The Visuel Performance Analysis and Monitoring Tool for Cluster Environments[*]

Li-Jen Chang[1], Hsiang-Yao Cheng[1], Hsun-Chang Chang[1], Kuan-Ching Li[1], Hsiao-Hsi Wang[1], Chao-Tung Yang[2], and Liang-Teh Lee[3]

[1]Parallel and Distributed Processing Center (PDPC), Dept. of Computer Science and Information Management, Providence University, Taichung 43301, Taiwan ROC
E-mail: { ljchang, s9011124, hcchang, kuancli }@cs.pu.edu.tw
[2]High Performance Computing Laboratory, Dept. of Computer Science and Information Engineering, Tunghai University, Taichung 40744, Taiwan ROC
E-mail: ctyang@mail.thu.edu.tw
[3]Dept. of Computer Science and Engineering, Tatung University, Taipei 10451, Taiwan ROC
E-mail: ltlee@cse.ttu.edu.tw

Abstract

In this paper, we present Visuel tool for performance measurement and analysis of MPI parallel programs in cluster environments. Most of tools available today for cluster systems show system performance data (e.g., CPU load, memory usage, network bandwidth, machine-room temperature, server average load, among others), being more suitable for system administrators who maintain such system. The Visuel tool is designed to show performance data of all computer nodes involved in the execution of MPI parallel program, such as CPU load level and memory usage. Moreover, this tool is able to display comparative performance data charts of multiple executions of the application under development of an MPI application.

Keywords. Monitoring tools, MPI parallel programs, distributed computing, performance visualization.

1. Introduction

In recent years, the cluster computing technology has become a cost-effective computing infrastructure because it aggregates resources of computational power, communication and storage [7, 9]. It is also considered to be a very attractive platform for low-cost supercomputing.

Cluster of workstations are easy to build, cost effective and highly scalable. Basically, it consists of several workstations that are interconnected through a high-speed network (Gigabit Ethernet, SCI or Myrinet) for information exchange and coordination among them. They run commodity operating systems, such as NT/2000, Linux or UNIX. Parallel applications for distributed systems are developed using message passing libraries, such as MPI. With the advances in networking technology, connecting PCs and workstations is not a problem anymore. Despite of this fact, there is still much to do in the software domain.

Parallel programs can behave in a number of unexpected ways, because of their complex structure, the parallel system on which they run, the number of nodes used to execute the application in a cluster environment, the dataset used by the parallel code, the regularity of applications and algorithms in space and time, the variability in programming environments, the heterogeneity of software and hardware platforms, among others. In addition, effective partitioning, allocation and scheduling of application programs on a network of workstations are crucial to obtain good performance. Thus, the performance is very sensitive to the strategy used to distribute data to the processors [1].

[*] This research is supported in part by the National Science Council, Taiwan, under grant no. NSC93-2213-E-126-010.

There are several performance monitoring tools available, to visualize graphically the performance of an application's execution, e.g., VAMPIR [11], Paradyn [5] and DIMENAS [10]. One way to improve the performance of a parallel application is to analyze performance data, e.g., CPU load, memory usage, I/O load, among others, and see what happened with the execution of that MPI parallel application.

In this paper, we designed and implemented Visuel, a tool to provide graphical performance data visualization of the execution a MPI parallel application in a cluster system. This tool is useful during the development process of an application, tuning its performance, and to proceed with "what-if" analysis.

The remainder of this paper is organized as follows. In section 2 is discussed some related researches in monitoring tools for distributed systems. Section 3 introduces the Visuel tool. Later, in section 4, an example of use of Visuel, when performing the execution of Matrix Multiplication (MM) parallel program in a cluster computing system, and finally in section 5, a brief conclusion and future works are presented.

2. Related work

Nowadays, several performance monitoring tools are available, in order to visualize graphically the performance data of an application's execution, e.g., VAMPIR [11], Paradyn [5] and DIMENAS [10].

