Implementing a Web Services Architecture Framework in a Grid Infrastructure

Serena Pastore

1 INAF, National Institute of Astrophysics, Astronomical Observatory of Padova, vicolo Osservatorio 5, 35122, PADOVA, ITALY
pastore@pd.astro.it

Abstract. A major paradigm shift in distributed computing technologies has led during recent years towards two most notably solutions that are represented by grid computing principles and web services technologies and are converging versus a common framework. The grid model focus on large-scale heterogeneous resource sharing in distributed system in a flexible, secure and coordination fashion, while interoperability, integration and ubiquity are essential characteristics of web services architectures. By combining elements from the two paradigms, the paper shows the implementation of a web service framework hosted in a grid infrastructure covering the mapping of web services process actors and grid components. The resulting architectural framework has been tested with a grid application composed by a set of java web services able to access astronomical databases. The document also focuses on the service discovery issue that being a key component of the two underlying architectures requires a suitable solution for both grid and web services environments.

1 Evolution of distributed systems

Distributed system technologies have seen in the last years a great improvements in the evolution of the two most notably solutions represented by grid computing [1] and the web services paradigm [2]. Approaching this computing field, the two models are tending versus a common framework essentially modeled as a service oriented architecture (SOA) [3] whose aim is to provide a service layer hiding the underlying implementations. Grid computing principles focus on dynamically share and coordinate dispersed heterogeneous resources (data, files, storage and applications) that are generalized as services provided by several Virtual Organizations [1] to be efficiently organized and utilized across geographically distributed sites in a transparent and secure access. Web service paradigm instead enables distributed applications and services running on different platforms to easily communicate and interoperate thanks to an infrastructure based on current and emerging standards. The likely convergence of web and grid services architectures will offer a solid platform in environments where distributed computing and managing of a very large amount of massive and complex data spread worldwide is a key factor. In the specific field of astrophysics,
scientists deal with a great amount of distributed data stored in vast databases\(^1\) owned by different institutes or organizations. Providing the astronomical community facilities of searching, accessing and processing such data is nowadays an important need that an international alliance (IVOA\(^2\)) tries to realize by defining a specific architecture (the so-called “Virtual Observatory” vision) and common standards. A workable environment based on such technologies [4] will allow to implement applications compatible with IVOA standards able to a global, uniform and controlled access to data shared by each virtual observatory/organization through a common standard interface.

1.1 Motivation of the implemented framework

Web service paradigm allows to develop and deploy applications designed as services performing basic tasks (searching, retrieving and analyzing data), while a grid infrastructure could facilitate sharing securely across institutional and geographical boundaries. By core services, interfaces and protocols users could access remote resources to enhance processing and storage capability as if they were located within their own machine, while simultaneously preserving local control over who can use them and when. The grid infrastructure used in this project is the Italian production grid for scientific applications\(^3\) provided by the Italian National Institute of Nuclear Physic (INFN). It is built on a version (called INFN-GRID) of the grid middleware developed by CERN for the LHC project (tagged LCG2 [5]) that is in turn a collection of packages derived by several projects and organizations\(^4\). The use of large data in astronomical applications introduces a main issue about the method used to store and manage information through grid functionalities. Dealing with data stored on database system in grid has two not-exclusive solutions that are different way to provide shared services:

- the first is the porting on grid the data source containers that implies to model the entire database management system and its engine as a grid resource,

\(^1\) Sky observations are made in different band of light and at different wavelengths in every portion of the electromagnetic spectrum: the resulting data take many forms depending on the instrument that collected them and the format of the medium where they are stored. Astronomers make a storage distinction between archives and catalogs: an archive contains collections of observational data (such as images or spectra) while a catalog stores observational results (data about physical characteristics of the sky object with a particular emphasis to the date of observation).

