An Architecture for Remote Monitoring-Management of Distributed Applications

A. Sanna† **T. Stefanuto**‡ **C. Zunino**† **F. Lamberti**† **P. Montuschi**†

Dipartimento di Automatica e Informatica,
Politecnico di Torino, corso Duca degli Abruzzi 24,
I-10129 Torino (Italy)

†{sanna,c.zunino,lamberti,montuschi}@polito.it, **‡tstefa@libero.it**

ABSTRACT

Computationally intensive applications often require the employment of parallel/distributed solutions in order to strongly reduce execution times. Workstations and PCs connected by a fast local network provide effective, low-cost, and general purpose solutions that are quickly replacing special purpose proprietary architectures for a wide spectrum of disciplines. The issue to monitor the performance (fundamental to check the efficiency of a distributed application) of cluster architectures is addressed in this work. This paper presents a multi-channel architecture for remote monitoring-management tailored to display information on PDA devices. Users can remotely monitor the system critical activities and cluster resources utilization; moreover, we provide the possibility to perform some management tasks (for instance, shutdown and reboot) useful for maintaining the integrity of the system and to harness the full cluster potential.

Keywords: Information visualization, cluster monitoring, remote system management, distributed and parallel applications, PDA.

1. INTRODUCTION

The term information visualization denotes the capability to graphically represent a large number of items, possibly retrieved from large datasets, in order to allow the user to quickly identify patterns, groups of items, or individual items. This can be useful in a wide spectrum of disciplines where a tabular and chart representations could be not suitable.

This paper tackles the issue to provide an effective and portable tool for monitoring cluster architectures based on Linux/Windows. Computationally intensive applications often require parallel/distributed platforms and cluster architectures proved to be excellent low-cost solutions. The efficiency of a distributed application strongly depends on resources utilization that can be assured only by a continuous real-time control. Proprietary hardware platforms provide ad-hoc software for resources monitoring-management but clusters often lack of these tools.

Two key issues have to be considered in the design of a monitoring software: the efficiency and the graphics interface. The term efficiency denotes the ability to gather information about resources without affecting system performance; for instance, a process demanded to monitor the CPU load has to require CPU cycles as less as possible but, on the other hand, it has been able to trace the processor activity in real-time. Another critical topic is data representation. Cluster architectures composed by tens/hundreds nodes involve a large amount of data and the visualization interface has to present information by means of icons able to enhance data comprehension.

Moreover, this work considers Personal Digital Assistants (PDAs) as target front-ends. PDAs are handheld computers that originally were designed as personal organizers; the basic features of any PDA are a date book, address book, task list, and memo pad. Nowadays PDAs are composed by a input unit that is a stylus for navigation and data entry via a handwriting recognition system. The typical display size is 160x200 pixels, even if some products reach 240x320 pixels of resolution. More expensive PDAs have a color display while the medium price devices have gray scaled displays.

The proposed application can be defined multi-channel as it provides monitoring capabilities to virtually any device. Although the graphics interface, entirely written in Java, is tailored for PDA displays, any wireless/wired device can be used. Moreover, the proposed solution provides some basic management features. Shutdown and reboot of a node can be performed as well as a process, on a selected node, can be killed.

The rest of the paper is organized as follows: Section 2 reviews main works concerning cluster monitoring, Section 3 describes the proposed architecture, while Sections 4, 5, and 6 present details of modules composing the whole system. Finally, some remarks on experimental tests can be found in Section 7.

2. BACKGROUND

Parallel/distributed system management and monitoring has already been addressed in many projects. In particular, some projects aimed to design platform-independent monitoring tools, mainly based on the use of the Java programming language. The most common approach consists in developing the statistics collection subsystem using platform-dependent applications while deploying a portable Java based end-user interface using either applets and/or applications.

Java Dynamic Management Kit (JDMK) [6] is one of the first Java-based tools for distributed system monitoring developed using the Java Programming Language. In JDMK each node acts as an individual information source and the access to the node must be explicit.

