Clustered Workstations and their Potential Role as High Speed Compute Processors

Report RNS-94-003 April 1994

A report of the NAS Distributed Computing Team
(in alphabetical order)
Karen Castagnera[§], Doreen Cheng[†], Rod Fatoohi[†], Edward Hook[†], Bill Kramer[§], Craig Manning[†], John Musch[§], Charles Niggley[†], William Saphir[†], Douglas Sheppard[#], Merritt Smith[¶], Ian Stockdale[†], Shaun Welch[#], Rita Williams[§], David Yip[§]

Team Leader
William T.C. Kramer
Chief, NAS Computational Services Branch
NAS Systems Division
NASA Ames Research Center
Moffett Field, CA 94035
(415) 604-4600
kramer@nas.nasa.gov

Members of the NAS Distributed Computing Team

Karen Castagnera[§]	Doreen Cheng[†]	David Dinucci[†]	Rod Fatoohi[†]
Karen Gundy-Burlet[¶]	Edward Hook[†]	Louise Kokinakis[#]	Bill Kramer[§]
Craig Manning[†]	John Musch[§]	Charles Niggley[†]	William Saphir[†]
Douglas Sheppard[#]	Grant Smith[*]	Merritt Smith[¶]	Ian Stockdale[†]
Shaun Welch[#]	Rita Williams[§]	David Yip[§]	

Abstract

This is a report of the work of the NAS Distributed Computing Team, and some conclusions about the viability and role of workstation cluster computing for solving areoscience problems. The report discusses the year-long activity at NAS to implement a large, loose cluster of workstations from the existing SGI pool of systems. The team's goals and approach are discussed, followed by reported results for both the NAS Parallel Benchmarks and other computationally intensive work using the environment.

[§] NAS Systems Division, NASA Ames Research Center, Moffett Field, CA 94035
[†] Computer Sciences Corporation, NASA Contract NAS 2-12961, Moffett Field, CA 94035-1000
[#] Sterling Federal Systems, NASA Contract NAS 2-13619, Moffett Field, CA, 94035-1000
[¶] Fluid Dynamics Division, NASA Ames Research Center, Moffett Field, CA 94035
[*] Thermal Sciences Institute NASA Contract NAS 2-14031, Eloret, Moffett Field, CA, 94035-1000

Table Of Contents

1.0 ..... Introduction 1
1.1 ..... The Distributed Computing Team 1
1.2 ..... Objective 2
1.3 ..... Approach 3
1.3 ..... Overview of the Work 4
2.0 ..... PVM - The Parallel Virtual Machine 4
2.1 ..... How PVM Works 5
2.2 ..... Advantages and Disadvantages of PVM 5
3.0 ..... Batch Systems 5
3.1 ..... DQS 7
3.2 ..... Condor 9
3.3 ..... The DCF Configuration 10
4.0 ..... DCF from the Application Developer Viewpoint 11
4.1 ..... OVERFLOW-PVM 11
4.2 ..... NAS Parallel Benchmarks 14
4.2.1 ..... Cost Comparisons 22
4.2.2 ..... DCF Network Usage for the NPB 24
4.3 ..... Porting Cray Codes to Workstations 25
4.4 ..... Summary of Other Users 28
4.4.1 ..... DQS/PVM Users 28
4.4.2 ..... Condor Users 29
5.0 ..... DCF Usage Analysis 30
5.1 ..... Measuring System Usage 30
6.0 ..... Summation of the DCF Experience 31
7.0 ..... The Role of Cluster Computing in Industry 32
8.0 ..... Expectations for Future Clustered Systems 32
8.1 ..... Processor Speed 32
8.2 ..... System Software 33
8.3 ..... Application Software 34
8.4 ..... Distributed Parallel I/O 34
8.5 ..... Networks 34
8.6 ..... Physical Connection/Protocol 34
8.7 ..... Network Protocols 35
8.8 ..... Message-Passing Libraries 35
9.0 ..... Summary 35

1.0 Introduction

The Numerical Aerodynamic Simulation (NAS) Systems division at NASA Ames is designed as one of the most advanced supercomputing environments for a national client base of scientists working primarily in the field of computational fluid dynamics and related disciplines. The goals of the NAS program are:

"1) to provide a national computational capability available to NASA, the Department of Defense (DOD) and other government agencies, industry, and universities, as a necessary element in ensuring continuing leadership in computational fluid dynamics and related disciplines;

2) to act as a pathfinder in advanced, large-scale, computer systems capability through systematic incorporation of state-of-the-art improvements in computer hardware and software technologies; and

3) to provide a strong research tool for NASA's Office of Aeronautics and Space Technology." [1]

The NAS Processing System Network (NPSN) began operation in September 1985, with the arrival of the first Cray Research, Incorporated (CRI), Cray-2. In October of 1988, NAS received the first customer shipped CRI YMP system. It supports a national base of scientists.

NAS currently supports two CRI C90 systems, one with 16 processors, another with 8, three Convex 3820s for mass storage, a Thinking Machines Corp. (TMC) CM-5 with 128 processors, an Intel iPSC/860 with 128 processors, an Intel Paragon with 208 processors, a Convex 3420 for visualization support, various file and support servers, more than 250 Silicon Graphics Inc. (SGI) workstations and about 85 Sun workstations. The current networking technology is subnetted Ethernets, FDDI and ULTRAnet, connected by Wellfleet routers. NAS also as implemented Aeronet, a nation wide network connecting major research and industry sites that are involved in aerosciences. This network consists of a T-3 backbone and a number of leaf connections using T-1 and T-2 rates.

An early design requirement of the NAS system was to have a single, consistent user interface, regardless of the underlying hardware Therefore, all the operating systems are UNIX based. All these varieties of UNIX are based on AT&T System V[2] with the addition of TCP/IP networking[3] and Berkeley UNIX extensions such as sockets[4].

1.1 The Distributed Computing Team

At the 1993 NAS User Interface Group (UIG) meeting, several aerospace firms asked NAS to explore the possibilities of doing compute intensive tasks on distributed workstation systems. NAS had already anticipated the need to investigate this type of computing and announced the formation of the NAS Distributed Computing Team (DCT) at the meeting.

---

[1] Bailey, F. Ron: *Status and Projections of the NAS Program. Proceedings of a Workshop on Supercomputing Environments*, NASA Ames Research Center, Moffett Field, CA, June 24-26, 1986.

[2] Kevorkian, D.E., ed.: *System V Interface Definition.* Indianapolis, IN, 1985.

[3] Feinler, E.J., et.al., eds.: *DDN Protocol Handbook.* Vol 1, Defense Technical Information Center, Alexandria, VA, 1985.

[4] Laffler, S.J.: *A 4.2 BSD Interprocess Communication Primer.* Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, 1983.

There are several reasons for forming the DCT: interest in tightly coupled workstation clusters as replacements for large systems has grown, and these clusters are claimed to be a viable alternative to supercomputers. As the realities of making MPPs work become more evident, many in the industry are turning to the potential of distributed computing. The three major drivers of this trend are:

- The rate of increase in workstation CPU performance is outstripping that of MPPs and traditional supercomputers.
- Large memory workstations are available at reasonable prices.
- Many corporations have a large installed base of workstations that could be used for distributed computing.

Economic realities require companies to increase the effective use of their existing workstations. One alternative is a loosely clustered configuration.[5] For example, most workstations are used less than 25 percent of the time. Workstations have become ubiquitous and are found in almost every computing environment.

1.2 Objective

The approach NAS used was influenced by the fact that several other groups (such as NASA Lewis, Argonne Laboratory, Lawrence Livermore) were working with tightly coupled workstation clusters. While NAS has experience with Silicon Graphics system in a tightly clustered configuration such as the four 4D/380 systems connected with Ultranet and FDDI and used as general support processing systems, NAS decided to investigate loosely coupled systems because:

- NAS has a strength in effectively managing large groups of workstations.
- NAS uses automation in its workstation environment, therefore avoiding many of the tedious activities involved in maintaining a consistent environment across a large number of machines.
- There is a large need for work in this area and no major project existed to investigate it for aeroscience computations.

A cross organizational Distributed Computing Team was put together with the following mission.

> ***Establish a prototype computing environment across large groups of workstations that allows efficient computing for batch and batch/parallel jobs while not co-opting those systems from their primary role.***

This environment is called the NAS Distributed Computing Facility (DCF). While the team consisted of approximately 15 staff members from the three NAS branches, the total effort was between two and three full time equivalents (FTE) for nine months to accomplish this work. There was no additional funding was spent on this work.

---

[5] "Tightly clustered" workstations refer to systems attached to dedicated, high-speed networks and switches, while "loosely coupled" clusters refer to workstations connected with non-dedicated networks.

1.3 Approach

The specific approach was to establish a prototype loosely coupled cluster of workstations with one or both of the two major NAS workstation architectures (SGI 4D and Sun SPARC 4). Then, identify and resolve the major system management issues in providing reasonable cycle recovery from these systems without disrupting the primary function. Performance evaluation tests were run based on the NAS Parallel Benchmarks (NPB) and other codes, including *OVERFLOW-PVM*, a full-fledged Computational Fluid Dynamics (CFD) application code. This report summarizes the activities related to the prototype cluster and identifies areas that need improvement, development, and research in order to make such a system a reliable, competitive computing environment.

As the goal was to quickly establish the prototype to gain experience, existing technology was used instead of developing new technology. The overall approach of this work is represented in Figure 1.

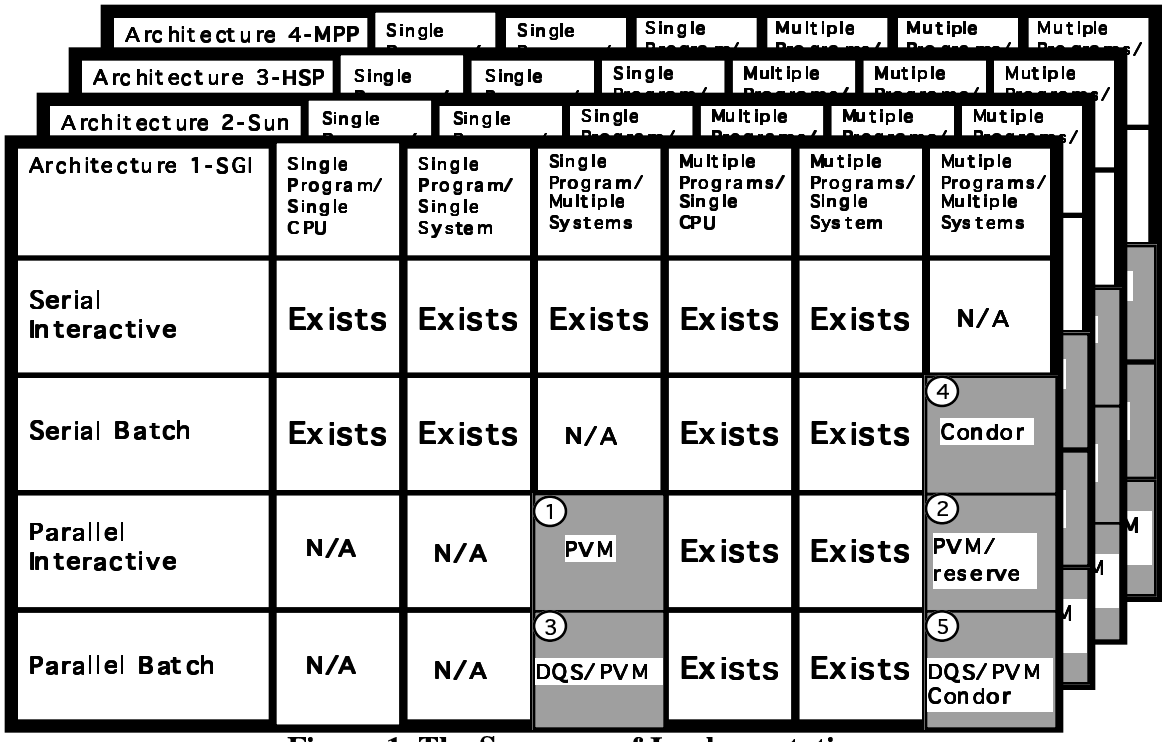


Figure 1. The Sequence of Implementation

Specific steps included:

1) Setting up a PVM environment
2) Setting up a PVM environment with a reservation system
3) Setting up a batch environment to support PVM and integrating the batch and PVM environments
4) Implementing a batch system for serial programs
5) Integrating all the environments together

Finally, the environment was expanded by increasing the number and types of systems and increasing the number of users (including remote users.) All these steps are described in more detail in subsequent sections.

