**Unit – II**

**http://sumathigcc.blogspot.com/p/unit-2.html**

**GRID SERVICES**

**Open Grid Services Architecture (OGSA)**

            The OGSA is an open source grid service standard jointly developed by academia and the IT industry under coordination of a working group in the Global Grid Forum (GGF). The standard was specifically developed for the emerging grid and cloud service communities.

**OGSA Framework**

                        The OGSA was built on two basic software technologies: the Globus Toolkit widely adopted as a grid technology solution for scientific and technical computing, and web services (WS 2.0) as a popular standards-based framework for business and network applications. The OGSA is intended to support the creation, termination, management, and invocation of stateful, transient grid services via standard interfaces and conventions. The OGSA framework specifies the physical environment, security, infrastructure profile, resource provisioning, virtual domains, and execution environment for various

Grid services and API access tools.

**OGSA Interfaces**

                        The OGSA is centered on grid services. These services demand special well-defined application interfaces. These interfaces provide resource discovery, dynamic service creation, lifetime management, notification, and manageability. Two key properties of a grid service are transience and statefulness. These properties have significant implications regarding how a grid service is named, discovered, and managed. Being transient means the service can be created and destroyed dynamically; statefulness refers to the fact that one can distinguish one service instance from another.

**Grid Service Handle**

                        A GSH is a globally unique name that distinguishes a specific grid service instance from all others. The OGSA employs a “handle-resolution” mechanism for mapping from a GSH to a GSR. The GSH must be globally defined for a particular

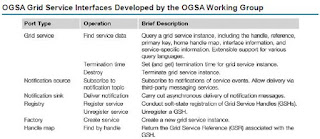
Instance.

**Grid Service Migration**

                        This is a mechanism for creating new services and specifying assertions regarding the lifetime of a service. The OGSA model defines a standard interface, known as a factor, to implement this reference. Any service that is created must address the former services as the reference of later services. Each dynamically created grid service instance is associated with a specified lifetime.

**OGSA Security Models**

                        The OGSA supports security enforcement at various levels. The grid works in a heterogeneous distributed environment, which is essentially open to the general public. At the security policy and user levels, we want to apply a service or endpoint policy, resource mapping rules, authorized access of critical resources, and privacy protection. At the Public Key Infrastructure (PKI) service level, the OGSA demands security binding with the security protocol stack and bridging of certificate authorities (CAs), use of multiple trusted intermediaries, and so on. Trust models and secure logging are often practiced in grid platforms.

[](https://2.bp.blogspot.com/-Wk6hasK6nwQ/V3Pu4g866bI/AAAAAAAAACo/ybcOpR67IlkMYwQLEQgOeDUD9ThJyNojACLcB/s1600/1.jpg)

**OGSA Services**

            The OGSA developed within the OGSA Working Group of the Global Grid Forum, is a service-oriented architecture that aims to define a common, standard, and open architecture for grid-based applications. “Open” refers to both the process to develop standards and the standards themselves. In OGSA, everything from registries, to computational tasks, to data resources is considered a service.

            These extensible set of services are the building blocks of an OGSA-based grid.

OGSA is intended to:

            • Facilitate use and management of resources across distributed, heterogeneous

Environments

            • Deliver seamless QoS

            • Define open, published interfaces in order to provide interoperability of diverse resources

            • Exploit industry-standard integration technologies

            • Develop standards that achieve interoperability

            • Integrate, virtualize, and manage services and resources in a distributed, heterogeneous environment

            • Deliver functionality as loosely coupled, interacting services aligned with industry accepted web service standards.

            OGSI, developed by the Global Grid Forum, gives a formal and technical specification of a grid service. Grid service interfaces correspond to *portType*s in WSDL. The set of *portType*s supported by a grid service, along with some additional information relating to versioning, are specified in the grid service’s *serviceType*, a WSDL extensibility element defined by OGSA. The interfaces address discovery, dynamic service creation, lifetime management, notification, and manageability; whereas the conventions address naming and upgradeability. Grid service implementations can target native platform facilities for integration with, and of, existing IT infrastructures.

