UNIT II

GRID SERVICES

Introduction to Open Grid Services Architecture (OGSA) – Motivation – Functionality
Requirements – Practical & Detailed view of OGSA/OGSI – Data intensive grid service models – OGSA services.

2. Introduction to Open Grid Services Architecture (OGSA)

- Initially grid standard published by the Global Grid Forum – OGSI. (OGSI defines grid services and the basic mechanisms for creating, managing, and exchanging information between them).
- A second standard appeared a year later (and was still in initial draft form at press time): the Open Grid Services Architecture (OGSA).
- Open Grid Services Architecture (OGSA), the technology driving it, and problems addressed by the Grid Service Specification proposed by the Global Grid Forum (GGF)

2.1 MOTIVATION FOR OGSA

OGSA has three main predecessors:

- Globus Toolkit
- Autonomic computing initiative
- Emerging Web Services standards

Globus Toolkit

- Globus Toolkit Version 2 has been considered by many as the de facto standard for the implementation of grids, providing many key technologies, such as directory service for resource discovery, authentication, and job scheduling.
- The toolkit grew from a realization that to make a grid work for scientists and engineers required a significant middleware infrastructure.

Autonomic computing

- The IBM autonomic computing initiative has much in common with Globus in that it strives to unify a user’s view of a heterogeneous distributed system through the provision of common facilities.
• It has long been recognized that middleware requires a set of functionality, such as logging, security, failover, clustering, heartbeat monitoring, first failure data capture, trace, and so on across a variety of platforms and technologies.
• Application servers, databases, or messaging engines all require similar functionality, but are often implemented in different ways by different middleware. For this reason, IBM embarked in this project to provide a common set of core infrastructure functions across all platforms.

Emerging Web Services standards
• Through a significant cooperative venture by some of the biggest players in the industry (including IBM and Microsoft) working through a number of standards organizations, a common set of technologies have emerged. These technologies allow for the registration, discovery, and use of distributed services.

2.1.1 Basis for OGSA
• Web Services
• The Web Services model
• Operations in a Web Service architecture
• Standards
• Grid security

2.1.1.1 Web Services
• Web Services is an interface that describes a collection of operations that are network-accessible through standard XML messaging.
• Web Services is intended to facilitate the conversation or communication between computer programs.
• Web Services will do for program-to-program communication
• Web Services uses a program-to-program communications model built on existing standards, such as Hyper Text Transmission Protocol (HTTP), Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description Discovery, and Integration (UDDI).
2.1.1.2 The Web Services model

![Figure 2.1 Roles in the Web Service model](image)

**Figure 2.1 Roles in the Web Service model**

- **Service Requestor**
  
  This is the business entity asking for a resource or service. From an architectural viewpoint, this is the application that is looking and invoking or initiating an interaction with the service. The Service Requestor role can be played by a browser in the hands of a person or by another service provider.

- **Service Provider**
  
  This is the resource owner or service owner. From an architectural viewpoint, this is the environment that hosts access to the service.

- **Service Registry**
  
  This is where requestors can find information about available resources or services. This is a searchable registry in which service providers publish their service descriptions and the availability of their services. Service Requestors find services and obtain binding information from the service descriptions for static binding or dynamic binding. A Service Requestor can bind directly to a Service Provider if they already have the binding information, then the Service Registry can be considered optional.

2.1.1.3 Operations in a Web Service architecture

- Three different actions or operations can be taken that are related to a resource: 
  
  *Publish, Find and Bind.*

- **Publish:**
  
  To be accessible, a service description needs to be known (published).
• **Find:**
The entity looking for a specific type of resource must find the resource with the necessary characteristics.

• **Bind:**
Eventually, the service can be invoked and bound to be used.

• The Web Services architecture defines two important concepts that are called artifacts. They are:
  • **Service**
    – This is the implementation of the Web Service interface.
  • **Service Description**
    – This is the details of the interface and the implementation of the service. This includes its data types, operations, binding information and network location. The service description might be published directly to a service requestor or to a service registry.

2.1.1.4 **Standards**

• OGSA proposes the heavy use of three standards:
  – *Simple Object Access Protocol* (SOAP),
  – *Web Service Description Language* (WSDL), and
  – *Web Service Inspection* (WSI).

**SOAP**

• This is a means for providing messaging between a Service Requestor and a Service Provider.
• It is a simple enveloping mechanism for XML payloads and defines a Remote Procedure Call (RPC) mechanism and conventions.
• SOAP payloads can be carried on HTTP, FTP, Java Messaging Service (JMS), and so on.