1.3 Overview of the Work

The initial steps of the work emphasized organizing an environment with 50 SGI and 40 Sun workstations. The Parallel Virtual Machine[6] (PVM) software was selected as the best existing tool for message-passing and parallel program control because it was being used in a number of cluster environments, and was important because PVM versions 3.0 and later versions run on other platforms including Massively Parallel Processors (MPPs) as well as distributed workstations. Part way through the prototype evaluation (Spring 1993) the team evaluated the batch queuing systems that were available for this environment and identified the strengths and issues as described in the following sections.

The prototype has been used by a small, but growing group of users. The NAS Parallel Benchmarks have been run on the NAS cluster as well as several others. A CFD application code, *OVERFLOW-PVM*, has run in the environment since the beginning of the project. Several network experiments were conducted to judge the effectiveness of loosely clustered workstations networked via Ethernet. These and several other user experiences are discussed below.

As a specific recommendation, UIG representatives suggested that NAS sponsor a conference in this area. On October 18-20, 1993, NAS sponsored a workshop on "Distributed Computing for Aeroscience Applications." It was attended by 180 people, including 23 representatives from Aerospace companies and 44 from other government groups. The evaluation of this meeting indicated that the attendees thought the meeting was excellent. A separate report has been produced.[7]

2.0 PVM - The Parallel Virtual Machine

The Parallel Virtual Machine (PVM) is a message-passing library designed to allow a person to create a code that can run across a set of distinct and otherwise unrelated systems, whether homogeneous or heterogeneous. Given present-day operating systems, such a program is actually a set of independent processes, running on the various machines, that cooperate with one another.

Restricting attention to single UNIX systems, there are really only a few ways to manage cooperating processes on the same machine, all of which are fairly well understood. In cases where these systems contain multiple CPUs, vendor-provided software typically coordinates processes in a proprietary manner. The task is more complicated when implemented on networked UNIX systems. The cooperating processes on different machines need to communicate with one another. Programmers doing this by hand traditionally had to master TCP/IP sockets or some equivalent method and spend a good deal of effort providing communication and synchronization capabilities in their programs. Computational scientists have begun to overcome these obstacles and various tools are available to ease the difficulty of creating a multi-machine job. To date, the most popular of these tools is PVM, which was developed jointly by Oak Ridge National Laboratory, the University of Tennessee at Knoxville and Emory University, and is freely available from *netlib@ornl.gov*.

---

[6] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek
Vaidy Sunedram, *PVM 3 User's Guide and Reference Manual*, Oak Ridge National Laboratory, May 1993

[7] *Report of the NAS Workshop on Distributed Computing for Aeroscience Applications*, October 18-20, 1993.

2.1 How PVM Works

PVM presents a fairly simple picture to the user. In order to construct a "virtual machine" out of a collection of computers, the programmer invokes the PVM daemon on one computer (designated the *home* machine) and provides it a list of the component systems to be connected together. The daemon supervises all the communication details, such as starting up slave daemons on each of the hosts and establishing a socket-based communications network link. By calling routines provided in the PVM libraries, a user's program can spawn tasks on various hosts and these tasks can communicate among themselves by sending and receiving messages using straightforward mechanisms. From the programmer's perspective, a parallel virtual machine is very much like programming using a message-passing paradigm on any MPP although network of workstations joined by Ethernet and an MPP incorporating a custom high-speed back plane have very different hardware bandwidths.

2.2 Advantages and Disadvantages of PVM

The DCT chose PVM as its parallel paradigm because PVM's wide availability and usefulness have made it a defacto standard for cluster computing. Codes developed using the PVM libraries are "portable" to many other environments as long as the basic algorithm is applicable. Coming from the other direction, it is fairly easy to translate codes developed for message-passing MPPs to use PVM calls instead, so a programmer can take advantage of existing parallel methods and algorithms. Creating a cluster environment such as DCF is made easier by the existence of tools such as PVM, which has also been integrated with some utilities and job control packages.

The disadvantages of using PVM should be noted while pointing out that it is the best available software at the moment. First, PVM jobs are subject to the traffic on a general purpose network and thus can experience substantial delays. In addition, PVM's message-passing implementation incorporates several layers of software overhead in addition to the network software protocol layers. While this overhead may not be a bottleneck on an Ethernet-based network, it might become one as higher speed networks are used. PVM developers are addressing these issues. Releases of PVM beyond version 3.0 reduce some of this overhead, although absolute performance is still below that of PVM 2.4. For instance, release 3.2 allows direct TCP connections between tasks, instead of routing messages through a daemon. Future releases may allow in-place data packing, which reduces the amount of buffering required. Overhead remains a concern since PVM has to deal with a generalized environment, while comparable MPP-based message-passing libraries can assume many efficiency improving safeguards. On a slow network, such as Ethernet, the overhead introduced by PVM has a minor effect.

Other limitations include a lack of global operations and no parallel library support. The process of building the virtual machine is not as robust as it might be. For example, sometimes the daemon does not recover or report errors if one of the hosts happens to be unavailable. This was particularly true of PVM release 2.4 and has improved with releases after 3.0.

3.0 Batch Systems

Effective use of a network of computers requires intelligent management of heterogeneous resources. Running multiprocessor jobs requires a tool to work with an execution environment such as PVM, and scheduling all the processes of a single job together. It must satisfy the resource requirements of multiprocessor jobs while balancing

the workload. The batch control system for DCF project had to have the ability to dynamically allocate hosts, provide a mechanism for queuing jobs on a loosely coupled cluster of workstations and support PVM.

A major goal of the project was to investigate the recovery of compute cycles that are available on workstations, which are for the most part, dedicated to individual workers. Any software system selected had to be as unobtrusive as possible to the primary user of the workstation. It should allow tasks to run when the system is idle, but stop the task from using resources when it is determined the primary user is present and using the resources. Ideally, it would also allow for the migration of programs from a busy system to an idle system. It should provide fault tolerance to the distributed user with functions like checkpoint-restart, and provide process migration when the workstation's primary users need resources. In addition, the tool must provide user control for resource selection and monitoring.

At first, a workstation reservation system was implemented through a set of shell scripts. This provided a good service but was only viable for a small group of users since a workstation was reserved for a full night. It served until a true batch system was selected. At the same time, creation of a submission script for PVM made up for some shortcomings in PVM that were disruptive to users.

Based on the general requirements mentioned above, the DCT compiled a list of 30 useful functions and used them to compare existing resource management tools: Network Queuing System (NQS), Distributed Job Management System (DJM), Portable Batch System[8] (PBS), LSF[9] (formerly called Utopia), Distributed Queuing System (DQS), and Condor. The major differences are summarized in Table 1. The DCT also asked our users to rate the importance of these features. No single tool provided all the desired features. DJM only supported Thinking Machines Corp. platforms. Condor did not support multiprocess jobs. DQS and Utopia did not provide checkpointing. NQS did not support PVM, did not checkpoint on workstations and did not take into account interactive work. PBS was not yet implemented. Complete descriptions of these comparison, along with other parallel tools is contained in "A Survey of Parallel Programming Languages and Tools", NAS Technical Report RND-93-005, March 1993.[10]

---

[8] David Tweten, et al, *Portable Batch System External Reference Specification*, PBS Document, March 1993.

[9] Zhou, S. *LSF: Load sharing in large-scale heterogeneous distributed system*, In Proceedings of the Workshop on CLuster Computing, Tallahassee, FL., Dec. 1992.

[10] Doreen Y. Cheng, *A Survey of Parallel Programming Languages and Tools*, NAS Technical Report RND-93-005, March 1993.

Batch System	Strengths	Issues
NQS - Network Queuing System	Available on Crays, workstations, and other systems at NAS	Not integrated with PVM Not integrated with interactive work Did not checkpoint on workstations
DJM - Distributed Job Management System	Available on CM-5, CM-2	Not ported to other systems Not integrated with PVM or interactive work Did not checkpoint
PBS - Portable Batch System	Specifications were good	Not yet implemented
LSF	Supports PVM and serial jobs	Did not checkpoint
DQS - Distributed Queuing System	Integrates with PVM Good job management	Did not checkpoint
Condor	Integrates with interactive work Checkpoints on workstations	Did not support PVM Did not support shell scripts

Table 1. Summary Comparison of Cluster Batch Software

Easy access to source code became an important factor in selecting a tool since this was an evolving prototype. Both Condor and DQS are in the public domain and are used in many organizations. The DCT felt it was important to understand the workings of at least two systems so both DQS and Condor were selected. DQS is used for scheduling multiprocess jobs in the evening and weekends, and Condor used for cycle-stealing by single-process jobs all the time.

3.1 DQS

DQS[11] is an experimental UNIX-based queuing system that is being developed jointly by Supercomputer Computations Research Institute at the Florida State University, Pittsburgh Supercomputing Center, and Center for High Performance Computing in the University of Texas System

There are two main components to DQS: A *qmaster* daemon and a *dqs_execd* daemon. The *qmaster* daemon runs on a master machine to monitor the machines in the DQS pool. It is responsible for knowing if a machine in the DQS pool is available and can accept jobs, the status of the queues, which jobs are currently running in the queue if any, and the load average of the machines in the DQS pool. The *qmaster* does not run jobs; it is only an agent that handles the request for hosts. The *dqs_execd* runs on each machine in the DQS pool. It starts up at boot time and registers with the *qmaster* daemon. *Dqs_execd* communicates with the *qmaster* via a known TCP port, and is responsible for executing the jobs on its DQS host. DQS is capable of supporting multiple queues.

There are also various ancillary programs, which provide the user with an interface to DQS. These programs provide the tools needed to manage the queues, monitor, submit and remove jobs, and suspend and/or enable the various queues.

---

[11] Louis S. Revor, *DQS Users Guide*, DQS Documents, Sept. 1992 and Green and Snyder, *DQS, A Distributed Queuing System,* March 1993

Typically, a user submits a job via *qsub*. (e.g. `qsub dqsjob`). *Dqsjob* is a script file that sets the various options and defines which program to run. The job submitter selects the master controlling queue (usually his/her own machine) and the numbers and types of other queues needed.

In the DCF parallel environment, a queue corresponds to an individual processor and a user submits a parallel job to a set of queues. Another minor implementation detail was to change the name of the DQS commands. Since this was a prototype environment, in order not to confuse users or conflict with the existing COSMIC NQS 2.0 commands, each DQS command was preceded with a "d" (e.g. *qconf* became *dqconf*).

Remembering that the primary role of DQS at NAS is to reserve hosts for PVM and not all machines are in the DQS machine pool, the DCF implementation had to allow a user to include machines outside of the DQS pool even if there were no existing PVM accounts for them. Therefore, DCT added a *-ph filename* option to the *qsub* command. This option allows a user to specify a file that contains a list of hosts to be included when starting up the PVM daemon.

A NAS-specific problem was encountered with the DQS configuration. The *qmaster* was installed on a fileserver that does not have any user accounts. When a user submits a job, DQS checks to see if the user was authorized by checking for an account on the *qmaster* host. The check would fail and the user would not be allowed to submit the job. This checking was disabled based on the assumption (possibly invalid in the general case) that if the host is trusted, then the user submitting the job is trusted.

Another NAS-specific problem with DQS/PVM is that each machine has its own /etc/passwd file. Because NAS has a centralized account management system[12] instead of a global password file or Yellow Pages (YP), accounts had to be set up on each workstation for the PVM users. In order not to impact the "primary users" of the workstations, the accounts are only accessible between 10pm and 5am. For the same reason, the user's home directory is NFS-mounted from the primary local workstation, or for remote users from a shared workstation. Thus, PVM jobs only consume CPU and memory resources, and have no impact either temporary or permanently on the disk space of the workstation. To implement this, a *cron* job is run at 10pm on each machine in the DQS pool to check the DQS queues, and any user who has a job queued has their account enabled. Similarly, a *cron* job turns the account off at 5am. A current problem with this solution is that a user cannot submit a job after 10pm and have their account enabled, which in turn means their job will fail. A solution, which is currently being worked on, is to have the *qmaster* or *dqs_execd* daemon enable and disable the accounts.

DQS is simple to install and configure. It is easy to add, delete, and modify queues. Most of the adding, deleting and modifying of the queues has been automated. Since the queue names were based on the hostname, administrative problems were encountered when, for example, a hostname changes, making the DQS queue name invalid. Startup scripts for *dqs_execd* to handle this problem.

DCF currently runs version 2.1 of DQS. Two bugs, but neither critical, need to be resolved. There appears to be an inconsistency when a user requests 32 hosts. The last host in the list gets truncated. There is also a problem when a user submits a job and requests specific queues. *dqstat* reports the last host requested by the user as the controlling queue, when in fact it is usually not the case.

---

[12] *NAS Login Account Mangement Manual,* NAS Computational Services Branch, 1989.

