Monitoring Grid Applications with Grid-enabled OMIS Monitor*

Bartosz Baliś[1,2], Marian Bubak[1,2], Włodzimierz Funika[1,2], Tomasz Szepieniec[2], Roland Wismüller[3], and Marcin Radecki[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 – Technische Universität München, D-80290 München, Germany
{balis,bubak,funika}@uci.agh.edu.pl, t.szepieniec@cyf-kr.edu.pl
wismuell@in.tum.de, m.radecki@cyf-kr.edu.pl
*p*hone: (+48 12) 617 39 64, *f*ax: (+48 12) 633 80 54, *p*hone: (+49 89) 289-28243

Abstract. In this paper, we present our approach to monitoring Grid applications with a Grid-enabled OMIS Monitor – the OCM-G. The OCM-G is a monitoring infrastructure for tools supporting application development, and it provides various monitoring services to obtain information about, manipulate, as well as detect events in an executing application. The services are accessible via the standarized interface – OMIS (On-line Monitoring Interface Specification). We describe the architecture of the system and present some design details important for the monitoring system to fit well the Grid environment, and support monitoring of interactive applications.

Keywords: Grid, monitoring, services, interactive applications, OMIS

1 Introduction

Monitoring services are indispensable part of each Grid environment. These services may be focused both on the Grid infrastructure and the activity of running applications. In our view, these two different types of monitoring can be characterized as follows.

– *infrastructure monitoring* collects information about Grid components, such as hosts or network connections; this information is indispensable for basic Grid activities as resource allocation or load balancing; often this type of information has historic value, thus it is stored in a database for a later analysis (e.g., statistical, forecasting, etc.),
– *application monitoring* aims at observing a particular execution of an application; the collected data is useful for tools for application development support, which are used to detect bugs, bottlenecks or just visualize the application's behaviour; this kind of information in principle does not have historic value – it is meaningful only in the context of a particular execution.

* This work was partly funded by the European Commission, Project IST-2001-32243, CrossGrid [7]

A few efforts exist to address application monitoring in the Grid: GrADS / Autopilot, DataGrid / GRM, GridLab / GRM.

The Autopilot toolkit [17] in the GrADS project [10] is oriented towards automatic performance tuning based on behavior patterns. This is a view rather different than ours, since we are interested in providing feedback to the user who is intersted in a particular performance loss.

The application monitoring system developed within the GridLab project [1] implements on-line steering guided by performance prediction routines deriving results from low level, infrastructure-related sensors (CPU, network load). The system is different to ours in the following respects. First, the manipulations on the target application are not supported. Second, the system seems to rely only on full traces, at least at the current state of development. Finally, the predefined semantics of all metrics requires all Grid users to agree on this, which is rather restrictive [12].

The DataGrid project [8] introduces the GRM monitor [14]. The system collects information semi-on-line and delivers it to the R-GMA, a relational information infrastructure for the Grid [15]. Monitoring in GRM/PROVE is mainly based on event tracing. While the GRM/PROVE environment is well suited for the DataGrid project, where only batch processing is supported, it is less usable for the monitoring of interactive applications. First, due the high communication latency introduced by the R-GMA. Second, since it is not possible to achieve low latency and low intrusion at the same time when monitoring is based on trace data. If the traces are buffered, the latency increases, if not, the overhead for transmitting the events is too high.

In this paper, we describe the OCM-G – Grid-enabled OMIS Monitor, a versatile infrastructure for on-line application monitoring designed to support interactive applications.

2 Monitoring of (Interactive) Applications in the Grid

In Fig. 1, layers of a monitoring environment are presented. The bottom layer represents objects to be monitored, e.g., application processes, but also Grid infrastructure, e.g., CPU and network. In the top layer there are various tools, interactive, e.g. for performance analysis or debugging, as well as automatic, for example for automatic load balancing via process migration. Between tools and the monitored objects there is a monitoring infrastructure.

The Grid environemnt poses new requirements for an application monitoring infrastucture. Even more requirements are important when we consider interactive applications. Below we assemble a list of the most important requirements for a Grid application monitoring system, especially to support interactive applications.

