

XPVM-W95 – A Performance Monitoring Tool for PVM Clusters on Windows Operating Systems ¹

Daniel Kikuti
Paulo Sérgio Lopes de Souza
Simone do Rocio Senger de Souza

Computer Department
State University of Ponta Grossa – UEPG
Praça Santos Andrade, s/n – C.P.: 992/993
84.010-330 – Ponta Grossa – PR – Brazil
danielkikuti@yahoo.com.br - pssouza@uepg.br - srocio@uepg.br

¹ This work was developed with CNPq financial support.

Abstract

This paper presents XPVM-W95, a new monitoring tool for parallel applications that utilize PVM-W95 (Parallel Virtual Machine for the Windows Operating System); a message-passing environment for distributed platforms running Windows. The development is justified by the absence of a tool that allows visualizing and evaluating the performance of both the parallel virtual machine on a Windows System and the parallel applications. This tool has functionality analogous to XPVM (X-Window interface for PVM) developed for UNIX systems. Even so, it shows significant differences. The GUI's (Graphical User Interface) implementation is treated as a separate module that is independent of the functionality, thus, allowing more flexibility and portability. The monitoring activity presents information about the tasks, the parallel environment and the workload of each host, making monitoring more extensive. The results discussed in this paper demonstrate that the tool provides adequate help for a correct, friendly and objective performance evaluation.

Keywords: Performance monitoring, PVM, Windows, Cluster, Distributed parallel computing

1 Introduction

The increase of programs' complexity and the need to obtain shorter response-time motivates the parallel computing practice. PVM (Parallel Virtual Machine), a message-passing environment for parallel processing in heterogeneous environments, allows hosts¹ connected by a network to be considered as a single virtual machine [2] [8].

To make users able to evaluate the performance of their applications, it is necessary to have a tool able to inform how the current configuration of the virtual machine is, which tasks are being executed and where they are running. The tool must also notify about the workload of each host, thus helping the correct distribution of the tasks, considering their particular characteristics.

In order to provide a better interaction between users and the PVM system, the display of collected information is optimized by means of a friendly graphical user interface (GUI). This makes fast comparisons possible and an efficient analysis based on the information is obtained in the environment. The tool must be interactive, i.e., users may be able to choose which information would be more useful to be displayed and, by having the information, users should be able to tune the environment according to their requirements. The task monitoring should not have significant influence in the performance of other running tasks in the same environment; therefore, its functionality must consume the minimum amount of system resources.

The capacity of dealing with the platform's heterogeneity is another desirable feature for the tool, since PVM was designed to run on different architectures (workstations, parallel computers with distributed or shared memory, personal computers and so on) and in diverse operating systems. This factor makes the algorithm used to collect information about the workload less portable. Despite of any implementation detail, the values returned by the monitoring tool, which are used to quantify the workload on the platform, must be able to allow the correct comparison of different hosts and architectures present on the parallel virtual machine.

This paper describes XPVM-W95, a new monitoring software-tool for parallel applications that employ PVM-W95 (Parallel Virtual Machine for Windows Operating System) and it is in accordance with the principles previously discussed [8][9]. The main objective of this work is providing appropriate information about the performance of the distributed platform (the parallel virtual machine) and about the parallel applications running on it, during the execution.

XPVM-W95 shows some similarities to XPVM (PVM's graphical interface for UNIX systems), however, they have completely distinct implementations and also some different functionality [3][5][6]. Besides of the information already available through XPVM, the tool proposed on this paper shows information in a graphical way to allow the analysis of each hosts' workload. It is expected with this tool to diffuse the monitoring of parallel machines among programmers that utilize the PVM-W95 environment and later extend its functionality to other platforms that need an analogous monitoring.

2 Monitoring

The monitoring activity detects events while they take place on the environment. It is the capability to abstract information for immediate or future utilization. This is only accomplished by preventing the monitoring process from causing not wanted influence in the normal functioning of the environment or that carry to wrong conclusions about the state of the platform.

The first step on the development of a monitoring tool for a parallel environment is to define what to

¹ The terms host, processor and machine are used as synonymous in this paper.

monitor. XPVM-W95 has defined three scopes of performance monitoring: PVM tasks, the parallel virtual machine and each host that compose the parallel virtual machine.

