Grid Computing and Web Services

Lect. Carmen TIMOFTE, PhD
Academy of Economic Studies, Bucharest, Romania

Grid computing makes it possible to dynamically share and coordinate dispersed, heterogeneous computing resources. Flexibility and ubiquity are essential characteristics of Web services technologies such as WSDL (Web Services Description Language), SOAP (Simple Object Access Protocol), and UDDI (Universal Description, Discovery, and Integration).
***Keywords**: OGSA, UDDI, SOAP, WSDL.*

Introduction

Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across "multiple" administrative domains based on their (resources) availability, capability, performance, cost, and users' quality-of-service requirements.

If distributed resources happen to be managed by a single, global centralized scheduling system, then it is a cluster. In cluster, all nodes work cooperatively with common goal and objective as a centralized, global resource manager performs the resource allocation. In Grid, each node has its own resource manager and allocation policy.

Why Grid?

The reasons to choose grid are the following:

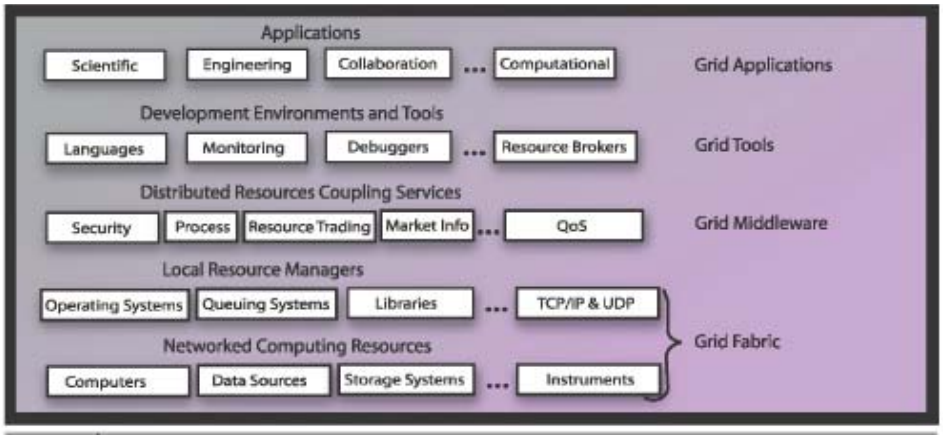
- The scale of the problems *human beings* have to face to perform frontier research in many different fields is constantly increasing;
- Performing frontier research in these fields already today requires world-wide collaborations (multi domain access to distributed resources);
- GRIDs naturally address this need for collecting and sharing resources (CPUs, data storage, data), providing – thanks to always growing throughputs and QoS in the underlying networks – unprecedented possibilities to access large data processing power and huge data storage and data access possibilities;
- Large community of possible GRID users: high energy physics, planet earth's health studies (*geology, environmental studies, earthquakes forecast, geologic and climate changes, ozone monitoring*), biology, genetics, earth observation, astrophysics, new composite materials research, astronautics.

OGSA

The Open Grid Services Architecture (OGSA) combines technologies to unlock and exploit grid-attached resources. OGSA defines mechanisms to create, manage, and exchange information between Grid Services, a special type of Web service. The architecture uses WSDL extensively to describe the structure and behavior of a service. Service descriptions are located and discovered using Web Services Inspection Language (WSIL). By combining elements from grid computing and Web services technologies, OGSA establishes an extensible and interoperable design and development framework for Grid Services that includes details for service definition, discovery, and life-cycle management.

OGSA defines and standardizes a set of (mostly) orthogonal multipurpose communication primitives that can be combined and customized by specific clients and services to yield powerful behavior. OGSA defines standard interfaces (portTypes in WSDL terminology) for basic Grid services.

Grid applications use grid tools and middleware components to interact with the fabric.



Relationship between grids, services and OGSA

OGSA introduces several service concepts that need to be adopted in order to qualify as Grid services. Necessary components are *factories, registries, and handle maps*. Additional mechanisms that can be used to build our data grid are *notification* and *lifetime management*.

