Monitoring Access to Stateful Resources in Grid Environments

Sangmi Lee Pallickara, Beth Plale, Scott Jensen, Yiming Sun
Department of Computer Science, Indiana University
(leesangm, plale, scjensen, yimsun)@cs.indiana.edu

Abstract

Currently, Grid technologies are widely used in large-scale scientific applications. Grids support stateful interactions with explicit exposure of state information across the boundaries of a service. In this paper, we present a stateful Web service architecture that provides efficient sharing of a service instance between heterogeneous service requesters with monitoring of the interactions. We describe how we compose the state information based on the formalized sequence of the interactions. We also describe how the shared service instance is managed, and interacted with, through standard Web services interfaces. We analyze the performance of our approach in a large-scale scientific Grid application.

Keywords: Web Services, Grid Services, Stateful resources, monitoring, LEAD, WS-I+, WSRF

1. Introduction

Grid technologies facilitate efficient sharing of resources in heterogeneous distributed environments and are widely used in large-scale scientific applications [1], [2], [3]. By employing Web services technology, Grids offer a convenient paradigm for resource sharing through resource virtualization [4]. Web services, by their nature, do not maintain state information for their interactions. However, there are resources which allow only specific sequential set of actions to access their resource objects.
There have been approaches to allow stateful resource manipulation via Web service operations; That is Open Grid Service Interface (OGSI) and it's follow-on Web service Resource Framework (WSRF). These approaches enable access to stateful resources with the maintenance and retrieval of state information. However, for stateful resources shared by multiple service requesters, state management is not accomplished by only providing the state information to the service requesters.

This is true especially for data-centric scientific computing, where experimental data products are often created as a result of a sequenced set of activities with a specific sequence. For example, raw data from a scientific instrument is often manipulated in stages by separate distributed process. To provide more intelligent usage of data product, the application specific metadata information must be tracked and managed along with the dataflow.

As shown in several recent approaches [1], [2], [3], modern data-centric scientific computing provides data discovery and personalization mechanisms with metadata management. To provide an application specific metadata from the correlated data manipulation, significant participation of each service is required. Although the metadata service maintains its own state information, since the service requesters are located across various network domains, it is difficult to guarantee that the requests are delivered in the time sequence expected by the metadata service.

This paper posits that monitoring is an essential piece of maintaining stateful interactions for Grid services. Although the Grid infrastructure enables stateful interactions, monitoring and automated scheduling of service requests are required to maintain the QoS (Quality of Services) of the stateful Web services. In this paper, we propose a stateful Web service architecture which provides the following features.

- Composing the state information document derived from a Finite State Machine,
- Maintaining a stateful service instance shared by multiple service requesters,
- Monitoring and scheduling service requests,
- Leveraging with standard Web service interfaces to interact with the service requesters.

Moreover, we exemplify our approach with myLEAD [6] which is a personal catalog service for a large-scale scientific metadata. MyLEAD provides a virtual working environment to the scientists. This working environment is built by virtually structuring the experimental data based on the metadata that is known about these data objects. MyLEAD facilitates sharing of the data product between individuals and groups of researchers in addition to incorporating strategies for data preservation. This virtual working space is enabled by generating and maintaining the metadata about the data product and application specific information. The MyLEAD is a part of the Linked Environments for Atmospheric Discovery (LEAD) project [7]. The LEAD project addresses the

requirements of the cyber infrastructure to facilitate the identification, access, preparation, assimilation, analysis and visualization of a broad array of mesoscale meteorological data and model output independent of format and physical location. As part of the LEAD project, a group at Indiana University has developed myLEAD, an active, personalized catalog for managing scientific metadata. The myLEAD project includes specialized facilities – for search, content storage, data-object cataloging, and active engagement – through which users can capture new data objects.

