PAD Cluster: An Open, Modular and Low Cost High Performance Computing System

Volnys Borges Bernal[1], Sergio Takeo Kofuji[2], Guilherme Matos Sipahi[3], Alan G. Anderson[4], Marcio Lobo Netto[5]

[1,2,3,5] Laboratório de Sistemas Integráveis, Escola Politécnica da USP
Av. Prof. Luciano Gualberto, Trav. 3, n. 158, São Paulo, SP, Brazil
{volnys,kofuji,sipahi,lobonett@lsi.usp.br}

[4] Elebra Defesa e Controles Ltda.
Rua Bogaert 326, Vila Vermelha, São Paulo, SP, Brazil
{agilmor@elebra.ind.br}

Abstract—

This work presents the PAD Cluster, the result of PAD Elebra Project. It was built on COTS (commodity off-the shelf) components. It is shown the hardware and software architecture of the cluster. Some performance results are presented in the implementation of an user level communication library. Cluster management tools for a single point of control are also presented.

Keywords— **High performance computing, Cluster of workstations, parallel processing**

I. INTRODUCTION

High performance computing systems implemented with a cluster of workstations and personal computers are becoming more and more popular. This kind of systems is being used nowadays in several academic and industrial computing centers around the world. The NASA Beowulf [BEC95] [www.bewoulf.org] class of high performance computing systems is a good example of this approach.

This work presents the PAD Cluster, a high performance system implemented with this philosophy.

The PAD Cluster is being industrialized by Elebra and was developed with technical support from LSI-EPUSP and financial support from FINEP (Financiadora de Estudos e Projetos). The development of this system was based on the SPADE-II Project [www.spa.lsi.usp.br] that has been developed at LSI-USP with support from FINEP.

In its default configuration, the PAD Cluster is composed by eight processing nodes, one access workstation and one administration workstation, as shown in figure 1. Each processing node has two 333MHz Pentium II processors. All processing elements (workstations and nodes) are interconnected by a Fast-ethernet network. The processing nodes are also interconnected by a Myrinet [BOB95] [www.myri.com] high performance network.

