This paper was published in
Lecture Notes in Computer Science
(http://www.springer.de/comp/lncs/index.html)

An Infrastructure for Grid Application Monitoring*

Bartosz Baliś[1], Marian Bubak[1,2], Włodzimierz Funika[1], Tomasz Szepieniec[1], and Roland Wismüller[3,4]

[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
[4] Institute for Software Sciences, University of Vienna, A-1090, Wien Austria
{balis, bubak, funika}@uci.agh.edu.pl, szepieniec@icsr.agh.edu.pl, wismuell@in.tum.de
*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 a concept of the OCM-G — a distributed monitoring system for obtaining information and manipulating distributed applications running on the Grid. The purpose of this system is to provide a basis for building tools supporting parallel application development. We present Grid-specific requirements, propose a suitable architecture for the monitoring system, and describe the integration of the OCM-G with Grid services. OMIS is applied as an universal interface between the OCM-G and tools.

Keywords: Grid, monitoring, services, performance analysis, tools

1 Introduction

With the recent emergence of the Grid [8–10], application and tool-environment developers are faced with more complicated problems than in e.g. cluster computing. Grid resources are unprecedented both in amount and geographical distribution. The current solutions, mostly designed for clusters or supercomputers, cannot meet the scalability and efficiency requirements needed on the Grid. Tools for Grid environments are still at an early stage of development, however a foundation for a Grid tool environment has already been laid with the definition of a Grid Monitoring Architecture (GMA) [18]. This specification defines the monitoring architecture on the Grid in terms of the producer-consumer model.

A comparison of grid-oriented monitoring tools is given in [1]. The Globus Heartbeat Monitor [14] supports monitoring the state of processes. It reports failures of processes and provides notification of process status exception events. Network Weather Service [21] periodically monitors performance of networks and computational resources and forecasts this performance. The Information Power Grid project is developing its own monitoring infrastructure [20] which

* This work was partly funded by the European Commission, project IST-2001-32243, CrossGrid [6]

consists of sensors, actuators, and event services; XML is used to represent monitoring services. NetLogger [13] is designed for analysis of communication inside applications. GRM and PROVE [16] may be used for performance monitoring of applications running in distributed heterogeneous systems. Autopilot/Virtue [19, 17] is a performance measurement and resource control system. The DataGrid project [7] is developing monitoring systems oriented towards applications which operate with large data files; recently they proposed Relational Grid Monitoring Architecture which will be further developed within the Global Grid Forum.

The subject of this paper is an application monitoring system for the Grid called OCM-G. The OCM-G is meant to enable investigation and manipulation of parallel distributed applications running on the Grid, and provide a basis for building tools supporting parallel application development on the Grid.

In [4] and next, in [5], we considered alternatives for an architecture of the Grid application monitoring environment. This paper presents more detailed and mature concepts for our application monitoring system which is being developed in the framework of the CrossGrid project [6], where it will supply data to the programming tool environment [3]. This environment is being developed on the basis of our previous research [2] and on results of the APART project [11].

2 Monitoring of Grid Applications

Our approach differs from other approaches to Grid monitoring in several aspects. First, it is *application oriented* – it provides information about *running* applications. It works *on-line* – the information is not normally stored anywhere for later use but immediately delivered to its consumers (tools) and processed. The information is collected by means of *run-time instrumentation* which can be activated or deactivated on demand. Thus, data to be gathered can be defined at run-time which results in a reduced overhead of monitoring, since we can focus on the data we are interested in at the given moment.

Using the OCM-G has several benefits. As an autonomous monitoring infrastructure, the OCM-G provides abstraction and increases modularity — the tools can be developed independently from the monitoring system. The standardization of the communication interface between the OCM-G and tools also gives preconditions for porting the existing tools compliant with the interface to the Grid.

The interface in question is the *On-line Monitoring Interface Specification* (OMIS) [15]. In our experience with OMIS and an OMIS-compliant monitoring system for clusters of workstations, the OCM, we learned that OMIS enables to adapt existing tools to new programming paradigms, such as PVM or MPI, with minimal changes to the tools' code [2].

3 Monitoring System Interface

Based on our experience in developing an OMIS-based tool environment [2], we decided to use OMIS [15] as a communication interface between the OCM-G and tools. In OMIS, the target system is viewed as a hierarchical set of objects. Currently defined objects are *nodes*, *processes*, *threads*, *messages* and *message queues*. The objects are identified by symbols called *tokens*, e.g. *n_1*, *p_1*, etc.

The functionality of the monitoring system is accessible via *services* which are divided into three classes: *information services* to obtain information about objects, *manipulation services* to manipulate objects, and *event services* to detect events and program actions to be executed whenever a specified event occurs.

