Paper presented at the 22nd Asian Conference on Remote Sensing, 5 - 9 November 2001, Singapore.

USING A BEOWULF CLUSTER FOR A REMOTE SENSING APPLICATION

Chao-Tung Yang
Department of Computer Science and Information Engineering
Tunghai University
181 Taichung-kang Road, Sec. 3
Taichung, 407, Taiwan
Tel: +886-4-23590121 ext. 3249
E-mail: ctyang@mail.thu.edu.tw

Chih-Li Chang, Chi-Chu Hung, and Frank Wu
National Space Program Office
8F, 9 Prosperity 1st Road, Science-based Industrial Park
Hsinchu, 300, Taiwan
Tel: +886-3-5784208 ext. 8461, Fax: +886-3-5799972
E-mail: CLChang@nspo.gov.tw

KEY WORDS: Beowulf, Cluster Computing, Parallel Processing, Remote Sensing, and Geometric Correction

ABSTRACT: To use remote sensing data has been growing. Supercomputers that are single big expensive machines with a shared memory and one or more processors meet the professional need. However, many general users process the data with PCs. They suffer slow output for the heavy burden of computation. A large-scale processing and storage system that provides high bandwidth at low cost is then their expectation. A cluster is a collection of independent and cheap machines, used together as a supercomputer to provide a solution. In this paper, a cluster, called NSPO Parallel TestBed, was built. The system architecture and benchmark performances of the cluster are presented. The parallel version of radiometric and geometric corrections was implemented and experimented on the cluster. The results show that the cluster can speed up for the remote sensing application.

1. INTRODUCTION

Extraordinary technological improvements over the past few years in areas such as microprocessors, memory, buses, networks, and software have made it possible to assemble groups of inexpensive personal computers and/or workstations into a cost effective system that functions in concert and posses tremendous processing power. Cluster computing is not new, but in company with other technical capabilities, particularly in the area of networking, this class of machines is becoming a high-performance platform for parallel and distributed applications [1, 2, 8, 9].

Scalable computing clusters, ranging from a cluster of (homogeneous or heterogeneous) PCs or workstations to SMP (Symmetric MultiProcessors), are rapidly becoming the standard platforms for high-performance and large-scale computing. A cluster is a group of independent computer systems and thus forms a loosely coupled multiprocessor system. A network is used to provide inter-processor communications. Applications that are distributed across the processors of the cluster use either message passing or network shared memory for communication. A cluster computing system is a compromise between a massively parallel processing system and a distributed system. An MPP (Massively Parallel Processors) system node typically cannot serve as a standalone computer; a cluster node usually contains its own disk and equipped with a complete operating systems, and therefore, it also can handle interactive jobs. In a distributed system, each node can function only as an individual resource while a cluster system presents itself as a single system to the user.

The concept of Beowulf clusters originated at the Center of Excellence in Space Data and Information Sciences (CESDIS), located at the NASA Goddard Space Flight Center in Maryland. The goal of building a Beowulf cluster is to create a cost-effective parallel computing system from commodity components to satisfy specific computational requirements for the earth and space sciences community. The first Beowulf cluster was built from 16 Intel DX4™ processors connected by a channel-bonded 10 Mbps Ethernet, and it ran the Linux operating system. It was an instant success, demonstrating the concept of using a commodity cluster as an alternative choice for high-performance computing (HPC). After the success of the first Beowulf cluster, several more were built by CESDIS using several generations and families of processors and network.

Beowulf is a concept of clustering commodity computers to form a parallel, virtual supercomputer. It is easy to

build a unique Beowulf cluster from available components. Currently, RGS (ROCSAT Ground System) at NSPO conducted and maintained an experimental Linux SMP cluster (SMP PC machines running the Linux operating system), named NPTB (NSPO Parallel TestBed), which is served as a computing resource for testing. NPTB is made up of 16 PC-based SMPs (eight dual-Celeron SMPs and eight dual-PIII SMPs). Nodes are connected using Fast Ethernet with a maximum bandwidth of 100Mbits, through two 3Com 24-port switches. The NPTB is operated as a unit system to share networking, file servers, and other peripherals. The system can provide a cost-effective way to gain features and benefits (fast and reliable services) that have historically been found only on more expensive proprietary shared memory systems. The typical architecture of a cluster is shown in Figure 1. The shaded boxed and the bold lines show the configuration of NPTB cluster.

