The G-PM Tool for Grid-oriented Performance Analysis *

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-85747 Garching, Germany
[4] Department of Software Science, 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) 289-17676

Abstract. The paper presents the functionality and software design of G-PM, a tool for evaluation of Grid applications performance. Due to the highly distributed and dynamic nature of the Grid, the issues of run-time measurement definition, selective instrumentation, and efficient data collection mechanisms need to be addressed. Providing data on the interaction of distributed application components which is meaningful in the context of an application is essential. The tool offers standard and user-defined measurements as well as means for performance visualization. It gathers 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 a consistent user interface, allowing for 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 the application development cycle. With the introduction of cross-organization computing, the inefficiency in the application behaviour leading to waste of resources becomes a problem of all parties involved in the Grid. This is particularly important within the CrossGrid project [5], 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, e.g. the simulation of vascular blood flow and flooding crisis support applications developed within the CrossGrid project. Therefore, performance analysis is one of the key issues in the project. It is addressed by G-PM, a dedicated performance analysis tool.

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

Compared to existing tools, discussed briefly below, the G-PM features on 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 nature of interactive applications. It allows to extend the standard performance metrics, providing the user with a way to specify some higher-level, more abstract metrics suitable in the application context. Contrary to some previously developed tools, the gathered performance data includes the characteristics of the application environment and the Grid infrastructure.

The paper is arranged as follows. Section 2 presents a concise analysis of the existing Grid performance analysis tools. In Section 3, the functionality of the tool is presented. Then, in Section 4, the architecture and design of the tool, as well as the interaction with other parts of the CrossGrid infrastructure are studied. Section 5 presents some details on the idea of user defined metrics based on the probe concept. Finally, Section 6 informs on the development schedule of the tool and on the features included in the first prototype.

2 Related work

Currently, a number of performance tools already adapted to the Grid [1] exist. Most of those supporting the monitoring of applications are based on an off-line analysis of event traces. This is not suitable for interactive applications, since here the opportunity to relate on-line the performance data with the user's interactions is essential. However, on-line Grid tools are available mainly for infrastructure monitoring. An example is the Network Weather Service [11] that measures and forecasts network and node performance. An exception is the monitoring system GRM [2], adapted to the Grid within the Datagrid project[1]. It 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 of them. Autopilot [9] exploits a concept called *sensors* corresponding to our probes. User-defined instrumentation is used in the TAU performance analysis environment [7]. 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 [8] belongs to the few tools that strictly exploit the on-line approach, including the instrumentation procedure, but it does not support user-defined metrics.

3 Tool Functionality vs. Grid

In this section we relate the functionality of performance tools to the features of Grid computing and show the designed functionality of G-PM.

[1] `www.eu-datagrid.org`

Specifics of interactive Grid computing. 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 meaningful in the application steering process, such as performance data, have to be available on-line. Next, the interpretation of the performance data in a heterogeneous, open Grid system, can only be done in the context of the application's environment. Thus, the performance characteristics of the Grid infrastructure itself are required. Finally, the user of the application may be a specialist other than a computer scientist. Hence, the information provided to the user must not be limited to low-level data (e.g. data volume exchanged between two processes), but rather be more high-level and application specific. The tools have to be configurable, extendible and application-oriented.

Basic performance metrics. The G-PM tool aims at satisfying the outlined requirements and constraints. It operates in the on-line mode, displaying current performance data in form of various visualization graphs. Some basic standard metrics, or measurable quantities, have been specified, both application and Grid infrastructure related. The first allow for a low-level analysis of the application behaviour (e.g. data volume of MPI or file operations). Drilling down is allowed by restricting the analysis to a subset of the application processes, hosts, or to specific regions in the source code. The Grid infrastructure metrics specify both dynamic (e.g. CPU or network load) and static (e.g. host memory size) information on the application environment.

User-defined metrics. The standard 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. The *first* way is to add some specific events to the application code. These events, called *probes*, then allow to limit any measurement to the specific, relevant program phases. Probes may also carry some additional, application-specific information to the G-PM, which can then be presented by the tool using the standard visualizations. The *second* way, which aims at customizing the performance data presented to the user, is to use a metrics specification language. The user can build some higher-level metrics by combining and transforming the standard metrics, the metrics based on probes as well as other previously specified higher-level metrics. 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.

4 The G-PM Tool Design

The architecture of G-PM is designed as part of the programming tools environment communicating to the monitoring and other Grid services via the OCM-G monitoring system (see Fig. 1) only. Below we discuss particular aspects of the tool's design.