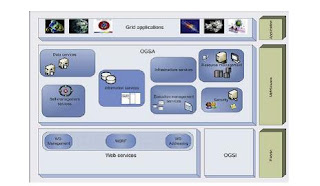
[](https://3.bp.blogspot.com/-SZA-JDVvTxc/V3PvCy71EcI/AAAAAAAAACw/ffz_uJGhgH4ixLPN23teVWdHitlLwRcCACLcB/s1600/2.jpg)

Figure: OGSA Architecture

OGSA services are summarized as follows:

• **Infrastructure Services**Refer to a set of common functionalities, such as naming, typically required by higher level services.

• **Execution Management Services**Concerned with issues such as starting and managing tasks, including placement, provisioning, and life-cycle management. Tasks may range from simple jobs to complex workflows or composite services.

• **Data Management Services**Provide functionality to move data to where it is needed, maintain replicated copies, run queries and updates, and transform data into new formats. These services must handle issues such as data consistency, persistency, and integrity. An OGSA data service is a web service that implements one or more of the base data interfaces to enable access to, and management of, data resources in a distributed environment. The three base interfaces, *Data Access, Data Factory*, and *Data Management*, define basic operations for representing, accessing, creating, and managing data.

• **Resource Management Services**Provide management capabilities for grid resources: management of the resources themselves, management of the resources as grid components, and management of the OGSA infrastructure. For example, resources

can be monitored, reserved, deployed, and configured as needed to meet application QoS requirements. It also requires an information model and data model (representation) of the grid resources and services.

• **Security Services**Facilitate the enforcement of security-related policies within a (virtual) organization, and supports safe resource sharing. Authentication, authorization, and integrity assurance are essential functionalities provided by these services.

• **Information Services**Provide efficient production of, and access to, information about the grid and its constituent resources. The term “information” refers to dynamic data or events used for status monitoring; relatively static data used for discovery; and any data that is logged. Troubleshooting is just one of the possible uses for information provided by these services.

• **Self-Management Services**Support service-level attainment for a set of services (or resources), with as much automation as possible, to reduce the costs and complexity of managing the system. These services are essential in addressing the increasing complexity of owning and operating an IT infrastructure.

**Functionality Requirements**

• Resource virtualization.

– Essential to reduce the complexity of managing heterogeneous systems and to handle diverse resources in a unified way.

• Common management capabilities.

– Mechanisms for uniform and consistent management of resources.

• Resource discovery and query.

– Discovering resources with desired attributes , retrieving their properties.

– Discovery and query should handle a highly dynamic and heterogeneous system.

• Standard protocols and schemas.

– Used to Simplify the transition to using Grids.

**PRACTICAL VIEW OF OGSA/OGSI**

                        It is called an *architecture*because it is mainly about describing and building a well-defined set of interfaces from which systems can be built, based on open standards such as WSDL.

The objectives of OGSA are:

         Manage resources across distributed heterogeneous platforms.

         Support QoS-oriented Service Level Agreements (SLAs). The topology of grids is often complex; the interactions between/among grid resources are almost invariably dynamic. It is critical that the grid provide robust services such as authorization, access control, and delegation.

         Provide a common base for autonomic management. A grid can contain a plethora of resources, along with an abundance of combinations of resource configurations, conceivable resource-to-resource interactions, and a litany of changing state and failure modes. Intelligent self-regulation and autonomic management of these resources is highly desirable.

         Define open, published interfaces and protocols for the interoperability of diverse resources. OGSA is an open standard managed by a standards body. Exploit industry standard integration technologies and leverage existing solutions where appropriate. The foundation of OGSA is rooted in Web services, for example, SOAP and WSDL, are a major part of this specification.

            OGSA’s companion OGSI document consists of specifications on how work is managed, distributed, and how service providers and grid services are described. .WSDL provides a simple method of describing and advertising the Web services that support the grid’s application. A set of services based on open and stable protocols can hide the complexity of service requests by users or by other elements of a grid. Grid services enable *virtualization;*virtualization, in turn, can transform computing into a ubiquitous infrastructure .OGSA relies on the definition of grid services in WSDL, which, as noted, defines, for this context, the *operations names, parameters,*and their *types*for grid service access .

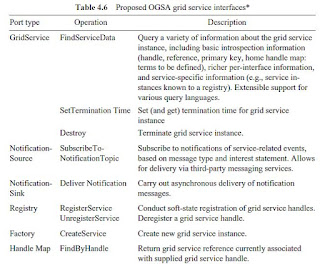
            It is an open and standards-based solution. This implies that, in the future, grid services can be built that are compatible with the OGSI standard, even though they may be based on a variety of different languages and platforms. The UDDI registry and WSIL document are used to locate grid services. The transport protocol SOAP is used to connect data and applications for accessing grid services.

