Design of a OGSA-Compliant Grid Information Service Using .NET Technologies

Ranieri Baraglia¹, Domenico Laforenza¹,
Angelo Gaeta², Pierluigi Ritrovato², and Matteo Gaeta²

¹ Information Science and Technologies Institute, Italian National Research Council,
Pisa, Italy
(ranieri.baraglia, domenico.laforenza)@isti.cnr.it
² CRMPA - University of Salerno,
Salerno, Italy
(agaeta, ritrovato, gaeta)@crmpa.unisa.it

Abstract. The Grid’s new trend is represented by the migration towards a model built on concepts and technologies that are inherited from Grid and Web Services communities. From the merging of these two technologies, the new concept of Grid Service has emerged and has been formalized with the introduction of the Open Grid Services Architecture (OGSA). OGSA defines the semantics of Grid Service instance, such as, how it is created, how it is named, how its lifetime is determined. OGSA does not place any requirements on implementation aspects. No mandatory constraint is requested for implementing Grid Services; the developers can exploit the features that characterize their target implementation environment.

In the framework of the Grid Based Application Service Provision (GRASP) European project, we are developing a OGSA-compliant middleware by using .NET technologies. According to this goal we are investigating the design of a Grid Information Service (GIS) by exploiting these technologies, which we consider an interesting operating environment for Grid Services, due to the high level integration of the services provided. Advantages and drawbacks of both these technologies and our solution are pointed out.

Keywords: Grid, OGSA, Middleware, Grid Information Service, Web Services.

1 Introduction

A Grid Information Service (GIS) maintains information about people, software, services and hardware that participate in a Grid, and more generally in a virtual

---

³ Work partially supported by the European Community under the Innovation Society Technologies (IST) programme of the 5th Framework Programme for RTD - project GRASP, contract no.: IST-2001-35464. This document does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of data appearing therein.
organization (VO). This wide correlated set of static and dynamic information is available upon request. A GIS should also provide protocols, methods, APIs, etc., for description, discovery, lookup, binding and data protection [1].

In a Grid environment there are much diversities in terms of the nature of the information sources (e.g. physical characteristics, location, sensitivity), the demands placed on those information sources (e.g. access rates, query vs. subscribe, accuracy demands) and the ways in which information is employed (e.g. discovery, brokering, monitoring).

Currently, GIS architecture is based on two fundamental entities: highly distributed information providers and specialized aggregate directory services. Information providers form a common VO neutral infrastructure providing access to detailed and dynamic information about Grid entities. Aggregate directories provide, often specialized, VO-specific views of federated resources, services and so on [2].

The goal of this paper is to investigate the design of a GIS by using .NET technologies, which we consider an interesting operating environment for Grid Services, due to the high level integration of the services provided. This work was conducted within the activities related to the Grid Based Application Service Provision (GRASP) European project [3]. One of the main project objectives is to design and implement a layered architecture for Application Service Provision (ASP) using GRID technologies. The GRASP system environment internal roles and their interactions are shown in Figure 1. The internal roles are shown as rectangles and the relations among internal roles and entities (e.g., service instance, service directory) are shown using dashed lines. The Information Service, shown in Figure 1 as clouds with inside "I", is not a centralized one, but it is distributed inside the GRASP environment. Figure 1 also shows the interactions among internal roles and the Information Service.

The paper is organized as follows. Section 2 and Section 3 describe "a service view" of an information service implemented in GT3, and the Microsoft (MS) technologies that can be exploited to implement a OGSA-compliant GIS, respectively. Section 4 describes a possible use of a GIS in a service-oriented environment. Finally, in Section 5 we draw some conclusions along with future directions.

2 The current service view of an Information Service

As reference for our work we have considered the Information Service developed in the Globus Toolkit (GT3) [4]. GT3 is based on Open Grid Service Architecture (OGSA) specifications [5]. In GT3, Monitoring and Discovery Services (MDS) are changed in fundamental ways with respect to the GT2 [6]. Here there is not a standalone service (like the MDS in GT2) but the Information Service is built upon Service Data and Service Data Definitions, which are part of the OGSA core, and specialized collective-layer Index Services that are delivered as part of GT3.
Fig. 1. Interactions between internal roles and entities of the GRASP environment.