**WSDL**

This is a XML document for describing Web Services as a set of endpoints on messages containing either document-oriented (messaging) or RPC payloads.
WSI

- This is a simple XML language, with related conventions for locating service descriptions published by a service provider.
- A WSI language (WSIL) document can contain a collection of service descriptions and links to other source of service descriptions. It is a service description and, normally, is a URL to a WSDL document, but occasionally can be a URL to another WSI document.
- With WS-Inspection, a service provider creates a WSIL document and makes the document network accessible.

The Web Services stack

![Web Services Conceptual Stack](image)

**Figure 2.2 Web Services Conceptual Stack**

**XML messaging using SOAP:**

1. A service requestor’s application creates a SOAP message invoking a Web service operation provided by a service provider.
2. The network infrastructure delivers the message to the service provider’s SOAP runtime (for example, a SOAP server).
3. The response SOAP message is presented to the SOAP runtime with the service requestor as the destination.
4. The response message is receive by the networking infrastructure on the service requestor’s node.
2.1.1.5 Grid security:

The minimal functional requirements to provide security for the service and resources are:

- **Authentication**
  Verifying the validity of a claimed individual and identifying who he or she is.

- **Access Control**
  Assurance that each user or computer that uses the service is permitted to do what he or she asks for.

- **Data Confidentiality**
  Assurance that sensitive information must not be revealed to parties that it was not meant for.

- **Data Integrity**
  Assurance that the data is not altered or destroyed in an unauthorized manner.

- **Key Management**
  The secure generation, distribution, authentication and storage of keys used in cryptography.

The Grid Security Infrastructure (GSI) and a Public Key Infrastructure (PKI) provide the technical framework.
2.1.2 OGSA in detail

- The Web Services Description Language (WSDL) provides a mechanism to define service interfaces in XML.
  - These descriptions set out the structure and sequence of exchanges between the invoker and the service.
- WSDL allows the same service to support multiple protocol bindings for a single interface.
  - This capability contributes greatly to support of heterogeneous distributed systems.
  - Not only can the binding be, for example, SOAP over JMS or HTTP, but a service can support multiple bindings for each offering, for example, different qualities of service or authentication mechanisms.
- OGSA brings the grid and Web service communities together to address the problem of services across a distributed, heterogeneous, dynamic, and virtual organization.
- The core of the OGSA architecture is the grid service.
- Grid services may be computational resources, storage resources, programs, or databases.
- Taking the Web services model as the example, grid services map very well to the concepts of registration, discovery, and use.
- The two critical aspects for users in such a
  - Service Orientated Architecture (SOA) are definition of the service interfaces and identification of the protocol(s) that can be used to invoke a given service.
- Grid Services extend the Web Services concept by laying out a set of well-defined interfaces that address discovery, dynamic service creation, lifetime management, notification, and manageability and a set of conventions for naming and upgrade ability.
- These interfaces and conventions are vital for allowing reliable interoperability between services and with invoking applications.
- WSDL refers these interface as portTypes.

These port Types include:

- Grid Service port Type
– To allow the discovery of data relating to the service and enable lifetime management of the service
  • Notification-Source port Type
    – To allow the sending of notification messages
  • Notification-Sink port Type
    – To allow the receiving of notification messages
  • Notification-Subscription port Type
    – To allow a Notification Source port Type to subscribe to a set of notifications for a period of time.
  • Registration port Type
    – To allow a service instance to register/unregister to enable/disable discover of the service instance
  • Factory port Type
    – To allow the creation of service instances
  • Handle Resolver port Type
    – To allow a grid service handle to be converted into a Grid Service Reference, necessary for binding to the service

2.2 FUNCTIONALITY REQUIREMENTS

• Basic Functionality Requirements
• Security Requirements
• Resource Management Requirements
• System Properties Requirements
• Other Functionality Requirements
• Discovery and brokering
• Metering and accounting
• Data sharing
• Deployment
• Virtual organizations
• Monitoring
• Policy
**Discovery and brokering:**

- Mechanisms are required for discovering and/or allocating services, data, and resources with desired properties.
- For example, clients need to discover network services before they are used, service brokers need to discover hardware and software availability, and service brokers must identify codes and platforms suitable for execution requested by the client.

**Metering and accounting:**

- Applications and schemas for metering, auditing, and billing for IT infrastructure and management use cases.
- The metering function records the usage and duration, especially metering the usage of licenses.
- The auditing function audits usage and application profiles on machines, and the billing function bills the user based on metering.

**Data sharing:**

- Data sharing and data management are common as well as important grid applications.
- Mechanisms are required for accessing and managing data archives, for caching data and managing its consistency, and for indexing and discovering data and metadata.

