Scalable monitoring and configuration tools for grids and clusters

Philippe Augerat[1], Cyrill Martin[2], Benhur Stein[3]
[1] INPG, ID laboratory, Grenoble
[2] BULL, INRIA, ID laboratory, Grenoble
[3] UFSM, Santa-Maria, Brasil

Abstract

We present the Ka-admin project that addresses the problem of collecting, visualizing and feeding back any grid information, trace or snapshot, compliant to an XML-like model. Real use includes performance analysis of parallel applications and cluster administration.
Ka-admin includes a generic "filter" module that processes monitored data independently of what they represent. Filters can remove, aggregate and transform data or pass them to external applications. The end user is responsible for activating the filters from within an interactive graphical interface. This allows him to focus on important information. We also present a "scatter/gather" module that allows efficient collection and distribution of data and commands in a large cluster. Early work on "MPI/threads" applications and system monitoring tools proved that the combination of both modules matches the objective of a scalable visualization of large data sets.

1. Introduction

Architectures for distributed computing have become very popular over the last decades making a wide range of services available on the Internet. While resources and needs (high performance computing, data sharing, web services) exist, the problem is now to match software with the available resources. Resource monitoring has thus become a critical part of the global infrastructure. We are considering in this paper a virtual cluster built from a large number of Internet resources, and the problem of collecting, visualizing and tuning the cluster resources.

A simple model for a virtual cluster could be an intranet or a grid of PC clusters because it has both a very large number of heterogeneous resources and a limited administrative scope. Thus we can expect administrators and users in such a model to be able to execute commands or access information on all nodes in the virtual computer.

As far as collecting information is concerned, bwatch [1] was the first step to obtain a global view of a cluster. In the context of network administration, there are also numerous tools that monitor system information. Some of them like Performance Co-pilot [8] capture both PC system information and network device information. Finally, software has been developed to trace the behavior of dedicated applications. Vampir [5], Tau [6] and Pajé [12] are examples of such programs for high performance message passing computing.

Once the information is collected, next problem is to move information collected from distributed agents to a centralized monitor server. Ganglia [2] uses a multicast protocol to achieve efficient real-time monitoring. ClusterProbe [11] is an open and flexible monitoring system based on a multi-protocol interface and a hierarchy of monitor servers to achieve scalability of communications.

A complete monitoring tool should include or link to a management tool in order to have the administrator or the tool itself launch actions on the virtual cluster (reboot computers or schedule applications for instance). In addition to monitoring tools, Smile [3] provides administrative tools for clusters. Basically, it allows Unix commands to be distributed efficiently on a cluster. C3 [9] and rshp [7] projects also provide general frameworks for such parallel commands.

A few works extend to multi-cluster and grids. The Grid Performance Working Group [4] propose a general framework for monitoring services and focuses on interoperability with other grid services such as authentification. The M3C project [10] joins several modules to monitor, manage a grid, schedule jobs, etc.

The aim of the Ka-admin project is to design a generic tool that could leverage most existing software while being more scalable and efficient. The main concern is the definition of an extensible language and the associated filtering system that help focus on the important information. By combining the usage of Pajé, Performance Co-pilot and rshp, recent experiments have proved it possible to achieve our aim.

The next section of this document describes the existing implementation and utilization of Ka-admin. The third

section deals with the expected properties of monitoring tools as far as genericity and scalability are concerned.

2. Description of the Ka-admin software

Ka-admin is built from two independent modules. The first one, named "scatter/gather module" is in charge of collecting and updating data from/to the grid while the second named "filter module" is in charge of extracting interesting data to be displayed and selecting actions to be performed on the grid. Figure 1 shows the interaction between the two modules.

