Data Transport and OGSA

William (Bill) E. Allcock
Argonne National Laboratory
A Story of Evolution

- The concept of Grid Computing has been around since the 1960s
- Definition of Grid problem has been stable since original Globus Project proposal in 1995
  - Though we’ve gotten better at articulating it
- But the approach to its solution has evolved:
  - From APIs and custom protocols...
  - to standard protocols...
  - to Grid services (OGSA)
- Driven by experience implementing and deploying the Globus Toolkit, and building real applications with it
But Along The Way...

- Heterogeneous protocol base was hurting us
- Increasing number of virtual services that needed to be managed
- Moore’s Law had time to work
- Network speeds increasing faster than Moore’s Law
- Web services (WSDL, SOAP) appeared
Web Services

- At the heart of Web services is:
  - WSDL: Language for defining abstract service interfaces
  - SOAP (and friends): Binding from WSDL to bytes on the wire

- Web services appears to offer a fighting chance at ubiquity (unlike CORBA)

- But Web services does not go far enough to serve a common base for the Grid...
Transient Service Instances

- “Web services” address discovery & invocation of 
  persistent services
  - Interface to persistent state of entire enterprise
- In Grids, must also support transient service instances, 
  created/destroyed dynamically
  - Interfaces to the states of distributed activities
  - E.g. workflow, video conf., dist. data analysis, subscription
- Significant implications for how services are managed, 
  named, discovered, and used
  - In fact, much of Grid is concerned with the management of 
    service instances
Standard Interfaces & Behaviors:

- **Naming and bindings**
  - Every service instance has a **unique name**, from which can discover **supported bindings**

- **Lifecycle**
  - Service instances created by **factories**
  - Destroyed **explicitly** or via **soft state**

- **Information model**
  - **Service data** associated with Grid service instances, operations for accessing this info
  - Basis for service introspection, monitoring, discovery

- **Notification**
  - Interfaces for **registering existence**, and **delivering notifications** of changes to service data
OGSI Grid Service Specification

- Defines WSDL conventions and GSDL extensions
  - For describing and structuring services
  - Working with W3C WSDL working group to drive GSDL extensions into WSDL
- Defines fundamental interfaces (using WSDL) and behaviors that define a Grid Service
  - A unifying framework for interoperability & establishment of total system properties
GT2 Evolution To GT3

- What happened to the GT2 key protocols?
  - Security: Adapting X.509 proxy certs to integrate with emerging WS standards
  - GRIP/LDAP: Abstractions integrated into OGSI as serviceData
  - GRAM: ManagedJobFactory and related service definitions
  - GridFTP: Unchanged in 3.0, but will evolve into OGSI-compliant service in 2004

- Also rendering collective services in terms of OGSI: RFT, RLS, etc.
GT3 Core: OGSJ Specification

The Specification Defines how Entities can Create, Discover and Interact with a Grid Service

**Required:**
- Introspection (service data)
- Explicit destruction
- Soft-state lifetime

- Other interfaces ...

**Optional:**
- Service creation
- Notification
- Registration
- Service Groups

**+ application-specific interfaces**

**Service Implementation**

**Service locator**

Includes 0 or more Grid Service Handles (GSHs)
Includes 0 or more Grid Service References (GSRs)
GT3 Core: OGSII Implementation

- GT3 includes a set of primitives that implement the interfaces and behaviors defined in the latest version of the OGSII Specification.
- The implementation supports a declarative programming model in which GT3 users can compose OGSII-Compliant grid services by plugging the desired primitives into their implementation.
GT3 Core: OGSI Specification (cont.)

GridService portType

- Defines the fundamental behavior of a Grid Service
  - Introspection
  - Discovery
  - Soft State Lifetime Management
- Mandated by the Spec
GT3 Core: 
OGSI Specification (cont.)

Factory portType
- Factories create services
- Factories are typically persistent services
- Factory is an optional OGSI interface

(Grid Services can also be instantiated by other mechanisms)
GT3 Core:
OGSI Specification (cont.)