One of the issues with the DQS/PVM setup for DCF is that DQS only verifies that a host is available and can accept DQS jobs. It does not check to determine if the user account is accessible or that PVM started up correctly. If DQS monitored the daemon's output for errors, this and other problems would largely disappear. In the end, DQS does a good job of allowing dynamic allocation of hosts for PVM use.

3.2 Condor

Condor is a batch queue system designed to run jobs remotely on a collection of machines. One machine is the master, which monitors and executes jobs on the rest of the cluster. It currently only directly supports the execution of serial jobs. Condor was written at the University of Wisconsin.[13]

Condor runs on a variety of platforms. The version used in the DCF initially was an offshoot of the official version. The prototype cluster that was established for the DCF was originally composed only of SGI systems. A special version of Condor exists for this platform.

There were several problems bringing it up this version of Condor. Checkpointing did not work on SGIs. Specifically, a job would start off correctly. Condor would checkpoint it after a period of time to save its status in case of a system failure. Thereafter the job would only accumulate system time and no user time and eventually it would abort and start over again. The temporary solution was to turn off checkpointing. Currently a job suspends, but never checkpoints or migrates. Condor was set up so that it would never try to migrate a job since this would abort it. A job now runs on a particular system until it completes, no matter how long or often it is suspended. Work is going on to properly correct the bug.

The Condor build for the Sun Platform was straightforward, as this release was well maintained by the authors. Only the typical modifications had to be made to the *imakefile* and *makefile* for the compiler options and the location of the binaries. It was built on a SPARCstation 2 and moved to fileservers for distribution to the user systems.

An SGI system was identified as the master for the Condor cluster; both the SGI and Sun architectures use this same master. Accounting is accumulated for both architectures on the individual machines. Unlike DQS, which maintains a central log, Condor keeps a log on every system. Due to different binary formats between the SGI and Sun versions the accounting procedures must be executed once for each architecture.

The DCF experience with Condor has been mixed. After the initial problems with checkpointing were solved, the first test user reported that the system was working well. Currently, there appears to be intermittent problems with some jobs not completing, and others aborting unexpectedly. Many jobs normally take a few days to run. The fact that some jobs abort needs to be investigated. This is particularly significant since job migration has not been implemented. Another issue is that primary users are running out of memory when they use their machines while a Condor job is suspended. Whether we have to increase the size of swap or find some other solution, this issue needs to be addressed before the DCF environment can be expanded.

Staff members are working with the authors of Condor on the issue of checkpointing on SGIs. Once this is resolved, most of the current difficulties should be resolved.

---

[13] Allan Bricker, Michael Litzkow and Miron Livny, *Condor Technical Summary*, Condor Documents, Sept. 1991.

3.3 The DCF Configuration

The DCF Condor cluster consists of 31 machines, 9 SGIs and 22 Suns. There are 49 SGIs in the DQS environment. Tables 2 and 3 summarizes DCF system types and capabilities.

System	Number of Systems	Processor	Average Memory Size (megabytes)	Average Swap Space (megabytes)
4D/RPC	14	R3000	48	50
4D/25	11	R3000	16	40
4D/30	2	R3000	20	42
4D/35TG	4	R3000	32	50
4D/320	5	R3000	48	40
4D/380	5	R3000	256	750
4D/4**	2	R3000	256	195
4D/RPC	4	R4000	64	50
Indigo2 (4DTWO50EX)	2	R4000	48	130
4D/CR	2	R4000	96	128
Sparc1	13	SPARC	28	12
Sparc1+	2	SPARC	24	16
Sparc2	9	SPARC	32	31

Table 2. NAS Distributed Computing Facility Systems

		MIPS	MFLOPS (DP Linpack)				
	MHz	VAX Drystone	1000 by 1000	100 by 100	SPEC89	fp92	int92
SGI Configurations:							
4D25G	20	16			14		
4D25TG	20	16			14		
4D30TG	30	27		4.7	27		
4D35TG	36	33		6	31		
4D320VGX	2x33	59	20		47		
4D380S	8x33	234	60		128		
4D380VGX	8x33	234	60		128		
4D420	2x40	72	23		56		
4D440IG2	4x40	143	42		106		
4DCRVGX	50	85		16	70	61.5	58.3
4DRPC	33	30		4.2	26	24.2	22.4
4DRPC50	50	85		16	70	60.5	58.3
4DRPC50EG	50	85		16	70	60.5	58.3
4DRPCEG	33	30		4.2	26	24.2	22.4
4DTWO50EX (Indigo2)	50	85	16			60.6	58.6
Sun Configurations:							
Sun 4/60							
Sun 4/65		16			12		
Sun 4/75		29	4		25		

Table 3. NAS Distributed Computing Facility System Descriptions

These systems add up to between 10 and 15 percent of the processing capability and almost twice the memory of CRI Y-MP/8256 that was in place at NAS until October 1993.

4.0 DCF from the Application Developer Viewpoint

This section discusses the experiences of several users the DCF and other cluster systems. The 4.1 discusses a large CFD application that was one of the earliest uses of the DCF. The results in 4.2 include NAS Parallel Benchmarks (NPB) work funded by NAS that was done on other cluster environments in addition to work done on the NAS DCF. 4.3 explores the issues with porting vectorized codes that run very successfully in the Cray environment to workstations. The 4.4 is a summary of impressions and experiences from some of the later users of the DCF.

4.1 OVERFLOW-PVM

As a test application, one of us ported the multimethod overset grid flow solver OVERFLOW[14] to the DCF, calling it first MEDUSA and then *OVERFLOW-PVM*[15]. The parallel code uses a manager-worker control paradigm with parallelism extracted at the grid level. Every grid is operated upon by a separate process, each of which communicates with the manager process at the completion of each solution step. PVM communications are used throughout. These processes are distributed among the processors of the DCF by a static load-balancing routine included in the manager process, to better utilize the available resources. While this parallel decomposition is not generally scalable, the coarse granularity provides considerable efficiency considering the problems of message latency and limited network bandwidth common to cluster computing systems.

A system of nine SGI R3000 and R4000 workstations computed the flow over the wing of the AV-8B Harrier fighter aircraft using OVERFLOW-PVM on a grid system of over 0.5 million points. Grids of this size are typical for aircraft component performance analyses. With this system, a sustained single-precision computation rate of 50.5 million floating point operations per second (MFLOPS) has been achieved, representing more than 40 percent of the combined maximum LINPACK MFLOPS rate (indicated in Table 3 above) of the machines used. Figure 2 suggests that larger problems with more grids should achieve proportionately greater performance.

---

[14] Bunning, P. G., and Chan, W. M., *OVERFLOW/F3D User's Manual*, NASA Ames Research Center, March 1991.

[15] Smith, M. H. and J. M. Pallis, *MEDUSA- An Overset Grid Flow Solver for Network-based Parallel Computer Systems*, AIAA-93-3312, , July 1993.

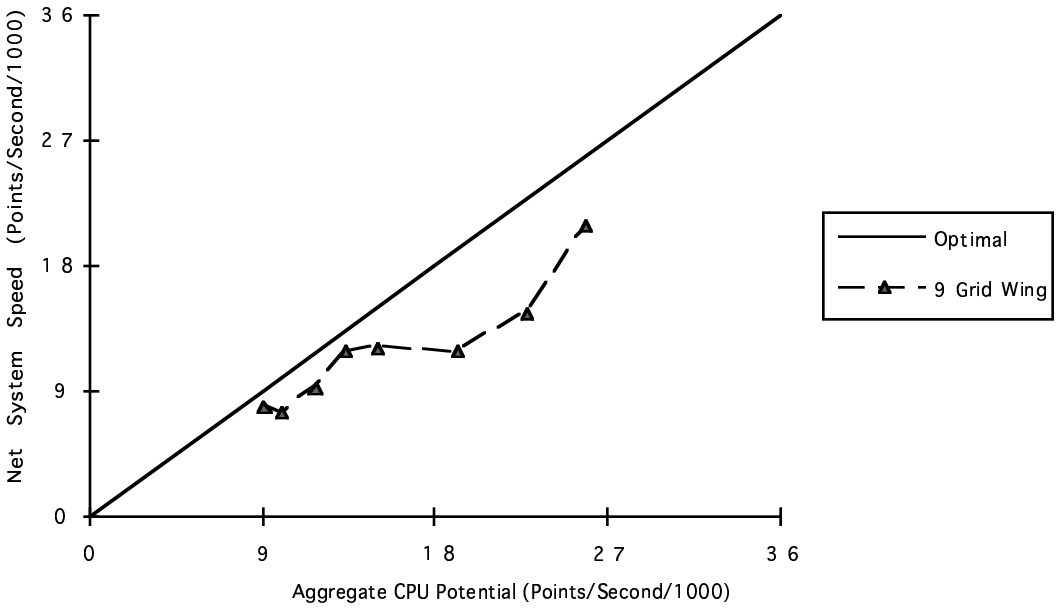


Figure 2 - Larger Scale Usage for Overflow-PVM

The same problem was run on two other cluster configurations; two HP 755s and a combination of six SGI R4000s and three SGI R4400s. The solution times for these runs were 1529 seconds and 1042 seconds respectively. Using the "pixie" utility, the problem is estimated to have 54.3 billion floating point operations, which yields effective MFLOP rates off 35 MFLOPS for the two HPs and 52 MFLOPS for the 9 SGIs.

From a user's perspective, the greatest deficiency in the NAS DCF is the lack of debugging and performance monitoring tools. At this time, performance statistics have been generated using the UNIX *time* command. *Top* and the SGI tool *gr_osview* are used to determine the activities of the cluster. *Gr_osview* provides a graphical display of system resources including, but not limited to, CPU and memory usage, and CPU wait time. *Gr_osview* is implemented using a client-server model, which allows for remote monitoring of system performance. The graphical nature of *gr_osview* and the fact that monitors for multiple machines may be displayed simultaneously, creates an easily understood display of an application's operation on the cluster. Analogous tools designed specifically for cluster computing are needed. Figure 3 shows a typical run of *OVERFLOW-PVM* on three systems using *gr_osview*.

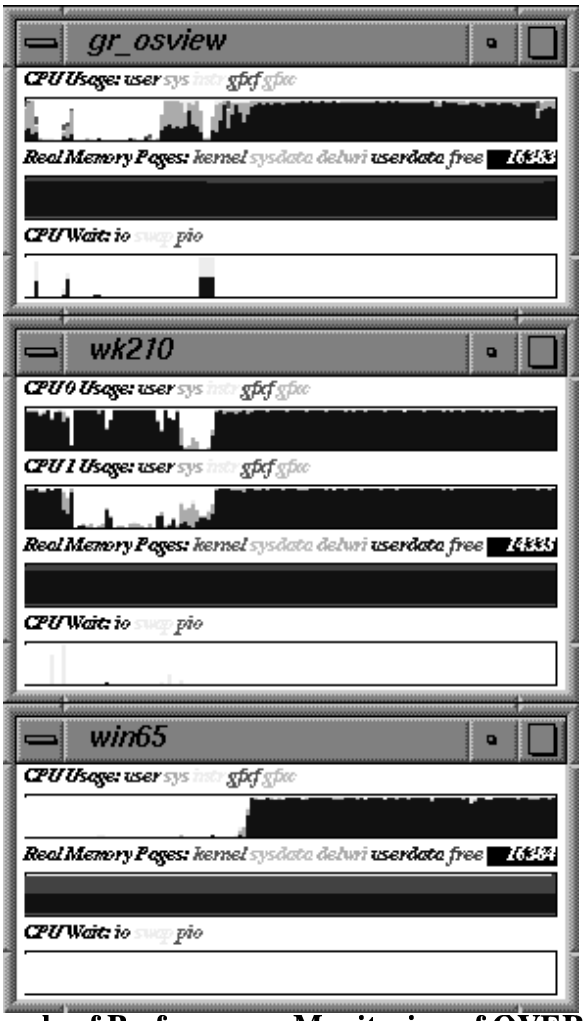


Figure 3. Example of Performance Monitoring of OVERFLOW-PVM

Debugging in the cluster environment is significantly more difficult than on serial machines. Not only must logical and typographical errors be found, but the timing of operations can be essential to the proper functioning of a parallel code. Users currently have the option of either using the *dbx* debugger on individual processes, or inserting well-placed write statements to assess the status of an application's execution. In most instances, the write statements have proven the most useful. Unfortunately, debugging in this manner results in an enormous number of recompilations to remove a single bug. To further confound debugging, a program that fails, due to a segmentation violation and dumps core, often will not completely flush the output buffer, leaving an incomplete record of the run.

