Fourth International Conference on Semantics, Knowledge and Grid

A Dynamic Ontology Mapping Architecture for a Grid Database System

Nelson C.N. Chu, Quang M. Trinh, Ken E. Barker and Reda S. Alhajj

Department of Computer Science, University of Calgary
2500 University Dr. NW, Calgary, AB, Canada
{chuncn,qtrinh barker,alhajj}@cpsc.ucalgary.ca

***Abstract*—** Most large-scale heterogeneous distributed computing systems, such as Grids, rely on Service Oriented Architectures (SOA) to interact with others in different platforms and computing languages. However, we still need to solve the semantic heterogeneity problem of data; we must interpret the data from different systems in some semantically related ways. Ontologies are the most common and well-accepted methodology to handle this problem at multiple levels of granularities across different systems. Nevertheless, using ontologies in a dynamic environment, such as a Grid, to share some common concepts is still a challenge. It is difficult to keep a static mapping between ontologies; the corresponding semantic mapping changes must occur consistently. Therefore, we adopt the concept of Tuple Space and propose a flexible approach for managing ontologies in a Grid. It enables systems and users to interoperate semantically and dynamically by sharing and managing the concepts and semantic ontology mappings in a flexible approach.

I. INTRODUCTION

Interoperability between data resources or systems in large-scale distributed environments such as the Grid must be addressed dynamically for the following three reasons: (i) a large number of heterogeneous participant systems; (ii) resources are designed and built independently; and (iii) interactions and participation between the resources are dynamic. It implies dynamic resources existing in the system are irregular and unpredictable; they could be holding very interesting or important information about physical resources, such as data storage. One of the most challenge problems is to handle semantic heterogeneity of concepts between autonomous participants in a Grid. In the past, existing solutions for addressing the interoperability problem between systems are mainly focused on a small number of systems that are in static environments so these solutions are not appropriate for systems in large-scale environments. In large scale environments, a compromise is very difficult to achieve between systems because of a large number of participant systems with different requirements that are designed and built independently. For example, semantic heterogeneity is one such problem. Semantic heterogeneity is a very difficult problem to address dynamically in large-scale environments; however, ontologies are keys success factors for addressing this problem. They are used to describe the descriptions, meanings, and the semantic relationships of the data in each system. Ontology provides effective methods for mapping concepts and integrating information in a Grid. Furthermore, it reduces the problems caused by the multiple levels of concept granularity and hence increases the degree of information interoperability and collaboration.

Ontology management in large-scale environments is a challenge for two reasons: (i) ontologies will evolve so changes in semantic relationships with other ontologies must be consistent. Ideally, the changes should occur concurrently and consistently; (ii) dynamic ontology participation inherited from the participation of systems in a Grid are dynamic, and ontology participation in a Grid is as well. This requires the ontology management approach be flexible enough to accommodate the dynamic participation of ontologies while maintaining their scalability, reliability, availability, and versioning requirements [1]. This would allow systems in a Grid to interoperate with each other semantically and dynamically.

Ontology is a well-recognized structure for integrating information, but managing different ontologies from different data resources becomes a major challenge in a dynamic environment. To provide a flexible ontology managing system the Tuple Space paradigm [3] is a suitable solution for the dynamic environment. This paradigm provides a simple but powerful mechanism for a member to interact with the system. Our approach applies a Tuple Space paradigm for managing ontologies and their semantic relationships while allowing them to evolve. Tuple Spaces is appropriate for managing ontologies and their semantic relationships in large-scale environments because [5]: (i) a Tuple Space's architecture provides fully autonomous and flexible communication between systems; (ii) Tuple Spaces can easily combine with a database system to provide

978-0-7695-3401-5/08 $25.00 © 2008 IEEE
DOI 10.1109/SKG.2008.49

transactional-database-like contexts that ensure the changes are concurrent and consistent.