In CARD (Cluster Administration using Relational Databases) [2] data are gathered and stored in a relational database. New subsystems can access the data through SQL without requiring modifications of old programs. SQL is used both to execute ad-hoc queries over the database and to extract data for visualization in a Java applet-based end-user interface.

Monitoring and management problems of Beowulf clusters are tackled in SMILE [15][17]. The project SMILE [16] (Scalable Multicomputer Implementation using Low-cost Equipment) allows to gather information from every node in real-time, and monitoring and management can be achieved by using a set of APIs developed in C, Java, and Tcl/Tk.

Similar to SMILE is Berkeley NOW (Network Of Workstations) system administration tool [1] that gathers and stores data in a relational database. A Java applet provides the graphics interface allowing users to monitor the system from any Internet node.

GARDMON [3] is a Java-based monitoring tool for non-dedicated cluster computing systems. It follows client-server methodologies and provides transparent access to all monitored nodes from a single point of control, the monitoring server. GARDMON monitoring data can be only accessed by ad-hoc GARDMON clients, developed and implemented using Java computing technologies. The cluster-node monitoring server is developed using the C programming language.

PARMON [13] is a Java-based monitoring system allowing cluster resources monitoring at various levels: entire system, node and component level. PARMON architecture consists of the PARMON-client and the PARMON-server. The server, developed as a multi-threaded application using the C programming language, acts as system resource activities and utilization information provider. The client is a Java GUI-based application responsible for interacting with the PARMON-server and users for data gathering in real-time and presenting information graphically for visualization. The server is running on all nodes to be monitored and the communication between the client and the server is performed by message passing.

To ensure the portability of the whole monitoring architecture, both the measuring subsystem and the end-user interface could be deployed using the Java language.

Furthermore, a real multi-channel architecture is needed in order to allow the access to monitoring information to virtually any device. An example of a existing multi-platform monitoring tool entirely developed in Java is ClusterProbe [9]. By employing Java RMI APIs, this tool provides an open environment able to support multiple communication protocols and pre-formatting modules for platform-independent cluster resource related information retrieval and presentation.

SIMONE (Simple Network Management Protocol-based Monitoring System for Network Computing) [14] is a highly efficient modular monitoring system designed for heterogeneous wide-area computing systems that be used on grid architectures.

Recently, two multi-channel applications for cluster monitoring have been presented in [7] and [20]. Both are entirely based on the Java language and aim to provide platform-independent tools for remote monitoring. In particular, in [7] issues concerning GSM/GPRS and IEEE 802.11 connection of PDAs have been deeply analysed.

Some commercial system monitoring tools are also available. The HP OpenView Performance manager, agents, and monitor [11] combine to provide a powerful and flexible multiplatform distributed management solution. This solution is a single interface for centrally monitoring, analysing, and forecasting resources utilization for distributed environments. Piranha [12] is an interesting solution for WAP-enabled cell phones and PDAs. Piranha provides real-time system-status reports and also allows to make maintenance and configuration adjustments.

Currently, Piranha supports readout of system hostname, uptime, average load, number of users, and detailed memory information. Maintenance tasking capabilities allow a user to restart both a web server and an entire computer. XMON [19] is a powerful system monitoring tool from Fujitsu running on XSP operating system; it is able to monitor system activities and reports information in real-time by providing an immediate view of system activities and rapid identification of system problems. Through a user friendly interface, channels, peripherals, real memory, CPU cycles, page datasets, and I/O operations can be monitored in order to provide a complete picture of system performance.

3. THE PROPOSED ARCHITECTURE

The proposed architecture is composed by three different components:

- A client application able to retrieve information running on every machine to be monitored (described in details in Section 4).
- A server application that gathers data coming from clients and stores them into a database (described in details in Section 5).
- A visualization application that reads data from database and presents resource status in a graphics way (described in details in Section 6). This application allows a remote monitoring via web and, above all, it is tailored to be used for PDA devices. Moreover, the user can perform a set of management tasks useful for maintaining the integrity of the system and to harness the full cluster potential.