4.2 NAS Parallel Benchmarks

The NAS Parallel Benchmarks[16] were implemented on clusters of workstations using PVM. The five kernels of NPB (EP, MG, CG, FT, IS) were implemented on three cluster environments characterized as low-speed shared, medium-speed shared, and medium-speed switched. These environments consisted of:

- A low-speed shared environment consisting of 16 Sun Sparcserver 1+ workstations connected by Ethernet at Emory University.
- A medium-speed shared environment consisting of seven IBM RS/6000 model 560 workstations and one RS/6000 model 320 workstation connected by FDDI at Utah Supercomputing Institute.
- A medium-speed switched environment consisting of eight SGI R4000 computers connected by DEC FDDI Gigaswitch at Sandia National Labs.

The implementation of these kernels on these environments was performed by Vaidy Sunderam and his colleagues at Emory University under a grant from NAS Applied Research Branch[17]. The results for the CRI Y-MP, Intel iPSC/860, the NAS DCF, and NASA Lewis cluster were obtained by one of us[18]. Tables 4-8 show a summary of the results obtained for the five NPB kernels (EP, MG, CG, FT and IS) on the clusters of workstations using PVM:

Computer System	Problem Size	Number of Processors	Network	Time (sec)	Ratio to Y-MP/1
CRI Y-MP	2^28	1		126	1.00
Intel iPSC/860		128		26	4.91
Sparcs		16	Ethernet	1603	0.08
RS/6000 model 560 (with one RS/6000 model 320)		8	FDDI	442	0.29
SGI R4000		8	Gigaswitch	446	0.28
NAS DCF		8	Ethernet	695	0.18

Table 4. NAS Parallel Benchmark Results for EP

[16] David H. Bailey, Eric Barszcz, Leonardo Dagum and Horst Simon, *NAS Parallel Benchmark Results 3-94*, RNR Technical Report, RNR-94-006, March 1994.

[17] White, S., Alund, A., and Sunderam, V., *The NAS Parallel Benchmarks on Virtual Parallel Machines,* NAS Report RNR-93-xxx, under review.

[18] Work was done by Rod Fatoohi, NAS Applied Research Branch

Computer System	Problem Size	Number of Processors	Network	Time (sec)	Ratio to Y-MP/1
CRI Y-MP	256^3	1		22	1.00
Intel iPSC/860		128		9	2.58
Sparcs	N/A				
RS/6000 model 560 (with one RS/6000 model 320)		8	FDDI	110	0.20
SGI R4000		8	Gigaswitch	168	0.13
SGI R3000	N/A				

Table 5. NAS Parallel Benchmark Results for MG

Computer System	Problem Size	Number of Processors	Network	Time (sec)	Ratio to Y-MP/1
CRI Y-MP	2.0*10^6	1		12	1.00
Intel iPSC/860		128		9	1.38
Sparcs		16	Ethernet	605	0.02
RS/6000 model 560 (with one RS/6000 model 320)		4	FDDI	203	0.06
SGI R4000		9	Gigaswitch	108	0.11
SGI R3000	N/A				

Table 6. NAS Parallel Benchmark Results for CG

Computer System	Problem Size	Number of Processors	Network	Time (sec)	Ratio to Y-MP/1
CRI Y-MP	256^2x128	1		29 (library result)	1.00
Intel iPSC/860		128		10 (library result)	2.96
Sparcs	N/A				
RS/6000 model 560 (with one RS/6000 model 320)		8	FDDI	412	0.07
SGI R4000		8	Gigaswitch	228	0.13
SGI R3000	N/A				

Table 7. NAS Parallel Benchmark Results for FT

Computer System	Problem Size	Number of Processors	Network	Time (sec)	Ratio to Y-MP/1
CRI Y-MP	2^23x2^19	1		11	1.00
Intel iPSC/860		128		14	0.84
Sparcs	N/A				
RS/6000 model 560 (with one RS/6000 model 320)		8	FDDI	318	0.04
SGI R4000		8	Gigaswitch	258	0.04
SGI R3000	N/A				

Table 8. NAS Parallel Benchmark Results for IS

The three simulated applications of NPB were implemented on the NAS cluster of SGI workstations and on a cluster of IBM RS/6000-560 workstations at NASA Lewis Research Center in Cleveland. The IBM cluster configuration has the ability to run jobs on Ethernet, FDDI and the IBM Allnode switch[19].

The objective of this work is to evaluate the performance of representative algorithms that involve significant amounts of communication on a loosely clustered set of workstations. The three simulated applications (LU, SP, BT) are concerned with the solution of a coupled system of PDEs on structured grids using time implicit relaxation techniques. The LU benchmark employs an SSOR scheme resulting in the solution of regular-sparse, block (5 x 5) lower and upper triangular systems. Both the BT and SP benchmarks use variants of the three-factor, approximate factorization schemes similar to the classical ADI method. In the SP benchmark this results in the solution of a sequence of multiple, independent scalar pentadiagonal systems, each oriented along one of the three mutually orthogonal directions of the computational space. The BT benchmark is a close relative of the SP benchmark, with the primary difference being the solution of block (5 x 5) tridiagonal systems, instead of scalar pentadiagonal systems. All verification tests setup for the serial code should be passed, which is a very important requirement in a NPB implementation[20].

The simulated applications have been implemented on the Intel iPSC/860 using different parallel algorithms. These algorithms have different communication requirements such as communication pattern, number of messages and message sizes. The approach taken to port these benchmarks to the NAS DCF was to develop a set of FORTRAN routines to convert Intel iPSC/860 message-passing calls to PVM 2.4 and PVM 3.2 calls. This means the same codes run on the iPSC/860, the NAS DCF and the Lewis cluster. Within the NAS DCF, up to 16 SGI machines connected by Ethernet were used. Each of these machines has at least a 33 megahertz (MHz) clock rate (refer to Table 3) and 32 MB of local memory. On multiprocessor machines, only a single processor was employed.

The parallel LU implementation is based on the skew hyperplane mapping approach and 2-D partitioning into blocks[21]. This implementation requires sending many short messages to nearest neighbors. (On four processors, a total of 15400 messages with a

---

[19] Fatoohi, Rod and Sisira Weeratunga, *Performance Evaluation of Three Distributed Computing Environments for Scientific Applications*, submitted to Supercomputer 94, April 1994.

[20] *The NAS Parallel Benchmarks*, NAS Report RNR-91-002, 1991.

[21] Barszcz, E., Fatoohi, R., Venkatakrishnan, V., and Weeratunga, S., *Solution of Regular, Sparse Triangular Linear Systems on Vector and Distributed-Memory Multiprocessors*, NAS Report RNR-93-007, 1993.

total communication volume of about 2 MB was sent every time step.) The results of this implementation in the environments (iPSC/860 and NAS DCF) using the standard problem size (64 x 64 x 64) are given in Table 9 and Figure 4.

Computer System	Problem Size	Number of Processors	Network	Time (sec)	Ratio to Y-MP/1
Y-MP	64x64x64	1		333.5	1.00
Intel iPSC/860		64		690.8	0.48
RS/6000 model 560		8	Allnode	2214.6	0.15
NAS DCF SGI R3000		8	Ethernet	11300	0.03

Table 9. NAS Parallel Benchmark Results for LU

Figure 4 shows the implementation of LU in the NAS DCF and the iPSC/860 using different numbers of processors. A speedup of about 2.5 was achieved on 4 and 16 SGI processors, respectively, so it showed no improvement beyond four machines. The results for 16 processors show that the iPSC/860 is about six times faster than the cluster. For the DCT implementation of the three pseudo-applications, the same amount of work is given to each processor, so the slowest processor in the group determines the overall speedup.

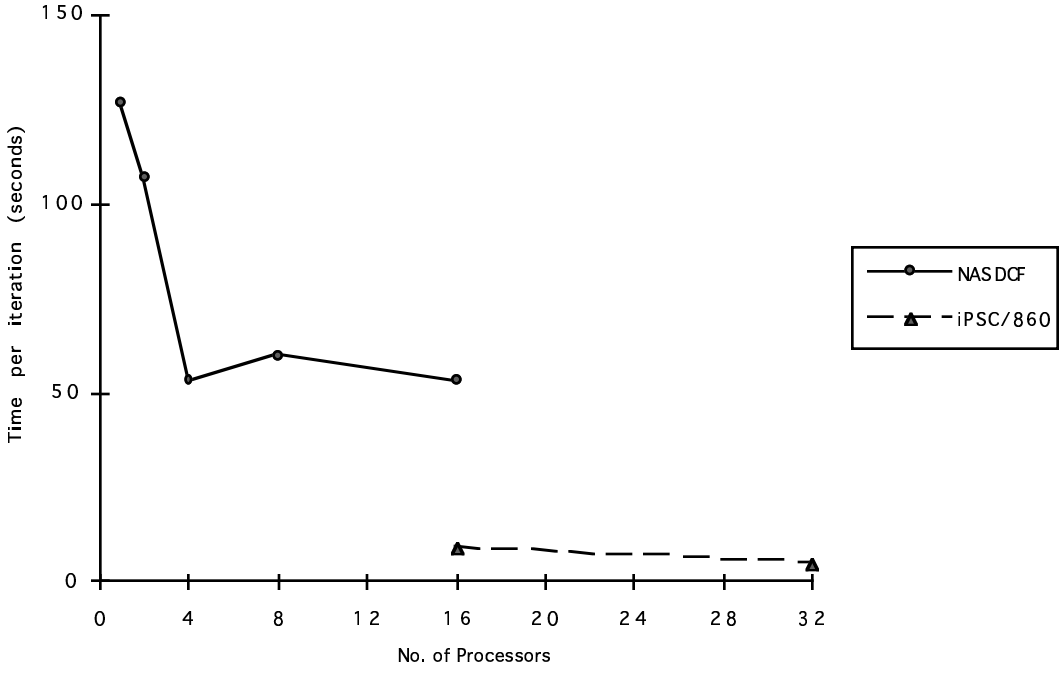


Figure 4. LU Comparison for the NAS DCF and iPSC/860

Two parallel implementations of the SP benchmark were considered. A transpose-based algorithm is employed using either one dimension or two dimension partitioning. This implementation requires sending a few long messages. (On four processors using 1D partitioning, a total of 64 messages with a total communication volume of about 32 megabytes was sent every time step.) The results are given in Table 10 and Figure 5. A

speedup of about 2.2 was achieved using 16 SGI processors. The results for 16 processors on the Intel iPSC/860 and the DCF show that the iPSC/860 is more than an order of magnitude faster than the cluster.

Computer System	Problem Size	Number of Processors	Network	Time (sec)	Ratio to Y-MP/1
Y-MP	64x64x64	1		471.5	1.00
Intel iPSC/860		64		667.3	0.71
RS/6000 model 560		8	FDDI	866.7	0.54
RS/6000 model 560		8	Allnode	1003.9	0.47
NAS DCF SGI R3000		8	Ethernet	6400.0	0.07

Table 10 - NAS Parallel Benchmark Results for SP

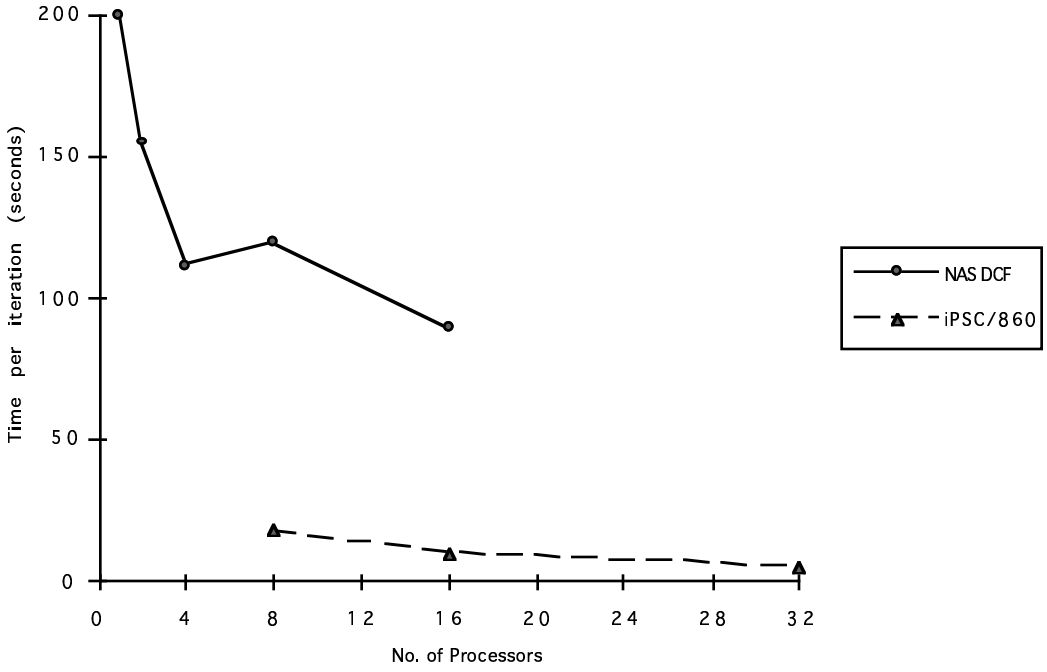


Figure 5. SP Comparison for the NAS DCF and iPSC/860

Another parallel implementation of SP is based on the pipelined Gaussian elimination algorithm using 1D, 2D, or 3D partitioning. This implementation requires sending many short messages to nearest neighbors. (On four processors using 2D partitioning, a total of 46160 messages with a total communication volume of about 4 MB was sent every time step.) The results of this implementation, not given here, show little speedup on the NAS DCF.

