Managing Dynamic Metadata as Context in P2P/Grid Environments

Mehmet S. Aktas, Geoffrey Fox, and Marlon Pierce
Community Grids Lab, Indiana University
{maktas, gcf, mpierce }@cs.indiana.edu

Abstract— **As the amount of metadata describing services and their access frequencies increased in P2P/Grid Computing Environments, there is an emerging need for Information Services to make such metadata available. The metadata describing services might include frequently changing information, so it presents a dynamic behavior. We use context as metadata to capture such frequently changing dynamic information. Here, context information can be used not only for discovering services, but also by services after they have been discovered. In this paper, we are interested in providing and managing context information parts of which is likely to be changed very frequently. We describe our initial efforts in building Fault Tolerant and High Performance Information Systems (FTHPIS) to make both dynamic and static context information available.**

1. INTRODUCTION

The Computational Grid introduces large amount of services managed by different organizations or individuals to let users utilize distributed computing resources, applications and data. Peer to Peer computing also provides services where researchers package their own resources as services to offer others in their community. As the services may come and go frequently over time and these services have complex characteristic (some of which may be dynamically generated such as operational state of services), metadata pieces describing these services have very dynamic behavior. Such metadata can also be describing the interaction between two or set of services. So, services in both environments are rich in their context and they have not only static metadata but also dynamic metadata describing themselves.

For the purposes of our research, context is a piece of information describing the activities and characteristics of a resource in particular a service. Here, we use resource in general sense of World Wide Web Consortium (W3C) as used in Resource Description Framework (RDF).

Context encapsulates not only activities of a service but also the characteristics of service itself as an entity. We broadly classify context information as static context metadata and dynamic context metadata. Dynamic context information is the session metadata generated by one or more services as a result of their interactions. Static context information is rather static information describing the service characteristics such as location of the service. This type of metadata is independent of any interaction.

We find context information very valuable for discovering and managing services. Here, the context metadata is not only used for discovering services, but also used by services after they have been discovered. However context information is either not available to consumer of services or it does not capture dynamic behavior. Locating dynamic metadata of interest is a fundamental problem in P2P/Grid environments. An effective methodology to facilitate metadata discovery is to provide and manage metadata.

Another motivation for management of dynamic metadata can be summarized as follows. Grids are collection of services where services are put together for particular functionality, such as visualization, sensor, and collaboration grid. Though, SOAP and WSDL technologies define how to interact with individual services, there is no available technology to lead users to interact with collection of services.

The scalability of information systems forms another key research issue that needs to be investigated in our research. For instance, storage of context metadata requires scalability not only in performance but also in numbers. To this end, we see a greater need for scalable information systems to make dynamic and static context metadata of services available in peer-to-peer/grid environments. Such information systems should also allow users to expose and interact with collection of Grid services.

In this research, we are investigating how to link peer-to-peer and centralized metadata management strategies which will then provide an effective solution to service discovery problem in peer-to-peer/grid environments. We research database models with messaging where metadata is captured in messages. We also research ways of providing both high performance by melding caching with database approaches and fault tolerance by replicating databases.

The organization of this paper is as follows. First we discuss relevant work. Then, we discuss various application scenarios to present the scope of this research. Next, we discuss our architecture in building Information Systems followed by conclusions.

2. RELEVANT WORK

Existing service discovery architectures can be broadly categorized as centralized and decentralized by the way they handle with service information storage. In centralized approach there is a central look-up mechanism where all services are dependent on one node. Example mainstream projects of centralized discovery can be summarized as following; JINI [3], Salutation and Salutation-lite [1], Service Location Protocol [4]. In decentralized discovery, there is no central database. Decentralized discovery architectures can be categorized further by the manner of in which decentralization is realized such as hierarchical, structured peer-to-peer and unstructured peer-to-peer (ad-hoc). In hierarchical discovery architectures, service information is stored in distributed databases where databases are organized based on a hierarchy. In structured P2P discovery architectures, nodes in the systems are equally enabled and controlled and service information is disseminated to all nodes. CAN [10] and Chord [11] are two well known examples of structured P2P networks. In unstructured P2P service discovery architecture, there is complete lack of control on the capabilities of the network nodes. Example projects of this architecture are Konark [7] and DeapSpace [6]. A good discussion on P2P search models can be found in [9].

