An Ontological Framework for Large-Scale Grid Resource Discovery

Juan Li, Son Vuong
Computer Science Department, University of British Columbia
{juanli, vuong}@cs.ubc.ca

Abstract

This paper presents GOIDS (Grid Ontological Integration and Discovery System), a comprehensive architecture for resource sharing and discovery in large-scale grids, where nodes integrate local ontologies to expand semantic knowledge of shared grid resources. A peer-to-peer based ontological directory service is used to facilitate the formation of the semantic small-world – ontological communities. Ontological communities help prune the searching space and reduce the cost of searching. Inside communities, resource knowledge is integrated between nodes, and a DHT overlay is used for knowledge dissemination and discovery. This framework architecture improves interoperability among grid participants and aids efficient resource discovery through an expressive query language.

1. Introduction

A large-scale grid enables the sharing of a wide variety of resources, including hardware components, software packages, knowledge information, devices, and other grid services. However, the resource discovery problem in large-scale grids is challenging because of the abundance of resources and the dynamic, heterogeneous, and distributed nature of resources. Equally challenging is the task of integrating information about different properties and usage policies of these resources.

In this paper, we describe appropriate enhancements for grid resource discovery. We focus on the design of GOIDS, an effective resource discovery and integration framework which reduces the complexity of information sharing and discovery on Internet-scale grids. The system relies on ontologies to describe the structure and semantics of resource properties to increase the system's expressiveness and interoperability.

To improve the scalability of this approach, we reconfigure the network topology according to nodes' ontological interests, so that the network exhibits the "small-world" properties [1]. Specifically, we use an ontology directory overlay to help nodes form virtual communities. The community discovery, construction, and maintenance are manipulated in a decentralized and automatic manner. Communities help discriminatively distribute queries to ontologically related nodes, thus reducing the search space and improving the scalability.

Inside each community, nodes share similar interests, but they may use different ontologies. An ontology integration strategy is presented. It permits different community members to establish mappings among the various ontologies. Resource knowledge within a community is indexed in a DHT overlay and is accessible through an expressive ontology-based query language. We proposed two different coarse-grained indexing schemes; an application can choose one scheme according to its specific need.

2. Related work

In medium-sized grids, resource discovery is usually guided by centralized servers [2]. To discover resources in more dynamic, large-scale, and distributed grid environments, P2P techniques have been used (e.g., [14, 15]). Flooding is the predominant search method in unstructured P2P networks. This method, though simple, does not scale well in terms of message overhead. An efficient approach is the use of distributed hash tables (DHTs) [3-6], which has been shown to be scalable and efficient. However, a missing feature is the inherent support for expressive queries.

Research on the Semantic Web project [7] has recently gained much attention for its knowledge integration vision. Its focus is to exploit the power of semantic technologies to aid in information representation, retrieval and aggregation over the web. Most of the Semantic Web projects use the standard

1-4244-1521-7/07/$25.00 ©2007 IEEE

RDF language [12] to describe data. Ontology languages such as DAML+OIL [17] and OWL [18] build on top of RDF allow a user to describe relations between resources, defining a more abstract and expressive resource sharing environment.

In the last few years, several projects aimed to combine the strengths of grid and semantic web technologies, particularly for the use of resource sharing. For example, the UK myGrid [8] project uses ontologies to describe and select web-based services used in the Life Sciences, while OntoGrid [9] mixes in techniques from agent computing and P2P for distributed discovery of semantic knowledge.

The integration of Semantic Web and P2P technologies can also serve to benefit each other. For example, in the InfoQuilt [10] system, nodes use the DAML+OIL ontological language to describe their information. Ontology knowledge is registered to a central server for efficient indexing, while queries are forwarded between nodes in a P2P manner. Helios [11] is another system that uses the P2P model for semantic knowledge sharing. Unlike InfoQuilt, both the ontology knowledge and data instance discovery in Helios are based on an unstructured P2P network.

Like the mentioned systems, our system takes advantages of the semantic web and the P2P technologies to enhance resource sharing and searching. We focus on improving the interoperability and scalability of an Internet-scale grid.