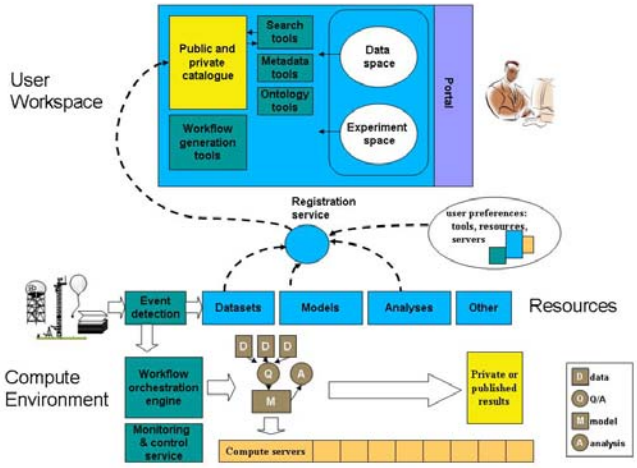

Figure 1: Functional view of LEAD system

As figure 1 illustrates, various services access the myLEAD service along the dataflow. The myLEAD service is accessed by distributed services including instrumental datasets gathering services, scientific modeling services, analysis services, and portal service. These accesses are performed in a specific sequence and essentials construct a collection of metadata which forms a virtual workspace for subsequent use by individuals and groups of scientists.

This paper is organized as follows. We discuss related works and our motivation in Section 2. In Section 3, an overview of the myLEAD architecture is presented. Section 3 illustrates the lifecycle of the myLEAD service; Section 4 describes the scheduling and monitoring of the service requests. We present the basic performance measurements for scheduling in Section 5. The conclusion of the paper is provided in section 6.

2. Related work and Motivations

Several approaches have been taken to enable stateful Web services. OGSI [4] defines a set of conventions and extensions based on the use of Web Service Definition Language (WSDL) [8] and XML schema to enable stateful Web services. It includes creating, naming, and managing the lifetime of service instances. The approach of OGSI defines declaring and inspecting service state data, asynchronous notification of service state changes, representing and managing collections of service instances, and handling of service invocation faults. More recently OGSI has been replaced by the WS-RF [5] which follows the same conceptual model, but without altering the underlying Web services standards and specifications. Both of these approaches provide a model for accessing stateful Web services.

However, the approaches of the OGSI and WS-RF do not satisfy the requirement of every use case. If the stateful resource is required to be shared among heterogeneous service requesters, it is tough to be managed efficiently with only sharing of the state information. First, service designers cannot expect that all of the participating services are aware of the stateful accesses and behave according to the state transitions. When we build a scientific Grid, we start with some number of existing services. Access to the newly shared resource should be easy not only for legacy services but also for newly developed services as well. Second, if the sharing of the stateful resource is performed by heterogeneous service requesters, it is not simple to provide a notification service to each requester. Since the service requesters can be orchestrated dynamically, using notifications to inform about the state change information can cause complexity of interactions between services.

Meanwhile, the UK e-Science center has proposed a strategy, termed WS-I+, which plans for interoperability with, or migration to, Web services and Grid standards. WS-I+ provides stable medium-term access to the core functionalities required by a wide range of current scientific applications. MyLEAD adapts the strategy of the WS-I+ for utilizing Web services standards. Two key Web services standards utilized by myLEAD, WS-Addressing and WS-Eventing, are part of the core set of standards that WS-I+ identifies.

The Web service Composite Application Framework (WS-CAF) [9] proposes a standard suite containing the management of stateful interactions between Web services. WS-CAF correlates the works of services participating within the same activity by propagating additional information, known as context, to those participants using the WS-Context specification. The context contains information such as a unique ID and allows a series of actions to share a common outcome.

The expression of the information is quite similar to the state document of the myLEAD system; however myLEAD considers information about only current state,

including a unique identifier, the name of the current state, and valid actions. The state information in the myLEAD system is derived from the Finite State Machine formalizing stateful access.

3. Overview of the architecture

MyLEAD is a Grid service providing data discovery and personalized information space to individuals and groups of users. The myLEAD architecture consists of three primary functional units: a server-side service, a client-side service, and a user interface [6]. The myLEAD server-side service is a persistent Grid service built on top of a relational database management system (RDBMS). The myLEAD server-side service extends the MCS schema and interface; MCS is built on top of the Open Grid Services Architecture Data Access and Integration (OGSA-DAI) Web service – a set of generic Grid services that provide access to any database management system. OGSA-DAI interact with the database using Java Database Connectivity (JDBC), bringing the DBMS onto the Grid by providing a generic query-and-update Web service interface to it.

The MCS catalog service [10] extends OGSA-DAI by providing methods to access particular tables in the database schema. MyLEAD extends the schema further with support for spatial and temporal attributes and introducing the notion of experiments. These are supported by adding methods for database access and making performance improvements over MCS by means of database stored procedures. Eventually, replicated myLEAD servers will be distributed across the LEAD test bed, achieving reliability through a master-client replication scheme.