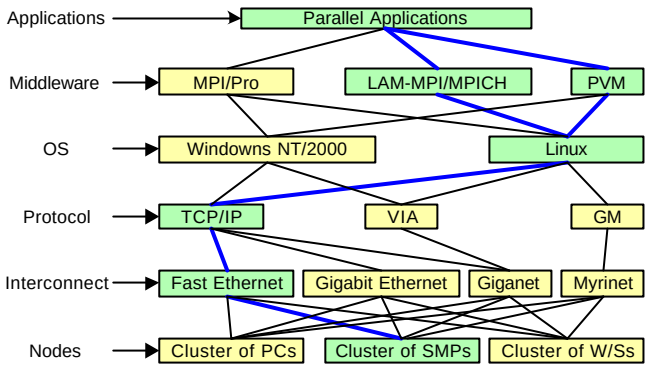

Figure 1: Typical cluster architecture.

In this paper, the cluster NPTB was built. The system architecture and benchmark performances of the cluster are presented. To meet the high performance requirements of remote sensing data processing for ROCSAT-2 [3], the parallel version of radiometric and geometric corrections was implemented and experimented on the NPTB [5, 6, 7].

2. SYSTEM DESCRIPTIONS

Our SMP cluster is a low cost Beowulf-type class supercomputer that utilizes multi-computer architecture for parallel computations. The NSPO Parallel Testbed consists of two PC clusters as shown in Figure 2. It consists of 16 PC-based symmetric multiprocessors (SMP) connected by two 24-port 100Mbps Ethernet SuperStackII 3300 XM switches with Fast Ethernet interface. Its system architecture is shown in Figure 3. There are one server node and 15 computing nodes. The server node has two Intel Pentium-III 945MHz (750 over-clock, FSB 126MHz) processors and 768MBytes of shared local memory. Each Pentium-III has 32K on-chip instruction and data caches (L1 cache), a 256K on-chip four-way second-level cache with full speed of CPU. There are two kinds of computing nodes, one is P-III-based, and the other is Celeron-based. Each P-III-based computing node with two 945 P-III processors has 512MBytes of shared local memory. Each Celeron-based computing node with two Celeron processors has 384MBytes of shared local memory. Each Celeron also has 32K on-chip instruction and data caches (L1 cache), a 128K on-chip four-way second-level cache with full speed of CPU. Each individual processor is rated at 495MHz, and the system bus has a clock rate of 110 MHz.

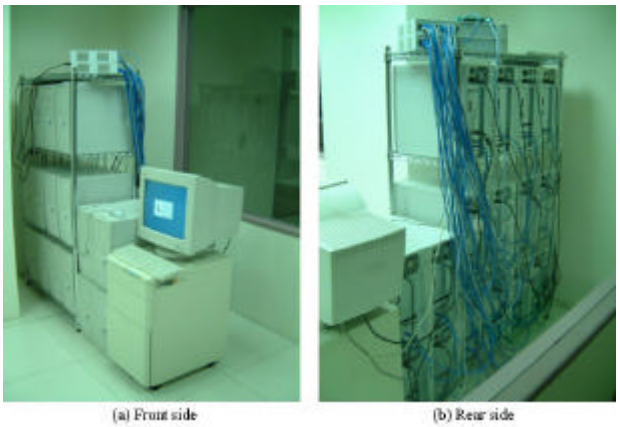

Figure 2: The snapshot of NSPO Parallel Testbed.