3. System overview

3.1. Two-tier structure

To make the searching more focused, we cluster nodes sharing similar ontological interests together; as a result, the queries can be forwarded to nodes that are likely to contain relevant resources. However, in an open global-scale grid, it is difficult to discover nodes with similar interest. We solve this problem by using an ontological directory to organize dispersed ontologies, and to allow nodes find others sharing similar interest. The directory works as a "rendezvous" for nodes meeting others with similar ontological properties, thus facilitating the construction of semantic clusters. The directory does not need to be predefined; it spontaneously grows as the network ontology evolves.

The system is organized by employing two layers – the directory overlay and the community overlay; and the former facilitates the formation of the latter. Both overlays adopt the P2P-based DHT structure. When a node joins the system, it first locates the interested ontological community by looking up the directory overlay, from where it gets one or more contacts of the interested community. It then joins the community through those contacts. After it joins the community, the new node publishes its resource ontology knowledge in the community overlay. It also maps its ontology with other related ontologies through the community overlay. Based on the ontology knowledge indexed in the community overlay, nodes can share and discover resource information efficiently.

3.3. Peer ontology

3.3.1. Knowledge repository. Nodes use ontologies to explicitly describe details of their possessed resources, as well as their knowledge of concepts and relationships regarding the resources. In our system the ontology knowledge is separated in two parts: the terminological box (T-Box) and the assertion box (A-Box). The T-Box statements describe concepts and relationships between concepts, for example a set of classes and properties. The A-Box represents the concrete knowledge about individuals within the domain, for example the instances of the classes defined in the T-Box. A node also uses inference engines to derive additional facts from instance data and class descriptions.

3.3.2. Ontology mapping. Nodes in the community may use different ontologies. To reconcile the ontology differences, a node maps its ontology with those of other nodes. The ontology mapping is created by mapping related elements from different ontologies. We define the following inter-ontology mappings:

(1) $C_1 \xrightarrow{Mc} C_2 \quad M_c \in \{\subseteq_c, \supseteq_c, =_c, \approx_c\}$

Mc: class mapping between class C_1 and C_2

$\subseteq_c$: *subClass* mapping

$\supseteq_c$: *superClass* mapping

$=_c$: *equivalentClass* mapping

$\approx_c$: *referentialClass* mapping

(2) $P_1 \xrightarrow{Mp} P_2 \quad M_p \in \{\subseteq_p, \supseteq_p, =_p, \leftrightarrow_p\}$

Mp: property mapping between property P_1 and P_2

$\subseteq_p$: *subProperty* mapping

$\supseteq_p$: *superProperty* mapping

$=_p$: *equivalentProperty* mapping

$\leftrightarrow_p$: *reverseProperty* mapping

The defined mappings between different ontologies either refer to the same concept, relation, or one is a special (or general) case of the other. We also note that various ontologies may contain different supplementary information about the same real world individual; thus

we add a special *referentialClass* relation between concepts. This allows individuals to be merged if specific properties match, creating an aggregated entity. The ontology mapping information is indexed in the community overlay. When a node joins the community, it uploads its ontological concepts and properties to the community overlay, where it also learns knowledge of other related concepts and properties. The knowledge gives hints and suggestions to the new node when it maps its ontology to those of others.

4. Ontological directory overlay

This section describes details of the directory overlay: the first layer of the two-layer architecture.

4.1. Introduction

The function of the directory overlay is to assist the construction of community overlays. The directory is represented by the ontology hierarchy which is shaped by the high-level concepts within the whole resource ontologies. Nodes in the directory overlay are also user nodes that are stable and have good Internet connectivity compared with the rest nodes. Concepts are organized into a hierarchical directory, and the directory is distributed among the overlay nodes. A directory path starting from the root is used to represent the ontology domain (e.g., */science/computer science/AI*).

A problem we need to solve here is how to index and search the hierarchical directory data. Many applications use hierarchical DHTs like Canon [16] or HIERAS [2] to index hierarchical data, but they are not applicable to our system: specifically, in Canon, information is only stored in real nodes (leaves nodes), while all internal nodes are purely virtual; however, directory information in our system is stored in real (non-virtual) internal nodes. In addition, it is difficult to implement directory browsing inside the Canon network. HIERAS constructs the multi-level hierarchy by creating multiple DHT overlays for each sub-domain. Maintaining multiple DHTs requires a big overhead; it is a huge waste if the sub-domains are not large enough. In this paper, we propose a simple flat DHT structure to index the hierarchical directory. This structure enables economical, flexible, and balanced lookup services.

