

CRONO: A Configurable Management System for Linux Clusters

Marco Aurélio Stelmar Netto¹, César A. F. De Rose²

¹*Research Center in High Performance Computing CPAD-PUCRS/HP, Brazil*

²*Catholic University of Rio Grande do Sul, Computer Science Department, Brazil*

Abstract

There are several application fields in which parallel processing is an essential tool. Using parallel architectures, it is possible to solve very complex problems that need high computing power. In this context, cluster computing has become an excellent alternative to mainframes, usually much more expensive. In order to exploit all available resources and to try to achieve optimum performance, efficient cluster management tools are needed. When looking for such a manager for the clusters of our research lab we were not satisfied with the services and level of configuration provided by the most popular systems like Computing Center Software, DQS, and Portable Batch System. Although very powerful and configurable, these systems are reasonably complex to install and configure. Therefore we decided to implement our own management system, a highly configurable one, but easier to install and to use. CRONO is being used for the last 6 months to manage our three Linux clusters and it is already very stable. Each cluster has its own peculiarities, having a different number of nodes (4, 16, and 32), different interconnection networks (combinations of SCI, Fast-Ethernet and Myrinet) and also different user profiles. Although being simple to install and configure, CRONO is a highly configurable system, allowing the system manager to configure the access privileges individually for each machine and group of users.

1 Introduction

Cluster architectures [1] are becoming a very attractive alternative when high performance is needed. With a very good cost/performance relation and good scalability big cluster systems with hundred of nodes are becoming more popular in universities, research labs and industries. Linux is usually the operating system used in these

systems because it's free, efficient and very stable. One remaining problem in such a system is to manage all the nodes efficiently as one machine and deal with issues like access rights, time and space sharing, reservation and jobs queuing. When looking for such a manager for the clusters of our research lab¹ we were not satisfied with the services and level of configuration provided by the most popular systems like Computing Center Software [2], DQS [3] and Portable Batch System [4]. Although very powerful and configurable, these systems are reasonably complex to install and configure. Therefore we decided to implement our own management system, a highly configurable one, but easy to install and to use. Although being simple to install and configure, CRONO is a highly configurable system, allowing the system manager to configure the access privileges individually for each machine and group of users.

This paper presents the CRONO cluster management system and is organized as follows. Section 2 presents some advantages of managing clusters on Linux. Section 3 gives an overview of the main functionalities of CRONO and section 4 describes its architecture and main configuration files. Section 5 shows how easy is to install and configure CRONO and section 6 discuss the use of CRONO in our lab. In section 7 our conclusions are presented.

2 Cluster Management over Linux

The CRONO system was developed to be a highly configurable manager for Linux clusters. It has been coded using the C language and rely on several scripts and configuration files to be adapted to specific administration needs and machine configurations. The Linux operating system has several advantages for the development of a cluster management system (CMS). Linux is an open source system and its configuration files are very simple to read and modify, therefore the CMS can define environment variables and access restrictions easily. Another main advantage is the possibility to modify the kernel to provide better support to the CMS. This is very useful when, for example, the CMS needs to manage not only nodes, but also other resources like memory, processor and network adapters.

3 Crono Main Functionalities

CRONO provides two basic allocation modes, space-sharing and time-sharing. The first one is used when the user needs exclusive access to allocated nodes, for example when application performance is being measured. The second one is used in situations where the users are only testing their programs and, therefore, do not care about performance. Space-sharing is a very interesting alternative in teaching environments, allowing large groups of students to use the cluster at the same time.

Another main feature of CRONO is its flexibility to define access rights. Using configuration files the system administrator can create user categories and associate access restrictions to these categories or to individual users. These restrictions are defined by the maximum time and maximum amount of nodes used in allocations

¹Research supported by HP Brazil

and reservations. There is also the possibility to define restrictions based on periods of the day, day of the week and target machine.

To configure the execution environment for programs, CRONO supplies scripts for pre- and post-processing of requisitions. When the user time initiates, CRONO will use two scripts: one of them controlled by the administrator, and the other by the user itself. This mechanism is very useful, for example, to automatically generate MPI [5] machine files. When the time of a user is over, two post processing scripts will be used in the same way. Users can interact to the system through a graphical interface or using commands in the shell (*bash*, *tcsh*, etc). Several services are supported for users and the system administrator like: information about the allocation queue and on access rights, submission of execution jobs, requests to release the resources and configuration of the execution environment.