Notification portTypes

- A subscription for notification causes the creation of a NotificationSubscription service
- NotificationSinks are not required to implement the GridService portType
- Notifications can be set on Service Data Elements
- Notification portTypes are optional
Service group portTypes

- A ServiceGroup is a grid service that maintains information about a group of other grid services
- The classic registry model can be implemented with the ServiceGroup portTypes
- A grid service can belong to more than one ServiceGroup
- Members of a ServiceGroup can be heterogenous or homogenous
- Each entry in a service group can be represented as its own service
- Service group portTypes are optional OGSI interfaces
GT3 Core: 
OGSI Specification (cont.)

HandleResolver portType

- Defines a means for resolving a GSH (Grid Service Handle) to a GSR (Grid Service Reference)
  - A GSH points to a Grid Service
    (GT3 uses a hostname-based GSH scheme)
  - A GSR specifies how to communicate with the Grid Service
    (GT3 currently supports SOAP over HTTP, so GSRs are in WSDL format)

- HandleResolver is an optional OGSI interface
1. A Grid Service Container is started up; it contains an RFT Factory service; The RFT Factory service registers itself.

* The scenarios in this presentation are offered as examples and are not prescriptive.
2. From a known registry, the client discovers a factory by querying the Service data of the registry.

* The scenarios in this presentation are offered as examples and are not prescriptive.
3. The client calls the createService operation on the factory and passes in a TransferRequest.

* The scenarios in this presentation are offered as examples and are not prescriptive.
RFT in Action

Grid Service Container

RFT Factory

RFT Service Instance
- Start the Instance
- Deserialize XML to Java
- Write Request via JDBC
- Persist Service State

Client

4. The instance is started, and the factory returns a locator

* The scenarios in this presentation are offered as examples and are not prescriptive
RFT in Action

5. Client calls Start(), subscribes to notifications, etc.

* The scenarios in this presentation are offered as examples and are not prescriptive
RFT in Action

- Service is OGSI compliant
- Uses existing GridFTP (non-OGSI) protocols and tools to execute 3rd Party Transfer for the user
- Provides extensive state transition notification

* The scenarios in this presentation are offered as examples and are not prescriptive
A Notification Scenario

1. NotificationSink calls the subscribe operation on NotificationSource
A Notification Scenario

1. NotificationSink calls the subscribe operation on NotificationSource

2. NotificationSource creates a subscription service
A Notification Scenario

1. NotificationSink calls the subscribe operation on NotificationSource

2. Notification Source creates a subscription service

3. Notification Source returns a locator to the subscription service
A Notification Scenario

1. NotificationSink calls the subscribe operation on NotificationSource

2. NotificationSource creates a subscription service

3. NotificationSource returns a locator to the subscription service

4.a deliverNotification stream continues for the lifetime of NotificationSubscription

4.b The NotificationSink and Subscription service interact to perform lifetime management
A Notification Scenario

1. NotificationSink calls the subscribe operation on NotificationSource

2. NotificationSource creates a subscription service

3. NotificationSource returns a locator to the subscription service

4.a deliverNotification stream continues for the lifetime of NotificationSubscription

4.b The NotificationSink and Subscription service interact to perform lifetime management

The sole mandated cardinality: 1 to 1
Data Services in OGSA

Note: This is still evolving and will likely change. Tracking the GGF DAIS Working Group is the best way to stay current.
Background

- The current GGF DAIS (Data Access and Integration Services) specification focuses on data access to databases
  - DAIS Goal: It must be possible to support existing unmodified data systems using the proposed interfaces through additional code
- The OGSA Data Services proposal (August 2003) has been produced in order to:
  - Incorporate DAIS requirements and general approach
  - Supports a broad, flexible, and extensible definition of "data service", beyond just the relational and XML database access interfaces that are being considered by DAIS (e.g. file systems, streams, devices, programs)
  - Incorporate WS-Agreement and Quality of Service concepts
  - Incorporate management interfaces as well as access interfaces
  - Exploit OGSI v1.0 (e.g. use service lifetimes to model client sessions rather than separate mechanisms)
Data Service Definitions [1]