Two parallel implementations of the BT benchmark were also completed. A transpose-based algorithm, very similar to the SP implementation in communication requirements, was employed. The results of this implementation are given in Table 11 and Figure 6. Not

surprisingly since SP and BT are similar, these results are similar to the SP implementation.

Computer System	Problem Size	Number of Processors	Network	Time (sec)	Ratio to Y-MP/1
Y-MP	64x64x64	1		792.4	1.00
Intel iPSC/860		64		714.7	1.11
RS/6000 model 560		8	Allnode	1446	0.55
NAS DCF SGI R3000		8	Ethernet	14080	0.06

Table 11. NAS Parallel Benchmark Results for BT

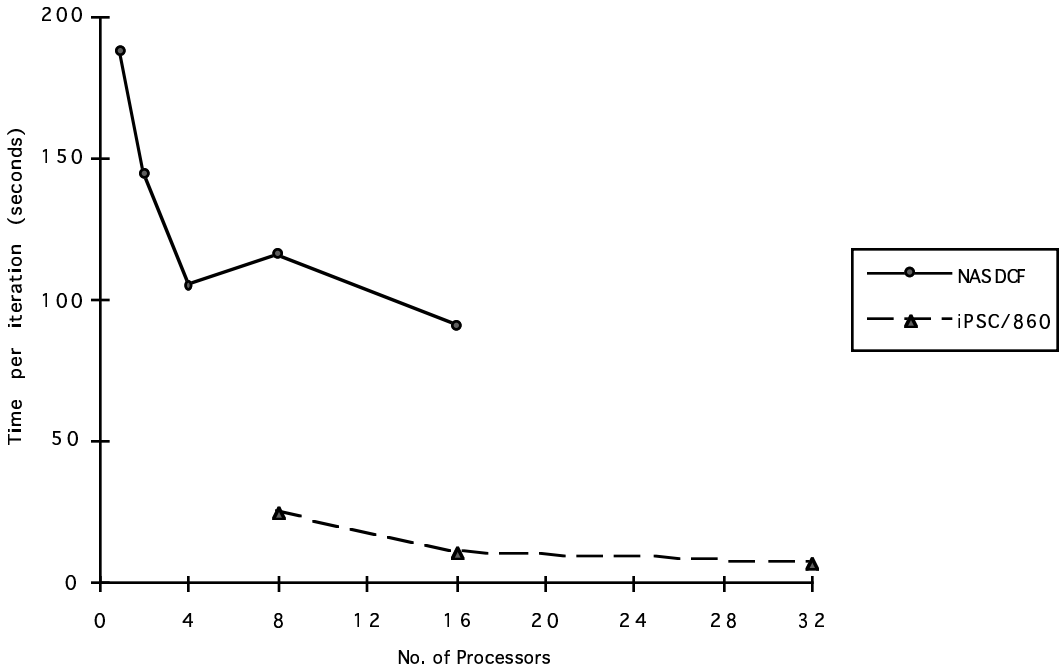


Figure 6 - BT Comparison for the NAS DCF and iPSC/860

Another parallel implementation of BT is based on the sub structured Gaussian elimination algorithm using 1D partitioning. This implementation requires sending few long messages. (On four processors, a total of 58 messages with a total communication volume of about 25 MB was sent every time step.) The algorithm used in BT suffers from higher arithmetic overhead. The results of this implementation, not given here, show no speedup on the NAS DCF.

The kernel EP was also implemented on the NAS cluster for comparison. This kernel has almost no communication and therefore is an embarrassingly parallel algorithm). The results are given in Table 12 and Figure 7. A speedup of 7.96 was achieved on eight SGI processors. For this benchmark, the Intel iPSC/860 processor is about 50 percent faster than the SGI processor.

Computer System	Problem Size	Number of Processors	Network	Time (sec)	Ratio to Y-MP/1
Y-MP	2**28	1		126.2	1.00
Intel iPSC/860		64		51.4	2.46
RS/6000 model 560		8	FDDI	137	0.18
SGI R3000		8	Ethernet	694.6	0.92

Table 12. NAS Parallel Benchmark Results for EP

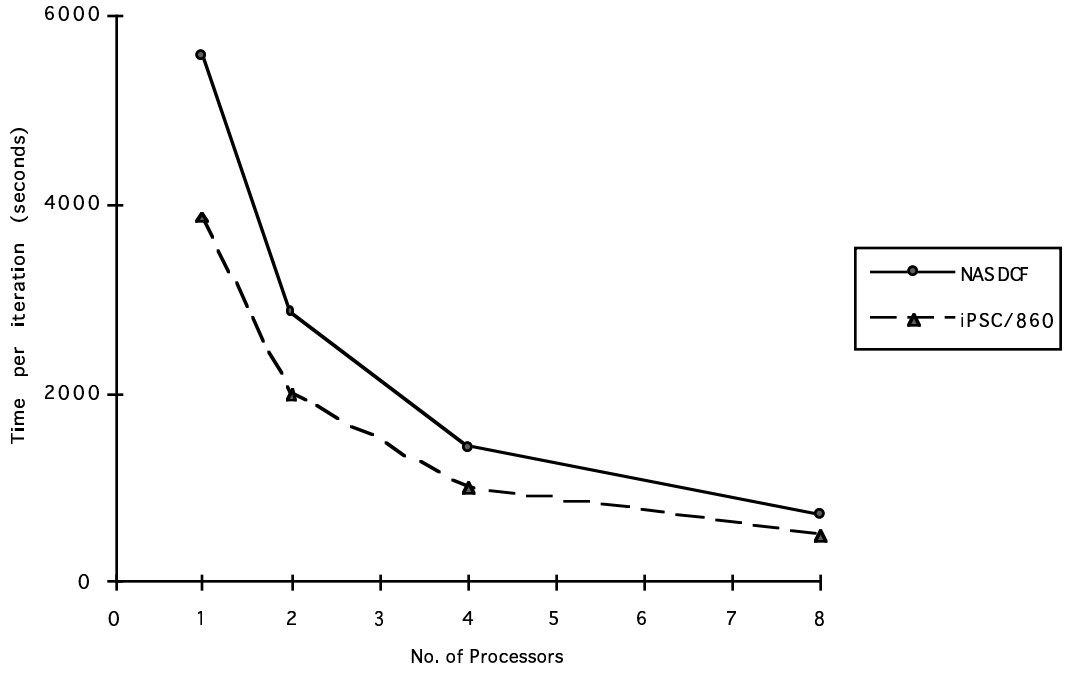


Figure 7 - EP Comparison for the NAS DCF and iPSC/860

This study shows that Ethernet may not be sufficient for CFD codes based on grid partitioning algorithms which preserve the implicitness of the numerical algorithm and which run on more than a handful of processors. A comparison between Ethernet and the iPSC/860 network shows that the latter out performs the former in the three main network aspects: topology, bandwidth, and latency. Ethernet is a broadcast bus while the iPSC/860 network is a hypercube. The sustained bandwidth for the iPSC/860 network is about 2.5 MB/sec per link while it is about 0.5 MB/sec for Ethernet. Latency for the iPSC/860 network is about 150 microseconds while it is about 1 to 3 milliseconds for Ethernet.

A performance comparison between the NAS DCF and the current generation of multiprocessor machines shows that the cluster lags in both the network and processor speed. The SGI processor for many NAS cluster machines achieves between 1.3 and 4.8 MFLOPS for the four benchmarks while the processors for many multiprocessor machines as well as current generation workstations can achieve performance an order of magnitude faster than the NAS DCF machines. Also, multiprocessor machines have faster and richer networks than Ethernet.

The NAS DCF, composed of systems a generation behind the current workstations[22], lags behind many dedicated clusters that have faster processors and networks. Many of these clusters have switches that can achieve over 10 MB/sec transfer rates simultaneously over multiple connections. Although some NAS machines are connected by FDDI, currently this is limited to high-speed processors and general support machines. These results indicate that it may not be useful to pursue NPB type problems on clusters similar to the NAS DCF, particularly if they consist of older generation workstations. It remains unknown how well these problems will do in clusters that incorporate the latest generation of systems and networks.

Figure 8 shows a summary of the performance of each of the environments relative to a single processor of a YMP. Figure 9 shows the same data, without the Intel and on a different scale for a clearer picture of different cluster configurations. The environments listed are:

1 Intel iPSC/860 - 128 Processors
2 Intel iPSC/860 - 64 Processors
3 YMP
4 Sparcs 1+ with Ethernet
5 RS/6000 - 560/320 with FDDI
6 RS/6000 - 560/Ethernet
7 RS/6000 - 560/Allnode
8 SGI R4000 with Ethernet
9 SGI R3000 with Ethernet
10 NAS DCF

[22]NAS is in the process of selecting its next generation of workstation architectures(s) at this time.

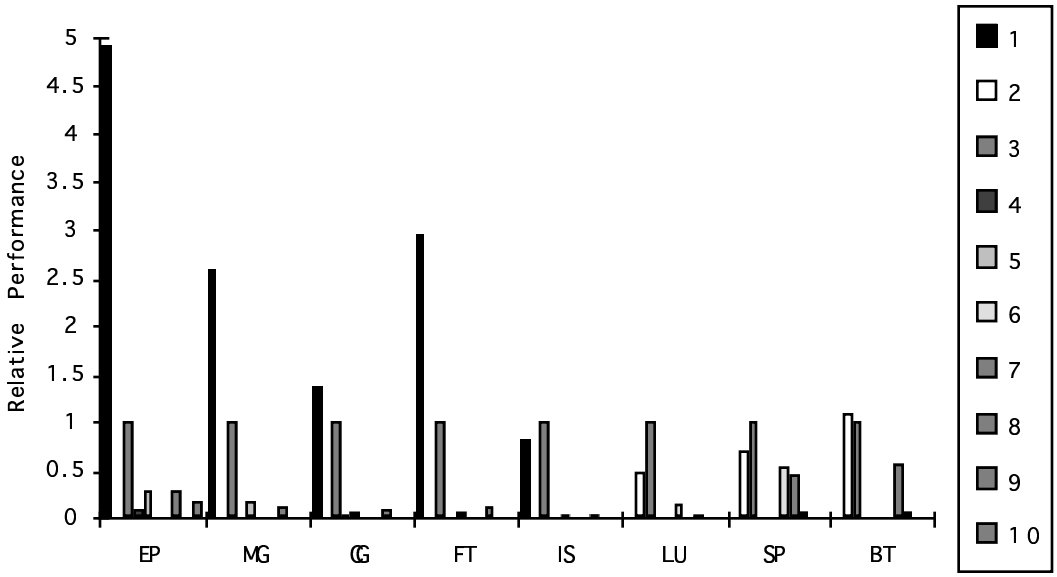


Figure 8. NPB Performance Relative to a CRI YMP/1

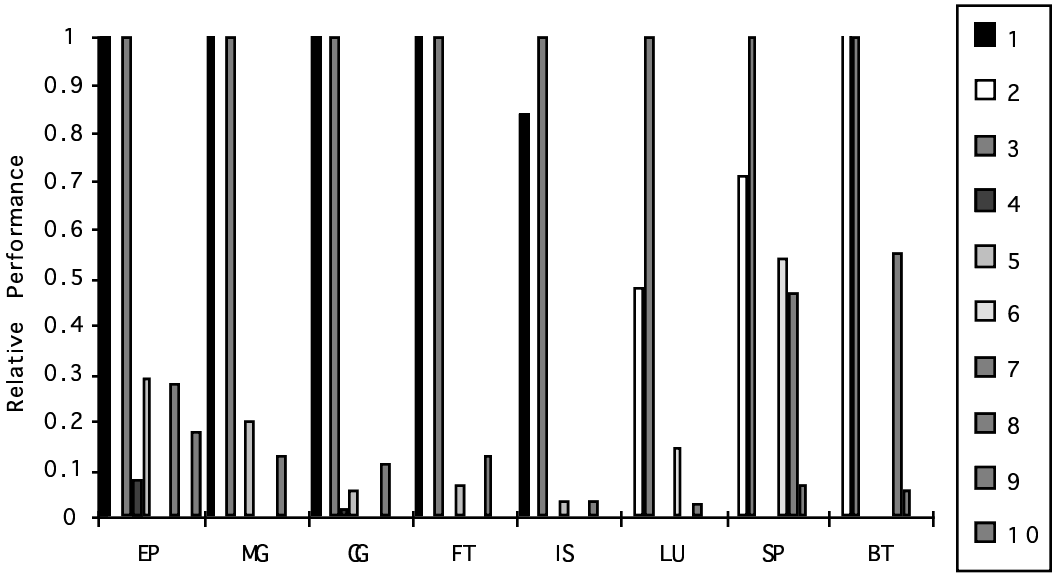


Figure 9. NPB Performance Relative to a CRI YMP/1 - Exploded Scale

4.2.1 Cost Comparisons

While there is NPB performance data for some cluster implementations reported, no cost comparison has been done because of the difficulties of pricing configurations. The DCT attempted to price all the configurations given above, as well as some reported in the latest NPB report. Table 13 summarizes this information as the Performance per $1 Million, using the same methods as the NPB reports. All results are for the Class A NPB problems.