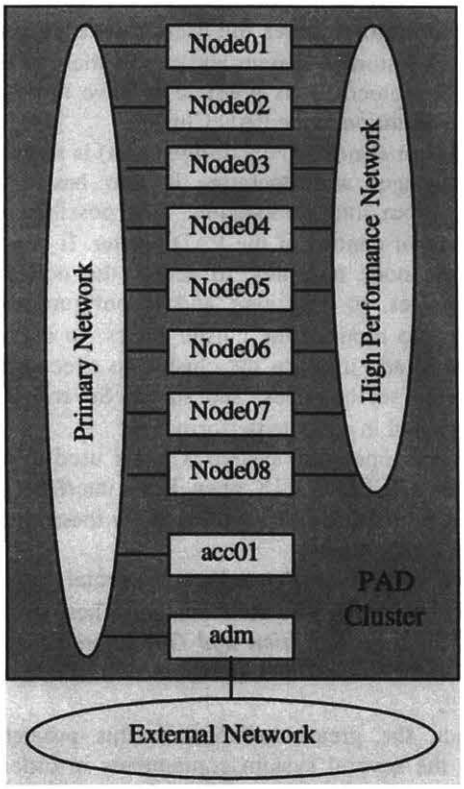

Fig. 1. A high level view of the PAD Cluster

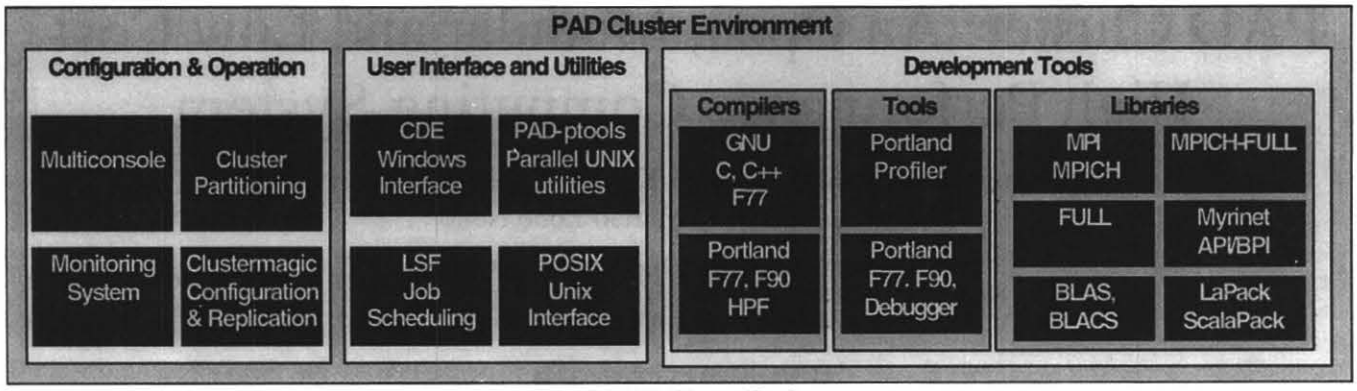

Fig. 2. PAD Cluster Environment

The *Primary Network* is intended to be used as a general purpose communication infrastructure. The purpose of the *High Performance Network* is to allow a high bandwidth and a low latency communication for data exchange among the processes of a parallel application.

Users log in the cluster through the *Access Workstation* (acc01). This is done in two ways: locally, with graphical interface based on the CDE (Common Desktop Environment) or remotely, (from outside the cluster environment) with telnet. Users edit, compile and submit their jobs through the *Access Workstation* (acc01). The Access Workstation stores the users' home directories in a RAID level 5 storage system and export them to the nodes using NFS protocol. It is possible to have more than one *Access Workstation* in the PAD Cluster.

The *Administration Workstation* (adm) is reserved to the system manager and operator. It also has a graphical interface. From this workstation, it is possible to have a single point of control of the PAD cluster. It is possible to observe the node consoles, to access the nodes by their serial consoles, to configure and reconfigure the cluster parameters, to monitor the cluster nodes, to define cluster partitions, to add users to the cluster, to execute programs in a selected set of nodes, and so on. Several tools have been developed in order to perform this.

The Linux operating system is being used in all nodes. It provides a IEEE POSIX open Unix interface [LEW91] and an open software platform that allow the addition of the necessary functionality.

Several proprietary and commercial tools were integrated into the system environment. These tools may be classified in *"Configuration and Operation Tools"*, *"User Interface and Utilities"* and *"Development Tools"* as shown in figure 2.

In fact, the great challenge in this project was to integrate the several system components in order to form one integrated system. Not only hardware components, but also software ones, as described in the following sections.

II. PAD CLUSTER ARCHITECTURE

A. System Architecture

Figure 3 presents the PAD Cluster Architecture. There are 4 interconnection systems: one Fast-Ethernet switched network for standard communication services; a high performance Myrinet network [BOB95] for parallel program communication; an experimental synchronization network to improve the performance of synchronization and collective operations; and, finally, the serial communication lines which connect each *Processing Node* to the *Administration Workstation.*

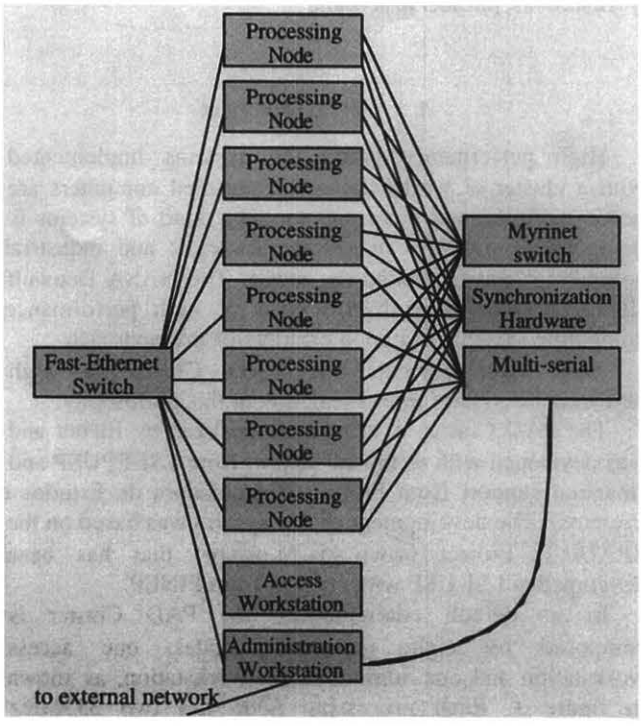

Fig. 3. PAD Cluster Architecture

B. Node Architecture

Each PAD Cluster node has two 333 MHz Intel Pentium II processors with 512 Kbytes of L2 cache, 512 Mbytes of memory, Fast-Ethernet PCI controller, SCSI PCI controller, Myrinet PCI controller and an Lm78 controller, as shown in figure 4. The Lm78 makes information about board voltages, system temperature, processor fan RPM and chassis fan RPM available.