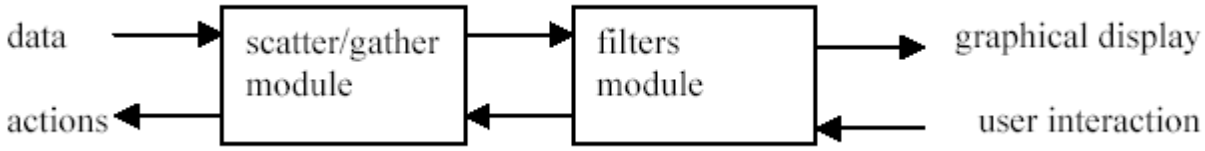

Figure 1: Ka-admin overview

a. Data collection

The scatter/gather module collects information from all node monitor agents then moves it to the server node, which may then feed back the agents with user-defined commands. The module is based on rshp, a general framework for starting programs on large-scale clusters. The rshp library was first used as an alternative solution to start the rsh command on different nodes in the cluster. Extensions to standard Unix commands also have been generated using the rshp library. Standard protocols (rsh, ssh) are used with modifications on the clients only. Using standard daemons instead of dedicated agents avoids splitting the scatter and gather functions in two different modules. To collect information from all nodes, rshp leverage the stdout/stdin properties of these protocols to create a permanent socket network that moves information from nodes to the centralized monitor node. This logical network can be built as a multicast network or as unicast binomial or n-ary trees

Actual research on rshp is to define a generic framework to test and build the best permanent logical network for a grid of clusters. In general, the best case would be grid-dependant and rshp would adapt itself to the architecture. Early experiments on a 225 nodes Ethernet multi-switched cluster shows that the ideal network may change depending on the function (scatter or gather). The best method we found so far for scattering data is to parallel the rsh client (a three step connect/authenticate/start process) and to use a flat tree for the connect phase. A comparison of the methods tested is shown in figure 2.

As far as gathering information is concerned, a binomial tree performs better. Yet, a balance between the monitor server load and the network load should be also considered as well as the possible integration of intermediate filters on each node of the logical communication tree.

On individual nodes, the scatter/gather module only consists in a wrapper to existing system tools. Actual implementation encapsulates either Performance Co-pilot software developed by SGI or homemade software. Performance Co-Pilot is a monitoring system by SGI for Irix and Linux computers. It monitors information which can alternatively be found in the /proc directory. It has both a C and shell interface. The system is designed to limit intrusion.

If no monitoring software is available on a node or the user does not have the rights to use it, Ka-admin defaults to a command that access the /proc information directly.

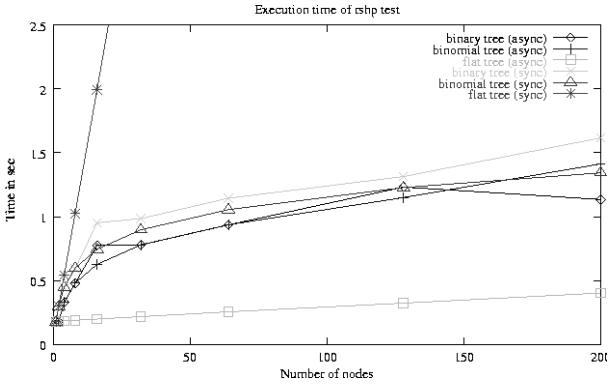

Figure 2: parallel rsh start on a 200 nodes cluster

b. Data visualization

The Ka-filters module extends Pajé [12], a scalable visualization tool designed for parallel applications. Main usage of Pajé is to visualize programs based upon MPI and threads libraries [13]. However, it has been designed to deal with various models and has been tested to visualize a data-flow programming interface and a java virtual machine behavior [14].

The genericity of Pajé is based on the independence of the trace format and the visualization tool. Pajé defines a grammar to describe a complex hierarchy of objects. A user can then define his own object model and generate compliant traces. Using Pajé to display traces, the user will interactively select a time window, adapt the granularity of the visualization (for instance processes vs. threads in the case of a parallel multi-threaded application) and finally filter non-important information. Pajé also allows generating automatic statistics or data aggregation. The figure 3 gives the model for an MPI/threads program.

```
<Execution>
  <node>
    <thread>
      <state > </state>
      <MPI_event > </MPI_event>
    </thread>
    <semaphore>
      <state> </state>
    </semaphore>
    <lock>
      <state> </state>
    </lock>
  </node>
</Execution>
```