The monitoring of PVM tasks informs which tasks are being executed and which have already finished, how much processing time each task has spent and where they are or have been executed. This functionality permits the user to have a notion about the task distribution through the virtual machine, even if the data is not grouped for this purpose. It is also possible to observe, on a chart, the amount of time that each task has consumed (see Fig. 4).

The diagram showed at the Fig. 1 shows information about the current virtual machine configuration and represents with icons the architecture of each machine and its name. This diagram also represents the host's state by colors (white if it is idle or green if at least one task is being executed on it). To get information about the number of tasks that are being executed or which is the identifier of a particular host, the user just needs to click on the icon related to that host.

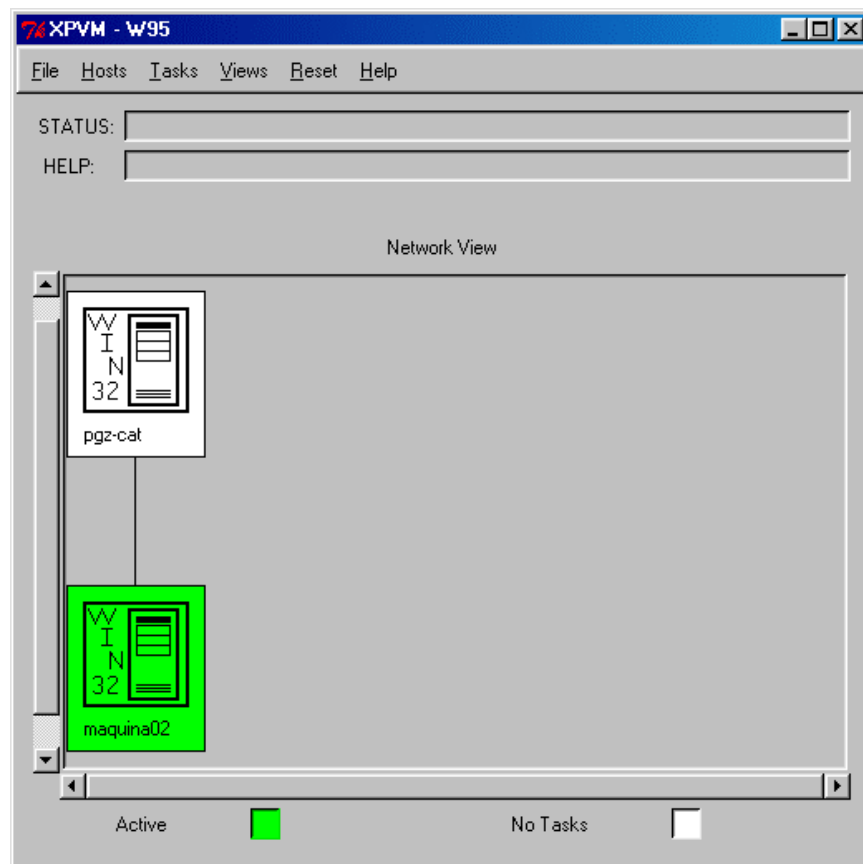


Figure 1 – Virtual-machine representation diagram

The behavior of each host, not considering the parallel environment provided by PVM, is another issue taken into account. The goal is to offer information about workload generated by other applications in the system, not only PVM applications, therefore, making the users able to figure out whether a certain host is supporting their applications properly or not. The display of this information is represented by measures related to processor utilization, which includes the number of running processes and the processor's utilization percentage. Memory measures are also used, showing information about the percentage of utilization or the amount of available memory.

Having defined what should be monitored, the next step is to know how the monitor must act. Prior to the description of this activity, it is important to contemplate the environment that the tool had been projected to and the impact of this in the project.

Windows 9x, the operating system currently used by PVM-W95 and consequently also used for the monitoring tool, has some restrictions concerning to the collection of information about the load of the computational platform. An example of these limitations is the difficulty to obtain the number of processes that are executing on a host at a certain instant [1]. Having only the information returned by the Windows 9x API, it is not possible to determine which processes are running, sleeping or blocked waiting I/O. Consequently, the correct comparisons between different machines' workload become more challenging. To overcome this constraint the monitoring tool employs as a load index the percentage of CPU utilization available to the user.