4 Crono's Architecture

The CRONO's architecture is composed by the following four modules, which will be described in more detail in the following subsections:

- The User Interface (UI) is the main entrance to the cluster nodes and is composed by several tools;
- The Access Manager (AM) is responsible for the authentication and the verification of access rights;
- The Requisition Manager (RM) does the scheduling of the user requests and the preparation of the execution environment;
- The Node Manager (NM) is the module running on each node of the cluster and its main function is to block the access to the nodes.

The communication between the modules is done through the sockets interface [6], therefore allowing modules to be in different machines. Moreover, the modules are organized like a chain, that is, if the User Interface needs to send a message to the Node Manager, the message will pass through the Access Manager and the Requisition Manager.

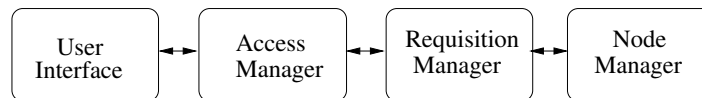


Figure 1: CRONO's architecture.

4.1 User Interface

The User Interface is responsible for providing interaction to the system, through a graphical user interface or the UNIX shell environment (like *bash*, *tcsh* or *csh*). There are seven commands available:

`crqview` for displaying information of the requisitions queue, like user names, starting and finishing time, cluster name, number of nodes available and allocation modus (space-sharing or time-sharing);

`cralloc` for node allocation in the case the user wants the resources as soon as possible;

`crrels` for node release or cancellation of an user request;

`crnodes` to obtain a list of nodes which the user has access;

`crinfo` for displaying information about the clusters which the user has access. These information include access rights, number of cluster nodes, special periods of use, maximum values for allocation and reservation, etc;

`crnmc` to execute operations directly on the nodes that are allocated. The available operations are provided by the administrator, and the users may obtain which commands are available also with this command. An example of user operation could be `killp` to kill all processes running on the nodes;

`crsetdef` to define and set environment variables for the most common command parameters. In the case parameters are omitted the system will look for default values in these variables.

4.2 Access Manager

The Access Manager is the module of CRONO responsible for receiving the user requisitions from the User Interface and validate them, before forwarding them to the Requisition Manager, if necessary. The Access Manager daemon can manage many clusters and may use distinct policies for each them. CRONO allows the system administrator to attribute access rights to individual users and groups of users. These users groups are not the same groups used by Linux. For request validation, the Access Manager uses three files: the groups file, the users file and the priorities file. The groups file is used to define the groups and is shown below:

```
#####
# group <group name>
#   <user>
#   <user>
#   ...
# endgroup
#   ...
#####
group master
    lisa
    bart
    maggie
endgroup
group jedigroup
    luke
    leia
    yoda
endgroup
#####
```

The users file contains the priorities for the CRONO users and users groups. For users not included in this file, the system will use the default priority. An example of users file is showed below:

```
#####
# priority <number>
#   <user>
#   <user>
#   ...
# endpriority
#   ...
#####
priority 1
  master
  stelmar
  homer
endpriority

default 10
#####
```

The priorities file defines the policies for cluster access control. The administrator can define the maximum time and number of nodes for allocation and reservation. Furthermore it's possible to define special periods for using the cluster, hence each priority can have two definitions, one for normal periods and the other one for special periods. For example, it's interesting to extend the time and number of nodes limits at weekend and at night, when there are fewer users requests to the resources. The following priorities file exemplify such a definition:

```
#####
# PN=> priority number
# T=> type(normal[n],special[s])
# MTA=> max time allocation
# MTR=> max time reservation
# MNA=> max nodes allocation
# MNR=> max nodes reservation
#####
# special
#   <week day> <initial time>-<final time>
#   <week day> <initial time>-<final time>
#   ...
# endspecial
#
# <PN>.<T>.<MTA>.<MTR>.<MNA>.<MNR>
#####
# Period of day and week that
# has a special treatment
#####
special
  sun 00:00-23:59
  mon 00:00-8:00
  tue 00:00-8:00
  wed 00:00-8:00
  thu 00:00-8:00
  fri 00:00-8:00
  sat 00:00-23:59
endspecial
# Priorities
1.s.60.180.16.8
1.n.30.0.8.0
10.s.30.0.8.0
10.n.15.0.4.0
#####
```

After the Access Manager daemon checks these files, it forwards the request to the Request Manager if necessary or sends a message to the user informing that the

user doesn't have access to the requested cluster.