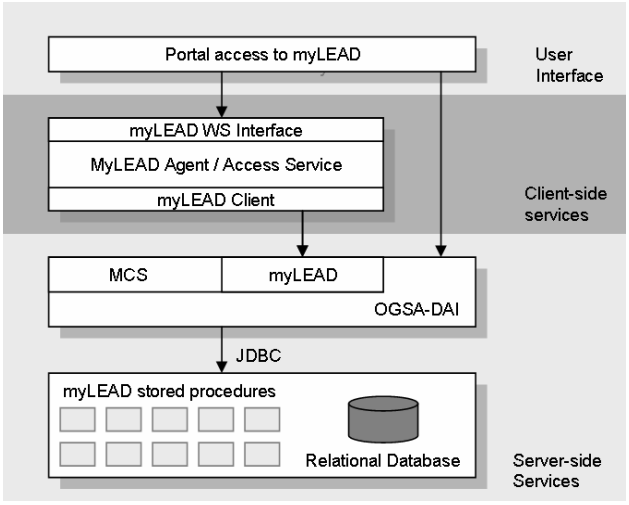

Figure 2: MyLEAD architecture

The next level is the myLEAD access service which includes the agent and persistent access service. The myLEAD agent is a transient, short-lived service that serves a single user for a single session. Meanwhile, the access service is a permanent Web service accessible from other Web services directly. Both types of client-side services manage stateful interactions between the myLEAD Server-side service and service requesters.
At the user-interface level, a client (user) interacts with myLEAD through the LEAD portal service. The tool provides several portlets for managing, browsing and searching personal information spaces.

4. State Management by means of Sessions

MyLEAD provides stateful access to the myLEAD server-side services with its client-side services. The lifecycle of the myLEAD system is defined within the myLEAD system as a *myLEAD session*, and also accommodates other services such as a RDMBS built on top of OGSA—DAI. The myLEAD session is a service instance of myLEAD which contains one or more service requests, and is initiated by the service requester. In this section, we discuss about the aspects considered for designing the lifecycle of the myLEAD session.

4.1 Creation of the myLEAD session

The myLEAD session is created with the service request which specifies types of session and user identifiers. After the myLEAD system is requested to create a session, it generates a session identifier and register a new session with the resource factory. The uniqueness of a session identifier is ensured within the myLEAD service domain. The service requester will receive the session identifier which is associated with a newly created session. The service requester uses the session identifier to access the particular session.

4.2 Access to the myLEAD session

The myLEAD session is shared from one or multiple service requesters. For example, in the myLEAD scenario, a single session is accessed by a portal user interface, a workflow engine, and by workflow components to manipulate the dataflow of scientific analysis. The portal service accesses myLEAD to build a higher level structure for their experiment such as project or experiment which eventually comprises data output or input datasets. For the same experiment, the workflow

engine will provide notification messages to the myLEAD service to inform about the status of the workflow. Similarly, the workflow components access myLEAD service to move the data to the storage. Therefore, the myLEAD session should manage concurrent accesses from heterogeneous service requesters.

The session identifier enables access to a session without confusion between heterogeneous service requesters. When a service requester triggers other service requesters, it passes the myLEAD session identifier to share the myLEAD session.

Figure 2 illustrates the access from the service requesters to the myLEAD metadata catalog through the myLEAD session.

Step 1. Service request through the Web Service interface with the given session identifier
Step 2. Access the resource factory containing the index of sessions.
Step 3. Access the session instance with the session identifier
Step 4. Push the service request into the request queue
Step 5. Reschedule the request queue with the FSM engine
Step 6. Process the request
Step 7. Access the metadata catalog
Step 8. Update the state document in the resource factory
Step 9. Return the value of the service request

