Fine-Grained Instrumentation and Monitoring of Legacy Applications in a Service-Oriented Environment*

Bartosz Baliś[1], Marian Bubak[1,2], and Krzysztof Guzy[1]

[1] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Kraków, Poland
[2] Academic Computer Centre – CYFRONET, Nawojki 11, 30-950 Kraków, Poland
{bubak, balis}@uci.agh.edu.pl
Tel.: (+48 12) 617 39 64; Fax: (+48 12) 633 80 54

Abstract. Legacy applications are still heavily used in Grid systems. In modern service-oriented Grid environments such applications are adapted to work as services. Monitoring of applications is often necessary for various purposes, such as debugging, performance analysis, fault-tolerance, etc. In this paper we present a framework for monitoring of legacy applications wrapped as services. We adapt the OCM-G system for monitoring of distributed legacy applications (such as MPI) to work as part of a broader service-oriented monitoring infrastructure GEMINI. We address monitoring and instrumentation at both service-level and legacy-level. Our instrumentation is fine-grained, i.e. we support instrumentation and monitoring at the level of individual code regions.

Keywords: grid, monitoring, instrumentation, legacy applications, service-oriented architecture.

1 Introduction

Grid systems based on modern service-oriented architecture (SOA) still heavily use legacy code. For needs of debugging or performance measurement there is necessity to monitor such legacy applications both at the level of service invocations and legacy code. The basis of monitoring of applications is instrumentation: we insert additional instructions into the application's code to generate events and pass monitoring information to the monitoring system. Instrumentation in a service-oriented environment cannot be statically inserted whenever we need to monitor an application, since we usually do not have the opportunity to change the source code, compile and deploy the application on demand.

We propose a solution for instrumentation which is: (1) dynamically enabled and disabled, (2) fine-grained to enable monitoring at the level of code regions, (3) accessible through a standardized instrumentation service to expose instrumentation functionality to arbitrary tools and services. This feature involves

* This work is supported by EU-IST Project K-WfGrid Knowledge Based Workflow System for Grid Applications, IST-2002-511385, http://www.kwfgrid.net

a standardized high-level representation of the application to let the user easily pick code regions to be instrumented, and a standardized language for expressing instrumentation requests.

For legacy applications wrapped as services, it is important to address monitoring at both service-level and legacy-code level in a coordinated way, e.g. to relate delays at the service-level to respective code regions at the legacy-level. However, those two aspects involve completely different execution environments. If different monitoring and instrumentation tools are used for those two levels, we are likely to fail in providing a unified view of the monitoring data. Therefore, a generic monitoring system is desirable to collect data from various sources.

To our knowledge, though some grid application monitoring approaches do exist, e.g. Mercury [8], none of the current efforts addresses the described problems in a comprehensive way.

We present a framework to instrument and monitor legacy applications wrapped as services. In this effort, we employ several existing systems and specifications. The existing system OCM-G [1] [2] is used for monitoring of MPI applications. We extend the OCM-G to support the concept of Standard Intermediate Representation (SIR) [3] to provide an abstract view of the application as a convenient way for the user to pick individual code regions to be instrumented. We have designed an instrumentation service compliant with a standardized language WIRL (Workflow Instrumentation Request Language) for specifying instrumentation requests. The mentioned functionality is integrated with the GEMINI monitoring infrastructure [4] [5], which provides us with opportunity to build custom sensors in order to collect information about any entity we want to monitor (legacy application in our case), and ensures the transport of the monitoring data over the Grid.

2 Legacy Monitoring Framework

Our framework used to monitor legacy applications is depicted in Fig. 1. The main monitoring system is GEMINI – Generic Monitoring Infrastructure for Grid resources and applications. GEMINI accepts requests for monitoring data and instrumentation, and also transports those requests and monitoring data over the network. GEMINI also deals with instrumentation at the service-level of the application. For this we used application sensors developed with GEMINI. However, in this paper we will focus on the legacy-level aspect. More information about GEMINI can be found in section 4.