4.3 Request Manager

The Requisition Manager is the CRONO module responsible for scheduling the requests authorized by the Access Manager and preparing the execution environment. This module has two sub-modules: the scheduler, which allows space-sharing and time-sharing, and the environment set up tool, which runs scripts before and after the user allocation time. The next subsections describe each of the sub-modules in detail.

4.3.1 Scheduler

There are many scheduling algorithms already available that could be used in the CRONO scheduler, like Shortest Job First (SJF), Longest Job First (LJF) or First In First Out (FIFO). However, these algorithms have problems to cope efficiently with the allocation modes available in CRONO. The Request Manager tries to make good use of available resources (time and nodes) that would be wasted using the First In First Out algorithm, but without injure the users that are already waiting for resources. If an user expects to be attended at a specific time, this user will be attended in the worst case in that time. The scheduling policy implemented in the CRONO scheduler is based on the FIFO scheduling and can be understood through two examples. These examples are based on a requisitions sequence of eight users, as shown in Table 1. The system administrator defines the maximum number of users that can share the same node at the same time (time-sharing). In the following examples, this value is set as two, and the cluster is composed by eight nodes.

User	Requisition Time	Requested Time(min)	Nodes	Access Type
U1	8:00	15	3	Exclusive
U2	8:03	10	8	Exclusive
U3	8:05	2	5	Shared
U4	8:05	5	4	Shared
U5	8:18	10	5	Exclusive
U6	8:20	5	7	Exclusive
U7	8:22	7	1	Shared
U8	8:22	10	2	Exclusive

Table 1: Example of users requisitions

Figure 2 illustrates the requests queue for Table1 using the FIFO algorithm. In this graph we observe the resources that could be used without increase the attendance time of the users who still do not own the resources. The sequence of users and scheduler operations are shown in the Table 2.

After 55 minutes, all requests were attended and the resources were released. With the intention to reduce the amount of wasted resources, the FIFO policy was modified to allow some requests to overtake a waiting request only in the case that the expected time of attendance of the blocked request will not be exceeded by this

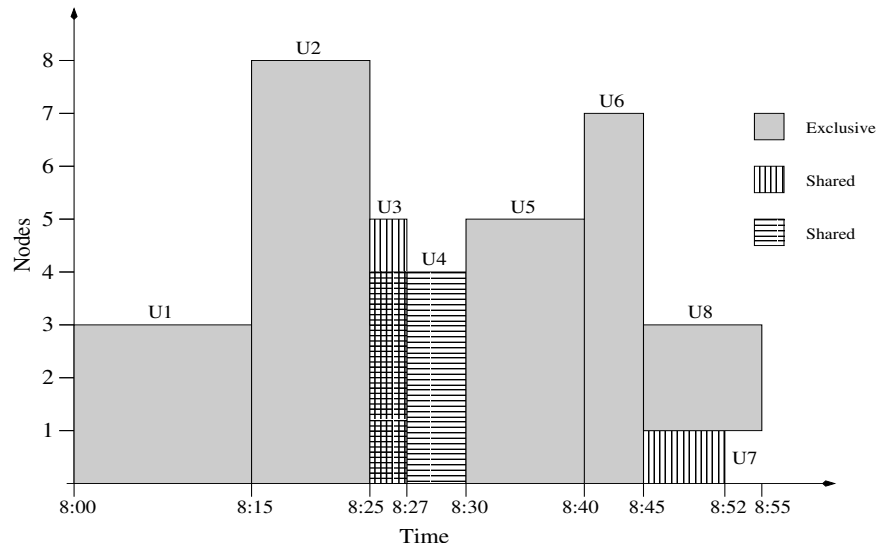


Figure 2: Queue using the FIFO scheduler.

behavior. Figure 3 shows the utilization of the resources using the CRONO scheduler, for the same requisitions of Table 1.

In this second case, the requisitions of users U3, U4, U8 and U7 overtake the other users, as shown in Table 3.

As we can see, the resource waste was minimized, since all the users have been taken care of in 40 minutes instead of 55 minutes. When a requisition is ready to be attended, the environment should be prepared to the users, as explained following.

4.3.2 Environment set up tool

It's interesting to execute some tasks when the user time starts and when it finishes. This sub-module of the Request Manager is responsible for doing these tasks through four scripts:

- The master pre-processing script (MPREPS) is used by the administrator and defines the operations that are executed when the user time starts;
- The master post-processing script (MPOSTPS) is used by the administrator and defines the operations that are executed when the user time finishes;
- The user pre-processing script (UPREPS) is used by the user and defines the operations that are executed when the user time starts;
- The user post-processing script (UPOSTPS) is used by the user and defines the operations are executed when the user time finishes.

