State Management Issues and Grid Services

Xie Yong[1] and Y.M Teo[1,2]

Singapore-Massachusetts Institute of Technology Alliance

Department of Computer Science, National University of Singapore

Outline

- Introduction
- Fundamentals
- Analysis
  - Web Service
  - Grid Service
  - Web Service Resource Framework (WSRF)
- Proposed Grid Services Model
- Implementation
- Conclusion and Future Work

Evolution …

CHALLENGE: How components across the world can collaborate with each other to carry out applications over the Internet ???

…… CORBA, Java RMI, COM → None prevailed

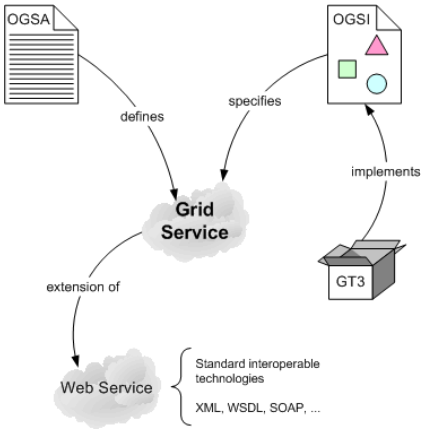
Recently: Web Service, Grid Service, Web Service Resource Framework (WSRF) …… ???

Web Service

- Proposed in year 2000
- Simple Object Access Protocol (SOAP)
- Web Services Description Language (WSDL)
- Universal, Description, Discovery and Integration (UDDI)
- W3C's Definition: *… a software system designed to support interoperable machine-to-machine interaction over a network, and it has an interface described in a machine-processable format (specially WSDL), and other systems interact with the web service in a manner prescribed by its description using SOAP-messages, …*

Grid Service

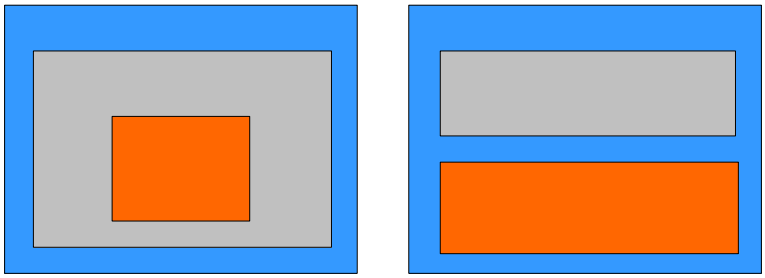
- A group of researchers lead by Ian Foster
- Definition
- Open Grid Service Architecture (OGSA)
- Open Grid Service Infrastructure (OGSI)
- State management using the *Factory-Instance* approach



WS-Resource Framework (WSRF)

- Refactoring and evolution of OGSI
- Separates state and stateless service
- WS-Resource factory creates the resource
- End point reference contains address of the service and the resource id of the resource

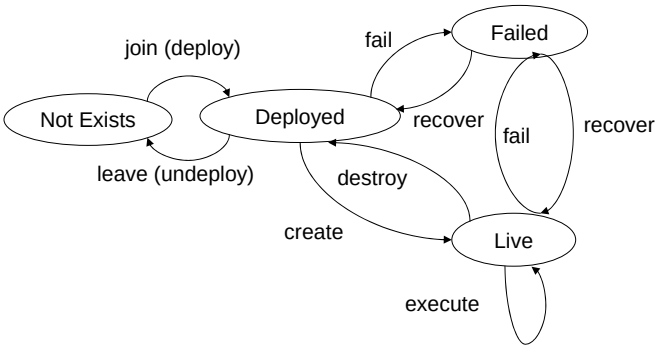
WS-Resource Framework (WSRF)



Source:

http://www-unix.globus.org/toolkit/docs/development/3.9.2/core/index.html

Web/Grid Service Life Cycle



Container

Service

State

- Service's internal data or attributes which are needed to persist across multiple invocations.
- Web service's internal data or attributes which are needed to maintain for multiple clients.
- Web service's internal data or attributes which are needed to deal with the dependencies among multiple services.
- Service's status at system level.

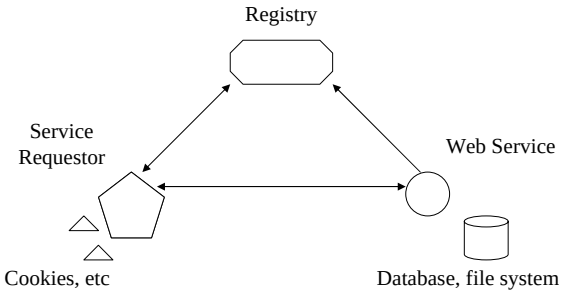
State Management

- State management: how state information are stored, retrieved and maintained
- Requirements:
  - Naming
  - Scalability
  - Communication costs (transferring of states)
  - Heterogeneity (state models and management schemes)
  - Security (state repository and access to state)

Analysis of Web Service

- Stateless and non-transient
  - Cannot remember info. from one invocation to another
  - Services outlive all their clients
- Stateless implementation, stateful interfaces
  - Client Side (cookie/session)
  - Server Side (database/file)
- Application (Industry)
  - Short-lived, simple/no state

Analysis of Web Service



Analysis of Web Service (cont.)

