Open Grid Services Architecture (OGSA)

- *de facto* standard for building the next generation of service-oriented Grid systems
- OGSA extends Web services by introducing interfaces and conventions in three main areas
  - Grid services are dynamic and potentially transient
    - need interfaces to manage their creation, destruction, and life cycle management
  - Grid services can have state that is attributes and data associated with them
  - Clients can subscribe listing their interests in services
    - Need call-back operation from services to clients to notify them of changes

OGSA Core & Platform Services

- Core services:
  - service creation, destruction, life cycle management, service registration, discovery and notification
  - OGSA has service interfaces to support these services
    - GridService, Factory, Registration, HandleResolver, and Notification
- Platform services:
  - user authentication and authorization, fault tolerance, job submission, monitoring, and data access
  - OGSA introduces the concepts of service instance and service data associated with each Grid service to support transient and stateful Grid services
OGSA, OGSI and WSRF

- OGSA defines what interfaces are needed, but does not specify implementation details
- OGSI (Open Grid Services Infrastructure) was designed to specify exactly what needed to be implemented to conform to OGSA
- IBM realised that OGSI was too far from standard Web services to be acceptable to the Web services community
- This led to Web Services Resource Framework (WSRF) which effectively declared that OGSI was obsolescent
- WSRF represents a refactoring and evolution of OGSI that delivers essentially the same capabilities in a manner that is more in alignment with the Web services community

OGSI v WSRF

- WSRF does not break WSDL
  - can expect better tooling support for WSRF
- WSRF is less object oriented
- WSRF is more mix and match
  - in OGSI, one talked to service instances about their service data
  - in WSRF, one talks to the service about its resources and their properties
- Many OGSI ideas and patterns go through in WSRF, if you substitute “resource” for “service instance”
- A advantage to WSRF is that it permits multiple service interfaces to the same stateful resource
- Globus Toolkit 3.0 implemented OGSI
- Globus Toolkit 4.0 implements WSRF

OGSA service instance semantics

- Grid services can be transient
- OGSA provides a soft-service management by introducing the concept of Grid service instances
  - A Grid service instance is an instantiation of a Grid service that can be dynamically created and explicitly destroyed
- A persistent Grid service that can create a service instance is called a service factory
- A client can request a factory to create many service instances
- Multiple clients can access the same service instance.

User Job Submission

[Diagram showing user job submission process]
Grid Service Handle (GSH)

- A globally unique URI identifies a specific Grid service instance
- Grid services may be upgraded during their lifetime, for example to support a new protocol version or to add alternative protocols
  - GSH carries no protocol- or instance-specific information
- This information (and other instance specific info) is encapsulated into the Grid Service Reference (GSR)
- GSR(s) for a Grid service instance can change over that service’s lifetime, and has explicit expiration time
- GSR format depends on communication binding
  - For SOAP binding, GSR will be annotated WSDL document

Service Data Semantics

- Service data: used to describe information about a service instance and runtime state
- Collection of XML elements encapsulated as Service Data Elements (SDE)
- Client can use the standard FindServiceData() method defined in the GridService portType for querying and retrieving service data associated with a Grid service registered in a registry, e.g.,
  - the service type;
  - if it is a service instance, the GSH of the service instance;
  - the location of a service factory; and the runtime states

Hierarchical View of Factory, Service Instances and Data

OGSA portTypes

- These are extended WSDL portTypes
- GridService portType
  - A Grid service must implement this
  - Analogous to the base Object class
  - Three methods encapsulated
    - FindServiceData()
    - SetTerminationTime()
    - Destroy()
- Factory portType
  - persistent Grid service that can be used to create transient Grid service instances with createService() method
OGSA portTypes (ctd.)

• HandleResolver portType
  - can be used to resolve a GSH to a GSR using its FindbyHandle() method.

• Registration portType
  - supports service discovery by maintaining collections of GSHs and their associated policies.
  - Clients query a registry about service availability, properties, and policies.
  - Two elements define a registry service
    • the registration interface: allows a service instance to register a GSH with the registry service,
    • set of associated service data: contains registered GSH and the runtime states of the service instance.
  - RegisterService() and UnRegisterService() are the methods for service registration and un-registration.

OGSA portTypes (ctd.)

• NotificationSource/NotificationSink portType
  - interested parties can subscribe to service data elements and receive notification events when their values are modified
  - To subscribe for notification to a particular Grid service, a notification sink invokes notification source using the SubscribeToNotificationTopic() method in the NotificationSource interface, giving it the service GSH of the notification sink and the topics interested
  - A notification source will use the DeliverNotification() method in the NotificationSink interface to send a stream of notification messages to the sink
  - the sink sends periodic messages to notify the source that it is still interested in receiving notifications.
  - To ensure reliable delivery, a user can define an appropriate protocol binding for the service.

Grid Service Structure in OGSA