Cluster prices were calculated by getting the most complete configuration information possible for the various clusters and using prices from several existing contracts available to NAS to procure the same equipment. For equipment no longer on existing contracts or sold by vendors, the last known price was used. In one case, where the amount of disk was not included in the configuration, the amount shipped with a standard configuration was used. Network infrastructure costs were not included, but special networking hardware such as interface cards were. Also special network hardware such as the Gigaswitch was included. The total cost was then prorated on a per processor basis, and allocated by how many processors were used for the exact test. The NASA Lewis cluster cost was provided by staff from NASA Lewis. Costs for non cluster systems are taken from the most recent version of the NPB report. Readers are referred to this report for a complete description of the performance and costs. The table is sorted by date of system introduction.

Computer System	Date	Cost per Processor			Benchmarks					
			BT	SP	LU	IS	FT	CG	MG	EP
IBM RS/6000 model 590-1	1994	$53,114	11.9	8.94	9.73	11.9	8.88	42.0	7.73	34.9
IBM SP-1/8	1994	$41,563	4.46	3.21	3.44	4.46	1.98	1.68	1.92	27.6
IBM SP-1/64	1994	$41,563	3.13	1.77	1.98	3.13	1.67	0.95	0.90	27.6
TMC CM-5E/32	1994	$31,250	5.43	2.79	2.19	5.43	3.89	1.06	5.70	11.0
TMC CM-5E/128	1994	$31,250	4.13	1.93	1.28	4.13	2.48	0.96	4.27	10.5
SGI R4000	1992	$14,074					1.12	0.87	1.17	2.51
LeRC LACE - IBM 560	1992	$46,875		1.45				0.26	0.35	1.23
LeRC LACE - IBM 560	1992	$46,875	1.46	1.25	0.40	1.46		0.26	0.35	
PVM RS/6000/560	1992	$38,856						0.94		
IBM RS/6000 model 560-4	1992	$38,856						0.94	0.78	
IBM RS/6000 model 560-8	1992	$38,856					0.22		0.65	0.92
CRI C-90/1	1991	$1,915,625	1.16	1.33	1.10	1.80	1.46	1.75	1.42	8.23
CRI C-90/4	1991	$1,915,625	1.08	1.24	0.99	1.08	1.46	1.62	1.32	2.74
CRI C-90/16	1991	$1,915,625	0.91	1.18	0.62	0.91	1.03	1.14	0.76	2.57
Intel Paragon/64	1991	$27,422		1.05	0.52	1.92	1.80	1.66	1.51	6.88
Intel Paragon/128	1991	$27,422	1.75	0.97	0.41	1.75	1.67	1.03	1.41	6.86
PVM RS/6000/550	1991	$25,000						0.59	0.76	1.42
Intel iPSC/860/128	1990	$14,609	1.02	0.56	0.40	1.02	1.59	0.91	1.38	2.63
Sparcs 1+	1990	$9,097								0.52
PVM Sparcs	1990	$9,097								0.52
IBM RS/6000 model 320	1990	$8,616						0.55	11.8	2.76
CRI YMP/1	1988	$2,750,000	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
CRI YMP/8	1988	$2,750,000	0.32	0.33	0.31	0.32	0.31	0.23	0.34	0.36
PVM SGI 4D/25	1988	$14,074								0.88
NAS DCF - SGI R3000	1988	$49,893	0.14	0.18	0.07	0.14				0.46

Table 9. Approximate Sustained Performance Per Dollar of Selected Systems

Several observations can be made from this data. First, some workstation clusters, particularly the IBM 560 clusters, compared well to CRI C90 and Y-MPs. They also compare well with MPP systems introduced in similar time periods. On the surface, some clusters, particularly loosely clustered ones such as the NAS DCF, do not compare as well with supercomputer or MPP systems. For the three pseudo-application NPBs (SP, LU and BT), the NAS DCF average is .13 while the CRI Y-MP/1 is .36. However, all the systems in the NAS DCF were acquired for an entirely different purpose than compute capability - scientific visualization. The NAS DCF workstations do an excellent, cost effective job for the primary function of visualization. Additionally, there are significant idle cycles for these systems when their users are not active. The absolute performance

comparison between loose clusters and supercomputers or MPPs must be adjusted to take such considerations into account. It may be viewed that the only cost to consider is whatever equipment, software and support are needed to made such workstation usable for compute intensive work.

Unfortunately, no results exist for clusters with processor technology introduced after 1992. The only workstation system with results is a single processor (non-networked) IBM 590, that compares very favorably with every other system in the table. However, with only a single processor result for one workstation introduced after 1992, one can not conclude how cost effective workstations clusters of current generation systems are in solving large problems that do not fit on a single system. The single system results are encouraging, but clearly, workstation vendors must be encouraged to provide results for clusters composed of the latest generation of workstations.

4.2.2 DCF Network Usage for the NPB

The NAS DCF network is essentially a collection of Ethernet subnets, concentrated on an FDDI backbone. When a job runs in the DCF, all files are accessed via NFS so there is no permanent use of disk resources when the primary user is working on a machine in the cluster. (The DCT realizes there is a performance penalty for this.)

One of the major issues raised by the use of the DCF was the interaction of network usage with the overall performance of the applications. Some work was done to investigate these interactions, although as noted in some of the earlier sections, much work remains to fully explore them. Specifically, improved ways to measure the performance of a program relative to the activity on the network, both the activity generated by the program and other activity on the network.

Running a number of small jobs across four machines on a single Ethernet subnet can bring a single segment's utilization well past the 80 percent point. These jobs utilize the entire effective bandwidth of an Ethernet, so that higher speed networks between the machines could improve application performance. It is unclear, however, exactly how much improvement would result. The DCT plans to investigate jobs running across FDDI and ULTRAnet, comparing job run times with Ethernet.

Another experiment involved running a short three minute job which transferred about 32 MB of data between four machines. This is typical of some of the NPB codes reported earlier. On a single Ethernet segment, the percent utilization increased from under 20 percent to close to 100 percent. The job certainly dominated the subnet while it ran since subnets with more than 40 percent experience significant collisions, causing slowdowns on the network. Thus, it is expected such jobs would affect other work on systems on the network segment for the duration of the time they execute.

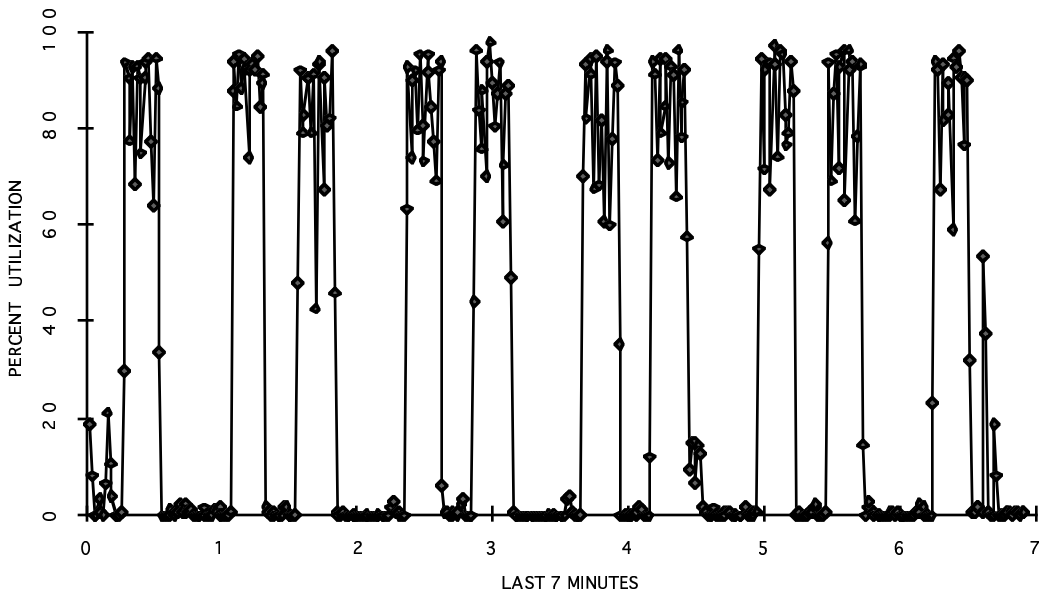


Figure 9. Example of Ethernet Utilization

Running the same test on four machines using an FDDI ring showed an increase in percent utilization from under 8 percent to between 20 percent and 25 percent. As the effective bandwidth of an FDDI ring is approximately 10 times that of an Ethernet, so the bottleneck may have moved from the network to protocol processing.

These network experiments show that if programs with significant message-passing and I/O requirement run on Ethernet-based systems, they are likely to impact other systems on the subnet, even if the program is running on an idle workstation. Thus, running DQS/PVM jobs during the day, even on idle systems can have a major impact on other workers.

4.3 Porting Cray Codes to Workstations

Another area the DCT investigated was the difficulty and benefits of moving some portion of work from the supercomputers to an environment similar to the DCF. This is because a large number of problems currently running on the supercomputers may run effectively on individual workstations. For examples, even on the NAS Cray systems, configured specifically for large memory (greater than 2 gigabyte) problems, more than 50 percent of the CPU capacity is used by batch jobs less than 96 MB.

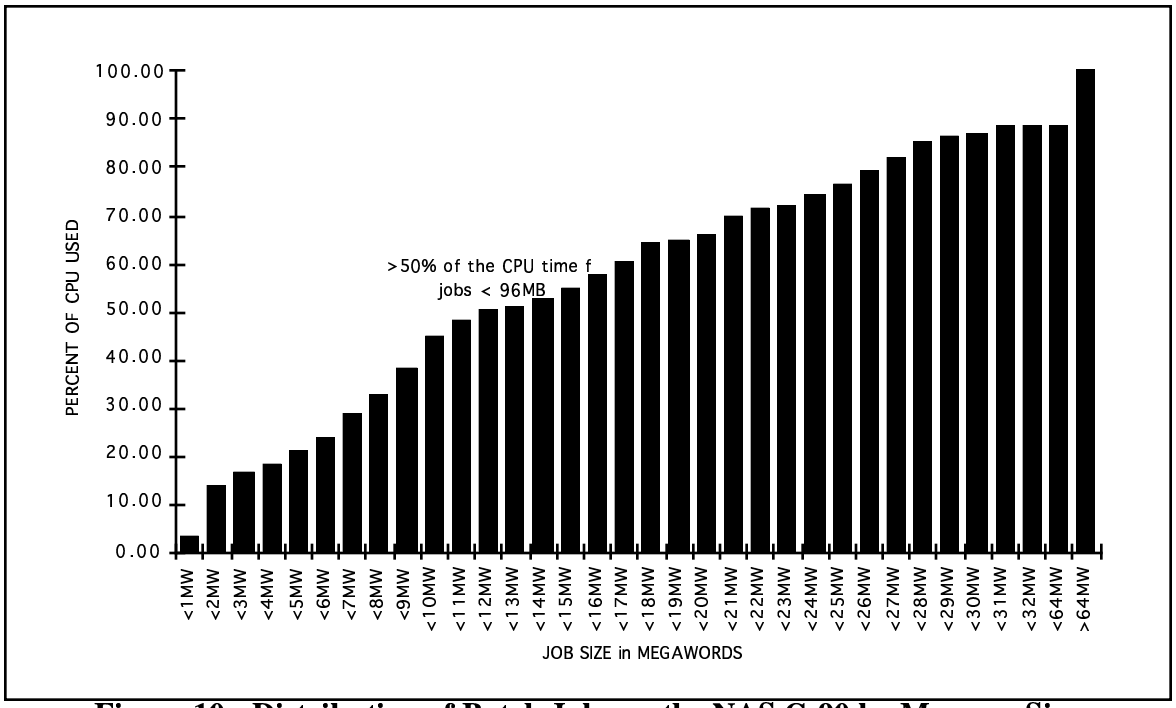


Figure 10 - Distribution of Batch Jobs on the NAS C-90 by Memory Size

In order to investigate this, a suite of 32 standard-conforming FORTRAN 77 codes was ported from the CRI C90 to SGI workstations. This section describes the problems encountered and then remarks on potential problems with nonstandard conforming code to illustrate of the issues involved in moving highly vectorized codes from Crays to workstations. The major point of this exercise was to assess the possibility of moving some portion of work from the supercomputers to an environment similar to the DCF. To that end, these highly vectorized codes were compiled and run as single-threaded executables on the workstations. These codes were not run in parallel.