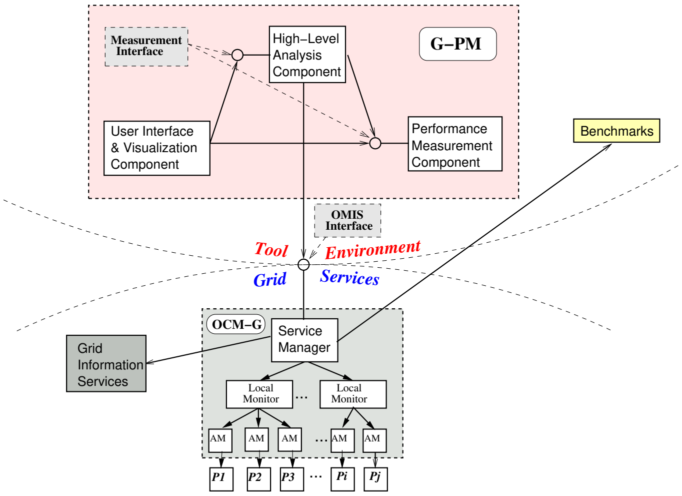

Fig. 1. Structure of G-PM

4.1 Monitoring system

To provide performance measurements in the on-line mode, G-PM is built on top of the OCM-G [3] monitoring system, an implementation of the On-line Monitoring Interface Specification OMIS 2.0 [6]. OCM-G, which is part of the Grid services, is responsible for supplying low-level monitoring data. This includes the communication with instrumented libraries linked to the application, and the gathering of information supplied by these libraries and other monitoring data sources. Also, the results of micro-benchmarks are accessible through OCM-G.

OCM-G comprises two types of components: Service Managers and Local Monitors (see the lower part of Fig. 1). Service Managers (SM) are the permanent part of OCM-G and there is one SM per site of the Grid. An SM distributes requests to Local Monitors, if the target object is located on the same site, and/or to the other SMs in order to reach objects located outside the site. Local Monitors (LMs) are transient components in OCM-G responsible for execution of OMIS requests; these are created on each host of the Grid where application processes to be monitored exist.

The G-PM tool is implemented as a process running on a user's local workstation. It connects to the nearest Service Manager, to which it submits monitoring requests and from which it receives the corresponding replies (see the upper part of Fig. 1). Once the tool is connected to OCM-G, it may get information about all running applications. Then an application can be chosen to which the tool should *attach*. The application can be monitored only upon the tool have attached to it. The tool can attach to a running application and begin monitoring at any time, provided that the application is linked with an instrumented library.

4.2 Supply of monitoring data

Within G-PM performance measurements are defined and presented while the application is running. The results of measurements are numerical values evolving over time. G-PM either displays the current values as e.g. bar graphs or pie charts, or their temporal development in form of a curve at regular intervals. There are principally two kinds of accessing the monitoring data to enable measurements: automatically and user-defined.

Instrumentation. The G-PM offers a wide variety of standard metrics for performance analysis. These include application-related metrics, like e.g. time elapsed, or volume of interprocess communication and I/O. In addition, the tool allows to measure infrastructure-related metrics, like e.g. node load or link bandwidth. This information is provided and forwarded from proper Grid services. To access the data on standard metrics, e.g. `send data volume`, at a preparatory step the application has to be re-linked with instrumented versions of some run-time libraries. The instrumentation consist of wrappers around the original routines (e.g. communication or I/O) that acquire the necessary performance information. In order to enable the instrumentation, the user has to start the application with a special command line argument handled by the instrumentation code. Additional command line argument can be used to prevent the program from running until there is an explicit request. This allows to define measurements before the application starts executing.

As a rule, measurements are based on counters and integrating timers [3], which are updated by the instrumentation code within the context of the local application process. Consequently, for a typical measurement only a limited number of values has to be read, which results in a low overhead. In cases where very detailed information is required and the higher overhead is acceptable, events can be logged, too.

Probes vs. user defined metrics. As mentioned in Section 3, a performance tool must also be able to provide high-level performance data which is more meaningful in the application's domain. The application-specific data can be classified into three categories: (1) particular, relevant events, meaningful for the application user/developer, e.g. the start/end of a user interaction, (2) correlated events, e.g. start/end events in different processes that indicate the entry into/exit from the same computational phase, (3) performance data computed within the application itself, e.g. the residuum value in a numerical solver.

To provide the tool with this information, the user inserts a set of *probes* into the application's source code. A probe is a special function call, that signals the event connected with the particular position in the source code. This call can receive user-provided parameters that contain application-specific data. If the G-PM does not need the data provided by the particular probe the underlying instrumentation is disabled. Consequently the introduced overhead is low.