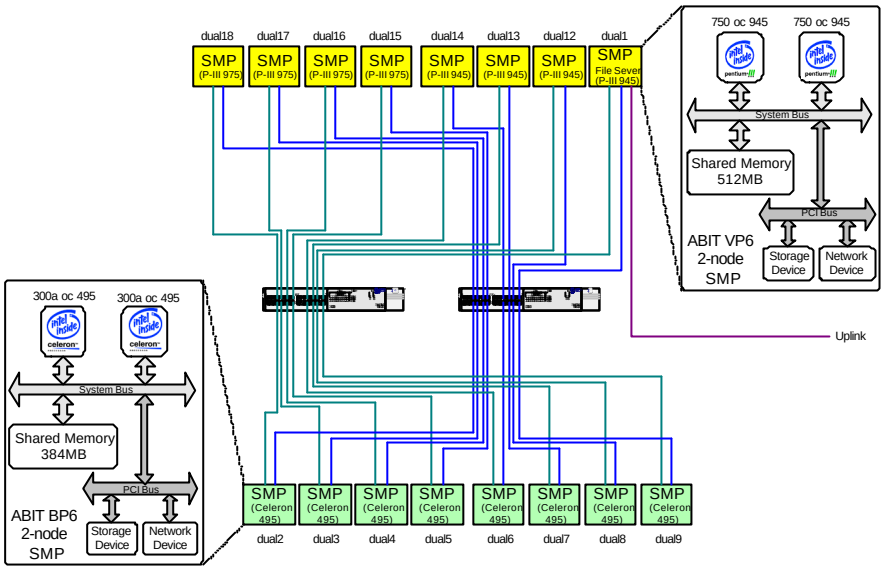

Figure 3: The NSPO Parallel Testbed system architecture.

Linux is a robust, free and reliable POSIX compliant operating system. Several companies have built businesses from packaging Linux software into organized distributions; RedHat is an example of such a company. Linux provides the features typically found in standard UNIX such as multi-user access, pre-emptive multi-tasking, demand-paged virtual memory and SMP support. In addition to the Linux kernel, a large amount of application and system software and tools are also freely available. This makes Linux the preferred operating system for clusters. The idea of the Linux cluster is to maximize the performance-to-cost ratio of computing by using low-cost commodity components and free-source Linux and GNU software to assemble a parallel and distributed computing system. Software support includes the standard Linux/GNU environment, including compilers, debuggers, editors, and standard numerical libraries. Coordination and communication among the processing nodes is an key requirement of parallel-processing clusters. In order to accommodate this coordination, developers have created software to carry out the coordination and hardware to send and receive the coordinating messages. Messaging architectures such as MPI or Message Passing Interface, and PVM or Parallel Virtual Machine, allow the programmer to ensure that control and data messages take place as needed during operation.

PVM, or Parallel Virtual Machine, started out as a project at the Oak Ridge National Laboratory and was developed further at the University of Tennessee. PVM is a complete distributed computing system, allowing programs to span several machines across a network. PVM utilizes a Message Passing model that allows developers to distribute programs across a variety of machine architectures and across several data formats. PVM essentially collects the network's workstations into a single virtual machine. PVM allows a network of heterogeneous computers to be used as a single computational resource called the parallel virtual machine. As we have seen, PVM is a very flexible parallel processing environment. It therefore supports almost all models of parallel programming, including the commonly used all-peers and master-slave paradigms.

MPI is a message-passing application programmer interface with protocol and semantic specifications for how its features must behave in any implementation (such as a message buffering and message delivery progress requirement). MPI includes point-to-point message passing and collective (global) operations. These are all scoped to a user-specified group of processes. MPI provides a substantial set of libraries for the writing, debugging, and performance testing of distributed programs. Our system currently uses LAM/MPI, a portable implementation of the MPI standard developed cooperatively by Notre Dame University. LAM (Local Area Multicomputer) is an MPI programming environment and development system and includes a visualization tool that allows a user to examine the state of the machine allocated to their job as well as provides a means of studying message flows between nodes.

3. SYSTEM PERFORMANCE

3.1 NAS Parallel Benchmark

The NAS Parallel Benchmark (NPB) is a set of 8 programs designed to help evaluate the performance of parallel supercomputers. The benchmarks, which are derived from computational fluid dynamics (CFD) applications,

consist of five kernels and three pseudo-applications. NPB 2.3 is MPI-based source-code implementations written and distributed by NAS. They are intended to run with little or no tuning, and approximate the performance a typical user can expect to obtain for a portable parallel program. The LU benchmark is based on the NX reference implementation from 1991. This code requires a power-of-two number of processors. A 2-D partitioning of the grid onto processors occurs by halving the grid repeatedly in the first two dimensions, alternately x and then y, until all power-of-two processors are assigned, resulting in vertical pencil-like grid partitions on the individual processors. This ordering of point based operations constituting the SSOR procedure proceeds on diagonals which progressively sweep from one corner on a given z plane to the opposite corner of the same z plane, thereupon proceeding to the next z plane. Communication of partition boundary data occurs after completion of computation on all diagonals that contact an adjacent partition. This constitutes a diagonal pipelining method and is called a "wavefront" method. It results in relatively large number of small communications of 5 words each.

