Internet technologies and the grid paradigm: designing a custom environment for web service-based applications

SERENA PASTORE
National Institute of Astrophysics (INAF)
Astronomical Observatory of Padova
Vicolo Osservatorio 5 – 35122 - PADOVA
ITALY

Abstract: - Methodological and computational difficulties introduced by the expanding knowledge base of science domains reflected in the growing volumes of data that must be stored, retrieved and analyzed can be overcome by applying distributed technologies from a grid, web services and internet technologies in general. Applications are developed as services-based paradigms, making use of for example Java platforms that support interoperable data handling, scalability, load balancing and other capabilities. Moreover they should be deployed in an embedded secure environment that may provide all necessary functionalities like description, processing, discovery, composition etc. In the context of specific grid infrastructures and astrophysical applications, the paper starts with issues that have arisen in deploying grid applications, then describes the design and custom solutions proposed in the areas of discovery, composition and security. The introduction of specific technology components implementing standard specifications can enhance grid computing infrastructure providing a method for an integrated solution.

Key-Words: - web services specifications, grid technologies, service discovery, composition, security

1 Introduction

Methodological and computational difficulties introduced by the expanding knowledge base of science domains reflected in growing volumes of distributed data which is routed to different data centers to be stored, retrieved and analyzed, can be overcome by applying technologies from a grid [1], web services and from internet technologies in general [2]. Distributed systems have implemented a remote method invocation by using frameworks [3] that are able to provide a remote object that works like a service comprising a set of functions. The service concept exists in web and grid services as a set of XML-based specifications which allow it to be exposed, discovered and securely accessed. This approach provides scale, scope ubiquity and ease of deployment and utilization. Pursuing performance improvement of computer-intensive, long-running applications, Web Services technologies play a critical role in two areas of high performance computing and distributed parallel computing: communications/deployment (application adaptation) and classifications/discovery services of resources (infrastructure layer). The deployment and execution of services requires an infrastructure that provides all necessary functionalities in terms of description, discovery and management in a secure way. Grid architecture forms a practical basis for building systems that are robust, maintainable and secure. In the specific scientific context of a research project whose objectives were to study the porting of astrophysical applications in a specific grid infrastructure developed according to the specifications introduced by the community in order to provide uniform access and use of data [4], grid and web services technologies have been studied and experienced [5]. Each interaction within the scenario has raised a set of problems that require customization of the grid framework by including components that provide web service-based functionalities. Evaluating development, deployment and the tuning of service-based applications within the grid environment, it is argued that the general infrastructure must support robust, reliable and resilient distributed messaging within distributed systems. A message encapsulates information pertaining to transactions, data interchange and the searching and discovery of resources as well policy information. Developing grid applications with a web services computing paradigm [6] solves interoperability issues, but introduces new problems regarding their use in the discovery, composition and security areas of the environment. Production grid architecture, based on middleware developed in European grid projects (LCG-2 software (http://lcg.web.cern.ch/LCG) and gLite toolkit

(http://glite.web.cern.ch)), is only partially based on grid services and web services specifications that had been initially developed for fast job executions within world-wide nodes. Developing and deploying applications in this context has required an approach that tries to integrate with the current middleware specific modules. The designed solution required the implementation in the grid of web service specifications like UDDI (http://www.uddi.org), BPEL (Business Process Execution Language) and WS-Security that are all developing as OASIS (http://www.oasis-open.org) standards. The paper describes the design of the components introduced in order to provide discovery facilities and to enable automation between entities involved together with security constraints related to the use of such technology. While discovery and security solutions have led to some results, the composition area is still in the study phase.