The contributions of this paper are two fold: (i) proposes a Tuple Space based architecture to resolve the semantic problem of information, multiple levels of granularity, by using ontology in a Grid database system. (ii) It resolves dynamic participation issues of the ontologies in such system. The rest of the paper is organized as followed. Section 2 provides a summery of two key concepts in this research: Ontology and Tuple Space. We proposed an efficient and flexible architecture in Section 3 to manage ontologies in a Grid environment. An implementation of the proposed architecture is described in Section 5.

II. Related Work

A. Ontology

A multiple domain system is very difficult to maintain in a federated way. The reason is the differences between the policies of each domain. For each domain, they have their own guideline for authentication, authorization, access control, *etc*. To join a federation, all of these issues must agree on a set of rules that satisfies everyone's need. The main idea of a Grid is resources sharing with emphasize on interoperability and collaboration. To achieve this goal, finding a simple and flexible mechanism in a dynamic environment is the key milestone.

In general, an ontology is a document that contains the explicit and formal descriptions of concepts and the relationships that are defined and used in a single system or domain [4]. Three of the goals of ontologies are: (i) provide a high degree of abstraction so that concepts can be resolved at multiple levels of granularity; (ii) enabling common "understanding" of data provided by different systems between humans and across computer systems [6, 9]; and (iii) provide consistency checking of concept definitions defined in multiple systems [6]. Ontologies play an important role in database interoperability because they can bridge the "semantic gaps" between database systems so they provide database interoperability with capabilities beyond the thesauri and define an "entity-relationship model"-like description for complex and diverse data types in different systems.

B. Tuple Space

Tuple Space [3] is a shared memory system with tuples that are ordered-sets of typed constants, variables, and parameters, which are accessible to all processes in the system. The Tuple Space paradigm is suitable for Grids for several reasons. It is an asynchronous paradigm with no assumptions about timeliness, ordering, or synchronization [2]. The scalability of a space is easier to achieve because no strict order of tuple matching is required. The Space operations are fully anonymous communications (destination uncoupling) [9]. Tuple spaces decouple processes with respect to time and space. The life cycle of a tuple is independent (time uncoupling) of the tuple generation and consumption. Tuple Space is able to provide a globally shared data space to all processes, regardless of machine or platform boundaries (space uncoupling).

III. Dynamic Ontology Mapping Architecture

ONMAS is a Tuple Space based architecture for managing ontologies. The Tuple Space acts as a communication hub and, therefore, the negotiation of communication between consumers (users) and service providers can be eliminated. This approach simplifies the communication process and provides flexibility of participation of all participants. The dynamic participation exactly describes the situation in a Grid system. Some users or services providers constantly participate in the system and some other participants may exist in the system on an irregular basis. ONMAS provides a flexible management mechanism for an ontology based information integration system to handle this dynamic environment. On one hand, an information resource produces, stores, and maintains an ontology in the Space to describe its structure and service for authorized consumers. On other hand, a consumer can simply get the ontology that met its requirements from the Space. On top of the basic structure, the leasing structure is a promising mechanism implemented to permit dynamic participation and guarantees the stability and accuracy of the ontologies in the system.

The Tuple Space paradigm is suitable for Grids for several reasons. It is an asynchronous paradigm with no assumptions about timeliness, ordering, or synchronization [10, 11]. The scalability of a space is easier to achieve because no strict order of tuple matching is required. The Space operations are fully anonymous communications (destination uncoupling). The tuple producer does not require knowledge about the future usage and destination of a tuple. A tuple persists in the tuple space until it is consumed. This persistence property also enables tuple spaces to decouple processes with respect to time and space. The life cycle of a tuple is independent (time uncoupling) of the tuple generation, out operation, and consumption, in operation. Tuple Space is able to provide a globally shared data space to all processes, regardless of machine or platform boundaries (space uncoupling).