- Join
  - Register with UDDI registry
  - not affect other existing services
- Service Failure
  - No effect on existing services, and state information remains
- Leave
  - No effect on existing services
- Scalability
  - Client overloaded
  - Server overloaded

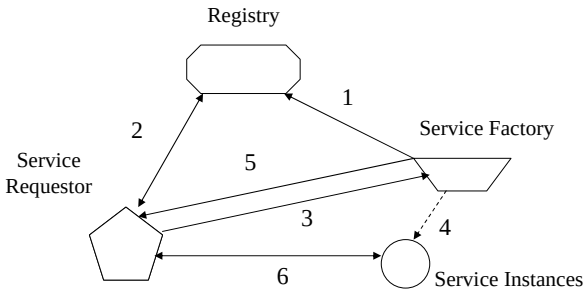
Analysis of Web Service (cont.)

- State Information Security
  - State on Client: Safe
  - State on Server: Depending on service provider's security policy
- Communication Cost
  - Stateless: client <-> service
  - State on Client: parameter passing
  - State on Server: communication with system component such as database (could be remote)
- Naming
  - ???

Analysis of Grid Service

- Long-lived application with complex states
- Factory and instances
  - Client requests factory to create service instance
  - Service instance are transient and stateful
  - State maintained by service instance
- Join (deploy)
  - Register with service container (runtime environment)
  - Restart of Container (GT3)

Analysis of Grid Service



Analysis of Grid Service (cont.)

- Service instance failure
  - State Information is lost with the instance
  - Other Service Instances will not be affected
- Service instance destruct
  - Explicitly by client, or implicitly when lifetime expires
  - States are gone
- Service leave
  - Restart Container (GT3)
  - Other services will be stopped as well
  - No persistent service
- Communication cost
  - Between client and service instance

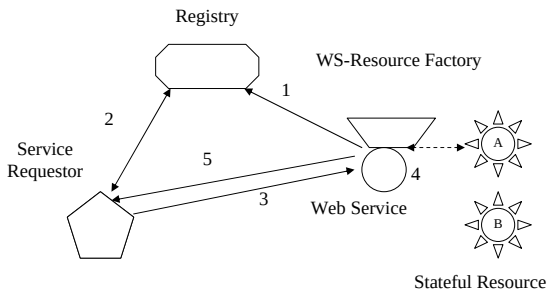
Analysis of Grid Service (cont.)

- State Information Security
  - The encapsulation of states inside a service instance provides certain level of security
- Scalability
  - Service Instance can be remotely created on the grid (OGSI)
  - Not supported in GT3 implementation
- Naming
  - Grid Service Handle (GSH)

Analysis of WSRF (cont.)

- Join
  - Web Service and its Resource Factory
- Service failure
  - the resource is independent of the service
  - No effect on other services
- Leave
  - No effect on other services

Analysis of WSRF (cont.)



Analysis of WSRF (cont.)

- Communication Cost
  - Client <-> service
  - Service <-> resource (depends on implementation)
- State Information Security
  - Tight-coupling between the factory and service
  - Web Service provider's security police must be trusted
- Scalability
  - Only one resource factory for each web service

Limitations of WSRF

- Web-service specific state management
  - Require state management service for newly created stateful Web service
  - More work for Web service developer
- Tight-coupling between service and factory restricts scalability
- Security Issue
  - Web service provider controls the state management and storage of state information
  - Client cannot have the control: what if the security policy at service side is changed? Still trusted?

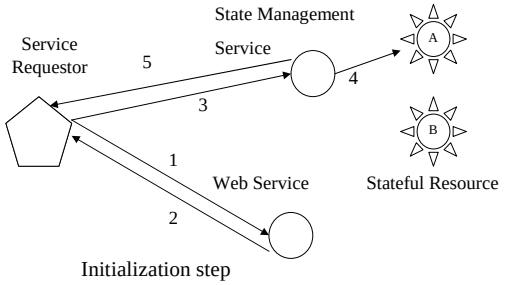
	Web Services	Grid Service OGSI (GT3)	WSRF	Proposed Model
Target Application	Industry, simple state, short-lived	Academic, complex state, long-lived	Industry and academic	Industry and academic
State Repository	Client or Server	Service Instance	WS-Resource	Stateful Resource
Service Join	No effect on others	Restart Container	No effect on others	No effect on others
Service Failure	No loss of State	Loss of State	No loss of State	No loss of State
Service Leave	No effect on others	Restart Container	No effect on others	No effect on others
Communication Cost	Client – service, depending on where state is stored	Client - Service	Client – Factory, Client – Service, Service – WS-Resource	Client – Service (for initialization), Client – State Management Service, Client – Service (for service interaction), Service – State Management Service – Resource
State Security	•Non-transient for stateless service •State on client: secure •State on server: require extra work from dispatcher	Encapsulated in service instance: secure	secure, given access control to resources and server is trusted by requestor; otherwise, not secure as server can access state resource	Client controls where to store state information: secure with access control
Scalability of State Information	Both ways of maintaining state information have restriction on scalability.	Remote deploy service instance: scalable (GT3 does not support)	Tight coupling between factory and service, only allow one copy of factory: not scalable	Loose coupling between state management service and web service, allow multiple copies of state management service: scalable
Naming of states	Not supported	Using GSH	WS end point reference	Many ways: e.g. Combine service id, client id, and application id.

