Experience integrating a General Information System API in LCG Job Management and Monitoring Services

A. Delgado Peris, P. Méndez Lorenzo, S. Campana, F. Donno, R. Santinelli, A. Sciabà
CERN, Geneva, Switzerland

Abstract

In a Grid environment, both middleware services and applications require information on system resources. Unfortunately, different implementations of Grid Information Services use different data models, and define different access protocols. We have detected the necessity of a high level API that makes the applications independent of the underlying technology of the service, and shields them from possible changes in its interface.

In this article, we present the design of such an API, which can interact with several existing implementations of the Information Service. It is flexible enough to adapt to future changes. We have also implemented a prototype of the API, which can query the Globus MDS and R-GMA, while presenting a single interface to the user.

INTRODUCTION

Access to information on Grid resources is a necessity in order to perform common tasks such as matching job requirements with available resources, accessing files or presenting monitoring information. Both middleware services, like workload and data management, and applications, like monitoring tools, require therefore an interface to the Grid Information Service (IS) which provides that data.

The Problem

Even though a unique schema for the published information is defined, actual implementations use different data models and define different access protocols. Applications interacting with the IS must therefore deal with several APIs, and be aware of the underlying technology in order to use the appropriate syntax to query it or to publish new information.

As an example, the current IS used in the LHC Computing Grid (LCG) [1] is the Globus Monitory and Discovery Service (MDS) [2], which is based on LDAP (Lightweight Directory Access Protocol) and on LDAP specific database backend. MDS is essential for the Grid: the workload management service uses it to get information about available resources that match job requirements. Also monitoring applications, such as GridICE [3], depend on it to periodically retrieve data about the status of the GRID in order to build reports and statistics.

However, for several reasons, a new IS, the Relational Grid Monitoring Architecture (R-GMA) [4], which is based on a relational database and uses the Standard Query Language (SQL), is being deployed. Applications that want to use this new service will have to use not only a new interface but also a different query language. If in the future a new IS, such as the new MDS3, based on the Extensible Markup Language (XML), will replace MDS and R-GMA, every application using the old APIs will have to adapt to the new technology.

Proposed Solution

In order to overcome the described problem, we have designed a new high level C++ API to MDS and R-GMA, which can be extended to future implementations of the IS. A query language and a data model is defined for this general interface. The queries are translated internally and transparently so that they are understood by these Information Services.

To demonstrate the validity of our solution, we have implemented a prototype that can query the Globus MDS and R-GMA, while presenting a single interface to the user. Our proposed interface is flexible enough to be extended in the future if needed.

As will be described later, the prototype has been tested with some common queries used in LCG, for both MDS and R-GMA.

RELATED WORK

There has been much research in data integration, data schema mapping and query translation. Some examples are [5], [7] and [6]; although there are many more efforts in these and related fields, which are not mentioned here.

There is however very little data integration work applied to Grid environments. The only similar effort we know about is that carried out within the ECCE project ([8]), which presents an architecture for Information Services integration. We have, though, gone further by implementing a prototype that actually performs schema and query translation and that has been tested in a real production system (LCG).

A GENERAL INFORMATION SYSTEM API

Overview

The general information system API that we propose should provide applications with query, insert and delete capabilities together with the possibility of dynamically publishing a new schema or modifying an existing one. For the moment the proposed prototype only deals with

queries. Figure 1 shows a schema of the proposed solution including only the query feature. As explained before, it presents a single interface to the applications, defining a query language and a canonical view of the information schema (i.e. of the data that can be queried).

As shown in Figure 1, SQL and the relational model have been chosen for those tasks. The reasons for that is that they are powerful and flexible (offering features, such as the *join* operation, not found in the LDAP architecture) and they are mature and well understood (in comparison, e.g., to XML query languages, such as XQuery [9]).

