NASA Technical Memorandum 113158
ICOMP–97–12

Parallelizing Navier-Stokes Computations on a Variety of Architectural Platforms

D.N. Jayasimha
Ohio State University
Columbus, Ohio

M.E. Hayder
Institute for Computational Mechanics in Propulsion
Cleveland, Ohio

S.K. Pillay
Lewis Research Center
Cleveland, Ohio

Prepared for
Supercomputing '95
cosponsored by ACM, SIGARCH, and the IEEE Computer Society
San Diego, California, December 3–8, 1995

National Aeronautics and
Space Administration

Parallelizing Navier-Stokes Computations on a Variety of Architectural Platforms

D. N. Jayasimha *
Dept. of Computer and Information Science
The Ohio State University, Columbus, OH 43210
jayasim@cis.ohio-state.edu

M. E. Hayder
ICOMP, Ohio Aerospace Institute
NASA Lewis Research Center
Cleveland, OH 44142
fshyder@icomp.lerc.nasa.gov

S. K. Pillay
Scientific Engg. Computing Solutions Office
NASA Lewis Research Center
Cleveland, OH 44142
spillay@lerc.nasa.gov

*Part of this work was done while this author was a Visiting Senior Research Associate at NASA Lewis Research Center during 1993-94.

Abstract

We study the computational, communication, and scalability characteristics of a Computational Fluid Dynamics application, which solves the time accurate flow field of a jet using the compressible Navier-Stokes equations, on a variety of parallel architectural platforms. The platforms chosen for this study are a cluster of workstations (the LACE experimental testbed at NASA Lewis), a shared memory multiprocessor (the Cray YMP), distributed memory multiprocessors with different topologies— the IBM SP and the Cray T3D. We investigate the impact of various networks, connecting the cluster of workstations, on the performance of the application and the overheads induced by popular message passing libraries used for parallelization. The work also highlights the importance of matching the memory bandwidth to the processor speed for good single processor performance. By studying the performance of an application on a variety of architectures, we are able to point out the strengths and weaknesses of each of the example computing platforms.

1 Introduction

Numerical simulations play an important role in the investigation of physical processes associated with many important problems. The suppression of jet exhaust noise is one such problem which will have a great impact on the success of the High Speed Civil Transport plane. The radiated sound emanating from the jet can be computed by solving the full (time-dependent) compressible Navier-Stokes equations. This 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 (see [1]) to relate the far-field noise to the near-field sources. This technique requires obtaining the time-dependent flow field. In this study we concentrate on such flow fields near the nozzle exit. We solve the Navier Stokes equations to compute time accurate flow fields of a supersonic axisymmetric jet. Our code is computationally very intensive and requires many hours of CPU time on the Cray Y-MP. With the advent of massively parallel processors and networks of workstations (NOWs), scientists now have the opportunity to parallelize computationally intensive codes and reduce turnaround time at a fraction of the cost of traditional supercomputers. Recognizing this, a number of researchers [2, 3, 4, 5] have studied CFD (Computational Fluid Dynamics) applications on specific parallel architectures. Our goal in this study is to implement the numerical model derived from the CFD application described above on a variety of parallel architectural platforms.

The platforms chosen for this study, all from the NASA Lewis Research Center, represent a spectrum of parallel architectures that have been proposed to solve computationally intensive problems: a shared memory vector multiprocessor (the Cray YMP), two distributed memory multiprocessors with different topologies— the IBM SP and the Cray T3D, and a cluster of workstations connected via many networks (the Lewis Advanced Cluster Environment (LACE) [6] experimental testbed). One important architecture that has not been considered in our study is cache-coherent, massively parallel processors typified by the DASH architecture [7].

Architectures such as LACE (an example of NOW) are becoming increasingly popular because they show promise as a low cost alternative to expensive supercomputers and massively parallel processors. We have therefore laid more emphasis on this aspect of the study in this paper. An earlier paper by the authors [8] presented the initial results of a study on LACE and the Y-MP. This paper differs from the earlier one in two important aspects: i) It is comprehensive covering a gamut of architectures. ii) It focuses on the relationship of the computation and communication characteristics of the application, and hence its performance, to the architectural aspects of the networks and the processing nodes.

In the next two sections we briefly discuss the governing equations and the numerical model of the application. Section 4 has a discussion of the parallel architectures used in the study and the tools used for parallelizing the application. The parallelization of the application is the subject of Section 5. Section 6 describes the experimental methodology. Section 7 presents a

detailed discussion of the results. The paper concludes with a brief discussion of the lessons learned from this study.

2 Governing Equations

We solve the Navier-Stokes and the Euler equations to compute flow fields of an axisymmetric jet. The Navier-Stokes equations for such flows can be written, in polar coordinates as

$$LQ = S$$

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

where

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

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

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

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

F and G are the fluxes in the **x** and **r** directions respectively, and S is the source term that arises in the cylindrical polar coordinates, τ_{ij} are the shear stresses and κT_j are the heat fluxes. In the above equations p, ρ, u, v, T, e and H denote the pressure, density, axial and radial velocity components, temperature, total energy and enthalpy. For a perfect gas,

$$E = \frac{p}{(\gamma - 1)} + \frac{1}{2}\rho(u^2 + v^2)$$

$$H = \frac{E + p}{\rho}$$

where γ is the ratio of specific heats. One obtains the Euler equations from the above equations, by setting κ and all τ_{ij} equal to zero.

3 Numerical Model

We use the fourth order MacCormack scheme, due to Gottlieb and Turkel [9], to solve the Navier-Stokes and the Euler equations. This scheme uses predictor and corrector steps to compute time accurate solutions. It uses one sided differences (forward or backward) to compute spatial derivatives at each predictor or corrector step. For the present computations, the operator L in the equation $LQ = S$ or equivalently $Q_t + F_z + G_r = S$ is split into two one-dimensional operators and the scheme is applied to these split operators. 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. The predictor step in $L_1 Q$ for the one dimension model/split equation $Q_t + F_z = S$ is written as

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

and the corrector step as

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

Similarly in L_2 the predictor step is

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

and the corrector step is

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

This scheme becomes fourth-order accurate in the spatial derivatives when alternated with symmetric variants [9]. 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. In order 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. We use the characteristic boundary condition at the outflow. In our implementation, we solve the following set of equations to get the solution at the new time for all boundary points.

$$p_t - \rho c u_t = 0$$

$$p_t + \rho c u_t = R_2$$

$$p_t - c^2\rho_t = R_3$$

$$v_t = R_4$$

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 Navier-Stokes equations. Thus R_i contains viscous contributions even though the basic format is based on inviscid characteristic theory. In implementing these differential equations we convert them to conservation variables ρ, m = ρu, n = ρv, and E. Assuming an ideal gas,

$$p_t = (\gamma - 1)(E_t + \frac{u^2 + v^2}{2}\rho_t - um_t - vn_t)$$

$$u_t = \frac{m_t}{\rho} - \frac{u\rho_t}{\rho}$$

$$v_t = \frac{n_t}{\rho} - \frac{v\rho_t}{\rho}$$

For subsonic outflow, we calculate R_2, R_3,R_4 from the Navier-Stokes equations. For supersonic flows, all the R_i at the outflow boundary can be calculated from the Navier-Stokes equations or by extrapolation of all the characteristic variables from the interior. This framework of outflow boundary condition implementation is discussed by Hayder and Turkel [10]. Further discussions of our numerical model including other boundary treatments are given in Hayder et al. [11] and Mankbadi et al. [12].