Proposed Framework

- Separation of state management from Web Service
- Generic state management service: Loose coupling of service and state management
  - Independent of web service implementation
  - Freely available, can be deployed anywhere
  - Retrieve & update state information
  - Handle complex state (XML)
  - Initialization & interactions steps

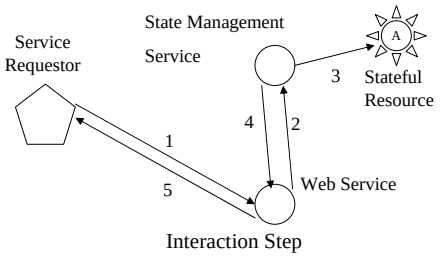
Two Steps of using a Web Service

- Step 1: Initialization Step



Two Steps of using a Web Service

- Step 2: Interaction Step

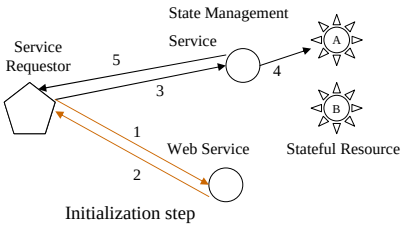


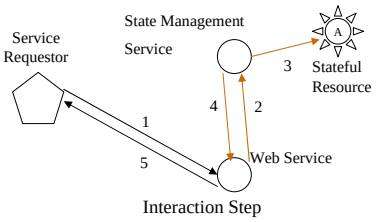
Advantages

- Reduce work of service provider
- Security: give control of state management to service requestor
- Flexibility: can choose where to deploy state management services
- Enhance Scalability: deploy multiple copies of state management service

Overhead

- Increase Communication Cost





	Web Services	Grid Service OGSI (GT3)	WSRF	Proposed Model
Target Application	Industry, simple state, short-lived	Academic, complex state, long-lived	Industry and academic	Industry and academic
State Repository	Client or Server	Service Instance	WS-Resource	Stateful Resource
Service Join	No effect on others	Restart Container	No effect on others	No effect on others
Service Failure	No loss of State	Loss of State	No loss of State	No loss of State
Service Leave	No effect on others	Restart Container	No effect on others	No effect on others
Communication Cost	Client – service, depending on where state is stored	Client - Service	Client – Factory, Client – Service, Service – WS-Resource	Client – Service (for initialization), Client – State Management Service, Client – Service (for service interaction), Service – State Management Service – Resource
State Security	•Non-transient for stateless service •State on client: secure •State on server: require extra work from dispatcher	Encapsulated in service instance: secure	secure, given access control to resources and server is trusted by requestor; otherwise, not secure as server can access state resource	Client controls where to store state information: secure with access control
Scalability of State Information	Both ways of maintaining state information have restriction on scalability.	Remote deploy service instance: scalable (GT3 does not support)	Tight coupling between factory and service, only allow one copy of factory: not scalable	Loose coupling between state management service and web service, allow multiple copies of state management service: scalable
Naming of states	Not supported	Using GSH	WS end point reference	Many ways: e.g. Combine service id, client id, and application id.

Prototype

- illustrate: separating state management from Web service
- Simple service that does mathematic computations: +, −, ×, ÷
  - State: an integer
- Two persistent services (on GT3):
  - CalcService (implements Calc): Providing Calculation functionalities
  - CalcStateService(implements StateManageService): general, state management

Prototype

- Calculation Service

```
public interface Calc {
    public String start();
    public int add(String uri, int b);
    public int subtract(String uri, int b);
    public int multiply(String uri, int b);
    public int divide(String uri, int b);
}
```

- State Management Service

```
public interface StateManageService {
    public String create(Object initial_state, String clientid);
    public void set (String uri, Object stateinfo);
    public String get(String uri);
}
```

URI: hostname + directory + 'state' + clientid +'.xml'

Conclusion

- Summary
  - Clearly defined 'state'
  - Articulated requirement for state management
  - Analyzed existing state management schemes
  - Proposed a new approach
  - Implemented a prototype for proof of concept
- Future Work
  - Reduce communication overhead
  - Access control

Thank You!

Q & A

Proceedings of the 3rd International Conference on Grid and Cooperative Computing, Springer-Verlag Lecture Notes in Computer Science Series, vol. 3251, pp. 17-25, China, October 21-24, 2004.