The actual instrumentation and monitoring of the legacy application is done by the OCM-G monitoring system [1] [2]. We adapt the OCM-G to work as a GEMINI sensor. The OCM-G which supports multi-site parallel applications is based on the OMIS specification. The OCM-G custom sensor is an adaptation layer between GEMINI and the OCM-G. From the viewpoint of the OCM-G, the sensor is a tool connected to the OCM-G which sends monitoring requests expressed in OMIS. For GEMINI, however, this component works as a sensor compliant to the Generic Sensor Infrastructure. In OCM-G, the requests for

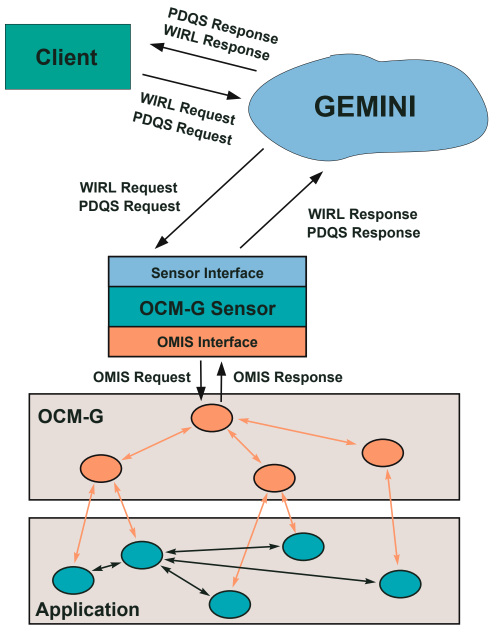

Fig. 1. Legacy Monitoring Framework

data can be expressed in a powerful imperative manner using OMIS[7]. GEMINI, on the other hand, employs an less complex, declarative monitoring request language PDQS combined with the instrumentation request language WIRL. GEMINI also uses a different data representation than the OCM-G. Thus, the main task of the high-level OCM-G sensor is to process queries from monitoring service of GEMINI (expressed in WIRL and PDQS) and convert them to appropriate OMIS monitoring requests supported by OCM-G. And also to convert OMIS responses containing monitoring data to the data representation compliant with GEMINI.

Additionally, we have also adopted the OCM-G to fully support fine-grained instrumentation of legacy applications (see section 3).

Clients of GEMINI are provided with service interface which enable them to control the instrumentation and monitoring and to obtain the monitoring data. Client thus can request SIR to have an abstract view of application and then select individual code regions to be instrumented.

3 Fine-Grained Instrumentation

Our approach to instrumentation is to combine source code instrumentation and binary wrapping with the dynamic control of the measurement process at runtime. The instrumentation is inserted statically via patching of source code or binary libraries, while activation and deactivation of the instrumentation is done at runtime. In our opinion, this approach could work for services: while developers of applications might be required to instrument their code, the cost of inactive instrumentation will be insignificant, and it will be possible to activate or deactivate required parts of instrumentation at runtime. Future implementations of instrumentation may be based on a fully dynamic approach. We provide a tool to automatically insert probe functions at defined places into source files and to generate SIR descriptions of the code. We include SIR into the application executable.