Using a probe does not yet imply a specific user-defined metrics. In fact, the same probe can be used to define many different metrics. Based on the concept of probes, G-PM allows the user to define his/her own metrics, which can be built on already existing metrics (standard ones and user-defined ones) and/or any available probes and probe data in the application.

4.3 Component Structure of G-PM

The tool is composed of three main modules, the *Performance Measurement Component* (PMC), the *High-Level Analysis Component* (HLAC) and the *User Interface and Visualization Component* (UIVC), see Fig. 1.

Performance Measurement Component. The PMC implements all standard metrics. This is done by properly programming the monitoring system when a measurement is defined. See [4] for a more detailed description.

High Level Analysis Component. Since the HLAC handles the user-defined metrics it is involved in parsing the metrics specifications and translating the high-level metrics into requests to PMC for the underlying standard metrics. Also, it is responsible for handling the events generated by *probes* and for processing the information passed by these probes from the application to the G-PM tool. While the HLAC reuses the PMC for performing standard measurements, it also communicates directly with OCM-G through the OMIS interface in order to efficiently implement probe-based metrics. 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 OCM-G in an distributed way, instead of within the G-PM tool. For more details on the issue please refer to [10].

User Interface and Visualization Component. The User Interface comprises the program's *main window*, which presents a list of defined measurements and visualizations. Another important part is the *measurement definition window* that allows to choose a metrics and to specify its parameters e.g. restrictions on the hosts/code regions where the measurement should be performed. Once the measurement is defined, the user can specify the type and parameters of the measurement's *visualization window* using the *visualization definition window*. The tool provides, a set of *visualization windows*, ranging from bar graphs and pie charts to value versus time function plots.

5 User-defined Metrics

As mentioned in Section 4, the HLAC allows to measure application-specific metrics based on user-defined instrumentation of the application's source code. Among others, the following kinds of user-defined metrics are possible: (1) a metrics which is defined by an existing metrics (e.g. standard metrics provided by the PMC), measured only during an execution phase delimited by any two probe executions; (2) a metrics which can be defined by any parameter of any probe; (3) a metrics which can be derived from any existing set of metrics by aggregating or comparing their values.

An example of a metrics specification of the first kind is shown in Fig. 2. In this example the programmer inserted two probes – *begin* and *end* – into the

```
IO_volume_for_interaction(Process[] processes, File[] files,
                          Region[] regions, TimeInterval currTime)
{
  Process p; VirtualTime vt;
  Value volume[][]; Value globalVol[];
  PROBE begin(Process, VirtualTime); PROBE end(Process, VirtualTime);
  volume[p][vt] = IO_volume(p, files, regions) AT end(p, vt)
                - IO_volume(p, files, regions) AT begin(p, vt);
  globalVol[vt] = SUM(volume[p][vt] WHERE p IN processes);
  RETURN SUM(globalVol[vt] WHERE globalVol[vt].time IN currTime);
}
```

Fig. 2. Example of a user-defined metrics

application, to mark the beginning and end of an end-user interaction. Based on these probes and the standard metrics `IO_volume` for the total volume of disk I/O, the user has defined a new metrics for the disk I/O during a single end-user interaction. For each process `p`, the value of this metrics is defined as the difference of `IO_volume` at the *end* event and `IO_volume` at the corresponding *begin* event, where `IO_volume` is the accumulated volume since the start of the measurement. The following two assignments in the metrics specification define how the result is accumulated over space (i.e. the processes) and time.

Since the HLAC is aimed at the optimised implementation of measurements based on user-defined metrics, the processing of probe events should happen as locally as possible, i.e. in the best case within the context of the appropriate process. This implies that the evaluation suggested by the metrics specification should be performed in a distributed way, as discussed in [10].

6 Concluding Remarks

The first prototype of G-PM is available since February 2003. Due to the limitations of the OCM-G, this prototype it is restricted to a single cluster of workstations. Also, only a subset of standard metrics is available (send/receive volume, delay time, application CPU time). Two visualization types are implemented: the bar graph and a multi-curve value-versus-time plot. The HLAC component of the tool currently supports two specialized types of high-level metrics, which are configurable to a limited extent. The fully functional version of the G-PM tool is currently being implemented and 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. Bubak, M., Funika, W., and Wismüller, R.: The CrossGrid Performance Analysis Tool for Interactive Grid Applications. 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, 50-60, Springer-Verlag, 2002
5. CrossGrid - Development of Grid Environment for interactive Applications, EU Project, IST-2001-32243, Technical Annex.
   http://www.eu-crossgrid.org
6. 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
7. 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
8. 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
9. 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
10. 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 Faberges de la Tour, France. To appear.
11. 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