Another issue to be considered is that the tool is based on version 3.3 of PVM and, thus, some of the trace facilities provided at version 3.4 are not present[4][5]. This difference does not impose any limitation for XPVM-W95 concerning the final result; however it imposes some extra work in the development of the algorithm designed to collect the required information. XPVM-W95 utilizes the `pvm_tasks()` function to obtain information about the tasks and the `pvm_notify()` function to learn about when a task has finished.

The diagram of the virtual machine is created through the information returned by the `pvm_config()` function, which obtains the number of hosts and their respective configuration. Through the `pvm_notify()` function, the tool is informed about when a host has been added to or removed from the virtual machine.

The charts that symbolize the workload of the hosts are obtained through primitives that collect data from the operating system. These functions have been implemented in a module apart. Thus, the activity of collecting the load index of each host does not remain consuming the machine's resources unless the user requests so. This modularization also allows a significant improvement in terms of portability, because each local module is responsible for collecting information about the host on which it is running, in compliance with the operating system and architecture.

3 The XPVM-W95 tool

The performance monitoring tool XPVM-W95 is open source, public domain and not commercial. Its main features are portability, friendly graphical interface and ability to deal with the platform's heterogeneity.

The tool's portability was one of the key features of the development, which got extra attention. As a solution it was decided on separating the functionality from the interface, developed in using the C-ANSI language and TCL/TK, respectively [7][10]. The idea behind this separation was to provide more versatility to XPVM-W95, besides making future changes in the interface much easier. Beyond these advantages, the separation also allows the functionality to be adapted for different platforms without any change at the user interface.

The C language was chosen because it offers enough resources to the development of this project, it is widely disseminated in the academic environment and it is also used in PVM source code. It allows the tool to be easily carried to other platforms, considering the characteristics of each operating system and the language version available. Another factor that contributed to the language choice was the availability of free versions of the compiler.

TCL (Tool Command Language) is a script interpretation language able to provide the functionality available in other scripting languages currently used in shells. It is a multi-platform and has a wide variety of built-in commands. TK is a tool kit initially designed to the X-window system. It allows the creation

and manipulation of widgets² through TCL commands. TK is a popular tool kit due to its flexibility and easy development of systems. Another positive aspect is the capability of controlling the interaction between applications, including their look and feel and their functionality.

In order to assure the desired flexibility the development has been divided into two modules. An upper layer responsible for displaying the visual elements and for the interacting with the users - called Front End; and another layer responsible for providing the functionality of the graphical interface - named Back End. Fig. 2 illustrates the modules functioning and how the interaction with the user occurs.

The user interacts directly with the Front End, where all the visual components that the user can manipulate are displayed. It is through these elements that services requests are made and parameters are passed to the Back End. The Front End also has the purpose of displaying to the user information provided by the Back End. A straight communication between the user and the lower layer can be perfectly made, but the use of the Front End makes data representation more organized and clever. Moreover, its use is extremely easy and convenient. The implementation of the Front End has been done using the positive aspects of TCL/TK, however this does not hinder the future use of other language, for example Java, GTK+ (GUIMP Tool Kit), Qt (acronym to cute), LessTif, and others. This freedom of language choice is possible due to the standardization of the call to the modules and the use of arguments in both Front End and Back End.

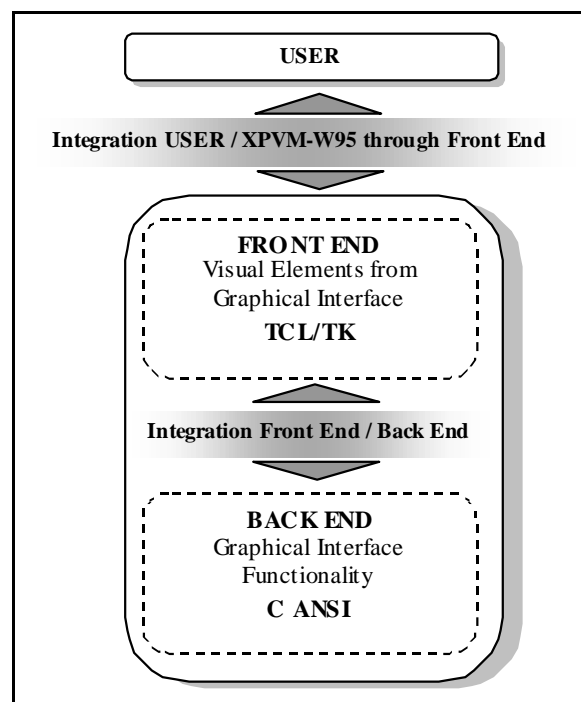


Figure 2 – Graphical interface module

The Back End is responsible for the functional part of the tool and it must always supply the requests coming from the upper layer (Front End). When an event is triggered at the Front End, it is translated into a TCL command that calls the executable files. The Back End receives these data, performs the requested