The OGSI [7] core in GT3 provides a generic interface for mapping Service Data queries and subscriptions for Service Data notification to service implementation mechanisms. In essence, this subsumes the role of the Grid Resource Information Service (GRIS) backend server module in MDS, while relying on more basic OGSA binding mechanisms for secure access to the query interface in place of the GSI-enabled LDAP protocol [8].

GT3 will provide Service Data Providers as part of its base service implementation. These Service Data sources within the base service implementations subsume the role of the host-information provider scripts in MDS. The base service implementations expose status information as well as probed, measured, or discovered platform information according to well-defined Service Data Definitions in their Service Type WSDL [9]. These Service Data Definitions in WSDL subsume the role of MDS schema written in the RFC2252 LDAP schema format, and are used to standardize the information exchange between providers and consumers within specific service domains.

The Index Service components in GT3 provide the following features: a) An interface for connecting external information provider programs to service instance, b) A generic framework to aggregate Service Data, c) A registry of Grid Services. Typically, there is one Index Service per VO. However, when multiple large sites compose a VO, each site can run its own Index Service that will index the resources available at site level. In this case each Index Services would be included in the VO's Index Service.
With the GT3 Index Service, there will be multiple ways to obtain information: MDS-style query propagation, Cache prefetch (each source of information can be independently selected for cache prefetch), and Subscription/Notification. In Figure 2 a “service view” of the Information Service is shown.

![Diagram of the Information Service and Grid Services](image)

**Fig. 2.** An information service view implemented in GT3.

According to the service oriented view, each resource is virtualized in a service. In Figure 2 the Grid Services representing resources are depicted. Each resource has a Service Data Provider that takes the role of the GRIS service of MDS. As mentioned above, GT3 offers some Service Data Providers but it is even possible to use customized and/or external providers. Like in MDS, a client can directly query a resource by invoking the FindServiceData operation of the Grid Service PortType.

Each Grid Service instance has a particular set of service data associated with it. The essence of Index Service is to provide an interface for operations such as accessing, aggregating, generating, and querying service data. The Service Data of each resource can be set by subscription to an Aggregator mechanism in Index Service. Index Service is equivalent to the Grid Index Information Service (GIIS) [10]; it collects information from registered "child". The FindServiceData operation of the GridService interface is used to query this information.

3 .NET and other MS Technologies to build a GIS

In order to implement a OGSA-compliant GIS by exploiting .NET technologies we have to address some issues like the following:

1. implementation of Grid Services with .NET technologies,
2. managing the dynamic nature of the information describing the resources (e.g., CPU usage, Memory usage, etc.),
3. extension of the MS directory service functionalities (e.g., Active Directory) in order to implement the OGSA Index Service functionalities.

The first issue is related to the implementation of Grid Service Specification prescribed in [7], in a MS .NET language [11], [12]. In the frame of the GRASP project, we have selected the implementation of Grid Service Specification provided by the Grid Computing Group of the Virginia University, named OGSI.NET [13].

To manage the dynamic nature of information describing the resources, GT3 leverages on Service Data Providers. In MS environment we rely on Performance Counters and Windows Management Instrumentation (WMI) architecture to implement the Service Data Providers. For each component of a MS system we have a performance object (e.g. Processor Object) gathering all performance data of the related entity. Each performance object provides a set of Performance Counters that retrieves specific performance data regarding the resource associated to the performance object. For example, the "%Processor Time" is a Performance Counter of the Processor Object representing the percentage of time during which the processor is executing a thread. The performance counters are based on services at operating system level, and they are integrated in the .NET platform. In fact, .NET Framework [11] provides a set of APIs that allows the management of the performance counters.

To perform the collection and provisioning of the performance data to an index service, we leverage on Windows Management Instrumentation (WMI) [14] architecture. WMI is a unifying architecture that allows access to data from a variety of underlying technologies. WMI is based on the Common Information Model (CIM) schema, which is an industry standard specification driven by the Distributed Management Task Force (DMTF) [15]. WMI provides a three-tiered approach for collecting and providing management data. This approach consists of a standard mechanism for storing data, a standard protocol for obtaining and distributing management data, and a WMI provider. A WMI provider is a Win32 Dynamic-Link Library (DLL) that supplies instrumentation data for parts of the CIM schema. Figure 3 shows the architecture of WMI.

