Performance Analysis of Grid Interactive Applications with the G-PM Tool*

Marian Bubak[1,2], Włodzimierz Funika[1,2], Roland Wismüller[3,4], Tomasz Arodź[1,2], and Marcin Kurdziel[1,2]

[1] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Kraków, Poland
[2] Academic Computer Centre – CYFRONET, Nawojki 11, 30-950 Kraków, Poland
[3] LRR-TUM, Institut für Informatik, Technische Universität München, D-80290 München, Germany
[4] Institute for Software Sciences, University of Vienna, A-1090 Vienna, Austria
{bubak,funika}@uci.agh.edu.pl,wismuell@in.tum.de
phone: (+48 12) 617 39 64, *fax:* (+48 12) 633 80 54, *phone:* (+49 89) 289-28243

Abstract

In the paper we present the functionality and software design of the G-PM tool for evaluation of Grid applications performance on the basis of standard and user-defined measurements as well as means for performance visualization. Due to highly distributed nature and dynamic changes, the issues of run-time measurement definition, selective instrumentation, and efficient data collection mechanisms need to be addressed. This necessitates to focus on the interaction of distributed application components and to provide data meaningful in the context of an application. The tool under discussion collects information about a selected application via the well defined interface to the OCM-G monitoring system. High-level performance properties are defined by the user via measurement definition dialog and enable flexible and extendible insight into the performance of the application.
Keywords: Grid, performance, tools, monitoring, interactive applications

1 Introduction

The performance analysis and tuning of distributed applications is an essential step in an application development cycle. This is particularly true in the grid environment. With the introduction of cross-organization computing, the inefficiency in the application behaviour, resulting in unnecessary waste of computational and network resources, becomes the problem of all organizations involved in the grid. This is particularly important within the CrossGrid project [4], which aims at extending the grid architecture to *interactive applications*. Due to the involvement of humans into the computation loop, these applications are often strongly time-constrained, for example two of the demonstrators developed within the project, the simulation of vascular blood flow and flooding crisis

*This work was partly funded by the European Commission, project IST-2001-32243, CrossGrid

support tools. Therefore, performance analysis is one of the key issues in the CrossGrid project, which are addressed by the G-PM, a dedicated performance analysis tool.

Now, there is a number of performance tools which are already adapted to the Grid [1]. Most of the tools which support the monitoring of applications are based on an off-line analysis of event traces, which is not appropriate for for interactive applications, since they cannot present the performance data concurrently with the user's interactions with the application. On the other hand, on-line tools for the Grid are available mainly for infrastructure monitoring, used for resource management. An example is the Network Weather Service [10] that forecasts network and node performance. An exception is the monitoring system GRM [2], which is in its structure similar to OCM-G. GRM is adapted to the Grid within the Datagrid project and offers on-line performance data of Grid applications. However, GRM is mainly based on event traces, does not allow for user-defined metrics, and does not provide infrastructure-related performance data. A number of performance tools implement some ideas close to the ones presented in this paper, but none of them covers all the ideas. Autopilot [8], a distributed system, exploits a concept called *sensors* corresponding to our probes. User-defined instrumentation is used in the TAU performance analysis environment [6]. In all of these systems, however, the metrics connected with the user-defined instrumentation is already fixed by the instrumentation itself and cannot be configured at run-time. Paradyn [7] belongs to the few tools that strictly exploit the on-line approach, including the instrumentation procedure.

Compared to the existing tools G-PM features on the one hand a rather unique combination of on-line measurement, Grid awareness, and automatic instrumentation for standard application-related metrics, and on the other hand a support for manual instrumentation and user-definable metrics. The G-PM is designed to meet the specific requirements and constraints introduced by the grid and by the specific nature of the applications. It allows to extend the standard performance metrics, providing the user with a way to specify some higher-level, abstract metrics that are suitable in the application context. Also, contrary to some previously developed tools, the gathered performance data include the characteristics of the application environment and grid infrastructure. Finally, the results of micro-benchmarks are integrated into the tool, allowing for some performance prediction.

The paper is arranged as follows. In Section 2, the functionality of the tool is presented. Then, in Section 3, the design and architecture of the tool, as well as interaction with other parts of CrossGrid infrastructure are studied. A use case of performance measurement with G-PM is presented in Section 4. Finally, Section 5 gives information on the development schedule of the tool and on the features included in the first prototype.

