AIAA JOURNAL
Vol. 34, No. 4, April 1996

Navier–Stokes Simulations of Jet Flows on a Network of Workstations

M. Ehtesham Hayder*
NASA Lewis Research Center, Cleveland, Ohio 44135
and
D. N. Jayasimha†
Ohio State University, Columbus, Ohio 43210-1107

We study a Navier–Stokes application code on a network of workstations to investigate the computational, communication, and scalability characteristics. This code uses a high-order finite difference scheme to solve for the time-accurate flowfield of a jet using the compressible Navier–Stokes equations. The network of workstations that is used in this study is the Lewis Advanced Cluster Environment (LACE). LACE has 32 nodes connected through various networks—Ethernet, IBM's ALLNODE switch, FDDI, ATM, etc. Each node of LACE has an RS6000 processor as its CPU. On the LACE test bed, we study the communication characteristics of different networks and the overheads induced by the PVM message-passing library used for parallelizing the application. We demonstrate that clustering of workstations is effective and has the potential to be computationally competitive with supercomputers at a fraction of the cost. To illustrate this last point, we compare the performance of LACE with the Cray Y-MP.

I. Introduction

THE advent of massively parallel processors gives the opportunity for scientists to parallelize their computationally intensive codes and reduce turnaround time. Recognizing this, a number of researchers[1–4] have studied many CFD applications on variety of parallel computers. In recent years, scientific workstations have become considerably faster, with sustained performance in the range of 30–80 Mflops. It is now possible to create a cost-effective parallel computing environment by clustering workstations. In this study we examine one such workstation cluster. This network of workstations is located at NASA Lewis Research Center and is known as the Lewis Advanced Cluster Environment (LACE).[5] The experimental test bed, composed of RS6000 processors, is connected through a number of different networks such as Ethernet, FDDI, ATM, IBM's ALLNODE switches, etc. Cluster of workstations like the LACE are becoming increasingly popular because they show promise as a low-cost alternative to expensive supercomputers and massively parallel processors.

Our goal is to study specific CFD application code on this network of workstations to understand the computational, communication, and scalability characteristics. Since the computation and communication requirements of this application are typical of many numerical applications and since the LACE test bed is representative of emerging workstation clusters, we expect that some general results can be drawn from this study. The code solves the full Navier–Stokes equations to compute the flowfield of an axisymmetric jet. Numerical simulations with this model are done to predict the jet noise associated with the flowfield of an axisymmetric jet. This problem is motivated by the need to suppress the jet exhaust noise of aircraft. The success of the High Speed Civil Transport (HSCT) plane partially depends on the reduction of radiated noise level. One can compute the sound emanating from the jet by solving the full (time-dependent) compressible Navier–Stokes equations. The computation can, however, be very expensive and time-consuming. The difficulty can be partially overcome by limiting the solution domain to the near field, where the jet is nonlinear, and then using acoustic analogy to relate the far-field noise to the near-field sources. The technique requires obtaining the time-dependent flowfield. Our numerical model in this study is designed to calculate such flowfields near the nozzle exit.

In this study, we have followed a pragmatic approach by parallelizing only those segments of the algorithm that are computationally intensive. Such an approach can reduce the time for parallelizing sequential codes that are already in use. Using a profiler, we found four computationally intensive routines that could be fully parallelized. We also found that only approximately 0.025% of the total operation count arises from the serial portion of the code.

Our distributed-memory implementation on LACE uses PVM,[6] a portable message-passing library developed at the Oak Ridge National Laboratory. Some of our earlier results are discussed in Hayder et al.[7] A discussion of the parallelization of this application for different architectural platforms is found in Jayasimha et al.[8]

In the next section we briefly discuss the governing equations and the numerical model of the physical problem. Section III has a discussion of the application characteristics, a more detailed description of LACE, and the experimental methodology. Results are presented in Sec. IV, and the conclusions from this study in Sec. V.

II. Problem

A. Governing Equations

We solve the Navier–Stokes (NS) equations to compute flowfields of an axisymmetric jet. The NS equations for such flows in polar coordinates can be written as

$$LQ = S$$

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial r} = S$$

where

$$Q = r \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho e \end{pmatrix}$$

Received June 26, 1995; revision received Nov. 2, 1995; accepted for publication Nov. 18, 1995.

*Senior Research Associate, Institute for Computational Mechanics in Propulsion; currently Staff Scientist, ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, VA 23681.

†Assistant Professor, Department of Computer and Information Science.

$$F = r\begin{pmatrix} \rho u \\ \rho u^2 - \tau_{xx} + p \\ \rho uv - \tau_{xr} \\ \rho uH - u\tau_{xx} - v\tau_{xr} - \kappa T_x \end{pmatrix}$$

$$G = r\begin{pmatrix} \rho v \\ \rho uv - \tau_{xr} \\ \rho v^2 - \tau_{rr} + p \\ \rho vH - u\tau_{xr} - v\tau_{rr} - \kappa T_r \end{pmatrix}$$

$$S = \begin{pmatrix} 0 \\ 0 \\ p - \tau_{\theta\theta} \\ 0 \end{pmatrix}$$

F and G are the fluxes in the x and r directions, respectively, S is the source term that arises in the cylindrical polar coordinates, and τ_{ij} are the shear stresses. One obtains the Euler equations from the above equations by setting κ and all τ_{ij} equal to zero.