A NAS benchmark that we chose to present here is LU. For the LU benchmark, the sizes were class A and B. The execution time of LU was shown in Figure 4. The performance numbers for 16 processors as reported in Figure 4 by the LU benchmark were 715.06 MFLOPS and 778.62 MFLOPS for class A and class B, respectively. As a measure of scalability, we selected parallel speedup, as classically calculated. The serial time was obtained by running the benchmarks on one processor. The speedup of LU benchmark is reported in Figure 4.

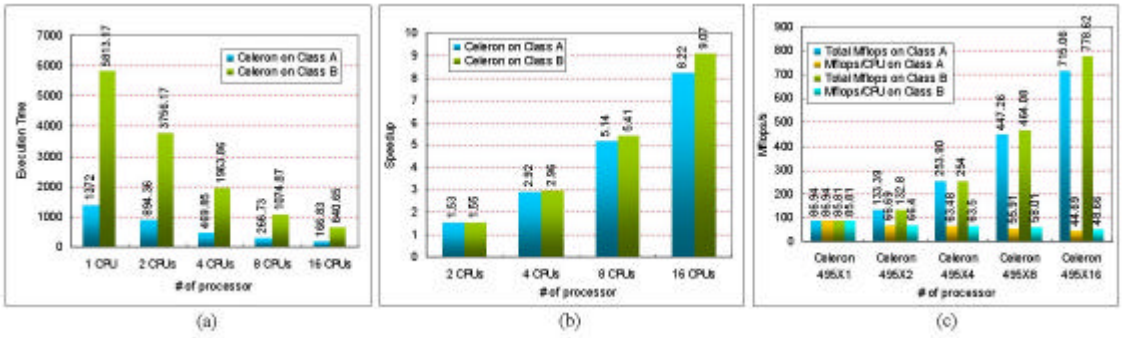

Figure 4: (a) Execution time of LU. (b) Speedup of LU using 16 processors. (c) Total Mflops/s obtained using 16 processors.

3.2 High Performance Linpack (HPL) Benchmark

HPL is a software package that solves a (random) dense linear system in double precision (64 bits) arithmetic on distributed-memory computers [4]. It can thus be regarded as a portable as well as freely available implementation of the High Performance Computing Linpack Benchmark. The HPL software package requires the availability on your system of an implementation of the Message Passing Interface MPI (1.1 compliant). An implementation of either the Basic Linear Algebra Subprograms BLAS or the Vector Signal Image Processing Library VSIPL is also needed. Machine-specific as well as generic implementations of MPI, the BLAS and VSIPL are available for a large variety of systems.

This software package solves a linear system of order n: Ax=b by first computing the LU factorization with row partial pivoting of the n-by-n+1 coefficient matrix [A b]=[[L, U] y]. Since the lower triangular factor L is applied to b as the factorization progresses, the solution x is obtained by solving the upper triangular system Ux=y. The lower triangular matrix L is left unpivoted and the array of pivots is not returned. The data is distributed onto a two-dimensional P-by-Q grid of processes according to the block-cyclic scheme to ensure "good" load balance as well as the scalability of the algorithm. The n-by-n+1 coefficient matrix is first logically partitioned into NB-by-NB blocks, which are cyclically "dealt" onto the P-by-Q process grid. This is done in both dimensions of the matrix. The right-looking variant has been chosen for the main loop of the LU factorization. This means that at each iteration of the loop a panel of NB columns is factorized, and the trailing submatrix is updated. Note that this computation is thus logically partitioned with the same block size NB that was used for the data distribution.