In this study, we consider a jet with the mean inflow profile

$$\bar{U}_r = U_\infty + (U_c - U_\infty)g_r$$

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

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

where θ is the momentum thickness. The subscripts c and ∞ refer to the centerline and free stream 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

$$U(r,t) = \bar{U}(r) + \epsilon Re(\hat{U}e^{i\pi S_t t})$$

$$P(r,t) = \bar{P}(r) + \epsilon Re(\hat{P}e^{i\pi S_t t})$$

$$\rho(r,t) = \bar{\rho}(r) + \epsilon Re(\hat{\rho}e^{i\pi S_t t})$$

$$V(r,t) = \epsilon Re(\hat{V}e^{i\pi S_t t})$$

$\hat{U}$, $\hat{V}$, $\hat{\rho}$ and $\hat{P}$ are the eigenfunctions of the linearized equations with the same mean flow profile, ϵ is the excitation level and S_t is the Strouhal number.

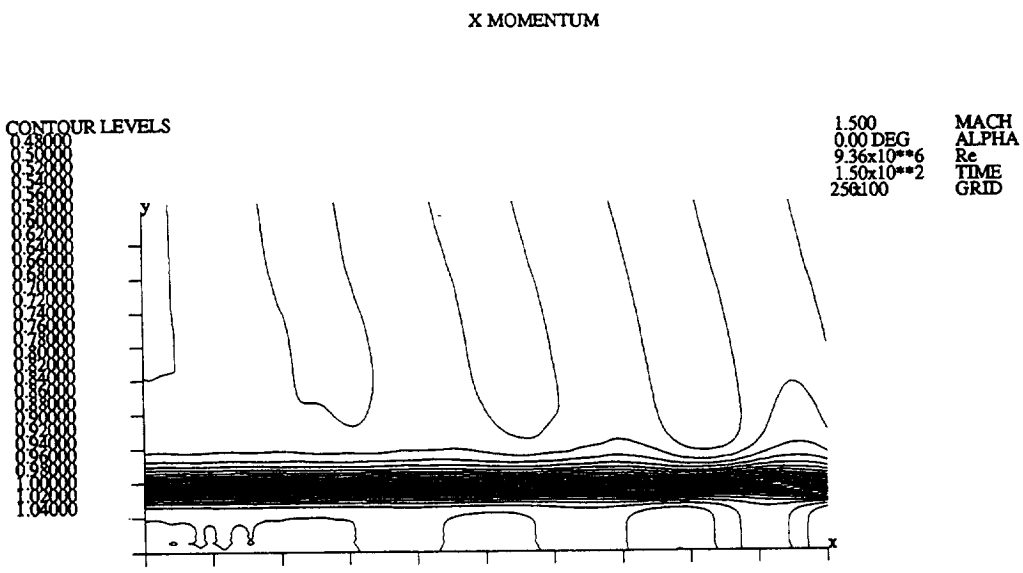

Figure 1: Axial momentum in an excited axisymmetric jet

We consider a case with $\frac{U_\infty}{U_c} = \frac{1}{4}$, $\frac{T_\infty}{T_c} = \frac{1}{2}$, momentum thickness,$\theta = \frac{1}{8}$ and Strouhal number,$S_t = \frac{1}{8}$. The jet center Mach number is 1.5 while the Reynolds number based on the jet diameter is 1.2 million. Our present mean flow and eigen function profiles are same as those in Scott et al. [16].

In Figure 1 we show a contour plot of axial momentum from the solution of the Navier Stokes equations. A grid of size 250x100 was used in this computation. This result was obtained after about 16,000 time steps. For all other results in this paper, we have used the same grid, but ran the experiments for 5000 time steps to keep the computing requirements reasonable.

4 Parallel Computing Platforms

This section contains a brief discussion of the various platforms used in the study together with the parallelization tools used.

4.1 NOW

The LACE testbed is continually upgraded. The present configuration has 32 RS6000 processor nodes (nodes 1–32) and an RS6000/Model 990 (node 0) which 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 networks (10 Mbits/sec (Mbps)), 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 Mbps. 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 (the CPU has a 66.5 MHz clock, 256KB data- and 32KB instruction caches) with the following networks interconnecting the nodes: an ATM network capable of a peak bandwidth of 155 Mbps and IBM's ALLNODE switch, referred to as ALLNODE-F (for fast), capable of a peak throughput of 64 Mbps per link. The upper half has the slower RS6000/Model 560 CPUs (the CPU has a 50 MHz clock, 64KB data- and 8KB instruction caches) and is connected through IBM's ALLNODE prototype switch, referred to as ALLNODE-S (for slow), capable of a peak throughput of 32 Mbps per link. The ALLNODE switch is a variant of Omega interconnection network and is capable of providing multiple contentionless paths between the nodes of the cluster (a maximum of 8 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 MB, 128 MB, 256 MB, and 512 MB). We have used the popular PVM (Parallel Virtual Machine) message passing library (version 3.2.2) to implement our parallel programs. We will refer to the LACE cluster with RS6000/Model 560 processors as the LACE/560 and those with the RS6000/Model 590 processors as the LACE/590.

4.2 Shared Memory Architecture

We used the Cray Y-MP/8, which has eight vector processors, for this study. The Cray Y-MP/8 has a peak rating of approximately 2.7 GigaFLOPS. It offers a single address space and the communication between processes executing on different processors is through shared variables. We parallelized the application by using explicit DOALL directives in addition to exploiting the features of the parallelizing compiler on the Cray.

4.3 Distributed Memory Architecture

We parallelized the application on two distributed memory multiprocessors– the IBM SP1 and the Cray T3D. The IBM SP1 has 16 processing nodes (the CPU at each node is a RS6K/370– the CPU has a 50 MHz clock, 32KB data and instruction caches). The original system has been software upgraded to make it function like a SP2. We will refer to this system as the IBM SP in the paper. The nodes of the SP are interconnected through a variant of the Omega network [14]. This network, similar in topology to ALLNODE, permits multiple contentionless paths between nodes. We parallelized the application using MPL (Message Passing Library), IBM's native message passing library and PVMe, a customized version of PVM (version 3.2) developed by IBM for the SP.

The Cray T3D is also a distributed memory multiprocessor with the topology of a three dimensional torus [15]. The machine used in our study has 64

nodes ($8 \times 4 \times 2$) (each node has a CPU with a clock speed of 150 MHz and a direct mapped cache of 8KB) of which only 16 were available in single user mode. Though the T3D supports multiple programming models, we programmed the machine using the message passing paradigm resorting to Cray's customized version of PVM (version 3.2).

5 Parallelization

The factors which affect parallel performance are listed below.

1. Single processor performance: we will explain various optimizations which resulted in 80% improvement in performance.

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, usually leads to **finer** granularity of communication which then leads to a higher number of startups.

4. Bursty communication: 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.

From the above discussion it is clear that there is a subtle relationship among communication startup cost, overlapping communication with computation, and bursty communication. After some experimentation, we chose to decompose the domain by **blocks** along the axial direction only.

