Information Services for Dynamically Assembled Semantic Grids

Mehmet S. Aktas[(1), (2)], Geoffrey C. Fox[(1), (2), (3)], Marlon Pierce[(1)]

(1) Community Grids Laboratory, Indiana University
501 N. Morton Suite 224, Bloomington, IN 47404
{maktas, gcf, mpierce}@cs.indiana.edu
http://www.communitygrids.iu.edu/index.html
(2) Computer Science Department, School of Informatics, Indiana University
(3) Physics Department, College of Arts and Sciences, Indiana University

Abstract

Many large semantic systems can be described as Semantic Grids of Semantic Grids with large amounts of relatively static services and associated semantic information combined with multiple dynamic regions (sessions or subgrids) where the semantic information is changing rapidly. We design a hybrid Information Service supporting both the scalability of large amounts of relatively slowly varying data and a high performance rapidly updated Information Service for dynamic regions. We use the two web service standards UDDI and WS-Context in our system.

1. Introduction

E-Science Semantic Grids can often be thought of as dynamic collection of semantic subgrids where each subgrid is a collection of modest number of services that assembled for specific tasks such as forecasting earthquakes [1]. We term an actively interacting (collaborating) set of managed services as a *Gaggle* where services are put together for particular functionality. Semantic Grid may consist of several *Gaggles* each featuring intense local activity with less intense inter-gaggle interactions. Each *Gaggle* maintains most dynamic information which is the session related metadata generated as result of interactions among Grid/Web Services. *Gaggles* are also called as Grid Processes in the China National Grid. An infrastructure of the Semantic Grid is discussed in [2] where Grid Processes may be defined as cooperative processes that support management and integration of business processes. We also note that *Gaggles* may be composed from other "sub" *Gaggles* hierarchically.

Extensive metadata requirements of both the worldwide Grid and smaller sessions or "gaggles of grid services" that support local dynamic action may be investigated in diverse set of application domains such as sensor and collaboration grids. For example, workflow-style Geographical Information Systems (GIS) Grids such as the Pattern Informatics (PI) application [1] require information systems for storing both semi-static, stateless metadata and transitory metadata needed to describe distributed session state information. The PI application is an earthquake simulation and modeling code integrated with streaming data services as well as streaming map imaginary services for earthquake forecasting.

Handling information requirements of these applications requires high performance, fault tolerant information systems. These information systems must be decentralized, relocate metadata to nearby locations of interested entities and provide efficient access, storage of the shared information, as the dynamic metadata needs to be delivered on tight time constraints within a *Gaggle*. Information Services support discovery and handling of services through metadata and are vital components of Grids [3].

We identify the following problems in Information Services supporting both traditional and Semantic Grids. First, Grid Information Services need to be able to support dynamically assembled service collections gathered at any one time to solve a particular problem. Most of the traditional Grid Information Services [4-5] however are not built along this model. Second, Information Services should scale in numbers and geographical area. Most existing solutions [4-5] however have centralized components and do not address scalability and high performance issues. Third, Information Services need to be able to take into account user demand changes when making decisions on metadata access and storage. Fourth, Information Services need to be able to provide uniform interface for publishing and discovery of both dynamically generated and static information. Existing Grid Information Services however do not provide such

capabilities. We therefore see this as an important area of investigation. This paper presents our design of an architecture to address the identified problems above. We describe a novel architecture for fault tolerant and high performance Information Services in order to manage distributed, dynamic session related metadata while providing consistent, uniform interface to both static and dynamic metadata.

The main contributions of this paper are two-fold. First, we present a novel architecture for a WS-Context [13] complaint metadata catalog service supporting distributed or centralized paradigms. We use an extended version of UDDI [14] for slowly varying metadata and present a uniform and consistent interface to both short-lived dynamic and slowly varying quasi-static metadata. We explore the application of context (session-related dynamic metadata) management in Grid systems to correlate activities in workflow-style applications, by providing a novel approach for management of widely distributed, shared session-related dynamic metadata. We investigate the problem of distributed session management in Grid applications, by providing an approach for distributed event (session metadata) management system enabling session failure recovery or replay/playback capabilities. We also address lack of search capabilities in Grid Information Services, by providing uniform search interface to both interaction independent and conversation-based metadata enabling service discovery through events.

Our second contribution is the application of topic-based publish/subscribe methods to the problems of dynamic replication methodology to support dynamic metadata. We utilize a multi-publisher, multicast communication middleware and a topic-based publish/subscribe messaging system as a communication middleware to exchange messages between peers.