B. Numerical Model

The numerical model uses a high-order MacCormack scheme due to Gottlieb and Turkel[9] to solve the NS equations. For the present computations, the operator L in the equation $LQ = S$ or equivalently $Q_t + F_x + G_r = S$ is split into two one-dimensional operators, and the scheme is applied to these split operators. For the one-dimensional split equation $Q_t + F_x = S$, we express the predictor step with forward differences as

$$\bar{Q}_i = Q_i^n - \frac{\Delta t}{6\Delta x}\left[7\left(F_{i+1}^n - F_i^n\right) - \left(F_{i+2}^n - F_{i+1}^n\right)\right] + \Delta t\, S_i$$

and the corrector step with backward differences as

$$Q_i^{n+1} = \frac{1}{2}\left(\bar{Q}_i + Q_i^n - \frac{\Delta t}{6\Delta x}[7(\bar{F}_i - \bar{F}_{i-1}) - (\bar{F}_{i-1} - \bar{F}_{i-2})] + \Delta t S_i\right)$$

This scheme becomes fourth-order accurate in the spatial derivatives when alternated with symmetric variants.[9] We define L_1 as a one-dimensional operator with a forward difference in the predictor and a backward difference in the corrector. Its symmetric variant L_2 uses a backward difference in the predictor and a forward difference in the corrector. For our computations, the one-dimensional sweeps are arranged as

$$Q^{n+1} = L_{1x}L_{1r}Q^n$$

$$Q^{n+2} = L_{2r}L_{2x}Q^{n+1}$$

This scheme is used for the interior points. To advance the scheme near boundaries, the fluxes are extrapolated outside the domain to artificial points, using a cubic extrapolation to compute the solution on the boundary. At the outflow boundary, we then solve the following set of equations to get the solution at the new time for all boundary points:

$$p_t - \rho c u_t = 0, \qquad p_t + \rho c u_t = R_2$$

$$p_t - c^2\rho_t = R_3, \qquad v_t = R_4$$

where R_i is determined by which variables are specified and which are not. Whenever the combination is not specified, R_i is just those spatial derivatives that come from the NS equations. Thus, R_i contains viscous contributions even though the basic format is based on inviscid characteristic theory. This framework of outflow boundary condition implementation is discussed by Hayder and Turkel.[10] Further discussions of our numerical model and its use for the noise computations are given in Hayder et al.[11] and Mankbadi et al.[12]

In this study, we consider a supersonic axisymmetric jet. Details of typical computations can be found in Refs. 11 and 12. Here we consider a jet with the mean inflow profile

$$\bar{u}_r = u_\infty + (u_c - u_\infty)g_r$$

$$\bar{T}_r = T_c + (T_\infty - T_c)g_r + [(\gamma - 1)/2]M_c^2(1 - g_r)g_r$$

$$g_r = \frac{1}{2}\left\{1 + \tanh\left[\frac{(1/r) - r}{4\theta}\right]\right\}$$

where θ is the momentum thickness. The subscripts c and ∞ refer to the centerline and freestream values, respectively. At inflow, we assume the radial velocity is zero and the static pressure is constant. The size of our computational domain is 50 radii in the axial direction and 5 radii in the radial direction. We excite the inflow profile at location r and time t as

$$u(r,t) = \bar{u}(r) + \epsilon\,\mathrm{Re}[\hat{u}\exp(i\pi Srt)]$$

$$p(r,t) = \bar{p}(r) + \epsilon\,\mathrm{Re}[\hat{p}\exp(i\pi Srt)]$$

$$\rho(r,t) = \bar{\rho}(r) + \epsilon\,\mathrm{Re}[\hat{\rho}\exp(i\pi Srt)]$$

$$v(r,t) = \epsilon\,\mathrm{Re}[\hat{v}\exp(i\pi Srt)]$$

Here $\hat{u}$, $\hat{v}$, $\hat{\rho}$, and $\hat{p}$ are the eigenfunctions of the linearized equations with the same mean flow profile, ϵ is the excitation level, and Sr is the Strouhal number. We consider a case with $u_\infty/u_c = \frac{1}{4}$, $T_\infty/T_c = \frac{1}{2}$, momentum thickness $\theta = \frac{1}{8}$, and Strouhal number $Sr = \frac{1}{8}$. Unless otherwise mentioned, the nozzle radius is used as the length scale. The jet-center Mach number is 1.5, and the Reynolds number based on the jet diameter is about 0.13 million. This is an ideally expanded hot jet, and the physical parameters are similar to those used by Hayder and Turkel[13] to study the boundary conditions of jet flow computations. The fluctuating sound source in the near field[12] can be calculated from the flowfield given by the numerical simulations in this study. One can then use these results to obtain the far-field sound level by either using the acoustic analogy or patching the near field to linearized Euler equations describing the far field.

