Improving Performance Analysis Using Resource Management Information*

Tiago C. Ferreto
Research Center in
High Performance Computing
CPAD-PUCRS/HP
Porto Alegre, Brazil
ferreto@cpad.pucrs.br

César A. F. De Rose
Catholic University of
Rio Grande do Sul
Computer Science Department
Porto Alegre, Brazil
derose@inf.pucrs.br

May 9, 2003

Abstract

In this paper we present *Clane*, an utilization and performance analysis environment for clusters. It combines resource management and monitoring data to provide reliable information for cluster users and administrators in application and system performance analysis. *Clane* uses the XML standard to represent its internal information base, providing more flexibility in data manipulation and simplicity to extend the environment with other analysis tools. The environment is composed by an *Information Server*, which gathers and combines information from the resource management and monitoring systems, and an *Analysis Tool* to present the combined information using statistics, graphs and diagrams. The *Analysis Tool* also enables a performance comparison among several executions of the same application in the cluster.

*This research was done in cooperation with HP-Brazil

1 Introduction

Resource management and monitoring systems are among the basic services available in a cluster environment. They are usually implemented as separated tools and for different purposes. Resource management is usually directly related to the cluster user allowing him to get access to a group of cluster nodes, whereas the monitoring system is designed for the cluster administrator to get an idea on how well the machine resources are being exploited by the end users. Some resource management systems provide also a basic monitoring service, but the information is usually presented in real-time and without any details of the applications that are running on each node. This view is well suited to the cluster administrator to evaluate the global performance of the machine, but not to the end user to evaluate the performance of his application.

Believing that a stronger integration of resource management and monitoring tools could lead to a better system and application analysis we developed *Clane*, an analysis environment for clusters. *Clane* combines resource management and monitoring data to provide performance information related to the applications that are executed in a cluster. The acquired information is formatted to better suit the needs of system administrators as well as end users. This approach differs from system monitoring that is provided by some resource management systems because the information is stored for future analysis (post-mortem monitoring) enabling comparisons among several runs of an user application, evaluation of the system performance impact of a change in the machine configuration, and other useful performance analysis.

2 Resource Management and Monitoring

Resource management systems are responsible for distributing applications among computers to maximize their throughput. It also enables the effective and efficient utilization of the available resources [1].

Some of the main functionalities that should be provided by resource management systems for clusters are: resources allocation and freeing, application execution using the allocated resources through interactive or batch jobs, and allocation queue management using an allocation policy. Other functionalities provided by the resource management system are load balancing, process migration, support to many allocation policies, allocation queue management based on priorities, suspension and restart of applications, etc.

There are several cluster resource management systems freely available. Some of the most used resource management systems are OpenPBS [2], and CCS [3].

OpenPBS is a flexible batch queuing and workload management system originally developed by Veridian Systems for NASA. It operates on networked, multi-platform Unix environments, including heterogeneous clusters of workstations, supercomputers, and massively parallel systems. The purpose of the PBS system is to provide additional controls over initiating or scheduling execution of batch jobs; and to allow routing of those jobs between different hosts. The batch system allows a site to define what types of resources and how many resources can be used by different jobs. Some of the main features of OpenPBS are configurable job priority, transparent job scheduling, easy integration with other applications through a comprehensive API, and automated load-leveling based on HW configuration, resource availability and keyboard activity. OpenPBS supports real time system monitoring at application level through an internal tool called *xpbsmon*.

CCS is a resource management system developed at the PC^2 (Paderborn Center for Parallel Computing) at the University of Paderborn. It has been designed for the user-friendly access and system administration of parallel high-performance computers and clusters. It supports a large number of software and hardware platforms and provides a homogeneous, vendor independent user interface. For system administrators, CCS provides mechanisms for specifying, organizing and managing various high-performance systems that are operated in a computing service center. Robustness, portability, extensibility, and the efficient support of space sharing systems, have been among the most important design criteria. CCS provides system and application monitoring

integrated to its environment through the *ccsMon* and *SPROF* tools.

Monitoring systems are designed to collect system performance parameters such as node's CPU utilization, memory usage, I/O and interrupts rate, and present them in a form that can be easily understood by the system administrator [4]. This service is important for the stable operation of large clusters because it allows the system administrator to spot potential problems earlier. Moreover, other parts of the systems software can also benefit from the information provided. For example, the information can be used to modify the task scheduling, in order to improve load balancing. Some of the main characteristics that should be provided by the monitoring tools are:

Low intrusion One of the main issues considered in the project and implementation of monitoring systems is the intrusion generated by the monitoring system. To obtain the highest possible performance in a cluster, the parallel application should be able to get all of the available processing power. However, the monitoring system has some processing and communication costs and it will compete with the running application. Therefore, to enable high performance execution in the presence of monitoring, the monitoring tool should have low intrusion, producing minimal interference.