The OCM-G will accept all requests compliant with OMIS 2.0. This includes two type of requests: unconditional and conditional. The unconditional requests have a single effect on the target system and yield an immediate response. They are composed of one or more actions which specify either information to be returned or manipulations to be performed. For example, the request *stop process* `p_1` will attempt to stop the process identified by `p_1`, while the request *return process list for node* `n_1` will return a list of processes[1] residing on node `n_1`. All requests return a status information (whether the request succeeded or not).

The conditional requests specify an event and a list of actions which shall be executed whenever the event occurs. Thus, responses to conditional requests can be produced multiple times, each time the specified event takes place. Example: *when function X is invoked, return the time stamp of this event.*

OMIS should be extended to fit the Grid requirements and architecture. First, the object hierarchy should be changed to reflect the complex Grid structure. We propose to add a new layer at the highest level of the OMIS object hierarchy — *sites* which are meant to identify administratively uniform domains and encompass multiple systems such as clusters or supercomputers. This extensions will be appropriately reflected in the OCM-G architecture (see 4.2).

Other extensions directly concern the monitoring system's functionality which is reflected in new services. Possible extensions include security-related services, e.g. for authentication, services required for interoperability with the Grid information system or other Grid services [12, 10], and also services required for monitoring new types of events such as generic I/O operations. We also reason that mechanisms already existing in OMIS fit well into the concepts of the Grid Monitoring Architecture described in [18]. For example, the mechanism of event services in fact enables the subscribe operation from consumer to producer which means a request for a continuous stream of data as opposed to a single request / reply.

4 Grid-enabled OMIS-compliant Monitor

This Section introduces the OCM-G — a Grid application monitoring system. We start from outlining the requirements imposed on the monitoring system in a Grid environment. Then we present a design of the OCM-G architecture. We conclude with considerations related to the integration of the OCM-G with Grid services.

[1] More exactly: *attached* processes; in OMIS, a tool creates its own context of the target system by explicitly attaching to all objects it wants to deal with.

4.1 Grid-specific requirements

The problems we have to deal with when monitoring applications on the Grid are related to reliability, scalability and efficiency of the monitoring system.

Reliability In such a widely distributed system as the Grid, the issue of reliability is twofold: it concerns the problem of keeping information up-to-date and ensuring the mechanism of recovery from crashes, when parts of the monitoring infrastructure are lost, e.g., due to a hardware problem.

Scalabitity The Grid monitoring system not only will potentially encompass a huge number of systems but it will have to be able to handle a great amount of entities (applications, users, and tools – see section 4.3). Therefore, its design should be properly distributed and decentralized.

Efficiency The problem of efficiency of monitoring is related to the collection and distribution of the monitoring data. The data mostly comes from the detection of events, which may occur excessively, with an unpredictable frequency. To avoid frequent context switches we must report the events in an efficient manner, i.e., provide local buffering instead of reporting each event immediately. On the other hand, in order to not impose high requirements on the local buffer's size, we should provide preprocessing of the monitoring data, i.e. instead of buffering a raw trace of events, store a filtered and/or summarized information. For example, if a measurement concerns a count of certain events, one integer is enough to store this information.

Having the above considerations in mind, we have distinguished three types of components out of which the monitoring system will be composed. They are the *Local Monitor* (LM), the *Service Manager* (SM), and the *Application Monitor* (AM). These components are described in the following section.

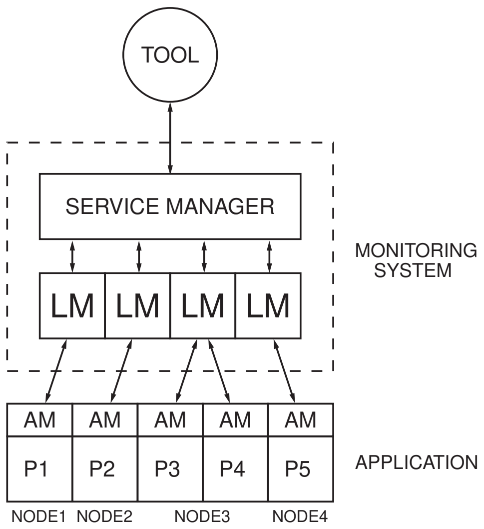

Fig. 1. Architecture of the monitoring environment

4.2 Components of the OCM-G

Parts of the OCM-G are stand-alone components, while other parts reside directly in the application's address space. Fig. 1 shows the monitoring environment. A service manager is the part of the OCM-G to which tools submit requests and from which they receive replies. Service Managers are related to sites in the target system, as shown in Fig. 2. Local monitors reside on each node of the target system. A local monitor receives OMIS requests from a service manager, executes them and passes the reply back to the service manager. An application monitor is embedded in each process of the monitored application. Its role is to execute performance-critical requests directly in the application context and to buffer performance data before passing it to higher layers.