As described above, the interoperability of data resources or systems in a Grid has been greatly affected by the number of participant systems, independently designed and maintained resources, and dynamic interactions and participation in such system. The proposed architecture adapts ontology and Tuple Space concepts to resolve the semantic problem of information with multiple levels of granularity, and to handle dynamic participation of the resource providers and/or users in such environments. The rest of this section gives the details of the proposed architecture.

A. Elements in ONMAS

User: is a data consumer in the system.

Ontology tuple: provides meta-information and descriptions of the corresponding data resource. It also gives a mapping of subsequent data or concepts in the system. The mappings are implemented with a lease list mapping resources and services to requests.

Resource: responsible for generating and updating the ontology that describes the data resource. It maintains the services promised to the users listed in the lease list of the ontology.

Resource list(s) tuple: a *Register* collects data in the space and publishes a resources list, which provides information and statistic of available resources in the system. The list could be divided in different categories based on the nature of the resources to minimize unnecessary scanning of a single long list.

Tuple Space Register/Server (SR): is a Tuple Space manager providing system cooperatation. It is responsible for registering all other entities in the system, managing all leases between entities and the Space, supplying data for Grid Information Service, providing event notification service and transition service, and communicating with other domains in the Grid.

B. Entity Registration

Registration is a mandatory process for all entities which participate in the system. The registration allows the Register to update the resources list in the Space and activates other features, such as event notification. Every resource or ontology registration will be responded to with a new lease to indicate a time period defining the visibility of the entity.

C. Leasing Structure for Dynamic Participation

In a dynamic Grid environment, there are many factors affecting the availability of the resource, such as network connection failure, congestion, and hardware failure. The leasing mechanism in ONMAS eliminates this kind of uncertainty that affects the quality of services in the system by limiting the impact in the leasing period. The leasing mechanism sets the time contract between the service provider and consumer. During the time period, consumer can assumes the service or resource is available in the system. Any lease must be renewed before it expires, otherwise, the service or resource will be considered unavailable in the served domain. A lease will be issued for each resource registration or ontology entity. The lease between user and resource is an agreement to indicate the interest of a user to use the corresponding ontology, and the willingness of the resource to provide and maintain the information or services specified in the ontology. The lease is recorded in the lease list of an ontology. The lease between resource and space is a contract between the system and data resource; in that it designates the its existence and willingness to serve as a system resource.

IV. Managing Ontologies

In ONMAS, the basic operations are inherited from the original Tuple Space concepts [3]. The three basic operations (Read, Write, and Take) are implemented in two modes of the tuple accessing primitives: blocking and non-blocking. For example, a read operation in blocking mode waits until a matching tuple is found in the space, while a non-blocking operation returns empty tuple to indicate no matching tuple in the space.

A. Ontology Generation

Ontologies are generated dynamically by using the RDB2ONT tool [7]. The RDB2ONT tool uses the meta-data and structural constraints of the underlying relational data model to generate ontologies with the Web Ontology Language (OWL) [8] to describe the generated ontologies. Formally, relations are transformed into OWL classes, non-foreign key attributes are transformed to OWL data properties, and foreign key attributes are transformed into OWL object properties. The RDB2ONT tool preserves the structural constraints of the underlying relational model by transforming these structural constraints into OWL restrictions and enforces them on the classes, data properties, and object properties in the generated ontologies. The detail descriptions of the generation process in RDB2ONT can be found elsewhere [7].

B. Ontology Mapping

Ontology mappings are important since they relate semantically related resources from multiple data sources together. In general, mappings are not bidirectional, so for example, if one also wishes to say that the Staff from CompanyY is semantically the same as the Employee in CompanyX, then this mapping needs to be explicitly defined in the CompanyX ontology. In a

Grid, resources are dynamic so mappings between ontologies are also dynamic. It implies that mapping management between ontologies is necessary. Our framework uses a mapping manager called Mapper to keep track of the mappings between ontologies (see [7] for more details of how the Mapper works). A Mapper of an ontology issues a contract with an expiration time from itself to the Mapper of another ontology. The contract for the mapping between a pair of resources guarantees that the resource specified in the ontology will remain unchanged until the expiration time. Contracts can be renewed before their expiration times but it is up to the resource mapper to issue the renewals dependent on any existing contract.