2 Tool Functionality

The orientation of CrossGrid towards interactive applications, as well as the nature of the grid itself, poses new requirements and constraints on performance analysis. First, within CrossGrid the application user is allowed to control the application during its execution. Thus, the information that may be meaningful in the application steering process, such as performance data, have to be available on-line. Also, the performance aspect of the consequences of the user's decisions should be available to the user before the actual decision is made. Hence the need to integrate a kind of performance prediction, such as some basic benchmarks into the performance analysis tool. Next, the interpretation of the performance data in a heterogeneous, open system such as a grid, can only be done in the context of the environment of the application. Therefore, the information on performance characteristics of the grid infrastructure itself is required. Finally, we should bear in mind that the user of the application may be a specialist other than computer scientist. Therefore, the information provided to the user must not be limited to low-level data (e.g. send volume), but rather be more abstract and application specific. Thus, the tools have to be configurable, extendible and application-oriented.

The G-PM tool as the performance analysis component of the CrossGrid architecture is a tool that aims at satisfying the outlined requirements and constraints. The tool operates in the on-line mode, displaying current performance data in form of various visualization graphs.

Some basic, standard metrics, or measured quantities, have been specified. This includes both application-bound metrics and grid-infrastructure metrics. The application-bound metrics allow a low-level analysis of the application behaviour, specifying such properties as communication or file operations volume and time, or CPU usage. This metrics can be measured for the application as a whole, or can be restricted to a subset of the application processes, hosts, or specific locations in the application's code. On the other hand, the grid infrastructure metrics specify both dynamic (e.g. CPU load on a host or network load) and static (e.g. memory size of the host) information on the application environment.

While supplying very useful information, the above metrics may not carry information related to the environment the application is running in, vital in the application context, or may not summarize the information in the most suitable form. Thus, the application developer is enabled to specify his own metrics. This can be done in two ways:

- the developer may add some specific events to the application code. Such events, called *probes* may carry some additional, application-specific information to the G-PM tool, which can then be presented by the tool using the standard visualizations.
- in order to customize the performance data presented to the user we use a metrics specification language. Using this language, the user can build some higher-level metrics based on both standard metrics and user-defined metrics, including the metrics based on probes in the source code as well

as other previously specified higher-level metrics. For example, the specification language allows to combine the data from several other metrics into a single metrics, using some standard mathematical operators.

It should be noted, that for both standard and user-defined metrics, the object code of the application has to be linked against instrumented versions of the programming libraries.

3 The G-PM Tool Architecture

The architecture of the G-PM stems from it's functionality. For on-line performance measurements, the **OCM-G** [3] tool, an implementation of the On-line Monitoring Interface Specification OMIS 2.0 [5] is used. The OCM-G is solely responsible for low-level measurement of data, i.e. for communication with the instrumented libraries and gathering information supplied by them. The G-PM tool, implemented as a single process running on a user's local workstation, communicates with a single, top-level component of the OCM-G, the Service Manager (see Fig. 1). This Service Manager is responsible for forwarding the data requests and measured values to and from the specific particular hosts on which the analyzed application runs. Also, the results of the micro-benchmarks are forwarded through the OCM-G.

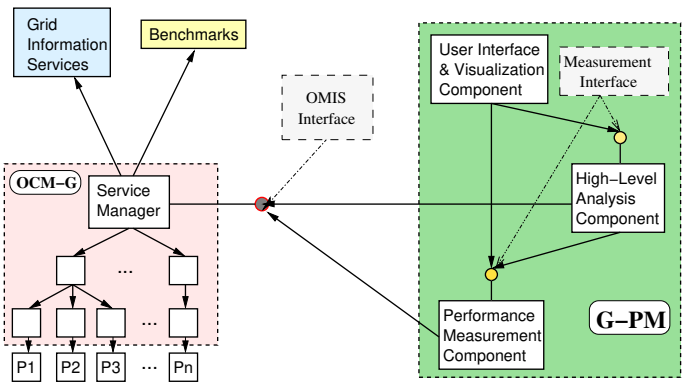

Fig. 1: Structure of G-PM

The **G-PM tool** is composed of three main modules, the *Performance Measurement Component* (PMC), the *Higher-Level Analysis Component* (HLAC) and the *User Interface and Visualization Component* (UIVC). The PMC is responsible for measurement of low-level, standard metrics via OCM-G.