III. Parallel Implementations

A. Application Characteristics

The factors that affect parallel performance are:

1) *Single-processor performance.* We have examined the single-processor performance in Refs. 7 and 8. We give a summary of those efforts in Sec. III.C.

2) *Communication cost.* This cost depends on both the number of communication startups and the volume of data communicated. Usually, the startup cost is 2–3 orders of magnitude higher than the per-word transfer cost. One method to reduce the effect of startup cost is to group data to be communicated into long vectors.

3) *Overlapped communication and computation.* It is desirable that communication be overlapped with computation as far as possible. Increasing the amount of overlapping, however, could lead to finer granularity of communication, which then leads to a larger number of startups.

4) *Bursty communication.* Communication may not be evenly spaced throughout the computation. Instead, it may occur in spurts at certain intervals. Such communication could overwhelm the network's throughput capacity temporarily, leading to increased communication cost and process waiting time. Some amount of burstiness is inevitable, since parallel programs are usually written in the SPMD (single-program multiple-data) style. There is also usually an inverse relationship between bursty data and the number of communication startups.

It is clear from the factors cited above that there is a subtle relationship among communication startup cost, overlapping communication with computation, and bursty communication. After some experimentation, we choose to decompose the domain by blocks along the axial direction only.

For the solution of the NS equations, hereafter referred to as NS, each internal subdomain exchanges its two flux values, velocity, and temperature along the boundary with its appropriate (left or

Table 1 Application characteristics

Application	Total computation, Mflops	Communication		
		Startups per processor	Volume per processor	
			Mbyte	Mbit
NS	145,000	80,000	120	960
Euler	77,000	20,000	64	512

Table 2 Computation-communication ratios

Number of processors	Flops/bytes		Flops/startup	
	NS	Euler	NS	Euler
2	604	601	906 K	1925 K
4	302	301	453 K	963 K
8	151	150	227 K	481 K
16	76	75	113 K	241 K

right) neighbor. To reduce the number of communication startups, we group communication—first, all the velocity and temperature values along a boundary are calculated and then packaged into a single send. We use a similar scheme for the flux values that need to be communicated.

With this decomposition, the computational and communication requirements of the application for 5000 time steps on a 250 × 100 grid are shown in Tables 1 and 2 (communication for time-step computations and initial broadcast of data is small and hence not included). The solution of the Euler equations, hereafter referred to as Euler, is also parallelized in a similar manner to NS. As Table 1 shows, it has roughly 50% of the computational requirements, 50% of the communication volume, and 25% of the communication startups of NS. Note that the communication requirements are shown on a per-processor basis. Table 2 shows the average number of floating-point operations performed by a processor on the average before it needs to communicate a byte of data or before a communication message is initiated.

To give some idea of the effects of communication, consider NS to be executed on a network of 10 workstations connected via Ethernet. Assume a reasonable throughput of 20 Mflops per processor and the maximum throughput of 10 Mbit/s for Ethernet. From Table 1, the computation time will then be approximately 725 s [145,000/(10 × 20)], whereas a lower bound on the communication time, ignoring the effect of startups and assuming the maximum Ethernet throughput of 10 Mbit/s, is 960 s (960 × 10/10).

B. LACE Architecture

The LACE test bed is constantly being upgraded. The present configuration, shown in Fig. 1, has 32 RS6000 processor nodes (nodes 1–32) and a node (node 0) that is the file server. These nodes or subsets of them are connected through various networks with different speed and connection characteristics. All the nodes are connected through two Ethernet network (10 Mbit/s); one of them is for general use, and the other is dedicated to parallel use. Nodes 9–24 are interconnected through a FDDI interface with a peak bandwidth of 100 Mbit/s. It is convenient, for our purposes, to consider the nodes to be partitioned into a lower half (nodes 1–16) and an upper half (nodes 17–32). The lower half has RS6000 Model 590 CPUs capable of 125 Mflops peak, with the following networks interconnecting the nodes: an ATM network capable of a peak bandwidth of 155 Mbit/s, and IBM's ALLNODE switch capable of a peak throughout of 64 Mbit/s per link. The upper half has the slower RS6000 Model 560 CPUs, capable of 100 Mflops peak, and is connected through IBM's ALLNODE prototype switch capable of a peak throughput of 32 Mbit/s per link. The ALLNODE switch is a variant of the Omega interconnection network and is capable of providing multiple contentionless paths between the nodes of the cluster (a maximum of eight paths can be configured between source and destination processors). The present setup does not permit the use of more than 16 processors using the faster networks. The nodes have varying main memory capacity (64, 128, 256, and 512 Mbyte). We have used the popular parallel virtual machine (PVM) message-passing library (version 3.2.2)[6] to implement our parallel programs. We