- **On-line mode.** On-line monitoring greatly reduces the data rate, since it allows to specify the information of interest at run-time. In contrast, in off-line monitoring, the measurements and corresponding monitoring information must be specified in advance. In addition, in case of tracing, the off-line

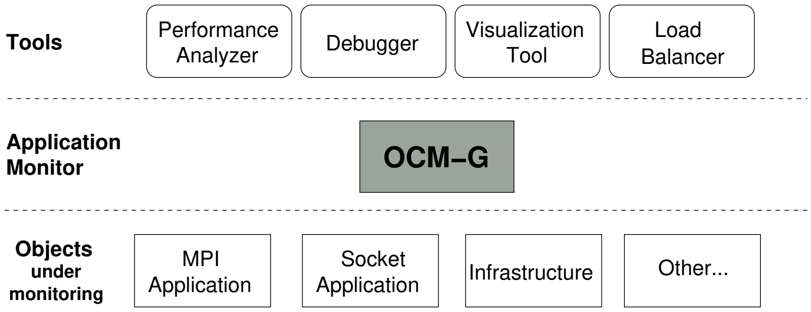

Fig. 1. Layered structure of application monitoring environment

approach involves huge trace files. Moreover, for some monitoring activities, such as manipulations (debuggers), on-line operation is essential. For interactive applications on-line monitoring is important for one more reason: the user wants to relate the performance to his interactions, i.e., immediately see the impact of his activities.

– **Efficiency and scalability.** The Grid may be a highly distributed system, and the running applications may be composed of a large number of processes spread across multiple sites. Consequently, the monitoring infrastructure should be designed to handle high data rates and scale to large number of monitored objects.
– **Low intrusiveness.** Each monitoring activity introduces a probe effect which may distort the normal execution of the application. Especially in case of interactive applications, the overhead due to monitoring should be kept as low as possible so that, e.g., low latency is sustained.
– **Transparency for the user.** This requirement includes several aspects:
  • The user should not need to know the architecture or details of the operation of the monitoring system. All he is supposed to do is to send monitoring requests and collect replies.
  • The instrumentation of the code, essential e.g. for performance metrics, should be as much automatic as possible.
  • The compilation of the application to enable monitoring should be as easy as to add some additional parameters to the usual compilation command.
  • The submission of the application (via portal, script, etc.) should not change due to monitoring with the exception, perhaps, of additional command-line parameters.
  • Tools should be able to attach to an application and monitor it at any point of its execution.
– **Grid service.** In a grid evironment, the application monitoring should be available as a kind of grid service. This means the monitoring system should run permanently, there should be a possiblity to discover it and request some monitoring services.

- **Security.** The monitoring system should ensure the desired level of security, i.e., user authentication and authorization, integrity and confidentiality of monitoring information (if desired), etc.

3 OMIS – a Universal Monitoring Interface

As an intermediate layer, the monitoring system has two interfaces: one towards the tools, the other one towards the running application. While the monitor/application interface is strongly system-dependant, since typically OS services are involved, the tool/monitor interface may be standardized. An example of such a standardized tool/monitor interface is OMIS (On-line Monitoring Interface Specification) [13].

The target system, from OMIS point of view, forms a hierarchy of objects. In the Grid these are *sites*, *nodes* and *processes*. The objects are uniquely identified by so called *tokens*.

OMIS defines a variety of monitoring services which are divided into three classes: *information* services – to obtain descriptive data about objects, *manipulation* services – to change objects' state, and *event* services – to detect events in the target system (especially inside applications). The information and manipulation services are also called *actions*.

By combining these services one obtains a *monitoring request* to perform some monitoring activities. The monitoring request can belong to one of two types: *unconditional* and *conditional*.

- The unconditional service requests are composed of one or more information and/or manipulation services (referred to as actions). They result in an immediate action in the target system, for example *stop process p*; in OMIS syntax: `thread_stop([p])`.
- The conditional service requests (CSR) are composed of an event service and a list of actions. Their semantics is as follows: whenever the event occurs, execute the actions, for example *whenever process p starts a new process, get information about that process*; in OMIS syntax:
`thread_creates_proc([p]): proc_get_info([$newproc], 0x1)`.