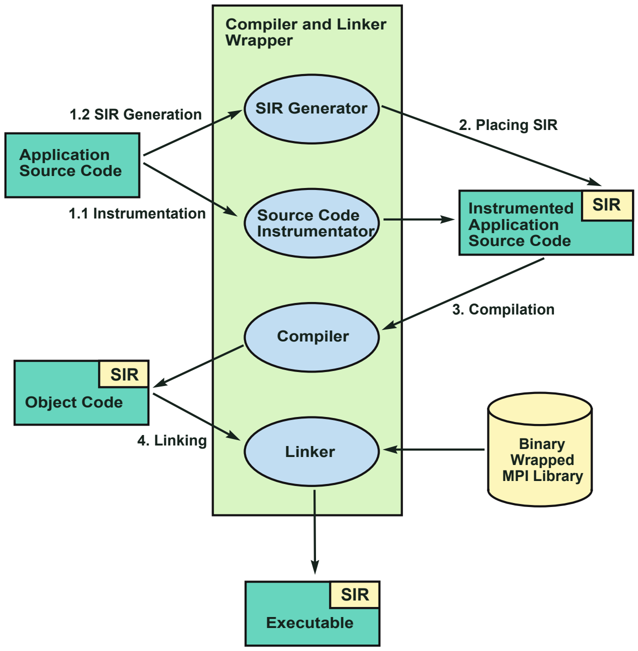

Fig. 2. Instrumentation process

To specify instrumentation requests, GEMINI exposes a service interface compliant with WIRL language (Workflow Instrumentation Request Language). WIRL is a XML-based language in which the user specifies which code regions are to be instrumented and what metrics to compute for those code regions (e.g. *wall-clock time for code region A*). The WIRL requests are translated into OCM-G's OMIS requests.

Our goal is to support monitoring at the level of individual code regions. Currently, the OCM-G does not allows this, we can only monitor *all* calls to a specified function. However, the OCM-G offers the mechanism of probes which can be inserted at arbitrary places into the source code of the application to generate custom events. We have extened the OCM-G with the capability to extract the SIR description from the application. With this and the usage of probes, the OCM-G can fully support instrumentation at the level of individual code regions. The detailed process of instrumentation and SIR generation is presented in Fig. 2.

1. First we take the application's source file and run a tool to instrument all code regions in it (currently function invocations) – step 1.1 Instrumentation. The tool, based on the parsing of the code, also generates the SIR description of the code – (step 1.2 SIR Generation).
2. Then SIR is inserted (as a static string variable) into the instrumented source file (step 2. Placing SIR).
3. Instrumented code is next compiled by the OCM-G compiler wrapper (step 3. Compilation).
4. Received object code is linked with an MPI library, probably pre-instrumented by the OCM-G (step 4. Linking).

After we have done those steps, we have the instrumented application executable which includes SIR that can be obtain by the monitoring system. In SIR, code regions are described, among others with unique names that can be transformed into probe names of the OCM-G. In this way, user "clicks" that indicate code regions described in SIR can be transformed into GEMINI and finally OCM-G monitoring requests.

4 GEMINI – Generic Monitoring Infrastructure

Our main monitoring infrastructure is GEMINI [4] [5]. GEMINI is a generic infrastructure designed to collect monitoring data from arbitrary sources (applications, resources, services, etc.) and transport this data over a distributed network.

GEMINI is generic not only in terms of a variety of supported monitoring data types, but also in that if offers standardized interfaces to query and subscribe for the data and a standardized monitoring data representation. GEMINI also provides a Generic Sensor Infrastructure which enables to easily develop and deploy sensors which produce monitoring data to GEMINI.

Fig. 3 presents the peer-to-peer architecture of the GEMINI Framework. This is a planned architecture; the current prototype provides a full monitoring functionality though only multiple separate Monitors can be deployed. GOM is an external registry that is used to publish GEMINI services and monitoring data provided. D-Monitors and Monitors form a super-peer architecture to transport monitoring data and data requests. D-Monitors expose interfaces to query and subscribe for monitoring data in PDQS and also to issue instrumentation requests in WIRL. Sensors are data producers which are connected to Monitors and D-Monitors.

To specify instrumentation requests in GEMINI, WIRL (Workflow Instrumentation Request Language) language is used. In WIRL one can request to attach application to be ready for instrumentation, enable/disable or finalize instrumentation and get the SIR of an application. The second interface exposed by GEMINI is PDQS (Performance Data Query and Subscription). It is employed to specify query and subscription requests for monitoring data.