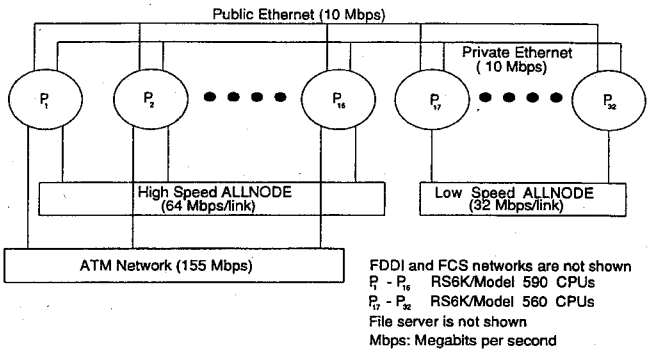

Fig. 1 Lewis advanced cluster environment.

will refer to the LACE cluster with RS6000 Model 560 processors as LACE/560, and to that with RS6000 Model 590 processors as LACE/590.

We also used the Cray Y-MP/8, which has eight vector processors, for this study. A single processor of the Cray Y-MP has a peak rating of approximately 333 Mflops.

C. Experimental Methodology

The performance indicator is the total execution time. Single-processor experiments were done on an IBM RS6000 Model 560 workstation. All experiments were conducted in single-user mode. In almost all the experiments, NS and Euler show similar trends; hence, unless otherwise mentioned, quantitative comments refer to NS. We found initially that most parts of our application were limited by the poor performance of the memory hierarchy involving the cache and the main memory. Improved cache performance was the key, and this was achieved by accessing arrays in stride 1 fashion wherever possible (achieved through loop interchanging). [Stride is the distance (in words) that separates two logically adjacent elements of a data structure (a matrix, for example), stored in memory. Fortran stores matrix elements in column-major order. Consider the storage of an $n \times n$ matrix A in Fortran. Adjacent (row) elements $A(i, j)$ and $A(i, j + 1)$ have a stride of n, also referred to as stride n, while adjacent (column) elements $A(i, j)$ and $A(i + 1, j)$ have a stride of 1.] To utilize the cache memory efficiently, it is important that we access elements with stride 1. We experimented with a number of other modifications, the following of which yielded some improvement: better register usage by collapsing multiple COMMON blocks into a single one, strength reduction (replacing exponentiations by multiplications wherever feasible, for example), and replacement of division operations by multiplications wherever feasible, since the former are relatively expensive. All these optimizations yielded an overall improvement of roughly 80% (from 9.3 to 16.0 Mflops on an RS6K/560). We have used this optimized version of the code for parallelization.

In all the experiments we have separated the execution time into two additive components: processor busy time, and the communication time that is not overlapped with computation. The processor busy time is itself composed of the actual computation time and the software overheads associated with sending and receiving messages. An accurate separation of these components is not possible, however, unless we have hardware performance monitoring tools. The nonoverlapped communication time could also include the idle time of a processor waiting for a message. The next section presents a detailed discussion of the results of our experiments on LACE.

IV. Results

Figure 2 shows the pressure contours from the solution of the NS equations. A 250 × 100 grid was used in this computation. These results were obtained after about 17,000 time steps and are presented here to give an idea of the physical problem. To keep execution times small, all other simulations are for 5000 time steps. Even with 5000 time steps, computational requirements are not insignificant, as shown in Table 1. All our timing results were obtained in single-user mode, and hence there were only minor discrepancies in the execution times. To illustrate this point we show timings for NS on

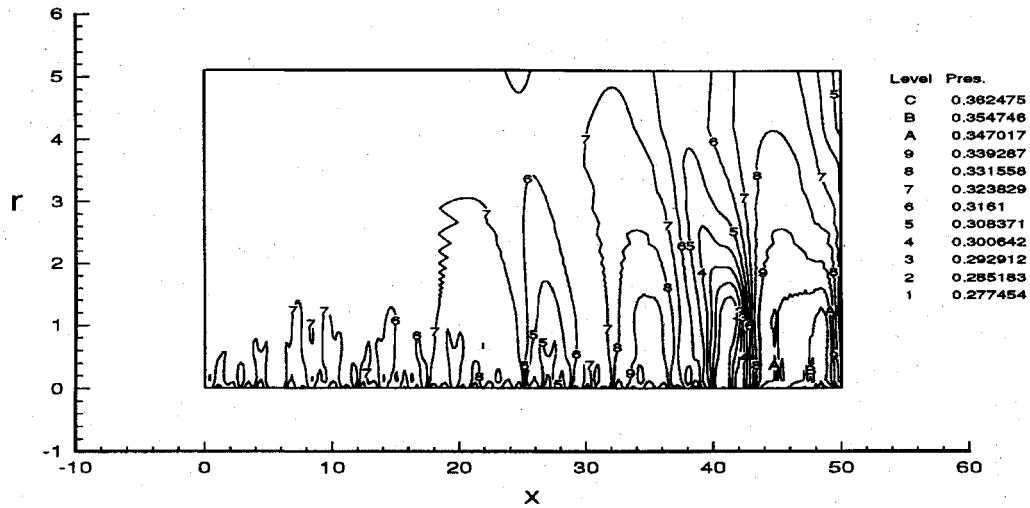

Fig. 2 Snapshot of pressure in an excited jet.