² Widget is an element of a determined class that has particular appearance and behavior. Examples of widgets are buttons, menus, scrollbars, text box, etc.

function and returns the information to the Front End. The Back End takes care of the two kinds of solicitations: of PVM console and of the monitoring.

Fig. 3 illustrates the lower layer divided in three modules, which will respond to solicitations coming from the Front End. The communication between the Front End and the Back End is made with different purposes for each module and each element is a program developed in C. Although this division exists, the way the communication between the upper/lower layers is made remains unchanged. These divisions just separate the functionality according to its specific functions.

The monitor module (trace.exe) is initiated when the interface of XPVM-W95 is started. This module has the objective of launching the PVM Deamon (PVMD), if it has not been running yet. At this moment it's possible to gather information about the current state of the parallel virtual machine. This module will be active for as long as XPVM-W95 is up. The monitor is in charge of notifying the interface as soon as the PVMD halts, as a result, the interface will end immediately. Besides starting up and terminating the parallel virtual machine, the monitor is in charge of supplying the Front End with the information necessary to show some useful charts that will be discussed in the next section.

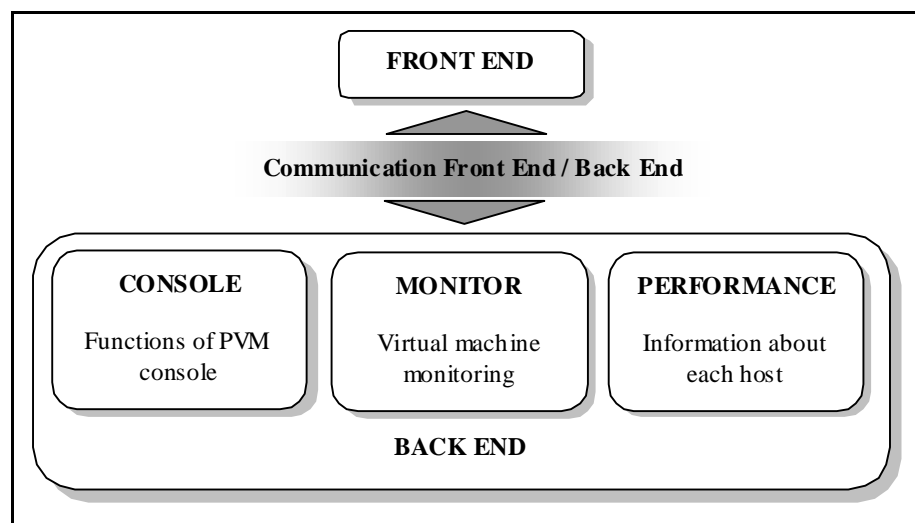


Figure 3 – Back End structure

The console module is an executable file (xpvm.exe) that has the objective of providing the main functionality available in the PVM console. Through this application it is possible to add/remove hosts, quit PVM, kill a task or all of them, list the tasks that are running, send a signal to the tasks and spawn new PVM tasks. When a user requests one of these functions, the Front End sends a command with specific arguments to xpvm.exe that (through argc and argv) identifies which service has been requested, then executes it. The console module stays up only for the necessary time to execute the user request. Hence, XPVM-W95 does not consume processor resources while there are no user requests.

Finally, the performance module (performance.exe) is activated when the user requests information about the workload of each host concerning a specific load index. The performance module is launched and returns information to the Front End that display the results obtained. This application gets information about how many hosts compose the virtual machine and for each one it will startup a new performance monitor. These new processes will gather the load information on each host and send the result to the host running the XPVM-W95 Front-End. When the user does not want to examine the load information anymore a signal is sent to each task to make it finish its execution.

Although XPVM-UNIX inspired the development of XPVM-W95, the structure of this differs in many aspects from that. The main differences are the approach to this implementation, concerning separation between the functionality and the interface; and the performance analysis, bringing information about external applications.