Figure 3: a simplified object model for a multi-threaded MPI program

A parallel program can be described as a hierarchy of objects (figure 3). The leaves in the tree representation of this hierarchy are called entities. Other items are called containers. The trace of the program (figure 4) is a list of time-stamped events: creation of a container, creation of an entity value, etc

```
T1 PajeCreateContainer Execution
T2 PajeCreateContainer Execution.node 0
T3 PajeCreateContainer node0.thread0
T4 PajeDefineEntityValue thread0.state
T5 PajeDefineEntityValue  thread0.MPI_event
…
Tn PajeSetState thread0.state running
…
Tp PajeSetState thread0.state stopped
…
```

Figure 4: Part of a Pajé trace for an MPI/thread program

The visualization of an MPI/threads program execution is shown on figure 5. A multi-level visualization is demonstrated (each thread running on node 1 is displayed, while all threads running on node 0 are grouped on one line and node 3 and 4 are grouped on one line.
Another "filtering" functionality is demonstrated since only "thread states" and communications" are shown. Other types of entities like "semaphore states" have been filtered out. The end user has complete control over this "filters", thus can focus on the entities that matter.

Various sources (system, application and middleware information) and traces (Vampir, Tau, etc) have been converted using the Pajé extended markup language. The next section gives several examples.

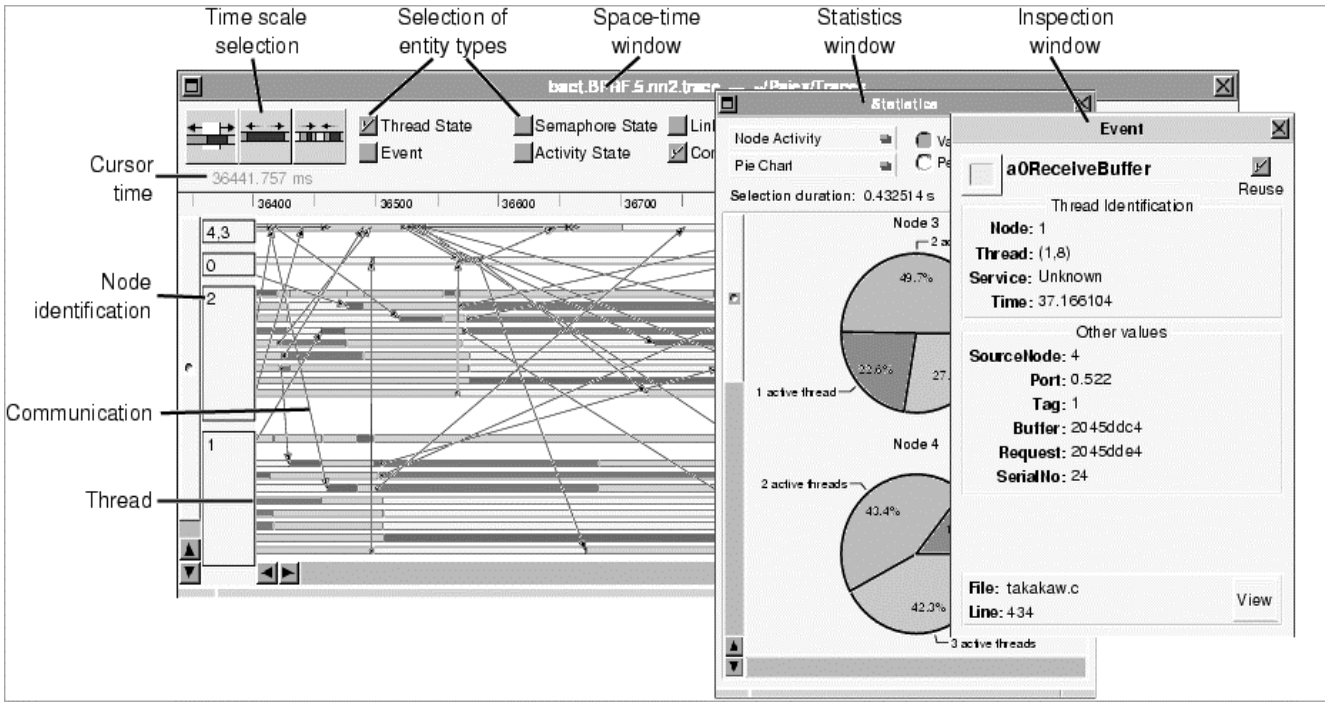

Figure 5: Snapshot of Pajé monitoring windows

c. Real use: post-mortem visualization

The Ka-admin system can be used to monitor post-mortem information. Real use already concerns system log files and application behavior.