When a request for management information comes from a consumer (see Figure 3) to the CIM Object Manager (CIMON), the latter evaluates the request, identifies which provider has the information, and returns the data to the consumer. The consumer requests only the information it wants, and it never knows the information source or any details of how the information data are extracted from the underlying API. The CIMON and the CIM repository are implemented as a system service, called WinMgmt, and are accessed through a set of Component Object Model (COM) interfaces.

WMI provides an abstraction layer that offers access to much system information about hardware and software and many functions to make calculations on the collected values. The combination of performance counters and WMI realizes a Service Data Provider that each resource in a VO should provide.
To implement an OGSA-compliant Index Service, we exploit Active Directory (AD) features [16]. AD is a directory service designed for distributed networking environments providing secure, structured, hierarchical storage of information about interesting objects, such as users, computers, services, inside an enterprise network. AD provides a rich support for locating and working with these objects, allowing the organizations to efficiently share and manage information about network resources and users. It acts as the central authority for network security, letting the operating system to readily verify a user identity and to control his/her access to network resources.

Our goal is to implement a Grid Service that, taking the role of a consumer (see Figure 3), queries at regular intervals the Service Data Providers of a VO (see Figure 5) to obtain resources information, collect and aggregate this information, and allows to perform searches, among the resources of a organization, matching a specified criteria (e.g. to search for a machine with a specified number of CPUs). In our environment this Grid Service is called Global Information Grid Service (GIGS) (see Figure 5).

The hosts that run the GIGS have to be Domain Controllers (DC). A DC is a server computer, running on Microsoft WindowsNT, Windows2000, or Windows Server 2003 family operating systems that manages security for a domain. The use of a DC permits us to create a global catalog of all the objects that reside in an organization, that is the primary goal of AD services. This scenario is depicted in Figure 4 (a) where black-coloured machines run GIGS and the other
ones are Service Data Providers. Obviously, black-coloured hosts could also be Service Data Providers.

![Diagram](image)

**Fig. 4.** Active Directory partitioning scheme.

In order to avoid that the catalog grows too big it becomes slow and clumsy, AD is partitioned in units, the triangles in Figure 4 (a). For each unit there is at least a domain controller. The AD partitioning scheme emulates the Windows 2000 domain hierarchy (see Figure 4 (b)). Consequently, the unit of partition for AD services is the domain. GIGS has to implement an interface in order to obtain, using a publish/subscribe method, a set of data from Service Data Providers describing an active directory object. Such data are then recorded in the AD by using Active Directory Service Interface (ADSI), a COM based interface to perform common tasks, such as adding new objects.

After having stored those data in AD, the GIGS should be able to query AD for retrieving such data. This is obtained exploiting the Directory Services Markup Language (DSML) Services for Windows (DSW). DSML provides a means of representing directory structural information and directory operations as an XML document. The purpose of DSML is to allow XML-based enterprise applications to leverage profile and resource information from a directory in their native environment. DSW uses open standards such as HTTP, XML, and SOAP, so a high level of interoperability is possible. Furthermore, it supports DSML version 2 (DSMLv2), a standard supported by OASIS [19]. The solution is schematized in Figure 5, where each host is virtualized in a grid service.

4 **Scenario**

The scenario that we will show in this section will be developed in the frame of the GRASP project. It is presented in order to demonstrate a possible use of a GIS in a service oriented environment.

The scenario here considered is made up of two different VOs, each of them encloses a set of Service Providers (SPs) that offer a set of services for their clients within or outside VOs (see Figure 6). Different domains, called **Hosting**
Fig. 5. Example of GIS in MS Environment.

Environments (HEs), compose a VO and each domain has, at least, one domain controller. The domains of a VO are connected by trust relationships, and share a common schema (e.g., a set of definitions that specifies the kinds of objects, and information related to those objects, that can be stored in Active Directory). This configuration is called domain tree.

Each HE could have a Service Locator (SL) and has a Service Instantiator (SI). The SL is devoted to locate services hosted in a VO. So, for each VO there must be at least one SL that can communicate with the SIs of the other VO.s. The SI is devoted to instantiate the required services on the machine that guarantees the requested Quality of Service (QoS). In Figure 4 (b), the three triangles show an example of HEs that compose a VO.