Both the users and administrator scripts are defined for each cluster managed by the system. This is very useful, for example, when multiple clusters with different interconnection technologies are managed, like Fast-Ethernet, Myrinet [7] or SCI [8].

OS Time	Operations	Attendance Time
8:00	U1 makes a request and U1 is attended	8:00
8:03	U2 makes a request and wait	8:15
8:05	U3 and U4 make a request and wait	8:25
8:15	U1 releases the resources and U2 is attended	-
8:18	U5 makes a request and wait	8:30
8:20	U6 makes a request and wait	8:40
8:22	U7 and U8 make a request and wait	8:45
8:25	U2 releases the resources and U3 and U4 are attended	-
8:27	U3 releases the resources	-
8:30	U4 releases the resources and U5 is attended	-
8:40	U5 releases the resources and U6 is attended	-
8:45	U6 releases the resources and U7 and U8 are attended	-
8:52	U7 releases the resources	-
8:55	U8 releases the resources	-

Table 2: Operation using the FIFO scheduler.

OS Time	Operations	Attendance Time
8:00	U1 makes a request and U1 is attended	8:00
8:03	U2 makes a request and wait	8:15
8:05	U3 and U4 make a request and overtake U2	-
8:07	U3 releases the resources	-
8:10	U4 releases the resources	-
8:15	U1 releases the resources and U2 is attended	-
8:18	U5 makes a request and wait	8:25
8:20	U6 makes a request and wait	8:35
8:22	U7 and U8 make a request and wait	8:40
8:25	U2 releases the resources, U5 is attended, U8 and U7 overtake U6	-
8:32	U7 releases the resources	-
8:30	U4 releases the resources and U5 is attended	-
8:35	U5 and U8 release their resources and U6 is attended	-
8:40	U6 releases the resources	-

Table 3: Operation using the CRONO scheduler.

Hence, it's necessary to create specific machine files for each environment the users programs can use.

Besides executing the pre-processing scripts at the starting time of a allocation request, CRONO sends a message to the users through the `tty` terminal informing that the resources are available. Because users are usually accessing with more than one terminal, CRONO uses the `utmp` file to discover the terminal with the least idle time, and sends the message to that terminal. This is done because there is a greater probability that the user is reading that terminal.

4.4 Node Manager

The Node Manager is the CRONO module executed on each node of the managed cluster and it is responsible for the access control and the execution of some opera-

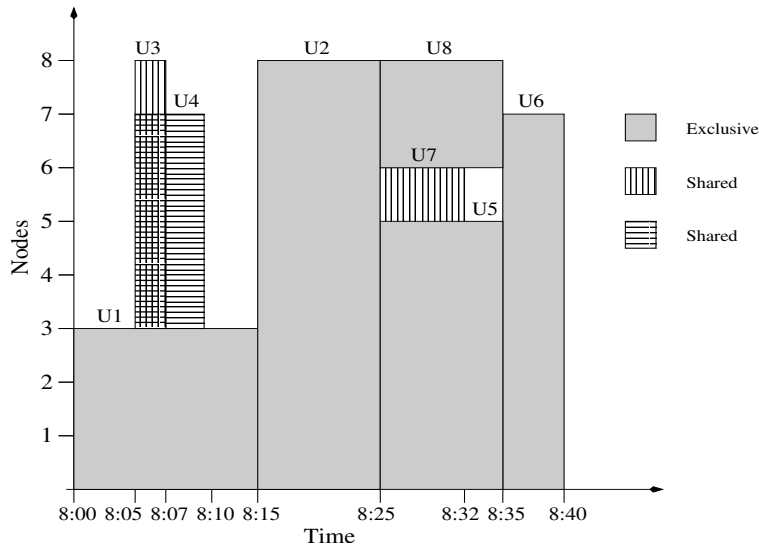


Figure 3: Queue using the CRONO scheduler.

tions on the node.

4.4.1 Access Control

Sometimes users are only debugging their programs and do not need the exclusive access to the allocated nodes and sometimes they need exclusive access to make some performance analysis. Therefore, it's necessary to provide space-sharing and time-sharing allocations to make a better use of the available resources. To control the access to the nodes in both cases, CRONO modifies the `login.access` and `hosts.equiv` files when a user allocation time starts and when it finishes. When a login process is done on a node, the `login.access` file is verified to allow or to block the user access. However, modifying the `login.access` is not enough to protect the access on a node because the users can execute the Remote Shell client (`rsh`) to execute commands on the nodes without the complete login process. In this case the `login.access` file is ignored. To solve this problem, it's also necessary to modify the `hosts.equiv` file on the node and use the option to ignore the `.rhosts` file in the home user directory when starting the Remote Shell server.