2 The grid paradigm, web service technologies and possible implementations in a scientific context

Web services are gaining acceptance as the predominant technology used to interconnect disparate applications; grid computing is dealing with interoperability problems in distributed environments. A grid provides infrastructure for parallel distributed computing: the Globus project (http://www.globus.org) and its proposed architecture specifications describe how web services can facilitate the creation, life-cycle management and security requirements. Mainly it addresses the infrastructure and resources associated with the grid and not the applications that run on it. The most common approach to enabling an application for a grid environment is to decompose it into smaller, independent subtasks (job chunks) and submit the job to the grid (schedule and deploy). This paradigm provides a good solution for batch cycle applications but it lacks the capability for more complex applications. Web services could be seen as a self-contained and self-describing web application that can be published, located and invoked across the web with the goal of performing functions. When deployed, other applications or services can discover and invoke them in the concept of reusability and in composing them into components. Moreover their modularity, exposition outside of the particular paradigm of system, a machine-readable format description and the implementation-independency of the interface are features that make them a good choice for distributed programming in an Internet environment. The motivating scenario examines the ability to develop applications that provide astronomical facilities as services. They have to be sharable in a grid as a means of searching for information in several data sources, retrieving results in a specific format and processing them through web-based tools. The astronomical search process comprises, for example, different atomic operations that have a return list of the available data repositories, a detailed description of a specific repository and the different kinds of searches available on those data. For the sake of simplicity, all these operations can be viewed as a single web service. The interactions can be synchronous, like the looking-up procedure and asynchronous, such as the query submission to the system which will compute the query. The operations may be designed so as to be executed serially or in parallel and should be discovered and secured so that only authenticated and authorized users can access them. The solution makes use of the current Java platform, since applications are developed by using the Java programming language. The Java platform, even though it is only one of the possible implementations, provides adequate support for XML data handling, scalability, load balancing and other capabilities concerned with, for example, security, distributed transactions and reliable messaging.

2.1 A WSDL-based service and its container

Each service deployed by the grid is described by a Web Service Description Language (WSDL) (http://www.w3c.org) document (Fig.1) that allows a remote interface to be exposed. The document describes the service as a collection of communications endpoints or ports; it includes abstract and technical information. All the interactions are supported by messages transported within the SOAP protocol and the data being exchanged are specified as part of the message included in the SOAP document. Every type of action allowed at an endpoint is considered an operation. Collections of operations possible on an endpoint are grouped together into port types. Messages, operations and port-types are all abstract definitions. Furthermore a port is defined by being associated with a network address with a reusable binding that is protocol and data format specified for a particular port type. The collections of ports define a service. Web Services are usually deployed in a web application container that is transformed from a

container for presentation and tightly coupled logic to an infrastructure that equally supports asynchronous messages and flow coordination.

