iSERVO: Implementing the International Solid Earth Research Virtual Observatory by Integrating Computational Grid and Geographical Information Web Services

Mehmet Aktas[(1)], Galip Aydin[(1),*],Andrea Donnellan[(2)], Geoffrey Fox[(3)], Robert Granat [(4)], Lisa Grant[(5)], Greg Lyzenga[(6)], Dennis McLeod[(7)], Shrideep Pallickara[(1)], Jay Parker[(8)], Marlon Pierce[(9),*], John Rundle[(10)], Ahmet Sayar[(1)], and Terry Tullis[(11)]

(1) Community Grids Lab, Indiana University, Bloomington, IN 47404-3730 (2) NASA Jet Propulsion Laboratory, Mail Stop 183-335, 4800 Oak Grove Drive, Pasadena, CA 91109-8099, USA (email: Donnellan@jpl.nasa.gov). (3) Community Grids Laboratory, Departments of Computer Science, Physics, and School of Informatics, Indiana University, Bloomington, Indiana 47404-3730, USA (email: gcf@indiana.edu; phone +1 812-856-7977). (4) NASA JPL, Mail Stop 126-347, 4800 Oak Grove Drive, Pasadena, CA 91109-8099 (email: Robert.Granat@jpl.nasa.gov). (5) Environmental Analysis and Design, University of California-Irvine, Irvine, California, 92697-7070, USA (email: lgrant@uci.edu) (6) NASA JPL, Mail Stop 126-347, 4800 Oak Grove Drive, Pasadena, CA 91109-8099 (email: Gregory.Lyzenga@jpl.nasa.gov). (7) University of Southern Califoria, Mail Code 0781, 3651 Trousdale Parkway, Los Angeles, CA 90089-0742, USA (email: mcleod@pollux.usc.edu). (8) NASA Jet Propulsion Laboratory, Mail Stop 238-600, 4800 Oak Grove Drive, Pasadena, CA 91109-8099, USA (email: jay.w.parker@jpl.nasa.gov). (9) Community Grids Laboratory, Indiana University, Bloomington, Indiana 47404-3730, USA (email: mpierce@cs.indiana.edu, phone: +1 812-856-1212). (10) Department of Physics, University of California-Davis, One Shields Avenue, Davis, CA 95616-8677 USA (email: rundle@physics.ucdavis.edu). (11) Department of Geological Sciences, Brown University, Providence, RI 02912-1846 USA (email: Terry_Tullis@brown.edu).

* Corresponding author

Implementing ISERVO Using Grid and GIS Services

Abstract

We describe the goals and initial implementation of the International Solid Earth Virtual Observatory (iSERVO). This system is built using a Web Services approach to Grid computing infrastructure and is accessed via a component-based Web portal user interface. We describe our implementations of services used by this system, including Geographical Information System (GIS)-based data grid services for accessing remote data repositories and job management services for controlling multiple execution steps. iSERVO is an example of a larger trend to build globally scalable scientific computing infrastructures using the Service Oriented Architecture approach. Adoption of this approach raises a number of research challenges in millisecond-latency message systems suitable for internet-enabled scientific applications. We review our research in these areas.

Key Words: Web Services, computing web portals, computational grids, grid computing, earthquake simulation.

Introduction

In this paper we describe the architecture and initial implementation of the International Solid Earth Research Virtual Observatory (iSERVO) (quakesim.jpl.nasa.gov). We base our design on a globally scalable distributed computing infrastructure (often termed "cyber-infrastructure" or simply "Grid infrastructure" (Foster and Kesselman, 2003; Behrman, Hey, and Fox, 2003; Atkins et al, 2003) that enables on-line data repositories, modeling and simulation codes, data mining tools, and visualization applications to be combined into a single cooperating system. We build this infrastructure around Web Services-based approach.

Challenges for Solid Earth Research

The Solid Earth Science Working Group of the United States National Aeronautics and Space Administration (NASA) has identified several challenges for Earth Science research (Solomon, 2002). Particularly relevant for iSERVO are the following:

- How can the study of strongly correlated solid earth systems be enabled by space-based data sets?
- What can numerical simulations reveal about the physical processes that characterize these systems?
- How do the interactions in these systems lead to space-time correlations and patterns?
- What are the important feedback loops that mode-lock the system behavior?

- How do processes on a multiplicity of different scales interact to produce the emergent structures that are observed?
- Do the correlations allow for the capability to forecast the system behavior?

In order to investigate these questions, we need to couple numerical simulation codes and data mining tools to observational data sets. These observational data (including crustal fault data from the literature, GPS data, and seismic activity data) are now available on-line in internet-accessible forms, and the quantity of this data is expected to grow explosively over the next decade.

The challenges in solid earth modeling motivate a number of interesting research and development issues in distributed computer science and informatics. Key among these are providing programmatic access to distributed data sources; coupling remote data sources to application codes, including automated searching and filtering; coupling of complementary application codes that are deployed on geographically separated host computers; and providing human level interfaces to these remote services.