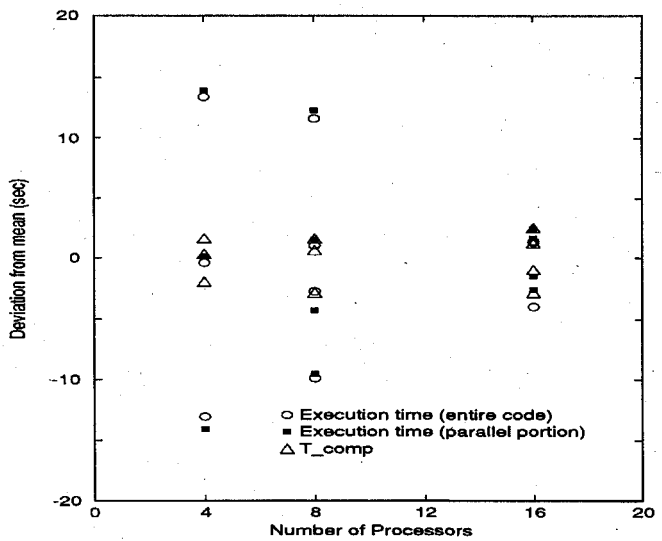

Fig. 3 Uncertainties in time measurements.

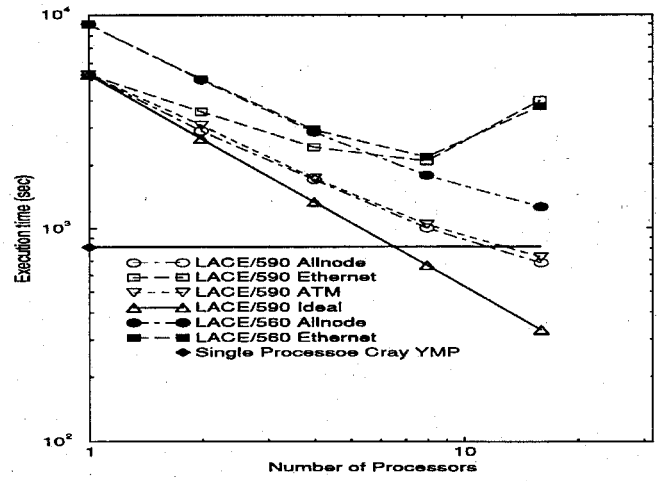

Fig. 4a NS execution time.

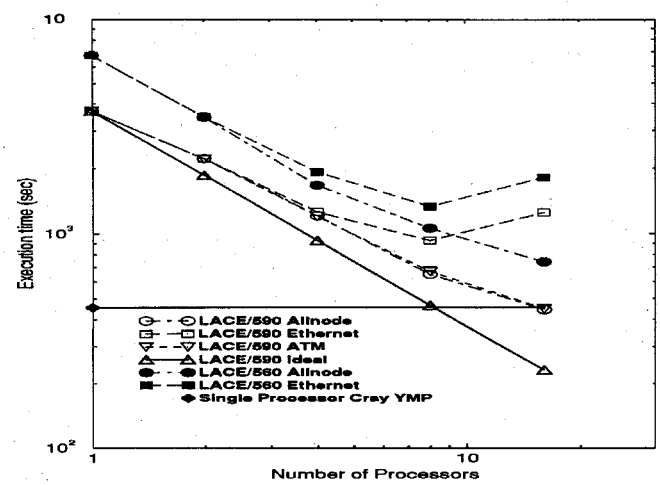

Fig. 4b Euler execution time.

LACE/590 with ALLNODE switch for 4, 8, and 16 processors, in Fig. 3. Computations were repeated thrice for 8 and 16 processors and twice for 4 processors. We observe that the deviation from the mean is about 1% or less.

Figures 4a and 4b show the performance of NS and Euler, respectively, on different networks of LACE—the lower- and upper-half Ethernets, the lower- and upper-half ALLNODE switches, and the lower-half ATM network. A log–log graph has been chosen to facilitate presentation. With this plot, the ideal execution time falls linearly with increasing number of processors. A graph of the ideal execution time, with the single processor 590 as the base, is shown for comparison. The figures also show the execution time on a single-processor Cray Y-MP. For ease of visual comparison, we draw a horizontal line through the execution time on a single processor Y-MP.

In an earlier study,[7] we have found that the performance of the FDDI network is almost identical with the upper-half ALLNODE switch, and hence the performance of the former is not shown in the figures to reduce clutter. We found that the superior speed of FDDI is balanced by the availability of slower but multiple paths in the ALLNODE switch.