According to this vision, the GIS can be efficiently distributed among all the HEs of a VO using the Global Catalog Server of Active Directory (GCS). A GCS is a domain controller that stores a copy of all Active Directory objects in a forest, that is a logical structure formed by combining two or more Microsoft Windows 2000 or Windows Server 2003 domain trees. In addition, the GCS stores the most common accessed object's attributes, and a full copy of all objects in the directory for its host domain. Moreover, it stores copies of the most used
objects belonging to other domains within the same forest. GCS provides efficient searches without unnecessary referrals to domain controllers, and it is created automatically on the initial domain controller in the forest.

In Figure 6 the mentioned scenario is shown. It includes two VOIs: VOI is composed of a domain tree and VO2 is composed of a single domain. VOI and VO2 made up a forest. Each domain in a VO has one SI, in order to instantiate service instances inside the domain, and could have one SL. In Figure 6 there is just one Service Requestor (SR) component, however, due to implementation choices each VO can have its own SR. The SR implements the necessary functionalities to invoke and operate a service.

In Figure 6 the black-coloured machines are domain controllers and run the GIGS, and the numbers associated to each link represent the sequence of operations performed. When the SR issues a request for a service, it contacts (1) a SL of VOI passing it a structure that contains some attributes that identify the service being requested (e.g. QoS required, type of service). In order to execute more requests, the SL is made up of three processes that run in pipeline fashion:

1. **Request Recipient** (RR): receives and processes all service location requests,
2. **Request Manager** (RM): accesses its related Service Directory to find the requested service and sends to all the other SLs a request for the same service,
3. **Search Result Manager** (SRM): receives the research results (a Service Lists), both local and remote. SRM, after receiving the original request from the SR, acts as master that collects the Service Lists from all the other SLs and sends them to the SR.

When the processed request arrives to the RM, it searches for the requested service inside its own service directory (UDDI) and contacts (2) the Information Service (invoking the GIGS belonging to its domain) in order to obtain the addresses of the other SLs. Once obtained the addresses of the other SLs, the RM sends the request to them to perform the same search simultaneously.

RM sends the number of SLs involved in the search to the SRM. The search of a service is considered as completed when SRM has received a number of Service Lists equal to the number of SLs involved in the search.

The Information Service is invoked through GIGS. Since the search among all the others HIs of VOI and VO2 is done using the AD global catalog, the SL has to invoke only the GIGS of its HE. In Figure 6 with a dashed line we want to emphasize the features of the AD searches using the GCS. The GCS stores a partial copy of all objects and services (e.g. the other SLs) for all the other domains in the forest, which provides efficient searches without unnecessary referrals to domain controllers.

Structuring VOIs as a forest, this feature can be used to search in other VOIs. In order to belong to the same forest, VOIs must find an agreement; all trees in a forest share a common schema, configuration, and global catalog. All trees in a given forest trust each other according to transitive hierarchical Kerberos relationships. A forest exists as a set of cross-reference objects and Kerberos trust relationships known by the member trees.
Once received all the service lists, the SRM (3) sends the search results to the SR, which, at that point, knows the SPs that match the requirements of the requested service. After selected (4) the SP starts the SI (5) of the HE that is able to provide the required service.

The main tasks performed by the SI are:

1. checks the SP rights in order to verify if it can create the requested service with the requested QoS,
2. chooses, between all the hosts of a HE that could run a service instance of the required type, the host able to guarantee the requested requirements (e.g., QoS, Price),
3. manages pools of idle instances.

To perform these tasks, the SI has to interact with other subsystems of the GRASP environment, such as the Information Service and Accounting. For example, to choose the right factory the SI has to be able to refine the information about QoS required in a set of host-figures (e.g., CPU rate, Memory required). Factory Manager, an object of the SI subsystem, does this operation. It accepts in input a level of QoS related to the service, and refines it in hosts data. Factory Manager (6) performs a search in the Information Service, filtered by hosts data, price data and service type. A SI queries only the domain controller of its HE since it is able to instantiate only inside its HE. Information Service returns a list of hosts able to guarantee the QoS required. Factory Manager selects a host, eventually performing a more fine-grained selection on the basis of information.
received by other subsystems (e.g. the Monitoring subsystems of the selected hosts). By exploiting these two selection levels, we perform a load-balancing task inside the HE. Once obtained the right host, the SI (7) invokes the factory service on that host and obtains the service locator (a structure that contains zero or more GSHs and zero or more GSRs) [7] of the created instance. The service locator is then returned to the SP (8) and the SP returns it to the SR.