We note that the services described in this paper are different from, but complementary with, more traditional code parallelization techniques in high performance computing. As we discuss in more detail below, communication in parallel applications demands microsecond latencies, which can only readily be achieved in tightly coupled systems (such as clusters and "big iron" parallel machines of various flavors). As we describe below, these systems are best thought of as highly specialized services that communicate with each other on Internet timescales (milliseconds or longer).

The iSERVO team possesses a broad range of skills and tools that may be used to investigate solid earth research challenges. Team expertise includes the development high performance modeling and simulation applications for both the study of large, interacting earthquake systems and the detailed study of individual fault properties; federated database and ontology design; geological characterization of faults; and high performance visualization codes. Welding all of these components into a common distributed computing infrastructure is the subject of this paper.

A Web Service Grid Architecture

Problems in managing distributed computing resources, applications, data and users have been studied for many years. Viewed collectively, when such systems are managed by different organizations, we have what is typically called a computational Grid. Typical desired functionality in these systems includes remote command execution, data transfer, security, and high performance messaging. To scale globally, these systems must abandon tight coupling approaches such as distributed object systems and micro-second latency solutions such as MPI. Instead, they should adopt a Service Oriented Architecture (SOA) (Booth et al, 2004) that is compatible with millisecond (or longer) communication speeds (Fox, Pallickara, and Parastatidis, 2004). SOAs are implemented around two basic components: service definition languages (which describe how to invoke the remote service) and message formats for over-the-wire transmissions. In iSERVO, we have adopted the Web Service approach to building an SOA: we use WSDL (Christensen et al, 2001) for service description and SOAP (Gudgin et al, 2003) for message formats. This use of XML for both service description and messaging provides programming language

independence: the client does not know or need to know the implementation language of the service.

Web Service systems have an important design feature: service implementations are decoupled from the user interface components. This enables us to build a number of different clients that can interact with the same remote service, and vice versa. Browser-based computing portals are a typical way of managing client user interfaces and have been the subject of research and development work for a number of years (Fox and Hey, 2002). Currently this field is undergoing a revolution as component-based portal systems are being widely adopted, and standard component programming interfaces have been developed (Abdelnur, Chien, and Hepper, 2003; Gannon et al, 2004). This so-called "portlet" approach enables reusability of components: portals may be built out of standard parts that aggregate content and functionality from many different sources.

SOA and portal standards are not the only relevant standards for building systems such as iSERVO. The Open Geographical Information Systems (GIS) Consortium (OGC) (http://www.opengis.org) defines a number of standards for modeling earth surface feature data and services for interacting with this data. The data models are expressed in the XML-based Geography Markup Language (GML) (Cox et al, 2003), and the OGC service framework is being adapted to use the Web Service model. In this paper we describe implementation of GIS services to describe data relevant to the geophysical community (GPS, seismic events, and faults) that we then couple to more typical Grid services for code execution and file management.

Implementing iSERVO

We have implemented an initial set of services and portal components for addressing the problems described in the introduction. We have followed a Web Service-based Grid design described above that uses Web Service standards. The components of the system and their interactions are summarized in Figure 1. Users interact with remote services through a Web browser portal that is run by the User Interface Server (UIS). This portal generates dynamic web pages that collect input information from the user and deliver response messages. The UIS does not directly implement services such as job submission and file transfer. Instead, it maintains client proxies to these remote services. These proxies are responsible for generating the SOAP messages appropriate to the particular services' WSDL descriptions and for receiving the responses from the services. The UIS and most services are implemented in Java using the Apache Axis toolkit (http://ws.apache.org/axis/), but we have also implemented C++ services using gSOAP (van Engelen and Gallivan, 2002) for simple remote visualization.

A typical interaction involves the user selecting a code through the portal, setting up an input file in part through interactions with databases (such as the QuakeTables Fault Database (Chen et al, 2003, Grant et al, 2004), invoking the code and monitoring its progress, and having the output visualized through various third party tools of varying sophistication. These interactions are based on a dataflow model: services communicate by exchanging data files, which must be pulled from one server to another.