This paper is organized as follows. Section 2 reviews the state of art in existing information services and replica hosting environments. Section 3 reviews our design for information systems to support *Gaggles* paying particular attention to distributed data management aspects of the system. In Section 4, we summarize and discuss future work.

2. Background

Most existing decentralized solutions to Information Services can be broadly categorized by the manner of in which decentralization is realized such as a) hierarchical, structured and b) unstructured, peer-to-peer (P2P). In structured architectures, components of the system are strictly controlled and may depend on each other for publishing and discovery of information. For an example, Globus Monitoring and Discovery System (MDS4) [4] has a hierarchical architecture where there is a single top-level Information Service that presents a uniform interface to clients to access data, while the data is collected by lower-level information providers. Another example is the structured P2P systems where the nodes in the systems are equally enabled and controlled and service information is disseminated to all nodes [6].

Unstructured P2P architectures can be characterized as systems where there is lack of control on the capabilities of the system nodes and where there is no organizational structure. For an example, Relational Grid Monitoring Architecture (R-GMA) [5] presents a P2P architecture where consumers directly connect to information providers to retrieve the data without intermediary nodes. An extensive survey on Grid Information Services can be found at [7]. Architectures with pure decentralized storage models have focused on the concept of distributed hash tables (DHT) [6]. DHT approach assumes possession of an identifier such as hash table that identifies the service that need to be discovered. Each node forwards the incoming query to a neighbor based on the calculations made on DHT. Although the DHT approach provides good performance on routing messages to corresponding nodes, it has various limitations such as primitive query capabilities on the database operations. Here, we design an architecture which can be defined as an unstructured P2P approach to P2P/Grid environment. We use multi-publisher message broadcasting through a topic-based publish/subscribe messaging system, which support access and storage decisions among distributed nodes.

Well-defined descriptions of resources, services and data constitute metadata. Metadata can be represented using varying metadata models such as XML Schema or Semantic Web languages (RDF, OWL, etc.). Here, we are mainly concerned with managing the metadata and delivering to clients, not with knowledge processing. We presume the metadata models to be application-specific and not defined by us. To this end, we are concentrating on distributed computing problems of managing metadata in the Semantic Grid.

We use replication, a well-known and commonly used technique to improve the quality of metadata hosting environments, in our architecture. Sivasubramanian et

al. [8] give an extensive survey on reviewing research efforts on designing and developing World Wide Web replica hosting environments, as does Robinovich in [9], paying particular attention to dynamic replication. As the nature of our target data is dynamic, we focus on data hosting systems that are handling with dynamic data. These systems can be discussed under following important design issues: a) distribution of client requests among data replicas b) selection of hosting environments for replica placement c) consistency enforcement.

Distribution of client requests is the problem of redirecting a client to the most appropriate replica server. Most existing solutions to this problem are based on DNS-Server such as in [10-11]. These solutions utilize a redirector/proxy server that obtains physical location of collection of data-systems hosting a replica of the requested data, and choose one to redirect client's request. **Replica placement** is another issue that deals with selecting data hosting environments for replica placement and deciding how many replicas to have in the system. Existing solutions, that apply dynamic replication, monitor various properties of the system when making replica placement decisions [11-12]. For instance, Radar [11] replicates/migrates dynamic content based on changing client demands. Spread [12] considers the path between the data-system and client and makes decisions to replicate dynamic content on that path. The **consistency enforcement issue** has to do with ensuring all replicas of the same data to be the same. Various techniques have been introduced in consistency management. For instance, the Akamai project [10] introduces versioning where a version number is encoded to document identifier, so that client would only fetch the updated data from the corresponding data hosting system. Radar [11] applies primary-copy approach where an update can be done only on the primary-copy of the data.

Our architecture differs from web replica hosting systems in the following ways. First, the intended use of our architecture is not to be a web-scale hosting environment. The scale of our target systems is in the order of a few dozen to at most a thousand entities participating in a session. Our target domains range from collaboration systems such as GlobalMMCS [17] project to geographical information systems such as Pattern Informatics GIS-Grid. The participant entities of these systems might dynamically generate metadata during a session. Such metadata can be expected to be small in size and big in the volume depending on the Grid application. Second, existing solutions to dynamic replication assume all data-hosting servers to be ready and available for replica placement and ignore "dynamism" in the network topology. In reality, data-systems can fail anytime and may present volatile behavior. We use a pure Peer-to-Peer approach, which is based on multi-publisher multicast mechanism, when distributing access and storage requests to data-systems.