\(^2\) International Virtual Observatory Alliance (IVOA), http://www.ivoa.net

\(^3\) The INFN Production grid for scientific applications, https://grid-it.cnaf.infn.it/

\(^4\) The principal components of LCG2 middleware package are provided by the Globus Toolkit (http://www-unix.globus.org/toolkit) developed by the Globus Project (http://www.globus.org), the Condor system (http://www.cs.wisc.edu/condor) and the DataGrid (EDG) project (http://eu-datagrid.web.cern.ch/eu-datagrid/). Tools are integrated and packaged as the Virtual Data Toolkit (http://www.cs.wisc.edu/vdt). Moreover more components come from the packages of the gLite (http://glite.web.cern.ch/glite) toolkit developed in the EGEE project (http://egee-intranet.web.cern.ch).
the second leaves the database management system outside the grid and provide an interface to access specific services that perform the operations needed to interact with.

The choice approach depends on the grid infrastructure and the standards adopted. This paper refers to the adoption of the second solution: grid applications making use of services interacting with data managed by database systems that are located outside the production grid [6], are deployed in an appropriate framework realizing the normal interactions of a web services process. The document reports about the implementation of the architectural implementation embedded in the grid structure by making grid nodes to act as the roles of web service process actors. After a brief description of the main grid components (sections 2), web services architecture and details about the test application (section 3), the document presents the details about the mapping of provider, requestor and registry to their corresponding grid components (section 4). Finally the paper focuses on the critical problem of discovery proposing a solution based on current web services specifications (section 5).

2 Structure of the grid framework

The INFN production grid is a testbed that comprises several sites connected by the European GEANT network joining different Virtual Organizations part of European research institutes and universities. The structure of grid middleware organizes each site in a set of machines performing several functionalities: it normally distinguishes the so-called “core sites” providing main grid capabilities by the so-called “support sites” offering basic grid resources. The entire grid infrastructure is so composed by several support sites and few core sites gathering the different machines (grid nodes) performing grid functionalities as the figure 1 shows. Core sites essentially provide the four main functionalities needed by a grid: security services, resource and data management and information and monitoring tools. Specific software packages that could be installed as rpm packages, offer such key services:

- security is realized by a server machine implementing the Grid Security Infrastructure (GSI) developed by Globus [7] that enables users to be authorized by their digital certificates released by well-know certification authorities,
- resource and data management are provided with software packages derived by the EDG project: in particular the workload management system [8] includes a specific component called Resource Broker (RB),

5 For example the Open Grid Service Architecture-Data Access Integration (OGSA-DAI) (http://www.ogsadai.org.uk) software is a solution to assist with access and integration of data from separate data sources via the grid that is well integrated with the last released of Globus Toolkit, but its integration within Italian grid infrastructure is yet foreseen in the future components of the middleware rather that in the current version.
6 The European GEANT network (http://www.geant.net)
7 Up to now the packages are developed for the Scientific Linux (https://www.scientificlinux.org) operating system.
information and monitoring services are provided by the Globus Monitor and Discovery System (MDS) \cite{9} software and a customized version of the Berkeley Database Information Index (LCG2-BDII\textsuperscript{8}). Both the packages realize an information service based on a Lightweight Directory Access Protocol (LDAP) directory server.

Fig. 1. Main relationships between grid sites and their nodes performing grid functionalities.

Each support site provides computing and storage facilities through respectively Computing Element (CE) and Storage Element (SE) nodes: the CE implements a batch system whose goal is to manage one or more Worker Nodes (WNs) that really elaborate grid jobs, while the SE is an interface to storage. To access the grid users usually interact through the User Interface (UI) node that provides basic commands to work with grid software. Normally each of these roles is played by a single machine even if not necessary. The CE and SE nodes are accessible by the entire grid, since they report about the real site shared resources, while WNs are only accessible by the referring CE. Finally the UI element is any machine connected to Internet. Against the grid structure constraints, the choice of implementing a web service framework in grid follows the technological trends of distributed programming. Grid model, in particular, with the redefinition of its architecture has changed from an implementation based on grid services (in OGSI Implementation) to another based on stateful resources and web services that interacts with (in the WS-Resource Framework\textsuperscript{9}). Even if the used grid software does not still support such model, the proposed solution theoretically maintains the possibility to ensure compatibility and follows the future evolution in terms of grid and web services standards and their implementations.