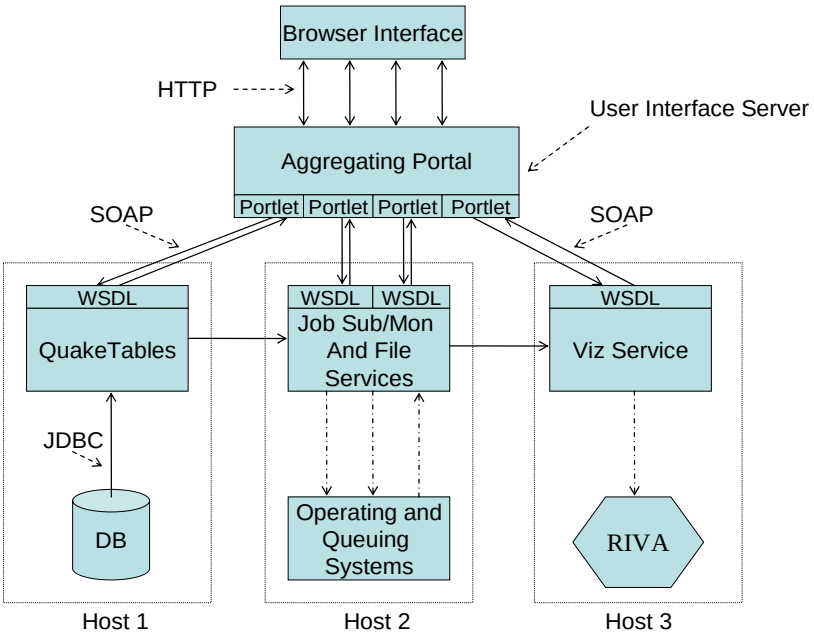

Figure 1 The architecture for the iSERVO portal and services uses Web Service and portal standards. JDBC stands for Java Database Connectivity and DB is an abbreviation for "database". "RIVA" is an example parallel visualization program. Other terms are defined in the text.

In building iSERVO, we have implemented a number of innovations on the standard model components. The portlet component model normally assumes local portlets with content that navigates to other web sites (news portals such as Yahoo and CNN are examples). We have built extensions to this simple model to allow portlet content to be managed remotely, have its display maintained within its component window through a series of navigations, maintain HTTP sessions state with remote content, pass HTTP GET and POST variables, and support SSL security.

Basic iSERVO services include remote command execution, file upload and download, and host-to-host file transfer. We do not directly alter the geophysical applications included in the portal but instead follow a "proxy wrapping" approach (Youn, Pierce, and Fox, 2004). Typically, applications require preprocessing of input files, post processing, and in general require task executions that are distributed across many different hosts. To support this sort of distributed service orchestration, we have developed a simple "workflow" service based on the Apache Ant project (http://ant.apache.org/). This service uses Ant as an engine that may be invoked remotely (as a service on Host 2 in Figure 1) and may also coordinate service invocations on remote hosts, as needed to complete its task.

More information on the iSERVO, including code downloads, documentation, and information for accessing demonstrations is available from http://quakesim.jpl.nasa.gov/.

iSERVO couples typical "Execution Grid" services such as described above with "Data Grid" services described in the following section. iSERVO applications work with many

different data sources, and we have developed services to automate the coupling of this data to application services. A typical problem is as follows: the iSERVO application RDAHMM (a Hidden Markov Model application) (Granat, 2004) needs as input either GPS or seismic activity records. Both data sources are available online, but there is no programmatic way of working with the remote data archives to, for example, filter data based on a user's criteria, assemble data sets from multiple archive records, or reformat the data based on the application's input formatting. Instead, a researcher typically downloads the data files and edits them by hand. To solve this problem, we have implemented GML-based services for describing these data records, and in the process we have unified several different data formats. These services allow the application user to build search filters on the desired data set (for example, returning seismic events larger than magnitude 5.0 within a particular region of interest since 1990). Additional filters reformat the data into one suitable for RDAHMM, and the data is then shipped to the location of the remote executable, which can then be invoked automatically. We thus replace the process of downloading and hand-editing the entire catalog.

Geographical Information System (GIS) Data Services

iSERVO data service requirements represent an excellent opportunity for further work leveraging open standards for services that will tie iSERVO to this larger community, allowing us to potentially incorporate many additional third party data sources and tools. The NASA OnEarth project (http://onearth.jpl.nasa.gov/) is an excellent example of a GIS project that may be incorporated with iSERVO in the future. As part of our GIS development work, we are currently re-implementing the OGC standard services Web Feature Service and Web Map Service as iSERVO-compatible Web Services.

We note that the GIS community has other data model and service standards than those defined by the OGC: The commercial vendor ESRI provides another prominent set of data standards along with extensive client tools. Our adoption of OGC standards is intended to take advantage of the significant amount of freely available GIS data that already exists in OGC formats. More importantly, OGC standards define an open architecture that may be integrate with Grid/Web service standards for distributed scientific computing discussed in the previous sections. We note further that ESRI and OGC interoperability tools already exist for obvious reasons, so adopting OGC standards does not preclude later integration of our data services with sophisticated ESRI software clients.

Advances in Geographical Information Systems (GIS) introduce several challenges for acquiring, processing and sharing data among interested parties. Different research groups, organizations, and commercial vendors develop their own data models and storage structures. Consequently the data is expressed in various formats and stored in various archives. These archives are often remotely accessible only through simple protocols (like FTP) that do not allow queries and filtering and which are difficult to integrate with geophysical applications. On the other hand the nature of the geographical applications requires seamless integration of spatial data from a range of providers to produce layers, maps, etc. As a result we see the interoperability between applications and data stores as a significant goal for any GIS.