A number of monitor tools that generate HTML pages that contains performance graphical images and data are also available. MRTG (Multi Router Traffic Grapher) [6], based on Perl and C, it is a tool to monitor the traffic load on network links, by providing visual representation in HTML pages. It consists of Perl script that uses SNMP to read the traffic counters and a fast C program that logs the traffic data. RRD (Round Robin Database) [8] is a tool that stores and displays time-series data (e.g., network bandwidth, machine-room temperature, and average load) in a compact way that will not expand over time.

The advantage of MRTG over RRD is that it is easier to use, while RRD has more graphical display options than MRTG. Though, the main disadvantage is that MRTG has fixed format data (it can only shown the data over time), and it depends fully on the use of SNMP to obtain the data, otherwise, it cannot work.

Several well-known tools such as Ganglia Cluster Toolkit [2], CACTI and NMIS are particular implementations of RRD tool done by independent teams around the world. As mentioned before, tools such as Ganglia and CACTI can only show the data over time of each one of the computer nodes in a cluster system, not possible to show in particular time periods, such as the start and end of execution of an application in a cluster system.

3. System Overview

Visuel[†] tool is created and implemented by using RRD tool [8]. The main reason why we started to work on this tool is that we need a tool to visualize the performance data an MPI application during its execution, as it starts and it finishes. Additionally, we need a tool that we can process "what-if" analysis, that is, performance tuning of a parallel application during its development phase.

Visuel is scalable, i.e., it is able to measure long running MPI applications on as much computing nodes as they are involved in the computations. It is able to generate from minutes to several hours MPI parallel programs' executions.

This tool supports heterogeneous and homogeneous clusters of workstations; basically, this tool can work on any environment where RRD tool is able to run and installed.

In the next subsection, we will discuss the components of this performance tool.

3.1 Components of the System

The Visuel performance tool is composed of two components. The first one is Visualization Manager (VM), which provides to the user graphical visualization of application execution's data, while the second component is the User Interface Manager (UIM), responsible to handle data collection and time histograms.

Basically, the Visuel tool is designed as a shell script in Linux and it is relied on RRDtool. The Visualization Manager processes the visualization in three steps, as show in figure 1.

1. rrdtool create: set up a new Round Robin Database (RRD)
2. rrdtool update: store new data values into an RRD
3. rrdtool graph: create a graph from data stored in one or several RRD

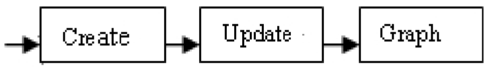


Figure 1. Visualization creation process in Visuel tool.

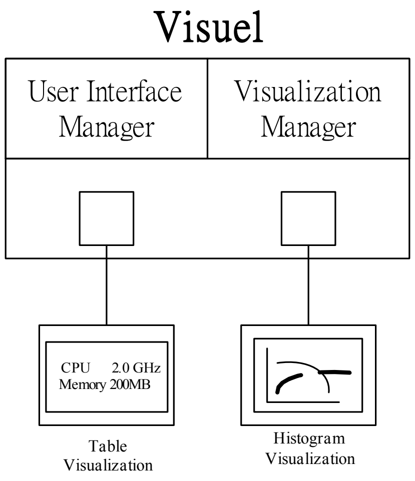


Figure 2. Visuel tool and its components.

The Visualization Manager (VM)'s main objective is to monitor how much of resources a MPI parallel program needs to be executed in a cluster system, from its start point to its end point, e.g., CPU load, memory usage, network bandwidth. These data are used for future performance analysis.

Different from other monitoring tools, such as Ganglia and Cacti, it provides performance data since the system is on, not being able to provide specific measurements in a period of time. Additionally, if a given period of time is passed, we can not have in our chart these data for analysis.

In this sense, the programmer can use data obtained from successive executions of the MPI parallel program in the performance tuning of parallel code process, in order to observe the difference of performance of tuned MPI parallel program. The several results are visualized through our VM, in order to provide with visualizations of performance charts among these MPI program executions.