Fig. 1. WSDL document that represents the QueryServiceBase service. It provides queries to a database that supports the select operation.

```
<wsdl:definitions targetNamespace="urn:AServices">
 <wsdl:types><schema targetNamespace="http://AServices">
    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
    <complexType name="ASRow" /><complexType name="ASResult" /></schema>
</wsdl:types>
   <wsdl:message name="selectRequest">
      <wsdl:part name="databaseName" type="xsd:string"/>
      <wsdl:part name="selectList" type="impl:ArrayOf_xsd_string"/>
      <wsdl:part name="tables" type="impl:ArrayOf_xsd_string"/>
      <wsdl:part name="predicate" type="xsd:string"/>
      <wsdl:part name="maxRows" type="xsd:int"/>
   </wsdl:message>
  <wsdl:message name="selectResponse">
<wsdl:part name="selectReturn" type="tns1:ASResult"/></wsdl:message>
<wsdl:portType name="QueryServiceBase">
  <wsdl:operation name="select" parameterOrder="databaseName selectList tables predicate maxRows">
   <wsdl:input message="impl:selectRequest" name="selectRequest"/>
    <wsdl:output message="impl:selectResponse" name="selectResponse"/></wsdl:operation>
</wsdl:portType>
<wsdl:binding name="QueryServiceBaseSoapBinding" type="impl:QueryServiceBase">
   <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="select"><wsdlsoap:operation soapAction=""/>
      <wsdl:input name="selectRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:AServices"
use="encoded"/>
      </wsdl:input>
      <wsdl:output name="selectResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="urn:AServices"
use="encoded"/></wsdl:output></wsdl:operation>
</wsdl:binding>
<wsdl:service name="QueryServiceBaseService">
    <wsdl:port binding="impl:QueryServiceBaseSoapBinding" name="QueryServiceBase">
    <wsdlsoap:address
location="https://griditt001.oapd.inaf.it:8443/C5/services/QueryServiceBase"/>
    </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

For Java technologies, the Tomcat (http://tomcat.apache.org) /Axis (http://ws.apache.org/axis) framework by Apache provides a container/engine implementation and allows applications to be exposed as web services through SOAP/HTTP interface. If these services are shared in a grid environment, the framework must be present in a grid node exporting services to the whole environment.

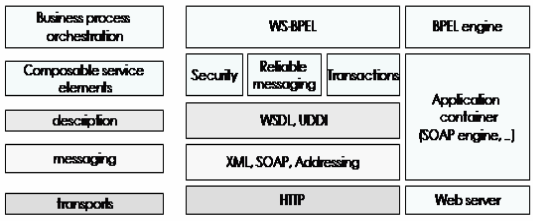
2.2 The logic infrastructure

Production grid [7] is structured as a collection of logical machines implementing grid functionalities. The infrastructure is composed of several sites belonging to different organizations (the Virtual Organizations or VOs [1]). Each site belongs to an institute that shares resources, aims to be part of the grid, and which should implement different features to allow a constant connection to the network. All nodes in a grid environment are centrally managed and monitored to follow the dynamic changes of the infrastructure. The main logic roles in sharing resources are played by the so-called Computing Element (CE) and Storage Element (SE) nodes which essentially provide processing and storing facilities. Moreover such nodes may host an application container and engine. Sharing a service-based application in such a grid node requires that the resource be catalogued as part of the main infrastructure. This approach indicates the description of the resource following a schema. In fact, grid resources need to be indexed, automatically discovered and processed by the main grid components. In a grid system the execution of an application has to assure, in a secure and easy way, the localization of the best component as a means of using the best distributed resources available in the moment it is required by the process. The log of all the processes guarantees that it is simple to know the state of the execution. In our context, the grid procedure means the use of a discovery engine that is LDAP-based, the absence of a process orchestration engine and the application of security strategies that partially address web services issues offering transport basic security capabilities, but not all end-to-end requirements. Since we have found as these tools are not sufficiently adequate to describe, find and share services-based grid applications, we have chosen to integrate in this environment specific web services components.

3 Web services components for the grid infrastructure

To overcome such problems two complementary approaches have been proposed: the development of a new grid middleware component [8] that has led to a new logical element of the infrastructure and an integrated design in the current middleware of modules implementing web services specifications for the target objectives.

Fig.2. A comparison between the web services layers, specifications and proposed implementations