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

The computational and communication requirements of the application are shown in Table 1. It is seen that the solution of Euler equations, hereafter referred to as **Euler**, has roughly 50% of the computation and roughly 75% of the communication requirements of **Navier-Stokes**. Note that the communication requirements are shown on a **per processor** basis. To give some idea of the effects of communication, consider **Navier-Stokes** to be executing on a network of 10 workstations connected via Ethernet. Assume a reasonable throughput of 20 MFLOPS per processor and the maximum throughput of

Table 1: Application Characteristics

Appln	Total Comp. (in FP Ops (x 10^6))	Comm./Processor	
		Start-ups	Volume (MB)
N-S	145,000	80,000	125 (1000Mb)
Euler	77,000	60,000	95 (760Mb)

Table 2: Computation-Communication Ratios

No. of Procs.	FPs/Byte		FPs/Start-up	
	Nav-Stokes	Euler	Nav-Stokes	Euler
1	∞	∞	∞	∞
2	580	405	906K	642K
4	290	203	453K	321K
8	145	101	227K	161K
16	73	51	113K	80K

10 Mbps for Ethernet. The computation time will then be approximately 725 seconds $(145,000/(10 \times 20))$ while a **lower bound** on the communication time, ignoring the effect of startups, is 1000 seconds $(1000 \times 10/10)$! Table 2 shows the ratio of computation to communication for the application in units of floating point operations/byte transferred per processor and floating point operations/startup per processor.

The parallelization on the Cray Y-MP was done differently (it was much easier also) since it is a shared memory architecture: we did some hand optimization to convert some loops to parallel loops, used the DOALL directive, and partitioned the domain along the orthogonal direction of the sweep to keep the vector lengths large and to avoid non-stride access to most of the variables.

6 Experimental Methodology

The performance indicator is the total execution time for **Navier-Stokes** and **Euler**. All experiments were conducted in single user mode. In almost all the experiments, **Navier-Stokes** and **Euler** show similar trends; hence, unless otherwise mentioned, quantitative comments refer to **Navier-Stokes**.

Experiments using a single processor were done on an IBM RS6K (Model 560) workstation of LACE. The performance of the original code for both applications is shown in Figure 2.

We found that most parts of the application were limited by the poor performance of the memory hierarchy involving the cache and the main memory.

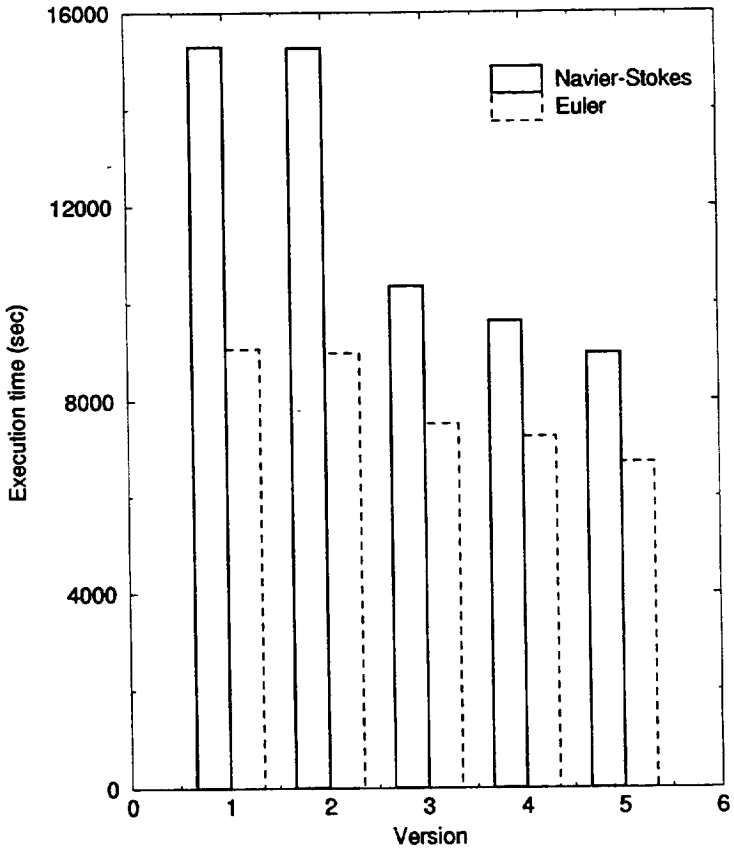

Figure 2: Execution time on a single processor (RS6000/560)

Improved cache performance was the key and this was achieved by accessing arrays in **stride-1** fashion wherever possible (using the loop interchange optimization). The modified program, called Version 3 (the optimizations were performed in a different order than presented in the paper), resulted in this version running faster by approximately 50%, compared to Version 2. 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 (Version 5), strength reduction (replace exponentiations by multiplications wherever feasible– Version 2), replace division by multiplication wherever feasible since the former are relatively expensive (a reduction from 5.5×10^9 divisions to 2.0×10^9 was achieved– Version 4). All these optimizations yielded an overall improvement of roughly 80% (from 9.3 MFLOPS to 16.0 MFLOPS) as illustrated in Figure 2. (The optimizations were incorporated in sequence so that Version 5 contains all the above mentioned optimizations).

We parallelized Version 5 on different computing platforms in accordance with the ideas presented in the last section. On each platform, we measured the execution time as a function of the number of processors (up to 8 with Cray Y-MP, up to 16 with LACE, IBM SP, and Cray T3D). One important goal of the study has been to examine the scalability of NOW. Toward this end, we have studied the performance of LACE with four networks of differing characteristics using "off-the-shelf" PVM as the message passing library.

With the IBM SP, we have studied the impact of parallelizing the application with two message passing libraries- IBM's native MPL and a customized version of PVM called PVMe.

In all experiments, wherever feasible, we have separated the execution time into two additive components: **processor busy time** and **non-overlapped communication time**. 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 non-overlapped communication time could also include the idle time of a processor waiting for a message.

Version 5 of the application does not make any special attempts to overlap communication with computation. Version 6 does overlapping by computing the stress and flux components of the interior part of each subdomain while the processor is waiting for the velocity and temperature vectors from its neighbors. As mentioned earlier, the two "flux columns" nearest each boundary are combined into a single send. We have experimented with sending the flux columns one at a time to avoid bursty communication. This variant is called Version 7.

We found that the execution time improvement with Versions 6 and 7 were either minimal or even worse in many experiments. Hence all our experiments were conducted with Version 5. We do mention, however, the impacts of these versions on different networks of LACE.

The next section presents a detailed discussion of the results of our experiments.

7 Results

The execution times of **Navier-Stokes** and **Euler** have been plotted as a function of the number of processors for each computing platform, using a log-log scale to facilitate meaningful presentation.