3. Information Services

We have designed a novel architecture to Information Services presenting a uniform interface to support handling and discovery of not only quasi-static, stateless metadata, but also session related metadata. In order to be compatible with existing Grid/Web Service standards, we based the interface of our system on the WS-Context and UDDI Specifications. We have extended and integrated both specifications to provide uniform and consistent service interface to both dynamic and static metadata.

Our approach is to utilize the existing state-of-art systems for handling and discovering static metadata and address the problems of distributed management of dynamic metadata. To do this, on receiving querying/publishing metadata requests, the system applies following steps to process service metadata. First, the system separates dynamic and static portions of the metadata. For instance, static metadata could be throughput or location of a service whereas dynamic metadata could be session identifier pointing to a workflow session in which the service is participating. Second, the system delegates the task of handling and discovery of static portion of the metadata to UDDI. As we research UDDI Specifications to integrate with our system, we have encountered various limitations in its capabilities which we address in a separate paper [15]. Third, the system itself provides handling and discovery using dynamic portions of the metadata in the metadata replica hosting environment.

The intended use of our approach is to support information in dynamically assembled Semantic Grids where "real-time" decisions are being made on which services to tie together in a dynamic workflow to solve a particular problem. One may think of WS-Context complaint Information Services as the metadata catalog for semantic metadata as in an RDF triple store. The semantic metadata expresses the relationships between resources and the applications that access the metadata catalog deduct further (inferred) information. In our

design, the distinctive semantic richness comes from the highly dynamic architecture with metadata from more than two services (in contrast WS-Transfer, WS-Metadata Exchange Specifications that only easily get semantic enhancement from the two services that exchange metadata). We discuss various research issues in building Information Services for dynamically assembled Semantic Grids in the following section.

3.1. Fault Tolerant High Performance Information Services

We have considered two application domains from sensor/GIS and collaboration grids to demonstrate the use of our system: Global Multi-media and Collaboration System (GlobalMMCS) [17] and PI GIS-Grid. GlobalMMCS is a peer to peer collaboration environment where videoconferencing sessions can take place. Any number of widely distributed services can attend to a collaboration session. GlobalMMCS requires persistent archival of session metadata to provide replay/playback and session failure recovery capabilities. The PI GIS-Grid is a workflow-style Grid application which requires storage of transitory metadata needed to correlate activities of participant entities. Both application domains require a decentralized metadata hosting environment which can support both scalability (of large amounts of information) and performance requirements (of rapidly updated dynamic information). To this end, we identify two important research issues that need to be answered in our design: fault tolerance and high performance.

We use replication technique to provide fault tolerance and high performance which improves the quality of our data hosting environment. If one of the redundant storage elements goes down, it automatically consults remaining elements to restore itself. The replication technique can also lead into high performance by reducing the time between a client issuing a request and receiving the corresponding response. As the nature of our data is very dynamic, we use dynamic data replication technique, where data replicas may be created, deleted, or migrated among hosting data-systems based on changing user demands. Two important aspects of dynamic replication are access and storage algorithms.

Access Algorithm: The access algorithm distributes client requests to appropriate replica hosting data-systems. Our model is based on pure Peer-to-Peer approach where each node can probe all other nodes in the network to look up metadata. A primary role of the access algorithm is the discovery of one or more data-systems hosting the requested metadata. This discovery process consists of two steps: data-system discovery and access. The first step concerns with selection of data-systems that can answer the client requests. The second step is to inform the data-system that is most appropriate for handling the request. In the first step, to find metadata, a node sends a probe message to all other nodes through a software multicast mechanism; target data-systems that host the metadata matching the probe send a response directly to requestor node. Here, response message consists of information regarding how well the data-system can handle this query. For instance, such information may include proximity information between the client and the data-system. On receiving response messages, the requestor node chooses the most appropriate data-system that can handle the request. In the second step, the requestor node sends the client request to the chosen data-system particularly asking to handle the request.