Portable Batch System [16] is a widely used batch system that allows real-time node allocation. We use Ka-admin to trace and visualize node reservation in PBS. Tracing is done on the PBS job master node at job start. Figure 6 gives a simple model for a PBS trace.

The "pbs-task" entity is initialized to nobody and when a user launches a PBS job, the "pbs-task" entity is set to the user name. The cluster administrator can use Pajé to visualize the cluster use. Figure 7 provides a view of a PBS trace. Each color matches a user. Pajé filters allow viewing a given list of users only.

```
<grid>
  <cluster>
    <switch>
      <node>
        <pbs-task ></pbs-task>
      </node>
    </switch>
  </cluster>
</grid>
```

Figure 6: Simple model for monitoring PBS with Pajé.

While the PBS package already provides a graphical view of a cluster, Ka-admin provides additional features.

Figure 7: A PBS trace with Pajé

Firstly the Ka-admin filters natively provide data aggregation. For instance, the end user can easily set a filter to count the number of active nodes in the cluster (all nodes for which the "pbs-task" entity is not set to nobody). Practically, Pajé automatically creates a new entity in the cluster container and sets its value to the number of active nodes.

Secondly any Pajé container or entity can be linked to an external command. Selecting an entity with the Pajé GUI will give access to this external command and pass the entity and its hierarchy as parameters to the command. For instance, the cluster administrator can kill a PBS job directly from the Ka-admin interface or send e-mail to active users.

Another example is the monitoring of the Linpack HPL [17] application. To monitor system information during the execution of an application, the scatter/gather module starts Performance Co-pilot on individual node monitors and then collects and converts all data at the end of the run. Measurements with the Linpack parallel benchmark show that the intrusion (the overhead in the execution time) is the same as the sequential overhead caused by Performance Co-pilot on one node.

Ka-admin allows monitoring any type of information collected by PCP. In the example in figure 8, statistics on the network interface, the memory and cpu loads have been captured by PCP.
The figure 9 shows a Pajé snapshot of a partial HPL trace. The user selected only «network.out.bytes» entities. In addition, the user has created new objects, network.load and network.load.all using Pajé filters. The first filter automatically aggregates the network.out.bytes values on each switch to compute the network.load value. The second filter computes the aggregation on the whole cluster (network.load.all).

```
  <cluster>
    <switch>
      <node>
        <processor>
          <kernel.all.cpu.user>
          <kernel.all.cpu.idle>
          <kernel.all.cpu.sys>
          <network.out.bytes>
          <mem.freemem>
…
```

Figure 8: An object model for tracing system information with Performance Co-pilot