4 Results

The results obtained have the purpose of demonstrating the tool behavior regarding to the applications running on PVM and the load analysis of each host.

Figure 4 displays a chart called “Space Time View”. This graphic informs about which tasks are running on the virtual machine in relation to the time space that each one consumes during their execution. It is a simple bars chart that contains, on the left side, information about the tasks that are running, which consists of the name (when started by spawn command) and its task ID (TID). On the right side, the tasks are displayed in the shape of bars, having their width changed according to the time that they remain executing. On the upper right side there is a field that informs the amount of time elapsed since the moment at the first task started until the moment that there are no tasks executing. On the bottom side the amount of time related to a specific task is displayed.

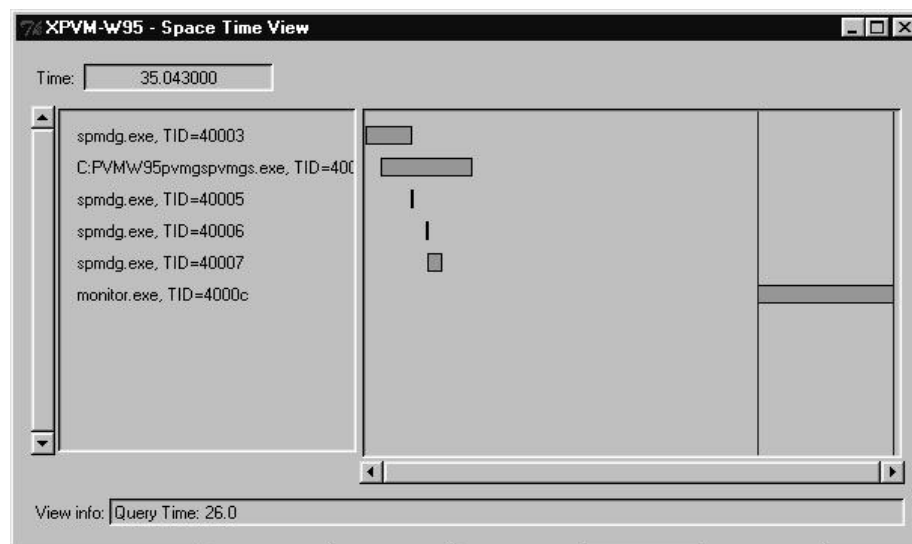


Figure 4 – Space-time view

As new PVM tasks arrive, information about each one is displayed below the last task already registered. In the bars chart there is a vertical line that moves continuously while at least one task is running, indicating the total time elapsed. Another line, which has a free orientation and is controlled by the user shows the amount of time elapsed until any specific instant.

The chart that shows the space-time view and the representation diagram of the parallel virtual machine are exhibited and refreshed by the interface using the data collected in the monitor module, that is part of the Back End layer.

The following graphics are representations of the load of each host that composes the PVM, according to a selected measure. For the processors load analysis the tool uses two measures: the number of processes in the run queue (Fig. 5) and the percentage of processor utilization (Fig. 6). To evaluate the memory utilization two measures are used: the percentage of used memory (Fig. 7) and the total memory used (in

MB) (Fig. 8). The purpose of these graphics is to give necessary information to allow a better distribution of tasks by PVM.

The graphic showed in Fig. 5 classifies each host as idle (white), moderate (green) or overhead (red). The thresholds used to define these three ranges can be determined by the user, allowing a better flexibility due to its capacity of being fitted on the real needs of the users.