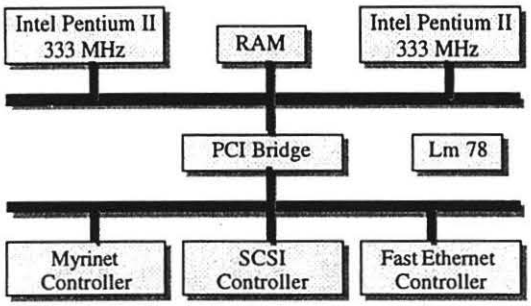

Fig. 4. Node Architecture

III. COMMUNICATION LIBRARIES

The PAD Cluster provides TCP/IP protocol stack over the *Primary Network* and *High Performance Network*. For the High Performance Network, there are two additional communication libraries: FULL and MPICH-FULL.

FULL [GER98a] [GER98b] is a user level communication library that allows one user process to access directly the physical communication interface, bypassing the TCP/IP protocol stack. This approach avoids the huge overhead imposed by the TCP/IP protocol stack and the operating system calls overhead, providing a low latency and high bandwidth communication. The implementation was based on Cornell's UNET [WEL97] architecture that was used as a basic communication layer. Over this layer was implemented another layer providing a reliable communication with flow control.

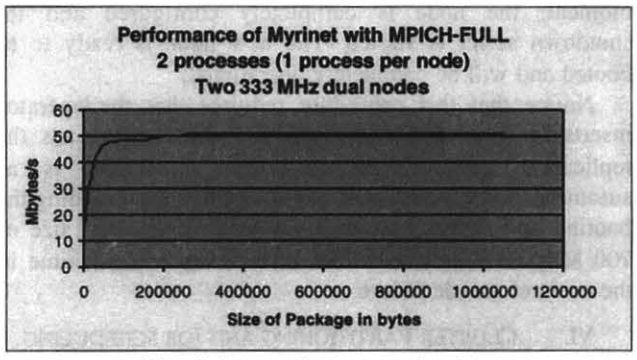

Fig. 5. Internode communication

The MPICH-FULL is implemented over the MPICH 1.1.1 distribution which is an implementation of the MPI (Message Passing Interface) [PAC97]. The communication between nodes was directly implemented over the FULL library providing low latency and high bandwidth communication. This library exploits completely the SMP architecture characteristics, implementing internode communication by message passing and intranode communication by shared memory mechanism.

Figure 5 shows the MPICH-FULL communication performance between two processes, each one running on a different node.

Figure 6 shows the MPICH-FULL communication performance between two processes running in a same node, using the SMP support.

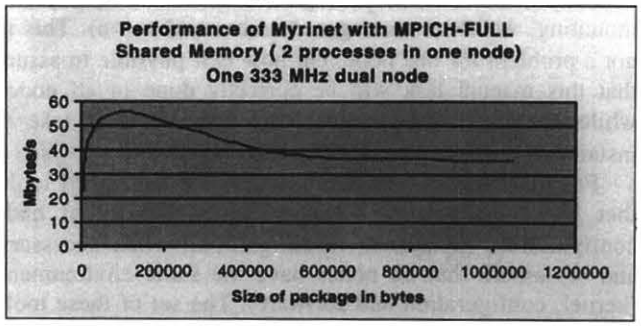

Fig. 6. Intranode communication

Finally, figure 7 shows the performance of MPICH-FULL communication performance among 4 processes in 2 nodes, where there are two processes per node.

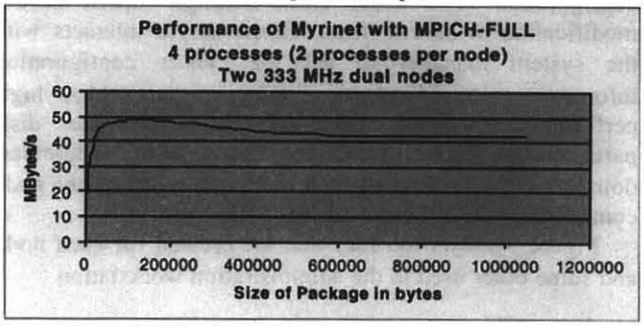

Fig. 7. Mixed intra/internode communication

Besides this implementation, there are other developments of enhanced versions of MPICH undergoing at LSI-USP, mostly trying to improve the collective performance operations using software and/or hardware assisted solutions.

IV. SYNCHRONIZATION HARDWARE

The synchronization subsystem that is being developed uses FPGA (Field Programmable Gate Array) programmable logic and is based on PAPERS [DIE96]. The main advantage of this approach is the possibility of "on the fly" reprogramming [HOR98]. The first implementation has used the standard 8 bits parallel port and provides support for data communication and synchronization, besides a global time (wall clock). More details of this subsystem may be obtained from [TOR97] [TOR98].