\textsuperscript{8} The LCG2 Berkeley Database Information Index (LCG2-BDII) (http://www-it.desy.de/physics/projects/grid/testbed/EDG/BDII.html) is a customized version made by CERN for the LHC project of the Berkeley Database (http://www.sleepycat.com).


\textcopyright GESTS-Oct.2005
3 The web services model

Web services should be a natural choice for grid computing since they deal with multiple nodes in various places and provide a standard way to get information from one system to another without having to resort to platform or language specific methods. A web service is defined as a “software system identified by a URI whose public interfaces and bindings are defined and described using XML language10”. It can be accessed via the Internet through its exposed interface whose description declares the operation which can be performed by the service, the types of messages being exchanged during the interactions and the physical location of ports where information should be exchanges. It supports interoperable machine-to-machine interaction over a network by the use of the platform and language independency thanks to XML. Generally the key characteristic is that web services are suitable for loosely coupled distributed systems where a client may have no prior knowledge of a web service until it actually invokes it and could be implemented with several programming languages. Before to describe the proposed framework, the document shows a brief overview about main WS specifications and standards.

3.1 The web services architecture and its main specification

The web services stack (see figure 2) is the conceptual architecture which middleware developers use to contextualize the various WS protocols in relation to one-another. The major standard organizations agree that the architecture stack consists of three main components: wire, description and discovery. To allow meaningful communication between peers that are not possible using the key protocols of the current web infrastructure, it has been defined the SOAP protocol [10] that is meant to promote shared understanding of data in a way that machines can easily and correctly parse them by using XML as the principal data format. SOAP is the transfer protocol for web services that make use of several mechanism to transport its messages (not only HTTP). Layered on top of SOAP, the Web Service Description Language (WSDL) [11] describes the message exchanges that a service is willing to participate in. Through such description, the web service shows an interface described in a machine-processable format that is network-addressable: the other systems interact with it in a manner shown in the WSDL file via standard protocols and data formats with SOAP-messages. The discovery layer on the top, should enable the location of service providing and effective and automated search and selection both for human users and program software agents. Two main specifications that could be integrated has been defined for the aim: the Universal Discovery, Description and Integration (UDDI) [12] and the WS-Inspection language [13] specification, offering two approaches similar in scope, but seen as complementary means rather than competitive approaches to service discovery. UDDI uses a centralized model of one or more repositories containing information on entities and the services they provides, while WSIL

10 This is a general definition provided by the W3C consortium which is one of the organizations that defines standards and specifications about web services areas.
implements a decentralized approach where service description information can be
distributed to any location using a simple XML document format that provides dis-
covery and aggregation of web services descriptions in a simple and extensible fash-
ion. Publisher could advertise their services defining how a service requestor can
discover a WSDL file on a web server since WSIL documents are located using sim-
ple conventions\textsuperscript{11}. The foundation of the web services architecture relies on the three
basic specifications SOAP, WSDL and UDDI that are becoming the de facto standard
for such technology.