The **HLAC** handles the user-defined metrics. Thus, it is involved in parsing the metrics specifications and translating the high-level metrics into a request to PMC for the underlying standard metrics. Also, it is responsible for the handling

of the events generated by the *probes* and for the processing of the information passed by the probes from the application to the G-PM tool. The probes are implemented as function calls, whose arguments can be used to pass information to the G-PM. While the high-level metrics component communicates with the OCM-G through the PMC, for the probe-based metrics the communication with OCM-G is handled by the HLAC itself through the OMIS interface. For optimization reasons, the implementation of the high-level measurements may require that the evaluation of the measurement based on the metric specification is performed within the OCM-G in an distributed way instead of within the G-PM tool. For more details on the issue please refer to [9].

Within the G-PM, both standard and custom metrics can be exploited (e.g. visualized) by the user in the same manner. To achieve this, a single **Measurement Interface** is specified for both the PMC- and HLAC-provided metrics. This interface consists of three main **classes**: `Metrics`, `MeasurementSpec` and `ActiveMeasurement`. The `Metrics` class and its subclasses provide a hierarchy of the metrics, or properties to be measured, from more generic (e.g. "send volume") to more specific (e.g. "blocking send volume"). Also, the `Metrics`-derived classes contain information on how to measure the appropriate quantities using OCM-G. The `MeasurementSpec` class summarizes the full information on the measurement, i.e. the measured metrics as well as the information on the objects (e.g. hosts, processes) on which the measurement should take place and source code restrictions (e.g. to specific functions). On the other hand, the classes derived from the `ActiveMeasurement` class contain information on a single instance of a measurement, which is currently performed. This information includes the specific identifier of the objects used by OCM-G in the process of the measurement or a corresponding visualization window.

The **User Interface** comprises the program's *main window*, which presents a list of defined measurements and open visualizations. Another important part is the *measurement definition window*. It allows the user to choose a metric and to specify its parameters, such as the above mentioned hosts or code restrictions. Once the measurement is defined, the user can specify the type and parameters of the visualization for the measurement using the *visualization definition window*. Finally, a set of *visualization windows*, ranging from bar graphs and pie charts to value-versus-time function plots, is available to the user to visualize different aspects of the application's execution.

4 Use Case of Performance Measurement

Let us assume that the user needs to get performance data on the *send volume* of different processes within an application and visualize it with a *multi-curve plot*. The realization of the measurement starts with the specification of the metrics type, the site, the nodes, the processes, the partner processes, and the fragments of the application code where the measurements will be carried out. This phase of the measurement is realized with the measurement definition window. Its time sequence diagram is show in Fig. 2. Within this part of the

measurement procedure: (1) a measurement definition is created, (2) the type of the measurement and its parameters are specified, and (3) a display window for the new measurement is created.