            The interfaces of grid services address discovery, dynamic service-instance creation, lifetime management, notification, and manageability; the conventions of Grid services address naming and upgrading issues. The standard interface of a grid service includes multiple bindings and implementations. OGSA also provides a grid security mechanism to ensure that all the communications between services are secure.

. A grid service capability could be comprised of computational resources, storage resources, networks, programs, databases, and so on. A grid service implements one or more interfaces, where each interface defines a set of method operations that is invoked by constructing a method call through, method signature adaptation using SOAP.

            There are two fundamental requirements for describing Web services based on the OGSI:

            The ability to describe interface inheritance—a basic concept with most of the distributed object systems. The ability to describe additional information elements with the interface definitions.

[](https://1.bp.blogspot.com/-C75nz_T-hF0/V3PvUSdXiBI/AAAAAAAAAC4/LEclhtkugUsK7lySetU-u_q6ga-HmSOOgCLcB/s1600/3.jpg)

**A Detailed View of OGSA/OGSI**

**Introduction**

                        The OGSA integrates key grid technologies with Web services mechanisms to create a distributed system framework based on the OGSI. A *grid service instance*is a service that conforms to a set of conventions, expressed as WSDL interfaces, extensions, and behaviours, for such purposes as lifetime management, discovery of characteristics, and notification. Grid services provide for the controlled management of the distributed and often long-lived state that is commonly required in sophisticated distributed applications. OGSI also introduces standard factory and registration interfaces for creating and discovering grid services.

            OGSI defines a component model that extends WSDL and XML schema definition to incorporate the concepts of

         Stateful Web services

         Extension of Web services interfaces

         Asynchronous notification of state change

         References to instances of services

         Collections of service instances

         Service state data that augment the constraint capabilities of XML schema definition

The OGSI specifies (1) how grid service instances are named and referenced; (2) the base, common interfaces that all grid services implement; and (3) the additional interfaces and behaviours associated with factories and service groups.

**Setting the Context**

                        GGF calls OGSI the “base for OGSA.” Specifically, there is a relationship between OGSI and distributed object systems and also a relationship between OGSI and the existing Web services framework.

***Relationship to Distributed Object Systems.***

A given grid service implementation is an addressable and potentially stateful instance that implements one or more interfaces described by WSDL portTypes. Grid service factories can be used to create instances implementing a given set of portType(s). Each grid service instance has a notion of identity with respect to the other instances in the distributed grid. Each instance can be characterized as state coupled with behaviour published through type-specific operations.

                                    Grid service instances are made accessible to client applications through the use of a grid service handle and a grid service reference (GSR).A client application can use a grid service reference to send requests, represented by the operations defined in the portType(s) of the target service description directly to the specific instance at the specified network-attached service endpoint identified by the grid service reference.

***Client-Side Programming Patterns.***

                                    OGSI exploits an important component of the Web services framework: the use of WSDL to describe multiple protocol bindings, encoding styles, messaging styles, and so on, for a given Web service. The Web Services Invocation Framework (WSIF) and Java API for XML RPC (JAX-RPC) are among the many examples of infrastructure software that provide this capability.

                                    Various tools can take the WSDL description of the Web service and generate interface definitions in a wide range of programming-language-specific constructs.

                                     A *proxy*provides a client-side representation of remote service instance’s interface. Proxy behaviors specific to a particular encoding and network protocol are encapsulated in a *protocol-specific (binding-specific) stub*. This includes both application-specific services and common infrastructure services that are defined by OGSA.

***Client Use of Grid Service Handles and References.***

                                    A grid service handle (GSH) can be thought of as a permanent network pointer to a particular grid service instance. The client resolves a GSH into a GSR by invoking a HandleResolver grid service instance identified by some out-of-band mechanism. The HandleResolver may have the GSR stored in a local cache. The HandleResolver may need to invoke another HandleResolver to resolve the GSH.

***Relationship to Hosting Environment.***

OGSI does not dictate a particular service-provider-side implementation architecture. A container implementation may provide a range of functionality beyond simple argument demarshaling.

**The Grid Service**

                        The purpose of the OGSI document is to specify the (standardized) interfaces and behaviours that define a *grid service.*In brief, a grid service is a WSDL-defined service that conforms to a set of conventions relating to its interface definitions and behaviours.

