Interoperability Architecture for Grid Networks Monitoring Systems

Authors:
Kazimierz Bałos[1], Leszek Bizoń[1], Michał Rozenau[1], Krzysztof Zieliński[1]
[1)] Institute of Computer Science, Academy of Science and Technology in Kraków

ABSTRACT

The aim of this study is to solve the problem of constructing interface of monitoring system which will be suitable for distributed and heterogeneous environments, especially for clusters and grid networks, according to OGSA [10] and OGSI [9] specifications. This study looks at development of scalable and easy to maintain system, that can be used to expose monitored parameters, like network traffic and nodes' infrastructure resource availability to outer applications for further processing. Paper covers topic of interoperability architecture for grid monitoring systems as hierarchical architecture consisting of cluster level layer, and interoperability layer for monitoring data interchange using Web Services as universal interface. Approach shown in this article presents current achievements in the area of using Sun's early implementation of Java Management Extensions (JMX™) [5, 6] technology for communication at cluster level and WS and SOAP protocol for communication at the grid network level. This also covers a way of dynamic monitored stations registering using an open implementation of discovery services, which can be used in all environments using Open Source license. Finally, there are presented results of SOAP Gateway implementation, which can be used for comparison purposes with other existing interfaces of monitoring systems.

1 Introduction

Article presents general architecture exposing monitored system described in [12] as a Web Service. WS have been adopted by Global Grid Forum as common interface used by Grid Services [11] and therefore they used to create the interface for monitoring service described in this paper. The purpose of this service is to make available real time monitoring information collected by monitoring infrastructure (in this case by JIMS - JMX Infrastructure Monitoring System) for cluster online analysis tools.

Paper presents layer of interoperability that complies to OGSA specification [10], including technical aspects of accessing all monitored stations in each cluster through one point of communication. It describes WS interface at one side, for exposing monitoring system as a Grid Service [9], and appropriate JMX interfaces and discovery mechanisms for cooperation and resource discovery of monitoring system at the other side.

In Section 2 functional requirements and SOAP Gateway concept is presented. The idea of interoperability architecture and discovery mechanisms are described. Technical aspects of interoperability layer, design and implementation, are discussed in Section 3. In the end there are shown results of performance evaluation of presented system, including its scalability and behavior in multithreaded environment (Section 4). There will be also discussed advantages and disadvantages of proposed solution.

2 Functional requirements and SOAP Gateway concept

SOAP Gateway (SG) concept is based on general approach described in OGSI (Open Grid Services Infrastructure) specification [9], where grid service is a WS defined using WSDL (Web Service Definition Language), conforming to a set of conventions (interfaces and behaviors) that define how clients and services interact. SG concept bases also on architectural approach taken from OGSA (Open Grid Services Architecture). According to OGSA, grid service should define following points of cooperation with outer application [10, 11]:

a. Grid Service Reference (GSR), which is a WSDL document that describes how to communicate with the grid service,
b. Grid Service Handle (GSH) – a globally unique URL uniquely identifying the instance for all time.

As any other grid service, layer of interoperability for infrastructure monitoring system should support transient service instances, created and destroyed dynamically, and give unified way to access all monitored resources. WS allow to hide complexity of managing monitored stations and exposes one, consistent with other grid services, interface. Because clusters in grids consist of many monitored computing elements, interoperability layer should also perform a role of router, forwarding requests from one outer point of communication to specified node. To achieve this goal it should store addresses of all available monitored stations. In big installations, where there are tens of nodes, administrative way of assigning RMI address of each monitored station in SG would be ineffective. To solve the problem of registering new stations appearing in cluster, as well as deleting from registry inactive ones, there is used special mechanism of active stations discovery. Proposed interoperability layer consists of one SOAP Gateway per cluster. In some cases SG-s can be doubled for fail-over facility.

SG resides in environment where there are used different protocols, i.e. SOAP at the client side, and RMI at the side of monitored stations. Because of this, it should perform a role of translator, connecting itself to monitored nodes through RMI connectors and with client applications through WS. To sum up, there can be specified following requirements for such interoperable gateway:

a. automatic installation to facilitate management of numerous nodes in clusters,
b. automatic configuration with dynamic discovery mechanisms for finding monitored stations that are currently available and heart beat mechanism for removing stations that do not operate properly and are not responding for a certain period of time,
c. hiding self adaptation mechanism (dynamic discovery and heart beat) from the user,
d. exposing one point of communication through one - due to firewalling - well defined application address and port, with WSDL describing its functionality,
e. forwarding requests from WS clients to specified monitored stations.