V CLUSTER CONFIGURATION AND NODE REPLICATION

There are two important problems related to the installation and configuration task that must be addressed in 'cluster of workstations' (COW) systems. One of them is to assure that each node (or workstation) has the same environment, as kernel, utilities and configuration. The other one is to make the installation and configuration task easier and faster. This includes operating system installation (kernel, modules, patches, packages, etc.) and configuration (kernel options, boot options, network, resolution, trusted relations, local and remote file system mounting, welcome messages, services and so on). This is not a problem for one node, but how is it possible to assure that this manual task will be correctly done in all nodes while keeping consistency? How long would it take to install and configure completely each node?

For the PAD system, there is a set of developed tools that helps the system manager in (a) the initial node configuration, (b) system reconfiguration when necessary and (c) assure that all nodes have the same environment (kernel, configuration and software). The set of these tools is called *clustermagic*.

A. Cluster Configuration

There are a lot of settings that must be done in the configuration of a node. These settings consist in the modification of several files. *Clustermagic* interacts with the system manager to get the cluster configuration information (node names, primary network, high performance network, external network, node disk partitioning, remote mountings, node MAC addresses, domain name, DNS servers, etc.) in order to create the node configuration files automatically.

Figure 8 shows the files that are created for each node and some other used in the administration workstation.

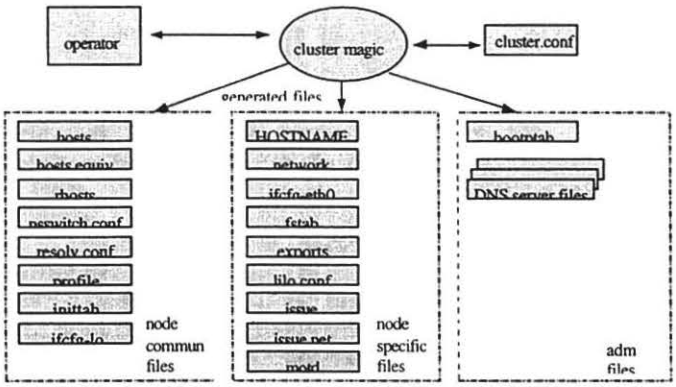

Fig. 8: *Clustermagic* tool and the created files

The *clustermagic* creates automatically the DNS server configuration files for the PAD cluster and the bootptab file which is used by the BOOTP server in the replication process.

B. Node Replication

Node replication means to install completely one node based on the replication of another node, called womb node. The womb image is a compressed file with the file hierarchy of the node.

A *clustermagic replication diskette* was developed to assist this task. It is a Linux environment specially designed to boot the Linux operating system and execute the replication task. The interaction with the operator is done in the serial console that is shown in a window in the administration workstation and also by the node display that shows some messages informing the current step.

When the node is booted with the replication diskette, the diskette boot sector is loaded and the boot program reads the compressed kernel and loads it into the memory. Then, the compressed root file system is loaded into a ramdisk and the kernel initialization is started. After that, the node sends a BOOTP [CRO85] [WIM93] request and the administration workstation answers with its identification and some other information (IP address, broadcast address, network address, default gateway, node name, domain name, DNS server and search list). The node network interface and the host name are configured. At the end of the operating system initialization, a script shows two options to the operator: single user shell or replication. If "single user shell" option is selected, a new shell is started for the operator. If "replication" option is selected, the system reads the cluster configuration information stored in the administration workstation and starts the node replication. First, the disk is partitioned. The partitions are initialized (mkfs and mkswap) and the partitions with file systems are mounted. Then, in the replication step, the womb image is copied to the local file system. After some minutes, the node configuration files (previously generated by clustermagic) stored in the administration workstation are copied to the local file system. Finally, the boot sector is initialized and the local file systems are dismounted. At this moment, the node is completely configured and the shutdown script is started. The new node is ready to be booted and will be completely operational.

Notice that this procedure requires that the operator inserts the boot diskette, resets the node and selects the replication option. The replication procedure is an automatic process and takes about 12 minutes, including the booting and shutting down, for a total file system size of 700 Mbytes. This also allows a small replacement time in the event of a node failure.

VI. CLUSTER PARTITIONING AND JOB SCHEDULING

The cluster partitioning tool allows the operator to group the nodes. These node groups are called cluster partitions. Figure 9 illustrates the cluster partitioning tool function. The following information is held for each cluster partition: *name* (the partition name), *description* (a message

with a description of the purpose of the partition), *nodes* (the included nodes), *users* (the users that are allowed to access the partition nodes), *state* (it may be active or inactive), and *goal* (the partition goal).

