An Architecture for Supporting Information in 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. We report initial results from a prototype that is applied to sensor and collaboration grids.

1 Introduction

Semantic Grid [1-2] is an approach to Grid where computing resources, services and data can be expressed in standardized ways which can be understood and processed by Grid applications. This way, resources and services can be discovered, linked together and plug into appropriate data sources in an automatic fashion. Major Grid Web Service families are identified as Data, Execution, Desktop, Information and Collaboration Grids in [3]. Each and every style of Grid Web Service family can also be identified as Semantic Grid if they built on the important W3C Semantic Web [4-5] initiative. By analogy with categorization described in [3], we can identify Semantic Grids as Semantic Data, Semantic Execution, Semantic Desktop, Semantic Information and Semantic Collaboration Grids. From all of these parts of identified Semantic Grid Web Service families, comprehensive science applications can be built. Members of these different types of Semantic Grids may be united into a collection of Semantic Grid services that are needed to build application Grid systems such as myGrid [6], the Collaboratory for Multi-scale Chemical Science (CMCS) [7], Scientific Annotation Middleware (SAM) [8] and Tupelo [9]. This unification can be called as “Semantic Grid of Semantic Grids”.

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 an earthquake [10]. 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 Web/Grid Services.

Handling and discovery of dynamic information requires high performance, fault tolerant information systems. These information systems better be decentralized, relocate metadata to nearby locations of interested entities and provide efficient access, storage of the information, as the dynamic metadata needs to be delivered on tight time constraints within a Gaggle. Information Services (IS) support discovery and handling of services through metadata and are vital components of Grids [11].

We identify following problems in Information Services supporting both Classical and Semantic Grids. First, Grid Information Services need to be able to support dynamically assembled services. Classic Grid Information Services [12-13] however are not built along this model. Second, Information Services should scale in numbers and geographical area and be tolerant to failures while providing high performance in serving the requests. Most existing solutions [16] however have centralized components and do not address scalability, fault tolerance and performance issues. Third, Information Services need to be able to take into account user demand changes when making decisions on metadata access and storage. Classical Grid Information Services [16] however store the metadata on pre-defined locations and ignore changing user demands. We therefore see this as an important area of investigation. This paper presents our design of an architecture and prototype 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.

Our architecture intends to meet the following requirements: a) Providing a hybrid Information Service supporting both static, relatively slowly varying information and dynamic, rapidly updated information. b) Providing a dynamic metadata hosting environment where metadata can be relocated based on changing user demands. c) Maintaining information regarding sessions and also state of entities in these sessions d) Enabling discovery of participant entities within a session. e) Enabling dynamic discovery of data-systems hosting the metadata in consideration. f) Enabling discovery, retrieval and reconstruction of any state that might need to be associated with a failed entity in a session.

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. We discuss the status and the evaluation of our prototype in Section 4. In Section 5, 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 (ad-hoc). a) 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) [12] 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 [14-15]. b) 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) [13] 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 [16]. Architectures with pure decentralized storage models have focused on the concept of distributed hash tables (DHT) [14-15]. 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, DHT approach provides good performance on routing messages to corresponding nodes, it has various limitations such as primitive query capabilities. Here, we focus on management of dynamically generated and widely-scattered metadata. 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. Unlike DHT approach, our architecture takes into account user demand changes when providing metadata access and storage.

Well-defined descriptions of resources, services and data constitute metadata. Metadata can be represented using varying metadata models such as XML Schemas or Semantic Web [4] 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 the distributed computing problems of managing metadata in the Semantic Grid, not the “semantic” part.

We use replication, a well-known and commonly used technique to improve the quality of metadata hosting environment of our architecture. Sivasubramanian et al. [17] gives an extensive survey on research efforts on designing and developing World

Wide Web replica hosting environments, so does Robinovich in [18] 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. a) 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 [19-20]. 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. b) 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 [20-21]. For instance, Radar [20] replicates/migrates dynamic content based on changing client demands. Spread [21] considers the path between the data-system and client and makes decisions to replicate dynamic content on that path. c) 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, Akamai project [19] 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 [20] applies primary-copy approach where an update can be done only on the primary-copy of the data. Our architecture mainly differs from these systems in the following points. First, the intended use of our architecture is not to be a web-scale hosting environment. The scale of the system that we are looking at is in the order of thousand entities participating in a session in which these entities dynamically generate metadata. 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 designed a novel architecture of an Information Service 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 Web/Grid Service standards, we based the interface of our system on the WS-Context [22] and Universal Description, Discovery and Integration (UDDI) Specifications [23] from OASIS (http://www.oasis-open.org). We extend both specifications to provide advanced capabilities and fulfill aforementioned requirements of our system.