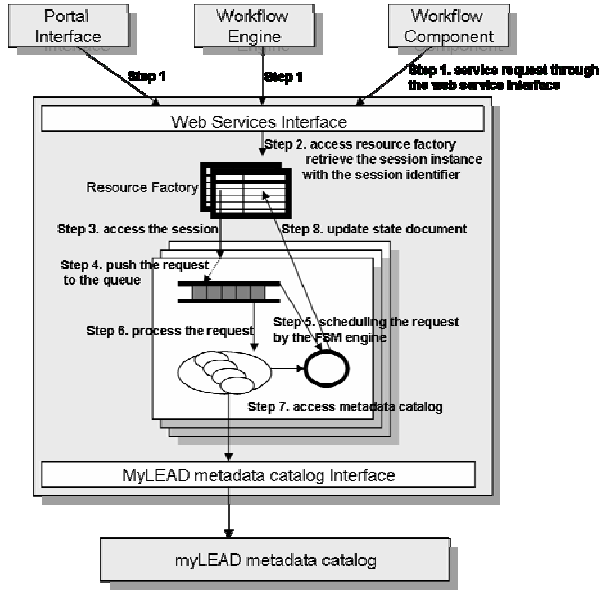

Figure 3: Access to the myLEAD session from the heterogeneous service requesters

Meanwhile, the concurrent access can cause problems with the manipulation of the distributed data product. Since the services are very loosely coupled and spread over multiple organizations, there is no guarantee of order among the service requests. The communication delay between the portal server and the myLEAD service can cause earlier delivery of notification messages from the workflow engine, which can cause an exception at the metadata catalog because the data structure which is specified in the notification is not generated yet. Finally the users can not retrieve the status of their workflow even though the workflow processed and informed the status properly. Therefore, myLEAD provides event scheduling based on Finite State Machine (FSM) within a session. We discuss about this in the section 5.

4.3 Lifetime Management

The lifetime of a myLEAD session is defined as the period between its creation and its expiration. The lifetime can be specified by a service requester when it is created or modified during the session. The lifetime of the myLEAD session is defined in the state document with three elements; time of creation, expiration, and last modification. The service requester can keep track of the lifetime utilization. We discuss about the state document in the section 5.1.

The metadata catalog of myLEAD extends MCS which is built on the OGSA – DAI service. Therefore, myLEAD should accommodate accesses to the OGSA-DAI based data resources.

The myLEAD session creates and maintains an OGSA-DAI service instance. As soon as the myLEAD session is expired or destroyed, the OGSA-DAI service instance is also destroyed.

Unless the myLEAD session is expired or destroyed externally, myLEAD maintains the session with its status. This independence provides a fundamental fault tolerant access to the myLEAD session. As far as the service requester provides the session identifier, the service requester is allowed to access the running session.

5. Role of Monitoring in State Management

A myLEAD session can be shared by heterogeneous service requesters. To solve the concurrency problem, myLEAD provides monitoring and scheduling of the requests in the myLEAD session. Each session maintains a request queue. When a new request is delivered to the session, the session validates that the request is in the valid order and reschedules the requests in the queue.