OCM-G shall interoperate with the Grid information system and use it to manage the startup of the monitoring infrastructure (section 4.3). It might also be enabled to put the monitoring data into the information system if a client demands it. This can be useful for tools that perform statistical observations and need historic data (e.g. 'which files have been most frequently accessed by an application during its many executions?').

Another issue is how the components of the OCM-G should cooperate to efficiently monitor a particular application. The process of assembling the individual components into one monitoring system is described in section 4.3. Here we focus on the logical structure of the OCM-G. As we argue in section 4.3, there should be a single instance of the OCM-G on the Grid which will handle multiple applications. Usually, only part of the monitoring system is involved in monitoring a particular application. This part (i.e. the set of LMs and appropriate SMs) constitutes a *virtual monitoring system* for this application. The components of this virtual system are distinguished by the information about the application they possess. The problem is how the multiple SMs should interoperate to enable the efficient distribution and collection of monitoring requests and data. We propose to designate one of the SMs involved in the application and make it the *Main Service Manager* (MainSM) for the application. The MainSM will hold the collective information about the whole application. This concept is illustrated in Fig. 2. This centralization is useful as it simplifies many algorithms and facilitates keeping the localization information up-to-date. We also hope that it will not put the scalability and efficiency of the system in danger. First of all, the centralization will concern only a single application. Furthermore, the messages will not necessarily be routed through the MainSM. The localization information can also be copied to other SMs and enable the direct communication between them. The special role of the MainSM can be limited to some particular cases, e.g., when a process is migrated, this is first reported to the MainSM which in turn propagates this information to the other SMs.

4.3 Integration with the Grid

The Grid environment is based on many sophisticated technologies that are crucial for achieving the Grid functionality. There are many consequences for

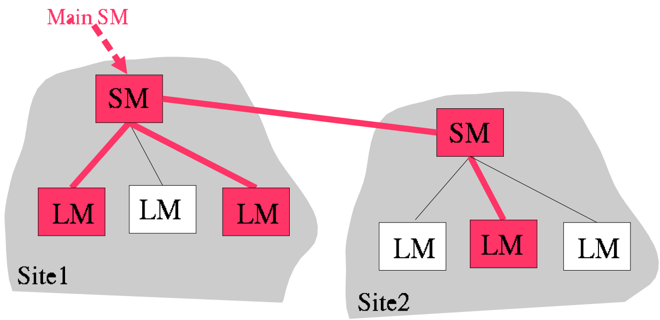

Fig. 2. Virtual monitoring system based on persistent architecture

the monitoring system due to this fact. In this section, we present some issues concerning cooperation between Grid components and the monitoring system.

OCM-G as a Grid service In the previous OCM versions, each monitored application had its own instance of the monitoring system — the OCM was started just before the application. This scenario seems not suitable for the Grid as a proper start-up of the monitoring system and the application may be problematic. Rather, we should have the monitoring system as a permanently running, global service that will manage multiple applications, users, and tools. With this approach, we can develop a portable mechanism of enrolment of the application at the monitoring system (see below).

Security Some security issues derive from the assumption of having a global monitoring system, since the single monitoring system will handle many users executing different applications. We need an authentication mechanism to ensure that a user is allowed to monitor only his own applications. Users will be identified by means of Grid mechanisms based on the Globus security infrastructure.

OCM-G component discovery Another consequence of the permanent-service approach is that the components of the monitoring system are started individually, together with the Grid middleware. We need a discovery mechanism to assemble the components into a monitoring system. This is valid both at the initialization stage when the communication between OCM-G components is set up as well as when creating a virtual monitoring system for applications. In the latter case, we must identify those service managers under which the particular application is running and designate the Main SM (section 4.2). The discovery process should be supported by an external mechanism which will be based on the Grid information services.

In the current information systems (such as Globus MDS [12] or R-GMA [7]), the support for global information search is still poor. Moreover, the future of the Grid information systems is still not clear. Thus, we work on a flexible localization mechanism that minimizes the dependence on the information system. We may also use the Grid job scheduler to obtain additional information.

Enrolment of an application at the OCM-G The standard scenario of monitoring an application is that the monitoring infrastructure (be it an autonomous monitoring system or a monitoring module integrated with a debugger) handles the application startup since it is the easiest way to take control over the application. The startup procedure on the Grid has many aspects, among others, resource allocation or copying of input data, and involves many kinds of job schedulers. Therefore, we should aim at not making the process of application startup dependent on the fact of its subsequent monitoring. In other words, the application should be started in a similar way whether it is monitored or not.