High configurable Due to the diversity of cluster topologies, and the increase of heterogeneous clusters, monitoring systems need to be high configurable. Some of the features that should be provided by monitoring systems are: adjustable monitoring frequency, selection of monitoring metrics, selection of resources to be monitored, online and offline monitoring, etc.

Extensibility Extensibility is a critical issue for any monitoring system survival. The extension of the system to support new metrics is really necessary to enable information gathering of new hardware technologies (e.g. Myrinet [5] and SCI [6] network boards). This feature is also useful to provide interconnection between the monitoring system and other systems, in which the last uses the information obtained by the monitoring system to execute some task (e.g. a load balancing system), or just to present it (e.g. a graphical tool).

There are several cluster monitoring systems freely available. Some of the most used are PCP [7], Ganglia [8].

Performance Co-Pilot (PCP) was developed by SGI as a commercial monitoring system for IRIX. In February 2000, SGI released the PCP infrastructure as Open Source software. Their goal was to provide a readily available, feature-rich and extensible framework for managing performance in large Open Source systems. PCP is composed of a framework and services to support system-level performance monitoring and performance management. It provides a range of services that may be used to monitor and manage system performance. Some of the main features provided by PCP include public interfaces and extensible frameworks at all levels, integrated logging (real-time) and retrospective (historical) data analysis support, single API for all performance data, and coverage of a large range of activity areas (e.g. CPU, memory, Cisco routers, Web servers).

Ganglia is a monitoring environment initially developed at the University of California, Berkeley Computer Science Division as a way to link clusters across the Berkeley campus together in a logical way. Since it was developed at a university, it is completely open-source and has no proprietary components. All data is exchanged in well-defined XML and XDR to ensure maximum extensibility and portability. Ganglia provides a complete real-time monitoring and execution environment that is in use by hundreds of universities, private and government laboratories and commercial cluster implementations around the world.

3 Clane: an analysis environment for clusters

Clane provides a complete analysis environment with generic interfaces and a graphical analysis tool, which presents information based on cluster resources allocation, applications execution, and performance information acquired through monitoring. The environment presents a novel approach combining system monitoring with resource allocation and execution events, providing

a different analysis view of the information.

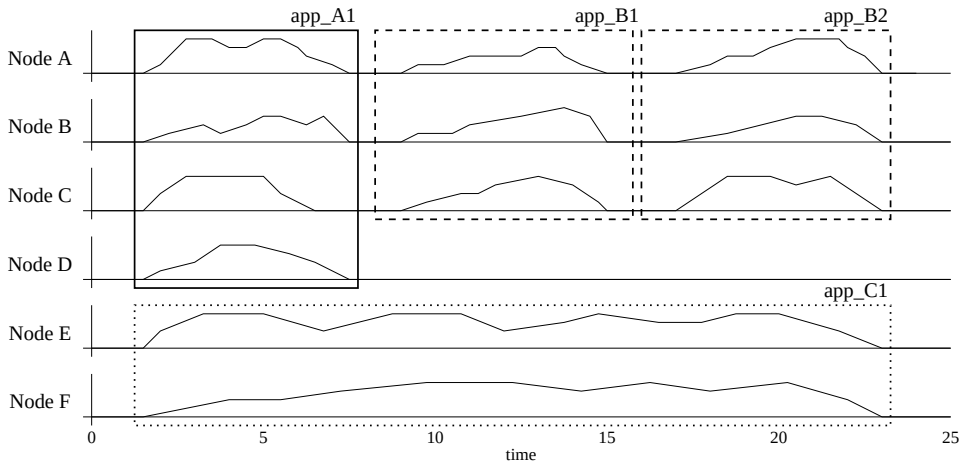
3.1 Motivation

Resource management and monitoring systems are most commonly used in clusters as two separated systems. This fact is clear by the increasing number of specialized cluster monitoring systems, and the few number of resource management systems that provide also system monitoring. Some resource management systems try to integrate a specialized monitoring system in its architecture to enable other functionalities (e.g. load balancing), but this integration is usually not easy and just the basic functionalities of the monitoring system are used. In most of the cases only online monitoring is used, the metrics monitored or the monitoring frequency can't be changed, and the information is gathered from all cluster nodes.

In the other hand specialized cluster monitoring systems provides online and offline information with a collection of metrics and configurable capture frequencies. These systems are really useful to provide a global view of the system during the time. However, the acquired information is not related to the applications being executed in the cluster.