The whole architecture is shown in Figure 1. The cluster is a set of hosts running Linux or Windows NT/2000/XP connected by a (fast) LAN where client applications run.

The server application, generally, runs on a machine not belonging to the cluster; on the same machine the database server is installed (in this way, performance to be monitored are minimally affected by the measurement system). Monitoring and management tasks can be remotely delivered both via web by wired front-ends and via wireless mobile devices.

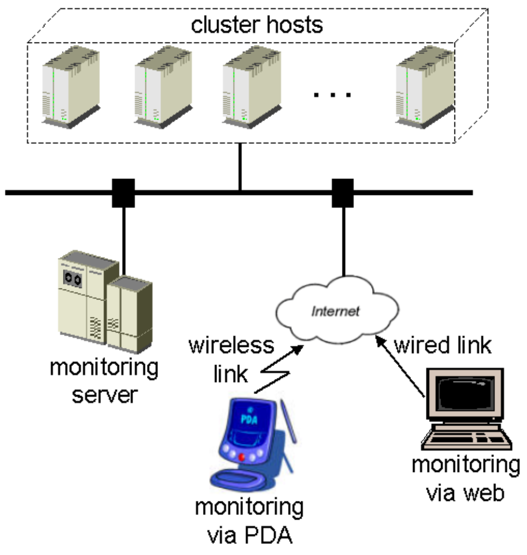

Figure 1 The whole architecture.

4. CLIENT APPLICATION

The client application runs on each machine to be monitored. The efficiency is a key issue for this part of the monitoring system as it has to affect the performance of the machine under analysis as less as possible in terms of CPU and memory used. For this reason the client has been written in C language.
The application can run with normal user privileges, but management tasks can be performed only running as root/administrator user.
Ad-hoc timers have been implemented in order to periodically check a set of parameters: CPU usage (up to four CPUs can be managed), memory occupation, file system space, network bandwidth occupation, users logged, and number of processes currently in the system.
The client transmits information to the server via TCP/IP. A socket connection is continuously kept open between client and server; if a loss of connection occurs, the client automatically attempts to reconnect itself to the server for a preset number of times.
Communications between client and server are performed by a tag language similar to HTML. Each tag can be opened and closed; for instance, the tag <net> indicates the beginning of information concerning network bandwidth occupation, while the tag </net> denotes the end of this kind of transmission. The server inserts data in the database only when end tags are encountered. This approach ensures data integrity and lower LOCK times of the database.

5. SERVER APPLICATION

The server application gathers data coming from clients and store them into a database. Two choices are possible: a distributed database on every client, a central database possibly running on a machine different from the cluster ones. Although critical for fault-tolerance, the second option has been chosen in order to minimally affect cluster performance.
Moreover, the server has been designed and implemented in a platform-independent way. It has been written in Java language [10] and MySQL [18] has been chosen as database

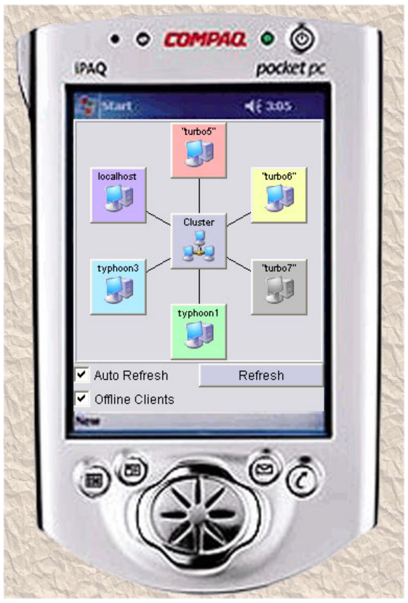

Figure 2 Cluster visualization.

server (it is available both for Unix-like platforms and Windows).
The server has a double "interface". From one side it interacts with clients, from the other side it communicates with the visualization application. Every data received from a client is parsed and inserted into the database by means of JDBC APIs [4]. The server manages concurrent accesses to the database that can occur when the visualization application attempts to read data and a client is transmitting. Finally, the server receives management commands (shutdown, reboot, kill of a selected process) and transmits them to the right client.