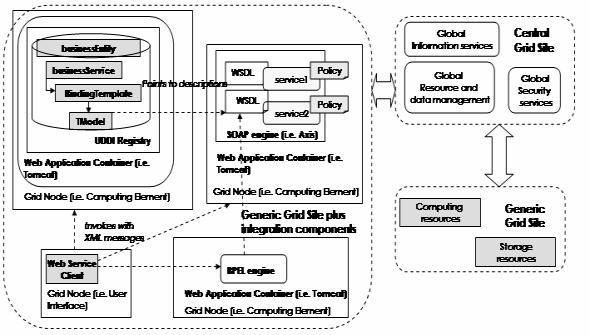


In the second approach, we have specifically analyzed components addressing the different areas with the following solutions:

- a discovery engine based on the UDDI specifications with the design of its enhancement with semantic technologies to introduce automation;
- a web service composition system based on a BPEL engine hosted in a grid machine;
- a security strategy based on the handlers' method to provide a security chain to web services.

While Fig.2 represents the web service architecture layers, specifications and possible implementation, Fig.3 shows the components deployed in specific nodes of the grid infrastructure that are strictly correlated each other.

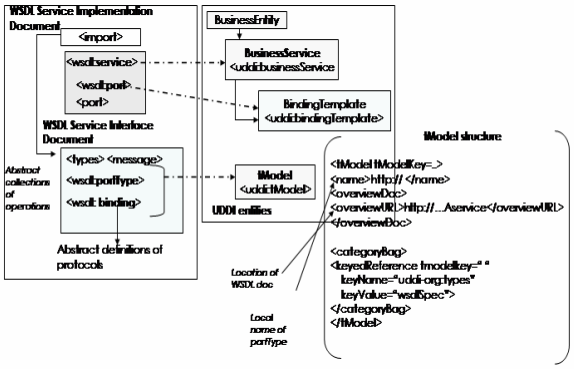
Fig. 3. Integration of web services components in the generic grid infrastructure



3.1 The discovery system

The grid discovery problem has been initially solved by using an LDAP-based approach implementing a hierarchy of index systems like the Monitor and Discovery System (MDS2) of the Globus Toolkit (http://www.globus.org/toolkit). According to this approach all grid resources are described in a specific schema and published in an LDAP directory. LDAP servers are present in different levels of the grid infrastructure: host, site and global level. The root LDAP is constantly updated with information about all sites composing the grid. This method has been revealed to be insufficiently expressive for describing a web services application and the information embedded in a WSDL document. In an ideal world, services would be able to advertise themselves and the discovery system should provide correct information. We have proposed the design and integration of an UDDI registry to better address web services functionalities in the grid. Among the different implementations of UDDI specification, the testing prototype [9] makes use of the Apache jUDDI solution (http://ws.apache.org/juddi) that acts as a repository of WSDL services. UDDI allows for a better description and categorization of services and entities. In such a perspective each web service representing one or more operations (i.e. a selection of the specific query) has a valid WSDL document that describes its interfaces. Even if targeted to a business environment, we have decided to define our VO as an entity that offers several services exporting a WSDL interface. UDDI categorization is mainly based on geographic location or on an industry classification system (i.e. the North American industry classification system or NAICS) of the business organizations, but its data structure allows it to handle other features of a WSDL-based service. tModels, for example, are an abstract concept that may represent standards, specifications and documents. Other structures that describe the organization and the services it offers are businessEntity, businessService and bindingTemplate; they can have relationships with tModel but with a different meaning. While businessService is a logical container of service, the bindingTemplate structure contains the accessPoint of the service as well as references to the tModels. The direct mapping between the WSDL structure and the UDDI data model introduced by OASIS [10] may enable the automatic registration of WSDL definitions in UDDI and UDDI queries based on WSDL artifacts and metadata (Fig. 4). tModels are used to represent technical specifications such as service types, bindings and protocols or to implement category systems that are used to categorize technical specifications and services. When a specification is registered in the UDDI registry as a tModel, it is assigned a unique key (tModelKey). This key is used to reference the tModel. Moreover, tModel contains an overviewURL which provides the address of the specification itself (i.e. a WSDL document) and additional metadata to address the identifier (in the identifierBag construct) or category systems (in the categoryBag construct). Each of these metadata contains a set of keyedReference elements that specifies the tModelKey of the category system and a name/value (keyName and KeyValue) pair that specifies the metadata. For example, we may use these data values (keyname uddi-org:types and keyvalue wsdlSpec to specify a WSDL namespace) as selection criteria when searching UDDI.

Fig.4. Mappings between WSDL structure and UDDI data and an example of tModel structure



Queries for discovery may, for example, once given the namespace or local name of a wsdl:porType or wsdl:binding find the tModel that represents them. Or, it may be given the name of a wsdl:service to find the businessService that represents the service or the tModel representing a portType; or a binding may find all tModels or bindingTemplates that represent that model or binding. The tModel may be classified as being of wsdlSpec type, referring to those services whose description is based on the