7.1 Performance of LACE

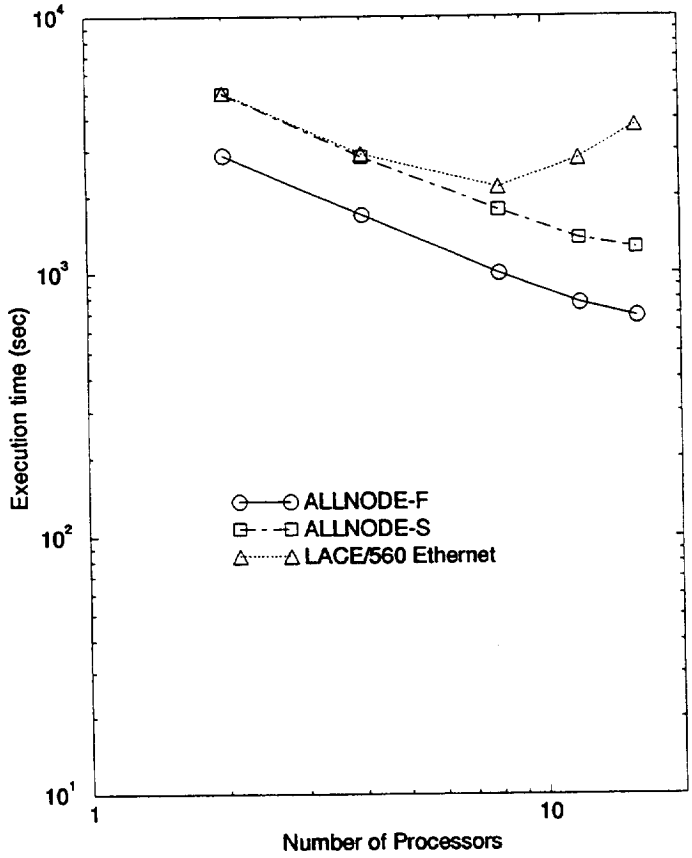

Figure 3: **Navier-Stokes** execution time on LACE

Figures 3 and 4 show the performance of **Navier-Stokes** and **Euler** respectively on different networks of LACE- ALLNODE-F, ALLNODE-S, and the upper-half Ethernet. The performance of the ATM and the FDDI networks are almost identical with ALLNODE-F and ALLNODE-S respectively. Hence the performance of the ATM and FDDI networks are not shown.

The close performance of ALLNODE-F and ATM, and ALLNODE-S and FDDI can be attributed to the following reason: the slower link speed of ALLNODE (64 Mbps/32 Mbps) is balanced by its ability to set up multiple contention-free paths while ATM (155 Mbps) or FDDI (100 Mbps) with their faster links do not permit multiple physical paths in the network.

The execution time falls almost linearly with increasing number of processors with ALLNODE– sublinearity effects begin to show, however, beyond 12

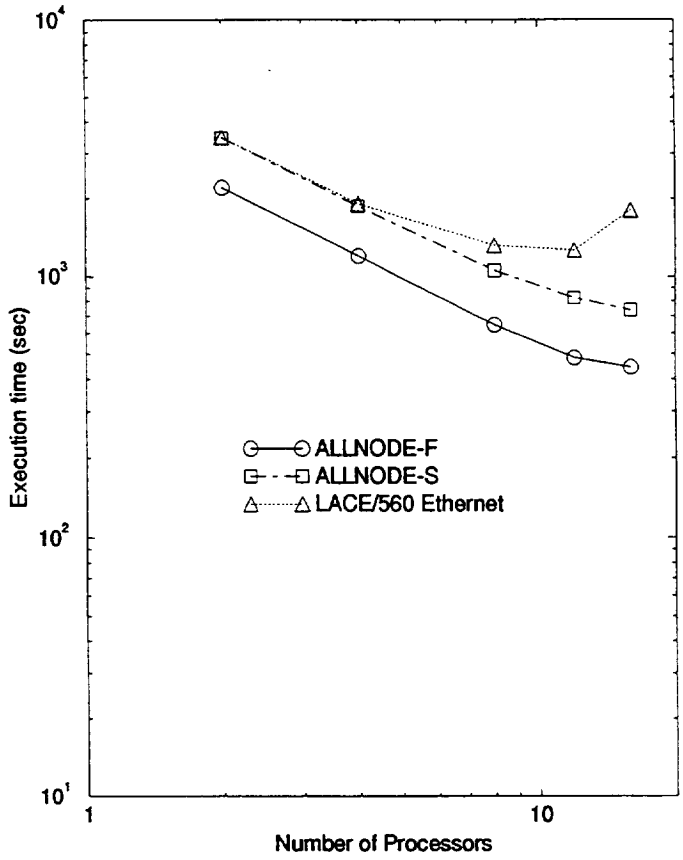

Figure 4: **Euler** execution time on LACE

processors. The LACE setup does not permit the use of more than 16 processors using the faster networks. ALLNODE-F is about 70%–80% faster than ALLNODE-S. This can be attributed to both an improved network (which is twice as fast) and the superior performance of the 590 model (33% faster clock, data and instruction caches which are 4 times bigger, and memory bus which is 4 times wider than the 560– these contribute to faster instruction execution, better cache hit ratios, and lower cache miss penalty respectively).

Not surprisingly, Ethernet performance reaches its peak at 8 processors for **Navier-Stokes** and at 10 processors for **Euler**. Beyond this, the communication requirements of the application overwhelm the network. The inability of Ethernet to handle traffic beyond 8 processors is shown by the following simple argument: Table 2 shows that with 8 processors, **Navier-Stokes**, on the average, produces a byte for communication after it has completed 145 floating operations on the average. Consider a 1 second interval and each processor operating at 20 MFLOPS. During this interval, each processor produces 0.14 MB or 1.12 Mb for communication, on the average. This translates to approximately 9Mbs from all the 8 processors. Ethernet is capable of supporting 10Mbps peak; it is not surprising, therefore, that Ethernet's performance gets steadily worse beyond 8 processors.

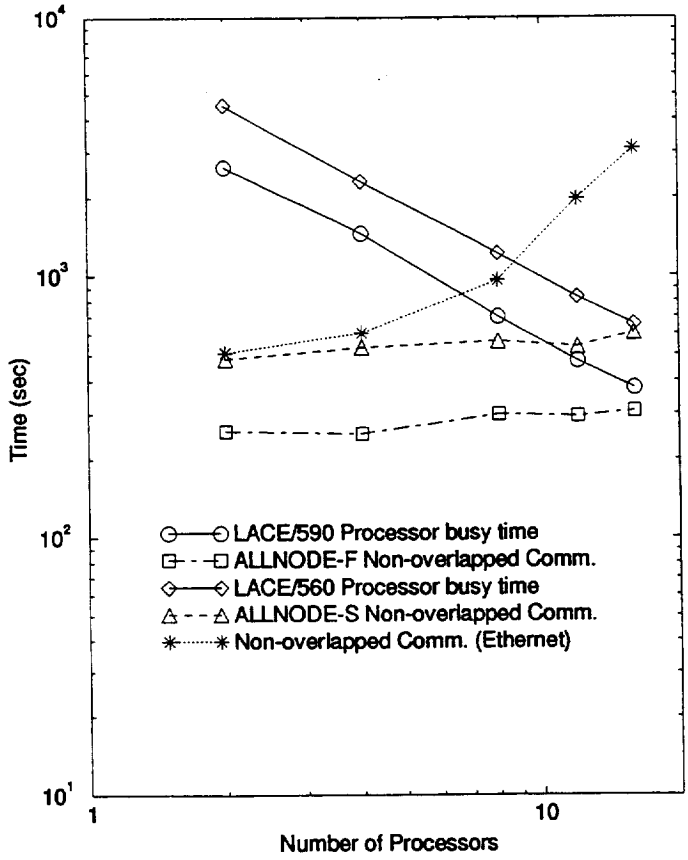

Figure 5: Components of execution time (**Navier-Stokes**; LACE)

Figures 5 and 6 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. It is seen that the processor busy time falls linearly

with the number of processors. With Ethernet, the non-overlapped communication time increases superlinearly with the number of processors. With