Pictures below compare monitored data access by one client application through RMI and through SOAP Gateway facility.

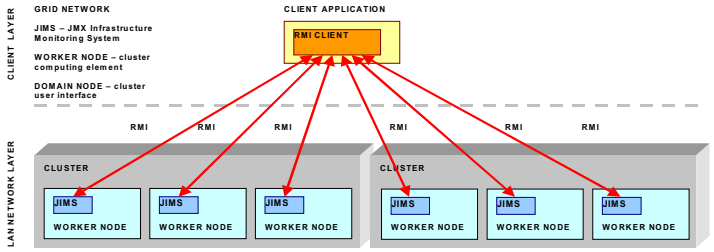

Fig 1: Direct access to each JIMS instance through RMI

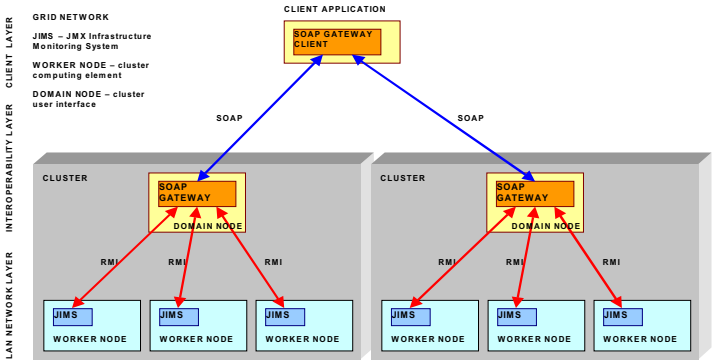

Fig 2: SOAP Gateway concept and its place in interoperability layer

Presented system exposes parameters of monitored stations through WS, performs a role of gateway, manages appearing active stations, maintaining a table of all active monitored stations in the cluster. This dynamic registry configuration mechanism is based on ActiveDiscovery and HeartBeat mechanisms from JDMK – commercial package from SUN for remote MBean servers management. Developed system has all mechanisms found in JDMK, but is based on open implementation of JMX from Sun Microsystems (JMX RI and JMX RI Remote API Early Access 2) [5,6].

3 Design and implementation

SG is a web service running in context of WS container. As WS container it was chosen WASP (Web Applications and Services Platform) from Systinet company, which is designing WS containers for applications written in Java and C++ [8]. WASP server scales very well in multithreaded environment, where many concurrent clients accesses the same web service [1]. Besides, WASP was chosen because it's free for stations having one processor. It is not an issue, because WASP server doesn't have to be deployed exactly in cluster – it can be deployed in any station which has free access, it means not firewalled for RMI communication, to all monitored stations.

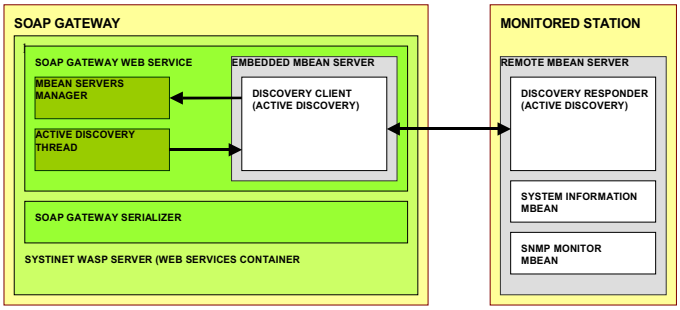

Fig 3: SOAP Gateway design, Web Services container and Active Discovery

Among the others, SG consists of following modules:

- **sg -** WS module with MBean Server Manager, which holds registry of JMX addresses of all monitored stations,
- **discovery –** active discovery library,
- **serializer** – module for serialization of objects that do not have default constructor,
- **jims –** monitored station providing monitored infrastructure information,
- **snmp –** module providing network information through SNMP.

SG autoconfiguration is based on two mechanisms: dynamic discovery and heart beat. First mechanism uses Discovery Monitors at the side of SG and Discovery Responders in monitored stations in order to provide Active Discovery like it is in JDMK [13]. SG periodically sends multicast requests to all monitored stations, and then they respond with theirs RMI addresses of JMX connectors. The second mechanism, heart beat, is a complementary process to discovery mechanism and is used for finding monitored stations that do not respond for some reasons. If a station is not responding certain number of times, it is removed from SG registry and not available anymore. For this purpose each station registered in SG has its own counter of retries which is started after first access failure.