WSDL document describing the interface and the overviewURL attribute contains a URL pointing to it. This classification is based on a finite set of values (key/value pair) and the method efficiently describes all applications. UDDI provides such a technical infrastructure for publishing several details of web services on the Internet that could overcome the limit of using the grid LDAP solution for this kind of resource. However it still requires human interaction since it implements a poor search model that is essentially key-based. To introduce automation capabilities, it is necessary to approach semantic web technologies; some researches has been done in this field [11]. Semantic allows a well-defined meaning to information to be given through the introduction of an ontology that is an explicit formal specification of how to represent the information with a language. A semantic annotation is the description of entities using semantic web standards. In the web services area, research aimed at annotating the WSDL with specific languages that reference URI in the ontology or by an external description. Once a web service is semantically annotated, this information can be advertised in some semantically-augmented service registry and that information can be used for service discovery. Furthermore different attempts to bring semantics to UDDI focus on the process of a semantic search based on an internal or external matchmaker engine [11]. The current work is trying to implement such techniques in the UDDI registry.

3.2 Addressing the composition goal

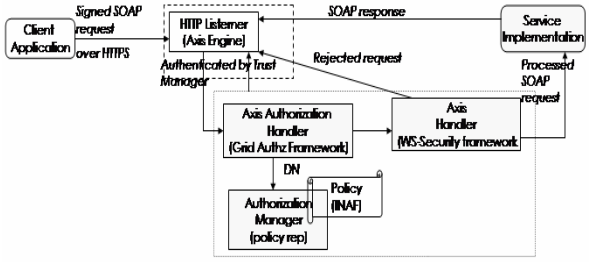
With the need to interconnect services, the key idea is to implement a workflow to connect simple, well-defined functionalities. A workflow, or process, is a set of activities in a graph that has one or more beginnings and one or more ends and gives a general description of everything that can and should occur. It could include as participants human participants as well as other pieces of software. The software program that executes a workflow is the engine that uses the general description given by the workflow to coordinate all tasks. This description, which helps the engine to understand and execute the workflow later on, is made by using a process description language (or grammar). Most recent process description languages are XML-based and we have decided on a tradeoff between cost, quality and requirements, to refer to the Web Services BPEL (WS-BPEL) and on a BPEL engine to prepare the environment for workflow execution. One possible implementation is ActiveBPEL (http://www.activebpel.org), an open source solution for the BPEL engine written in Java and therefore compliant within the analyzed environment. The engine provides a robust runtime environment that is capable of executing process definitions created to the BPEL 1.1 specification. It reads BPEL process definitions but also other inputs such as WSDL files and creates representations of BPEL processes. In this way, all the information with which it needs to communicate (i.e. port, name, operations, service and messages related to the service), are referenced in the process directly by its URL or by copying the WSDL into the project. As of now, this design is in the feasibility phase.

3.3 A security perspective

Security issues play an important role in both web services and grid environments since they address systems placed in a public network over TCP/IP connections. New concerns arise as a result of distributed computing and transacting business over the web. There are five primary areas to consider: authentication, that is the act of assuring an entity is who its says it is by providing some form of proof; authorization, the process of determining what an authenticated user is entitled to do; confidentiality, a means of guaranteeing that only the intended recipient can view the information being exchanged; integrity, a means of ensuring that information arrives at its intended destination unaltered; and no repudiation, the ability to trace or log the fact that a document was sent and received. The Grid Security Infrastructure [12], implemented in every grid middleware, provides open standard Public Key Infrastructure- based mechanisms to address authentication requirements and several authorization mechanisms to secure access for sites, hosts and resources in a VO structure. In the reference infrastructure users, hosts and services are authenticated by X.509 credentials (certificate and key), but users are authorized by the VOMS (VO Membership System) system [13] that is an account repository which helps to defines roles and attributes by embedding them in special X.509 based format (VOMS credential). The solution proposed to secure our grid applications [14] addresses the security of both the container/engine and the application, essentially by using a message handler technique as the ability to intercept the SOAP messages and pass them through a series of processing steps prior to actually delivering them to the service implementation code. This method can be configured and applied to either individual services or to all services deployed on the server. The process (Fig. 5) makes use of an authentication