C. Ontology Manipulation

Insert: An ontology will be registered with SR and hosted in the space for users to access. The Registrar keeps tracks all ontologies and maintains the resource list in the system.

Update: When updating the entire ontology must be unused so the resource waits for all leases listed in the lease list in the ontology entity to expire and then removes the ontology from the space. The new ontology can then be activated immediately. In a situation requiring updating attributes in an ontology, resources adds the new attribute into the ontology and marks the old attribute as dated. All users listed in the lease list are still able to access the dated attributes until the lease expires. Other users will obtain a lease based on the new set of attributes.

Delete: To terminate the service (a resource tuple) of the resource or the service of some information, a resource needs to wait until all related leases expire.

Search: The first scenario, the Space returns a tuple that match the template provided by the entity which issued the search. The second scenario, the search is caused by referencing the mapping of an ontology. The ontology engine also assists in mapping the corresponding definition of a concept.

V. Implementation

A scenario that uses an ontology for data integration has been created and an application of the proposed architecture has been developed to illustrate the whole process to manage ontologies for concept integration. We implemented the proposed system and simulate an inter-university network for sharing staff information. In this simulation, ontologies are used to describe the terminologies/concepts (such as {Staff, Faculty, Instructor} vs. {Employee, Teaching Staff, Lecturer}) that used in each university. The application also provides users the functionalities as discussed in pervious sections to manipulate (insert, update, delete, and search) the ontologies.

VI. Conclusion

There are two major contributions in this research. First, the proposed architecture resolve the semantic problem of information, multiple levels of granularity, by using ontology in a large distributed database system, particularly in Grid computing environment. In much scientific research, such as bioinformatics and physics, the nature of data is heavily depending on the linkage between each source but the amount of the data is huge. Our architecture provides an effective and efficient mechanism to manage ontologies and to integrate the data without losing their characteristics. Second, the proposed Tuple Space based architecture resolved dynamic participation of the resource providers and/or users in a Grid. It is an unavoidable problem and causes the availability of the services/resources to become unstable. Our system provides a flexible structure to guarantee the quality of services with dynamic participation of the participants. This is one of the most significant issues in such system with the concerns of quality of services.

References

[1] A. Das, W. Wu, and D. L. McGuinness. Industrial Strength Ontology Management. In I. Cruz, S. Decker, J. Euzenat, and D. L. McGuinness, editors, The Emerging Semantic Web. IOS Press, 2002.

[2] A. Friday, S. P. Wade, N. Davies, and G. S. Blair. The tuple space: An old solution to a new problem?

[3] D. Gelerner. Generative Communication in Linda. In ACM Transactions on Programming Languages and Systems 7, 1985

[4] T. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 1993.

[5] T. J. Lehman, S. W. McLaughry, and P. Wyckoff. T Spaces: The Next Wave. HICSS '99: Proceedings of the Thirty-second Annual Hawaii Intl' Conference on System Sciences-Volume 8. IEEE Computer Society, 1999.

[6] D. L. McGuinness. Ontologies Come of Age. In D. Fensel, J. A. Hendler, H. Lieberman, and W. Wahlster, editors, Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press, 2002.

[7] Q. Trinh, K. Barker, and R. Alhajj. RDB2ONT: A Tool for Generating Ontologies From Relational Database Systems. Technical report, University of Calgary, Sept 2004.

[8] W3C. Web Ontology Language (OWL), February 2005.

[9] P. Wyckoff, S. W. McLaughry, T. J. Lehman, and D. A. Ford. T Spaces. IBM Systems Journal, Nov 1998.