6. VISUALIZATION APPLICATION

Portability has been identified as the key issue for the design of the visualization interface in order to implement a multi-channel application. The visualization interface has been implemented by a Java applet that can be displayed by any compatible web browser. Design choices have particularly considered small size displays as target visualization devices, for this work, are PDAs.
The starting page allows the user to get a global view of the monitored cluster. All machines are placed around a central button (reserved for whole cluster statistics) in an elliptical way in order to maximize the number of buttons. Machines are denoted either by their names or by IP addresses and are identified by a color; the gray color is reserved for off-line nodes. Off-line nodes are visualized if the corresponding checkbox is selected. The auto refresh check box set an automatic refresh rate; on the other hand, manual refreshes can be activated. An example of the starting page is shown in
Figure **2** where a cluster of six machines (one off-line, turbo7) is depicted.
Tapping/clicking on a machine, the user can obtain detailed information about the status of a node. In particular ten buttons are presented:

- Host information: symbolic name, IP address, an internal ID code for the database, and status (on-line/active or off-line/inactive).
- CPU: a bar chart shows the percentage occupation; the color of the bar smoothly changes from green to

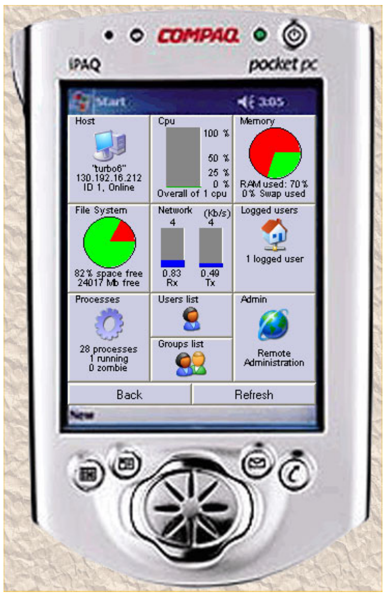

Figure 3 Visualization of a single node resources.

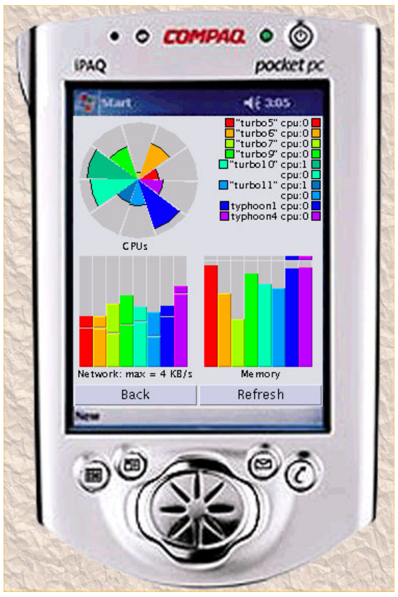

Figure 4 Characteristics of the whole cluster.

- red according to the bar height. Moreover, a label indicates the number of processors.
- Memory: a pie chart shows RAM occupation and a label indicates the percentage of swap space used.
- File system: a pie chart shows the whole status of all disk partitions.
- Network: two bar charts adaptively show incoming and out coming network traffic.
- Logged users
- The list of processes currently in the system
- User and group lists
- Connection to the administration menu

An example is shown in Figure 3. Each time an excessive or anomalous use of a resource occurs, the corresponding label changes color becoming red. Moreover, each button can be further tapped/clicked in order to achieve detailed information about the selected resource (for instance, the button Processes leads to a page where: occupation of CPU and memory, process status, owner user, and so on are listed). As the display of a PDA allows to visualize a reduced number of text lines, a search function for selective queries has been implemented.

The possibility to remotely perform management operation is extremely worthwhile. The user can shutdown/reboot a node of the cluster as well as kill a selected process. Java applet and server application communicate by a socket; the applet sends to server application the ID of the destination machine and the command to be executed. Finally, the client application returns an acknowledge message to the server. Moreover, more administrator operations can be performed from different front-ends; when a management operation is executed, a message appears on the applet of each administrator.