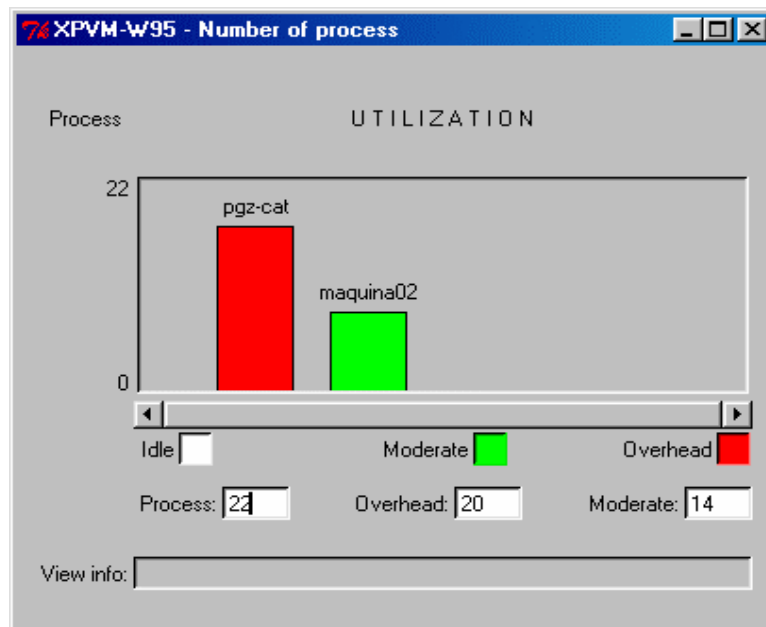


Figure 5 – Run queue

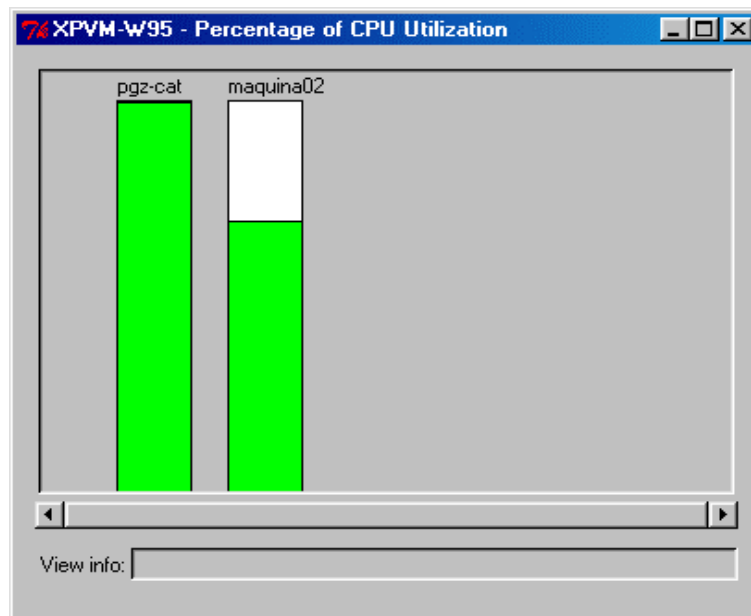


Figure 6 – Percentage of processor utilization

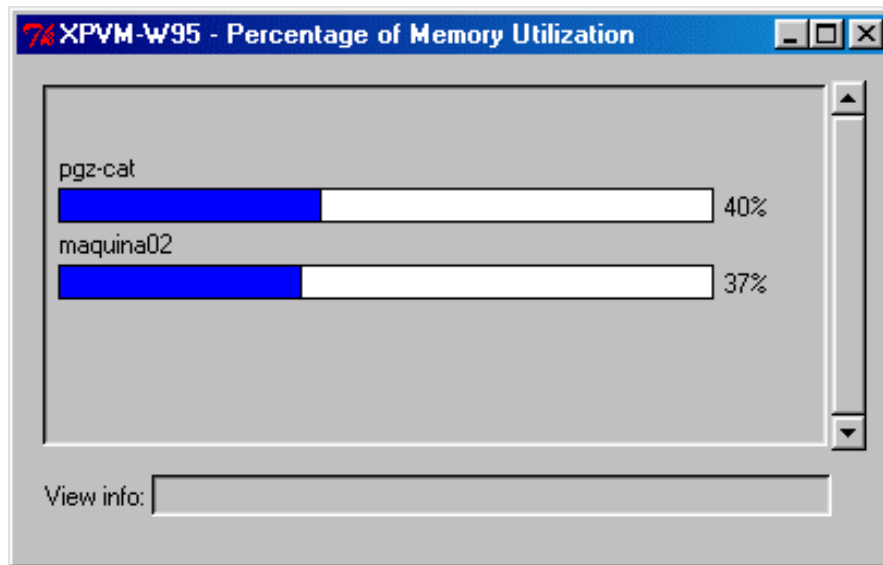


Figure 7 – Percent of memory utilization

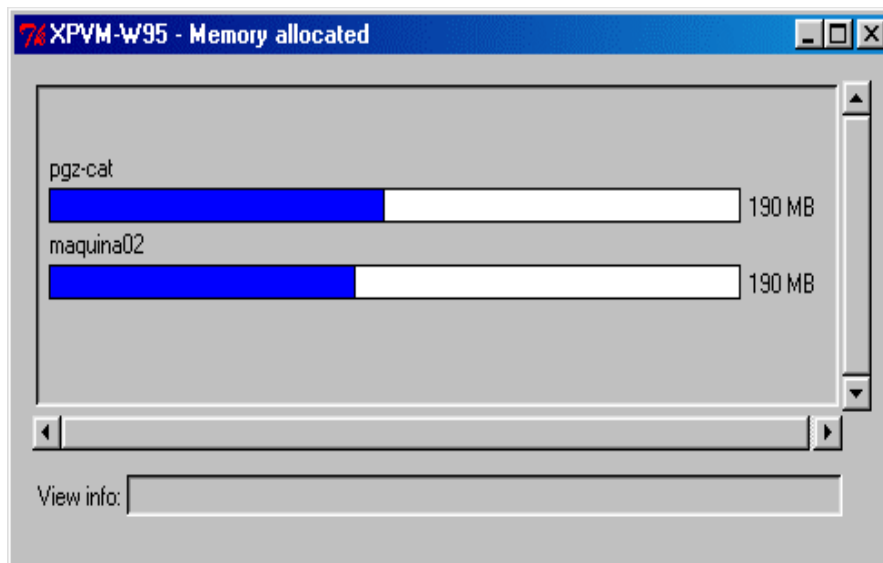


Figure 8 - Used memory

5 Concluding Remarks

This paper presented XPVM-W95, a performance-monitoring tool for parallel applications for Windows environments. The tool has been developed to offer a friendly graphical interface for the PVM-W95 console, monitoring of the parallel environment and support for workload analysis. The tool is flexible, portable and easy to be modified with no need to remodel its structure.

The results described demonstrate that XPVM-W95 has a stable behavior and reached the objectives proposed. Besides owning functionality similar to the existent in the tool XPVM for UNIX, XPVM-W95 it allows a great portability of its source code and also allows the performance monitoring of the parallel virtual machine, using different load indices.

From now on, it is expected to make the necessary changes in order to run the tool on the LINUX platform, analyzing the behavior of the graphic interface and the behavior of the modules responsible for the functionality of the tool. It is expected that Linux impose fewer limitations than Windows, making possible the expansion of the monitoring tool. It is expected that this tool will be able to support scheduling process on multi-user parallel virtual machines.

References