As an example of how this goal can be accomplished we describe our design of a Service Oriented Architecture for serving a subset of geographical information. We first

review the existing data formats in our domain of interest and summarize our initial work for generating a common data format. The next section explains how we employed pure Web Services approach for data conversion, storage and query capabilities. The next section gives a brief discussion about our experience and findings on XML and Relational Databases, and the user interfaces we created for testing the Web Services.

Goals of the Project

We designed a service-based architecture for solving the aforementioned challenges. However, before implementing this system we identified several goals to make the scope of this project clear. These goals are as follows:

- Making GPS and Seismic data easily available for humans and applications alike;
- Providing seamless access to data repositories and computing resources;
- Providing a common data format for each information area;
- Supporting search capabilities on the catalogs for certain properties, filtering the search results, and retrieving the results in various formats; and
- Integrating data with the scientific applications.

Figure 2 illustrates the major components of the system for achieving these goals. Existing public archives maintained by the Southern California Earthquake Center (SCEC) and the Southern California Integrated GPS Network (SCIGN) are accessed through Web Services that download and reformat the data into GML (steps 1 and 2 in Figure 2). Data sources that we relied upon are more extensively documented at www.crisisgrid.org. We then store the converted data in either native XML or relational databases (step 3).

The above steps summarize administrative services that need to be performed once per external archive for initialization, followed by regular updates. Application users do not need to use these services. They do, however, make use of the search services (right hand side of Figure 2). These are also Web Services defined in WSDL and so may be accessed by various client programs.

Data Format Issues and GML

Depending on the data provider's use the catalogs are formatted in various ways. Any application or user who wants to make use of these legacy formats should understand what each column or data segment means and should write scripts to convert them into the target application's own legacy format. This introduces the tedious and resource consuming conversion problem.

Therefore this was the initial challenge for us with making the geographic data easily available: *data in this domain comes from different sources in different formats*. To solve this problem, we designed and implemented common data formats for each data type (GPS, fault, seismicity) that encapsulates several existing formats.

Our choice for the common data format was GML since it is widely accepted as a common exchange format for spatial information in GIS community worldwide. GML is a actually an extensive collection of integrated XML Schemas that collectively provide data models for describing geographic entities: features, coordinate reference systems, geometry, topology, time, units of measure and generalized values.

Our GML Schemas support following formats:

- Seismic data formats:
    - SCSN, SCEDC, Dinger-Shearer, Haukkson
- GPS data formats:
    - JPL, SOPAC, USGS

More details of these formats (both original and our GML compatible versions) are available from http://www.crisisgrid.org/html/servo.html. GML schema for earthquake faults (available in both GML 2 and GML 3) are also available.

The next step after creating a common exchange format for the catalogs was to design a service-based system compatible with the architecture of Figure 1 to process these data. The first thing needed to be done in the system was to collect data from various sources. GPS and seismicity catalogs are provided by several organizations and they are available to the public via FTP or HTTP servers. We wrote clients for retrieving the catalogs and saving them in our server for further processing. These retrieval applications are also provided as Web Services so that any workflow scheme can schedule regular catalog updates.

The reference implementation of the user interface to the system allows a user to select one of the two paths to follow after acquiring the catalogs:

1. Convert the catalogs into GML format and save them as XML files. These files are then inserted into an XML database.
2. Insert the catalog data into relational database.

The choice in this step is introduced because of the use of both relational and XML databases in the system. We extensively tested and used both types of the databases.

Databases

Since the message exchange format of Web Services is XML and since we convert our data to XML at the beginning of the process we initially implemented data storage using native XML databases. Implementations were tested with both the Berkeley XML Database system (Sleepycat) and Apache Xindice. However, after extensive testing we found the XML databases performance to be unusable for GML records of the size needed to store existing on-line catalogs such as the Southern California Earthquake Center's seismic activity records. We re-implemented the storage mechanism (without changing the WSDL and so without updating the clients) to use a common, open-source relational database (MySQL), which demonstrated acceptable query time.

Searching the Catalogs

We provide a Web Service for searching the catalog databases. The results are sent to the client application as GML documents. Since there is only one GML Schema for a particular data group (GPS or Seismicity) it is easy to validate and parse the received GML document and extract the information. XPATH (Clark and DeRose, 1999) is used as the query language for the XML Databases.

Status

Documentation and software for this work are available from www.crisisgrid.org.