The OGSI document expands upon this brief statement by

         Introducing a set of WSDL conventions that one uses in the grid service specification; these conventions have been incorporated in WSDL 1.2.

         Defining *service data*that provide a standard way for representing and querying metadata and state data from a service instance

         Introducing a series of core properties of grid service, including:

         Defining grid service description and grid service instance, as organizing principles for their extension and their use

         Defining how OGSI models time

         Defining the grid service handle and grid service reference constructs that are used to refer to grid service instances

         Defining a common approach for conveying fault information from operations.

         This approach defines a base XML schema definition and associated semantics for WSDL fault messages to support a common interpretation; the approach simply defines the base format for fault messages, without modifying the WSDL fault message model.

         Defining the life cycle of a grid service instance

**WSDL Extensions and Conventions**

It uses WSDL as the mechanism to describe the public interfaces of grid services. er, WSDL 1.1 is deficient in two critical areas: lack of interface (portType) extension and the inability to describe additional information elements on a portType.

**Service Data**

                        The approach to *stateful*Web services introduced in OGSI identified the need for a common mechanism to expose a service instance’s state data to service requestors for query, update, and change notification. The GGF is endeavouring to introduce this concept to the broader Web services community.

                        Service data can be exposed for read, update, or subscription purposes. Since WSDL defines operations and messages for portTypes, the declared state of a service must be externally accessed only through service operations defined as part of the service interface. To avoid the need to define service Data-specific operations for each service Data element, the grid service portType provides base operations for manipulating service Data elements by name. Elements of the publicly available state exposed by the service’s interface.

***Motivation and Comparison to JavaBean Properties.***

The OGSI specification introduces the serviceData concept to provide a flexible, properties-style approach to accessing state data of a Web service. The OGSI specification has chosen not to require getXXX and setXXX WSDL

Operations for each serviceData element.

***Extending portType with serviceData.***

ServiceData defines a new portType child element named serviceData, used to define serviceData elements, or SDEs, associated with that portType. These serviceData element definitions are referred to as serviceData declarations, or SDDs.

***serviceDataValues.***

Each service instance is associated with a collection of serviceData elements: those serviceData elements defined within the various portTypes that form the service’s interface, and also, potentially, additional service-Data elements added at runtime. OGSI calls the set of serviceData elements associated with a service instance its “serviceData set.”

                                    Each service instance must have a “logical” XML document, with a root element of serviceDataValues that contains the serviceData element values. An example of a serviceDataValues element was given above.

***SDE Aggregation within a portType Interface Hierarchy.***

                                    WSDL 1.2 has introduced the notion of multiple portType extension, and one can model that construct within the GWSDL namespace. A portType can extend zero or more other portTypes. There is no direct relationship between a wsdl: service and the portTypes supported by the service modeled in the WSDL syntax.”

***Dynamic serviceData Elements.***

                                    The grid service portType illustrates the use of dynamic SDEs. This contains a serviceData element named “serviceDataName” that lists the serviceData elements currently defined.

**Core Grid Service Properties.**

This subsection discusses a number of properties and concepts common to all grid services.

***Service Description and Service Instance.***

One can distinguish in OGSI between the *description*of a grid service and an *instance*of a grid service:

         A *grid service description*describes how a client interacts with service instances. This description is independent of any particular instance. Within a WSDL document, the grid service description is embodied in the most derived

portType of the instance, along with its associated portTypes bindings, messages, and types definitions.

         A grid service description may be simultaneously used by any number of *grid service instances,*each of which

         Embodies some state with which the service description describes how to interact

         Has one or more grid service handles

         Has one or more grid service references to it

A service description is used primarily for two purposes.

             First, as a description of a service interface, it can be used by tooling to automatically generate client interface proxies, server skeletons, and so forth.

            Second, it can be used for discovery, for example, to find a service instance that implements a particular service description, or to find a factory that can create instances with a particular service description.

The service description is meant to capture both interface syntax and semantics. *Interface syntax*is described by WSDL portTypes. *Semantics*may be inferred through the name assigned to the portType.

***Modeling Time in OGSI.***The need arises at various points throughout this specification to represent time that is meaningful to multiple parties in the distributed Grid. Clients need to negotiate service instance lifetimes with services, and multiple services may need a common understanding of time in order for clients to be able to manage their simultaneous use and interaction.