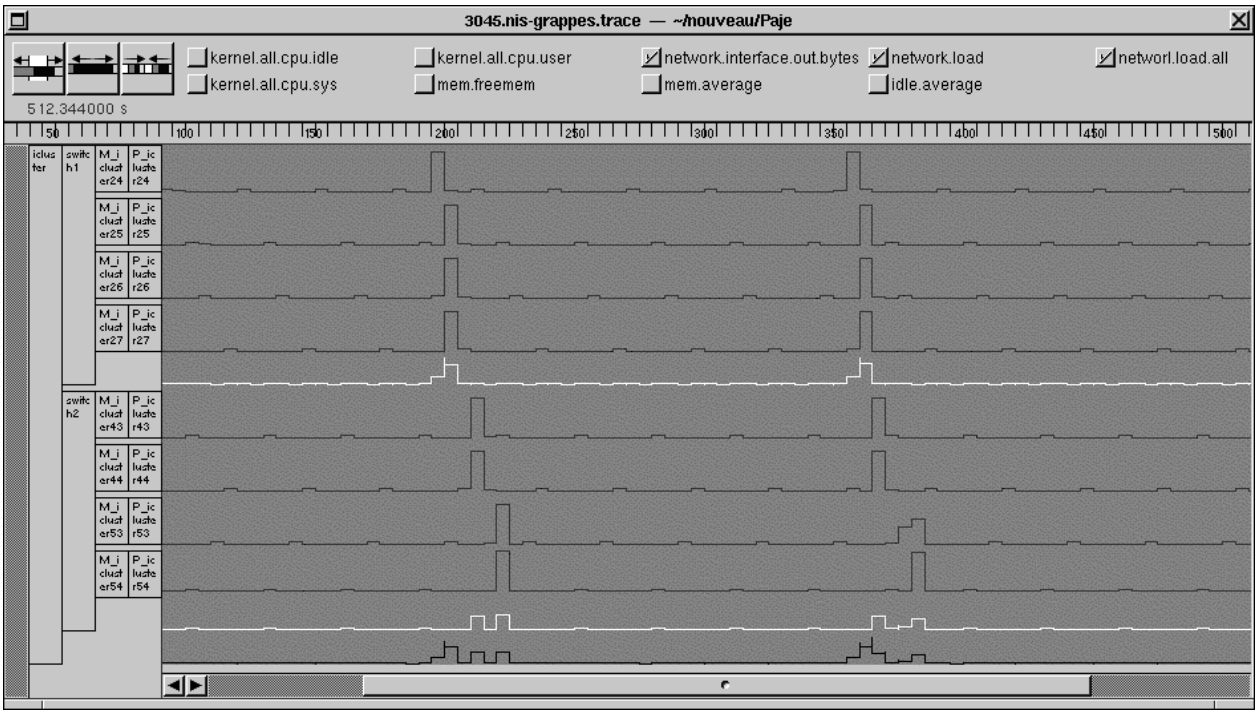

Figure 9: Pajé snapshot of Linpack execution and data aggregation

With Ka-admin, it is also easy to mix heterogeneous data such as MPI traces and system load indices. One research work has started to take advantage of both granularity levels (application and system) for parallel application performance analysis. Another interesting feature of Ka-admin is to allow a view of several executions of the same application.

3. Genericity and scalability for grid monitoring tools

A scalable view is one whose format, clarity, meaning and size are independent of the number of elements involved in computation [19]. From our experience with Pajé, we propose now two solutions to achieve this definition.

a. Genericity

Numerous administrative tools exist for clusters especially since every cluster administrator designs his own monitoring tool. Deciding which is the best would be as unfair and useless as comparing Latex and Microsoft Word. However this comparison is appropriate to underline that one major problem is the lack of a common language to describe data. Moreover, in order to handle various facility models (cluster, grid, etc) with heterogeneous resources, the language should be extensible to describe what is to be monitored. XML is an emerging standard that could help.

As presented, the Pajé visualization tool already uses a similar language. Converting XML data to a Pajé trace is straightforward.

The monitoring usages in the grid context include fault detection, security logging, performance analysis, scheduling, etc. Both an administrator (for log files, extreme system behaviors, software installation) and a user (for application tuning) may want to use and adapt the monitoring tool. This also encourages the defining of a high level grammar to describe the objects to be monitored independently of the monitoring tool. To allow both real-time and post-mortem visualization, the extensible language should also handle time stamped events that represent any object changes. A considerable work has been done in the context of the grid forum to define a schema for data formats for performance monitoring as well as a common interchange format for monitoring tools[20].

The monitoring tool should not restrict itself to display raw monitored data. It should also perform user-defined calculations upon the data (data aggregation in Pajé) or trigger action on the monitored resources (external shell commands in Pajé). The Pajé language could be extended to associate various executable commands (methods) with any monitored object.
Finally, the adequate monitoring tool should capture changes in the grid architecture. While the virtual cluster language description may be fixed for a specific usage, the virtual computer description itself should be obtained from existing detection software or from interactions with a user. Events like creating/deleting a node are

possible in the Pajé language. This is also a requirement for fault tolerant monitoring.

b. Scalability

Monitoring a large number of computers combines two hard scalability problems. One of these is getting the information without much system and network intrusion (the producer problem). The other is focusing on the most important information within a large set of information (the consumer problem).