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. 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. The intended scale for our design is in the order of thousand entities that are participating in a session in e-Science Classical or Semantic Grids. We discuss various issues in building a dynamic metadata hosting environment in the following section.

3.1. Fault Tolerant High Performance Information Services

There are various issues in a data hosting environment that need to be answered. One, for instance, is fault tolerance and another is high performance. We use replication technique to provide fault tolerance and high performance which improves the quality of our data hosting environment. As the data-systems in a data hosting environment can fail, replication technique can provide the corrupted data by switching into one of the remaining replicas which in turn provides fault tolerance. 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 [18]. We address two important issues of dynamic replication such as request distribution and replica placement in the following sections.

3.2. Access Algorithm

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 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 a 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, 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.

3.3. 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 is in high demand which is above a pre-defined threshold, then the metadata is replicated. If a metadata is in low demand which is below a pre-defined threshold, it will be deleted. To replicate a 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, 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 stored based on changing client demands.

3.4. 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 even know the location and identities of receivers. It publishes a message to a topic to which all nodes subscribe. We use NaradaBrokering (NB) [25] publish/subscribe mechanism as a communication middleware for message exchanges between peers. NB is a free, open source, software which may be thought of a as topic-based publish/subscribe messaging system: interested entities can register to a NB node to send and receive messages on particular topics.

3.5. 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.

3.5.1. Context Query and Publishing Modules: These modules present a uniform Web/Grid Service interface for publishing/discovering both static and dynamic meta-

data. We use two Web Service standards UDDI and WS-Context to be compatible with existing service standards. When a client posts a query, the query is processed and separated into two as dynamic and static queries. The dynamic query is passed to Expediter Module, where the cache is queried for requested metadata. The static query however is posed on the external UDDI Service. On receiving the results from both ends, the Query Module forwards the combined results to the client.

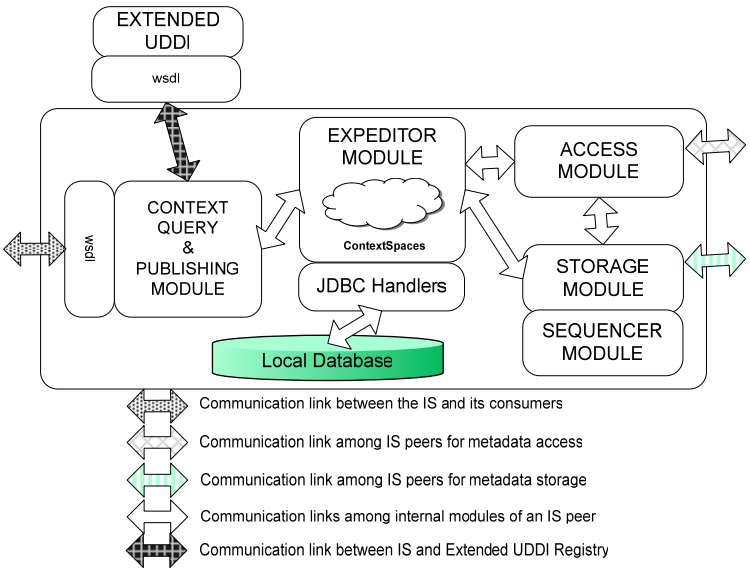

Fig.1. Architecture of an Information Service running on each peer

3.5.2. 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 Expeditor forms a built-in memory and it maintains metadata objects in Context Spaces. We term our implementation of Tuple Spaces concept [26] as Context Spaces. Context Spaces allow us to apply space based programming to provide mutual exclusive access and associative lookup.

3.5.3. Access Module: This module runs the aforementioned access algorithm. It supports request distribution by publishing messages to topics in NB. It also receives messages (in respond to client request) coming from other peers and forward these query messages to Expediter Module.

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

3.5.5. Sequencer Module: This module ensures that an order is imposed on actions/events that take place in a session. The Sequencer Module interacts with Storage Module and labels each metadata. This module interacts with Network Time Protocol (NTP) clients to achieve synchronized timestamps among the distributed nodes. This is to ensure that the replicated datasets are consistent with each other, while ensuring that the ACID properties are satisfied. We discuss an example scenario on how these components interact with each other in the following section.

3.6. Example Scenario

When receiving a query, 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 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 those 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 System Status and Evaluation

We implemented Information/UDDI Services handling and discovery of static metadata based on the WS-I standard UDDI Service Specifications [24]. Our implementation is a general purpose extension to the UDDI information model that allows us to insert both user-defined and arbitrary XML metadata into the repository. Here, XML metadata may be searched using XPATH queries, a standard way for searching XML documents (http://www.w3.org/TR/xpath).