**Data intensive grid service models**

                        Applications in the grid are normally grouped into two categories: computation-intensive and data-intensive. The grid system must be specially designed to discover, transfer, and manipulate these massive data sets. Transferring massive data sets is a time-consuming task. Efficient data management demands low-cost storage and high-speed data movement.

**Data Replication and Unified Namespace**

                        This data access method is also known as caching, which is often applied to enhance data efficiency in a grid environment. By replicating the same data blocks and scattering them in multiple regions of a grid, users can access the same data with locality of references. Furthermore, the replicas of the same data set can be a backup for one another. Some key data will not be lost in case of failures. The increase in storage requirements and network bandwidth may cause additional problems.

Replication strategies determine when and where to create a replica of the data. The factors to consider include data demand, network conditions, and transfer cost. The strategies of replication can be classified into method types: dynamic and static. Dynamic strategies can adjust locations and number of data replicas according to changes in conditions.

The most common replication strategies include preserving locality, minimizing update costs, and maximizing profits.

**Grid Data Access Models**

                        Multiple participants may want to share the same data collection. To retrieve any piece of data, we need a grid with a unique global namespace. There are four access models for organizing a data grid

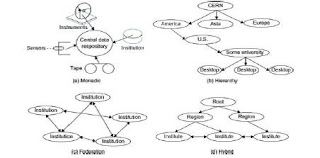
[](https://4.bp.blogspot.com/-mC7YMtqCk1Y/V3PvdRKrwJI/AAAAAAAAADA/2TC119cMsBENJ3G7PhX40MGpbEIAkeGKwCLcB/s1600/4.jpg)

Figure: Four architectural models for building a data grid.

**Monadic model:** This is a centralized data repository model. All the data is saved in a central data repository. When users want to access some data they have to submit requests directly to the central repository. No data is replicated for preserving data locality. This model is the simplest to implement for a small grid. For a large grid, this model is not efficient in terms of performance and reliability. Data replication is permitted in this model only when fault tolerance is demanded.

**Hierarchical model:** The hierarchical model, is suitable for building a large data grid which has only one large data access directory. The data may be transferred from the source to a second-level center. Then some data in the regional center is transferred to the third-level center. After being forwarded several times, specific data objects are accessed directly by users. Generally speaking, a higher-level data center has a wider coverage area. It provides higher bandwidth for access than a lower-level data center. KI security services are easier to implement in this hierarchical data access model.

**Federation model:** This data access model is better suited for designing a data grid with multiple sources of data supplies. Sometimes this model is also known as a mesh model. The data sources are distributed to many different locations. Although the data is shared, the data items are still owned and controlled by their original owners. According to predefined access policies, only authenticated users are authorized to request data from any data source. This mesh model may cost the most when the number of grid institutions becomes very large.

**Hybrid model:** This data access model combines the best features of the hierarchical and mesh models. Traditional data transfer technology, such as FTP, applies for networks with lower bandwidth. Network links in a data grid often have fairly high bandwidth, and other data transfer models are exploited by high-speed data transfer tools such as GridFTP developed with the Globus library. The cost of the hybrid model can be traded off between the two extreme models for hierarchical and mesh-connected grids.

***Parallel versus Striped Data Transfers***

                        Compared with traditional FTP data transfer, *parallel data transfer*opens multiple data streams for passing subdivided segments of a file simultaneously.

. In *striped data transfer*, a data object is partitioned into a number of sections, and each section is placed in an individual site in a data grid. When a user requests this piece of data, a data stream is created for each site, and all the sections of data objects are transferred simultaneously. Striped data transfer can utilize the bandwidths of multiple sites more efficiently to speed up data transfer.

**OGSA Services**

         Handle Resolution

         Virtual Organization Creation and Management

         Service Groups and Discovery Services

         Choreography, Orchestrations and Workflow

         Transactions

         Metering Service

         Rating Service

         Accounting Service

         Billing and Payment Service

         Installation, Deployment, and Provisioning

         Distributed Logging

         Messaging and Queuing

         Event

         Policy and Agreements

         Base Data Services

         Other Data Services

         Discovery Services

         Job Agreement Service

         Reservation Agreement Service

         Data Access Agreement Service

         Queuing Service

         Open Grid Services Infrastructure

         Common Management Model