Storage Algorithm: Storage algorithm selects data-systems for replica placement and decides how many replicas to have in the system. In our design, storage decisions are made autonomously at each node without any knowledge of other replicas of the same metadata. The storage decision is made based on the client requests served by that node. Storage process consists of two separate steps such as metadata placement and metadata creation. The first step has to do with selection of data-systems that should hold the replica and the second step has to do with metadata replica creation. In the first step, each node (data-system) runs the storage algorithm which defines client request thresholds for replica creation and deletion. If a metadata entry is in high demand which is above a pre-defined threshold, then the metadata is replicated. If a metadata entry is in low demand which is below a pre-defined threshold, it will be deleted. To replicate metadata, a node sends a “storage” message to all other nodes through a software multicast mechanism; target data-systems, that have available space, send a respond to directly requestor node. Here, the response message consists of various decision metrics such as client proximity information. On receiving the response messages, replica placement algorithm chooses the most appropriate data-system to replicate the metadata. In the second step, the requestor node sends a replica creation message directly to the chosen data-system asking to store a replica of metadata in consideration. This process creates a dynamic metadata storage in which metadata is moved based on changing client demands.

Multi-publisher Multicasting Communication Middleware: An importing aspect of our system is that we utilize software multicasting capability which is an important communication medium supporting the ability to send out access and storage requests to the all nodes of the system. Any node can publish and subscribe to topics which in turn create a multi-publisher multicast broker network as communication middleware. Here, the publisher does not need to know the location and identities of receivers. It publishes a message to a topic to which all nodes subscribe. We use NaradaBrokering's (NB) [16] publish/subscribe mechanism as a communication middleware for message exchanges between peers.

3.2. System Components

Our proposed architecture consists of various modules such as Query and Publishing, Expeditor, Access, Storage and Sequencer Modules. Architectural design of our system is illustrated in Figure 1.

Context Query and Publishing Modules: These modules receive client requests through a uniform service interface for publishing/discovering dynamic and static metadata. The client query/publishing requests are processed and dynamic metadata parts of the queries are extracted. Then, the request is forwarded to Expeditor Module to find the results. Likewise, static metadata portion of the requests is relayed to external UDDI Service to publish/discover services through static metadata.

Expediter Module: This is a generalized caching mechanism. Each node has a particular expediter. One consults the expediter to find how to get (or set) information about a dataset in an optimal fashion. The expediter is roughly equivalent to replica catalog in classic Grids.

Access Module: This module runs the access algorithm mentioned above. It support request distribution by publishing messages to topics in NB network. It also receives messages (in respond to client request) coming from other peers and forward these query messages to Expediter Module. The Access Module locates the nodes that are closest in terms of network distance with lowest load balance from the node requesting access to the communal node in question. It also takes into account the load balance of each responding data-system when choosing the right data-system.

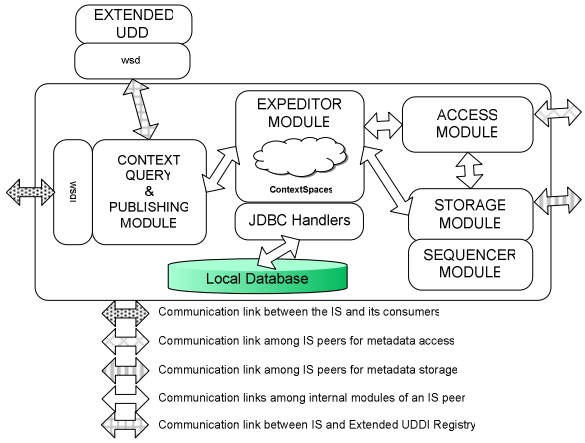

Figure 1 – Architecture of an Information Service running on each peer

Storage Module: This module runs the storage algorithm. It interacts with the Expediter Module and applies the storage algorithm to all local Context metadata. If the metadata is decided to be replicated, then the storage module advertises this replication by multicasting it to available peers through NB publish/subscribe mechanism. The storage module also interacts with the Sequencer module in order to label each incoming metadata with a time stamp.

Sequencer Module: This module ensures that an order is imposed on actions/events that take place in a session. The Sequencer Module interacts with the Storage Module and labels each metadata which will be replicated in this replicated metadata hosting environment. The Sequencer Module interacts with Network Time Protocol (NTP) clients to achieve synchronized timestamps among the distributed nodes.

When receiving a query, the Query Module first processes the query and extracts the dynamic metadata portion of the query. Then, the Query Module forwards the query to Expediter, where the Expeditor Module checks whether the requested data is in Context Spaces. If the Expeditor Module can not find the result in Context Space or if the requested metadata is expired, then the query is forwarded to the JDBC Handler to query the data in local database. If the query asks for external metadata, then the Expediter will forward the query to Access Module, where the Access Module multicast a probe message to available Information Services through NB and communicates with the Information Services that are the original data sources for this query. The query is responded by an

Information Service which may be the best qualified Information Service is to handle this query.