3.2 The prototype application
Against grid middleware constraints and the necessity to use compliant packages to
avoid conflicts, the grid application \cite{14} has been designed as set of java web ser-
vices by using all the available toolkits such as those provided by Apache Software
Foundation \cite{15}. Such prototype provides a set of basic services performing the re-
trieval of data sets from a catalog matching fixed query parameters. Information are
store in tables (collecting data and metadata) of a database managed by MySQL data-
base management system. The client application allows the user to select the data
source where the information is presumably stored, to make a detailed query accord-
ing to its type and to retrieve the results in a specific XML-based format (as set by
IVOA specifications) for further analysis or elaboration. The result is a web services
interacting with the database that accepts as input a SQL query or the request of the
service description as standard WSDL interface and retrieves an output a file in XML
format or again as WSDL. The tools used for development include Apache Ant \cite{15}
as programming tools to automate all the steps involved in building and packaging
the application and Apache Axis \cite{17} both as SOAP implementation by a server
which plugs into an application service and development tool to build web services.
The resultant web application, structured in a WAR (Web Archive) package, contains
files and libraries required to its execution with the SOAP engine embedded. More-
over since such web application is deployed on a grid infrastructure is also secured
\footnote{The XML document has a fixed name “inspection.wsil” that containing the web services
corresponding HTTP accessible WSDL document which describes their interface should be
accessible via an HTTP server.} [16] by using a specific middleware package that provides access only to authorized
grid clients\footnote{The web service is secured by means of the EDG java security (http://edg-
wp2.web.cern.ch/edg-wp2/security/edg-java-security.html) a package developed by EDG
project. Security process is performed by the authentication (X.590) of the grid client, and by
the authorization to execute web services through an AXIS handler inserted in the request
flow in front of the SOAP endpoint. All the authorizing files (described in XML format) that
specify the security policies applied to grid users are inserted in the WAR file and so de-
ployed in any web application server.}. Even if implemented in many ways, web services are usually deployed
in application containers (such as Apache Tomcat \cite{15}), in order to be invoked by
any application, agent or other web service. The runtime environment indeed pro-
vides a message processing facility and could receive messages from requestors.

\textcopyright{GESTS-Oct.2005}
3 Mapping of grid components acting as web services actors: implementation details

The web service architecture shows a common usage scenario that defines three entities performing publish, find and bind operations: a service requestor which invokes services by a SOAP messages request over HTTP/HTTPS, a service provider that offers its services described in a standard way (WSDL) and responds to requests and a service broker that registers and publishes the list of services (through WSIL or UDDI standards).

The provider publishes a description of services to a registry that includes a profile of the provider and of service itself together with the URL of its interface definition (i.e. WSDL description). When a developer needs for a new service, he finds it either by constructing a query or browsing a registry: then he interprets the meaning of the interface description (typically through the use of meaningful label) and binds (i.e. includes a call to invoke by using SOAP) the service within the application. All these architectural primary roles that interacts using the three operations should be played by grid nodes: the correspondence in the used middleware between such actors and grid components is shows in figure 2.

![Figure 2. Mapping of grid nodes that act as web services roles (provider, requester and broker) and a comparison with the web services protocol stack and the find-publish-bind model.](image)

The CE, SE, WN and UI nodes as building blocks of the support site are the only grid nodes a grid application developer directly interact with and should play the three roles. However such entities represent key components of the architecture that should not be confined to a site, but integrated with the main grid system otherwise they assume a local feature. The service requestor is essentially mapped by the UI functionality, the WN executing the web application plays the service provider role, while the “local” service broker is interpreted by the CE. The software implementation of the three components of web service model in grid nodes is now described.
3.1 The service provider: the web services offering “query services” through the grid

The runtime environment for web service hosting on the WN is provided by the Apache Tomcat 4 [15] web application server that is responsible for executing the code of the web services and for dispatching messages to them providing also other quality of services such as security. Tomcat 4 engine guarantees also the efficiency of database connections provided by a specific connector13, with the use of the connection pooling that prevents database connection pool leaks. Even if web services are actually deployed on one or more WNs, it is the CE must that must manage the request/response process being the node visible by the whole grid. From a grid client point of view, CE becomes the real service provider while user won’t know anything about the WN used.

3.2 Service requestor: grid client accessing the web services

A grid client or application normally use UI node to be authenticated and authorized to access the grid and its resources as affiliated to a specified Virtual Organization. It should contact both the service provider than the registry: normally it first interacts with a registry querying whose provider shares services and then it contacts directly the provider to discover service and how to address it. Anyway the service requester is mapped by the UI node and the client application/user contacts both the two actors through a command line interface or a web browser. A direct binding between UI and WN in this context is realized as far as concerns grid job management: the process is a job submission to the grid. This means that the application should be specifically structured to follow the rules of the resource manager software and so submitted by using as argument a simple text files written in a specific format14 in order to contain all the information (name of the executable, input/output parameters and other requirements) and specific requirements needed to find the suitable CE where to send the job for WNs dispatching.