**Deployment:**

- Data is deployed to the hosting environment that will execute the job (or made available in or via a high-performance infrastructure).
Also, applications (executable) are migrated to the computer that will execute them.

**Virtual organizations (VOs):**

- The need to support collaborative VO introduces a need for mechanisms to support VO creation and management, including group membership services.
- For the commercial data center use case, the grid creates a VO in a data center that provides IT resources to the job upon the customer’s job request.
- Depending on the customer’s request, the grid will negotiate with another grid on a remote commercial data center and create a VO across the commercial data centers.
- Such a VO can be used to achieve the necessary scalability and availability.

Monitoring
• A global, cross-organizational view of resources and assets for project and fiscal planning, troubleshooting, and other purposes.
• The users want to monitor their applications running on the grid.
• Also, the resource or service owners need to surface certain states so that the user of those resources or services may manage the usage using the state information.

Policy:
• An error and event policy guides self-controlling management, including failover and provisioning.
• It is important to be able to represent policy at multiple stages in hierarchical systems, with the goal of automating the enforcement of policies that might otherwise be implemented as organizational processes or managed manually.
• There may be policies at every level of the infrastructure: from low-level policies that govern how the resources are monitored and managed, to high-level policies that govern how business process such as billing are managed.
• High-level policies are sometimes decomposable into lower-level policies.

Security Requirements
• Multiple security infrastructures
• Perimeter security solutions
• Authentication, Authorization, and Accounting
• Encryption
• Application and Network-Level Firewalls
• Certification.

Multiple security infrastructures:
• Distributed operation implies a need to interoperate with and manage multiple security infrastructures.
• For example, for a commercial data center application, isolation of customers in the same commercial data center is a crucial requirement; the grid should provide not only access control but also performance isolation.
• For another example, for an online media and entertainment use case, proper isolation between content offerings must be ensured; this level of isolation has to be ensured by the security of the infrastructure.
Perimeter security solutions:
• Many use cases require applications to be deployed on the other side of firewalls from the intended user clients.
• Inter grid collaboration often requires crossing institutional firewalls.
• OGSA needs standard, secure mechanisms that can be deployed to protect institutions while also enabling cross-firewall interaction.

Authentication, Authorization, and Accounting:
• Obtaining application programs and deploying them into a grid system may require authentication/authorization.
• In the commercial data center use case, the commercial data center authenticates the customer and authorizes the submitted request when the customer submits a job request.
• The commercial data center also identifies his/her policies (including but not limited to SLA, security, scheduling, and brokering policies).
• Encryption: The IT infrastructure and management use case requires encrypting of the communications, at least of the payload.

Application and Network-Level Firewalls:
• This is a long-standing problem; it is made particularly difficult by the many different policies one is dealing with and the particularly harsh restrictions at international sites.

Certification: A trusted party certifies that a particular service has certain semantic behavior.
• For example, a company could establish a policy of only using e-commerce services certified by Yahoo.

Resource Management Requirements
• Provisioning.
• Resource virtualization.
• Optimization of resource usage
• Transport management
• Access.
• Management and monitoring
• Processor scavenging
• Scheduling of service tasks
• Load balancing
• Advanced reservation.
• Notification and messaging
• Logging
• Workflow management.
• Pricing.

Provisioning:
• Computer processors, applications, licenses, storage, networks, and instruments are all grid resources that require provisioning.
• OGSA needs a framework that allows resource provisioning to be done in a uniform, consistent manner.

Resource virtualization:
• Dynamic provisioning implies a need for resource virtualization mechanisms that allow resources to be transitioned flexibly to different tasks as required; for example, when bringing more Web servers on line as demand exceeds a threshold.

Optimization of resource usage:
• While meeting cost targets (i.e., dealing with finite resources) mechanisms to manage conflicting demands from various organizations, groups, projects, and users to implement a fair sharing of resources and access to the grid.

Transport management:
• For applications that require some form of real-time scheduling, it can be important to be able to schedule or provision bandwidth dynamically for data transfers or in support of the other data sharing applications.
• In many (if not all) commercial applications, reliable transport management is essential to obtain the end-to-end QoS required by the application.
• Access. Usage models that provide for both batch and interactive access to resources.

Management and monitoring:
• Support for the management and monitoring of resource usage and the detection of SLA or contract violations by all relevant parties.
  Also, conflict management is necessary; it resolves conflicts between management disciplines that may differ in their optimization objectives
**Processor scavenging:**
- It is an important tool that allows an enterprise or VO to use to aggregate computing power that would otherwise go to waste.
- For example, consider a collection of desktop computers running software that supports integration into processing and/or storage pools managed via systems such as Condor, Entropia, and United Devices. Issues here include maximizing security in the absence of strong trust.