5 Conclusions

Our work was focused on providing a possible implementation of a Grid Information Service using .NET technologies, trying to take full advantage from the MS technologies features. This is a work in progress in the frame of the GRASP project. Then, due to the project requirements, it could change under way.

On the basis of our experience in the frame of the GRASP project, OGSA seems to be an ideal environment for development of dynamic, collaborative, multi-institutional and VO based scenarios, as the one described in this paper (See Section 4), where a global information model is fundamental. In our opinion, the generation of these kind of scenarios, identifying key technologies and addressing user’s needs in different fields such as eLearning, eHealth, eCommerce, is one of major challenges of ICT. OGSA tries to address this challenge by defining requirements and service to support systems and applications in both eScience and eBusiness. This is not, obviously, a simple task and feedbacks have to arrive from international projects and initiatives, as the aforementioned GRASP. One potential risk, in our opinion, is related to usability of Grid. Grid architectures and systems must not be used only by IT skilled persons but mainly by doctors, business managers and students, otherwise the big challenge of any new technology, that is to improve the life of every user, will be dropped. While OGSA defines a solid foundation upon which IT professionals can build systems and applications, research must be done in order to define and develop intelligent user interfaces that should help to bridge the “semantic gap”, that is the gap existing between user’s demands (in a natural language) and how these demands are interpreted by the infrastructure, making Grid easy to handle and transparent. These and other important properties are research topics of the so called Next Generation Grids (NGG)[20].

In our work, we want to leverage on MS technologies in order to:

1. validate the use of MS technologies to develop grid middleware and grid services,
2. verify advantages of MS technologies,
3. investigate interoperability between grid services developed with different technologies.

The OGSA semantic definition of Grid Service instance does not dictate a particular hosting environment for Grid Services and promotes the development on more environments such as J2EE, Websphere, .NET, and Sun One. Currently, there are few projects that are going on this way. In our opinion, in
order to obtain an integrated environment, composed by distributed, heterogeneous, dynamic "virtual organizations" formed by many and different resources, the development of different hosting environments should be investigated and encouraged.

The adopted approach has some advantages. The MS technologies described in the paper are fully integrated in all the new MS operating systems and the solution proposed to implement a GIS is based directly upon them. In this way, we rely on technologies optimized for .NET hosting environments and it means an improvement from a performance viewpoint.

For example, among the Service Data Providers provided by GT3 there is a Java-based host information data provider that produces the following types of data: CPU count, memory statistics, OS type, and logical disk volumes. Obviously, this solution can be used in Windows operating systems. On the other side, our solution for the development of Service Data Providers that is based on the Performance Counter and WMI architecture represents a valid and feasible alternative that doesn’t need integration with third party software.

Furthermore, the features of the Active Directory (AD) present several advantages when used to implement an Information Service. In particular, we have seen that the use of global catalog permits us to distribute the Information service among the domains of a Virtual Organization (VO), allowing more efficiently searches of information, avoiding the necessity to query all the domain controllers.

The experience conducted has shown that the AD is flexible, robust, efficient, extensible and scalable. However, if we want to take advantage of AD, the set of VO members must share a common schema, configuration, and global catalog.

The studies performed in this work have shown that MS technologies and MS operating systems features can be used to develop grid middleware service oriented. Following the new steps towards a refactoring of the OGSI in terms of Web Services [18], as future work we plan to investigate the features of the new Microsoft operating system named Longhorn [17]. It presents key innovations for Presentation Layer, for Data Layer and for Communications Layer. All these layers provide their functionalities in terms of Web Services and our effort will be focused to investigate how these functionalities can be extended to realize an infrastructure for the development of Grid Services. In this way a developer can rely on an OGSA compliant infrastructure “embedded” in the operating system.

References

17. http://msdn.microsoft.com/Longhorn/