machine is already very competitive with vector architectures, not only in performance but in terms of cost, as well.

6. CONCLUSIONS

The results of this study indicate that the Connection Machine is a viable tool for the direct numerical simulation of turbulence. Its performance and performance per dollar can be very comparable to similar generation vector supercomputers. However, unlike vector supercomputers the performance of the CM-2 is a strong function of the problem size. Traditional performance measures need to be altered to account for these types of communication effects which are common to all massively parallel computers.

The architecture of the CM-2 limits its versatility. In short, it is a brute force machine, overwhelming problems with quantity rather than quality. As a result it tends to perform brute force algorithms, such as conjugate gradients or finite volume discretizations, just as efficiently as more cleverly constructed algorithms. This fact, is not entirely negative. It means that in those cases were tricks such as matrix factoring and variable transformation can not be applied (ie. general geometries), the CM-2 will not experience any loss in performance. Therefore, despite its lack of versatility the CM-2 may actually be more useful for complicated problems, because these problem are only amenable to brute force methods.

Direct numerical simulation of turbulence has always been closely tied to the performance of state of the art supercomputers. Many assumptions about turbulence simulation have been based on the vector nature of supercomputers. Many of these assumptions, such as the superiority of spectral methods, will be challenged as the availability of massively parallel computing begins to open computational horizons. It appears that the CM-2 and its successors will be key partners in these new explorations of turbulence simulation.

Acknowledgements

This work was funded by a grant from the National Science Foundation. All computer time on the CM-2 and Cray Y-MP was provided courtesy of the NAS division of NASA-Ames Research Center.

References

1. Committee on Physical, Mathematical, and Engineering Sciences, Grand Challenges: High Performance Computing and Communications, OSTP FCCSET report. National Science Foundation.

2. F. H. Harlow and J. E. Welsh, *Phys. Fluids* **8**, (1965) 2182-2189

3. J. B. Perot, An analysis of the fractional step method. Submitted to the *Jour. Comp. Physics*, (1992).

4. R. W. Hockney and C. R. Jesshope, Parallel Computers 2 Bristol and Philadelphia: Adam Hilger, (1988). 475-489.

5. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, Numerical Recipes, Cambridge University Press, (1986). 70-73.

6. B. L. Buzbee, G. H. Golub, and C. M. Nielson, On Direct Methods for Solving Poisson's Equation, *SIAM J. Numer. Anal.* **7-4**, (1970) 627-656.