**Scheduling of service tasks:**
- Long recognized as an important capability for any information processing system, scheduling becomes extremely important and difficult for distributed grid systems.
- In general, dynamic scheduling is an essential component. Computer resources must be provisioned on-demand to satisfy the need to complete a forecast on time.

**Load balancing:**
- In many applications, it is necessary to make sure deadlines are met or resources are used uniformly.
- For example, for the commercial data center use case, monitoring the job performance and adjusting allocated resources to match the load and fairly distributing end users’ requests to all the resources are necessary.
- For the online media and entertainment use case, the amount of workload is a direct result of how many concurrent online game players are being hosted on a game server.

**Advanced reservation:**
- This functionality may be required in order to execute the application on reserved resources.
- For example, for the commercial data center use case, the grid decides when to start the request processing based on the customer’s request.
- It interprets the job specification description language in which the request is written and it checks to see if the customer has the right to perform the request.

**Notification and messaging**
- Notification and messaging are critical in most dynamic scientific problems. Notification and messaging are event driven.
Logging:

- It may be desirable to log processes such as obtaining/deploying application programs because, for example, the information might be used for accounting. This functionality is represented as “metering and accounting.”

Workflow management:

- Many applications can be wrapped in scripts or processes that require licenses and other resources from multiple sources. Applications coordinate using the file system based on events.

Pricing. Mechanisms for determining how to render appropriate bills to users of a grid.

System Properties Requirements

- Fault tolerance
- Disaster recovery.
- Self-healing capabilities
- Strong monitoring
- Legacy application management.
- Administration.
- Agreement-based interaction.
- Grouping/aggregation of services

Fault tolerance:

- Support is required for failover, load redistribution, and other techniques used to achieve fault tolerance. Fault tolerance is particularly important for long running queries that can potentially return large amounts of data, for dynamic scientific applications, and for commercial data center applications.

Disaster recovery:

- Disaster recovery is a critical capability for complex distributed grid infrastructures. For distributed systems, failure must be considered one of the natural behaviors and disaster recovery mechanisms must be considered an essential component of the design.
- Autonomous system principles must be embraced as one designs grid applications and should be reflected in OGSA.
• In case of commercial data center applications if the data center becomes unavailable due to a disaster such as an earthquake or fire, the remote backup data center needs to take over the application systems.

**Self-healing capabilities** of resources, services and systems are required.

- Significant manual effort should not be required to monitor, diagnose, and repair faults.
- There is a need for the ability to integrate intelligent self-aware hardware such as disks, networking devices, and so on.
- *Strong monitoring* for defects, intrusions, and other problems. Ability to migrate attacks away from critical areas.

**Legacy application management.** Legacy applications are those that cannot be changed, but they are too valuable to give up or to complex to rewrite. Grid infrastructure has to be built around them so that they can continue to be used.

**Administration.** Be able to “codify” and “automate” the normal practices used to administer the environment. The goal is that systems should be able to self organize and self-describe to manage low-level configuration details based on higher-level configurations and management policies specified by administrators.

**Agreement-based interaction.** Some initiatives require agreement-based interactions capable of specifying and enacting agreements between clients and servers (not necessarily human) and then composing those agreements into higher-level end-user structures.

**Grouping/aggregation of services.** The ability to instantiate (compose) services using some set of existing services is a key requirement. There are two main types of composition techniques: selection and aggregation.

- Selection involves choosing to use a particular service among many services with the same operational interface. Aggregation involves orchestrating a functional flow (workflow) between services. For example, the output of an accounting service is fed into the rating service to produce billing records.
- One other basic function required for aggregation services is to transform the syntax and/or semantics of data or interfaces.

**Other Functionality Requirements**

- **Platforms.** The platforms themselves are heterogeneous, including a variety of operating systems (Unixes, Linux, Windows, and, presumably, embedded systems), hosting
environments (J2EE, .NET, others), and devices (computers, instruments, sensors, storage systems, databases, networks, etc.).

- **Mechanisms.** Grid software can need to interoperate with a variety of distinct implementation mechanisms for core functions such as security.

- **Administrative environments.** Geographically distributed environments often feature varied usage, management, and administration policies (including policies applied by legislation) that need to be honored and managed.