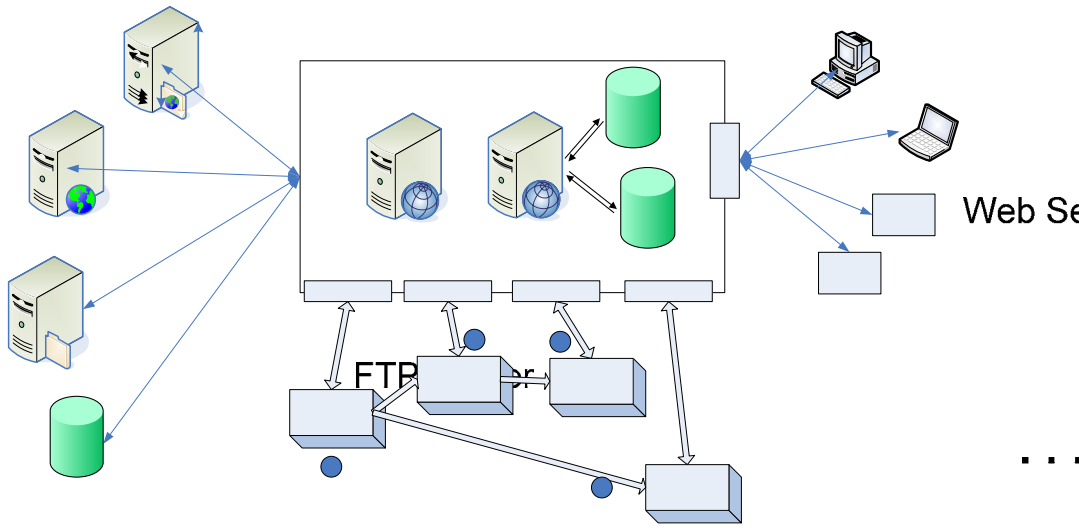

Figure 2 Major parts of the architecture and a sample workflow for processing geo-data using Web Services.

User Interfaces

We have created a set of user interfaces to the Web Services for demonstration purposes. These interfaces also show the steps for making a set of geographic data available to search and retrieve via web services. The search client enables users to create database queries via web page forms. The search results are shown to the user as a simple text page. We use the XML Pull Parser (www.extreme.indiana.edu/xgws/xsoap/xpp) to quickly extract data from received GML file and create the web page output.

Our architecture provides several WSDL interfaces to enable both human and application clients to use the services. WSDL does not require the client to be bound to a particular programming language or environment, thus supporting client tools in different programming languages such as Java classes or C++ applications. To demonstrate this feature we have created a simple C++ client using gSOAP (van Engelen and Gallivan, 2002) along with our JSP interfaces and Java clients.

Designing OGC Web Services

As we have described above, GML is a data modeling language that can be used to encode geophysical data. We may then store this data in various archival systems and design Web Services that can query, retrieve, and update the data. These web services are compatible with the "Execution Grid" services illustrated in Figure 1.

These Web Services, because they use a generic data model, may be standardized and generalized. The OpenGIS Consortium defines specifications for several such services, with the Web Feature Service and the Web Map Service as two prominent examples.

These services, unfortunately, are not designed to be Web Service compatible (they do not use WSDL or SOAP but rather lower level HTTP GET/POST conventions for messaging). In order to adapt these services to the QuakeSim architecture while taking advantage of existing OGC resources, we have redesigned these OGC services to use Web Service standards.

The Web Feature Service (WFS) (Vretanos, 2002) describes standards to publish, update, and delete geographic features, such as faults and GPS stations. We designed a Web Service version of OGC WFS that provides WSDL interfaces for the required capabilities. Instead of using HTTP Post, the user or the client application communicates with the WFS using SOAP messaging. The results of the requests are sent to the user as GML documents.

One important property of the WFS is that it can serve multiple feature types. Different features from different data stores are integrated with the WFS and the clients do not realize that the features are retrieved from several sources.

We also are implementing a Web Service version of Web Map Service to generate maps (de La Beaujardiere, 2004). The Web Map Service gets data from various Web Feature Services. We are also building bridging services that allow our Web Service-compatible WMS to interact with non-Web Service versions of WMS. This will allow us to make use of extensive existing mapping resources, such as the NAS OnEarth project (http://onearth.jpl.nasa.gov/). The interaction between these two services is based on SOAP messaging because of the Web Service standards. This work is currently under development.

Status

The initial Web Feature Service implementation work is completed and is being packaged for availability at www.crisisgrid.org. The Web Map Service implementation is in development but links to demonstrations will be available from www.crisisgrid org.

GIS Information Services

Services such as the Web Map and Web Feature service, because they are generic, must provide additional, descriptive metadata in order to be useful. The problem is simple: a client may interact with two different Web Feature Services in exactly the same way (the WSDL is the same), but the Web Feature Services may hold different data. One, for example, may contain GPS data for the Western United States while the other has GPS data for Northern Japan. Clients must be able to query information services that encode (in standard formats) all the necessary information, or metadata, that enables the client to connect to the desired service. This is an example of the very general problem of managing information about Web Services. To address these problems, we are designing a general purpose information system, the Fault Tolerant High Performance Information System (FTHPIS), that we are applying initially to problems in GIS information management .

An approach to solve the problem of locating resources of interests in iSERVO environment was first introduced in (Aktas et al 2004). This approach suggests a centralized discovery model by utilizing semantic web technologies to locate iSERVO resources where resources were limited to iSERVO codes and data.

