High Performance Computing
Course Notes 2008-2009

Grid Computing II
Standardize grid services

- A typical grid application requires:
  - Resource discovery and management service
  - Job management service
  - Security services
  - Data management services
  - VO management service

- Standardization: define a common interface for each type of service

- Open Grid Services Architecture (OGSA) aims to standardize the services in a grid application
  - What specific services a grid service should have
  - Specifying standard interfaces for these services
Assume that the services have been standardised, then how to represent these services?

- In theory, any distributed technologies can be used
  - RPC, RMI, CORBA, web services

- Web services were chosen

- However, standard web services are not stateful
  - Cannot keep state from one invocation to another
Feature of Web Service Invocation: Stateless

No state information is kept!
Web Service Resource Framework (WSRF)
- Specify how to make web services stateful

Relation between OGSA and WSRF
- OGSA defines the services (set and interface);
- WSRF uses web services to represent these services and makes these services stateful
Implementation of Grid services – GT4 (Globus Toolkit 4)

→ A software toolkit, developed by The Globus Alliance, is used to program grid-based applications

→ GT4 implements WSRF

→ GT4 implements a number of high-level services defined in OGSA, such as
  - a resource monitoring and discovery service
  - a job submission infrastructure
  - a security infrastructure
  - data management services
Relationship between OGSA, GT4, WSRF, and Web Services

- OGSA
- WSRF
- Globus Toolkit 4
- High-level services adequate for Grid applications
- Other software packages (WSRF.NET, ...)
- Stateful Web Services
- Web Services
Grid applications are based on the high-level services defined by OGSA (i.e. not implemented from scratch using WSRF).

OGSA

WSRF

Web Services

Applications

Layered diagram of OGSA, WSRF, and Web Services

Standards in the works (GGF)
- VO management
- Security
- Resource management
- Job Management
- Data services
- etc.

GT4 includes many of the services required by OGSA

Standardized (Oasis) and implemented (GT4)

Standardized (W3C) and implemented (e.g. Apache Axis)
How to make web services Stateful

In a nutshell, combine a web service with a resource

- Resource must be identifiable
- Have properties, expressed in XML
- Have lifecycle controlled by resource itself
A pairing of a web service with a resource is called a WS-Resource.
Web Services Invocation

1. Where can I find a “weather service”?
2. There’s a “weather service” in Server B
3. How exactly should I invoke you?
4. Take a look at this: WSDL
   - SOAP request: Invoke getWeatherInfo() with parameter ‘12345’
5. SOAP response: “Cloudy with a chance of meatballs”
Web Services Architecture

- **Processes**: Discovery, aggregation, choreography, etc.
- **Description**: WSDL, Web Services Description Language
- **Invocation**: The most popular invocation protocol is SOAP, but we could, in theory, use other protocols.
- **Transport**: The most popular transport protocol is HTTP, but we could, in theory, use other protocols.

Computer Science, University of Warwick
shopping cart example
WSDL description of non-WSRF web services

WSSimpleShoppingService

PortType: WSSimpleShoppingCartCreation
  Operation: WSCreateCart
    input
    output

PortType: WSSimpleShoppingCart
  Operation: WSGetCart
    input
    output
    WSCartUnknownFault
  Operation: WSPutCart
    input
    output
    WSCartUnknownFault
  Operation: CartCheckout
    input
    output
    WSCartUnknownFault
    WSBillingFault
Creating cart in non-WSRF web services

SOAP Request for creating a cart

```xml
<SOAP-ENV:Header>
  ...
  <wsa:To SOAP-ENV:mustUnderstand="1">
    http://www.example.com/WSSimpleShoppingService
  </wsa:To>
</SOAP-ENV:Header>

<SOAP-ENV:Body>
  <ws-ssc:WSCartCreateRequest>
    <ws-ssc:Quantity>1</ws-ssc:Quantity>
  </ws-ssc:WSCartCreateRequest>
</SOAP-ENV:Body>
```
Creating cart in non-WSRF web services

→ SOAP body of the reply message

```xml
<SOAP-ENV:Body>
  <ws-ssc:WSCartCreateResponse>
    <ws-ssc:Cart>S1</ws-ssc:Cart>
    ...
  </ws-ssc:WSCartCreateResponse>
</SOAP-ENV:Body>
```

→ The reply message contains the cart identifier S1;

→ S1 will be used as an input parameter of any call to the service
Call `getCart` operation in non-WSRF web service

1. S1 is used as input parameter in the call to the service

```xml
<ws-ssc:WSGetCart>
  <ws-ssc:Cart>S1</ws-ssc:Cart>
</ws-ssc:WSGetCart>
```

2. The response message of the service

```xml
<SOAP-ENV:Body>
  <ws-ssc:WSGetCartResponse>
    <ws-ssc:WSSimpleShoppingCart>
      <ssc:Item>
        <ssc:Quantity>1</ssc:Quantity>
        <ssc:ProductPrice>1.59</ssc:ProductPrice>
      </ssc:Item>
    </ws-ssc:WSSimpleShoppingCart>
  </ws-ssc:WSGetCartResponse>
</SOAP-ENV:Body>
```
WSRF web services

Differences in the definitions between non-WSRF and WSRF web services

- WSRF define a resource properties document (in XML) for a resource
- the resource properties document is associated to the web service (WSDL portType)
- Some operations can be standardised
Resource properties

...  