Let us consider the performance of the Ethernets first. The execution time falls for both the lower- and upper-half Ethernets with increasing number of processors, but the performance degrades drastically beyond 8 processors. Below 8 processors, the Ethernet connecting LACE/590 performs better than LACE/560 because of superior CPU performance of the former. Even with 2 processors, one can see the deviation from the ideal speedup because of the network. Network congestion progressively gets worse with the increase in number of processors, and beyond 8 processors the execution time actually increases. The inability of Ethernet to handle traffic beyond eight processors of NS is shown by the following argument: Let each processor have a reasonable throughput of 20 Mflops; consider a 1-s interval. Table 2 shows that during this interval, each processor produces 1.06 Mbit for communication, on the average. This translates to approximately 8.5 Mbit

for 8 processors. Ethernet is capable of supporting 10 Mbit peak and normally supports only a fraction of this bandwidth. Thus, it is not surprising that Ethernet's performance gets steadily worse beyond 8 processors. With 16 processors, Ethernet is not capable of supporting the communication demands of the application. As a result, the execution time increases with increasing number of processors—clearly anomalous behavior. It is also interesting to see the convergence of the performance of the lower- and upper-half Ethernets at and beyond 8 processors of NS. The superior speed of the CPU no longer matters, since the application becomes communication-bound. This is not the case, however, for Euler, where, even with performance degradation, the lower-half Ethernet outperforms the upper-half Ethernet because of superior CPU performance. We expect this trend in Ethernet behavior for other applications also, though the breakpoint when the performance degrades could change.

The execution time falls almost linearly for the ALLNODE switches and the ATM network, with some sublinearity effects beginning to show at 16 processors—the deviation from the ideal case increases as the number of processors increases, however. One sees a sustained performance difference between the faster LACE/590 and LACE/560. This improvement can be attributed to faster CPUs and a faster network. The applications are still not communication-bound. It is interesting to note that the ATM network and the lower-half ALLNODE switch are very similar in performance. The peak rate of 155 Mbit/s provided by ATM is equaled by the multiple paths provided by ALLNODE switch at 64 Mbit/s per link.

It is also seen that 12–16 processors of LACE/590 with the faster network give the same performance as a single processor of the Y-MP. We will comment more on this aspect in the Conclusion section.

Figures 5a and 5b aid in a more in-depth analysis of the performance of LACE. The execution time is separated into two additive components as explained in the previous section. In these figures we plot the ratio of the nonoverlapped communication time (T_{noc}) to the processor busy time (T_{comp}) against the number of processors. This yields some insight into the behavior of the networks and the communication demands of the application. With the Ethernets, the nonoverlapped communication time increases superlinearly with the number of processors. Relative communication times for the ALLNODE switches and the ATM network increase almost linearly with the number of processors. For NS with ALLNODE, the nonoverlapped communication time is about 10% of the processor busy times with 2 processors. With 16 processors, it increases to about 80% on the LACE/590 and 90% on the LACE/560. For 16 processors with Ethernet the increase is about 800% on the LACE/590 and 550% on the LACE/560. As shown in Fig. 5b, relative communications for Euler are somewhat less. Breit et al.[14] used the ratio of communication and computation times in their study of architectural balance for the BBN TC2000. In a balanced configuration communication times should be in the same order as computation times. This is difficult to achieve with Ethernet on a large number of processors.

An important measure of scalability is the speedup. The speedup with p processors is the ratio of the execution time of the application with a single processor to the execution time with p processors. Figure 6 shows the speedups for NS with different networks. All networks show sublinear speedups with increase in the number of processors. This is not surprising, since the application has formidable communication requirements (with a large percentage of it nonoverlapped with computation), as shown in Tables 1 and 2. On the LACE/590 with the ALLNODE switches and with the ATM network, we obtained speedups of about 8 for 16 processors. For Ethernet they were about 1.33. The maximum speedup for Ethernet was about 2.5 for 8 processors. On the LACE/560, speedups with Ethernet were slightly higher, though the actual performance of LACE/560 was worse than LACE/590.

So far, results have been presented with a problem of size 250 × 100. As the number of processors increases, each processor gets a smaller portion of the grid. Overheads with loop setups and other sequential computations (initialization, for example) become relatively important in those cases, which translates to poor

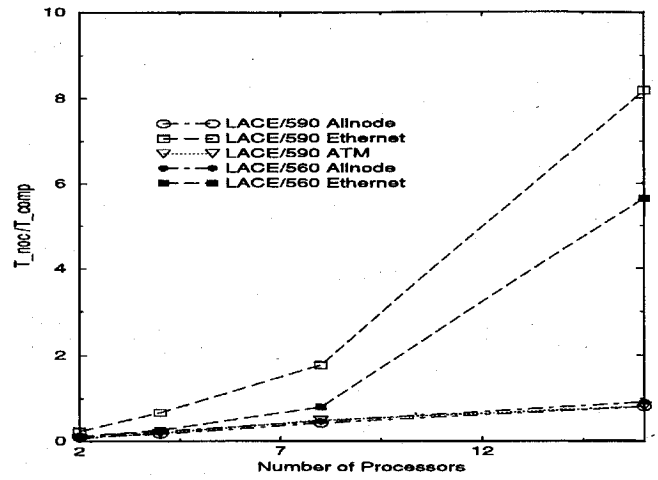

Fig. 5a NS computations on the LACE.