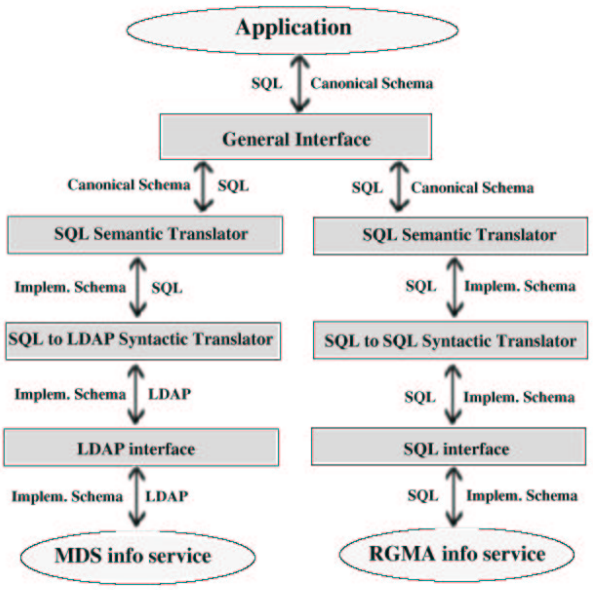

Figure 1: General schema of the high level Information Service API.

Depending on the actual IS that has to be queried (target IS), the user queries are translated, and the appropriate module to interact with this service is dynamically loaded. In order to query MDS, for example, the queries are adapted to its data model and translated to LDAP, and a module that can speak the LDAP protocol (LDAP interface) is loaded to retrieve the information.

Query and Schema Translation

The translation of queries is divided into two phases: *semantic* and *syntactic*.

The *semantic* translation maps the attributes of the general interface data model to those of the target IS (by using a mapping file). This mapping is possible because no matter which IS is used, the data it publishes is the same (in LCG, it is defined by the GLUE schema [10]). Only a correct correspondence between tables (of the relational model), entries (in the directory model) and objects (in the XML model) is needed. After this step, a new query, still expressed in the general interface query language but conforming to the target IS data model has been generated.

The *syntactic* translation completes the process by converting the query to the target IS query language. This may involve splitting the query in several subqueries (in the case that the target language lacks the conciseness of the one used in the general interface).

It is important to notice that, by the use of simple text mapping files for the semantic translation configuration, modifications on the information model in use imply only the adjustment of such files. Moreover, new Information Services can be easily plugged into the general information API (provided that a syntactic translator for their query language is provided, or exists already).

Interfaces to the Target Information Services

At the lowest level of the package there are those classes implementing the specific access protocols. We have defined LDAP and R-GMA interfaces. The system however can support further protocols.

General requirements: By means of a configuration file specified by the user application, it is possible to define the service, the protocol to be used to query the system and further information specific to the protocol in use. This configuration file is of the form: *key* = *value* and it can be parsed via the class *ConfigBuffer* included as well in the package.

LDAP Interface: This interface performs a query to a LDAP based database. This class establishes a connection with a server, performs the query on attributes specified by the application and closes the connection. For not trivial queries the output returned by the server is further processed to provide the correct result. In this case the configuration file includes parameters such as the name of the server to connect to, the access port, the connection timeout, etc.

R-GMA Interface: Each site must have a Monitor Box, a service which collects and stores in a database the information relative of that site. Both together are a collector of information and an information provider to a general R-GMA catalog to which each site must be registered. In our implementation the application performs the request using the SQL query language, while specific configuration parameters are taken from the standard R-GMA installation. For more details on R-GMA please refer to [4].

IMPLEMENTED PROTOTYPE

The main class is the *LcgInfoInterface* class. Applications must first instantiate an object of this class and then call the *initialize* method, which causes the API to read the configuration information and register all the necessary dynamic libraries.

For the described setup, a configuration file is required. The file contains information about the Grid Information Services that can be queried, indicates where the libraries

for the IS interfaces are, and gives the location of the mapping files to be used in the translations. A default configuration file could be present at Grid sites, or could be provided by the application linking the package.

After the initialization, the *connect* method can be invoked for a particular interface (to the target IS), causing the code for a *Querier* object for that interface to be loaded, and a pointer to it to be returned to the caller. By using the dynamic linking of libraries, only the code necessary to query the relevant IS has to be loaded at a given time.