<!-- Resource property element declarations -->
<xsd:element name="ProductCode" type="xsd:string"/>
<xsd:element name="Quantity" type="xsd:integer"/>
<xsd:element name="Price" type="xsd:float"/>
<!-- Resource properties document declaration -->
<xsd:element name="ShoppingCart">  
<xsd:complexType>
<xsd:sequence>
  <xsd:element ref="tns:Productcode"/>
  <xsd:element ref="tns:Quantity"/>
  <xsd:element ref="tns:Price"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

...
Associate resource property document to services

...<wsdl:portType name="SimpleShoppingCart"
    wsrf-rp:ResourceProperties="ssc:ShoppingCart">
<wsdl:operation
name="GetResourcePropertyDocument">
    <wsdl:input message="wsrf-rpw:
        GetResourcePropertyDocumentRequest"/>
    <wsdl:output message="wsrf-rpw:
        GetResourcePropertyDocumentResponse"/>
...
WSDL definitions

PortType: SimpleShoppingCartCreation
   Operation: CreateCart
     input
     output

PortType: SimpleShoppingCart
   Operation: GetResourcePropertyDocument
     input
     output
     ResourceUnknownFault
     ResourceUnavailableFault

   Operation: PubResourcePropertyDocument
     input
     output
     ResourceUnknownFault
     ResourceUnavailableFault

   Operation: GetResourceProperty
     input
     output
     InvalidResourcePropertyQNameFault
     ResourceUnknownFault
     ResourceUnavailableFault

   Operation: Checkout
     input
     output
     CheckoutFault
     ResourceUnknownFault
     ResourceUnavailableFault

ResourceProperties schema,
root element declaration

ssc:SimpleShoppingCart
   ssc:Item
     ssc:ProductCode
     ssc:Description
     ssc:Quantity
     ssc:ProductPrice
Creating shopping cart in WSRF web services

- SOAP request from client is the same as in the non-WSRF web service

- In the reply message, the cart id is part of the EndPointReference (EPR)

```xml
<ssc:CartCreateResponse>
  <wsa:Address>http://www.example.com/SimpleShoppingService</wsa:Address>
  <wsa:ReferenceParameters>
    <rpimpl:CartId>S1</rpimpl:CartId>
  </wsa:ReferenceParameters>
</ssc:CartCreateResponse>
```
Non-WSRF web service vs. WSRF

- Non-WSRF Web service: The resource identifier is contained in the SOAP message body

- WSRF: the resource identifier is packaged as part of EndPointReference of the web service

```xml
SOAP envelope
Header
<wsa:To>http://www.example.com/WSSimpleShoppingService</wsa:To>

Body
<ws-ssc:WSCheckoutRequest>
  <ws-ssc:Cart>
    S1
  </ws-ssc:Cart>
  ...
</ws-ssc:WSCheckoutRequest>

SOAP envelope
Header
<wsa:To>http://www.example.com/SimpleShoppingService</wsa:To>
<rpimpl:CartId wsa:IsReferenceParameter="true">S1</rpimpl:CartId>

Body
<ssc:CheckoutRequest>...
</ssc:CheckoutRequest>
```
GetResourceProperty

→ SOAP request for getting resource property

<SOAP-ENV:Header>
  ...
  <wsa:To SOAP-ENV:mustUnderstand="1">
    http://www.example.com/SimpleShoppingService
  </wsa:To>
  <rpimpl:CartId>wsa:IsReferenceParameter="true">S1
  </rpimpl:CartId>
</SOAP-ENV:Header>

... 

<SOAP-ENV:Body>
<wsrf-rp:GetResourcePropertyDocument/>
</SOAP-ENV:Body>

The operation of “GetResourcePropertyDocument” can be standardised
WSRF Specification

WSRF is a collection of five different specifications

- WS-Resource
- WS-ResourceProperties
- WS-ResourceLifetime
- WS-ServiceGroup
- WS-BaseFaults
WS-Resource

- Describe the relationship between a web service and a resource
- Define the pattern by which resources are accessed through Web services
- Define the means by which WS-Resources are referenced

Properties of a resource are presented via a resource properties document

WSRF associates an XML description of the resource (resource properties document) with the WSDL portType which defines the web service operations on the resource
WS-ResourceLifetime

- The lifetime of a WS-Resource is defined as the period between its initialization and its destruction.
- Standardizes the means by which a WS-Resource can be destroyed.
- Defines the means by which the lifetime of a WS-Resource can be monitored.
- Destruction of a WS-Resource:
  - Immediate destruction: can be accomplished using the message exchanges defined in this specification.
  - Scheduled destruction: set termination time.
- WSRF provides the resource property wsrf-rl:TerminationTime as a standard mechanism to record the lifetime of a WS-Resource.
- The wsrf-rl:CurrentTime property records the local time. When the TerminationTime is reached, the shopping cart may be removed.
WS-ServiceGroup

- Aggregates info about multiple WS-Resources or Web service
- Can be used to identify the resources or services which satisfy a specified expression
WS-BaseFaults

- Define a set of universal faults
  - ResourceUnknownFault
  - ResourceUnavailableFault

- Define a set of common info in faults
  - Timestamp
  - Originator of the fault
  - ...