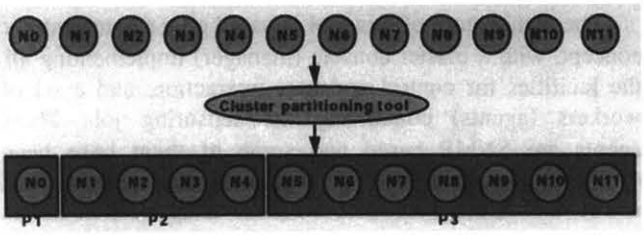

Fig. 9. Example of cluster partitioning

The access into each partition is restricted only to those users that are defined by the operator. This is possible because the tool interacts with PAM (Pluggable Authentication Modules). A partition may be in an active or inactive state. Sometimes the operator would like to perform some maintenance in the partition. Then, it may be marked as inactive and its entire configuration is maintained. It may be marked active again without the necessity to reconfigure it. There are three possibilities for *goal*: LSF, MPI and OTHER. If LSF (a job scheduler system) is chosen, the tool will automatically create the necessary configuration files to the LSF. In this case, LSF will be responsible for the resource management for that partition. If MPI is chosen, the tool will automatically create the MPI node files. Figure 10 gives an example of the role of the job scheduler and the node partition tool in a cluster.

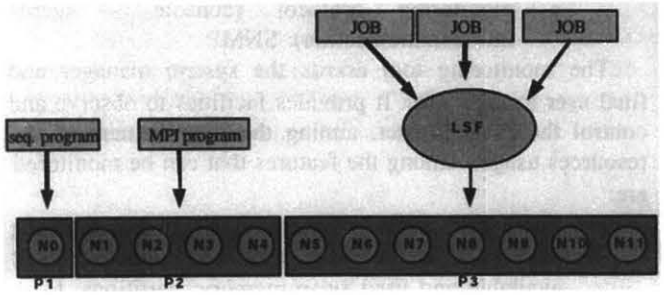

Fig. 10. Example of job scheduling over one partition

There is a graphical interface that allows the operator to perform the partition configuration. This interface is integrated in the Xadmin and allows the operator to create and configure a partition as shown in figure 11. The operator may assign a node to the created partition by dragging and dropping the node from the "unassigned partition" to the recently created partition.

There are some user utilities that allow to consult about cluster partition configuration. The cluster partition configuration is used by other utilities, like a monitoring tool, UNIX parallel tools, and an operational tool when performing cluster operations.

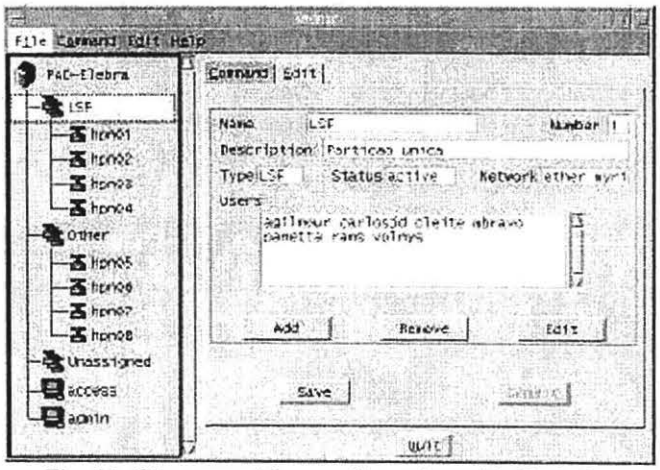

Fig. 11. Cluster partition configuration with Xadmin

The *job scheduling* tool used in the PAD cluster is the LSF, a commercial software that allows the user to submit jobs to the cluster and to monitor it. It has also integration for MPI applications.

VII. OPERATION TOOLS

The operation tools allow the operator to maintain the PAD cluster operational. There are two developed tools: the multiconsole and the Xadmin.

A. *Multiconsole*

The *multiconsole* permits the operator to access the node console from the administration workstation using the graphical interface. The node consoles are redirected to the serial line. The administration workstation has a multiserial board that connects the serial console from each node. The *multiconsole* presents a window console for each node, as shown in figure 12.

