NAS Experiences with a Prototype Cluster of Workstations

K. Castagnera,[1] D. Cheng,[2] R. Fatoohi,[2] E. Hook,[2] B. Kramer,[1] C. Manning,[2] J. Musch,[1] C. Niggley,[2] W. Saphir,[2] D. Sheppard,[3] M. Smith,[4] I. Stockdale,[2] S. Welch,[3] R. Williams,[1] D. Yip[1]

Submitted by William T.C. Kramer, Chief, NAS Computational Services Branch, NAS Systems Division, NASA Ames Research Center, Moffett Field, CA 94035

Abstract

This paper discusses the year-long activity at NAS to implement a large, loose cluster of workstations from the existing Silicon Graphics, Inc. (SGI) pool of systems. Issues related to establishing a loosely coupled cluster of workstations are presented. Included are steps needed to resolve system management issues intended to provided reasonable cycle recovery from these systems without disrupting the primary system users. 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. This paper summarizes the activities related to the prototype cluster and identifies areas that need improvement, development, and research in order to make workstation clusters a viable computing environment for solving aeroscience problems.

1: Introduction

The Numerical Aerodynamic Simulation (NAS) Systems Division at NASA Ames Research Center is one of the most advanced supercomputing environments nationwide. It provides access for a national client base of scientists working primarily in the field of computational fluid dynamics and related disciplines via a NAS-designed high-speed national network called AERONet.

NAS currently supports two CRAY C90 systems, a Thinking Machines Corp. CM-5, an IBM SP2, an Intel iPSC/860, an Intel Paragon, three Convex 3820s, a Convex 3420, and various file and support servers. NAS supports more than 250 SGI workstations and about 85 Sun workstations.

Interest in workstation clusters as viable alternatives to supercomputers is growing. The three major drivers of this trend are:

(1) The rate of increase in workstation CPU performance exceeds that of massively parallel processors (MPPs) and traditional supercomputers.

(2) Large memory workstations are available at reasonable prices.

(3) 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. Workstations have become ubiquitous and are found in almost every computing environment. Most workstations are used less than 25 percent of the time. One alternative is a loosely clustered configuration.

2: The Distributed Computing Team

NAS formed the NAS Distributed Computing Team (DCT) in January 1993 to investigate the issues in using clusters of workstations to solve compute-intensive aeroscience problems. NAS decided to investigate loosely coupled systems ("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) because:

(1) NAS effectively manages large groups of

1. NAS Systems Division, NASA Ames Research Center, Moffett Field, CA 94035-1000
2. Computer Sciences Corporation, NASA Contract NAS2-12961, Moffett Field, CA 94035-1000
3. Sterling Federal Systems, NASA Contract NAS2-13619, Moffett Field, CA 94035-1000
4. Fluid Dynamics Division, NASA Ames Research Center, Moffett Field, CA 94035-1000

1063-9535/94 $4.00 © 1994 IEEE

Architecture 1-SGI	Single Program/ Single CPU	Single Program/ Single System	Single Program/ Multiple Systems	Multiple Programs/ Single CPU	Mutiple Programs/ Single System	Mutiple Programs/ Multiple Systems
Serial Interactive	Exists	Exists	Exists	Exists	Exists	N/A
Serial Batch	Exists	Exists	N/A	Exists	Exists	4 Condor
Parallel Interactive	N/A	N/A	1 PVM	Exists	Exists	2 PVM/ reserve
Parallel Batch	N/A	N/A	3 DQS/PVM	Exists	Exists	5 DQS/PVM Condor

Figure 1: The Sequence of Implementation

workstations.

(2) 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.

(3) There is a need for work in this area and no major project existed to investigate it for aeroscience computations.

The prototype environment created is called the NAS Distributed Computing Facility (DCF). The total effort equalled between two and three full-time equivalents among the 15 team members. No additional funding was spent on this work.

The approach taken by the team was to first establish a loosely coupled cluster of workstations with one or both of the two major NAS workstation architectures (SGI 4D and Sun SPARC 4). Then, the team identified and resolved major system management issues to provide reasonable cycle recovery from these systems without disrupting their primary interactive visualization 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.