The HPL package provides a testing and timing program to quantify the accuracy of the obtained solution as well as the time it took to compute it. The best performance achievable by this software on your system depends on a large variety of factors. Nevertheless, with some restrictive assumptions on the interconnection network, the algorithm described here and its attached implementation are scalable in the sense that their parallel efficiency is maintained constant with respect to the per processor memory usage. In order to find out the best performance of your system,

the largest problem size fitting in memory is what you should aim for. The amount of memory used by HPL is essentially the size of the coefficient matrix. For example, if you have 4 nodes with 256 MB of memory on each, this corresponds to 1 GB total, i.e., 125M double precision (8 Bytes) elements. The square root of that number is 11585. One definitely needs to leave some memory for the OS as well as for other things, so a problem size of 10000 is likely to fit. As a rule of thumb, 80% of the total amount of memory is a good guess. If the problem size you pick is too large, swapping will occur, and the performance will drop. If multiple processes are spawn on each node (say you have 2 processors per node), what counts is the available amount of memory to each process. The performance achieved by this software package on our Parallel Testbed is shown in Figure 5. We compare the system performance of our cluster with P-III 550X16 data that from the HPL web site. Our cluster can achieve 6.179Gflops/s for the problem size 20000X20000 with channel bonding.

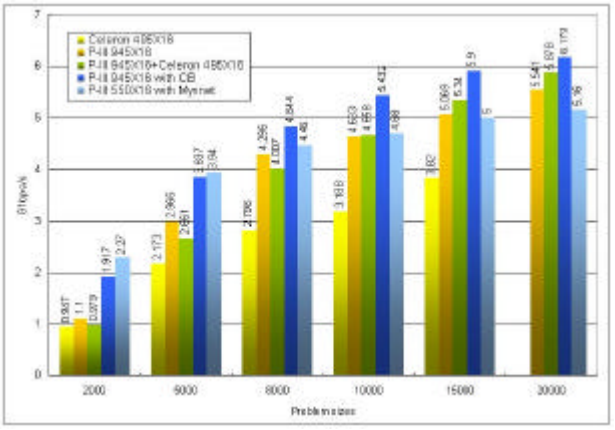

Figure 5: The system performance comparison of our cluster with HPL web site data.

4. PARALLEL RADIOMETRIC AND GEOMARTIC CORRECTION

The purpose of image rectification and restoration is to correct image for distortions or degradations that stem from the image acquisition process in geometry and radiance. The measured radiance is influenced by factors, such as changes in scene illumination, atmospheric conditions, viewing geometry, and detector response characteristics. The sources of geometric distortions include perspective, earth curvature, earth rotation, orbit inclination, atmospheric refraction, and relief displacement; aspect overlap, and variations in the altitude, attitude and velocity of the sensor platform. The parallel version of radiometric and geometric correction was implemented by NCCU [5]. The call block diagram of parallel program is shown in Figure 6.

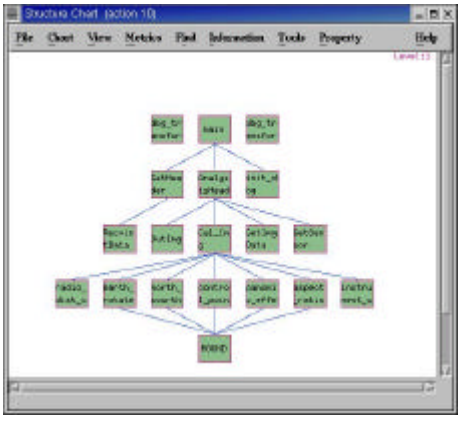

Figure 6: The call block diagram of parallel radiometric and geometric correction.

The parallel version of geometric correction consists of atmospheric correction, sensor response characteristics correction, aspect ration correction, earth rotation skew correction, image orientation to North-South, and correction of panoramic effects [6, 7]. We use the Landsat-5 Level-0 image data as input file. To correct atmospheric effects, the digital counts are converted to radiance. To correct sensor response characteristics, the different gains and offsets are applied to each pixel. Because the Landsat multispectral scanner performs a 79mX56m equivalent

ground spacing of the 79mX79m equivalent pixels, the Landsat image is too wide for its height by a factor of 79/56=1.411. Consequently to produce a geometrically correct image either the vertical dimension has to be expanded by this amount or the horizontal dimension must be compressed. To correct for the effect of earth rotation, it is necessary to implement a shift of pixels to the left that is dependent upon the particular line of pixels measured with respect to the top of the image. It is an inconvenience to have an image that is not oriented vertically in a north-south direction. To correct orientation distortion, it will be recalled that the Landsat-5 orbits in particular are inclined to the north-south line by about 9 degrees. To correct panoramic effects, the far pixels should be compressed. Finally, ground controlled points for geo-coding are applied for precise correction. The experiment was conducted on three P-III 550MHz with 128MB memory respectively and connected with 100Mbps Fast-Ethernet LAN. The resolution of input image data is 3600X2944 and size is about 10.11MB. The source and results were shown in Figure 7. The execution time of sequential program is 84.65 sec, and the execution time of parallel program with the same correction functions is 33.3 sec by using three processors. The experimental results show that a three-CPU cluster can achieve 2.54 speedup for the remote sensing data processing.