3.3 Service broker: implementation by using WS standards WSIL and UDDI

The CE plays the role of a local registry becoming an entry point of services offered by the virtual organization: its implementation by using different specifications is mandatory until the main grid information systems will be able to provide the same functionalities of web services publication and discovery. As first approach to discovery issue, both WSIL and UDDI solution could be implemented and installed. The WSIL solution works for small-scale integration and only its use with a UDDI registry could give a scalable and enhanced solution to discovery issue. The WSIL file containing the request information should be placed at common entry points for a web site managed by a web server (i.e Apache HTTP server [15]) accessible via the HTTP/HTTPS. However this is a top level document, and WSIL documents can be linked to others in order to allow service providers to organize their services listing in

---

13 The MySQL Connector/J driver (http://www.mysql.com) is able to convert JDBC (Java Database Connectivity) calls into the network protocol used by MySQL database.

14 The Job Description language used by this grid middleware make use of Condor ClassAds library (http://www.cs.wisc.edu/condor/classad/).

©GESTS-Oct.2005
a hierarchical manner. The application/user could make use of specific Java class\(^{15}\) library to locate and process WS-Inspection documents. The UDDI approach is a more complex mechanism, but allows to implement a structured way to store information and a data model to specify providers and its services. Among different implementations of this specification\(^{16}\), Apache jUDDI [15] (the choice solution) is an open source Java implementation packaged as web application to be deployed in a engine. It is structured as a set of servlets performing the basic operations and storing UDDI information in a relational database server. The management of the registry for allowing search and publish of information could be provided by existing software tools\(^{17}\). This approach deploys a private UDDI registry on a grid node that unfortunately is not related to the main grid information system. The simple deployment on CE doesn’t guarantee its integration, since the two discovery methods make use of different directory server technologies. Their interaction necessary requires a binding between LDAP and UDDI for example representing UDDI data as objects of an LDAP directory.

### 3.4 Client access to registry or the provider through the web

Since grid nodes are connected to Internet and both registry and provider have a web frontend, a grid user could request directly the service to the provider (knowing its location) making a secure connection (HTTP over SSL/TLS) to the node (CE) with a web browser or interact with a service registry browsing the web page indexing the WSIL documents or the UDDI registry both hosted on the CE (see figure 3).

![Fig. 3. Web access to the main web services discovery mechanisms and an example of direct access to the catalog through JSP technologies.](image)

\(^{15}\)WSIL4J library (http://cvs.apache.org/viewcvs/*checkout*/ws-wsil/java/README.htm) is a class library that provides API to locate, process, read, parse and generate WS-Inspection documents. The UDDI for Java API (UDDI4J http://www.uddi4j.org/) is required to use WSIL document in conjunction with a UDDI registry to generate and parse messages sent to and received from a UDDI server.

\(^{16}\) Another java implementation is the UDDIe project (http://www.wesc.ac.uk/projects/uddie/uddie/) that differs on the recommended database.

\(^{17}\) For example UDDI browser (http://uddibrowser.org) supports the entire UDDI API for manage all entities in the registry.
For example the user could access the catalog through JSP pages specifically written to use the application by a browser. In both cases the request flow is intercepted by the grid security mechanism that verifies user’s authentication examining the X.509 certificate installed in the browser and, if the user is authorized, allows for the connection to the catalog to retrieve query results. But these methods take advantage only of the grid security infrastructure and don’t allow for using other grid functionalities. An access via grid commands could be realized only if the registry is completely integrated in the information service.