We believe that the correlation of monitoring information and allocation/execution events, which are controlled by the resource management, are useful for cluster administrators and users. The information resultant of this correlation can be divided for each user, and internally for each application executed in the cluster, as presented in Figure 1. Figure 1(a) presents monitoring information for a group of nodes. Each rectangle drawn represents an individual application execution. Figure 1(b) presents the allocation and execution table for the monitored period. The rectangles drawn around the user names are correlated to the rectangles drawn in (a). This information can be used in application and cluster utilization analysis. Some of the benefits achieved by this approach are: it optimizes the utilization of cluster resources by the applications, detects possible bottlenecks in the cluster that can be minimized, and compares several runs of an application executed in the cluster.

(a)



(b)

Allocation and Execution Table

User	Time Period	Allocated Nodes	Executed Application
user_A	1 − 8	A, B, C, D	app_A1
user_B	8 − 16	A, B, C	app_B1
user_B	16 − 24	A, B, C	app_B2
user_C	1 − 24	E, F	app_C1

Figure 1: Correlation between resource management events and monitoring information. (a) Monitoring information. (b) Allocation and execution table.

3.2 Architecture

The architecture of *Clane* is presented in Figure 2. The environment is highly connected to the resource management and monitoring systems. This connection is established using functions provided by the *Information Storage Interface* which gathers the information and forwards it to be stored by the *Information Server*. The information is stored in XML providing simplicity and flexibility in data manipulation. The *Analysis Tool* access this information using the functions available in the *Information Access Interface*. These functions allow the selection of fragments from the total amount of stored information. The result of this selection is stored in an XML file. After the selection, the *Analysis Tool* is able to process the resulting XML file and present the information. The *analysis tool* provides different formats to present the information, facilitating

the analysis process.

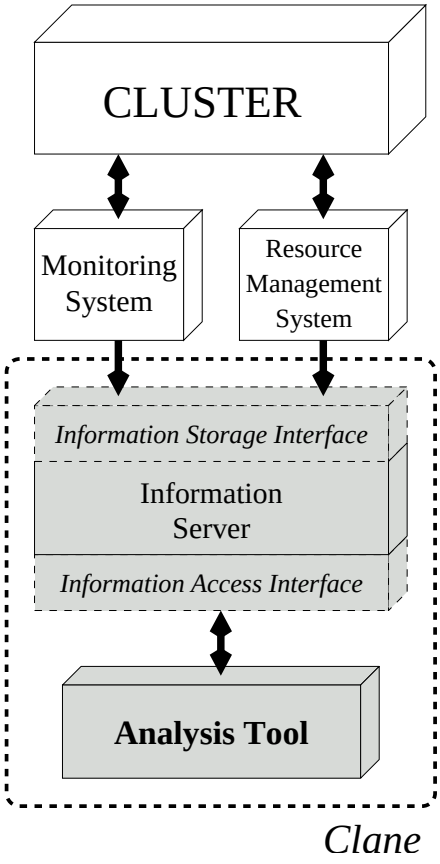


Figure 2: Clane architecture

3.3 Implementation

3.3.1 Information Server

The *Information Server* is responsible for storing allocation, execution and monitoring information, and to provide access to it. All information is stored in XML format due to its high flexibility for storing and accessing the information and to provide extensibility and portability.

The information is obtained using the resource management and monitoring systems with the functions available in the *Information Storage Interface*. These functions are divided in three

groups, one to store information related to the cluster allocation, another to store information about the applications executed during the allocation, and the last one to store system status information.

Functions *is_initalloc()* and *is_endalloc()* records respectively information about the start and end of an user allocation, such as: allocation identifier, username, nodes allocated, timestamp, etc. Functions *is_initexec()* and *is_endexec()* stores respectively information about the start and end of an application execution, such as: execution identifier, the allocation identifier related to this execution, timestamp, application name and parameters, application returned status, etc. Function *is_monentry()* records information about the each node of the cluster, such as percentage of CPU, memory and swap utilized, percentage of L1 cache misses, etc. These functions are all available in a shared library (written in C) and as individual programs, called *clane_addallocation*, *clane_addexecution*, and *clane_addmonentry*.

Each allocation or execution event started but not finished is mapped internally in the *Information Server* data structure. Each event has a unique identifier which serves as an index for a specific entry in this internal structure. The allocation or execution event is written to the appropriate XML file just after being finalized, and after this, its entry in the structure is removed.

To store all received information, the *Information Server* generates at least two files, one for information related to resource allocation and applications execution, and other one for monitoring information. The location of these files can be configured in the environment. It is also possible to create an allocation and execution file for each user, minimizing the overhead of multiple accesses to the same file.