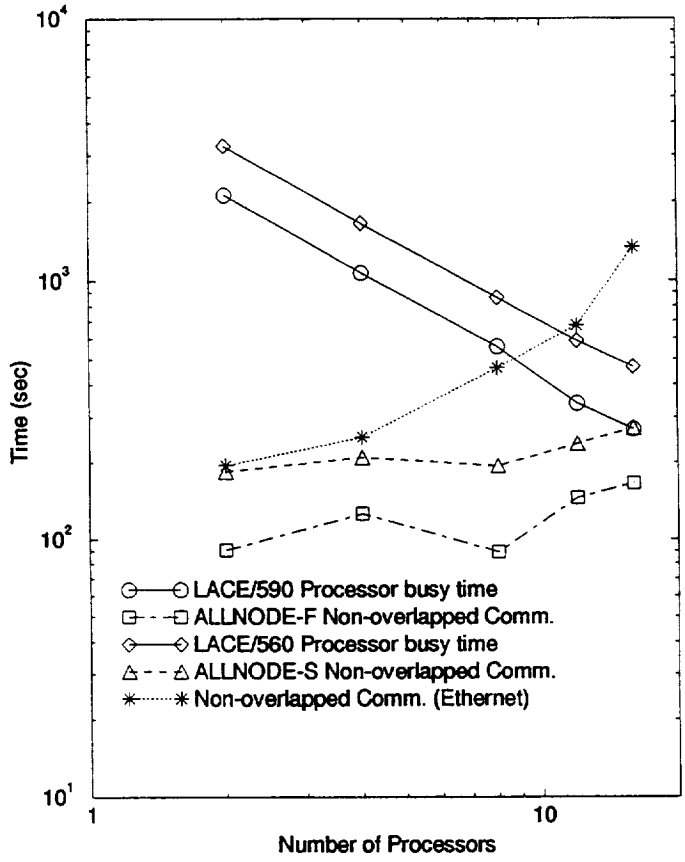

Figure 6: Components of execution time (**Euler**; LACE)

both ALLNODE switches, this time remains steady up to 10 or 12 processors beyond which it begins to rise. For **Navier-Stokes** with 16 processors, the communication time is comparable to the computation and PVM setup time while the ratio is about 60% for **Euler**. The difference in processor busy times and the communication times between the two ALLNODE configurations can be attributed to the superior node and the network respectively. A detailed examination of the data shows that both of these enhancements together contribute to the overall improved performance.

Figures 7 and 8 show the performance of Versions 5, 6, and 7 with Ethernet and ALLNODE-S (the trends are similar with ALLNODE-F). The performance of Version 6 (with overlapped communication and computation as explained in Section 6) is very close to that of Version 5 for both Ethernet and ALLNODE-S. Overlapping does not increase the number of communication startups. With Version 6, since computations for the subdomain have to be broken into separate ones for the interior and the boundary (only the former computations can be overlapped with communication), the loop setup overheads are higher. Further, the cache performance also degrades slightly due to loss of temporal locality. Consequently, these overheads offset any gain due to overlapping.

Version 7 attempts to reduce bursty communication at the cost of increased

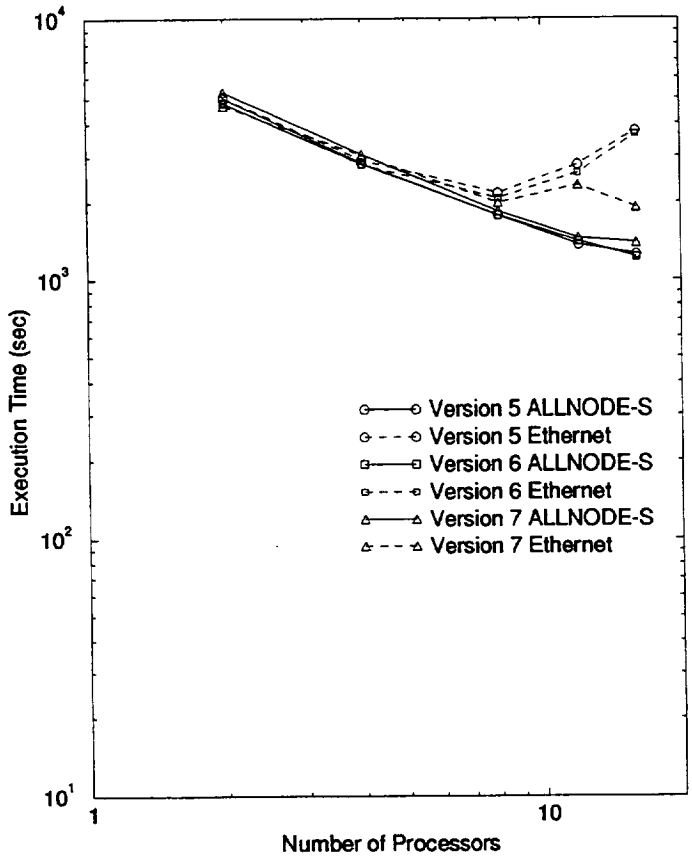

Figure 7: Communication optimization (**Navier-Stokes**; LACE)

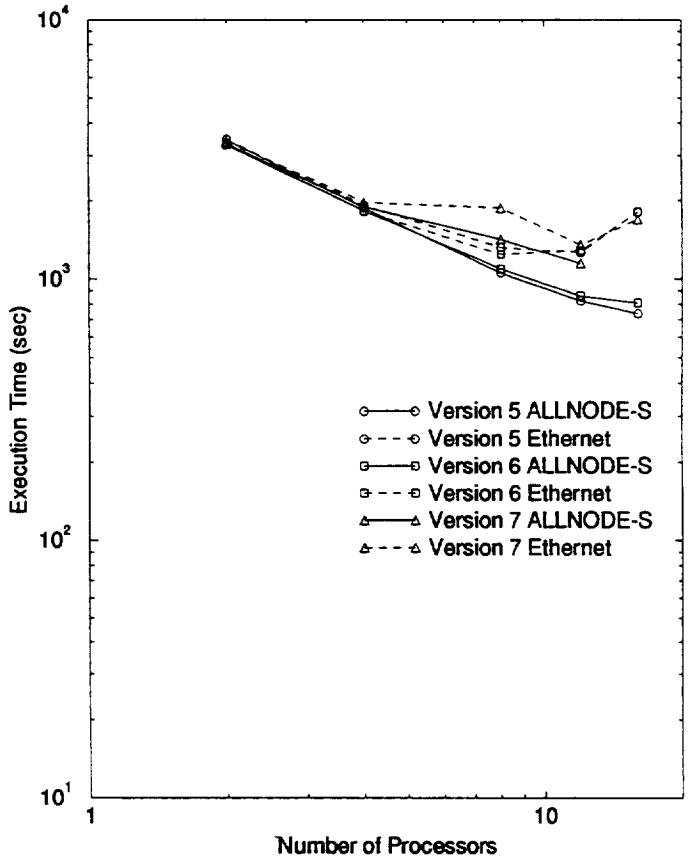

Figure 8: Communication optimization (**Euler**; LACE)

number of communication startups. Not surprisingly, Ethernet performs better with Version 7 than with Version 5. The performance of ALLNODE-S is appreciably worse than Version 5, however. Since ALLNODE-S can handle the communication requirements of the application, reducing bursty communication only harms the performance since the number of startups increase.