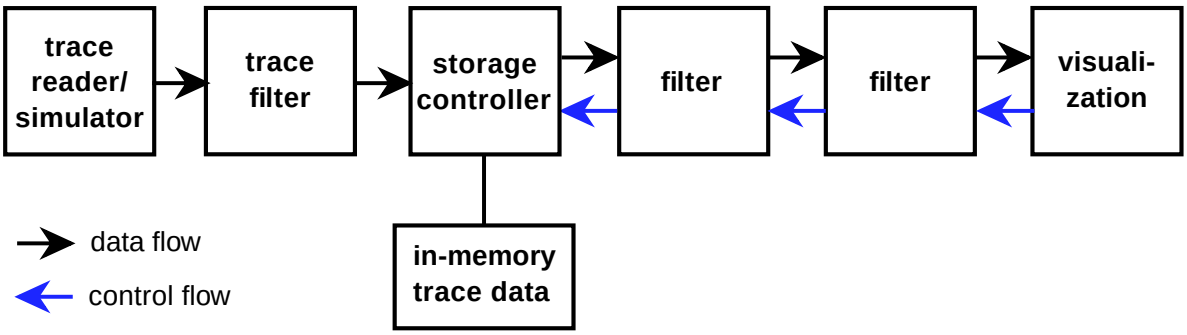

Figure 10: Pajé workflow

A solution to the producer problem is already handled by rshp, Ganglia or ClusterProbe. The next step should be to provide a generic tool with adaptative protocols on both the agent side and the wire side.

Pajé handles the solution to the consumer problem by implementing a graph of filters. All filters share the same description language. A filter obtains a grid description and data from the previous filter in the queue and passes it to the next filter after activating one or several methods. Figure 10 summarizes Pajé workflow.

The simplest example of a filter removes information from the previous one. For instance, it could remove computers that run Linux from a grid description.
One filter may feed objects with information, for instance, may capture the CPU load for all nodes in the grid. Other filters can aggregate information to create new information. For instance, one filter can cumulate network traffic on all nodes on a communication switch to compute the switch load.
Pajé implementation provides two types of a filter in a graph. The first acts on the flow of data, filtering it as data are read from the trace file. The second type of filter can dynamically change the view of all trace data stored in memory. While the first kind of filter acts as soon as data is available, the second kind acts on demand, answering to queries originated by user actions on the visualization interface. Between these two kinds of filters is a "storage controller" module, responsible for taking care of what portions of trace data should be kept in memory and for controlling the trace reader whenever more data is necessary.

Although it is not the case in current Pajé implementation, all modules may run on a different computer. Distributing the work of multiple "storage controller" will help storing a large amount of data. Distributing the other filters will help managing a very large number of entities.

c. Interactivity

Combining filters allows the user to work with a partial or aggregate view of the whole data set, then to go back to a complete detailed view. From this, we understand that interactivity is one of the solution to enforce visualization scalability.

The user chooses which objects he wants to display, not the visualization tool. All objects may be hidden, and any combination of objects may be created. Ideally, the user may also control the information collection directly from the visualization tool.

d. Perspectives

The Ka-admin system is currently in use on a large cluster at INRIA Rhône-Alpes and is to be deployed on a grid of clusters in the VTHD project, the biggest French grid test-bed.

Since the most powerful part of Pajé appears to be the "filtering system", we will work mainly now on this module and make it independent from any graphical tool or monitoring tool. The "filter" module should be usable above other monitoring system such as Ganglia or ClusterProbe, helping them to achieve scalability. On the other hand, the "scatter/gather" module is to become self-adaptable to a heterogeneous architecture.

Future work includes: to parallel Ka-admin by distributing the filters on different computers, to provide a generic way to define external commands so that any administrative command could be performed from the Ka-admin interface, to leverage the Ka-admin features in the real-time context.