As it can be expected, SG requires very little logic on the client side of application, because whole logic can be hidden behind the interoperability layer. It encapsulates all complexity of discovery and heart beat mechanisms, as well as other – not mentioned here – mechanisms of dynamic applets loading to MBean servers in monitored stations. The advantage of using SG as the point of monitored data access is location transparency of managed nodes. Each change of monitoring station (vanishing or changing JMX RMI address – due to MBean server restart or physical crash) is handled by SOAP Gateway, so the client4 application each time obtains proper list of valid and active servers.

Another problem is the security of the proposed solution. However, it is clear that even without any dedicated authorization mechanism it is more secure than system where client connects directly to each node. WASP architecture is based on interceptors concept, which can be used for authentication and authorization mechanisms by adding helper web services for intercepting HTTP requests and processing them according to security policy set in SG. The same mechanism could be used for SOAP envelopes encryption. In presented form SG runs without any

encryption, but using the same mechanism – interceptors – it can be equipped with encrypting interceptor, which would handle every HTTP request on the server side, encrypt before sending it to client and decrypt after receiving from client. Interceptors cannot be used for network traffic encryption at cluster level, because they operate only at WS side of SG. At the side of RMI connectors there should be used other security mechanism as SSL for RMI connections or TLS and SASL for JMXMP connectors [6].

4 Performance

To evaluate whether SG strongly affects performance of the whole grid monitoring system there was performed substantial number of tests. There is possibility to see how text protocol is scaling with increasing amount of data, number of repetitions and concurrent threads. It is also possible to see how monitoring data traffic affects monitored stations and theirs CPUs and CPUs of domain node, where SOAP Gateway is typically deployed.

It is assumed that SG works permanently, listening to incoming requests and delivering required system and network information data. Because grid infrastructure monitored data are rather static and not changing very fast and the most changeable properties, including CPUs load, are changing about once a minute, so SG usage pattern can follow sequential access rule, where all parameters are accessed periodically. In this case monitored stations parameters are collected in turn from each monitored station in the cluster, so operation of the whole monitoring system imposes periodical and constant but not heavy load to all monitored nodes. In this situation it is essential to test not only raw data access time to each monitored station, but also to examine influence of SG on monitored cluster and monitored stations. Although this is not required, it is also necessary to monitor load of domain node, where WS container with SG facility is deployed. In proposed interoperability architecture of monitored system SG can become a bottleneck, so it should be tested against the load caused by many concurrent connections. To sum up, performance tests should:

- identify and measure possible system overheads,
- provide information about worker nodes CPUs load during performance tests besides simple data access time,
- evaluate domain node CPUs usage during performance tests to show how SOAP Gateway is performing in distributed and multithreaded environment.

Considering above objectives [3] there have been developed test scenario including procedures as follows:

- monitored data access time for fixed amount of data for each node,
- repeated data access time for fixed amount of data, against worker nodes and domain node CPUs load,
- data access time for growing amount of data against all nodes CPUs usage,
- monitored data access time for growing number of executing threads operating on different nodes or on one node, against all nodes CPUs usage, focusing on worker node CPUs usage.

The last but not least aspect of following tests is comparing MBean servers access times through SG vs. RMI. It is clear, that SG increases MBean servers access time,

what was discussed earlier in this paragraph. Important thing is how much SG functionality affects performance of the whole system in order to decide whether it's worth to create the layer of interoperability in existing system monitoring architecture.

First scenario depicted in Fig. 4 shows access time to all monitored stations, while polling them sequentially and requesting full set of the monitored parameters. Charts show that all measured times (average of 100 performed tests) are always below 100 [ms], including overhead caused by SG (bigger values), which seems to be small enough comparing to direct access through RMI.