The following are steps to execute the VM:

Step 1: for the purpose of record the performance data selected at this initial step, the rrd database is built every time on those computing nodes involved in our computation.

Step2: before executing our MPI parallel program, the master node should execute MDF (Monitor Daemon Parent process), which goal is to fork MDC (Monitor Daemon Child process) to each of involved computing node. The MDC in each computing node involved is going to detect when master node starts with the distribution of tasks (segments of code of MPI parallel program), its job is at this moment recording the performance data, originated from the execution of MPI parallel program. Before the beginning of execution, each involved computing node is in "waiting" state, since the tasks did not reach to the computing nodes yet. See figure 3 for details.

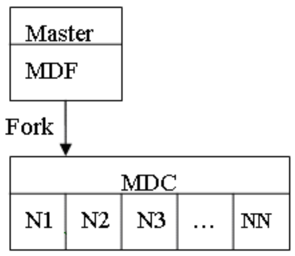


Figure 3. Master node MDF parent process is fork to child processes MDC, to every involved computing node

Step 3: As the MPI parallel program is started to run in each of involved computing nodes, MDC in each computing node is acknowledged. MDC will get defined system resource usage from each computing

† It means "visualization", in French.

node, and through network file system protocol, these performance data are written back to rrd database in Master Node and log file.

At the moment of overlapping two data charts of the same MPI parallel program executed, since their execution time are different, they will appear side by side in different execution times. The log file is used to correct this problem, and it helps us in overlapping the two performance data charts in the same execution, beginning at the same start point.

As MDF detects the end of execution of MPI parallel program, this process will broadcast to every computing node involved a message to stop monitoring the computing nodes, and the process of obtaining performance data can be stopped.

See figure 4 for additional explanations of this step.

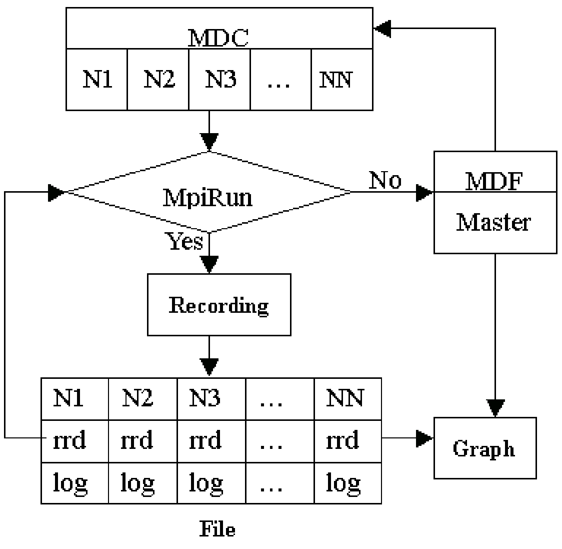


Figure 4. File system in rrd database and log file.

Step 4: During the programmer starts the performance tuning process, by reviewing several executions of the same MPI parallel program, the programmer chooses some of several executions of this MPI parallel program to display a combined data chart of these selected executions. The scheme in figure 5 shows details of this selection process.

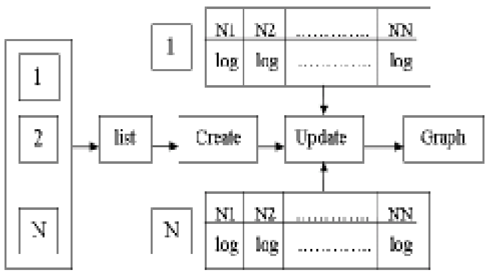


Figure 5. Selection of specific executions of a MPI parallel program during its development.

Step 5: The Visuel tool will clean up pending processes by checking each of computing nodes, since these only cause marginal errors in performance data. Otherwise, it is possible to cause programmers misinterpret obtained results.