4 Grid-enabled OMIS Monitor – OCM-G

In this section, we present the OCM-G – a Grid-enabled application monitoring system based on OMIS. First we describe the architecture of the system, then we provide some design details. Finally, we outline the monitoring functionality provided by the OCM-G.

4.1 Architecture

The OCM-G is a decentralized, distributed system, which is a collection of two types of components: *Service Managers* and *Local Monitors*.

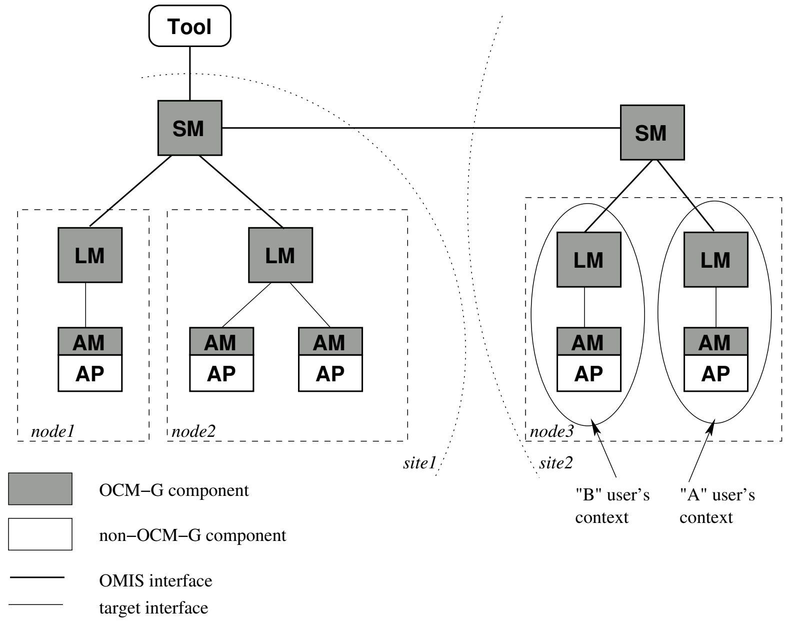

Fig. 2. OCM-G components distributed in a Grid environment

Service Managers (SM) reside permanently, one on each site of the Grid. They are the part of the OCM-G which exposes the monitoring services to tools. Since the SMs are permanent they can be well known and thus the OCM-G can work as a Grid service. The SMs distribute OMIS requests to appropriate Local Monitors on the same site.

Local Monitors (LM) are created on each host of the Grid where there are application processes to be monitored. LMs execute OMIS requests accepted from SMs and send the results back. LMs handle only objects which are on the same host. Because of security issues we decided create on LM per each (host, grid user) pair.

Some parts of the monitoring system are also embedded in the application; these are referred to as Application Module (AM). The AM is necessary to perform monitoring activities in the context of the application, it is essential for reducing data rate (buffering of monitoring information), etc.

Protocol for SM-SM and LM-SM communication is based on OMIS, while LMs and AMs share a common memory segment to communicate. The LMs use OS mechanisms (ptrace or the proc file system) to control the application processes.

Fig. 2 shows how the components of OCM-G are distributed in the Grid.

Beside this physical distribution, the OCM-G has also a logical structure – it is divided into so called Virtual Monitoring Systems (VMS). One VMS is created for each monitored application and is composed of all OCM-G compo-