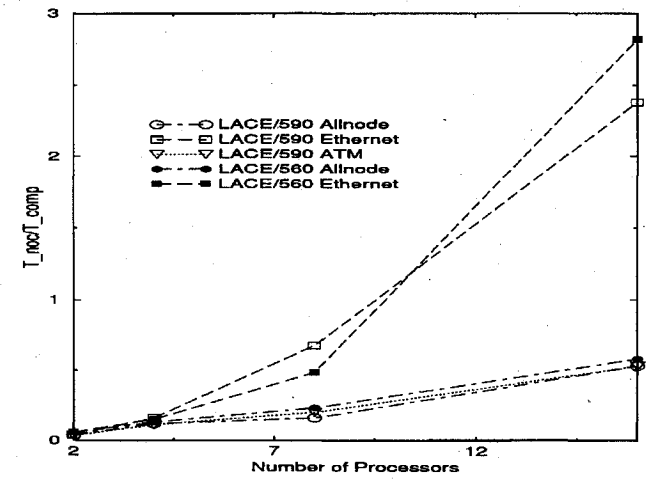

Fig. 5b Euler computations on the LACE.

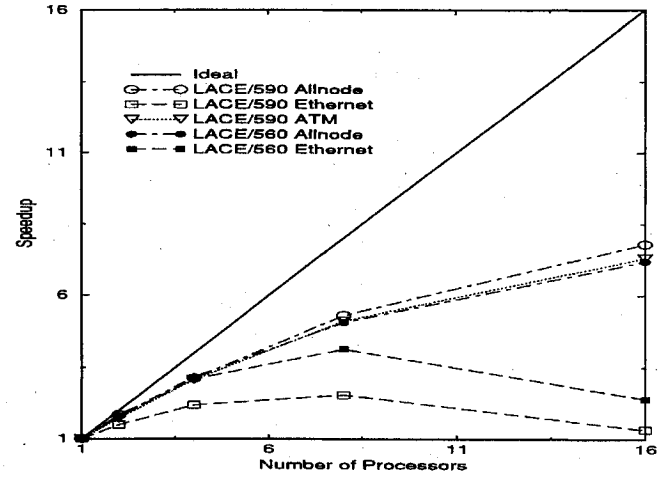

Fig. 6 Speedups for NS computations with different networks.

speedups. For the 250 × 100 grid the speedup on the 16-processor LACE/590 with the ALLNODE switch is about 8. What is the effect on the speedup for larger problem sizes? We see from Fig. 7 that as the problem size is increased, there is an increase in the speedup, and the sublinearity effects at 16 processors become less pronounced. For 500 × 100 and 1000 × 100 grids on 16-processor LACE/590, the speedups are about 10 and 12, respectively. This results from the fact that sequential overheads remain nearly constant with the increase in the problem size—consequently, the fraction of "parallel time" as a fraction of the total time increases, resulting in a higher speedup for the larger grid. Another reason for this improvement is the enhanced cache performance with large vectors.

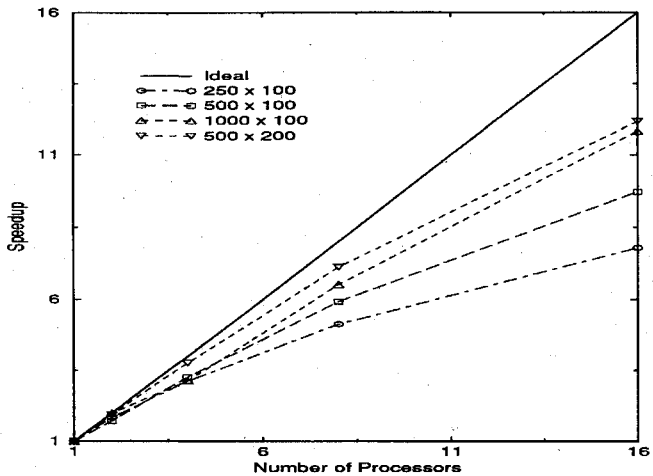

Fig. 7 Speedups for NS computations for different grid sizes.

An interesting property is exhibited in Fig. 7: the 500 × 200 grid exhibits better speedup than the 1000 × 100 grid, though the computational requirements of the two grids are the same. Note that with the 500 × 200 grid, the columns are twice as long as with the other grid. The elements in the column are accessed in nonunit stride for obtaining the solution vectors. Consequently, the sequential execution for the 500 × 200 grid is worse than for the 1000 × 100 grid by about 30%. With increasing number of processors, the amount of nonunit stride computations per processor decreases. Hence the 500 × 200 grid begins to exhibit better speedup behavior than the 1000 × 100 grid, which did not have that many nonunit stride accesses to begin with.