4. Example of Use

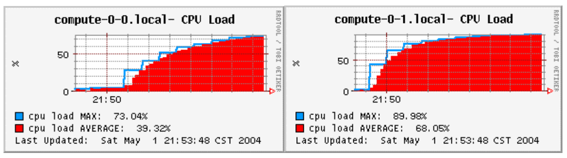
We have used Visuel tool to study several distributed applications. In particular in this section, we will show the performance data with visualization displays of the execution of a program, Matrix Multiplication (MM), downloaded at Matrix Market [4].

The experimental environment where the experiment was executed is:

— 16 PC nodes, with Fedora Core1 and Kernel 2.4.22-1.2115.nptl installed

— each node's configuration is: CPU AMD Athlon 2400+, 1GB DDR memory, 60GB HD, interconnected via Fast Ethernet.

Our experiment is to execute matrix multiplication (MM) using 8 nodes of the cluster system mentioned. From the charts show in figure 6, we can observe that the first 8 charts represent the CPU load in each one of computer nodes.



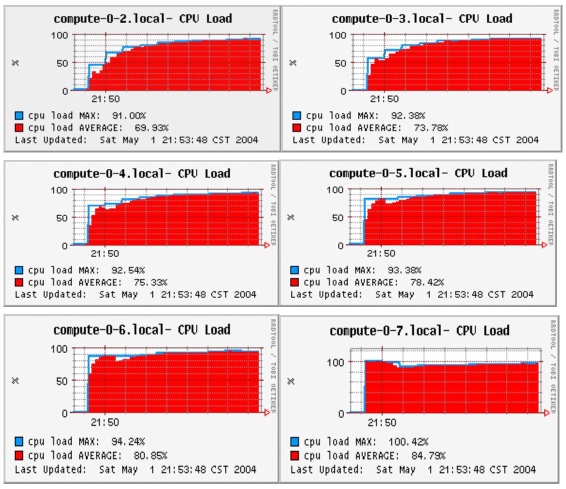


Figure 6. CPU usage in computing nodes of cluster, where compute-0-X means X-th node of the cluster system.

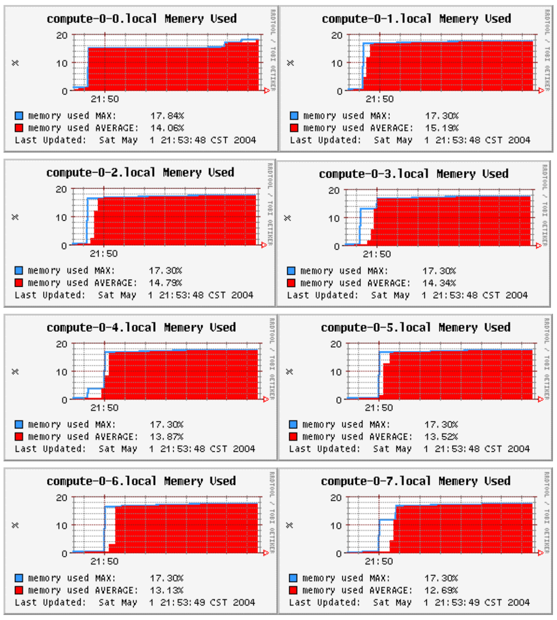


Figure 7. Memory Used in computing nodes of cluster, where compute-0-X means X-th node of the cluster system.

Our cluster is a 16-node system, so we proceeded with our experiment using 8 of 16 nodes. It means that the other 8 nodes were idle during the execution of this experiment.

There are two charts in the figure 8, where the left one is the CPU load and the right is the memory usage. This is the chart that may show for the other 8 computing nodes that were not involved the execution of our application.

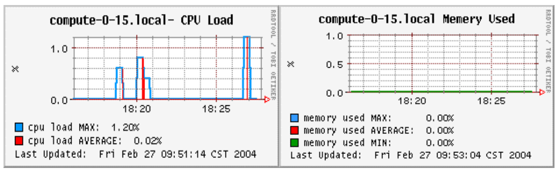


Figure 8. Example of chart that shows those computing nodes of the cluster system that **were not** involved the execution of the experiment.