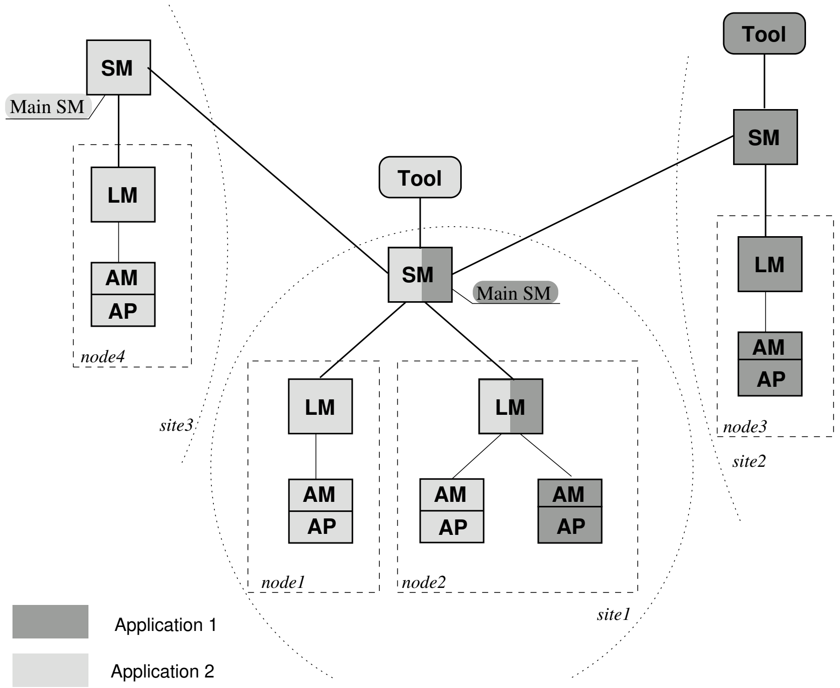

Fig. 3. Two Virtual Monitoring Systems sharing the same OCM-G components

nents involved in this application. This concept is introduced to improve the scalability of the OCM-G. The information about the application is only spread across the VMS, which is usually a small part of the whole OCM-G. Moreover, the connections between the SMs are created only if they are parts of the same VMS. Obviously, each LM, SM or a connection between them may belong to more than one VMS. To handle information management in a VMS, one of its SMs is assigned a special role, a so called MainSM. The MainSM plays a role of a central information service for this VMS. Fig. 3 shows an example of two VMSs sharing physical OCM-G components. Note that in the case shown in the figure, the two VMSs belong to the same user, thus not only the SM, but also the LM is shared between them.

4.2 Design details

In this section, we describe some design details in the OCM-G important to fit the requirements of the Grid environment and interactive applications, described in section 2.

- **Efficiency.** To ensure efficient monitoring, we try to minimize the data rate. This is done in principle with two techniques:

  - **Local buffering.** The monitoring information is buffered in the context of the application. Only on an explicit demand (pull model), it is extracted by the LM, and sent to be obtained by a tool.
  - **Distributed evaluation.** To further reduce the data rate, not only is it buffered, but also processed locally and stored in counters or integrating timers. Additionally, we provide the counters and integrators as distributed data structures, i.e. they may be composed of a global object, and some local components. The information stored in local components is on demand collected and combined in the global object.
- **Scalability.** The scalability in the OCM-G is in principle ensured by its decentralized and distributed architecture. However, to avoid problems in sharing information in such a highly decentralized system, and to avoid the necessity for each component to know about each other (which would make the system unscalable again), we introduced the concept of Virtual Monitoring Systems which are centralized logical monitoring systems for each application. This concept was described in the previous subsection.
- **Controllable intrusiveness.** To minimize the monitoring intrusiveness, we gather the information via **selective run-time instrumentation**. This approach to instrumentation combines the benefits of dynamic and static instrumentation. In general, the *hooks* are inserted statically in the application, while the actual instrumentation code is "hooked up" (hence the name) dynamically. This may be viewed as dynamic activation and deactivation of instrumentation on demand. Our tests showed that the overhead of inactive instrumentation is "in the noise". Moreover, due to the on-line approach to monitoring, the number of active instrumentation may be kept minimal at a given moment.
- **Transparency for the user.** This has several aspects in the OCM-G:
  - The instrumentation is based on binary wrapping, i.e., the user is provided with pre-instrumented versions of communication libraries (e.g., MPI). Thus, only relinking of the application is necessary to enable monitoring of communication subroutines. While this is enough for most performance metrics, the user can also define events in an arbitrary place of the code. This is the only case in which the user has to insert the appropriate code manually. However, even this is greatly simplified, since the code to be inserted is just a single function call.
  - The compilation of the application is handled automatically by a tool provided with the OCM-G. The user issues the normal command to compile the application, only preceded by the name of the tool.
  - The application is submitted to the Grid in the usual way, only a few additional command line parameters are necessary.
  - Tools may be connected to a running application at any time of its execution.
  - It should be mentioned that we require each process of the application to invoke a special function to register in the OCM-G. However, in case of MPI even this is transparent to the user, as the call to the function is hidden in the instrumentation of MPI_Init().