- [1] BonAmi Software Corporation. *Supplementing Windows 95 and Windows 98 Performance Data for Remote Measurement and Capacity Planning*. Proceedings of Computer Measurement Group 1998 International Conference, Anaheim, CA, December 1998.
- [2] Geist, G.A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V. *PVM: Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*. The MIT Press, 1994a.
- [3] Geist, G.A., Kohl J.A. and Papadopoulos, M.P. *Visualization, Debugging and Performance in PVM*. Proceedings of Visualization and Debugging Workshop. October, 1994b.
- [4] Geist, G.A., Kohl J.A., Papadopoulos, M.P. and Scott, S.L. *Beyond PVM 3.4: What We've Learned, What's Next, and Why*. Proceedings of Euro PVM-MPI, 1997.
- [5] Kohl, J.A. and Geist, G.A. *The PVM 3.4 Tracing Facility and XPVM 1.1*. Proceedings of 29th Hawaii International Conference on System Sciences (HICSS'96). January, 1996.
- [6] Kohl, J.A. and Geist, G.A. *XPVM 1.0 User's Guide*. Oak Ridge National Laboratory, Tennessee. May, 1995.
- [7] Ousterhout, J. K. *The Tcl and Tk Tool Kit*. Draft, September 8, 1994.
- [8] Souza, P. S. L. *Máquina Paralela Virtual em Ambiente Windows*. Master Dissertation. ICMSC/USP – São Carlos - Brazil. May, 1996.
- [9] Souza, P. S. L., Senger, L. J., Santana, M. J., Santana, R. C. *Evaluating Personal High Performance Computing with PVM on Windows and LINUX Environments*. In: Proceedings of Euro PVM-MPI, 1997.
- [10] Welch, B. *Practical Programming in Tcl and Tk Draft*. January 13, 1995. <http://www.sunlabs.com/~bwelch/book/index.html>.