- Other system components, including the following:
  - Both **single-process and multiprocess** (both local and distributed) applications covering a wide range of resource requirements.
  - **Flows,** that is, multiple interacting applications that can be treated as a single transient service instance working on behalf of a client or set of clients.
  - **Workloads** comprising potentially large numbers of applications with a number of characteristics just listed

### 2.3 OGSA/OGSI - A PRACTICAL VIEW

It is called an architecture because

- it is mainly about describing and building a well-defined set of interfaces

- from which Grid systems can be built, based on open standards such as WSDL

**Objectives of OGSA**

- Manage resources across distributed heterogeneous platforms
- Support QoS-oriented Service Level Agreements (SLAs)
- the interactions between/among grid resources are invariably dynamic.
- Provide a common base for autonomic management
- Define open, published interfaces and protocols for the interoperability of diverse resources.
- Exploit industry standard integration technologies and leverage existing solutions where appropriate.
- The foundation of OGSA is rooted from Web services.
- OGSI document consists of specifications
- OGSA relies on the definition of grid services in WSDL
- It defines Operation Names, parameters and its types
• All services adhere to specified grid service interfaces and behaviors.

**OGSA/OGSI - A MORE DETAILED VIEW**

• 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

The OGSI V1.0 specification

In Contrast, it specifies

• how grid service instances are named and referenced;
• The common interfaces (and associated behaviors) that all grid services implement and the additional (optional) interfaces and behaviors associated with factories and service groups.
• The specification does not address how grid services are created, managed, and destroyed within any particular hosting environment.
• So services that conform to the OGSI specification are not necessarily portable to various hosting environments (Servers)

**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 (and evolving) Web services framework.
• One needs to examine both the client-side programming patterns for grid services and a conceptual hosting environment for 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

**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 behavior published through type-specific operations.
• The architecture also supports introspection in that a client application can ask a grid service instance to return information describing itself, such as the collection of portTypes that it implements.
• Grid service instances are made accessible to (potentially remote) 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 stubs and helper classes isolate application programmers from the details of using grid service references. Some client-side infrastructure software assumes responsibility for directing an operation to a specific instance that the GSR identifies.
**Client-Side Programming Patterns.**

- how OGSI interfaces are likely to be invoked from client applications

OGSI exploits WSDL that describe

- multiple protocol bindings,
- encoding styles,
- messaging

The Web Services Invocation Framework (WSIF) and Java API for XML RPC (JAX-RPC) used to invoke the services.

It is possible, but not recommended, to build customized code that directly couples client applications to fixed bindings of a particular grid service instance.
Figure 2.6 Resolving a GSH

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

- grid service handle (GSH) can be thought of as a permanent network pointer to a particular grid service instance.

- The GSH does not provide sufficient information to allow a client to access the service instance.

- the client needs to “resolve” a GSH into a grid service reference (GSR).

- The GSR contains all the necessary information to access the service instance.

OGSI provides a mechanism, the HandleResolver to support client resolution of a grid service handle into a grid service reference.

- 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.

- The HandleResolver may use a handle resolution protocol, specified by the particular kind (or scheme) of the GSH to resolve to a GSR. The HandleResolver protocol is specific to the kind of GSH being resolved.
Relationship to Hosting Environment

- OGSi does not dictate a particular service-provider-side implementation architecture.
- A variety of approaches are possible,
  - From implementing the grid service instance directly as an operating system process
  - to a sophisticated server-side component model

![Diagram showing two approaches to the implementation of argument demarshaling functions in a grid service hosting environment](image)

Figure 2.6 Two approaches to the implementation of argument demarshaling functions in a grid service hosting environment

- A container implementation may provide a range of functionality beyond simple argument demarshaling.
  - lifetime management functions,
  - automatic support for authorization and authentication,
  - request logging, and
  - Terminating service instances
- avoids the need to reimplement these common behaviors
The Grid Service

The purpose of the OGSI document is to specify the (standardized) interfaces and behaviors that define a grid service.

- every grid service is a Web service the converse is not true.
- The OGSI document specifies Grid service creation includes
  - Introducing a set of WSDL conventions
  - 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
    - Defining the grid service handle and grid service reference
    - Defining how OGSI models time
    - Defining a common approach for conveying fault information from operations.
    - Defining the life cycle of a grid service instance

WSDL Extensions and Conventions

- 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
- WSDL 1.2 is a “work in progress” – cant be used now
- OGSI defines an extension to WSDL 1.1

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
- to expose Web service state, a data called “serviceData” was proposed
- Specified in the WSDL as part of the service interface
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 serviceData-specific operations for each serviceData element, the grid service portType provides base operations for manipulating serviceData elements by name.

Consider an example. Interface alpha introduces operations op1, op2, and op3.

Also assume that the alpha interface consists of publicly accessible data elements of de1, de2, and de3.

One uses WSDL to describe alpha and its operations. The OGSI serviceData construct extends WSDL so that the designer can further define the interface to alpha by declaring the public accessibility of certain parts of its state de1, de2, and de3.