The suite comprised a variety of codes, including kernels, pseudo-application codes, and user application codes (such as Navier-Stokes and Euler solvers.) Slightly over half of the codes in the suite were too big to run on any NAS workstation; however, all were compiled locally to investigate their portability.

Most problems encountered in compiling these codes were related to inadequate software support on workstations for high-precision floating point and integer arithmetic. About one-third of the codes contained specific forms of the FORTRAN 77 intrinsic functions. These codes failed to compile when the *-r8* (64-bit floating point) compiler option was invoked. These problems (usually REAL vs. DOUBLE PRECISION type mismatches) were typically solved by replacing the specific forms with the generic intrinsics (e.g. MIN for AMIN1, EXP for CEXP). AIMAG was the only specific intrinsic function without an analogous generic function in these codes. It was changed to DIMAG, a nonstandard DOUBLE PRECISION SGI intrinsic.

SGI *f77* provides no support for integers larger than 32 bits. Several codes relying on such large integers produced incorrect output until this problem was diagnosed. The problem was addressed by replacing the INTEGER calculations with REAL calculations.

Data alignment problems are related to the 64-bit integer problem. COMMON blocks with 32-bit INTEGERs and 64-bit REALs may be misaligned due to FORTRAN storage

association requirements. All codes with misaligned data ran successfully. However, several of these codes failed to compile when *-mips2* (R4000) optimization was invoked. This was overcome by using the undocumented *-align64* compiler option.

Finally, several codes failed to compile because *f77* tried to put large arrays on the calling stack. This problem was solved by invoking the *-static* option.

All of the codes used in this work so far were clean benchmark codes. Additional problems occur in codes from the production workload. Such codes combine the Cray file positioning commands GETPOS/SETPOS with BUFFER IN/BUFFER OUT statements. This combination allows high-speed random access I/O. Workstation versions of these codes must be modified to achieve random access I/O through the more restrictive READ/WRITE commands on direct access files. Many codes rely on the CRI FORTRAN compiler to initialize COMMON and set local variables to zero. In porting a code to a workstation, users may need to investigate arcane compiler options that provide the same effect, or preferably modify the code to initialize all data.

The NAS workstations provided reasonable performance, albeit much less than available on a CRI Y-MP as shown in Table 13. The suite performance on the R4000 was better with 64-bit floating point than with 32-bit floating point, due to the 64-bit hardware support. This was true to a less extent on the R3000, and was not the case on the R2000. There is considerably more variation in performance on a workstation than on a CRI Y-MP. While some patterns were discernible (e.g. FFT kernels tended to be slower), we did not find any correlation between application type and performance.

Vendor	**Processors**	**Performance Range (MFLOPS)**
CRI	Y-MP	100.0□-□170.0
SGI	R4000	2.4 □-□16.6
SGI	R3000	1.8□ - □□5.7
SGI	R2000	1.4□ - □□2.9

Table 13. Processor Performance Ranges on HSP Benchmarks Codes

Some of the slower codes might see performance gains if they were optimized to use the cache more effectively. Such optimizations include padding arrays to reduce the number of cache misses and reblocking array operations to fit into the cache.

The above study was performed on workstations acquired primarily for their good graphics performance. Other vendors, such as Hewlett-Packard and IBM, offer workstations with superior floating point performance on industry benchmarks such as SPECfp92. Further, workstations with memory in excess of 512 MB (64 MW) are now commercially available from several vendors. It is clearly feasible to assemble workstation clusters capable of processing traditional large-memory, number-crunching jobs in a productive and cost-effective manner.

4.4 Summary of Other Users

Since February 1993, NAS staff primarily relied on the input from four users to assist with the identification and resolution of major system management issues associated with the cluster. In order to gain more experience in this type of computing environment it was decided to expand the user base in mid-September 1993. An announcement was sent out to the NAS scientific user community offering the prototypical (DCF) for limited use. In October the number of DCF users grew to its current complement of 12. The additional users were selected based on:

- willingness to do actual science applications in a rudimentary environment and to provide feedback on their experiences working with that environment
- applications that ran on other massively parallel systems in order to draw performance and service comparisons
- applications that required short runs or small memory on the CRI C-90 or Y-MPs to evaluate the issues of off-loading these types of supercomputer jobs onto workstations. It was not necessary for these to be parallel applications
- applications that ran on tightly clustered workstations in order to compare performance and service with a loosely coupled environment

As mentioned above, the DCF offers two types of queuing systems (Condor and DQS) to our small community of users. Jobs submitted to DQS are parallel programs using PVM at night and jobs running under Condor are sequential programs utilizing idle cycles. As the queuing systems form two groups for different programming methods, so DCF users form into two groups. The following summarizes the experiences of these two groups of users utilizing the DCF cluster at NAS.

4.4.1 DQS/PVM Users

Since the DCF was made available recently, most of the activity of this group has been devoted to ramping up. Those users who are now ready to execute their programs have primarily targeted their comments to administrative details with some observations about using DQS. These comments and observations raise issues which need to be resolved in order to make this batch system and the cluster easier to use and more efficient.

DQS runs at night with jobs being submitted during the day, prior to 10. Users look at the results the next day. This method causes delays in development of PVM programs and in actually executing the program. It places the burden on the users to discern whether a problem is due to an error in their DQS command file or due to such things as a failure of an individual workstation. Further prone to delays since tracking down such problems occur since jobs only run at night. This issue can been addressed by making a policy change having additional DQS queues available during the day expressly for the purpose of debugging. Off course, this can impact the primary workstation user since DQS does not monitor all the workstations a job uses, just the workstation that controls the PVM job.

To avoid disrupting a host workstation as much as possible, NFS was used for all file accesses with job I/O routed through NFS to the DCF user's home workstation. This approach limits DCF user's ability to exploit an application's potential for doing embarrassingly parallel I/O, but insures the primary user of the workstation has all resources available during the day. By requiring I/O to route to the user's home workstation, a bottleneck was created whenever data was written out to solution and checkpoint files or initial grids were read in. The possibility of making a change and

giving users access to /tmp directories on individual workstations is being considered as are other shared file system implementations.

DQS has been deficient in providing information to the user. The mail notification feature does not work well. Failures due to the logistics of setting up a virtual machine, such as the inability to add a host, are not reported. If a portion of the job aborts (other than the host process), the job hangs and DQS takes no action.

The DQS queues are turned off at 5 am, and DQS suspends existing jobs on the respective host machines, but daemons related to the suspended job are left running. As a result, jobs will continue to use up workstation resources and could potentially interfere with the processes of the primary users.

4.4.2 Condor Users

User experience with Condor has been limited. Originally, the Condor cluster consisted of six SGIs making idle cycles available to two users. Currently, there is only one user. Initial problems with checkpointing jobs on the SGIs, Condor's inability to adequately detect interactivity, and not allowing the use of shell scripts have curtailed its anticipated potential.

One new individual discovered a basic stumbling block that could not be surmounted. While Condor has a command to identify an executable file for submission, normal UNIX shell scripts containing executable commands cannot be submitted. Therefore, it was not possible for this user to submit a multi-staged job using a complicated set of shell scripts for program control without extensive modification to program code and multiple manual submissions.

This same user also found that Condor was not as nonintrusive to the primary user of a workstation as advertised, in that only shell keyboard activity caused a running job to go dormant. Use of the mouse or interactive keyboard activity with a graphics or non-CPU application program like FAST[23] did not prevent Condor from attempting to swap jobs back into the system. In addition, when a job is suspended, some of the system main memory continues to be used, causing potential swapping problems for memory and/or time intensive programs.

On the positive side, CPU intensive jobs that required several days to a week to complete have run successfully. When these jobs ran uninterrupted, the remaining user found Condor to be very useful. This user overcame the initial checkpoint problem by not allowing Condor to restart jobs after being interrupted and manually resubmitting them. In this way several hours or days of computation with an occasional job running to completion were salvaged.

---

[23] Pamela P. Walatka, Jean Clacas, R. Kevin McCabe, and Todd Plessle, *FAST User Guide*, NAS Technical Reprot, RND-91-012, June 1992.

5.0 DCF Usage Analysis

During the time that the DCF environment was open to several users, the DCT investigated usage for both the workstations and networks. This section reports some of the findings. Only existing tools were used to record information. Accounting and usage reporting clearly have to be improved as part of providing a more general-purpose production environment.

5.1 Measuring System Usage

To provide usage information, Condor maintains a separate history log under the home directory of the Condor account on each of the machines in the pool. A command, *condor_summary*, is provided to combine and summarize those individual history files. Unfortunately, the distributed nature of the history files has proven to be somewhat difficult to work with. Careful attention must be paid to ensure the presence of the accounting files on each system. Systems added to the pool, and later removed may still contain valid accounting records. It is also not possible to summarize the usage records until all systems are available and accessible. As the number of systems grows, formal procedures to preserve accounting files will become necessary as the opportunities for problems will increase.

DQS, unlike Condor, maintains usage information in a single file on its master system. DQS provides a command, *dqacct*, to display or summarize usage information.

PVM does not provide any usage information. Instead, it relies on its hosts' underlying accounting system to record process accounting data. Unfortunately, process accounting data does not provide sufficient data to identify and group individual PVM processes, possibly spread across multiple machines, into a single PVM job. The situation is further complicated since PVM jobs can either be started interactively, or from within a DQS job, so that PVM jobs can also be run on many machines not even considered part of the DCF. Because of these problems, there was no attempt to record or report PVM usage information for the prototype other than the fact it is a superset of the DQS statistics.

An example of the usage information is provided in Figure 11.

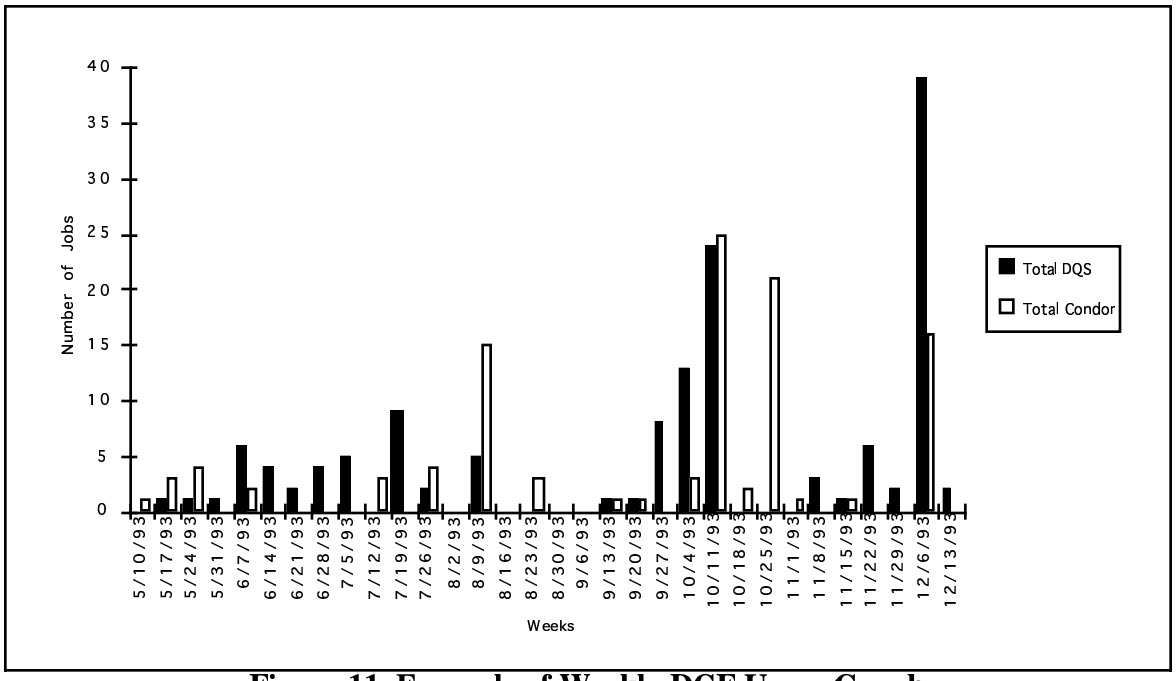


Figure 11. Example of Weekly DCF Usage Graph

6.0 Summation of the DCF Experience

Useful work in computational aerosciences can be performed on the DCF as currently configured. The total computational power and memory of the cluster is considerable. Further, the aggregate compute capacity (based on LINPACK estimates) of all 320+ NAS workstations (WKS-II technology) is 2.3 GFLOPS, which is approximately 30% of the entire CRI C90. Optimistically, assuming the 40 percent efficiency achieved by OVERFLOW-PVM can be achieved by other applications, assembling all the NAS workstation resources yields 920 sustained MFLOPS . Since workstations are unused 75 percent of the time, one may conclude there are 700 MFLOPS available, which is potentially equal to 1.8 CRI C90 processors. Small serial and coarse-grain parallel jobs are well suited to this environment and will yeild closer to 100% efficiency when running. Dozens of candidate jobs are run every day on the NAS CRI C90.