From a software-design perspective, a grid can be viewed as a collection of self-contained computing services that can be described, published, located, and invoked over any type of network. Services are location-independent and dynamic, span multiple computing architectures, and reach across administrative domains. The design principles of OGSA reflect a combination of elements from Web services and grid computing:

- *Resource virtualization:* each grid component is considered a service.
- *Standard interface definition mechanisms:* enabling multiple protocol bindings and transparency between local or remote services.
- *Standard foundation services:* defining service semantics, reliability and security models, and core functions such as life-cycle management, or discovery.
- *Implementation independence:* support for multiple development languages and hosting environments including Java, COBOL, C, J2EE, CICS, and .NET.

Since a grid consists of both existing and new hardware and software, multiple variations of communication protocols, security schemes, and transaction management systems exist and cooperate at the same time. Today's grids tend to be a patchwork of protocols and noninteroperable "standards" and difficult-to-reuse "implementations." The OGSA uses Web services technologies like WSDL, SOAP, and WSIL to abstract platform and implementation differences, giving transparent access to grid services.

Fundamental to OGSA are WSDL and interfaces for dynamic discovery and life-cycle management for a specific type of Web service - a Grid Service. WSDL conventions and extensions are used to describe and structure services, while core service activities are expressed using WSDL interfaces and behaviors. Every Grid Service instance has a unique and immutable name called the Grid Service Handle. Lifetime management is provided by mandatory support for the operations Destroy and SetTerminationTime.

GridService

A Grid Service is a WSDL-defined service that follows a set of conventions relating to its interface definitions and behaviors. The specification details how clients create, discover, and interact with a Grid Service. Three characteristics separate a Grid Service from a regular Web service:

- A Grid Service is an instance of a service implementation of a service type, a collection of specific interfaces.
- The instance implements a Grid Services Handle. This is a Uniform Resource Indicator (URI) for the service instance and is bound to a Grid Service Reference (GSR). The GSR is similar, for example, to the Interoperable Object Reference (IOR) in CORBA and contains the necessary information to bind and use the service. The HandleMap interface is used to

map between the GSH and GSR. Introducing the GSR in the specification allows the separation of the service name from the service implementation, facilitating service evolution such as "non-stop" service upgrades.

- The instance also implements a port called GridService supporting three operations:
  - *FindServiceData:* Allows a client to discover information about the service's state, execution environment, and additional semantic details not available in the GSR. It is a form of reflection used by the service consumer to learn more about the service.
  - *Destroy:* An operation to explicitly destroy an instance
  - *SetTerminationTime:* A method to extend the lifetime of a service

In addition to the required *GridService* port, OGSA also defines additional, optional service ports. These interfaces define properties commonly required by distributed systems, such as messaging, discovery and registration, instance creation, and full life-cycle management.

The *NotificationSource* and *NotificationSink* ports handle messaging. This is a simple publish-subscribe system similar to Java's JMS, using XML messages. Existing messaging systems such as Tibco or MQ Series can also be used to implement the service. Error notification, application monitoring, and dynamic discovery of services are areas where the Notification ports are useful.

The *Registry* port is used to bind the metadata of Grid Service instances, identified by their GSH, to a registry. The registry is a particular type of Grid Service that maintains a set of Grid Service Handles and policies associated with the collection. Extracting information from the registry service is done using the *FindServiceData* method on the *GridService* port.

Instances of a Grid Service can be created via a *factory* or *manually*. Similarly, they can be destroyed via soft state termination or explicitly. The factory interface operation *CreateService* can create transient application service and supports reliable service instantiation (once-and-only-once). Interaction with the factory requires creation-lifetime information and can be extended to include an XML document describing service-specific creation parameters. The interface returns a GSH that uniquely identifies the instance over time. The soft-state lifetime management support avoids clients having to tear down services and prevents resource "leaks" in hosting environments.