4.2. Directory indexing and lookup

The directory overlay provides three lookup interfaces: (1) exact path lookup, (2) directory browser-based lookup, and (3) keyword-based lookup. An exact path lookup query contains the complete directory path of the interested domain, for example *"/computer/hardware/CPU"*. The directory path of the query is hashed to a key and then a corresponding lookup of the hashed key on the DHT is executed. However we can not expect a user knows the exact directory path to locate an interested domain. Therefore, we provide users a more flexible interface: directory browser. Users can expand a directory tree node to browse its child branches until they find the desired directory entries. A node can also specify one or more key concepts in its ontology and use them as keys to lookup the directory overlay. The node that is in charge of the keyword keeps links to the corresponding directory entries. If multiple entries appear under the same keyword, they are returned to the user for further specification. Keywords are extended with WordNet [13] vocabularies and ontology knowledge obtained from the community overlay.

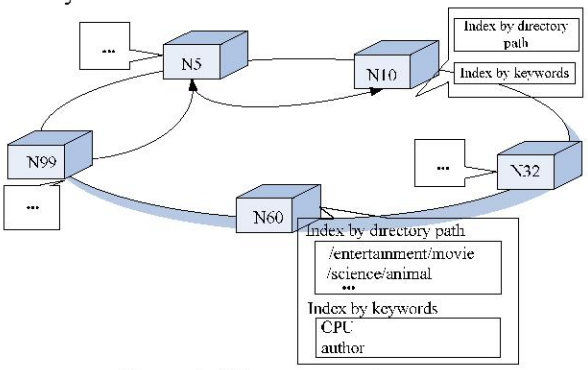


Figure 1. Directory overlay structure

We use Chord [5] to implement the directory overlay. As shown in Figure 1, for each directory path, sub path, or keyword, a hash value is computed using an SHA-1 algorithm. Each key is assigned to its successor node, which is the nearest node traveling the ring clockwise. Each successor node of the directory key maintains a LRU cache storing information of peers that are interested in this directory. To implement the directory browser's functionality, an overlay node that is in charge of a directory entry also stores that directory's children information. When the user chooses one directory, Chord routes to that directory entry and retrieves children directory information, such that the directory can be extended dynamically while browsing. An overlay node also stores keywords that are hashed to it, and links the keywords with related directory entries.

Based on its semantic interests, a new joining node registers to the directory overlay. The overlay node in

charge of that interest returns this new node one or more peer contacts from its cache. Then the new node can join the community through those contacts. A node with multiple interests can register with multiple communities.

4.3. Directory overlay load balancing

Nodes storing very popular directories run the risk of being overloaded by large amount of directory register requests. In this scenario, redistribution via shedding some local directories to other nodes does not guarantee any significant improvement of the situation since even a single directory, popular enough, could induce an overload. Our system uses a replication-based method to solve this problem. Each node periodically checks its current load. If the load is above an overloading threshold, the node will pick a light-loaded node to replicate its directory keys. Since more than one node is now responsible for a popular directory, algorithms can be applied that would encumber each responsible node with only a fraction of the total load. Nodes then periodically exchange their cached peer information to make sure the communities created with their assistance are well-connected.

As we indicated, key replication can relieve the register load of nodes in charge of popular directories. However, in a DHT overlay, another significant source of workload is from relaying messages among nodes. A node may be overwhelmed simply by the traffic induced by forwarding incoming queries, such as in the case of intermediary nodes on the lookup path of popular directories, which can be overloaded by having to forward requests to the directories. To address this problem, a node n can actively redistribute its routing load by replicating its routing table to another node m. Future queries can then be routed to either n or m, and since m has n's routing table, m can keep forwarding queries to the right destination, maintaining the correctness of the network.

In order for the previously outlined scheme to be of any use, however, replica nodes should obviously be advertised in the network so that other nodes know and can subsequently take advantage of their existence. For Chord-based DHTs, the replicas should be entered into the finger tables of related nodes. Similar to the node joining process, the algorithm starts with the i^{th} finger of the original node n and then continues to walk counter-clock-wise on the identifier circle until it encounters a node whose i^{th} finger precedes n. The total cost for the replica update, in terms of the number of messages exchanged, is $O(\log^2 N)$.