- **Data virtualization**: An abstract view of some data, as defined by operations plus attributes (which define the data’s structure in terms of the abstraction) implemented by a data service.  
  **Examples**: A file system, JPEG file, relational database, column of a relational table, random number generator

- **Data interface (base)**: DataDescription, DataAccess, DataFactory, and DataManagement define mechanisms for inspecting, accessing, creating, and managing data virtualizations, respectively. They are expected to be extended to provide virtualization-specific interfaces.
  - An interface is a WSDL portType comprised of a set of operations

- **Data service**: An OGSI-compliant Web service that implements one or more of the four base data interfaces, either directly, or via an interface that extends one or more base data interfaces, and thus provides functionality for inspecting and manipulating a data virtualization.
Data Service Definitions [2]

- **Data set**: An encoding of data in a syntax suitable for externalization outside of a data service, for example for communication to/from a data service. **Examples**: WebRowSet, XML, JPEG encoded byte array, ZIP encoded files.

- **Data source**: A necessarily vague term that denotes the component(s) with which a data service’s implementation interacts to implement operations on a data virtualization. **Examples**: A file, file system, directory, catalog, relational database, a sensor, a program.

- **Resource manager**: The logic that brokers requests to underlying data source(s), via a data virtualization, through the data interfaces of a data service. **Examples**: An extension to, or wrapper around, a relational DBMS or file system; a specialized data service.

- **DAIS-WG**: GGF Working group that is producing the Data Access and Integration specification.

- **DAIS**: Data Access and Integration Services specification.
Data Service Overview

Resource manager:
- implements the data virtualization
- manages access to data sources

Grid service handle

Data service interfaces
- GridService
- DataDescription
- DataAccess
- DataFactory
- DataManagement

Perhaps other interfaces

Underlying data sources
DataDescription: defines OGSI service data elements that describe the data virtualization supported by a particular data service
- E.g. RelationalDescription, RowSetDescription, FileSystemDescription, FileDescription, JPEGDescription

DataAccess: provides operations to access and modify the contents of a data service’s data virtualization
- E.g. SQLAccess, CursorRowSetAccess, StreamAccess, FileAccess, BlockAccess, TransferSourceAccess, TransferSinkAccess
Base Data Service Interfaces [2]

- **DataFactory**: supports a request to create a new data service whose data virtualization is derived from the data virtualization of the parent data service (the one that implements the DataFactory)
  - E.g. FileSelectionFactory, SQLFactory, TransferFactory, CollectionSelectionFactory
  - Some parallel the DataAccess specializations

- **DataManagement**: provides operations to manage the data virtualizations (and indirectly the data sources that underlie them) of a data service
A data service implements 1+ data interfaces; perhaps also other OGSA interfaces.

Base data interfaces:
- Data Description
- Data Access
- Data Management
- Data Factory

OGSI Agreement interfaces:
- Agreement
- AgreementProvider

OGSI interfaces:
- GridService

Data interfaces are typically extended to data-virtualization-specific forms, e.g., RelationalDescription & SQLAccess.
Data Virtualization and Data Sources

- Flexible mappings between data virtualizations and underlying data sources and services. Examples:
  - **one-to-one**: A Data Service corresponds to a DB2 system instance that supports SQL.
  - **one-to-many**: A Data Service corresponds to a federated view of two or more underlying databases.
  - **many-to-one**: A Data Service offering XPath access to an XML File and SQL access to the same file though DB2 Data Federation.
  - **many-to-many**: Different views, each represented as a Data Service, of the one-to-many federation.
Data Virtualization and Naming

- Each Data Virtualization (as a Grid Service) is represented by a GSH (Grid Service Handle)
- Each constituent data source has its own local namespace that describes the visualization
  - Operations against a Data Service may use names (e.g., table names, file names) that can only be interpreted within the context of the service, in particular the data virtualization, to which the operation is directed.
  - If a global name is needed, you should use DataFactory to create a new virtualization (and thus GSH) that is appropriately scoped for your needs
- Data Virtualization implementation is responsible for directing requests to appropriate data sources.
  - Implementation = Resource Manager
Data Virtualization and Service Lifetimes