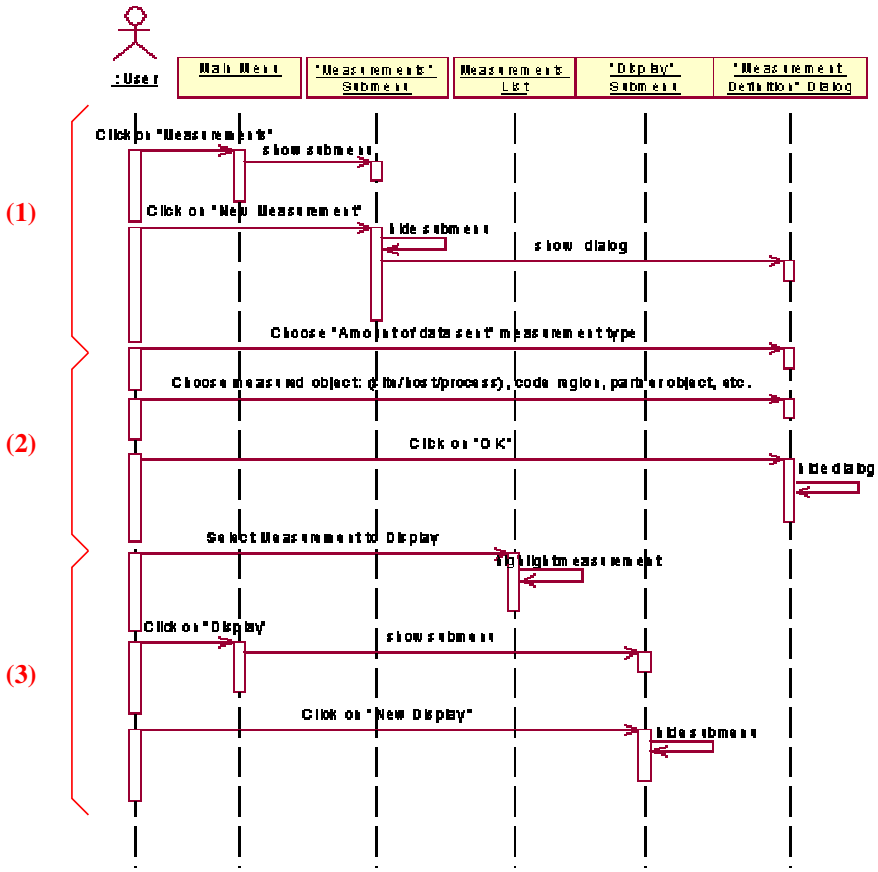

Fig. 2: Using G-PM: sequence diagram of measurement definition

Then it comes to defining the features of visualization by setting up the display type, the time mode (virtual or real), scale partition (linear or logarithmic), scale type (variable or fixed), minimal scale value, and update interval. The time sequence diagram of this phase is shown in Fig. 3. Within it: (1) a visualization widget and its properties are specified, (2) the measurement is activated, and (3) as a result measured values are displayed (see Fig. 4).