This declaration then facilitates the execution of operations on the service data of a stateful service instance implementing the alpha interface.

The serviceData declaration is the mechanism used to express the elements of the publicly available state exposed by the service’s interface. ServiceData elements are accessible through operations of the service interfaces such as those defined in this specification.

The private internal state of the service instance is not part of the service interface and is therefore not represented through a serviceData declaration.

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 serviceData concept is similar to the notion of a public instance variable or field in object-oriented programming languages such as Java, Smalltalk, and C++.
- ServiceData is similar to JavaBean™ properties. The JavaBean model defines conventions for method signatures (getXXX/setXXX) to access properties, and helper classes (BeanInfo) to document properties.
The OGSI model uses the serviceData elements and XML schema types to achieve a similar result.

The OGSI specification has chosen not to require getXXX and setXXX WSDL operations for each serviceData element, although service implementers may choose to define such safe get and set operations themselves.

**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.
- Initial values for those serviceData elements (marked as “static” serviceData elements) may be specified using the staticServiceDataValues element within portType.
- The values of any serviceData element, whether declared statically in the portType or assigned during the life of the Web service instance, are called serviceData element values, or SDE values.

**Service DataValues**

- 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.
- Each service instance must have a “logical” XML document, with a root element of serviceDataValues that contains the service Data element values.
- A service implementation is free to choose how the SDE values are stored; for example, it may store the SDE values not as XML but as instance variables that are converted into XML or other encodings as necessary.
- The wsdl:binding associated with various operations manipulating service Data elements will indicate the encoding of that data between service requestor and service provider. For example, a binding might indicate that the serviceData element values are encoded as serialized Java objects.
**SDE Aggregation within a portType Interface Hierarchy**

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.

The serviceData set defined by the service’s interface is the set union of the serviceData elements declared in each portType in the complete interface implemented by the service instance

- Because serviceData elements are uniquely identified by QName, the set union semantic implies that a serviceData element can appear only once in the set of serviceData elements.
- For example, if a portType named “pt1” and portType named “pt2” both declare a serviceData named “tns:sd1,” and a port- Type named “pt3” extends both “pt1 and “pt2,” then it has one (not two) serviceData elements named “tns:sd1.”

**Dynamic service Data Elements.**

- Although many service Data elements are most naturally defined in a service’s interface definition, situations can arise in which it is useful to add or move serviceData elements dynamically to or from an instance.
- The grid service portType illustrates the use of dynamic SDEs. This contains a serviceData element named “serviceDataName” that lists the serviceData elements currently defined.
- This property of a service instance may return a superset of the serviceData elements declared in the GWSDL defining the service interface, allowing the requestor to use the subscribe operation if this serviceDataSet changes, and the findServiceData operation to determine the current serviceDataSet value.

**Core Grid Service Properties**

- **Service Description and Service Instance**
  - A grid service description describes how a client interacts with service instances.
  - A grid service description may be simultaneously used by any number of grid service instances
  - A service description is used primarily for two purposes.
First, it can be used by tooling to automatically generate client interface proxies, server skeletons, and so forth.

Second, it can be used for discovery

**Modeling Time in OGSI**

The need arises at various points to represent time that is meaningful to multiple parties in the distributed Grid.

For example, information may be tagged by a producer with timestamps in order to convey that information’s useful lifetime to consumers.

The GMT global time standard is assumed for grid services, allowing operations to refer unambiguously to absolute times.

Grid service hosting environments and clients should utilize the Network Time Protocol (NTP) or equivalent function to synchronize their clocks to the global standard GMT time.

**XML Element Lifetime Declaration Properties**

- One can define three XML attributes that together describe the lifetimes associated with an XML element and its subelements. These attributes may be used in any XML element that allows for extensibility attributes, including the serviceData element.

- The three lifetime declaration properties are:
  1. `ogsi:goodFrom`. Declares the time from which the content of the element is said to be valid. This is typically the time at which the value was created.
  2. `ogsi:goodUntil`. Declares the time until which the content of the element is said to be valid. This property must be greater than or equal to the goodFrom time.
  3. `ogsi:availableUntil`. Declares the time until which this element itself is expected to be available, perhaps with updated values. Prior to this time, a client should be able to obtain an updated copy of this element. After this time, a client may no longer be able to get a copy of this element (while still observing cardinality and mutability constraints on this element). This property must be greater than or equal to the goodFrom time.
2.4 DATA-INTENSIVE GRID SERVICE MODELS

Data-Intensive Grid Service Models

Two Grid application categories are
– computation-intensive and data-intensive

• Data-intensive applications, deal with massive amounts of data, exceed several petabytes
• The grid system 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.
• However, data replication may demand periodic consistency checks. 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.**

• For the static method, the locations and number of replicas are determined in advance and will not be modified.
• Although replication operations require little overhead, static strategies cannot adapt to changes in demand, bandwidth, and storage availability.
• Dynamic strategies can adjust locations and number of data replicas according to changes in conditions
• The replication strategy must be optimized with respect to the status of data replicas.
• For static replication, optimization is required to determine the location and number of data replicas.
• For dynamic replication, optimization may be determined based on whether the data replica is being created, deleted, or moved.
• The most common replication strategies include preserving locality, minimizing update costs, and maximizing profits.