Existing technology was used versus developing new technology. The overall approach to this work is represented in Figure 1.

Steps included:

(1) Setting up a message passing environment.

(2) Adding a reservation system.

(3) Setting up a batch system integrated with the message passing environments.

(4) Implementing a batch system for serial programs.

(5) Integrating all the environments together.

The environment was expanded by increasing the number and types of systems and increasing the number of users (including remote users). A fully detailed report on this work is available [1]. As part of this activity, NAS sponsored a workshop, *Distributed Computing for Aerosciences Applications*, which has a separate report. [2]

3: Message Passing

The Parallel Virtual Machine [3] (PVM) software was selected as the best existing tool for message-passing and parallel program control. It was being used in a number of cluster environments,

and PVM 3.0 and later versions run on other platforms including MPPs.

It is fairly easy to translate codes developed for message-passing MPPs to use PVM calls instead, so that a programmer can take advantage of existing parallel methods and algorithms. Creating a cluster environment is relatively easy with PVM. Still, there are limitations and issues to resolve. PVM-based jobs are subject to traffic on general-purpose networks and can experience substantial delays.

PVM has to accomodate a generalized environment, while comparable MPP-based message-passing libraries can assume many safeguards to improve efficiency. PVM's message-passing implementation incorporates several layers of software overhead in addition to the network software protocol layers. On a slow network (such as Ethernet) the overhead introduced has a minor effect, but overhead effects may be significant as higher speed networks are used.

Releases of PVM after version 3.0 reduce overhead as well as increase functionality, 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. Overhead remains a concern since other PVM issues 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, but has improved with releases after 3.0.

4: Batch Systems

Part way through the prototype evaluation, the DCT evaluated batch queuing systems available for clusters. Running multiprocessor jobs requires a tool to work with an execution environment, such as PVM, in order to schedule all the processes of a single job together. The batch control system must dynamically allocate hosts, provide a mechanism for queuing jobs, support PVM, and be as unobtrusive to the primary user of the workstation.

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

The DCT compiled a list of 30 useful functions and used it to compare existing resource management tools: Network Queuing System (NQS), Distributed Job Management System (DJM), Portable Batch System [4] (PBS), LSF [5] (formerly Utopia), Distributed Queuing System [6] (DQS), and Condor [7]. The DCT also asked NAS users to rate the importance of these features. No single tool provided all the desired features. Descriptions of these comparisons, along with other parallel tools is contained in *A Survey of Parallel Programming Languages and Tools* [8].

Easy access to source code became an important factor since the DCF was an evolving prototype. It was important to understand the workings of at least two systems; both DQS and Condor were selected. DQS is used for scheduling parallel multiprocess jobs in the evening and on the weekend, and Condor is used for cycle-stealing by single-process jobs at any time.

In the DCF parallel environment, a queue corresponds to an individual processor and a user submits a parallel job to a set of queues. The job submitter selects the master controlling queue (usually his/her own machine) and the numbers and types of other queues needed. DQS is simple to install and configure. It is easy to add, delete, and modify queues, which has mostly been automated. Since the queue names were based on the hostname, administrative problems were encountered when, for example, a hostname changed, making the DQS queue name invalid. Startup scripts now handle this problem.

The DCF implementation had to allow users 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 DQS*qsub* command.This allows a user to specify a file containing a list of hosts to be included when starting up the PVM daemon. A NAS-specific problem with DQS/PVM is that each machine has its own */etc/passwd* file. NAS has a centralized account management system instead of a global password file or Yellow Pages (YP), so accounts are set up on each workstation for PVM users. In order to avoid impacting the primary users of the workstations, the accounts are only accessible between 10:00 p.m. and 5:00 a.m. Also, the user's home directory is NFS-mounted from the primary local worksta-