7.2 Comparative Performance

Figures 9 and 10 show the performance of the application on the four computing platforms we have chosen for this study- LACE, Cray Y-MP, Cray T3D and IBM SP. The performance of LACE is reported for ALLNODE-F and ALLNODE-S.

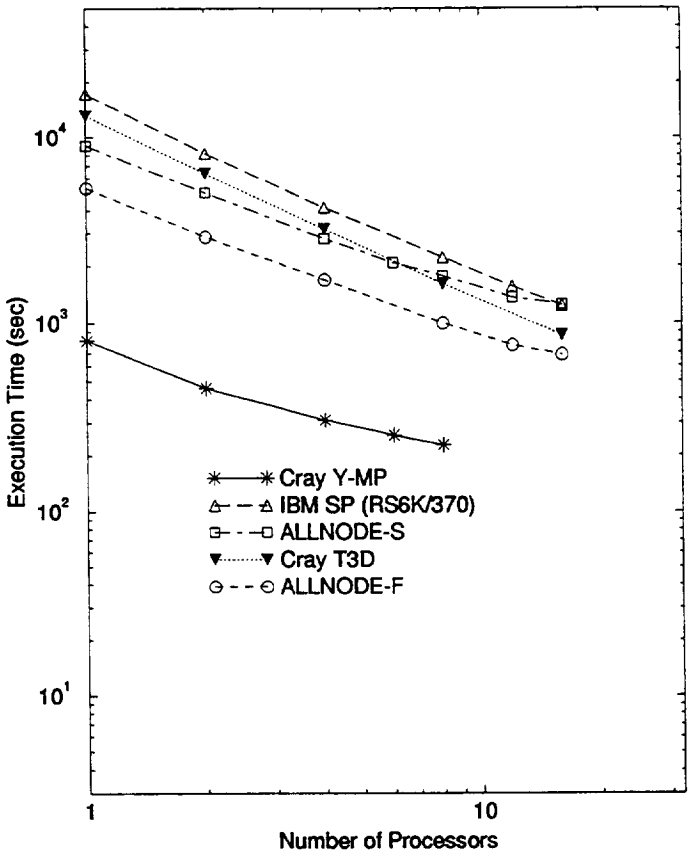

Figure 9: Execution time of **Navier-Stokes** on computing platforms

Surprisingly, LACE, even with ALLNODE-S, outperforms SP even though the former uses off-the-shelf PVM and the latter uses MPL, IBM's native message passing library. With (our version of) MPL, we were forced to use either blocking **send** or a constrained form of non-blocking **send** (for our communication requirements, both of the **send** primitives give similar results). This could possibly be one contributing factor to the relatively poor performance. The CPU on the SP is intermediate in speed (62.5 MHz clock) between the 560 (50 MHz) and the 590 (66.6MHz). Another contributor to

the poor performance of the SP is attributable to the data cache which is just 32KB (compared to 64KB on LACE/560 and 256KB on LACE/590).

For a comparison of ALLNODE-F and ALLNODE-S, see Section 7.1 (Figures 3, 4, 5, and 6).

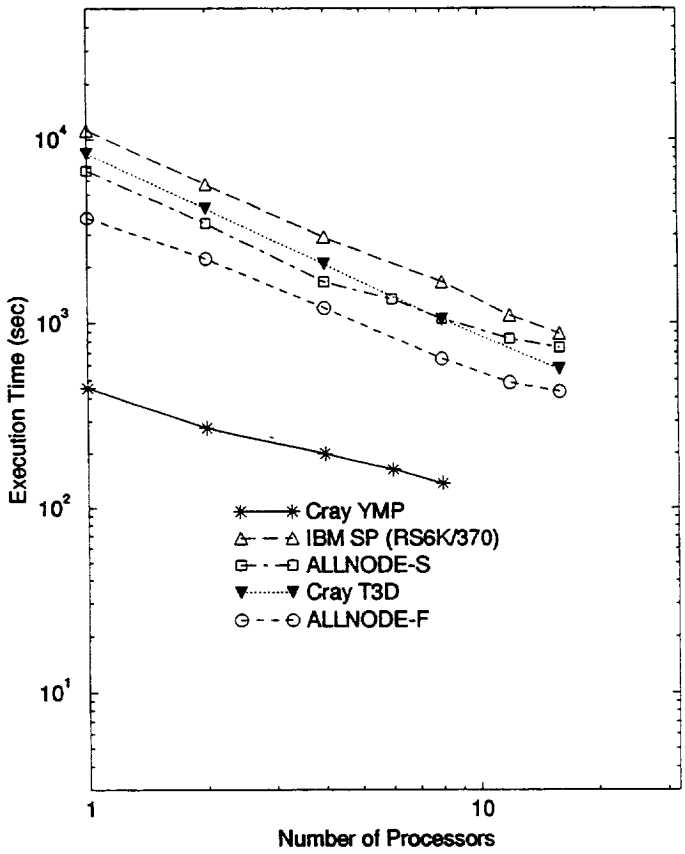

Figure 10: Execution time of **Euler** on computing platforms

Another surprising result is the relatively poor performance of Cray T3D which is consistently worse than ALLNODE-F and is worse than ALLNODE-S for less than 8 processors. The T3D's CPU has a peak rating which is 2.3X and 3X the rating of the 590 and 560 models respectively. We attribute the T3D's poor performance to the small direct-mapped cache of 8KB size (both the 560 and 590 have 4-way set-associative date caches of sizes 64KB and 256 KB respectively; in addition they have 2-way set associative instruction caches of sizes 8KB and 32KB). Poor single processor performance on the T3D has also been reported elsewhere [17]. Beyond 8 processors, T3D with its superior network speed (150 MB/sec peak transfer rate and a relatively small setup cost) performs better than ALLNODE-S. The T3D is still superior to the IBM SP.

The above results stress the importance of superior cache design to the overall performance.

Both T3D and SP exhibit very good speedup characteristics, with an almost linear drop in the execution time– indicating that the corresponding networks can sustain the communication requirements of the application. With

ALLNODE, a flattening of the speedup is seen beyond 12 processors. It is only reasonable to expect this trend to continue with increasing number of processors.

With architectures which use message passing libraries, the relatively poor performance can be attributed to large setup overheads and the resulting increase in processor waiting times with increasing number of processors. These overheads arise mainly from the multiple times that data to be communicated is copied and from the context switching overheads that arise in transferring a message between the application level and the physical layer of the network for transmission or reception. If NOW architectures are to be feasible as massively parallel processors, it is clear that both the interconnection network and the message passing library be implemented efficiently. Such effort is already under way [18].

Cray Y-MP has by far the best performance. The execution time shown is the connect time in single user mode (this includes the I/O time also which we were not able to separate from the computation time). The performance of LACE/590 with 16 processors is comparable to the single node performance of the Y-MP. The Y-MP (with a maximum of 8 processors) scales quite well for the applications.

7.3 Comparison of Message Passing Libraries

Figures 11 and 12 compare the performance of the PVMe and the MPL message passing libraries on the SP. The graphs show that MPL is consistently faster than PVMe by approximately 75% for **Navier-Stokes** and approximately 40% for **Euler**. Observe also that the amount of non-overlapped communication is not only negligibly small but that it decreases with the number of processors though the actual communication increases. This is an interesting phenomenon since it implies that there is increased overlapping of computation and communication with the number of processors. Note however that the computation part also includes the setup overheads of communication. This phenomenon is not seen in case of LACE (see Figures 3 and 4) where the non-overlapped communication increases, further attesting to our previous observation that the MPL (and PVMe) library does not perform as well as PVM does on LACE.