The DCF can also serve to off-load a portion of the jobs currently running on the NAS CRI C90. Small jobs (less than 96 MB of memory) make up a significant fraction of total usage (approximately 50 percent). If NQS queue wait time is taken into account, very reasonable throughput can be expected from the newer generation workstation processors.

Although the DCF has shown the potential to off-load a portion of HSP jobs and to run coarse-grain parallel jobs, it is not currently in position to replace the HSP or the MPP systems. The maximum number of users on the system is limited by several factors. First, the systems that make up the current DCF were originally selected for their graphics capabilities and not their CPU power. As they are not specifically computational engines, there is inadequate double precision hardware and compiler support, particularly in the earlier generation machines (R2000, R3000). Finally, a large number of distributed parallel jobs will quickly overwhelm Ethernet, affecting other local users.

As stated earlier, the types of parallel jobs that may run efficiently on the loosely coupled DCF is also limited. Programs written for MPP systems that require high bandwidth and/or low latency will perform poorly or not at all. Programs that require parallel I/O will be excluded.

The size of jobs transferred from the C90 to the DCF under Condor will be fairly limited, as most machines comprising the DCF have less than 64 MB of memory. This memory restriction will have a lesser impact on parallel jobs. There is also a profusion of older generation processors which do not have the capacity for providing adequate throughput for anything but the smallest serial jobs.

7.0 The Role of Cluster Computing in Industry

Economic pressures make cluster computing very attractive for the aerospace industry. Tightly clustered systems give cost-effective computing for specific jobs. There are several examples of successful, tightly coupled workstations in production at such locations as NASA Lewis and Lawrence Livermore.

Loosely clustered systems have been successfully demonstrated at McDonnell Douglas, where a system of over 150 HP workstations is in use[24]. Recent results for an aerodynamic simulation having 57 grids and 4.7 million grid points indicate that a cluster of 15 mixed HP 715 and 735 systems on Ethernet can outperform (wall-clock time to solution) a single dedicated CRI C90 CPU using McDonnell Douglas' production flow solver. Their processor allocation scheme is simple, similar to the original one used in the DCF, but five to six large jobs run every night in a non-production environment. This system has provided computational resources to an aircraft program under very tight budgetary and time constraints. As more users are added, McDonnell Douglas will need more sophisticated scheduling software, but their success is extremely encouraging.

8.0 Expectations for Future Clustered Systems

Many limitations of the current DCF have either already been dealt with elsewhere in this paper, or are well on their way to being overcome. At the same time, there are many areas in which NAS can make a significant contribution to this style of computing. The keys to an effective loosely-coupled cluster are described in the sections below.

8.1 Processor Speed

Workstation processor development is continuing at a dramatic pace, outstripping that of conventional supercomputers and MPPs. The level of competition in this market segment is significant. Near order-of-magnitude improvements in performance have been common. While this improvement cannot continue indefinitely, the large volume of sales in this market continue to bring in research and development funds. The relative simplicity of a workstation, as compared to an MPP, allows vendors to bring their latest technology to market in a workstation months or even years before it would appear in an MPP.

By measures such as the NAS Parallel Benchmarks run on single processor workstations, the current generation of workstation processors can provide a cost-effective

[24]Report of the NAS Workshop on Distributed Computing for Aeroscience Applications, October 18-20, 1993 and separate discussions with McDonnell Douglas staff.

computational platform to greatly reduce our reliance on the CRI C90 for smaller, shorter-running jobs. For instance, IBM's new release, the RS6000 590 system, has demonstrated at least 10 percent of the performance of a single C-90 processor on the NAS Parallel Benchmarks at a small fraction of the cost. Newer machines are not nearly as memory limited as older generations with capacities of 0.5 GB, and up to 4 GB is quite common, in the offerings of most vendors. New fast SCSI devices have increased significantly both the volume and the rate of disk I/O over previous generations of workstations. Shared-memory parallel machines like the SGI Challenge series (up to 18 MIPS TFP processors) also offer great flexibility in both parallel and serial environments. Like the current SPS machines, multiple processes may be run at once, or a serial job (or a single element of a distributed parallel job) may be parallelized in the shared memory environment by tools like SGI's PFA.

8.2 System Software

The high level of competition in the workstation market keeps single-system software offerings from the vendors complete and of high quality. Unlike the MPP systems, the compilers coming from the workstation vendors are relatively mature, and there are a variety of utilities and software packages available. Regardless, moving supercomputer-level applications into this environment requires improved compiler and language support for reliability and performance. Bugs need to be resolved and language support for distributed computing must be improved.

In the near future, we are likely to see many tools designed for clustered systems. Program debugging, which is very difficult on distributed systems, must be a priority. Performance analysis and process timing tools should follow. The Message-passing Interface (MPI) has already been implemented by IBM on its SP-1 system, and other vendors, both workstation and MPP, are expected to follow suit. This could make the possibility of a seamless parallel environment a reality. Development continues on communication products like PVM and P4, and a cluster-oriented version of High Performance FORTRAN should be available soon.

Commitment to cluster computing varies among the primary hardware vendors. IBM is supporting this style of computing with a number of tools and methods. HP has features planned, some of which have been implemented, to support clusters. SGI, on the other hand does not provide support for cluster computing, and Sun is relying mostly on public domain tools. Many new system administration tools must be developed in order to maintain and operate a large cluster in a production environment. NAS experience in working with multiple vendors and in managing large numbers of workstations will be invaluable in this development effort.

There is a significant need for a single integrated job-control system that supports parallel and serial jobs and is integrated with PVM and other parallel language constructs. This system must support job pre-emption for interactive use, adequate job checkpointing, restart and suspension, shell scripting and adequate job limits. To minimize user confusion, it should either incorporate the functionality of NQS, or should be entirely separate with distinct commands.

8.3 Application Software

Numerical algorithms specifically designed for distributed systems are in development and are showing promise. In addition to the coarse-grain methods like OVERFLOW-PVM, medium-grain matrix solvers have shown reasonable performance over a small number of processors. Medium-grain methods can be used to increase the usable number of processors in an otherwise coarse-grain parallel application.

Multi disciplinary and optimization problems contain an extra level of parallelism. Either simulations of several slightly different configurations, or simulations of several physical processes are required and may be run in parallel. Interest in this type of work is growing. Some level of education will be required before potential cluster users can run efficiently. The principles of vectorization on Cray machines are widely known in the aerospace industry, but strategies for cache management, important on RISC-based machines, are virtually unknown. The strengths and limitations of a distributed resource must be investigated in order to make full use of their potential.

8.4 Distributed Parallel I/O

While some research into distributed parallel I/O has been done, little if any has been applied particularly to the cluster environment. Systems far beyond the current NFS-based cross-mounting will be required for the efficient operation of some types of parallel jobs. This may be an area in which NAS can make a significant contribution.

8.5 Networks

Most distributed computing work today is done on machines connected by Ethernet. User processes running on different machines communicate with each other by means of a high-level message-passing library (e.g. PVM, p4). The high level library, in turn, usually relies on the vendor-supplied TCP/IP software to reach the network. Improved communications performance can (and should) be obtained by improvements to all three links in this chain. It is important to realize that any of these links may be a bottleneck. If a system's weak point is the Ethernet, an upgrade to FDDI may expose the message-passing library or network protocol as the new bottleneck.

8.6 Physical Connection/Protocol

Ethernet is the slowest commonly available network. The theoretical maximum throughput is 10 megabits per second (Mb/sec). Typical Ethernets sustain 5 Mb/sec. Several attractive alternatives are:

FDDI: This protocol runs over a fiber network. The theoretical maximum bandwidth is 100 Mb/sec; sustained bandwidth of greater than 25 Mb/sec may be expected in a good implementation.

HiPPi: This protocol uses parallelism to achieve a high data transmission rate. The peak bandwidth is 800 Mb/sec; at least 320 Mb/sec can be sustained.

ATM: This is a cell-based protocol which is capable of attaining very high bandwidth. It can provide a bandwidth of 45 Mb/sec today and has the potential to upgrade to much higher speeds.

Network bandwidth may also be increased by adding switches, such as the Digital equipment Gigaswitch, or the IBM Allnode switch. Using such a switch, SCRI[25] has measured user-level latency to be 30-215 microseconds (ms), as opposed to 2-5 ms when using Ethernet.

8.7 Network Protocols

Protocols such as TCP/IP are used to communicate across the network. Most current implementations of TCP/IP are slow. These protocol have difficulty effectively using the bandwidth of high-speed networks, and introduce high latencies. Better message-passing performance might be achieved by either improving the implementation of TCP/IP or by relying on light-weight protocols which incur less overhead[26]. A light-weight protocol would, for example, need less fault tolerance than a general protocol such as TCP/IP.

8.8 Message-Passing Libraries

Message-passing libraries are used by the programmer to implement message passing in scientific codes. They introduce additional overhead beyond that caused by the network protocols. As with the protocols, more efficient implementations of existing libraries (such as PVM) may improve performance. New libraries might be designed to reduce overhead. It may be possible to gain performance by writing message-passing libraries that combine the functionality of the current libraries and network protocols. Libraries should also take advantage of shared memory for codes that use shared memory multiprocessors.

9.0 Summary

Cluster computing is a legitimate and viable computing environment for compute-intensive work. The computing power and affordable memory of the current generation of workstations makes it possible to do many of the tasks currently running on the supercomputers.

Economic pressures on the aerospace and other industries have generated a great deal of interest in transferring computational load from traditional supercomputers to clusters of existing workstations during when primary users are not active. Research shows that coarse parallelism is quite common in aerosciences applications. To meet the industry's needs, distributed computing research is starting in other organizations and will need capabilities similar to the DCF to continue.

Scheduling and queuing software and procedures for parallel and cycle-recovery jobs is now in its early stages. DQS and Condor are, for the most part, effective, but neither is currently capable of performing all of the functions necessary for the trouble-free use of a DCF. The DCT is currently in contact with both development groups, and work is continuing on both systems, as well as others.

Unfortunately, not all primary users are immediately willing to allow their workstations to participate in an open environment, particularly during prime-time. Tools that address the issues of responsiveness to the primary users are basic and not supported by most vendors. It must be demonstrated that the activities of the cluster will minimally impact

---

[25] ref. D. Duke in the NAS Workshop on Distributed Computing for Aeroscience Applications, October 18-20, 1993.

[26] Culler, David, Thomas Anderson, and David Patterson, *NOW: Design and Implementation of a Distributed Supercomputer as a Cost Effective Extension to a Network of Workstations*, University of Cailifornia at Berkeley Project Proposal, October 1993.

the regular use of the machine. Fears of loss of control and security must be abated through education as well as software. Many users will object simply because they may see no direct benefit in providing their workstation resources for others to use.

Using an existing base of workstations for compute intensive work will not demonstrate being cost effective by most performance benchmarks, particularly if these systems are loosely clustered using basic networking technology such as Ethernet. However, such systems can provide valuable compute cycles for reasonably little extra cost. Since organizations have already invested in the existing base, the cost effectiveness tradeoff is based on the added support costs and additional equipment costs, not on the original costs of the systems.

The effectiveness of workstation cycle-recovery may be enhanced by the administrative control of the environment. The effective administration of a distributed environment is bound to be more complex than that of a centralized one. In tightly or loosely coupled workstation clusters, knowledge about job mix and machine resources is essential. Cycle-recovery will be maximized if a site carefully matches a job's resource requirements to the available workstations-based on memory, processor speed, swap space, network connectivity and available disk space.

In the final analysis this work has shown that workstation cluster computing is able to efficiently solve major CFD problems. Continued work is needed to make such systems reliable, efficient, and easier to use. Work is also needed to determine whether clusters can effectively run the full range of algorithms involved with aeroscience problems. Further work is needed to determine whether cluster computing is the best system architecture for aeroscience applications.