The *Querier* class is a pure virtual one, and defines the methods to be implemented in extended classes. Examples of extending classes are *SQLtoLDAPQuerier* and *SQLtoSQLQuerier*, which can be used to query respectivelly LDAP and SQL interfaces (like MDS and R-GMA), using SQL as the input language in both cases. New *Querier*s, such as a *SQLtoXMLQuerier*, could be added, and even the input query language could be changed, by implementing a *XMLtoLDAPQuerier* class. Of course, this is all transparent to the application because all the extending objects implement the methods defined by *Querier*. The application might even ignore which actual interface is being used: All it knows about is a pointer to a *Querier* object.

The classes performing the translations of queries are organized in a similar way to the *Querier*s, in order to permit the addition of new translations if required. Every one of them extends the pure virtual *QueryTranslator* and implements a particular input and output query language.

Finally, the *Querier* objects use a particular interface to the implemented IS.

LDAP Interface: The main LDAP interface is called *InfoFromLDAP* and it includes a basic method, *query*, which establishes a connection to an LDAP server, performs the query, retrieves the information requested by the application and finally closes the connection. Each operation is performed by specific classes used by *InfoFromLDAP*: a synchronous connection is established through the *ISSynchConnection*. This class is inherited from a pure virtual class called *ISConnection*. This class has been included to allow for any type of connection. Once the connection has been successfully established, the class *ISQuery* queries the LDAP based database following the request defined by the user application. Other classes, such as *ISObject* and *ISForwardIterator*, handle the attributes requested by the application and iterate through the whole buffer returned by the server.

R-GMA Interface: As in the LDAP case, the interface we provide for the R-GMA protocol, called *InfoFromRGMA*, includes a fundamental method called *query* which is responsible for establishing the connection to a R-GMA server, getting the requested information and providing it to the application. Specific classes of the R-GMA package have been used to implement this query. The class *Consumer* executes a SQL query and returns the information to the user. The class *ResultSet* returns the string value result of the query performed by the user application.

EXPERIENCE

We have compared our system with the one used by the LCG-2 Workload Management System (WMS) and verified the full compatibility with our interface: the integration of the WMS software with our proposed prototype C++ APIs is quite "easy" and allows for migration to new IS technologies without further changes in the WMS code.

As we have explained in the overview, the prototype that we present only includes SQL queries to retrieve information already stored in the IS. At this moment only SQL SELECT queries are supported. In this sense, the prototype has been successfully tested using typical WMS and monitoring packages queries as the following example shows:

```
SELECT SoftwareRunTimeEnvironment
UniqueID FROM Glue.ComputingElement
Glue.SubCluster Glue.SubCluster.\
ApplicationSoftware
WHERE
(Glue.ComputingElement.UniqueID=
'lxb0706.cern.ch:2119/jobmanager-pbs-long'
AND
Glue.SubCluster.ClusterUniqueID=
Glue.ComputingElement.ClusterUniqueID)
AND
Glue.SubCluster.ApplicationSoftware.\
SubClusterUniqueID =
Glue.SubCluster.UniqueID

VO-dteam-dteam1
lxb0706.cern.ch:2119/jobmanager-pbs-long

LCG-2
lxb0706.cern.ch:2119/jobmanager-pbs-long
```

The LCG-2 data management services and client tools have been mainly written in java whereas our prototype APIs are implemented in C++. One can interface the Java code with our C++ APIs using JNI. Although such JNI bridge establishes a certain performance penalty, the actual integration is rather straight forward and well understood.

In addition to these tests, we have already deployed in the LCG infrastructure a new tool called *lcg-infosites* which uses successfully the prototype API proposed in this article. This tool allows applications to query the status of the computing and storage resources for each VO.