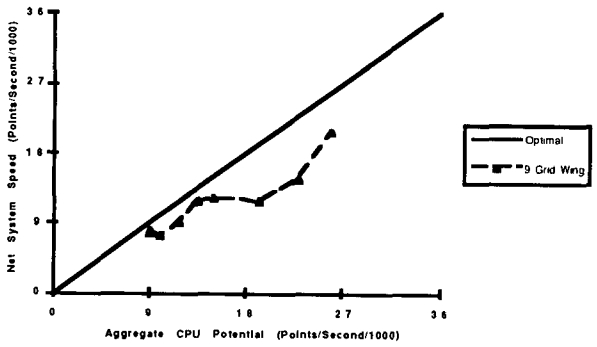

Figure 2: Larger Scale Usage for Overflow-PVM

tion, or, for remote users, from a shared workstation. Thus, PVM jobs only consume CPU and memory resources, and have no other impact–either temporary or permanent–on user disk space of the workstation.

The DCF experience with Condor has been mixed. Checkpointing did not work on SGIs. Specifically, a job would start correctly, then Condor would checkpoint it after a period of time to save its status in case of a system failure. Thereafter, the job would accumulate only system time and no user time, and eventually would abort and start again. The temporary solution was to turn off checkpointing. Currently a job suspends but never checkpoints or migrates, so many jobs typically take several days to run. Test users reported that the system was working well. There appeared to be intermittent problems with some jobs not completing and others aborting unexpectedly. Another issue is that primary users are running out of memory when a Condor job is suspended. Staff members are working with the authors of Condor on the issue of checkpointing on SGIs. When this is resolved, most of the current difficulties should be resolved.

5: The DCF Configuration

The DCF Condor cluster consists of 31 machines: 9 SGIs and 22 Suns. There are 49 SGIs in the DQS environment. This includes 8 R4000-based systems. The memory ranges from 16 MB to 256 MB. The two environments overlapped on some systems. The systems used represent a generation older technology than is available today.

6: Application Developer Experiences

This section discusses several experiences within the NAS DCF including: a large CFD application, which was one of the earliest DCF uses; NPB; and exploration of issues with porting vectorized codes that run very successfully in the Cray environment to workstations.

6.1: OVERFLOW-PVM

As a test application, the multimethod overset grid flow solver OVERFLOW was ported to the DCF, called OVERFLOW-*PVM* [9]. 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 step. PVM communications are used throughout. To better utilize the available resources, these processes are distributed among the DCF systems by a static load-balancing routine in the manager process. While this parallel decomposition is not generally scalable, the coarse granularity provides significant efficiency, considering the problems of message latency and limited network bandwidth.

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. A sustained single-precision computation rate of 50.5 million floating point operations per second (MFLOPS) was achieved, representing more than 40 percent of the combined maximum LINPACK MFLOPS rate of the machines. Figure 2 suggests that larger problems with more grids should achieve proportionately greater performance.

6.2: NAS Parallel Benchmarks

DCT members ported the same NPB implementations on the DCF and other systems [10] to evaluate the performance of representative algorithms that involve significant amounts of communication on a loosely clustered set of workstations. Three simulated applications (LU, SP, BT) are concerned with the solution of a coupled system of parallel distributed environments (PDEs) on structured grids using time-implicit relaxation techniques. All verification tests set up for the serial code were

Code	System	Proc.	No of Messages	Comm Volume (MBytes)	Comm Time (sec.)	Bandwidth (MBytes/s)	Total Time (sec)
LU	DCF	1					138.4
LU	DCF	2	7694	0.96	15.2	0.06	82.8
LU	DCF	4	15400	1.93	21.6	0.09	54.0
LU	DCF	8	30820	3.85	28.2	0.14	45.2
LU	DCF	16	46304	5.78	35.0	0.17	40.6
LU	iPSC/860	16	46304	5.78	2.5	2.31	10.0
LU	iPSC/860	32	77304	9.64	1.6	6.18	5.3
SP	DCF	1					75.8
SP	DCF	2	10	2.38	6.2	0.38	37.6
SP	DCF	4	40	4.76	8.8	0.54	22.6
SP	DCF	8	120	6.03	11.0	0.55	16.0
SP	iPSC/860	16	1104	9.34	1.6	5.70	3.7
SP	iPSC/860	32	2272	12.70	1.1	11.98	2.1
BT	DCF	1					166.8
BT	DCF	2	12	21.00	24.2	0.87	103.8
BT	DCF	4	64	31.50	61.6	0.51	77.6
BT	DCF	8	272	36.70	62.0	0.59	70.4
BT	iPSC/860	8	272	36.70	4.8	7.65	25.3
BT	iPSC/860	16	1088	39.30	3.0	13.10	12.8
EP	DCF	1					5530.9
EP	DCF	8					694.6
EP	iPSC/860	1					3811.0
EP	iPSC/860	8					476.4