The user can chose to have a global view of cluster resource by selecting the central button (Cluster) in the starting page. The visualization application displays main statistics concerning the entire cluster: CPU loads, memory, and network occupations. The task to represent the whole characteristics is one of the major challenges as a lot of information has to be presented on reduced size displays. On the other hand the user has to be able to immediately identify both cluster performance and single node behavior.

An example of a cluster of eight machines is shown in Figure 5. The idea is to use a representation based on the HSB (Hue, Brightness, Saturation) color system to codify resources of every node. The parameter H, varying in the range [0, 1.0] is used to denote a machine; H is proportionally divided according to the number of on-line machines by the formula:

$$H_i = \frac{1.0}{N} i$$

where N is the number of nodes, and i assumes values from 0 to N-1.

Multiprocessor nodes are managed by the B channel in order to obtain different tones of color. In the upper right part of Figure 5 cluster nodes are listed as well as the number of CPUs. CPU occupation is represented by a Kiviat chart able to provide a quantitative information for each processor.

Memory and network occupation statistics are displayed in the lower part of the screen by two bar charts. The upper part of the bar chart representing memory occupation denotes swap utilization (see typhoon1 and typhoon4 in Figure 5). The bar Figure 5 Characteristics of the whole cluster.
chart representing network statistics is split in two parts: the upper one denotes out coming traffic while the lower one incoming packets.

Tapping/clicking either on a bar or on a slice, information about a single node (as in Figure 3) are presented.

The portability/compatibility has been obtained writing the code in Java and using the AWT (Abstract Windowing Toolkit) library for developing the graphics interface.

7. REMARKS

The efficiency is one of the key issue in the design of monitoring systems. To evaluate the performance of the proposed architecture, the monitoring system has been installed in a cluster domain running a CPU/network intensive application for distributed three-dimensional scene rendering [8].

The rendering software is based on the Parallel Virtual Machine (PVM) [5].
Client performances have been tested using two different processors: and “old” 400 MHz Intel Celeron and a more recent 2 GHz AMD Athlon. Setting a refresh time of five seconds for all monitored resources, CPU occupation (often called CPU intrusion) due to monitoring clients are, on the average, of 0.4% for the Celeron and 0.2% for the Athlon, while RAM occupation is about 1 MB. Moreover, setting the above refresh rate, the network traffic due to information transmissions to the server is of 0.9 KB/s.
The application has been tested both on a Personal Digital Assistant device and on PC using three different Internet browsers: Intenet Explorer 6, Netscape 4, and Konqueror (a web browser has to support at least Java 1.1 or PersonalJava 1.2). The selected PDA is a Compaq iPaq H3630 equipped with the Microsoft PocketPC operating system. Some basic features are: 206 MHz StrongArm CPU, 4096 colors TFT LCD display, 240x320 pixels (2.26 x 3.02 inches) resolution touch screen, 32 MB RAM and 16 MB Flash ROM.

8. CONCLUSIONS AND FUTURE WORK

This paper presents an architecture for remote monitoring-management of clusters based on Linux/Windows. System resources can be monitored by an extremely efficient client running on every machine. Information are transmitted to a platform independent server that gathers data into a database. Data can be remotely displayed via web or, above all, via mobile devices such as PDAs.
The visualization interface has been designed and implemented in order to consider the reduced sizes of PDA displays; ad-hoc icons have been chosen to allows the user to immediately check the status of the whole cluster as well as to provide a complete frame of resources of a single machine. Some basic management tasks are also implemented.
Future work will address three different issues: to extend monitoring-management capabilities to Windows clients, to enhance management functionalities in order to allow task migration, to study new icons able to represent resources of a larger number of machines.

9. ACKNOWLEDGMENTS

This project is supported by the Ministero dell'Istruzione dell'Università e della Ricerca in the frame of the Cofin 2001 project: *Elaborazione ad alte prestazioni per applicazioni con requisiti di elevata intensità computazionale e vincoli di tempo reale*.