In a FTHPIS, there is a need for registry services to make the information about services available. We use the Universal Description, Discovery, and Integration (UDDI) (Bellwood, Clement, and von Riegen, 2003) specifications in our design as centralized registry. UDDI offers users a unified and systematic way to find service providers through a centralized registry of services. We design an extension to existing UDDI Specifications in order to provide dynamically updated service registry data.

UDDI provides a centralized approach to the Web Services registry research problem. However, one should be able to locate a Web Services satisfying a query in a decentralized, dynamically changing, distributed environment as well. To do this, discovery architecture needs to be defined. As we consider the volatile behavior of Web Services, such decentralized approach should provide dynamic discovery of services where the temporarily connected Web Service can be discovered. To this end, we design our implementation based on WS-Discovery Specifications which was recently released.

WS-Discovery (Beatty et al, 2004) defines a multicast protocol to locate services. It allows dynamic discovery of services in ad hoc and managed networks. However, WS-Discovery does not define a metadata model to describe Web Service. To this end, we also define an abstract metadata model for Web Service. This will allow us to pose more complex queries for WS-Discovery.

We classify metadata associated with Web Services as dynamic metadata and static metadata. Dynamic metadata is the session (or state) metadata generated by the individual interactions with Web Services. Such metadata describes the context of the session and has a lifetime. There are different approaches specifying session metadata. For instance, WS-Context (Bunting et al, 2003) provides an abstract context defining such metadata. Static metadata is the metadata describing a Web Service profile such as its usage cost, availability, bandwidth, computing power, storage capability, etc. We extend existing UDDI and WS-Context specifications in order to associate metadata with Web Service descriptions.

A FTHPIS consists of information services. An information service is a Web Service that provides registry to make the information about services available and also maintains a repository of context information of web services. An information service combines both WS-Context service and WS-Registry service. WS-Registry part of an information service extends existing UDDI technology where one can annotate service descriptions with metadata describing “Quality of Service” aspects of services. The WS-Context part of an information service extends the existing WS-Context service where one can track contexts shared between multiple participants in Web Service interactions. In Figure 3, we show our design where an Information Service provides a gateway to UDDI Registry and WS-Context replica groups. Our design suggests replication of databases both for registry of services or contextual information replicated with a sequential consistency model and replicated-write consistency protocol in order to provide fault tolerance and avoid single point of failure. We plan on utilizing a caching mechanism to improve performance and reduce the response latency to service discovery queries. In our design for FTHPIS, we suggest each information service peer caches SOAP messages as (query, respond) message pairs. We use read-only caches to improve response time, so updates operation can only performed on the servers where the origin of the cached data resides. In order to provide cache-coherence we simply suggest propagating the updates to caches.

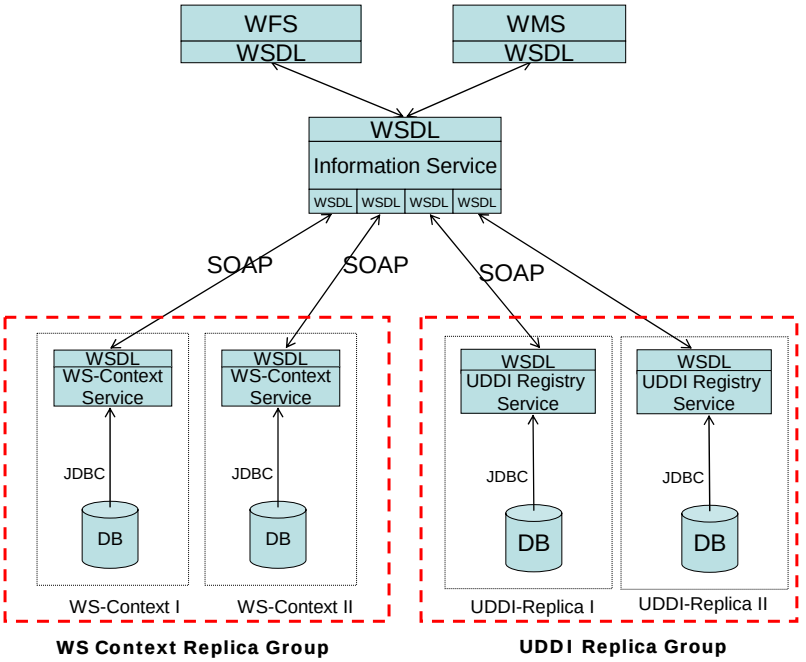

Figure 3 Interaction between an Information Service and OGC Web Services such as Web Map Service and Web Feature Service

We illustrate a motivating scenario where Information Services interacts with Web Map Services and Web Feature Services in Figure 4. In this scenario, Web Feature Services are published into the UDDI-Registry. Each Web Feature Service provides data layers corresponding to geographic entities. An important challenge is that UDDI does not natively support registry of services with a bounding box corresponding to a data layer and representing a location of interest. To overcome this problem, we use a standard capability of UDDI registries which is to classify service entries according to predefined taxonomies. We use geographic taxonomies to classify UDDI service entries based on spatial coverage. This methodology allows us to make coordinate based spatial queries on the UDDI-Registry. In Figure 4, Web Map Services interact with the Information Service to find out available WFS (data services) satisfying the data requirements of a map. As the Information Service responds a query of WMS with metadata of services satisfying the query, WMS can then start interacting with corresponding WFS to acquire the data layers needed to create maps.