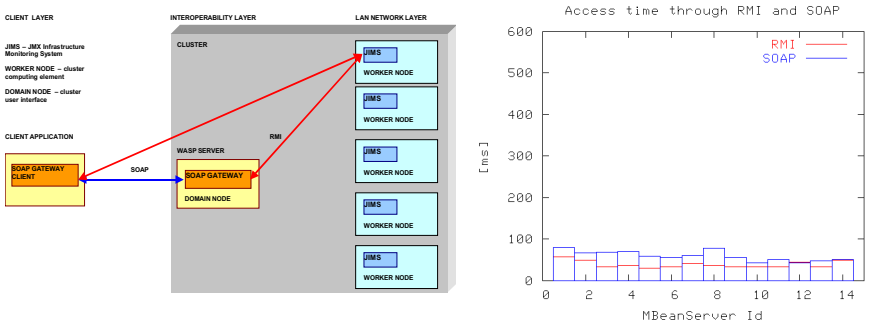

Fig 4: Scenario nr 1: sequential access time to each MBean server in cluster

In scenario number 2 and 3 shown in Fig. 5 and 6 respectively, there were performed tests of MBean servers' access times against the number of requests and number of attributes. There is about 100% overhead of access time through SG in comparison to RMI access.

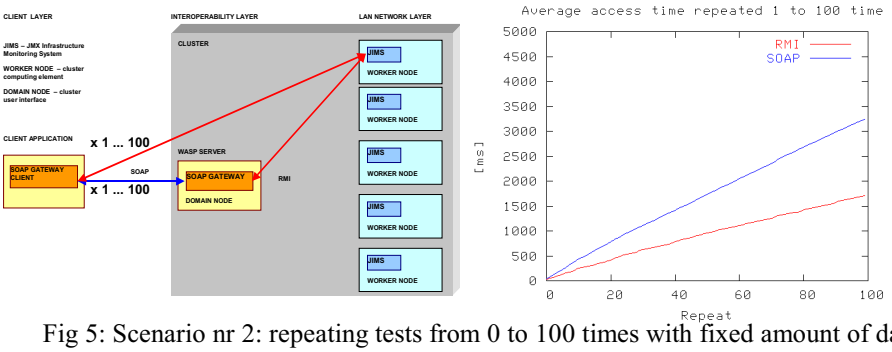

Fig 5: Scenario nr 2: repeating tests from 0 to 100 times with fixed amount of data

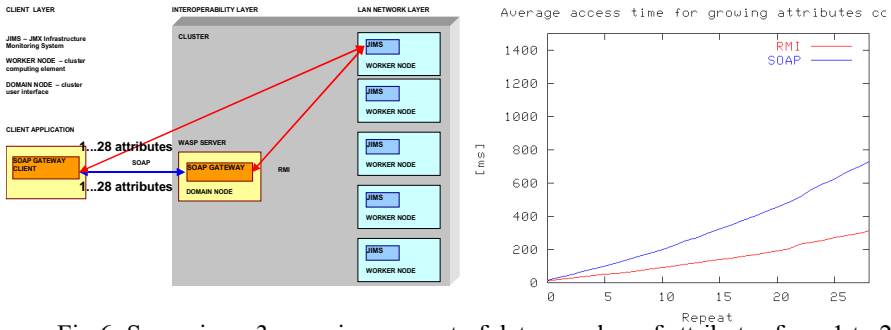

Fig 6: Scenario nr 3: growing amount of data, number of attributes from 1 to 28

More complex tests were performed to check SG operation when many clients concurrently are requesting monitored data from different MBean servers or the same MBean server (the worst case). Experimental results presented in Fig. 7 and 8 illustrate that providing SG neither introduce significant overhead (even during accessing the same monitored station) nor became a bottleneck, what could be expected considering its role as gateway.

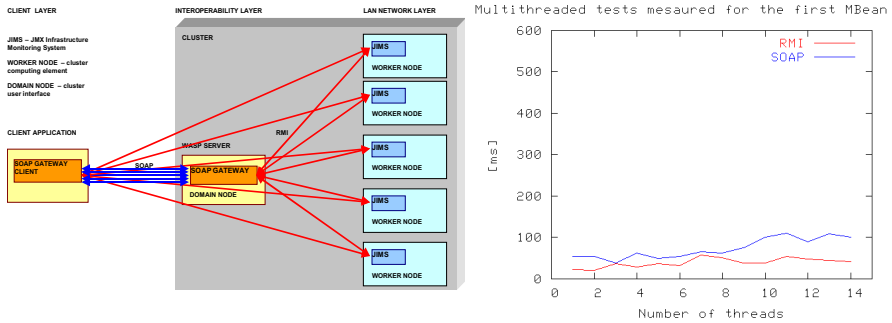

Fig 7: Scenario nr 4: multithreaded tests, access time for the first MBean server