4. Conclusions and Future Work

In this paper, we have identified an important gap in Information Services for Grids that is lack of support for dynamic information in dynamically assembled Semantic Grids. We have presented an architecture that addresses key issues of managing dynamic metadata such as a) providing an efficient metadata access and storage methodology by taking into account changes in user demands and b) providing a P2P approach for access/storage request distribution among the peers of the system to capture the dynamic behavior both in metadata and the network topology. We have discussed status of our implementation and report initial performance results from a prototype that is applied to sensor and collaboration grids.

Work remains to further develop a distributed metadata hosting environment by employing novel dynamic replication techniques and to evaluate the system as whole through extensive performance tests.

Acknowledgement: ***This work is supported by the Advanced Information Systems Technology Program of NASA's Earth-Sun System Technology Office.***

5. References

[1] Galip Aydin, Mehmet S. Aktas, Geoffrey C. Fox, Harshawardhan Gadgil, Marlon Pierce, Ahmet Sayar. SERVOGrid Complexity Computational Environments (CCE) Integrated Performance Analysis, Grid2005, Seattle

[2] H. Zhuge. Semantic Grid: Scientific Issues, Infrastructure, and Methodology, Communications of the ACM. 48 (4) (2005)117-119.

[3] B. Plale, P. Dinda, and G. Von Laszewski. Key Concepts and Services of a Grid Information Service. In Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems (PDCS 2002), 2002.

[4] Monitoring & Discovery System (MDS4) Web Site is available at http://www.globus.org-/toolkit/mds

[5] A. Cooke, A.Gray, L. Ma, W. Nutt, J. Magowan, P. Taylor, R. Byrom, L. Field, S. Hicks, and J. Leake. R-GMA: An Information Integration System for Grid Monitoring. Proceedings of the 11th International Conference on Cooperative Information Systems, 2003.

[6] Ratnasamy, Sylvia et al. A Scalable Content-Addressable Network. Proc. ACM SIGCOMM, pp 161-172, August 2001.

[7] Serafeim Zanikolas and Rizos Sakellariou. A Taxonomy of Grid Monitoring Systems. Future Generation Computer Systems, 21(1), January 2005, pp. 163--188.

[8] Sivansubramanian S., Szymaniak M., Pierre G., Steen M.V. Replication for Web Hosting Systems. ACM Computing Surveys. Vol. 6, No. 3, Sept. 2004, pp. 291-334.

[9] M. Rabinovich. Issues in Web Content Replication. Bulleting of the IEEE Computer Society Technical Committee on Data Engineering, 1998.

[10] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally distributed content delivery. IEEE Internet Computing 6, 5 (Sept.), 50-58. 2002

[11] M. Rabinovich, I. Rabinovich, R. Rajaraman, and A. Aggarwal. A Dynamic Object Replication and Migration Protocol for an Internet Hosting Service. Proc. 19th Int'l Conf. Dist. Computing Systems, pp. 101-113, June 1999.

[12] P. Rodriguez, and S. Sibal. SPREAD: Scalable Platform for Reliable and Efficient Automated Distribution Computer Networks, vol. 33, nos. 1-6, pp. 33-49, June 2000.

[13] Bunting, B., Chapman, M., Hurley, O., Little M,, Mischinkinky, 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

[14] Bellwood, T., Clement, L., and von Riegen, C. UDDI Version 3.0.1: UDDI Spec Technical Committee Specification. Available from http://uddi.org/pubs/uddi-v3.0.1-20031014.htm.

[15] Mehmet S. Aktas, Galip Aydin, Geoffrey C. Fox, Harshawardhan Gadgil, Marlon Pierce, Ahmet Sayar, Information Services for Grid/Web Service Oriented Architecture (SOA) Based Geospatial Applications, Technical Report, June, 2005

[16] Shrideep Pallickara and Geoffrey Fox NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids in Proceedings of ACM/IFIP/USENIX International Middleware Conference Middleware-2003, Rio Janeiro, Brazil June 2003. See also: http://www.naradabrokering.org

[17] Wenjun Wu, Geoffrey Fox, Hasan Bulut, Ahmet Uyar, Harun Altay "Design and Implementation of A Collaboration Web-services system", Journal of Neural, Parallel & Scientific Computations (NPSC), Volume 12, 2004.