7.4 Load Balancing

Finally, how well is the application load balanced? The amount of computation for the application is evenly distributed but this may not always translate to a load balanced execution. We were able to measure the processor busy times (this time does **not** include the processor waiting time) for **Navier-Stokes** on each processor of the SP. Figure 13 shows that we were able to achieve almost perfect load balancing.

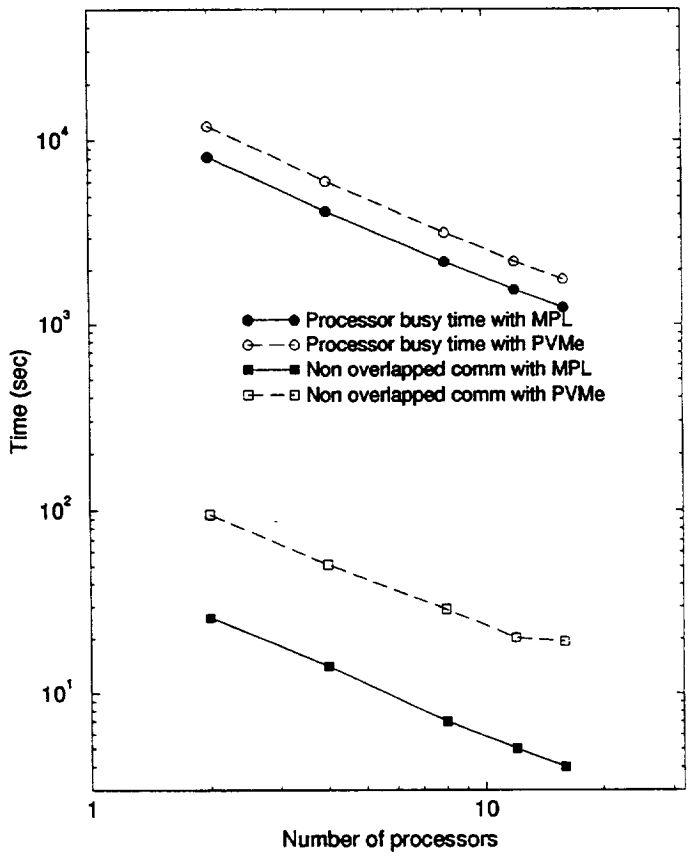

Figure 11: Comparison of MPL and PVMe (**Navier-Stokes**; IBM SP)

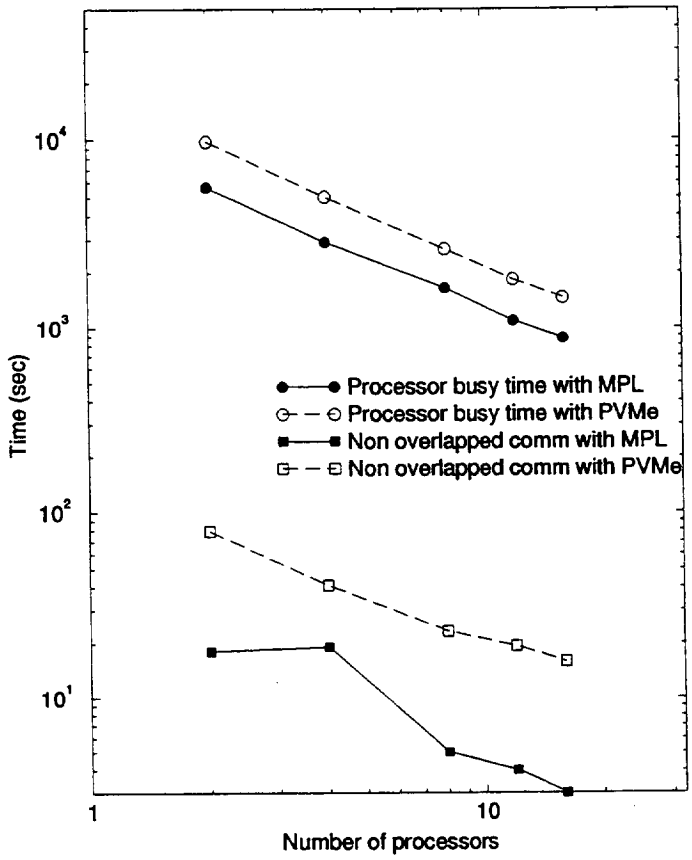

Figure 12: Comparison of MPL and PVMe (**Euler**; IBM SP)

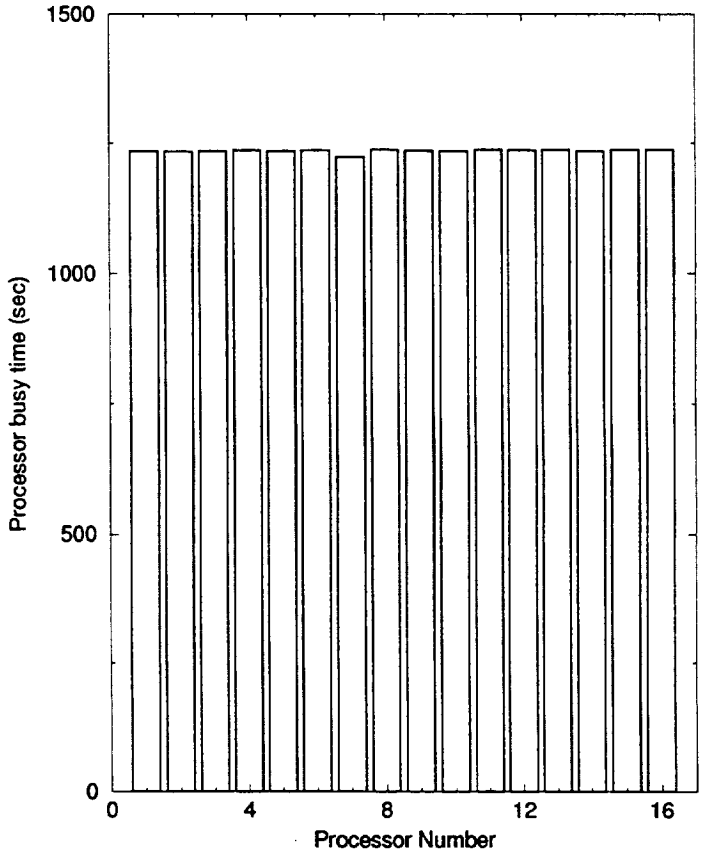

Figure 13: Processor busy times (**Navier-Stokes**; IBM SP)

8 Conclusion

In this paper we have studied the computational, communication, and scalability characteristics of a typical CFD application on a variety of architectural platforms. The study indicates that NOW have the potential to be cost-effective parallel architectures if the networks are made reasonably fast and message passing libraries are efficiently implemented to circumvent the traditional overheads involved in transferring a message between the application level and the physical layer of the network.

The study also highlights the importance of single processor performance to achieve good performance. With fast, off-the-shelf RISC processors available, the bottleneck seems to be the performance of the cache and the memory hierarchy. A proper cache design is critical to good performance. We believe that the reason for relatively poor performance of the T3D, in spite of a fast processor, is the small, direct-mapped cache.