Some experiments such as *Alice* or *Atlas* have included into their software production environment the *lcg-infosites* tool. In the case of Alice, additional tools depending on this API have been implemented in the official monitoring system (MonALISA [11]) to provide information regarding for instance resources consumption from the Alice VO. Using our proposed API permits a transparent transition

to different information technologies, without disruption in Atlas or Alice operations.

FUTURE WORK

The prototype will be extended to not only get information from the Information Services, but also to add, modify and delete data and change or publish a new schema. SQL being the input query language of the API, the addition of this functionality would require a careful mapping of the attributes affected by the *UPDATE* and *INSERT* statements to the target data model, as one single statement could affect several tables (or entries, or objects).

Since long time, LCG experiments have asked to be able to publish experiment specific information such as software tags and service information following a schema that is VO specific and can change overtime. In this sense, at the end of 2003 the EIS group at CERN has developed a tool called *lcg-ManageVOTAG* that allows experiment software manager to publish tags relative to the version of the software installed at each site. As of today, the tool is flexible enough. However it operates in a fixed mode and with a given static schema.

New middleware services are also continuously added. Every time this happens, a new schema supporting the introduced services needs to be deployed on the entire LCG infrastructure. Clients such as the GridIce monitoring sensors need to go through major upgrades. Having a high level interface capable of handling in a dynamic way such changes can alleviate the problems.

In general terms, the SQL syntax understood by the prototype could be extended to a larger subset than the one currently supported. In addition, a way to bypass the query translation in order to pose a query directly to the target IS (using directly the low level interface) can be easily included.

Furthermore, support for new target Information Services (such as MDS3, or others) could be added, and even different query languages (like XML) could be accepted as input language for the API. As explained before, the architecture of the classes of the prototype already foresees this possibility.

Finally, more *in production* testing and integration with real tools should be performed, in order to test the possibilities of our solution in a real Grid environment.

CONCLUSIONS

We have presented a high level API for Grid Information Services. This interface can successfully interact with several of such services, which may vary in access protocol, query language and data schema, and can be used by Job Management middleware or monitoring applications, among others.

The proposed solution is based on data model (semantic) and query language (syntactic) translation, and in dynamical loading of specific interfaces for the target Information Services. The translations are driven by text mapping files, which can be easily created and modified. This and the pluggable nature makes the system highly flexible and adaptable to information schema variations and new Information Services implementations.

In order to demonstrate the possibilities of our design, a prototype has been implemented. We successfully tested queries to Globus MDS and R-GMA for job management and monitoring purposes.

ACKNOWLEDGMENTS

This work received support from the Istituto Nazionale di Fisica Nucleare, Roma in Italy and Ministerio de Educación y Ciencia, Madrid in Spain.

REFERENCES

[1] LHC Computing Grid Project. http://lcg.web.cern.ch/LCG/

[2] MDS in the Globus Toolkit. http://www.globus.org/mds/

[3] The GridIce Monitoring System. http://gridice.esc.rl.ac.uk/gridice/site/site.php

[4] R-GMA: Relational Grid Monitoring Architecture. http://www.r-gma.org/

[5] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. The TSIMMIS approach to mediation: Data models and languages. Journal of Intelligent Information Systems, 8(2):117–132, 1997.

[6] The Fundamentals of Mapping Objects to Relational Databases Mapping 101, by Scott W. Ambler, 2003.

[7] R. Krishnamurthy, R. Kaushik, and J. Naughton. XML-to-SQL Query Translation Literature: The State of the Art and Open Problems. In Proc. of the 1st Int'l XML Database Symposium (XSym), pages 1–18, Berlin, Germany, September 2003.

[8] K. Schuchardt, B. Didier, G. Black: Ecce – A Problem Solving Environment's Evolution Toward Grid services and a Web Architecture. Concurrency and Computation: Practice and Experience 14(13-15): 1221-1239 (2002)

[9] XQuery 1.0: An XML Query Language. W3C Working Draft. 23 July 2004. http://www.w3.org/TR/xquery/

[10] The GLUE Schema. http://www.cnaf.infn.it/sergio/datatag/glue/

[11] The MonALISA Monitoring System. http://aliens3:8080