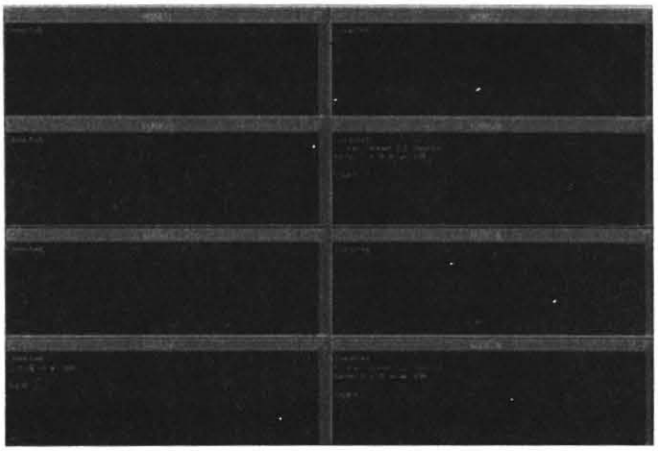

Fig. 12. The *multiconsole* interface

B. *Xadmin tool*

The *Xadmin* tool allows to partition the cluster, to restart and halt a node, to send messages to users, to update configuration files and to execute specific commands in graphically selected cluster nodes. The graphical interface of *Xadmin* is shown in figure 13.

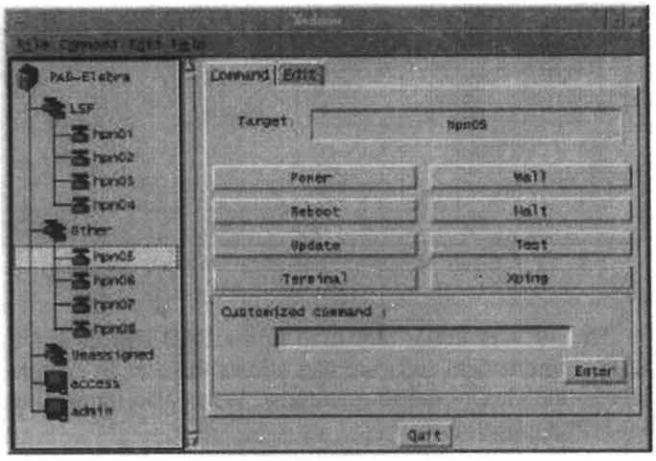

Fig. 13. *Xadmin* tool

There is a related tool that shows the operational state of of the cluster nodes, shown in figure 14. This state is always verified by the *Xadmin* tool before activating some remote operation.

Fig. 14. A view of the node operational state

VIII. PARALLEL UNIX TOOLS

In order to provide easy cluster operation by the users and operators, it was developed a set of Unix parallel tools. These utilities allow, for instance, to copy some files from the access workstation to a set of nodes or to a specific partition. Notice that these utilities are integrated to the partitioning tools.

IX. CLUSTER MONITORING

The monitoring tool has been designed in order to provide facilities to observe diverse features of this parallel computer, allowing its user/manager to have a better view about how the computation is being conducted. Two main aspects have been considered here. First, the ability to visualize the complete PAD Cluster, including the processing nodes and the communication network. This is important because it shows the current state of all processing and communication resources, and, therefore, it shows how well they are being used, i.e., the quality of the balance. The second reason for the usage of this monitor is to go inside the application for a deeper understanding about how well is being performed the parallel computation.

The architecture of this tool is based on a master-worker concept, with a master console (manager) implementing all the facilities for control and user interaction, and a set of workers (agents) conducting the sensoring job. These agents are SNMP based and some of them have been extended to provide more specific information. The console is a Java tool, designed to be available as application and as applet running in a Web Browser. The console concept is fully based on the exploitation of fine multithread parallelism.

The monitoring system has been designed to offer the following features:

a) to provide an efficient way to observe different variables of PAD Cluster:
- through the configuration of agent features (sensoring and control);
- through an appropriate visualization of the information got from agents.

b) to allow the user interaction through a graphical interface integrating all its functionalities:
- this interface will be provided as an applet, allowing therefore access through web browsers.

c) to use standards in its implementation:
- programming language: JAVA
- internet protocol: HTTP
- monitoring protocol (console - agents intercommunication): SNMP

The monitoring tool assists the system manager and final user in their jobs. It provides facilities to observe and control the PAD Cluster, aiming the improvement of the resources usage. Among the features that can be monitored are:

- inside each node: information about the operating system; total, available and used memory; total, available and used swap memory; partitions: total, available and used area; information about TCP/IP; information about CPU usage; user processes; system usage; information about the main board; temperature and fan.
- fast-ethernet network: bandwidth; latency.
- myrinet network: bandwidth usage.
- system: power supply; no-break autonomy; rack door state.