Table 1: NPB (Class A) on DCF/Ethernet and Intel iPSC/860

passed, which is a very important requirement in a NPB implementation. In work related to the DCT and funded by NAS, an implementation of the other NPB kernels on various cluster environments was performed by Vaidy Sunderam and colleagues at Emory University [11].

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 that the same codes run on the iPSC/860, the NAS DCF, and other clusters. 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 and 32 MB of local memory. For the DCT implementation, the same amount of work is given to each processor, so the slowest processor in the group determines the overall speedup. On multi-

processor workstations, only a single processor is employed.

The parallel LU implementation is based on the skew hyperplane mapping approach and two-dimensional (2-D) partitioning into blocks. [12] This implementation requires sending many short messages to nearest neighbors. The implementation of the SP benchmark was a two-way pipelined Gaussian elimination algorithm using three-dimension partitioning. [13] The implementation of the BT benchmark was a transpose-based algorithm using a one-dimension partitioning. This implementation requires sending several long messages. The kernel EP was also implemented on the DCF for comparison. This kernel has almost no communication and therefore is an embarrassingly parallel algorithm.

Table 1 lists the results of these implementations on the DCF and iPSC/860. In this table, the number of messages, the total communication volume, the measured communication time, the aggregate network bandwidth and the total time per time step are given. Using eight processors, the DCF achieved a speedup ranging between 2.4 (for BT) and 4.7 (for SP). A speedup of 7.96 was achieved on eight DCF processors for the EP benchmark, where the Intel iPSC/860 processor is about 50 percent faster than the R3000 processor.

This study shows that Ethernet may not be sufficient for CFD codes based on grid partitioning algorithms that 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 outperforms 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 per second (MB/sec) per link while it is about 0.5 MB/sec for Ethernet. Latency for the iPSC/860 network is about 150 microseconds and 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 the current generation workstations, can achieve performance an order of magnitude faster than the NAS DCF machines.

The NAS DCF, composed of systems a generation behind the current workstations, 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. These results indicate that it may not be useful to pursue the entire scope of NPB-type problems on loose clusters, particularly if they consist of older generation workstations.

6.3: Porting Single Processor Cray Codes to Workstations

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

A suite of 32 standard-conforming FORTRAN 77 codes was ported from the CRAY Y-MP and CRAY C90 to SGI workstations [14]. The suite comprised a variety of codes, including kernels, pseudo-application codes, and user application codes (such as Navier-Stokes and Euler solvers). These highly vectorized codes were compiled and run as single-threaded executables on the workstations. They were not run in parallel.

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 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.

COMMON blocks with 32-bit INTEGERs and 64-bit REALs may be misaligned due to FORTRAN storage 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* 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.

Several codes used in this work so far have been standards conforming 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 CRAY FORTRAN compiler to initialize COMMON and set local variables to zero. Users may need to investigate arcane workstation compiler options that provide the same effect or modify the code to initialize all data.

Vendor	Processors	Performance (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 2: Performance Ranges

The NAS workstations provided reasonable performance, albeit much less than that available on a CRAY Y-MP as shown in Table 2. 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. There is considerably more variation in performance on a workstation than on a CRAY Y-MP. 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.

7: DCF Network Usage

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. The DCT realizes that there is a performance penalty for this but feels that the commitment not to intrude on the primary system user takes priority at this time.

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 work remains to explore them fully. Specifically, improved ways to measure the performance due to 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 and switches between machines could improve application performance. The DCT plans to investigate jobs running across other switches, FDDI, ULTRAnet, HIPPI and ATM.