● **Data services can endure for either:**
  
  ◆ The lifetime of the Resource Manager
    ● Example: To hold the data underlying the virtualization for the duration of the data service, independent of any particular clients. The associated *DataFactory* request may have the side effect of starting a resource manager such as a database system instance

  ◆ The lifetime of the relationship between a resource manager and a set of clients (perhaps just one) interested in that data virtualization
    ● Example 1: To create a virtualization containing a view of the parents’ virtualization, to be shared with other clients
    ● Example 2: To enable the processing of an SQL select where the result sequence is returned an item at a time
(OGSI-) WS-Agreement

- Recall key criteria of a Grid:
  - Coordinates resources that are not subject to centralized control ...
  - using standard, open, general-purpose protocols and interfaces ...
  - to deliver non-trivial qualities of service.
- Implies need to express and negotiate agreements that govern the delivery of services to clients
  - Agreement = what will be done, QoS, billing, compliance monitoring
- All interesting Web/Grid services interactions will be governed by agreements!
WS-Agreement Contents

- **Standard agreement language**
  - A composition of a set of terms that govern a service’s behavior with respect to clients
  - Agreement language uses WS-Policy (currently)
  - Standard attributes for terms that express current state of negotiation
  - Other groups define specific terms
- **Standard agreement negotiation protocol**
  - Establish, monitor, re-negotiate agreement
  - Expressed using OGSI GWSDL interfaces
  - Each agreement represented by a service
WS-Agreement Interfaces

- **AgreementProvider Interface**:
  - extends the OGSI Factory interface
  - defines how the Factory CreateService operation is used with the agreement language to instantiate an agreement with a service provider;

- **Agreement Interface**:
  - extends the OGSI GridService interface:
    - The OGSI GridService interface provides operations for managing the lifetime of a service (and thus the agreement)
    - implemented by the service created by an AgreementProvider
  - provides operations for the monitoring and re-negotiation of the terms of the agreement.
Steps (Operations):
[1] Create Agreement
[2] Create Data Service

Agreement Overview

Agreement

AgreementProvider
(extends OGSI factory)

Agreement I/F
(extends GridService I/F)

DataFactory
(extends AgreementProvider)

DataAccess
(extends Agreement I/F)

Policy

Agreement Initiator

Data Service Consumer
Agreement and Service Lifetimes

- **Agreement Life Time**
  - The agreement selection is made at data service create time.
  - The selected agreement can be redefined at any time within the scope of the selected agreement.

- **Some Data Services** (e.g. those associated closely with a Resource Manager) may have general agreements that apply to all clients, e.g.,
  - All data returned will be at most 5 minutes old

- **Some Data Services** may have individual agreements by client. They may be derived from some pre-defined base agreements, e.g.,
  - Platinum: 1 sec response time max
  - Gold: 5 sec response time max
  - Silver: 20 sec response time max
OGSI Compliant Transport Today

- Via the Reliable File Transfer Service
- Accepts a TransferRequest
  - SOAP Message
  - Defines Default transfer parameters such as TCP Buffer Size, parallelism, etc.
  - List of Source/Destination URL pairs
  - Defaults can be over-ridden per pair, if desired
  - URLs can be a directory and it will move the entire contents of the directory
- Service is OGSI compliant, executes a standard (non-OGSI compliant) 3rd Party GridFTP transfer
OGSI Transport Tomorrow

- This is evolving and could change
- EVERYTHING will have a service interface.
- Transport will be negotiable
- Ideally, there will be autonegotiation based on proximity
  - Same process space: Shared memory
  - Same host: IPC
  - WAN: GridFTP
OGSI Transport

- Interface to transport services such as RFT will be via OGSI-Agreement
- ALL resources are represented as services. This includes files, file systems, databases, etc.
- Under the covers, likely that DataServices will have put() / get() interfaces on their DataAccess Interfaces.