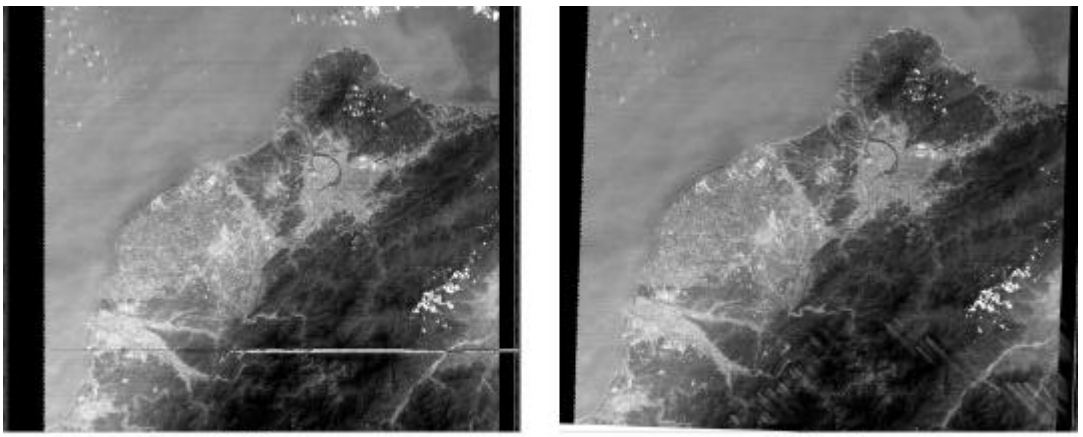

Figure 7: The image data before and after processing.

5. CONCLUSIONS

The Image Processing System (IPS) will provide real-time receiving from X-Band Antenna System (XAS). The off-line processes include the full capabilities for archiving, cataloging, user query, and processing of the remote sensing image data for ROCSAT-2. The XAS receives the high-rate link of the earth remote sensing data from ROCSAT-2 satellite and has the capability of receiving downlink data rate up to 320Mbps. The ROCSAT-2 satellite will provide the most space images with daily revisit for civil applications in Taiwan. What is emergent need for the general user is a large-scale processing and storage system that provides high bandwidth at low cost. In this paper, we presented the system architecture and benchmark performance of the NPTB Beowulf cluster. The parallel version of radiometric and geometric corrections were implemented and experimented on the cluster. The experimental results show that the cluster speeds up for the remote sensing application.

REFERENCES

[1] Buyya, R., 1999a. High Performance Cluster Computing: System and Architectures, Vol. 1, Prentice Hall PTR, NJ.
[2] Buyya, R., 1999b. High Performance Cluster Computing: Programming and Applications, Vol. 2, Prentice Hall PTR, NJ.
[3] Chang, C. L., 2000. Preliminary Design Concepts of Geometric Correction for ROCSAT-2 Remote Sensing Data Preprocessing, Proceedings of 19th Taiwan Surveying Research and Application Conference, Chunghwa City, pp. 589-598.
[4] http://www.netlib.org/benchmark/hpl, HPL – A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers.
[5] Lie, W. N., 2001. Distributed Computing Systems for Satellite Image Processing, Technical Report, EE, National Chung Cheng University.
[6] Lillesand, Thomas M. and Kiefer, Ralph W., 1994. Remote Sensing and Image Interpretation, Third Edition, John Wiley & Sons.
[7] Richards, John A., 1999. Remote Sensing Digital Image Analysis: An Introduction, Springer-Verlag.
[8] Sterling, T. L., Salmon, J., Backer, D. J., and Savarese, D. F., 1999. How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters, 2nd Printing, MIT Press, Cambridge, Massachusetts, USA.
[9] Wilkinson, B. and Allen, M., 1999. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice Hall PTR, NJ.