Another experiment involved running a 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 dominated the subnet while it ran, since subnets with more than 40 percent utilization experience significant collisions and slowdowns. on the network. Such jobs affect other work on systems on the segment while they execute.

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

These network experiments show that if programs with significant message-passing and I/O requirements 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 parallel jobs during the day, even on idle systems, can have a major impact on other workers.

8: Development Environment

From a user's perspective, the greatest deficiency in the NAS DCF is the lack of debugging and performance monitoring tools. Currently, performance statistics are 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 multiple systems for performance. Analogous tools designed specifically for cluster computing are needed.

Debugging in the cluster environment is significantly more difficult than on single 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 complicate debugging, a program that fails, often will not completely flush the output buffer, leaving an incomplete record of the run.

9: DCF Usage Analysis

The DCF environment was open to a small but increasing number of users. Only existing tools were used to record usage information. Condor maintains a separate history log under the home directory of the Condor account on each machine, and a command is provided to combine and summarize those individual history files. The distributed nature of the history files has proven to be difficult to manage. Careful attention must be paid to ensure the presence of the accounting files on each system. Systems added to the pool and then later removed may still contain valid accounting records. It is 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. DQS, unlike Condor, maintains usage information in a single file on its master system. DQS provides a command to display or summarize usage information.

PVM does not provide any usage information. Instead, it relies on its hosts' underlying accounting systems to record process accounting data. Unfortunately, process accounting 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.

10: Expectations for Future Clustered Systems

Many limitations of the current NAS DCF have already been recognized or are in the process of being overcome. The keys to an effective loosely coupled cluster are numerous, but approaches are already being successfully demonstrated in such locations as McDonnell Douglas, with a system of over 150 HP workstations.

Workstation processor development is continuing at a dramatic pace, outstripping that of conventional supercomputers and MPPs at least for the intermediate future. Newer machines are capable of having large amounts of memory. New fast SCSI devices have increased both the volume and the rate of disk I/O. Shared-memory multiprocessors also offer great flexibility in both parallel and serial environments. The current generation of workstation processors can provide a cost-effective computational platform to greatly reduce reliance on supercomputers for smaller, shorter-running jobs.

The high level of competition in the workstation market keeps single-system software offerings from the vendors complete and of high quality, but moving supercomputer-level applications into this environment requires improved compiler and language support for functionality, 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 some systems. This makes the possibility of a seamless parallel environment a near reality. Development continues on communication

products like PVM. A cluster-oriented version of High Performance FORTRAN is becoming available from several vendors.

Many new system administration tools must be developed in order to maintain and operate a large cluster in a production environment. 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.

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 number of usable processors in an otherwise coarse-grain parallel application.

Multidisciplinary and optimization problems contain an extra level of parallelism. While some research into distributed parallel I/O has been done, little if any has been applied specifically to the cluster environment. Systems far beyond the current NFS-based cross-mounting will be required for the efficient operation of parallel jobs.

Most loosely clustered 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. 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, as any of these links may be a bottleneck.

Most current implementations of TCP/IP are slow. These protocols have difficulty effectively using the bandwidth of high-speed networks, and introduce high latencies. Better message-passing performance might be achieved by improving the implementation of TCP/IP and/or by relying on lightweight protocols which incur less overhead.

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 may improve performance or reduce overhead. Performance gains may come from combining the functionality of the current libraries and network protocols. Libraries should also take advantage of shared memory for codes on shared memory multiprocessors.

11: Conclusions

Useful work in computational aerosciences can be performed on the DCF as currently configured. If the total number of workstations at NAS were integrated into a cluster used during non-business hours, with performance equal to that demonstrated, the total computational power is potentially equal to 1.8 CRAY C90 processors. Small serial and coarse-grain parallel jobs are well suited to this environment with a higher efficiency than the examples above. If NQS queue wait-time is taken into account, very reasonable throughput can be expected from newer generation workstations.