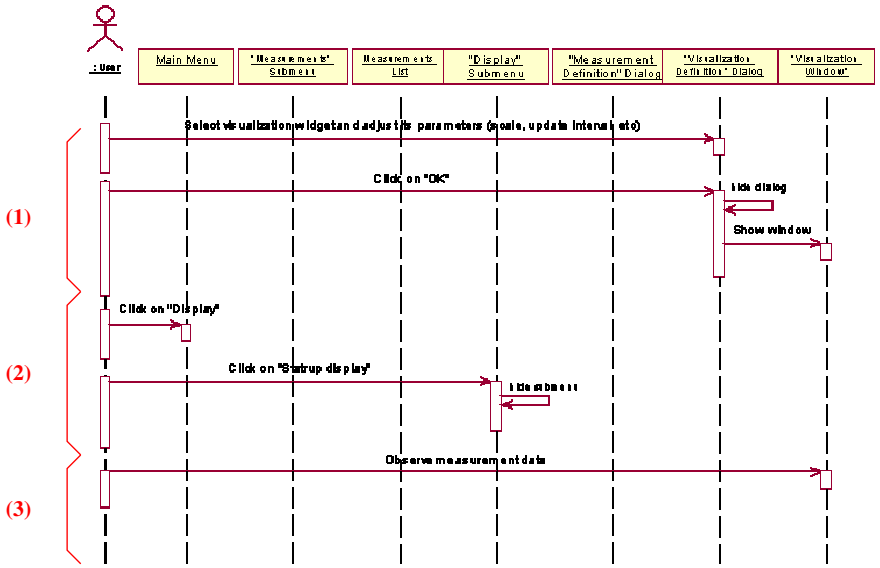

Fig. 3: Using G-PM: sequence diagram of visualization definition procedure

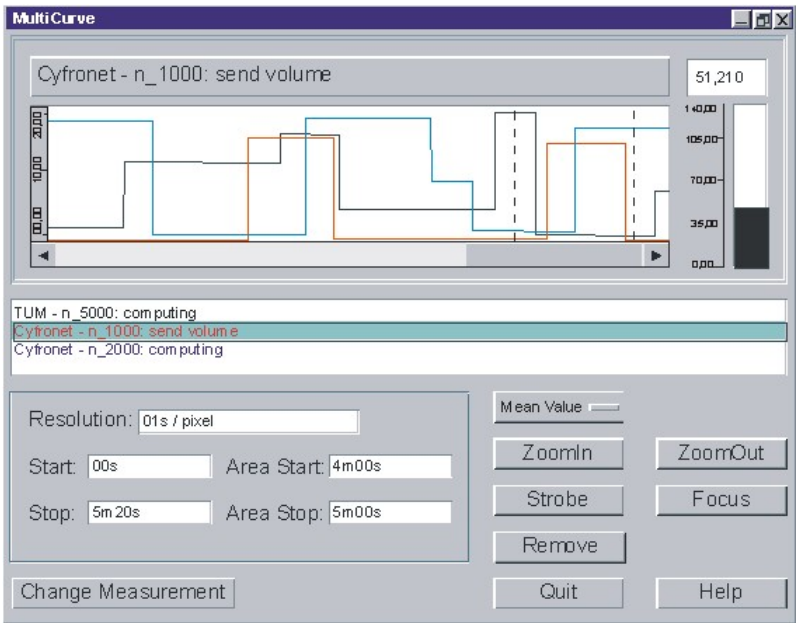

Fig. 4: Visualization of "send volume" measurement with multi-curve display

5 Concluding Remarks

The **first prototype of G-PM** will be available at the beginning of 2003. It will take into account the limitations of the first prototype of the underlying OCM-G tool, e.g. the restriction to a single cluster of workstations. Also, only a subset of standard metrics consisting of send and receive volume and delay time, as well as CPU usage of the application, will be available. Two visualization types will be available, the bar graph and a multi-curve value-versus-time plot. The HLAC component of the tool will support a number of specialized high-level metrics, but will not yet be fully configurable.

The fully functional version of the G-PM tool will be available by the end of 2004. Like all other system software developed in the CrossGrid project, it will then be freely available via a public software license.

Acknowledgements. We are very grateful to Bartosz Baliś, Tomasz Szepieniec, Marcin Radecki from AGH, Kraków as well as to Hamza Mehammed from TUM, Munich for their valuable remarks.

References

1. Balaton, Z., Kacsuk, P., Podhorszki, N., and Vajda, F.: Comparison of Representative Grid Monitoring Tools. Laboratory of Parallel and Distributed Systems (SZTAKI), LPDS-2/2000, 2000
   ftp://ftp.lpds.sztaki.hu/pub/lpds/publications/reports/lpds-2-2000.pdf
2. Balaton, Z., Kacsuk, P., Podhorszki, N., and Vajda, F.: From Cluster Monitoring to Grid Monitoring Based on GRM. In: Sakellariou, R., Keane, J., Gurd, J., and Freeman, L. (eds.), Euro-Par 2001 Parallel Processing, 7th International Euro-Par Conference, August 2001, Manchester, UK, pp. 874-881, vol. 2150, Lecture Notes in Computer Science,Springer-Verlag, 2001
   http://link.springer.de/link/service/series/0558/papers/2150/21500874.pdf
3. Baliś, B., Bubak, M., Funika, W., Szepieniec, T., and Wismüller, R.: An Infrastructure for Grid Application Monitoring. In: Kranzlmüller, D. and Kacsuk, P. and Dongarra, J. and Volkert, J. (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface, 9th European PVM/MPI Users' Group Meeting, September - October 2002, Linz, Austria, 2474, Lecture Notes in Computer Science, 41-49, Springer-Verlag, 2002
4. CrossGrid - Development of Grid Environment for interactive Applications, EU Project, IST-2001-32243, Technical Annex.
   http://www.eu-crossgrid.org
5. Ludwig, T., Wismüller, R., Sunderam, V., and Bode, A.: OMIS — On-line Monitoring Interface Specification (Version 2.0). Shaker-Verlag, 1997, Aachen, Germany, vol. 9, ISBN 3-8265-3035-7
   http://wwwbode.in.tum.de/~omis/OMIS/Version-2.0/version-2.0.ps.gz
6. Malony, A., and Shende, S.: Performance Technology for Complex Parallel and Distributed Systems. In: Kotsis, G., and Kacsuk, P. (eds.), Proc. Third Austrian-Hungarian Workshop on Distributed and Parallel Systems, DAPSYS 2000, 37-46, Kluwer, 2000
   http://www.cs.uoregon.edu/research/paracomp/papers/dapsys2k.ps.gz

7. Miller, B.P., et al.: The Paradyn Parallel Performance Measurement Tools. In: IEEE Computer, vol. 28(11): 37-46, Nov. 1995
http://www.cs.wisc.edu/paradyn/papers/overview.ps.gz
8. Vetter, J.S., and Reed, D.A.: Real-time Monitoring, Adaptive Control and Interactive Steering of Computational Grids. In: The International Journal of High Performance Computing Applications, vol. 14, pp. 357-366, 2000
9. Wismüller, R., Bubak, M., Funika, W., and Baliś, B.: A Performance Analysis Tool for Interactive Applications on the Grid. Workshop on Clusters and Computational Grids for Scientific Computing, September 2002, Le Chateau de Faverges de la Tour, France. To appear.
10. Wolski, R., Spring, N., and Hayes, J.: The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. In: Future Generation Computer Systems, vol. 15, pp. 757-768, 1999