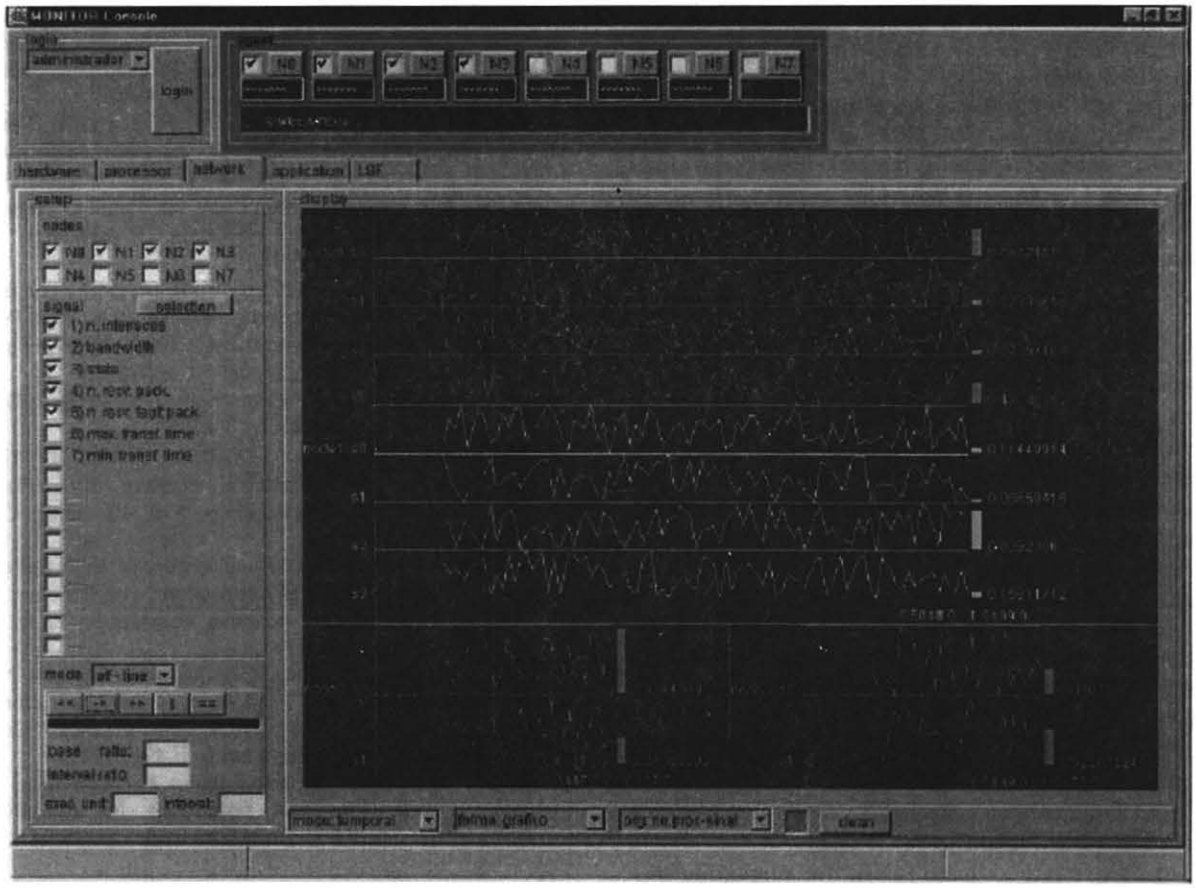

Fig. 15. The Monitoring System

X. DEVELOPMENT ENVIRONMENT

The choice of the tools for the cluster obeyed some rules: (a) the compilers should be native and standard; (b) the compilers should have a debugger and, if possible, a profiler as accessory tools, and (c) the use of the compilers with other tools available in the market. The presence of features different from the standard and the availability of precompiled sequential libraries should also be considered.

As C and C++ from GNU (GCC and G++) are standard for the Linux community, they were chosen to be our standard tools. With the Gnu C Compilers, the gdb (GNU debugger tool) and DDD, a graphical interface to gdb, are installed.

The choice of the Fortran compiler was more difficult. There were four different possibilities (NAG Fortran, NAS Fortran, Portland Group Fortran and Absoft Fortran). Two of them (NAG Fortran and NAS Fortran) were not native, so they were discarded. To decide which compiler should be used, the second and the third items had to be considered. The analysis of these two packages is presented below.

The Portland Group Fortran is a package that includes a Fortran77 (PGF77) and a Fortran90 compilers (PGF90), as well as a High Performance Fortran compiler (PGHPF). It also includes a graphical profiler (PGPROF) and a debugger (PGDBG). This package can also be used with the VAMPIR profiler and with the TotalView parallel debugger. The Portland Group Package does not provide any precompiled mathematical library.