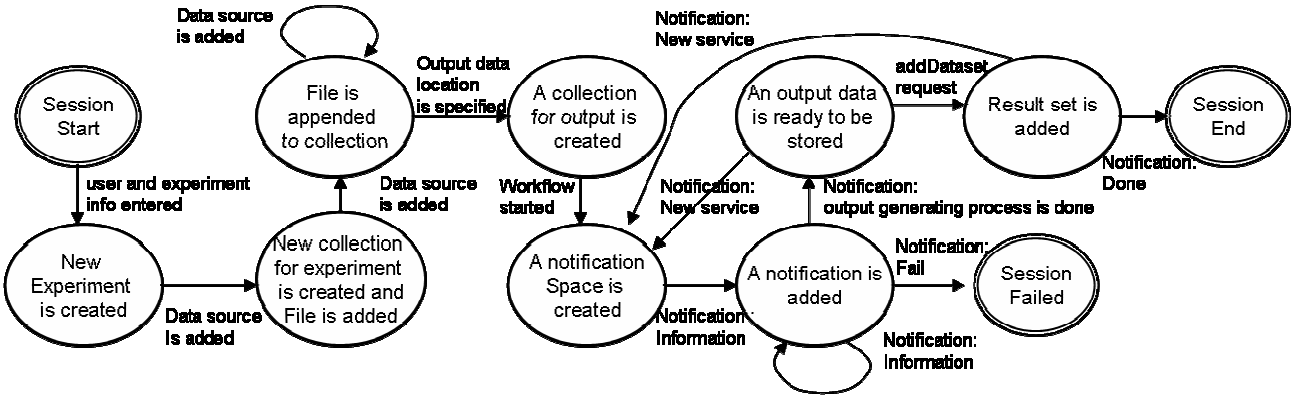

Figure 4: A State Transition Diagram for creating a data product

5.1 Defining the Finite State Machine

To validate and schedule the requests, first, we define the state of the myLEAD session. The states in the myLEAD system are formalized with the Finite State Machine (FSM). Figure 4 illustrates a simplified State Transition Diagram (STD) of the creation of a data collection. In the context of the myLEAD service, the data source is the input data set for a data analysis process. The LEAD service is composed of several workflow components spanning multiple organizations. For each component, the workflow engine provides notification to the myLEAD session as a service request. Through the notification informing the completion of an output generating process, the myLead session updates its state to "An output data is ready to be stored". Output data is moved to the storage service with its metadata. Meanwhile, if the workflow is failed, the myLEAD session is destroyed.

The FSM of the myLEAD session is defined as a set of state document and transition functions. First, the state document of the myLEAD session is represented as a simple XML document containing following elements,

- sessionID: the identifier of session that the FSM is associated with.
- type: the type of the session.
- state: the name of the current state.
- transitions: valid methods that can be processed in this state.
- created: the time this session was created.
- expires: the time this session ended.
- lastmodified: the time of the last modification of this document.

The sessionID element is a unique identifier of a myLEAD session. The type element represents the type of the FSM which drives this myLEAD session. The state element is a string representation of the current state. The transitions element is a list of allowed methods to be processed in this state. The created element provides the time of the creation of this session. This element is generated automatically by the myLEAD session. The expires element represents when the session will expire. When the session is created, or even during the session, the service requester can modify the lifetime of the session through the Web Service interface. The lastmodified element provides the time of the last modification of this document.