A traditional vector multiprocessor still outperforms multiprocessors of modest to medium size. Parallelizing an application using message passing libraries is rather tedious and even error-prone but with distributed memory multiprocessors, this effort is worthwhile since good scalability is achievable.

Resource limitations have forced us to limit our study to 16 processors. We hope to extend the study to larger multiprocessors and to other parallelization tools as resources become available. We will then explore other problem decompositions such as blocking along the radial direction, for example, and study their impact on the performance.

Acknowledgments

The authors would like to thank Kim Ciula, Dale Hubler, and Rich Rinehart for their assistance with various aspects of the LACE and IBM SP architectures.

References

[1] Lighthill, M. J. "On Sound Generated Aerodynamically, Part I, General Theory". *Proc. Roy. Soc. London*, vol. 211, 1952, pp. 564-587.

[2] 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.

[3] Landsberg, A. M., Young, T. R. and Boris, J. P. "An Efficient, Parallel Method for Solving Flows in Complex Three Dimensional Geometries". *32nd AIAA Aerospace Sciences Conference*, AIAA 94-0413, January 1994.

[4] Morano, E. and Mavriplis, D. "Implementation of a Parallel Unstructured Euler Solver on the CM-5". *32nd AIAA Aerospace Sciences Conference*, AIAA 94-0755, January 1994.

[5] Venkatakrishnan, V. "Parallel Implicit Unstructured Grid Euler Solvers". *32nd AIAA Aerospace Sciences Conference*, AIAA 94-0759, January 1994.

[6] Horowitz, J. G. "Lewis Advanced Cluster Environment". Distributed Computing for Aerosciences Applications Workshop, NASA Ames Research Center, October 1993.

[7] Lenoski, D. E., et al. "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor". *Int'l Conf. on Computer Architecture*, May 1990, pp. 148–159.

[8] Hayder, M. E., Jayasimha, D. N. and Pillay, S. K. "Parallel Navier-Stokes Computations on Shared and Distributed Memory Architectures". *33rd AIAA Aerospace Sciences Conference*, AIAA 95-0577, January 1995.

[9] Gottlieb, D. and Turkel, E. "Dissipative Two-Four Methods for Time Dependent Problems". *Math. Comp.* vol. 30, 1976, pp. 703-723.

[10] Hayder, M. E. and Turkel, E. "High Order Accurate Solutions of Viscous Problems". *31st AIAA Aerospace Sciences Conference*, AIAA 93-3074, January 1993.

[11] Hayder, M. E. Turkel, E. and Mankbadi, R. R. "Numerical Simulations of a High Mach Number Jet Flow". *31st AIAA Aerospace Sciences Conference*, AIAA 93-0653, January 1993.

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

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

[14] Stunkel, C. B., Shea, D. G., Grice, D. G., Hochschild, P.H., Tsao, M. "The SP1 High-Performance Switch", *Scalable High Performance Computing Conference*, May 1994, pp. 150–157.

[15] Oed, W. "The Cray Research Massively Parallel System- Cray T3D", *Technical Report*, Cray Research GmbH, November 1993.

[16] Scott, J. N., Mankbadi, R. R., Hayder, M. E. and Hariharan S. I. "Outflow Boundary Conditions for the Computational Analysis of Jet Noise". *31st AIAA Aerospace Sciences Conference*, AIAA 93-4366, January 1993.

[17] Bailey, D. H., Barszcz, E., Dagum, L., Simon, H. D. "NAS Parallel Benchmark Results". *NAS Technical Report NAS-94-001*, October 1994.

[18] Anderson, T. A., Culler, D. E., Patterson, D. A., NOW team, "A Case for NOW (Networks of Workstations)". *IEEE Micro*, February 1995, pp. 54 – 64.

D. N. JAYASIMHA received the B.E. degree in Electronics Engineering from the University Visvesvaraya College of Engineering, Bangalore, India, the M.E. degree in Computer Science from the Indian Institute of Science, Bangalore, India, and the Ph.D. degree in Computer Science from the Center for Supercomputing Research and Development, University of Illinois, Urbana, IL. He is an Assistant Professor at the Department of Computer and Information Science, The Ohio State University, Columbus, OH. He was a Visiting Senior Research Associate at the NASA Lewis Research Center Cleveland, OH during 1993-94. His research interests are in various aspects of parallel computing and parallel architectures, and in distributed sensor networks. He is a member of the ACM, IEEE, and Sigma Xi.

M. EHTESHAM HAYDER received his BS from Bangladesh University and MS from Tulane University in Mechanical Engineering. He then received his Ph.D. from Princeton University in Mechanical and Aerospace Engineering. His doctoral dissertation was on parallel computations of computational fluid dynamics algorithms on, and evaluation of the architecture of, a research supercomputer, known as the Navier Stokes Computer. After finishing his Ph.D., he worked as a research associate at Princeton before joining the Institute for Computational Mechanics in Propulsion program at NASA Lewis Research Center, Cleveland, OH. He is now a senior research associate at this institute. His current research interests are in the area of computational aero-acoustics and parallel computations.

Dr. SASI KUMAR PILLAY currently manages all scientific and engineering computing that is provided through the Central Computing Organization at the NASA Lewis Research Center. This includes parallel processing environments. In the past, Dr. Pillay has managed work groups responsible for data acquisition and analysis, data networking, mainframe systems, high performance computing and computer graphics.

Dr. Pillay received his Ph.D. in Computer Engineering from Case Western Reserve University and a M. S. in Management of Technology from Sloan School of Management of the Massachusetts Institute of Technology. Dr. Pillay is also the recipient of the NASA Exceptional Service Medal.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1997	Technical Memorandum

4. TITLE AND SUBTITLE

Parallelizing Navier-Stokes Computations on a Variety of Architectural Platforms

5. FUNDING NUMBERS

WU-523-36-13-00

6. AUTHOR(S)

D.N. Jayasimha, M.E. Hayder, and S.K. Pillay

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E-10908

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-113158
ICOMP-97-12

11. SUPPLEMENTARY NOTES

Prepared for Supercomputing '95 cosponsored by ACM, SIGARCH, and the IEEE Computer Society, San Diego, California, December 3–8, 1995. D.N. Jayasimha, Ohio State University, Department of Computer and Information Science, Columbus, Ohio 43210; M.E. Hayder, Institute for Computational Mechanics in Propulsion, Cleveland, Ohio; and S.K. Pillay, NASA Lewis Research Center. Responsible person, S.K. Pillay, organization code 7100, (216) 433–9300.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 64

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

We study the computational, communication, and scalability characteristics of a Computational Fluid Dynamics application, which solves the time accurate flow field of a jet using the compressible Navier-Stokes equations, on a variety of parallel architectural platforms. The platforms chosen for this study are a cluster of workstations (the LACE experimental testbed at NASA Lewis), a shared memory multiprocessor (the Cray YMP), distributed memory multiprocessors with different topologies—the IBM SP and the Cray T3D. We investigate the impact of various networks, connecting the cluster of workstations, on the performance of the application and the overheads induced by popular message passing libraries used for parallelization. The work also highlights the importance of matching the memory bandwidth to the processor speed for good single processor performance. By studying the performance of an application on a variety of architectures, we are able to point out the strengths and weaknesses of each of the example computing platforms.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Parallel computing; Cluster computing; Navier-Stokes computations	30
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	