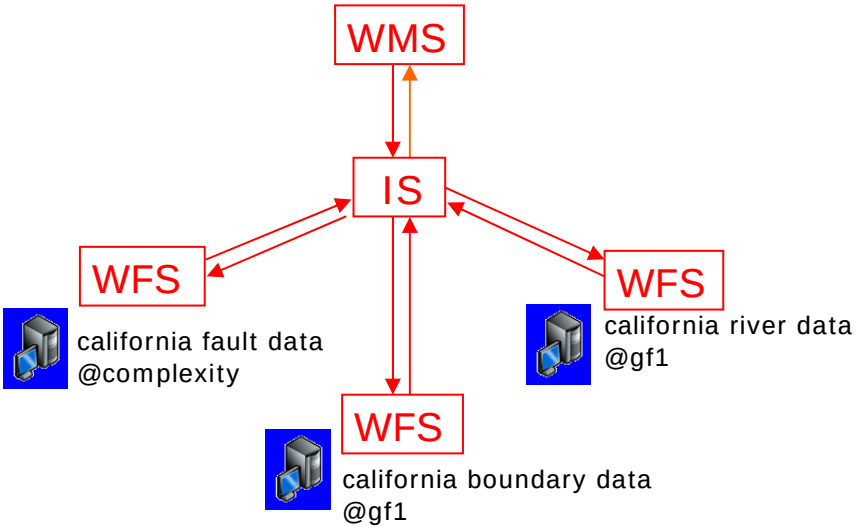

Figure 4: Interaction between an Information Service (IS) and OGC Web Services such as Web Map Service (WMS) and Web Feature Service (WFS).

Status

Our information system is currently under development.

Future Directions for iSERVO

Web Services in the SOA approach communicate with SOAP messages in a loosely coupled fashion. Such systems demand a number of features: fault tolerance, reliable messaging, message level security (such as authorization and encryption), and message virtualization for firewall tunneling. We term this general class of messages as the Web Service "Internet-On-Internet" (IOI) problem: many Web Service standards reimplement common TCP/IP features within the SOAP message. We see this as an interesting development, as it allows us to use a messaging system infrastructure (which we do control) to provide quality of service that is independent of the underlying network (which we do not control). We have implemented such a messaging system infrastructure, NaradaBrokering (http://www.naradabrokering.org), and are extending it to support Web Service invocations natively. This is described in more detail in Fox, Pallickara, and Parastatidas, 2004.

A common objection to the Web Service approach is that it is too slow. Message speeds across network connections are on the order of milliseconds at best, and so unsuitable for classic metacomputing. For typical iSERVO applications, this is not an issue: model runs may take several hours or days to complete. The applications themselves may be deployed on clusters or supercomputers and may be parallelized by traditional techniques; we treat such applications as a single service component (Youn, Pierce, and Fox, 2004). However, there are classes of problems, particularly in interactive remote

visualization and high performance transfers of large data sets, in which maximum network performance is needed. We are currently researching this within the NaradaBrokering system. The IOI approach for Web Services will allow us to replace TCP/IP with much more efficient UDP transmissions, while retaining desirable TCP/IP features in the SOAP messages.

On top of the IOI infrastructure, we must provide information services: how can one encode in machine readable way what a particular service in Figure 1 actually does? What data does the service provide or require? How do other components in the system discover it? How can it be classified? How can complicated service interactions be coordinated? We term the higher level information Grid that manages this sort of information as the "Context and Information Environment" (CIE). The information system described in this paper is an implementation of some of the foundation-level services that are needed to build a CIE. All the information requirements that we have enumerated are part of a larger problem in metadata management. In Web and Grid Services, this is an open problem with many competing solutions. iSERVO represents a excellent test case of a real Grid with real information system requirements that can be used to validate the competing solutions.

Acknowledgments

This work was funded by the Computational Technologies Program and the Advanced Information Systems Technology Program, both of NASA's Earth Science Technology Office.

References

Abdelnur, A., Chien, E., and Hepper, S., (eds.) (2003), *Portlet Specification 1.0*. Available from http://www.jcp.org/en/jsr/detail?id=168.