5 The Discovery issue: integration of grid and web services solutions

Service broker is a key element of both the architectures. There are three leading viewpoints on how to discovery service should be conceived: as a registry in an authoritative, centrally controlled store of information, as an index with a simple compilation or guide to show there the information exists and as peer-to-peer (P2P) discovery without centralized registries, letting the web services to discover each other dynamically. Both UDDI implementation and the Globus MDS2 are based on the first approach by using different service directory technologies. Particularly the used information system is built on a LDAP directory tree as a combination of several hierarchical components at different level (see figure 4).

According MDS architecture [7], computing and storage resources at each support site report their static and dynamic status via specific LDAP servers (the Grid Resource Information server or GRIses) to another LDAP server (the Grid Information Index Server or GII) whose role is to collect all support site information. In turn site GIIes refer to a top level LCG2-BDII server that is directly invoked by users and services to address primarily the resource selection problem, namely how a user iden-

---

18 The LDAP software implementation is provided by OpenLDAP (http://www.openldap.org)
tifies the host or set of hosts on which to run the application. The BDII acts as a cache storing data about grid status and contains essentially an updated index of LDAP URLs (the site GIISes). The LDAP protocol offers a hierarchical view of information and arranges entries in a directory information tree (DIT) where each resource is published: its information model is based on entries representing resources that are described by attributes compliant with a specific schema\textsuperscript{19}. Such structure allows for search on the DIT by filtering entries according a mechanism that is used by the component of the resource management system (RB) is responsible to find the specific resource. Usually each CE and SE host a GRIS to collect information about hardware resources: they register to a site GRIS also hosted in a CE which in turn refers to the central BDII. In this hierarchy the search could be done at the three level: a client could search directly a specific GRIS (lower level), query the site GIIS (medium level) or make use of the BDII (upper level). The private UDDI node that is also deployed at the lower level could be queried directly (figure 4). However to interrogate the registry at an upper level, its integration into the LDAP directory seems to be a mandatory solution. Such approach requires a mapping of UDDI-LDAP data by the use of a specific schema\textsuperscript{20}: in this way UDDI data could be embedded into a GRIS and so discovered by a client specifying UDDI parameters and attributes submitting the job.

6. Conclusions

The implementation of a web services architectural framework into a grid infrastructure has allowed to study the feasibility of adopting web service paradigm in the development of distributed astronomical applications in order to be deployed and used in a grid environment. The combined used of such technologies in the application test composed by java web services performing interaction with astronomical catalogs has shown the efficacy of the solution. Knowing the location of services directly or through standard web services discovery methods (WSIL or UDDI), a grid client could performs analysis by using the user interface on the catalog and retrieving results that could be elaborated using grid capabilities. But an open problem that is on study remains about the implementation of a “broker” that could provide at a high-level web and grid information services. UDDI and LDAP are different approaches to provide a directory server and both present advantages and limits addressing different scenarios. Their integration could be a downsize approach to the current direction of discovery system, but it is required to take advantage of UDDI features and developments working in a grid environment that is currently in evolution.

References

\textsuperscript{19} Specifically grid resources, modeled as object of the LDAP tree, must conform to the Grid Laboratory for Uniform Environment (GLUE) schema that defines a common conceptual model to be used for grid resources monitoring and discovery.

\textsuperscript{20} An LDAP Schema for UDDI is provided by an IETF internet draft “LDAP schema for UDDI” (http://ietfreport.isoc.org/idref/draft-bergeson-uddi-ldap-schema/).
Implementing a Web Services Architecture

[4] J. C. McDowell, “Dowloading the sky: astronomers and computer scientists are building the world’s best telescope-and it’s all online”, IEEE Spectrum online. 2004

Biography

▲ Name: Serena Pastore
Address: INAF – Astronomical Observatory of Padova, via Osservatorio 5 – 35122 – Padova, ITALY
Education & Work experience: Electronic Engineer. She started working in network and system integration field. She actually works at the Italian National Institute of Astrophysics (INAF) involved in R&D projects of assessing and testing technologies in astronomical fields.
E-mail: pastore@pd.astro.it

©GESTS-Oct.2005