The way to achieve that is to make the application self-register in the monitoring system in a bottom-up fashion. With this approach, each application process will contact the nearest LM to register itself in it. The LMs will then pass this information up to the appropriate SMs. The final step is to initialize a connection between all the SMs involved in the application. To do this, we need an external localization mechanism via which the SMs will find themselves. Next, the MainSM for the application will be chosen. In this way, for each application the virtual monitoring system will be created. The only additional requirements posed on the application to be monitored is that it should be linked with additional libraries containing parts of the monitoring system, which are required, among others, for the initial registration in LMs. Some information can also be passed to the application via command line parameters. The application elements will be identified by tokens, which are globally unique symbols, and will be known a priori (before the application start-up) or — if possible — based on unique grid identifiers.

5 Conclusions

We have presented the OCM-G — the Grid-enabled OMIS-compliant Monitoring System aimed for monitoring applications on the Grid. The autonomous nature of the OCM-G and the standardized interface for communication with tools should ease the development of new tools and enable porting, at least in part, of existing tools to the Grid.

The Grid requirements impose a new quality on the monitoring system, which should be properly distributed and decentralized and support efficient data collection and distribution. One of the most important issues is the proper integration of the monitoring system with Grid services.

One of the issues that is still to be addressed is the development of reliability mechanisms to ensure the integrity and proper functioning of the monitoring system in emergency situations, when parts of the Grid hardware or the monitoring system fail. However, this problem is strongly related to the problem of reliable functioning of applications when parts of the Grid fail, which is also a current topic of the Grid research.

References

1. Balaton, Z., Kacsuk, P., Podhorszki, N., and Vajda, F.: Comparison of Representative Grid Monitoring Tools. http://hepunx.rl.ac.uk/edg/wp3/meetings/budapest-jan01/NP%20Grid%20Tools.pdf

2. 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
3. Bubak, M., Funika, W., and Wismüller, R.: The CrossGrid Performance Analysis Tool for Interactive Grid Applications. Proc. EuroPVM/MPI 2002, Linz, Sept. 2002
4. Bubak, M., Funika, W., Żbik, D., van Albada, G.D., Iskra, K., Sloot, P.M.A., Wismüller, R., and Sowa-Pieklo, K.: Performance Measurement, Debugging and Load Balancing for Metacomputing. Proc. of ISThum 2000 – Research and Development for the Information Society, pp. 409-418, Poznan, 2000
5. Bubak, M., Funika, W., Baliś, B., and Wismüller, R.: A Concept For Grid Application Monitoring. In: Proc. PPAM 2001, Nałęczów, Poland, September 2001, pp. 307-314, LNCS 2328, Springer 2002
6. CrossGrid Project: http://www.eu-crossgrid.org
7. DataGrid Project: http://www.eu-datagrid.org
8. Foster, I., Kesselman, C. (eds.): The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999
9. Foster, I., Kesselman, C., Tuecke, S. The Anatomy of the Grid. International Journal of High Performance Computing Applications **15** (3) 200-222 (2001); http://www.globus.org/research/papers/anatomy.pdf
10. Foster, I., Kesselman, C., Nick, J.M., and Tuecke, S.: The Physiology of the Grid. January 2002, http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf
11. Gerndt, M., Esser, H.-G.: Specification Techniques for Automatic Performance Analysis Tools. Proc. 8th Int. Workshop on Compilers for Parallel Computers CPC 2000, Ecole Normale Superieure Lyon, January 2000, Aussois, pp. 1-11
12. Globus Project: http://www.globus.org
13. Gunter, D. et al.: NetLogger: A Toolkit for Distributed System Performance Analysis. Proc. IEEE Mascots 2000 Conference, Lawrence Berkeley Natl. Lab., Report LBNL-46269
14. Heartbeat Monitor Specification. http://www.globus.org/hbm/heartbeat_spec.html
15. 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
16. 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
17. Shaffer, E. et al.: Virtue: Immersive Performance Visualization of Parallel and Distributed Applications. IEEE Computer, 44-51 (Dec. 1999)
18. 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
19. 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)
20. Waheed, A., et al.: An Infrastructure for Monitoring and Management in Computational Grids. Proc. 5th Workshop on Languages, Compilers and Run-time Systems for Scalable Computers, (March 2000)
21. Wolski, R., Spring, N., and Hayes, J. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. Future Generation Computer Systems **15** 757-768 (1999)