Additionally, We made modifications in a matrix multiplication (MM) parallel program, turn it to two different versions. We would like to compare these two versions, to know which of them had a better performance tuning.

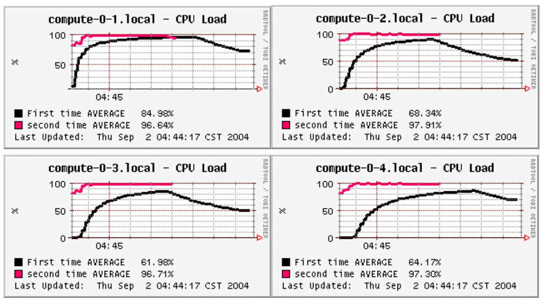


Figure 9. Comparison of the original matrix multiplication program and its modified version.

By analyzing the charts provided, we can understand the execution of a MPI application from the point it starts to when it finishes. We can have from these charts each computing node's information, e.g., CPU usage, memory load. In this way, HPC application designers will have information to process "what-if" analysis and improve application's performance.

With these charts in our hands, we can clearly have the total amount of resources needed in each one of successive executions of an application under development, challenging in each step a higher performance of this application.

By comparing the results obtained in each run during "what-if" analysis (these information are stored in the database), it is possible to know in which one of all runs we had higher performance. Also, it is possible that after several runs during "what-if" analysis; almost nothing has changed when comparing the charts. This means that the changes in the application made by application developer not sensitive to the application itself.

5. Conclusion

In this paper, a tool for performance monitoring and measurement is introduced. With it, analysis of bottlenecks and load balancing can be done, improving the performance of MPI parallel applications.

Visuel supports high-level parallel languages, allowing programmers studying the performance of their programs using the native abstractions of the language. In addition, Visuel provide a detailed, time-varying data displaying as easy-to-understand charts about a MPI program's performance. As result, to work with MPI large applications in Visuel can be as easy as to handle someone's small prototype application.

There are remaining many directions for our future work. The first one is to extend this monitoring tool for use in grid computing, so that we can improve performance of applications in this environment. A second research is to design a tool to locate communication bottlenecks, since we know that, for communication intensive applications, matching the collective communication patterns can reduce communication overheads and hence improve performance.

The number of computing nodes is increasing in most existing cluster systems, the complexity involved in managing the resources in these systems is high, making necessary to reduce database's data redundancy. Moreover, with the use of php in the interface layer, it is possible to easily visualize the performance data in computing nodes of cluster systems.

The goal is to assist software programmer with performance evaluations of his application, before and after code modifications, understanding the use of resources in each one of computing nodes involved in the computation, scheduling better the computation distribution of his computer program, and finally, improve the performance of this application.

References

[1] A. L. Cheung and A. P. Reeves. High performance computing on a cluster of workstations, IEEE 1992.

[2] Ganglia Cluster Toolkit. http://sourceforge.net

[3] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard, in http://www.mcs.anl.gov/mpi/mpicharticle/ paper.html, Argonne National Laboratory, 1996.

[4] Matrix Market webpage. http://math.nist.gov/MatrixMarket/

[5] B.P. Miller et al. The Paradyn parallel performance measurement tools, *IEEE Computer,* Special issue on performance evaluation tools for parallel and distributed computer systems, 28, 11, pp.37-46, 1995.

[6] MRTG Webpage, in http://www.mrtg.org

[7] D.K. Panda and L.M. Ni. Special Issue on Workstation Clusters and Network-based Computing: Guest Editors' Introduction. Journal of Parallel and Distributed Computing, 40(1), January 1997.

[8] RRDtool Webpage, in http://www.rrdtool.org

[9] J. Sang, C.M. Kim, T.J. Kollar and Isaac Lopez. High-performance cluster computing over Gigabit/Fast Ethernet, Informatica, 23, pages 19-27, 1999.

[10] DIMEMAS Tool Webpage http://www.cepba.upc.es/dimemas/

[11] Pallas Products Webpage (VAMPIR tool) http://www.pallas.de/e/products/vampir