The centralized storage scales better in performance for limited storage capability compared to decentralized approach, whereas decentralized approach can scale up to high amount of metadata where centralized approach fails. Pure decentralized storage models such as P2P service discovery architectures have focused on the concept of distributed hash tables (DHT). This approach assumes the 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. This method may provide better performance as the database operation messages are routed fast, however, it still does not provide the same performance to handle dynamic context metadata as centralized database does.

In our design, we are building information systems that scale both in performance and in numbers that the storage can handle. So we link both centralized and peer-to-peer storage strategies to provide a single storage for context metadata.

We can also investigate service discovery projects by the methodology they use for service descriptions and service matchmaking process. In existing service discovery projects, service description can be keyword-based (UDDI [14], Corba Naming [6]), unique identifiers-based (Blootooth [5]), interface-based (JINI [3]), XML based (UPnP [2]), attribute-based matching (Salutation and Salutation-lite [1], Service Location Protocol [4]) or ontology based (myGrid Service Discovery [15]).

In our design, we associate a life-time and context metadata with each service description in the database. We adopt attribute name/value pairs to describe context metadata in databases.

Another way of classifying service discovery architectures could be based on the formation of the network. In traditional wired networks, network formation is systematic since each node joining the system assigned an identity by another device in the system. These traditional service discovery protocols focused on LAN services provided by devices such as Printer, Fax Machines. As devices join or leave to the network infrequently, the network presents a uniform structure. Example wired network discovery architectures could be JINI [3], Service Location Protocol [5] and UPnP [2]. In unstructured P2P (ad-hoc networks), there is no controlling entity and there is no constraint on the resource dissemination in the network. Research in Peer-to-Peer service discovery is fairly new. Examples projects could be Konark [8] and DeapSpace [7].

Our design encapsulates both wired and wireless (peer to peer) network services; we provide an application level communication infrastructure which assumes an IP level connectivity between the entities of the system.

3. APPLICATION CASE SCENARIOS

In order to present the scope of this research we outline following application case scenarios. The first case scenario illustrates a large scale grid computing environment where grid data services facilitate query capabilities over large data sets. The second scenario illustrates a Peer to Peer environment where a videoconferencing takes place.

In the first scenario, a PC user running a geographical application which process GPS and Seismic data to do various simulations. Both data sources are available online through Geographical Information System (GIS) enabled data services. These data services might provide different data and data formats with varying spatial coverage. There is no programmatic way of working with the remote services, for example, choosing the services providing GIS data based on the user's criteria, and then assembling data from these data services to create maps in a mapping service supporting the data types. Instead, the user typically downloads the capability files of these data/mapping services to find out if they are satisfying his/her needs. This scenario is example of the general problem of managing information about services. The user must be able to query information services that provide a metadata catalog for GIS data service capabilities, which enables the user to connect to the desired service.

The second scenario describes a Peer to Peer application. A user has a laptop running a Videoconferencing application which is also a web service. The service consists of different collaboration systems such as conferencing (H.323, SIP, and AccessGrid), streaming, and instant messaging services. The user wants to start a real-time interaction environment which will allow him/her to make a distant seminar about his/her research. If such real-time interaction can take place, then the user can start a scientific collaboration through multiple collaboration tools such as audio, video, chat, whiteboard and PowerPoint as well. With peer to peer services moving around and their volatile behavior, providing such real-time interaction forms a challenging problem. To this end, there is a need for Information System to make the information about temporarily exist services available.

4. ARCHITECTURE

In this section, we discuss possible strategies that we investigate to build a fault tolerant and high performance information system (FTHPIS).

4.1. REQUIREMENTS

We outline the requirements of our domain and architecture as follows.

Availability and Fault Tolerance: The system should not have central point of failure. Each database needs to be replicated in order to provide context information available for users all the time. When making this information available, the system should take into account component failures such as node failures.

High Performance: The system should be enabled to provide low response times. An obvious solution for this is caching (query, response) pairs.

Consistency: As the accuracy of context information is very important, the system should be enabled to provide data coherency in all replicas.

Scalability: The scale of the system should be around 100-1000 users per session. We place no limits on the total number of sessions or on the number of sessions that a node can be a part of.

4.2. OVERVIEW OF THE ARCHITECTURE

FTHPIS can be considered as a node cloud gathered to make information about services available for users.

Each participant node has the capability to announce its local services and discover remote target services by sending out announcement / discovery messages to others. A target service is the service that wants to be discoverable through the Information System.