– **Security.** The OCM-G incorporates security at three levels:
  - users are authenticated using Grid certificates, while communicating with SMs,
  - OCM-G allows to monitor an application to its user only,
  - Local Monitors run as user processes, thus they are limited by the OS protection mechanisms.
– **Operation as a Grid service.** The Service Managers are designed to run permanently on the Grid and are well-known. This gives good prerequisites to run the OCM-G as a Grid service.
– **Versatility and flexibility.** It should also be noted that the OMIS/OCM-G approach enables high versatility and flexibility in monitoring.
  - The monitoring services in the OCM-G are various and support different kinds of tools. Moreover, the set of services is extendible by dynamically loaded extensions. Currently available monitoring services are described in section 4.3.
  - The services are relatively low-level, i.e., instead of high-level metrics they return a low level information. Different pieces of such information can be then combined into metrics. For example, instead of a high level request to obtain communication delay for MPI_Send(), one would issue a sequence of simple and lower-level requests: two requests returning time stamps for two events – beginning and end of invocation of MPI_Send(), and some requests to create an integrator and evaluate the delay properly from the time stamps. All this would be done in the context of the application, so there would be no overhead related to multiple requests. This approach allows to construct the metrics by the user with the semantics he needs, it is very easy and much more flexible than providing a set of metrics with a fixed semantics.

4.3 Monitoring Services

To make the description complete, in this section we overview the available monitoring services in the OCM-G. The OCM-G provides all monitoring services defined by the OMIS 2.0 specification, except for those which were undesirable in a Grid environment (e.g. creation of a new process). Additionally, new services are provided, specific for the Grid, or extending monitoring capabilities. Among others, the following functionality is provided by the OCM-G:

1. Debugging services. This includes services for suspending/continuing processes, reading/writing processes' memory, etc.
2. Performance analysis services, for example:
   – services for detecting beginnings and ends of function calls,
   – services for handling probes, which are used by the user to define and detect arbitrary events in an application,
   – services for creation and management of counters and integrators – efficient data structures which allow to buffer and preprocess monitoring information in the context of the application.

3. Services for infrastructure monitoring. Some services return information about the current state of infrastructure elements, e.g., examine the status of a network connection. This information may be useful, e.g. in performance analysis – when a user notices a performance loss, he might be interested whether it is caused by the network load or a problem with the algorithm.

5 Previous Experience

Our experience in monitoring and tools goes back to 1995 when the OMIS specification was defined at the Technical University in Munich. In 1997 the OCM – an OMIS-Compliant Monitor [18] for clusters was implemented as well as a few OCM-based tools were developed (such as performance analyzer PATOP and debugger DETOP). Since 1997, there is a collaboration between LRR-TUM and Institute of Computer Science AGH, Cracow. The result of this cooperation is the continuous development and improvement of the monitoring system and tools, among others, towards monitoring and performance analysis of Java applications, threaded applications on shared memory machines, and Grid applications.

6 Summary and Status

In this paper we presented the OCM-G – a universal and flexible monitoring infrastructure for Grid applications. We have shown that the OCM-G is well suited to work in the Grid environment and supports well interactive applications.

Currently the first prototype of the OCM-G is completed and has an operational status. The implementation of this prototype is based on the OCM. The first prototype works on one site only, and supports only one user. It provides all services defined by the OMIS 2.0 document, except for those which were unnecessary or undiserable in the Grid (e.g., for creation of a new process). Some of the new Grid-specific services were also impelmented. The first prototype of the OCM-G implements the new Grid-enabled start-up scheme.