**2.4.1 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. Similarly, we desire to have unique file names.
• To achieve these, we have to resolve inconsistencies among multiple data objects bearing the same name.
• Access restrictions may be imposed to avoid confusion. Also, data needs to be protected to avoid leakage and damage. Users who want to access data have to be authenticated first and then authorized for access.
2.7 Four Architecture models for building a Data Grid

2.4.1.1 Monadic model:
This is a centralized data repository model, shown in Figure 7.5(a). 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.

2.4.1.2 Hierarchical model:
- The hierarchical model, shown in Figure 7.5(b), 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.
- PKI security services are easier to implement in this hierarchical data access model. The European Data Grid (EDG) to be studied
2.4.1.3 **Federation model:**

This data access model shown in the Figure 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.

2.4.1.4 **Hybrid model:*

This data access model is shown in the Figure. The 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.
- Although the speed of each stream is the same as in sequential streaming, the total time to move data in all streams can be significantly reduced compared to FTP transfer.
- 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.

2.5 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

Handle Resolution

- OGSI defines a two-level naming scheme for grid service instances based on abstract, long-lived grid service handles (GSHs) that can be mapped by HandleMapper services to concrete, but potentially less long lived, grid service references (GSRs).
Virtual Organization Creation and Management

- VOs are a concept that supplies a “context” for operation of the grid that can be used to associate users, their requests, and resources.
- VO contexts permit the grid resource providers to associate appropriate policy and agreements with their resources.
- Users associated with a VO can then exploit those resources consistent with those policies and agreements. VO creation and management functions include mechanisms for associating users/groups with a VO, manipulation of user roles.

Service Groups and Discovery Services

- GSHs and GSRs together realize a two-level naming scheme, with HandleResolver services mapping from handles to references; however, GSHs are not intended to contain semantic information and indeed may be viewed for most purposes as opaque.
- Thus, other entities (both humans and applications) need other means for discovering services with particular properties, whether relating to interface, function, availability, location, policy, or other criteria.

Choreography, Orchestration, and Workflow

- Definition of a job flow, including associated policies.
• Assignment of resources to a grid flow instance
• Scheduling of grid flows (and associated grid services)
• Execution of grid flows (and associated grid services)
• Common context and metadata for grid flows (and associated services)
• Management and monitoring for grid flows (and associated grid services)
• Failure handling for grid flows (and associated grid services); more generally,
• managing the potential transiency of grid services
• Business transaction and coordination services

Transactions
• Transaction services are important in many grid applications, particularly in industries such as financial services and in application domains such as supply chain management.
• However, transaction management in a widely distributed, high-latency, heterogeneous RDBMS environment is more complicated than in a single data center with a single vendor’s software.

Metering Service
• Different grid deployments may integrate different services and resources and feature different underlying economic motivations and models; however, regardless of these differences, it is a qualified universal requirement that resource utilization can be monitored, whether for purposes of cost allocation (i.e., charge back), capacity and trend analysis, dynamic provisioning, grid-service pricing, fraud and intrusion detection, and/or billing.

Rating Service
• A rating interface needs to address two types of behaviors. Once the metered information is available, it has to be translated into financial terms. That is, for each unit of usage, a price has to be associated with it. This step is accomplished by the rating interfaces, which provide operations that take the metered information and a rating package as input and output the usage in terms of chargeable amounts
• Furthermore, when a business service is developed, a rating service is used to aggregate the costs of the components used to deliver the service, so that the service owner can determine the pricing, terms, and conditions under which the service will be offered to subscribers.
Accounting Service

- Once the rated financial information is available, an accounting service can manage subscription users and accounts information, calculate the relevant monthly charges and maintain the invoice information. This service can also generate and present invoices to the user. Account-specific information is also applied at this time.

Billing and Payment Service

- Billing and payment service refers to the financial service that actually carries out the transfer of money; for example, a credit card authorization service.

Installation, Deployment, and Provisioning

- Computer processors, applications, licenses, storage, networks, and instruments are all grid resources that require installation, deployment, and provisioning (other new resource types will be invented and added to this list.) OGSA affords a framework that allows resource provisioning to be done in a uniform, consistent manner.