4. Conclusion

We have presented the Ka-admin project that challenge the problem of collecting, visualizing and feeding back any grid information, trace or snapshot, compliant to an XML-like model.
While current usage is to have one different monitoring tool for each cluster or application, we propose a generic "filter" module to process data only taking into account their description language not the description itself. Thus data can be filtered (removed, aggregated, transformed, passed to external applications) independently of what they represent. The end user is responsible for activating the filters he wants, allowing him to focus on interesting information. We also present a "scatter/gather" module that allows efficient collection and distribution of data and commands in a large cluster. Early work on "MPI/threads" applications and system load indices proved that the combination of both modules matches the objective of a scalable visualization of large data sets.

References:

[1] Jacek Radajewski. bwatch. http://www.sci.usq.edu.au/staff/jacek/bWatch/.
[2] Matt Massie. Ganglia cluster monitoring toolkit. www.millennium.berkeley.edu/ganglia/.
[3] Putchong Uthayopas et al., 'Interactive Management of Workstation Cluster Using WorldWide Web', *ClusterComputing Conference* (CCC'97)', 1997.
[4] Brian Tierney and all, White paper : A Grid Monitoring Service Architecture, Grid Performance Working Group
[5] W.E. Nagel, A. Arnold, M. Weber, H-C. Hoppe, K. Solchenbach, VAMPIR: Visualization and Analysis of MPI Resources, Supercomputer 63, Vol.12, No. 1, pp. 69-80, 1996
[6] A. Malony and S. Shende, Performance Technology for Complex Parallel and Distributed Systems, Proc. Third Austrian-Hungarian Workshop on Distributed and Parallel Systems, DAPSYS 2000, "Distributed and Parallel Systems: From Concepts to Applications," (Eds. G. Kotsis and P. Kacsuk) Kluwer, Norwell, MA, pp. 37-46, 2000.
[7] C. Martin, O. Richard. Parallel launcher for clusters of PC, submitted to World scientific
[8] SGI Silicon Graphics Inc. Linux advanced cluster environment.http ://oss.sgi.com/projects/ace/.
[9] C3, http://www.epm.ornl.gov/torc/C3/
[10] M3C – An Architecture for Monitoring and Managing Multiple Clusters
[11] ClusterProbe: An Open, Flexible and Scalable Cluster Monitoring Tool
Zhengyu Liang , Yundong Sun, and Cho-Li Wang, Department of Computer Science and Information Systems, The University of Hong Kong , Pokfulam Road , Hong Kong
[12] J. Chassin de Kergommeaux, B. de Oliveira Stein, and P.E. Bernard. Pajé, an interactive visualization tool for tuning multi-threaded parallel applications. Parallel Computing, 26(10):1253_1274, aug 2000.
[13] J. Chassin de Kergommeaux and B. de Oliveira Stein. Pajé : an extensible environment for visualizing multi-threaded programs executions. In A. Bode, W. Ludwig,T. Karl, and R. Wismüller, editors, Euro-Par 2000 Parallel Processing, Proc. 6th In-ternational Euro-Par Conference, volume 1900 of LNCS, pages 133_140. Springer,
[14] F.-G. Ottogali, V. Olive, B. de Oliveira Stein, J. Chassin de Kergommeaux, and J.-M. Vincent. Visualization of distributed java applications for performance debugging. Soumis à publication.
[15] Cyril Guilloud, Jacques Chassin de Kergommeaux, Philippe Augerat, Benhur Stein , Outil visuel d'administration système pour grappe de processeurs de grande taille, RENPAR 2001, 24, 27 avril 2001
[16] http://www.openpbs.org/. The Portable Batch System Software (PBS v2.2p13), Veridian, PBS Products Dept., Mountain View, CA, March 2000.
[17] HPL http://www.netlib.org/benchmark/hpl/
[18] GNUstep http://www.gnustep.org.
[19] Couch, A.L. Categories and context in scalable execution visualization. Journal of Parallel and Distributed Computing, vol. 18, n. 2, 1993, pp. 195--204)
[20] Global Grid Forum Performance Area http://www-didc.lbl.gov/GridPerf/