4.4.2 Execution of Operations

The administrator can define a set of operations which can be executed by the users through the `crnmc` command. The user can execute the operation on all allowed nodes or a group of define nodes. An example of such operation could be killing the processes on the nodes, installing a kernel module, or some other operation only done by the administrator.

5 Crono Installation and Configuration

When implementing a cluster management system, a difficult issue is to provide a good level of configurability without increasing the complexity of the installation and configuration procedures. One of the major advantages of CRONO is the simplicity of these procedures maintaining a good level of configurability. To show how simple is to install and configure CRONO, we describe in the following section these procedures for a typical environment with two clusters (two of the three CPAD [9] clusters from Figure 4, the 16 nodes cluster amazonia and the 4 nodes cluster pantanal). The system modules for the example environment are also showed in Figure 4. For this case we will have one Access Manager running on the frontend, two Requests Managers (one for each cluster) and one Node Manager on each node of the clusters. The following items describe the configuration and installation steps for this example:

1. Download the file from `http://www.cpad.pucrs.br/crono/crono-vXXX.tar.gz`, where XXX is the CRONO version;
2. Unpacking the tar file should produce a single directory called `crono-XXX`, containing the GNU General Public Licence, Changelog, INSTALL files, and the `bin`, `docs`, `etc`, `sbin` and `src` directories;
3. Before the compilation, some default configurations can be modified in the `src/misc/misc.h`. Adjust them as necessary. To compile CRONO change to the `src` directory, type `make` to build and `make install` to install it;
4. After that, you will have the following programs:
The User Interface programs in the `bin` directory. The Access Manager (`cramd`), Request Manager (`crrmd`) and Node Manager (`crnmd`);
5. Assuming that CRONO was installed in the `/usr/local/crono` directory, the configuration files will be in `/usr/local/crono/etc/` directory, and we'll call it DIRCONF. The DIRCONF structure is:

```
<dirconf>/  
amconf - this file contains the hostname and port number of the Access Manager  
<cluster1>/ - configuration files for cluster1  
<cluster2>/ - configuration files for cluster2  
...  
...
```

For each cluster directory we have nine files, which will be explained in the following items.

6. In the `config` file are defined the server ports and hostnames, the `share` option, which defines the number of users that can share a node at the same time, the `single` option, which ignores the function calls that communicate with the

Node Manager (useful for test purposes, when a functional cluster is not available). File locations of queue, log and reserve must also be defined in this file. An example of config file could be:

```
#####
# CLUSTER: amazonia
#####
# Server ports
amport=7001
rmport=7022
nmport=7004
# Server hosts
amhost=frontend
rmhost=frontend
# Share Option
share=4
# Single Option
single=off
# Log File
logfile=/usr/local/crono/var/log/crono_amazonia.log
# Queue File
queuefile=/usr/local/crono/var/queues/crono_amazonia.queue
# Reserves File
reservesfile=/usr/local/crono/var/queues/crono_amazonia.reserves
#####
```

7. The groups, priorities and users files were described in section 4.1;
8. The nodes file has the hostnames of the cluster nodes. The login.access and hosts.equiv files are the original files of the Linux system; The mpreps and mposts files are the Master Pre-Processing Script and Post-Processing Script respectively. Our mpreps has programs to create the user machine file for MPI environments:

```
#####
#!/bin/sh
# Master Pre-Processing Script
# $1=<user id>
# $2=<cluster name>
/usr/local/crono/bin/crmachmpiether -u $1 -c $2
/usr/local/crono/bin/crmachmpigm -u $1 -c $2
#####
```

9. Finally we have the commands file, which defines the operation that users can execute using the crnmc command. In our environment, we have defined only the killp operation to kill the user processes:

```
#####
# Commands file
# $1=<user id>
killp: skill -KILL -u $1
/usr/local/crono/bin/crmachmpigm -u $1 -c $2
#####
```

10. After setting up the configuration files, just start the daemons. In frontend machine start the one cramd and two crrmd:


```
$cramd amazonia pantanal
$crrmd amazonia
```

```

$crnmd pantanal
On the amazonia nodes:
$crnmd amazonia
On the pantanal nodes:
$crnmd pantanal

```

6 Using Crono in Production Mode

For the last 6 months CRONO is being used to manage the three Linux clusters in our research lab. Access to the clusters is centralized through one host machine that runs one Access Manager and three Requisition managers, one for each of the clusters. Each node of the clusters has its own Node Manager. Each cluster has its own peculiarities, having a different number of nodes (4, 16, and 32) and different interconnection networks (combinations of SCI, Fast-Ethernet and Myrinet) as shown in Figure 4.