Distributed Logging

- Distributed logging can be viewed as a typical messaging application in which message producers generate log artifacts, (atomic expressions of diagnostic information) that may or may not be used at a later time by other independent message consumers.
- OGSA-based logging can leverage the notification mechanism available in OGSI as the transport for messages.
- However, it is desirable to move logging-specific functionality to intermediaries, or logging services.
- Furthermore, the secure logging of events is required for the audit trails needed to fulfill judiciary and organizational policy requirements, to reconcile security-related inconsistencies, and to provide for forensic evidence both after the fact and in real time. It is expected that standards and implementations for secure logging should be able to considerably leverage on the efforts associated with distributed logging.

Messaging and Queuing

- OGSA extends the scope of the base OGSI Notification Interface to allow grid services to produce a range of event messages, not just notifications that a serviceData element has changed. Several terms related to this work are:
Figure 2.9 Schematic of a Messaging Service Architecture

Event—Some occurrence within the state of the grid service or its environment that may be of interest to third parties. This could be a state change or it could be environmental, such as a timer event.

- Message—An artifact of an event, containing information about an event that some entity wishes to communicate to other entities.
- Topic—A “logical” communications channel and matching mechanism to which a requestor may subscribe to receive asynchronous messages and publishers may publish messages.

Events

- Events are generally used as asynchronous signaling mechanisms. The most common form is “publish/subscribe,” in which a service “publishes” the events that it exports (makes available to clients). The service may publish the events as reliable or best effort. Clients may then “subscribe” to the event, and when the event is raised, a call-back or message is sent to the client. Once again, the client can usually request either reliable or best effort, though the service may not be able to accept a reliable delivery request. There is also a distinction between the reliability of an event being raised and its delivery to a
client. A service may attempt to deliver every occurrence of an event (reliable posting), but not be able to guarantee delivery (best-effort delivery).

**Policy and Agreements**

- These services create a general framework for creation, administration, and management of policies and agreements for system operation, security, resource allocation, and so on, as well as an infrastructure for “policy aware” services to use the set of defined and managed policies to govern their operation.
- These services do not actually enforce policies but permit policies to be managed and delivered to resource managers that can interpret and operate on them.
- *The Policy Service Manager* controls access to the policy repository.
- *The Policy Service Agent* is a management service that other “policy aware” services depend on for delivery of their policies.
- The agent can provide additional services like understanding time-period conditions so it can inform policy consumers of when policies become active or inactive.
  - These enforcement points will need to interpret the policies and make the necessary configuration changes in the resource(s) they manage by using the Common Management
Other Data Services
- Data access and movement
- Data replication and caching
- Data and schema mediation
- Metadata management and looking

Discovery Services
- In an OGSA environment, it is normally recommended that entities of whatever type be named by GSHs; thus, discovery services are concerned with mapping from user-specified criteria to appropriate GSHs.

Job Agreement Service.
- Manageability interface
  - Supported job terms: defines a set of service data used to publish the job terms supported by this job service, including the job definition (command line and
application name), resource requirements, execution environment, data staging, job control, scheduler directives, and accounting and notification terms.

- Workload status: total number of jobs, statuses such as number of jobs running or pending and suspended jobs.

- **Job control**: control the job after it has been instantiated. This would include the ability to suspend/resume, checkpoint, and kill the job.

**Reservation Agreement Service**

- The reservation agreement service is created by the agreement factory service with a set of terms including time duration, resource requirement specification, and authorized user/project agreement terms.
- The reservation agreement service allows end users or a job agreement service to reserve resources under the control of a resource manager to guarantee their availability to run a job.

**Data Access Agreement Service**

- The data access agreement service is created by the agreement factory service with a set of terms, including (but not restricted to) source and destination file path, bandwidth requirements, and fault-tolerance terms (such as retrial times).
- The data access agreement service allows end users or a job agreement service to stage application or required data.

**Queuing Service**

- The queuing service provides scheduling capability for jobs. Given a set of policies defined at the VO level, a queuing service will map jobs to resource managers based on the defined policies.
- For example, a queuing service might implement a fair-share policy that makes sure that all users within the VO get reasonable turnaround time on their jobs, rather than being starved out by other users’ jobs ahead of them in the queue.
- QoS terms for the queuing service can include whether the service supports on-line or batch capabilities, average turn-around time for jobs, throughput guarantees, the ability to meet deadlines, and the ability to meet certain economic constraints.
- The following terms apply to the queuing service:
  - Enqueue—add a job to a queue
- Dequeue — remove a job from a queue