There are two main entities in our system such as information services and data-systems. Each participant node of the FTHPIS implements an information service. An information service is the endpoint that provides uniform interface to the two types of context metadata such as static and dynamic context metadata. There are also data-systems (databases) present in this system. A data-system is the entity that facilitates context discovery among large number of contexts. We will use the term data-system with database interchangeably for the rest of this paper.

An important design issue is how to make information about services available to others in the network. In our design, Information Services are interacting with each other using P2P communications. There are various options to provide communication between nodes of the system such as structured, unstructured P2P discovery models and classic middleware approaches where the discovery happens within the distributed server/broker network.

In our solution we link peer-to-peer and centralized metadata storage strategies to provide storage where there is both good performance and scalability. To this end, we use a classic middleware approach to facilitate discovery using XPath or SQL queries on the distributed databases within the distributed server/broker network. So, we use a publish\subscribe based

SOAP Handler Environment (SHE) (a software-based brokering system) to provide communication between nodes.

In our prototype implementation we are using NaradaBrokering project (http://www.narada-brokering.org) as a particular implementation of SHE. This methodology allows us to provide an application level communication infrastructure which is independent from low level transport protocols. In this scenario, messages (advertisement / discovery) are broadcast to all databases through brokering system. Each message includes a unique identifier identifying the peer initiated the request. On receiving the message, only databases that have the requested information reply with a respond message directly to the initiator of the query.

In order to facilitate information discovery, we provide and maintain context in databases. We separate context as static and dynamic context information. Static context information describes both functional and non-functional characteristics of services, whereas dynamic context information describes activities or sessions that a service is involved. The architecture places no constraints on the types of storages; it could be based on flat files, relational or XML databases.

In our prototype implementation of our design, we choose to implement the Universal Description, Discovery, and Integration (UDDI), since UDDI is a Web Service Interoperability (WS-I) standard. UDDI provides a standardized method for publishing and discovering information about Web Services. In UDDI, services can be discovered by name, by location, by business or by tModels (service types). UDDI has some limitations in describing service descriptions in an expressive way. Also, UDDI discovery relies on a keyword-based retrieval approach where the service discovery is based on keyword-matches between the service query and service descriptions. To this end, we extend UDDI Specifications in order to associate context metadata and lifetime with service descriptions where context metadata of a service consists of set of service attributes where each attribute has (name, value) pairs. Similar methodology has also been used in various projects as in [12], [13].

Service matchmaking process is a retrieval process that finds results by matching a service request with service descriptions. This research has been definitely investigated as in [13], [15], [16] and so not covered in our design. We view distributed system aspects of our architecture as higher priority.

In order to meet the requirements that we outlined earlier, we design following services available for the nodes of FTHPIS. These services are Search, Discovery, Expediter, Storage, Sequencer and Load Balancer Services.

Search Service: This service is implemented by all information services. It provides interfaces for various search capabilities. In our prototype implementation, we extend UDDI Inquiry Service Interface to provide search interface to our extended version of UDDI. This way, we are able to pose metadata oriented queries on the databases. Service metadata might also have associated metadata catalog (auxiliary metadata file) for data services. To this end, we also provide Xpath query abilities where users can pose queries on the metadata catalog to find the data services of interests. In providing such search capability, we investigate 2-phase discovery schemes as following. In this scheme, a node first discovers the dataset that can provide the information that it is looking for and then proceed to issue a refined search (Xpath query on the metadata catalog) to locate precisely the data that it is looking for.

Storage Service: This service is only implemented by data-systems. It implements a storage and access interface to a data-system. In our prototype implementation, we extend UDDI Publishing Service interface to provide access to our version of UDDI database. The Storage service supports replication of databases to provide low response times and high availability. In our problem, static context in databases are likely to be modified infrequently, whereas dynamic context is likely to be modified frequently. Also, the access rate for read request can be very high. As the service requestors might choose services based on frequently changed metadata, all database replicas have to provide fresh and consistent metadata about services. So, the replication service should be able to handle high loads of modifications to provide information consistency.

Discovery Service: This service is implemented by all information services and data-systems. It provides P2P communication among the nodes and discovery of entities and data-systems in the information system network.