V. Conclusion

Our study with a particular CFD application shows that a workstation cluster with a fast network and with about 12–16 processors performs as well as a single vector processor of the Cray Y-MP. The application has formidable computation and communication characteristics. We expect this kind of behavior for many numerical applications also. The message-passing library that we have used has not been customized for the workstation cluster. Consequently, it introduces heavy setup overheads, since a message has to traverse several layers of the protocol stack before it is actually transmitted or received. A number of vendors and third-party companies are now providing efficient support for message-passing libraries. With such efficient libraries, we can expect much better performance from workstation clusters. The results that we have obtained with noncustom PVM show that it is cost-effective to use a workstation cluster for the application as opposed to single node of the Y-MP, since the former costs only a fraction of the latter. Since the computation and communication requirements of the application that we have studied are typical of many numerical applications, and since the LACE test bed is representative of emerging workstation clusters, we expect that our conclusion holds for other scenarios that involve developing scientific computations on workstation clusters.

It has to be noted, however, that we have not included the software development cost—parallelizing or vectorizing an application on the Cray Y-MP is simple compared to parallelizing the same on a workstation cluster. Parallelizing the already vectorized application took 4–5 days on the Y-MP; the same task took approximately 2 months on the cluster. We expect, however, that with the availability of sophisticated software tools, this development cost will significantly reduce. With improvement in network technology and with the current emphasis on cost-effectiveness, a workstation cluster will be a strong candidate as a parallel processor and, in many cases, an alternative to traditional supercomputers.

References

[1]Hayder, M. E., Flannery, W. S., Littman, M. G., Nosenchuck, D. M., and Orszag, S. A., "Large Scale Turbulence Simulations on the Navier–Stokes Computer," *Computers and Structures*, Vol. 30, No. 1/2, 1988, pp. 357–364.

[2]Landsberg, A. M., Young, T. R., and Boris, J. P., "An Efficient, Parallel Method for Solving Flows in Complex Three Dimensional Geometries," AIAA Paper 94-0413, Jan. 1994.

[3]Morano, E., and Mavriplis, D., "Implementation of a Parallel Unstructured Euler Solver on the CM-5," AIAA Paper 94-0755, Jan. 1994.

[4]Venkatakrishnan, V., "Parallel Implicit Unstructured Grid Euler Solvers," *AIAA Journal*, Vol. 32, No. 10, 1994, pp. 1985–1991.

[5]Horowitz, J. G., "Lewis Advanced Cluster Environment," *Proceedings of the Distributed Computing for Aerosciences Applications Workshop*, NASA Ames Research Center, Oct. 1993.

[6]Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V., "PVM 3 User's Guide and Reference Manual," Oak Ridge National Lab., ORNL/TM-12187, Oak Ridge, TN, 1993.

[7]Hayder, M. E., Jayasimha, D. N., and Pillay, S. K., "Parallel Navier–Stokes Solutions of Shared and Distributed Memory Architectures," AIAA Paper 95-0577, NASA TM 106823, Jan. 1995.

[8]Jayasimha, D. N., Hayder, M. E., and Pillay, S. K., "Parallelizing Navier–Stokes Computations on a Variety of Architectural Platforms," *Supercomputing'95*, San Diego, CA, Dec. 1995; available at http://www.supercomp.org/sc95/Proceedings/.

[9]Gottlieb, D., and Turkel, E., "Dissipative Two–Four Methods for Time Dependent Problems," *Mathematics of Computation*, Vol. 30, 1976, pp. 703–723.

[10]Hayder, M. E., and Turkel, E., "High Order Accurate Solutions of Viscous Problems," AIAA Paper 93-3074, NASA TM 106267, July 1993.

[11]Hayder, M. E., Turkel, E., and Mankbadi, R. R., "Numerical Simulations of a High Mach Number Jet Flow," AIAA Paper 93-0653, Jan. 1993; also NASA TM 105985, Jan. 1993.

[12]Mankbadi, R. R., Hayder, M. E., Povinelli, L. A., "The Structure of Supersonic Jet Flow and its Radiated Sound," *AIAA Journal*, Vol. 32, No. 5, 1994, pp. 897–906.

[13]Hayder, M. E., and Turkel, E., "Boundary Conditions for Jet Flow Computations," AIAA Paper 94-2195, June 1994; also NASA TM 106648, June 1994.

[14]Breit, S., Celmaster, W., Coney, W., Foster, R., Gaiman, B., Selvidge, C., and Montry, G., "The Role of Architectural Balance in the Implementation of the NAS Parallel Benchmarks on the BBN TC2000 Computer," *CFD Algorithms and Applications for Parallel Processors*, FFD, Vol. 156, American Society of Mechanical Engineers, 1993.