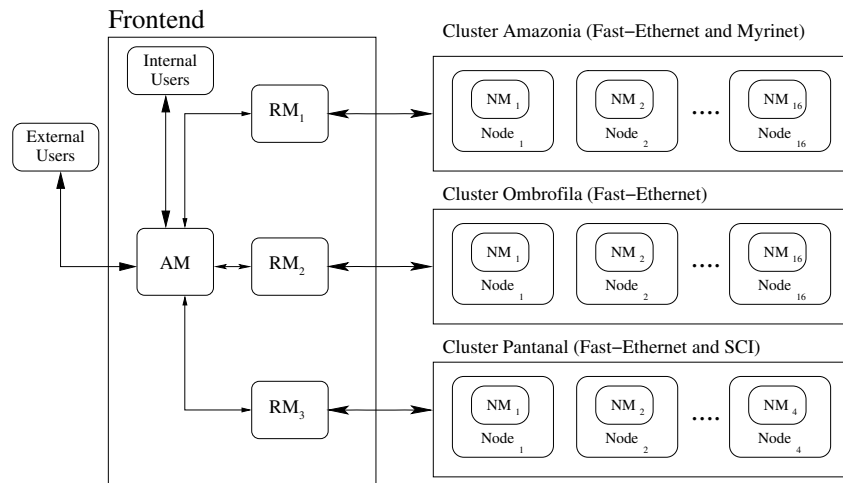


Figure 4: CRONO modules on CPAD.

We also support several program environments, including different MPI implementations and some own developed execution kernels. With simple commands like `cralloc`, `crcomp` and `crun` users can choose the cluster, the number of nodes, the target interconnection network and the programming environment they want. CRONO will automatically generate machine files, choose the right libraries for the compilation, load the processes and start the program. Because we have different types of users, access rights were defined for each user depending on status, project, target machine and day period separately. A student may have, for example, only access to the small cluster during the peak hours and to 16 nodes of the main cluster during the night. The system is running over the Linux Slackware 8 distribution and is already very stable. We developed the system in a way that it can be easily con-

figured to work with other cluster configurations and a different set of tools. More information about Crono can be found at www.cpad.pucrs.br/crono together with its source code that is distributed under the GNU license.

7 Conclusion

In this paper we described the functionality and the architecture of a new open source cluster management system called CRONO. CRONO was developed in our research lab to be a highly configurable system that supports mid-size Linux clusters with different configurations and network interconnections. Being used for the last 6 months in production mode CRONO is already very stable and is being updated constantly with patches and new features. We believe CRONO is already a very nice alternative to more complex systems like CCS, DQS and PBS for small research labs, like ours particularly, when the main focus is on controlling the access privileges of different machines and groups of users individually with a low maintenance cost.

References

- [1] Buyya, R., *High Performance Cluster Computing*. Prentice-Hall, 1999.
- [2] Keller, A. & Reinefeld, A., Anatomy of a resource management system for hpc clusters. *Annual Review of Scalable Computing*, **3**, 2001.
- [3] T. P. Green, J.S., Dqs, a distributed queueing system. *Annual Review of Scalable Computing*, 1993.
- [4] OpenPBS. <http://www.openpbs.com/>.
- [5] Gropp, W., Lusk, E., Doss, N. & Skjellum, A., A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, **22(6)**, pp. 789–828, 1996.
- [6] Stevens, W.R., *UNIX Network Programming Vol 1: Networking APIs - Sockets and XTI*. Prentice-Hall, 2nd edition, 1997.
- [7] Boden, N.J., Cohen, D., Felderman, R.E., Kulawik, A.E., Seitz, C.L., Seizovic, J.N. & Su, W.K., Myrinet: A gigabit-per-second local area network. *IEEE Micro*, **15(1)**, pp. 29–36, 1995.
- [8] IEEE standart 1596-1992, New York, *IEEE: IEEE Standart for Scalable Coherent Interface (SCI)*, 1993.
- [9] CPAD - Research Center in High Performance Computing. <http://www.cpad.pucrs.br>, 2001.