module to secure the Java container and an authorization framework. This consists of an handler being put in front of each web service to be controlled and a policy engine repository that implements an XML policy to allow or deny access to the application according to VO roles and attributes. Moreover, the flexibility of the handler technique allows for the implementation of a security chain to protect web service and adding, for examples, WS-Security elements.

Fig.5. Security chain of the web services application access in the grid environment.



A study is going to test the possibility of adding WS security specifications (XML encryption and digital signature) by using the Apache WSS4J (http://ws.apache.org/wss4j) implementation of the standard. This software provides a handler with the ability to add a WS-security layer to the web services with minimal impact to the service and client implementation itself.

4 Conclusion

While grid standards and implementation are evolving towards using web services architecture to provide basic functionalities, our work is focused on offering a specific environment where grid applications developed according to a web service paradigm (and mainly the Java programming language) could be deployed and easily used. This has required the implementation and customization of specific components that address web services assets while maintaining a compatibility with the primarily grid infrastructure. All the components that are added to the current grid infrastructure can be viewed as a method for an integration solution. Thanks to the Java platform, the runtime environment providing discovery and composition solutions (UDDI registry and BPEL engine) allows for separate modules. This structure allows for scalability and the possibility of separately managing and implementing such components in a grid machine that hosts an application server and a SOAP engine. A security solution employing the message handler technique may hide implementation specifics and could provide future capabilities like adding SAML assertion, providing XML encryption capabilities and so forth.

References:

[1] Maozhen Li, mark Baker, *The Grid: Core Technologies*, John Wiley and Sons. , 2005

[2] Kennet P. Birman, *Reliable Distributed Systems: Technologies, Web Services and Applications*, Springer, 2005

[3] Andrew S. Tanenbaum, Maarten van Steen, *Distributed Systems: Principles and Paradigms*, Prentice Hall, 1st edition 2002.

[4] J.C. McDowell, "Downloading the sky", *IEEE Spectrum Online*, 2004

[5] S. Pastore, Implementing a web services architectural framework in a grid infrastructure *GESTS International Transactions on Computer Science and Engineering*, Vol. 18-20, N.1, 2005

[6] A. Volpato, et al., Astronomical database related applications in the Grid.it project", proc. of *Astronomical Data Analysis Software & System*, October 2004, Pasadena, California.

[7] S. Campana, M. Litmaath, A. Sciaba', LCG-2 Middleware Overview, 498079, 2004.

[8] G. Taffoni, et al., Grid Data Source Engine: Grid gateway to the Virtual Observatory, *Astronomical Data Analysis Software and System XV*, ASP Conf. Series, Vol. XXX, 2005.

[9] S. Pastore, The service discovery methods issue: A Web Services UDDI specification framework integrated in a grid environment, *Special Issue Journal of Network and Computer Applications*, Elsevier Science, to be published 2006

[10] J. Colgrave, K. Januszewski, "Using WSDL in a UDDI Registry, Version 2.0.2 ", Technical Note OASIS Spec TC, 2004

[11] N. Srinivasan, M. Paolucci and K. Sycara, An Efficient Algorithm for OWL-S based Semantic Search in UDDI Int. *Work. on Semantic Web Services and Web Process Composition*, San Diego, California, USA, 2004.

[12] Foster, I., et al. A Security Architecture for Computational Grids. in 5th *ACM Conference on Computer and Communications Security*. 1998.

[13] R. Alfieri, et al., From gridmap-file to VOMS: managing authorization in a Grid environment". *Future Generation Computer System 21* (2005) 549-558

[14] S. Pastore et al., Delivering secure web services within INFN-GRID infrastructure by means of EDG security, *Proc. Of Workshop On Grid security and experience*, 2004