The Absoft Fortran is a package that includes a Fortran77 (PGF77) and a Fortran90 compilers. It also includes a graphical debugger and a set of precompiled sequential mathematical libraries. It does not provide a graphical profiling tool.

Both tools are very similar. Because of the graphical profiling tool and the HPF compiler availability the Portland Group package was chosen.

The following libraries were compiled with our choices of compilers to be used: MPICH, BLAS, BLACS, LaPack and ScaLaPack.

Three other choices of compilers are available in the system. As g77 (GNU Fortran77 compiler) is free, it is also available in the system. The package of Portland Group

compilers provides, with a small difference in the cost, C and C++ compilers (PGCC and PG++) that are also available on the system.

There are two tools that can be added to the system: TotalView debugger and VAMPIR profiler. These tools have recently been ported to Linux and have been made available lately. At present the ratio cost/benefit of the addition of these tools is being studied, because of its high cost, especially for the debugger.

XI. CONCLUSIONS

High performance communication and high performance processing elements are the main components of a high performance parallel system. However, it is very important in order to have a production quality system to have tools also for the installation, administration and management and to help the user on his job. This work has presented some of these tools and their basic mechanisms. It has also shown some details of the communication libraries as well as some performance figures.

The parallel version of RAMS, a weather forecast application, is being optimized to run over the PAD Cluster [MEN99]. For this application, the cluster has shown good price-performance ratio.

XII. ACKNOWLEDGMENTS

We would like to thank the PAD Cluster development team from LSI-EPUSP and Elebra Company.

We would also like to thank FINEP for supporting this work.

XIII. REFERENCES

[BEC95] BECKER, J. D.; et al. BEOWULF: A parallel workstation for scientific computation. Proceedings. International Conference on Parallel Processing, 1995.

[BOB95] BOBEN, N. J.; COHEN, D. et al. Myrinet:a Gigabit-per-second local area network. IEEE Micro. pp29-35, Feb. 1995

[CRO85] CROFT, W. J.; GILMORE, J Request For Comments #951: Bootstrap Protocol., 1985.

[DIE96] DIETZ, H. G. A fine-grain Parallel Architecture Based on Barrier Synchronization. Proceedings of the International Conference on Parallel Processing. Pp247-50. August 1996.

[GER98a] GEROMEL, Paulo A.; KOFUJI, Sergio T. Avaliação do U-NET em Clusters com Rede Myrinet. In: X Simpósio Brasileiro de Arquitetura de Computadores – Processamento de Alto Desempenho. Anais. Buzios, RJ. 1998. pp. 119-128.

[GER98b] GEROMEL, Paulo A. Protocolos leves de comunicação para sistemas de alto desempenho. São Paulo: Escola Politécnica da USP, 1998 (MSC Thesis).

[LEW91] LEWINE, D. POSIX Programmer's Guide. O'Reilly & Associates, 1991

[MEN99] MENDES, C.; PANETTA, J. Selecting Directions for Parallel RAMS Performance Optimization. In: Symposium on Computer Architecture and High Performance Computing, 11, 1999. Proceedings.

[PAC97] PACHECO, P. S. Parallel Programming with MPI. Morgan Kaufmann Publishers, 1997

[TOR97] TORRES, Martha X.; KOFUJI, Sergio T. The barrier synchronization impact on the MPI programs performance using a cluster of workstations. International Symposium on Parallel Architecture, Algorithms and Networks. Proceedings. Taipei. Taiwan, ROC, 1997.

[TOR98] TORRES, Martha X.; KOFUJI, Sergio T. The Influence of Barrier Synchronization on the MPI Programs Performance Using the AP 1000 Multicomputer. 16th IASTED International Conference on Applied Informatics; Garmisch-Partenkirchen, Germany, Proceedings. 1998. pp 334-336.

[HOR98] HORTA, Edson; KOFUJI, Sergio T. Using a reconfigurable switch to improve MPI performance. In: JCIS'98. Proceedings, Vol. III. North Carolina, EUA. pp 66-69.

[NET98] NETTO, Marcio L. Development of High Performance Parallel Graphical Applications with Efficient Parallel Processing and Adaptive Techniques, Ph.D. Thesis, Shaker Verlag, 1998.

[WEL97] WELSH, M. et al. ATM and Fast Ethernet Network Interface for user-level communication. In: Proceedings of High Performance Computer Architecture 3. San Francisco, Feb. 1997

[WIM93] WIMER, W. Internet Request For Comments #1542: Clarifications and Extensions for the Bootstrap Protocol. 1993.