Implementations

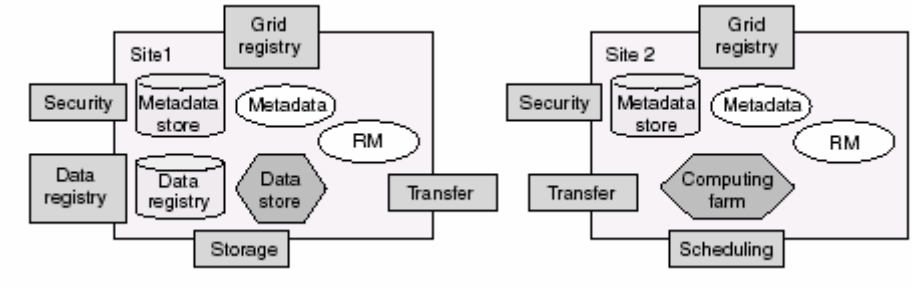
From a technical perspective, OGSA is critical to streamlining and accelerating the creation and deployment of grid applications. To expand the reach of grid applications beyond the level of a single enterprise the OGSA needs to more thoroughly address issues concerning:

- *Use of WSDL extensions* - The extensibility of WSDL has implications for interoperability with existing WSDL clients and servers. Full interoperability and accommodation of non-OGSA WSDL clients by grid servers will accelerate the adoption of the extensibility elements. This is important since the handling of extensions is spotty at best in most current toolkits.
- *Definition and use of service ports* - Discovery and life-cycle management deserve a separate interface. In the current OGSA specification, the services are combined in GridService. These services perform distinct roles and are used in different ways. For example, discovery of grid services can be open to anyone, while instantiation and destroy operations require a secure, authenticated connection.
- *Heterogeneous, end-to-end, security* - Addressing concerns around authentication, authorization, and trust relations is essential for grid implementations across organizational boundaries. The OGSA should include provisions for "pluggable" security mechanisms that can be discovered by the client using a service description. Service providers, in turn, can plug in a security architecture that fits their infrastructure for Grid Services.
- *Grid Service manageability* - Currently, the OGSA does not indicate how resources are exposed or, in an operational setting, how large sets of Grid Service instances can be monitored and managed. Grid Services are commonly subject to service level agreements that define specific availability, performance, and reliability characteristics.

A virtual organization (VO) example

The minimal set of services that a grid node should have depends on what kind of resources the node site offers. If the site has only storage capabilities, it will need most of the data management services discussed above. If it only has computing capabilities but no storage capabilities, then it needs only a subset of those, but it will need job management services instead. A site offering both storage and computing capabilities will require both data and job management services.

The next figure shows a simple VO layout with two sites and a possible set of services: one site capable of only storage and another capable of only computing. Ovals represent factories and rectangles represent exposed services. The replica manager (RM) factory may spawn other high-level data management services. There may be many other services such as monitoring and management services that are not shown in the picture.



Conclusion

OGSA brings utility-based computing a step closer, and allows the development and deployment of standards-based Grid Services today. The combination of Web services and grid computing virtualizes networked resources using common computing standards, making them accessible to a larger audience. To extend the reach of the architecture and maximize its potential benefits, additional services, such as security and management, need to become an integral part of the specification.

References

- www.cio.com, True Grid - Grid Technology, 2004;
- www.cio.com, In the Grid Game - CIO Magazine May 15,2004.htm;
- www.nwfusion.com/sungrid.html, Sun combines grid computing, Web services.htm, 2003;
- www.worldcommunitygrid.org/ Community Grid - About Us - Grid Computing Basics.htm, 2004
- www.globus.org/Globus Project Open Grid Services Architecture.htm, 2004;
- www.nwfusion.com, Grid and the Future of the Network Machine.htm, 2005;
- www.sys-con.com/webservices/ Web Services Journal, 2004;
- www.infoworld.com, Sun combines grid computing, Web services, 2003.
- www-1.ibm.com/grid
- wwws.sun.com/software/gridware
- www.microsoft.com/