Each session stores their state document in the resource factory. Figure 5 illustrates our initial XML schema of the state information document. The resource factory can be a local database, file, or data structure in the memory.

```
<?xml version="1.0"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://ddel.cs.indiana.edu/myleadst
ateinfo"
xmlns="http://ddel.cs.indiana.edu/myleadstateinfo"
elementFormDefault="qualified">
<xs:element name="myLEAD_STATE">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="type" type="xs:string"/>
     <xs:element name="sessionID" type="xs:string"/>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="created" type="xs:dateTime"/>
     <xs:element name="lastmodified"
                    type="xs:dateTime"/>
      <xs:element name="expires" type="xs:dateTime"/>
```

```
        <xs:element name="transitions" type="xs:string"
                    maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Figure 5: XML schema of the state information document

The state transition functions are defined as a combination of the current state, action, and resultant state. The myLEAD session maintains a set of state transition functions. For each process of the request, the myLEAD session accesses the state transition function and determines the state transition.

5.2 Retrieval of the state information of the myLEAD session

The state information of myLEAD is available to the service requesters via Web service notification from myLEAD and Web services interfaces. Except for the expires element, the elements of the myLEAD states are not allowed to be updated externally through the Web services interface. The states of myLEAD are stored in the resource factory conforming to the schema depicted in the figure 2. This schema can interoperate with WS-Resource Properties.

5.3 Scheduling of myLEAD requests

The scheduling of the myLEAD requests involve multiple components of the myLEAD session. There are four major components in the myLEAD session: request monitoring queue, request scheduler, lifetime handler, and state document handler.

Each session maintains a request queue for the myLEAD service requests and internal events. The requests stored in the queue contain information about the types of the request, and the status of the request. When a request is queued, a request is validated based on the pre-defined FSM. If the request is not a valid activity within the current state, the request remains pending until subsequent request results in the pending request being valid for the sequence. Otherwise, the request is queued after the most recent valid request.

If there is no request queued ahead of the request, the request is processed. After the processing of a request, the myLEAD session accesses the state transition function, and performs the state transition if it is needed. The myLEAD session, then adjusts the request queue again with the new state information. Based on the valid activity specified in the current state information, pending requests are validated again. Until the myLEAD session gets the final state information or the session expires or is destroyed, the myLEAD session repeats this process.

6. Performance evaluation

We have profiled the performance of the myLEAD monitoring process; the costs involved are depicted in figure 6 for a number of sessions running simultaneously. The cost here corresponds to the measurement of the delay from when the request is delivered to the myLEAD client-side service up until the time when the request is queued in the request queue. Please note that this is an overall cost that comprises searching a session, scheduling the request, and also the retrieval of state information. This performance evaluation was performed on a 2.8 GHz Hyper-Threading enabled Workstation with 1.0 GB of real memory and 2.2 GB of virtual memory. The run-time environment is JRE 1.4.

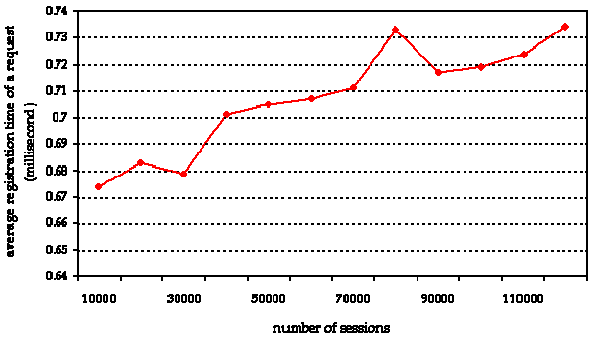

Figure 6: Performance Evaluation of the Monitoring process

7. Conclusions and future work

Grid technology enables access to stateful resources. However, the sharing of stateful resources between heterogeneous service requesters requires intelligent support from the service architecture in addition to the accessible state information. Hence, we have proposed an architecture for monitoring stateful interactions with Grid resources. This paper presents how we compose the state information based on the sequence in which these requests have been initiated, and formalize this aforementioned sequence using a FSM. We described the

process for monitoring and scheduling these requests. The analysis of the performance is also illustrated.

Our future work will focus on immutable experiments, publishing, and building trust. Immutable experiments allow users to reuse experiments without destroying the integrity of the earlier project. The user must have complete control over the mechanisms for publishing data products to the larger community. The system must also convey visual cues that build the user's trust—perhaps by guiding the user through a set of Web pages in a portlet that conveys a sense of entering a secure space. Publishing of data products would then be allowed only within that secure space.

References

[1] R.D. Stevens. A.J. Robinson, and C.A.Goble, "MyGrid: Personalized Bioinformatics on the Information Grid," Proc. 11th international Conference of Intelligent Systems for Molecular Biology, International Society for Computational Biology, 2003

[2] J.Peng, and K.Law, Reference NEESgrid Data Model, tech. report, Network for Earthquake Engineering Simulation Grid(NEESgrid), NEESgrid-2004-40,2004

[3] C.L.Qin et al.,"Myspace: Personalized Work Space I Astrogrid,"Proc, UK e-Science All Hands Meeting, University of Southampton,2003,pp361-365

[4] I.Foster et al., "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," white paper, Globus Project, 2002; www.globus.org/research/papers/ogsa.pdf.

[5] Karl Czajkowski et al., "The WS-Resource Framework," white paper, Globus Project, 2004, http://www.globus.org/wsrf/specs/ws-wsrf.pdf

[6] Beth Plale, Jay Alameda, Bob Wilhelmson, Dennis Gannon, Shawn Hampton, Al Rossi, and Kelvin Droegemeier User-oriented Active Management of Scientific Data with myLEAD *IEEE Internet Computing special issue on Internet Access to Scientific Data*, Jan/Feb 2005

[7] Kelvin K. Droegemeier, V. Chandrasekar, Richard Clark, Dennis Gannon, Sara Graves, Everette Joseph, Mohan Ramamurthy, Robert Wilhelmson, Keith Brewster, Ben Domenico, Theresa Leyton, Vernon Morris, Donald Murray, Beth Plale, Rahul Ramachandran, Daniel Reed, John Rushing, Daniel Weber, Anne Wilson, Ming Xue, Sepideh Yalda, Linked environments for atmospheric discovery (LEAD): Architecture, Technology Roadmap and Deployment Strategy, *To appear 21st Conf. on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology*, January 2005

[8] E. Christensen et al., "Web Services Description Language (WSDL)," white paper, W3C Note 15 March 2001, http://www.w3.org/TR/wsdl

[9] Bunting et al., "Web service Composite Application Framework (WS-CAF) Ver 1.0," July 2003, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-caf

[10] G.Singh et al.,"A Metadata Catalog Service for Data-Intensive Applications," Proc. ACM/IEEE Supercomputing 2003, IEEE CS Press, 2003, pp.33—49