10. REFERENCES

[1] T. Anderson, D. Culler and D. Patterson *A Case for Now. IEEE* Micro, pp. 54-64, 1995.

[2] E. Anderson and D. Patterson *Extensible, Scalable Monitoring for Cluster of Computers.* Proc. of the 11th Systems Administration Conference (LISA'97), 1997.

[3] R. Buyya, B.T. Koshy and R. Mudlapur *GARDMON: A Java-based Monitoring Tool for Gardens Non-dedicated Cluster Computing System.* Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), pp.2774-2780, 1999.

[4] G. Huwei, H.H.S. Ip and Z. Yanchun *Java-Based Approaches for Accessing Databases on the Internet and a JDBC-ODBC Implementation.* Computing & Control Engineering Journal Vol. 9, No. 2, pp.71-78, 1998.

[5] G.A. Geist and V.S. Sunderam *The PVM System: Supercomputer Level Concurrent Computation on a Heterogeneous Network of Workstations.* Proc. of the 6th Distributed Memory Computing Conf., pp.258-261, 1991.

[6] A. Keller and H. Reiser *Dynamic Management of Internet Telephony Servers: a Case Study Based on JavaBeans and JDMK.* Proc. of the 3rd International Enterprise Distributed Object Computing Conference, pp.135-146, 1999.

[7] F. Lamberti and A. Sanna *A Java Web-based Multichannel Architecture for Distributed System Monitoring.* Journal of Internet Technology Vol. 3, No. 4, pp. 235-244, 2002.

[8] O. Lazzarino, A. Sanna, C. Zunino and F. Lamberti *A PVM-Based Parallel Implementation of the Reyes Image Rendering Architecture.* Proc. of Euro PVM/MPI 2002, pp.165-173, 2002.

[9] Z. Liang, Y. Sun and C. Wang *ClusterProbe: an Apen, Flexible and Scalable Cluster Monitoring Tool.* Proc. of the 1st *IEEE* Computer Society International Workshop on Cluster Computing, pp.261-268, 1999.

[10] P. Naughton and H. Shildt *JAVA: The Complete Reference.* Mc-Graw Hill Inc., 1997.

[11] Openview: http://managementsoftware.hp.com/products/

[12] Piranha: http://www.elctech.com/products_ piranha.shtml

[13] R. Rajkumar, K. Mohan and B. Gopal *PARMON: A Comprehensive Cluster Monitoring System.* Proc. of the 5th International Conference on Parallel and Distributed Processing Techniques and Applications, 1998.

[14] R. Subramanyan, J. Miguel-Alonso, J. Fortes *Design and Evaluation of a SNMP-based Monitoring System for Heterogeneous, Distributed Computing.* Technical Report, TR-ECE 00-11, School of Electrical and Computer Eng., Purdue University, 2000.

[15] P. Uthayopas, C. Jaikaew and T. Srinak *Interactive Management of Workstation Cluster Using WorldWide Web.* Proc. Cluster Computing Conference, 1997.

[16] P. Uthayopas, S. Katchamart, J. Fakcharoenpol and P. Hemunnopjit *SMILE: Toward the Affordable High-Performance Computing Platform using Network-Based Multicomputer.* Proc. of the 1st National Computational Science and Engineering Symposium, 1997.

[17] P. Uthayopas, S. Paisitbenchapol, T. Angskun and J. Maneesilp *System Management Framework and Tools for Beowulf Cluster.* Proc. The 4th Int. Conference/Exhibition on High Performance Computing in the Asia-Pacific Region, Vol. 2, pp.935-940, 2000.

[18] M. Widenius and D. Axmark *MySQL Refernce Manual.* O'Reilly & Associates, Inc., 2002.

[19] Xmon: http://globalserver.fujitsu.com/en/ docs/xsp.pdf

[20] C. Zunino, F. Lamberti, A. Sanna and B. Montrucchio *A Wireless Architecture For Performance Monitoring And Visualization On PDA Devices.* In Proc. of SCI'02, Vol. XV, pp.143-148, 2002.