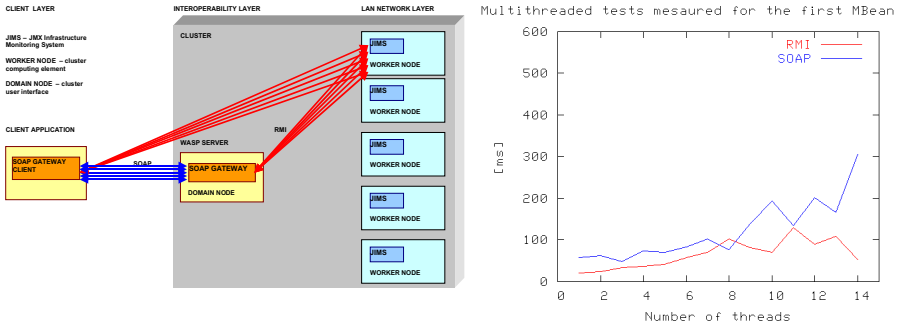

Fig 8: Scenario nr 5: Growing count of threads operating on the same MBean server

During tests there were observed following system behaviour:

1. long time of making HTTP connection to SOAP Gateway (taking about 2 seconds), what was not shown in charts, but will be discussed below,
2. substantial first MBean server access time (always the biggest).

Problem with long time taken to create HTTP connection to SOAP Gateway was solved by proper usage of Service-Locator pattern [7], requiring client application to cache reference to SOAP Gateway for further usage. The second problem was solved by earlier gathering MBeanInfo information causing to “pre-cache” dynamic MBeans in given MBean server. Presented pictures don’t show these issues, because there were applied all rules mentioned above, as using Service Locator pattern in testing software and obtaining MBeanInfo from MBeans before their access operation.

Besides the performance tests, SOAP Gateway passed also a long term test, because it has been running for months without a failure, proving that used Sun’s early access JMX Reference Implementation and its Remote API is reliable enough to

use in current and future implementations of similar projects. In contrast to JIRO technology used in early stage of this project [12], which was theoretically more powerful API for monitored systems implementations and which has been abandoned since some time, JMX is still evolving (its specification is still in progress) and allows to expect that it will be trustworthy successor of JIRO.

5 Conclusion

Presented architecture and experiments prove that idea of exposing functionality of grid infrastructure monitoring system through WS is very satisfactory and efficient as well as provides required level of interoperability. Using WS does not degrade system functionality, supporting in addition its full openness in scope of cooperation with other systems. The presented concept of the monitoring system access can be easily generalized in order to provide access to other system functionality, as its configuration and management. The described architecture and technology may be also easily extended with security functionality.

Acknowledgements

Presented monitoring system (JIMS) is a part of European CrossGrid project. All tests were performed on real cluster situated in ACK Cyfronet AGH in Krakow.

References

1. Bizon L., Rozenau M. 2003, *Zastosowanie Web Services w integracji systemów informatycznych*, M.A. thesis, Kraków
2. Visible Progress Technologies, *Software Performance Testing Considerations*, http://www.visibleprogress.com/software_performance_testing.htm
3. Bailey D. 2001, *Performance Metrics: Out of the Dark Ages*, Berkeley Lab
4. Jung J. 2003, *Grid Network Monitoring*, Multimedia Networking Laboratories, Hanyang University, KRnet
5. Sun Microsystems, *JMX RI v1.2*, http://java.sun.com/products/JavaManagement/
6. Sun Microsystems, *JMX Remote API Specification,* http://developer.java.sun.com/developer/earlyAccess/jmx/
7. Sun Microsystems, *J2EE Core Patterns,* http://java.sun.com/blueprints/corej2eepatterns/Patterns
8. Systinet, *WASP 4.6 White Paper, WASP 4.6 Product Datasheet,* http://www.systinet.com
9. *Open Grid Services Infrastructure (OGSI) Specification, v 1.0*, http://www.ggf.org/ogsi-wg
10. *Open Grid Services Architecture, Globus Tutorial*, Argonne National Laboratory, http://www.globus.org
11. F. Berman, G. Fox, T. Hey, *Grid Computing, Making the Global Infrastructure a Reality*, Wiley 2003
12. S. Zieliński, *Jiro Based Infrastructure Monitoring System,* CrossGrid deliverable, CG-3.3.3-CYF-D3.3-v1.1-JIRO.doc
13. Sun Microsystems, *JDMK*, http://java.sun.com/products/jdmk