The *Information Access Interface* provides functions to extract a subset of the information monitored, and store it in an XML file. This information selection is implemented by creating a file using the XSLT [9] and XPath [10] languages specifying the chosen filters. This file is applied to the allocation and monitoring XML files using an XSLT processor to generate the resultant XML file, which contains the selected information.

The *Information Access Interface* provides a shared library, with specific functions to filter the

XML files, and also a program, called *clane_query* which receives as parameters a group of filters used to provide to the user the subset of information desired. The filters can select information based on a specific user, application and period of time.

3.3.2 Analysis Tool

The *Analysis Tool*, presented in Figure 3 is written in Java due to its design and programming simplicity, code portability, and XML support. Each time the *Analysis Tool* is executed, the user needs to specify the filters that should be applied to the information stored by the *Information Server*. The tool uses these filters to generate the resulting XML file with the information selected using the functions of the shared library provided by the *Information Access Interface*. After this, the tool parses the XML file, presents the collection of allocations selected, and keeps waiting for user interaction. The main features of the *Analysis Tool* include statistics for one or more allocations and application executions, graphs representing resource utilization for allocations and executions, and textual or graphical comparison between a collection of allocations or application executions.

Figure 3 presents a typical view of the analysis tool in use. The window in the back, the *Allocations Window*, shows the allocations selected initially. The user selects one of these allocations and requests for the executions realized during this allocation, which appear in the *Executions Window*. After this, one of these executions is selected, and the monitoring information is displayed based on the monitored metrics, in this case, percentage of CPU, memory and swap utilization. *Clane* also provides some useful statistics to the user based on the information monitored. These statistics are divided in allocation statistics and execution statistics. The main allocation statistics provided include average allocation time and standard deviation for a group of selected allocations, average number of applications executed for a group of selected allocations, average number of nodes used for a group of selected allocations, allocation time for one allocation, number of nodes used in one allocation, and number of applications executed in one allocation. The main execution statistics

Figure 3: Clane Analysis Tool

provided include average execution time and standard deviation for a group of selected executions, average and standard deviation of a metric monitored for a group of selected executions, and average and standard deviation of a metric monitored for one execution.

The comparison between allocations or executions is based on the information and statistics generated for these events. The comparison can be presented textually or graphically. The *Analysis Tool* can be used by the cluster administrator or user. The only difference between the functionality of the tool for each one is based on the amount of information that is available for analysis. The administrator can view information of all users, whereas the user can just see information of its own allocations and executions.

4 Configuration

Clane was configured at our lab, the Research Center of High Performance Computing (CPAD-PUCRS/HP) [11]. The environment was connected to Crono [12] and RVision [13], a resource management system and a monitoring system respectively. These systems where developed in our lab and are in production in all CPAD clusters. The inclusion of the functions and programs provided by the *Information Storage Interface* to Crono and RVision was relatively simple, due to the high configurability and flexibility presented in these systems.

Crono executes a script for each started allocation (*mpreps*) and another one for each finished allocation (*mpostps*). These scripts are used to set necessary configurations before user interaction. The program to log allocation events, *clane_addallocation*, was inserted in these scripts to log information related to resources allocation.

To simplify the utilization of the system, Crono provides an unique application dispatcher, called *crrun*, which encapsulates all execution environments available to the users. This script receives the execution parameters and forwards it to the real execution environment which is responsible for the application execution. The program *clane_addexecution* was included in this script before and after calling the execution environment to log the information related to the application execution.

The connection between RVision and Clane was implemented with a special RVision monitoring client. This client calls the function *is_monentry()*, provided by the *Information Storage Interface*. This monitoring client was implemented as a daemon, due to its unique functionality of gathering information and sending it to the *Information Server*, instead of showing it.

The monitoring client was implemented with an internal area of memory, which is used to buffer the information received from the core of the monitoring system (*RVCore* module). This method was used to decrease the number of file openings and writings, therefore, the information is written periodically (each 30 seconds) in larger blocks.

RVision divides the information gathering in monitoring sessions. Each monitoring session is responsible for capturing a defined set of information from a group of nodes in a given periodicity. Therefore, each allocation starts a new monitoring session for the group of nodes allocated. The same approach used to log the allocation starting and ending events was used to start a new monitoring session and finishing it. The calls to start the monitoring client and to finish it were included respectively in the *mpreps* and *mpostps* scripts provided by Crono.