Although the DCF has shown the potential to off-load a portion of supercomputer jobs and to run coarse-grain parallel jobs, clusters, particularly those based on existing resources, are not currently in position to replace supercomputers 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, the systems in the current DCF have inadequate double-precision hardware and compiler support. A large number of distributed parallel jobs will quickly overwhelm Ethernet, affecting other local users. There are a large number of older generation processors that do not provide adequate capacity.

Using only compute intensive benchmarks does not demonstrate the cost effectiveness for an existing base of ethernet connected workstations. Such systems can provide valuable compute cycles for little extra cost. Since organizations have already invested in the existing base, the cost effectiveness is based on the added support costs and additional equipment costs, not on the original cost of the systems.

The computing power and affordable memory of the current generation of workstations makes it possible to move many tasks currently running on the NAS supercomputers. Coarse parallelism is

quite common in aerosciences applications. Still, there are a number of areas that need significant improvement. Specifically, scheduling and queuing software, and procedures for parallel and cycle-recovery jobs.

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 are not supported by most vendors. For wide acceptance, minimal impact on primary system users must be demonstrated. Fear of loss of control and security must be abated through education as well as through software. Many users will object simply because they see no direct benefit by providing their workstation resources for others to use.

The effectiveness of workstation cycle-recovery-will be enhanced by the administrative control of the environment. 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.

Regardless of the above issues with the current DCF implementation and the need for much work, cluster computing is a legitimate and viable environment for compute-intensive work. While major issues remain, in many ways, cluster computing is more viable at this point in its evolution than either MPPs or UNIX supercomputers were at similar points. It is certainly feasible for groups with large, well-managed sets of workstations to make additional use of their capability.

Acknowledgments

The authors would like to thank their fellow members of the NAS Distributed Computing Team: David Dinucci, Karen Gundy-Burlet, and Louise Kokinakis. They would also like to thank all the NAS staff who contributed workstations.

References

All NAS references may be accessed via the World Wide Web at:
URL http://www.nas.nasa.gov/home.html

[1] K. Castagnera et al., *Clustered Workstations and their Potential Role as High Speed Compute Processors*, NAS Computational Services Technical Report, RNS-94-003, April 1994.
[2] *Report of the NAS Workshop on Distributed Computing for Aerosciences Applications*, October 18-20, 1993.
[3] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, Vaidy Sunderam, *PVM 3 User's Guide and Reference Manual*, Oak Ridge National Laboratory, May 1993.
[4] David Tweten, et al; *Portable Batch System External Reference Specification*, PBS Document, March 1993.
[5] Zhou, S. *LSF: Load sharing in large-scale heterogeneous distributed system*, In Proceedings of the Workshop on Cluster Computing, Tallahassee, FL., December 1992.
[6] Louis S. Revor, *DQS Users Guide*, DQS Documents, Florida State University, September 1992; also Green and Snyder, *DQS, A Distributed Queuing System*, March 1993.
[7] Allan Bricker, Michael Litzkow and Miron Livny, *Condor Technical Summary*, Condor Documents, University of Wisconsin, September 1991.
[8] Doreen Y. Cheng, *A Survey of Parallel Programming Languages and Tools*, NAS Technical Report RND-93-005, March 1993.
[9] Smith, M. H. and J. M. Pallis, *MEDUSA- An Overset Grid Flow Solver for Network-based Parallel Computer Systems*, AIAA-93-3312, , July 1993.
[10] Fatoohi, Rod and Sisira Weeratunga,*Performance Evaluation of Three Distributed Computing Environements for Scientific Applications*, Supercomputing 94,November, 1994.
[11] White, S., Alund, A., and Sunderam, V., *The NAS Parallel Benchmarks on Virtual Parallel Machines*, NAS Report RNR-94-008, May, 1994.
[12] 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.
[13] Naik, N., Naik, V., and Nicoules, M., *Parallelization of a Class of Implicit Finite Difference Schemes in Computational Fluid Dynamics*, Int. J. of High Speed Computing, Vol. 5, No. 1, 1993, pp. 1-50.
[14] Stockdale, I., Barton, J., *Compute Server Performance Results*, RND Technical Report (in preparation).