References

1. Allen, G., Davis, K., Dolkas, K., Doulamis, N., Goodale, T., Kielmann, T., Merzky, A., Nabrzyski, J., Pukacki, J., Radke, T., Michael, M., Seidel, E., Shalf, J. , and Taylor, I.: Enabling Applications on the Grid: A GridLab Overview. International Journal of High Performance Computing Applications: Special issue on Grid Computing: Infrastructure and Applications, to be published in August 2003.
2. Balaton, Z., Kacsuk, P., Podhorszki, N., and Vajda, F.: Comparison of Representative Grid Monitoring Tools. `http://www.lpds.sztaki.hu/publications/reports/lpds-2-2000.pdf`
3. Balis, B., Bubak, M., Funika, W., Szepieniec, T., and Wismuüller, R.: An Infrastructure for Grid Application Monitoring In: Kranzlmueller, D., Kacsuk, P., Dongarra, J., Volker, J. (Eds.): Recent Advances in Parallel Virtual Machine and

Message Passing Interface, Proc. 9th European PVM/MPI Users' Group Meeting, Linz, Austria, September/October 2002, LNCS 2474, pp. 41-49, 2002

4. Bubak, M., Funika, W., Baliś, B., and Wismüller, R.: On-line OCM-based Tool Support for Parallel Applications. In: Yuen Chung Kwong (ed.): Annual Review of Scalable Computing, 3, Chapter 3, 2001, Singapore
5. Bubak, M., Funika, W., and Wismüller, R.: The CrossGrid Performance Analysis Tool for Interactive Grid Applications. In: Kranzlmueller, D., Kacsuk, P., Dongarra, J., Volker, J. (Eds.): Recent Advances in Parallel Virtual Machine and Message Passing Interface, Proc. 9th European PVM/MPI Users' Group Meeting, Linz, Austria, September/October 2002, LNCS 2474, pp.50-60, 2002
6. GrossGrid - Development of Grid Environment for Interactive Applications. Annex 1 - description of Work, available at: http://www.eu-crossgrid.org
7. The CrossGrid Project: http://www.eu-crossgrid.org
8. The DataGrid Project: http://www.eu-datagrid.org
9. Foster, I., Kesselman, C. (eds.): The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999
10. The GrADS Project: http://hipersoft.cs.rice.edu/grads
11. The GridLab Project: http://www.gridlab.org
12. GridLab deliverable 11.3: Grid Monitoring Architecture Prototype. `http://www.gridlab.org/Resources/Deliverables/D11.3.pdf`
13. Ludwig, T., Wismüller, R., Sunderam, V., and Bode, A.: OMIS – On-line Monitoring Interface Specification (Version 2.0). Shaker Verlag, Aachen, vol. 9, LRR-TUM Research Report Series, 1997. http://wwwbode.in.tum.de/~omis.
14. Podhorski, N., Kacsuk, P.: Design and Implementation of a Distributed Monitor for Semi-on-line Monitoring of VisualMP Applications. Proc. DAPSYS 2000, Balatonfured, Hungary, 23-32, 2000
15. R-GMA: A Grid Information and Monitoring System. `http://www.gridpp.ac.uk/abstracts/AllHands_RGMA.pdf`
16. Tierney, B., Aydt, R., Gunter, D., Smith, W., Taylor, V., Wolski, R., Swany, M., et al.: White Paper: A Grid Monitoring Service Architecture (DRAFT), Global Grid Forum. 2001. http://www-didc.lbl.gov/GridPerf
17. Vetter, J.S., and Reed, D.A.: Real-time Monitoring, Adaptive Control and Interactive Steering of Computational Grids. The International Journal of High Performance Computing Applications, 14 357-366, 2000.
18. R. Wismueller, J. Trinitis, T. Ludwig: OCM – A Monitoring System for Interoperable Tools. In: Proc. 2nd SIGMETRICS Symposium on Parallel and Distributed Tools SPDT 98, Welches, OR, USA, August 1998.