Aktas, M. S., Pierce, M., Fox, G. and Leake, D. (2004) *A We Based Conversational Case-Based Recommender System for Ontology Aided Metadata Discovery*, 5th IEEE/ACM International Workshop on Grid Computing (GRID 2004)

Atkins, D., Droegemeier, K., Feldman, S., Garcia-Molina, H., Klein, M., Messerschmitt, D., Messina, P., Ostriker, J., and Wright, M. (2003), *Revolutionizing Science and Engineering Through Cyberinfrastructure*, Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure. Available from http://www.cise.nsf.gov/sci/reports/atkins.pdf.

Beatty, J., Kakivaya, G., Kemp, D., Kuehnel, T., Lovering, B., Roe, B., St. John, C., Schlimmer, J., Simonnet, G., Walter, D., Weast, J., Yarmosh, Y., and Yendluri, P. (2004), *Web Services Dynamic Discovery (WS-Discover)*, available from http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-discovery.pdf.

Bellwood, T., Clement, L., and von Riegen, C. (eds) (2003), *UDDI Version 3.0.1: UDDI Spec Technical Committee Specification.* Available from http://uddi.org/pubs/uddi-v3.0.1-20031014.htm.

Berman, F., G. C. Fox, and T. Hey, (eds.), Grid Computing: Making the Global Infrastructure a Reality (John Wiley & Sons, 2003).

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D., *Web Services Architecture,* W3C Working Group Note 11 February 2004. Available from http://www.w3.org/TR/ws-arch/.

Bunting, B., Chapman, M., Hurley, O., Little, M., Mischkinky, J., Newcomer, E., Webber, J., and Swenson, K., *Web Services Context (WS-Context),* available from http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CTX.pdf.

Chen, A. Y., Chung, S., Gao, S., McLeod, D., Donnellan, A., Parker, J., Fox, G., Pierce, M., Gould, M., Grant, L., and Rundle, J (2003), *Interoperability and Semantics for Heterogeneous Earthquake Science Data* in Proceedings of 2003 Semantic Web Technologies for Searching and Retrieving Scientific Data Conference, October 20, 2003, Sanibel Island, Florida.

Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001), *Web Service Description Language (WSDL) 1.1*. W3C Note 15 March 2001.

Cox, S., Daisey, P., Lake, R., Portele, C., and Whiteside, A. (eds) (2003), *OpenGIS Geography Markup Language (GML) Implementation Specification.* OpenGIS project document reference number OGC 02-023r4, Version 3.0.

Clark, J. and DeRose, S., *XML Path Language (XPath) Version 1.0*, W3C Recommendation, 16 November 1999.

de La Beaujardiere, Jeff (2004), *Web Map Service,* OGC project document reference number OGC 04-024.

Foster, I. and Kesselman, C. (eds), The Grid 2: Blueprint for a New Computing Infrastructure. (Morgan Kaufmann, 2003).

Fox, G. and Hey, A (eds.) (2002) Concurrency and Computation: Practice and Experience, Vol. 14, No. 13-15.

Fox, G., Pallickara, S., and Parastatidas, S. (2004), *Towards Flexible Messaging for SOAP Based Services*, in Proceedings of the IEEE/ACM Supercomputing Conference, November 2004. Pittsburgh, PA.

Gannon, D., Alameda, J., Chipara, O., Christie, M., Dukle, V., Fang, L., Farrellee, M., Fox, G., Hampton, S., Kandaswamy, G., Kodeboyina, D., Krishnan, S., Moad, C., Pierce, M., Plale, P., Rossi, A., Simmhan, Y., Sarangi, A., Slominski, A., Shirasuna, S., and

Thomas, T. (2004), *Building Grid Portal Applications from a Web-Service Component Architecture*, to appear in IEEE Dist. Comp. Available from http://grids.ucs.indiana.edu/ptliupages/publications/portal-apps-arch.pdf.

Granat, R., (2004) *Regularized Deterministic Annealing EM for Hidden Markov Models*, Doctoral Dissertation, University of California, Los Angeles.

Grant, L., Donnellan, A., McLeod, D., Pierce, M., Fox, G., Chen, A., Gould, M. and Sung, S.-S (2004) *A Web Service Based Universal Approach to Heterogeneous Fault Databases*, submitted to Comp. Sci. Eng.

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., and Nielsen, H. (2003), SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation 24 June 2003. Available from http://www.w3c.org/TR/soap12-part1/.

Solomon, S. C., (chair), 2002, "Living on a Restless Planet", Solid Earth Science Working Group Report. Available from http://solidearth.jpl.nasa.gov/PDF/SESWG_final_combined.pdf.

van Engelen, R. and Gallivan, K (2002), *The gSOAP Toolkit for Web Services and Peer-to-Peer Computing Networks*, in Proceedings of IEEE CCGrid Conferences 2002. Available from http://www.cs.fsu.edu/~engelen/soappaper.html.

Vretanos, P (ed.) (2002), *Web Feature Service Implementation Specification*, OpenGIS project document: OGC 02-058, version 1.0.0.

Youn, C., Pierce M., and Fox, G. (2004) *Building Problem Solving Environments with Application Web Service Toolkits*, to be published in Future Generation Computing Systems (in press).