Expediter Service: This service is implemented by all nodes in order to improve response times and avoid performance loss caused by repetitive queries. An Expediter service provides a generalized caching mechanism. Each cache entity consists of (request, response) message pairs. A cache is considered empty at boot-strap of a node and it gets filled with (request, response) message pairs as the database is queried by the users. We also investigate various cache entity replacement policies such as Least Recently Used (LRU) to improve response times.

Sequencer Service: This service is needed by datasets supporting multiple clients updating a single dataset. The idea is simple, which is to label each message in the system. This ensures that an order is imposed on actions/events that take place in a session. Furthermore, this sequencer will also play a role in ensuring that the replicated datasets are consistent with each other, while ensuring that ACID properties are satisfied.

Load Balancer of storage: Databases have limited storage and memory capabilities. So, service metadata should be evenly spread out into the distributed databases in the system. This service is used to decide how to implement distributed storage of a single dataset. Here a given dataset may itself be spread over multiple locations. A bit torrent example of such a scenario is where fragments of a file may reside at multiple locations.

5. CONCLUSIONS

We have discussed various issues related building an information system which would provide distributed context metadata management. We gave a brief classification of relevant work. We explained application case scenarios on how such system will be useful. Starting from these scenarios, we discussed the requirements of the system. Then, we presented a general overview of our design on implementing information systems. In this research, the expected contributions can be summarized as following. We will identify a novel process for building P2P/Grid Fault Tolerant and High Performance Information Systems and the requirements to provide dynamic and decentralized context management in P2P/Grid Environment. We will identify the requirement of replicating highly dynamic context information databases. We will build a collection of services which will then form an Information System for P2P/Grid Environment. In current prototype of the FTHPIS, we completed implementation of databases. We have also implemented search and publication service interfaces to the databases.

REFERENCES

[1] The Salutation Consortium Inc 1999. Salutation architecture specification (part 1), version 2.1 edition. World Wide Web, http://www.salutation.org

[2] Rekesh John, UPnP, Jini and Salutation – A look at some popular coordination framework for future network devices, Technical Report, California Software Labs. 1999.

[3] Ken Arnold, Ann Wollrath, Byran O'Sullivan, Robert Scheifler, and Jim Waldo, The JINI Specification, Addison-Wesley, Reading, MA, 1999

[4] E. Guttman, C. Perkins, and J. Veizades., Service Location Protocol, RFC 2165.

[5] Blootooth SIG. Specification. http://bluetooth.com

[6] Object Management Group, "Catalog of Corba/IIOP specifications", http://www.omg.org/technology/documents/corbaservices_spec_catalog.htm

[7] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, A. Schade, DEAPspace--Transient ad hoc networking of pervasive devices, Computer Networks Volume 35 pp 411-428, 2001

[8] S. Helal, N. Desai, V. Verma and C. Lee, Konark--A Service Discovery and Delivery Protocol for Ad-hoc Networks, Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, March 2003.

[9] Fletcher, George , Sheth, Hardik and Börner, Katy. (2004). Unstructured Peer-to-Peer Networks: Topological Properties and Search Performance. Third International Joint Conference on Autonomous Agents and MUlti-Agent Systems. W6: Agents and Peer-to-Peer Computing, Moro, Gianluca, Bergmanschi, Sonia and Aberer, Karl, Eds., New York, July 19-23, pp. 2-13.

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

[11] 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

[12] UDDIe: An Extended Registry for Web Services
Ali ShaikhAli, Omer Rana, Rashid Al-Ali and David W. Walker. Proceedings of the Service Oriented Computing: Models, Architectures and Applications, SAINT-2003 IEEE Computer Society Press. Oralndo Florida, USA, January 2003

[13] Personalized Grid Service Discovery. Miles, S., Papay, J., Dialani, V., Luck, M., Decker, K., Payne, T., and Moreau, L. Nineteenth Annual UK Performance Engineering Workshop (UKPEW'03), University of Warwick, Conventry, England, 2003.

[14] 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.

[15] D. Chakraborty, F. Perich, S. Avancha, and A. Joshi, *DReggie: A Smart Service Discovery Technique for E-Commerce Applications*, In Workshop in conjunction with 20th Symposium on Reliable Distributed Systems, October 2001.

[16] Paolucci, M., Kawamura, T., Payne, T. R. and Sycara, K., *Importing the Semantic Web in UDDI.* (2002). In Proceedings of Web Services, E-Business and Semantic Web Workshop, CAiSE 2002. , pages pp. 225-236, Toronto, Canada.