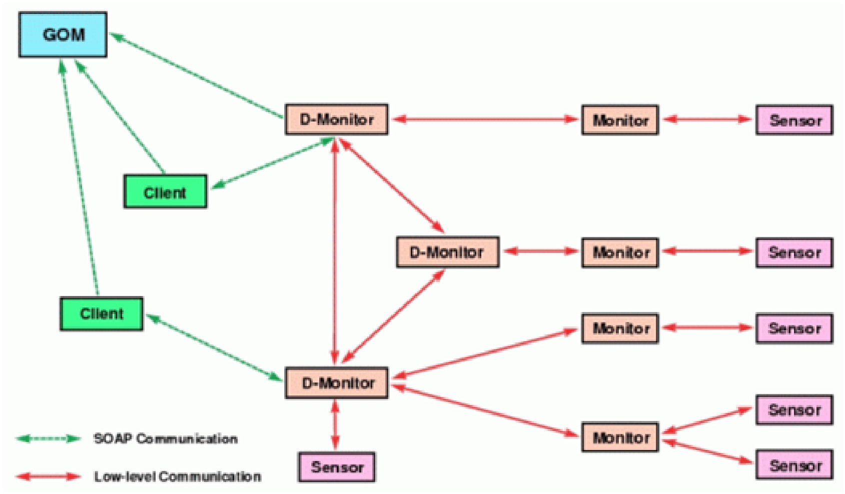

Fig. 3. Architecture of the GEMINI Monitoring Infrastructure

5 Summary

We have presented a framework for monitoring legacy applications wrapped as services and deployed in a grid. The framework is composed of a generic monitoring infrastructure GEMINI to transport the monitoring data and also data and instrumentation requests and a legacy system for monitoring distributed applications the OCM-G. The OCM-G was adapted to be compliant with GEMINI. A completely novel aspect of instrumentation was the support to instrument the application at the level of code regions. Thanks to GEMINI, unified view on monitoring data from the service-level and from the legacy-level of application is possible. This paper was focusing on the legacy level. In the future we would like to evaluate the presented solution on some real-world application scenarios.

References

1. B. Baliś, M. Bubak, M. Radecki, T. Szepieniec, and R. Wismüller. Application Monitoring in CrossGrid and Other Grid Projects. In Dikaiakos M., editor, Grid Computing. Proc. Second European Across Grids Conference, pages 212-219, Nicosia, Cyprus, January 2004. Springer.
2. The OCM-G homepage: http://www.icsr.agh.edu.pl/ocmg
3. C. Seragiotto Jr., H.-L. Truong, B. Mohr, T. Fahringer, M. Gerndt, T. Li. Standardized Intermediate Representation for Fortran, Java, C and C++ Programs. APART Technical Report Workpackage 1, http://www.fz-juelich.de/apart
4. B. Balis, M. Bubak, J. Dziwisz, H.-L. Truong, and T. Fahringer. Integrated Monitoring Framework for Grid Infrastructure and Applications. In P. Cunningham and M. Cunningham, editors, Innovation and the Knowledge Economy. Issues, Applications, Case Studies, pages 269-276, Ljubljana, Slovenia, October 2005. IOS Press.
5. The GEMINI homepage: http://gemini.icsr.agh.edu.pl

6. The K-Wf Grid Project homepage, http://www.kwfgrid.net
7. OMIS – On-line Monitoring Interface Specification. Version 2.0. Lehrstuhl fur Rechnertechnik und Rechnerorganisation Institut fur Informatik (LRR-TUM), Technische Universitat Munchen. http://wwwbode.informatik.tu-muenchen.de/ omis
8. N. Podhorszki, Z. Balaton, G. Gombas. Monitoring Message-Passing Parallel Applications in the Grid with GRM and Mercury Monitor. In: In: Dikaiakos M., editor, Grid Computing. Proc. Second European Across Grids Conference, Nicosia, Cyprus, January 2004, Springer.