The metrics defined to be monitored are percentage of CPU utilization, percentage of memory utilization, and percentage of swap utilization. These metrics are monitored each 2 seconds on each node and sent to the *RVCore* module based on a threshold of 2%, i.e. if the information gathered is 2% higher or smaller than the last one captured, then it is sent to the *RVCore* module, otherwise, the information is rejected. This method decreases the amount of data transmitted through the network (reducing the intrusion), and decreasing also the total amount of data stored.

The *Clane* environment was configured to generate an allocation and execution record file for each user, instead of a general one. This approach was chosen based on its better division of the data, simplifying the storing and parsing process. Since all information is date and time based, the network time protocol (NTP) [14] was configured to realize the clock synchronization in all machines, to allow a correct interpretation of the captured information.

4.1 Utilization

The *Analysis Tool* is used by the users mostly to compare the performance obtained for a specific application with different parameters. Since the tool generates graphs showing the performance and also statistics for each metric measured, it is possible to optimize the application based on this information. The tool is used by the administrator to visualize the percentage of resources utilization by each user. It is also used to determine the periods of the highest and lowest utilization.

In addition to the features provided by the *Analysis Tool*, a *Report Generator* was implemented. This tool uses the functions of the *Information Access Interface* to obtain information about users

allocation, executions and obtained performance. It was configured at CPAD to execute the following tasks: generate a weekly report based on users utilization (hours of utilization, number of machines); send weekly to each user an email containing the amount of hours allocated, the number of nodes, and the 10 best and worst executions based on the performance obtained through monitoring; and send monthly to the administrator (via mail) the periods of highest and lowest utilization of the clusters, and the name of the users that have most used the cluster.

5 Conclusions and Future Work

In this paper we presented a new approach to enhance system and application monitoring in cluster architectures based on the utilization of resource management information. Traditional monitoring data like processor and memory usage is combined with information about executed jobs like start and finishing times and number of allocated resources allowing the analysis among several runs of the same application. This is especially useful to generate statistics about resource use and also for application tuning.

The proposed architecture, called *Clane*, has defined interfaces in both ends to allow the adaptation to other monitoring and resource management system that rely on XML for data storage allowing a broader utilization of the system.

To investigate this new concept we implemented a prototype of the proposed architecture and linked together two tools that are already in production in our lab, the resource manager Crono and the monitoring system RVision. The initial results are promising and we are now studying the implementation of data compression and the support for more resource management systems and monitoring systems through the utilization of patches.

We believe that the correlation of monitoring and resource management information will allow administrators and users to better analyze the behavior of their systems and the executed parallel applications opening a new range of opportunities for the enhancement of monitoring tools.

References

[1] Rajkumar Buyya. *High Performance Cluster Computing: Architectures and Systems*, volume 1. Prentice-Hall, 1999.

[2] R. L. Henderson et al. Portable batch system: Requirement specification. Technical report, NASA Ames Research Center, April 1995.

[3] Axel Keller and Alexander Reinefeld. Anatomy of a Resource Management System for HPC Clusters. *Annual Review of Scalable Computing*, 3, 2001.

[4] M. Baker. Cluster computing white paper, 2000.

[5] C. L. Seitz et al. Myrinet - a gigabit-per-second local-area network. *IEEE Micro*, vol. 15(1), 1995.

[6] IEEE standart 1596-1992, New York. *IEEE: IEEE Standart for Scalable Coherent Interface (SCI)*, 1993.

[7] Mark Goodwin et al. *Performance Co-Pilot User's and Administrator's Guide*. Silicon Graphics, Inc., July 1999.

[8] Ganglia Development Team. *Ganglia Toolkit*. University of California, Berkeley, 2002. http://ganglia.sourceforge.net/docs/.

[9] World Wide Web Consortium (W3C). XSL Transformations (XSLT) Version 1.0 (W3C Recommendation). Technical report, W3C, November 1999.

[10] World Wide Web Consortium (W3C). XML Path Language (XPath) Version 1.0 (W3C Recommendation). Technical report, W3C, November 1999.

[11] CPAD Research Center. CPAD-PUCRS/HP. http://www.cpad.pucrs.br, 2003.

[12] Marco Aurélio Netto and César De Rose. Crono: A configurable management system for linux clusters. In *Proceedings of the 3rd LCI International Conference on Linux Clusters: The HPC Revolution 2002 (LCI'2002)*, St. Petersburg, Florida ,USA, 2002.

[13] Tiago Ferreto, César De Rose, and Luiz DeRose. Rvision: An open and high configurable tool for cluster monitoring. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluter Computing and the Grid (CCGrid'2002)*, pages 75–82, Berlin, Germany, 2002.

[14] D. L. Mills. Internet time synchronization: the network time protocol. *IEEE Trans. Communications*, (39):1482–1493, October 1991.