We also implemented a centralized version of Information/WS-Context Services handling and discovery of dynamic, session related metadata. Here, session related metadata is short-lived and dependent on the client. We extended existing WS-Context Specifications to provide advanced capabilities to manage session metadata between multiple participants in Web Service interactions. Both UDDI and WS-Context implementation of Information Services have been successfully applied to sensor and collaboration grids applications.

We have done preliminary testing on the centralized version of the Information Service's primary operations which are GetContext and SetContext [10]. Three measurement sets were made using a 50 byte string for GetContext. Each of the three sets consisted of 100 individual measurements. We also performed 3 sets of 100 meas-

urements on the SetContext method. In average, we measure ~116 ms for GetContext and ~125 ms for SetContext functions to be performed. Both of these measurements are internal timings to process requests. We note that these were subject to very large variations. We conclude from this that the actual internal processing time for small metadata pieces is typically smaller than the network invocation time and does not create an actual overhead.

5 Conclusions and Future Work

In this paper, we identified an important gap in Information Services that is lack of support for dynamic information in dynamically assembled Semantic Grids. We have presented an architecture that addresses key issues of managing distributed 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 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.

References

1. Semantic Grid Community Portal, http://www.semanticgrid.org
2. David De Roure, Nicholas Jennings and Nigel Shadbolt, The Semantic Grid: A Future e-Science Infrastructure, Chapter 17 of Grid Computing: Making the Global Infrastructure a Reality edited by Fran Berman, Geoffrey Fox and Tony Hey, Jon Wiley & Sons, Chichester, England, ISBN 04-470-85319-0, March 2003. http://www.grid2002.org
3. Geoffrey Fox, Shrideep Pallickara, and Marlon Pierce, Building a Grid of Grids: Messaging Substrates and Information Management to appear as chapter in book "Grid Computational Methods" Edited by M.P. Bekakos, G.A. Gravvanis and H.R. Arabnia
4. W3C Semantic Web Site, http://www.w3.org/2001/sw/
5. IEEE Intelligent Systems March/April 2001 pages 24-79, Semantic Web Issue
6. R. Stevens, A. Robinson, and C.A. Goble, myGrid: Personalised Bioinformatics on the Information Grid, in Proceedings of 11th International Conference on Intelligent Systems in Molecular Biology, 29th June–3rd July 2003, Brisbane, Australia, published Bioinformatics Vol. 19 Suppl. 1 2003, i302-i304, web site: http://www.mygrid.org.uk

7. James D. Myers, et al, A Collaborative Informatics Infrastructure for Multi-scale Science, Proceedings of the Challenges of Large Applications in Distributed Environments (CLADE) Workshop, June 7, 2004, Honolulu, HI, p 24-33.
8. Scientific Annotation Middleware (SAM) Project, available from, http://collaboratory.emsl.pnl.gov/docs/collab/sam/
9. Tupelo Project, available from http://dlt.ncsa.uiuc.edu/wiki/index.php
10. Galip Aydin, Mehmet S. Aktas, Geoffrey C. Fox, Harshawardhan Gadgil, Marlon Pierce, Ahmet Sayar, SERVOGrid Complexity Computational Environments(CCE) Integrated Performance Analysis, Accepted as poster and short paper in Grid2005, Seattle, USA
11. 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
12. Globus Toolkit 4 - Information Services: Monitoring & Discovery System (MDS4), available from http://www.globus.org/toolkit/mds/
13. 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.
14. Ratnasamy, Sylvia et al., A Scalable Content-Addressable Network, Proc. ACM SIGCOMM, pp 161-172, August 2001
15. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan, Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. IEEE/ACM Trans. on Networking, 11 (1): 17-32, February 2003
16. Serafeim Zanikolas and Rizos Sakellariou., A Taxonomy of Grid Monitoring Systems., Future Generation Computer Systems, 21(1), January 2005, pp. 163--188.
17. Sivansubramanian S., Szymaniak M., Pierre G., Steen M.V., Replication for Web Hosting Systems, ACM Computing Surveys, Vol. 6, No. 3, September 2004, pp. 291-334.
18. M. Rabinovich, Issues in Web Content Replication, Bulleting of the IEEE Computer Society Technical Committee on Data Engineering, 1998
19. 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
20. 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. Distributed Computing Systems, pp. 101-113, June 1999.
21. 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.
22. 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
23. 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.
24. 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 available from http://grids.ucs.indiana.edu/ptliupages/publications/UDDI_Aktas_Final_Fix.pdf
25. 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 the NaradaBrokering Web Site: http://www.naradabrokering.org
26. N. Carriero and D. Gelernter., Linda in Context. Commun. ACM, 32(4): 444-458, 1989.