5. Ontological community overlay

This section explains how to efficiently manage and discover resource knowledge in a relatively large community using the DHT approach. Sharing in a small community can be achieved using a similar gossip-based strategy applied to a Gnutella-like overlay.

5.1. Resource knowledge indexing

5.1.1. T-Box indexing. Like the database schema, a node's T-box knowledge is more abstract, describing the node's high-level concepts and their relationships. Basically, the T-Box knowledge includes class elements and property elements. They are indexed to the community overlay by hashing the *subject* and the *object* of their RDF triples. For example, a class assertion: <*os*><*superClasss*> <*unix*>, is first indexed by *subject*, and sends the following message to the overlay:

STORE {key, {("subject", <os>),
("predicate", <superClass>),
("object", <unix >)}}
where key=SHA1Hash("<os>")

In the message, the first attribute-value pair *("subject", os)* is the routing key pair, and key is the SHA1 hash value of *os*. Similarly, the triple is indexed by object as well, i.e., use the attribute-value pair *("object", unix)* as the routing key pair. The target DHT node stores the assertion and possibly generates new assertions by applying the entailment rules. These new assertions have to be sent out to other nodes. Thus, after finishing this process, the whole T-Box knowledge is accessible in a well-defined way over the community overlay.

5.1.2. A-Box indexing. T-Box indexing does not require creating and maintaining oversized indices of individual instances. The downside of keeping only the T-Box information in the community overlay, is that query still have to be forwarded to many unrelated (from the instance-level) nodes. To improve the accuracy of the query answering, nodes can index their A-Box instances in the community overlay. The A-Box indexing is based on indexing the RDF triples as well. We store each triple three times, indexing by the *subject*, *predicate*, and *object* respectively.

There is a tradeoff between query overhead and indexing overhead. When the system has a high requirement for fast and efficient query answering, it has to pay more for the knowledge indexing; on the other hand, if the system does not index the detailed

knowledge, it has to explore more nodes for searching the query results. Applications should determine the right indexing granularity according to their specific need.

5.2. Semantic query processing

Once a node has joined a community, it can query information within that community. Our current system uses RDQL [19] as the query language, which is based on matching {*subject, predicate, object*} triples.

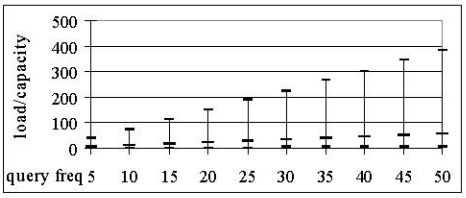
A query is generally constructed with the user's local ontology, and only nodes that use the same ontology can understand the query. To retrieve relevant data reachable through other ontologies, the initial query should be extended and reformulated based on the inter-ontology relationships stored in the community overlay.

Because of the distributed nature of the resource data, the system should be capable of breaking queries into appropriate sub-queries to be executed at different sites and then piecing together the results to compute answers to the original queries. The query analysis engine must use the ontological knowledge to efficiently navigate the search space as to reduce the size of the data that is aggregated at each inter-nodal junction in query processing.

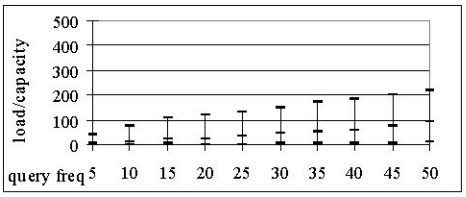
6. Experiment

Our first set of experiments evaluates the performance of the load balancing algorithm of our directory overlay. The initial overlay network size is 5000. Then directory paths are advertised to the overlay. The directory lookup queries are Zipf distributed, regarding the fact that most of the peers are interested in popular directories but only a few are interested in rare directories. Each node is assigned a value (from 1 to 5) representing its capacity. We varied the directory query frequency from 5 to 50 per time slice and conducted a separate experiment for each frequency.

Figure 2 plots the mean and the 10th and 90th percentiles of the peer *workload/capacity* ratio. The result can be used to represent the workload variances on the peers. The smaller the difference, the better the load balancing performs. From Figure 2, we can see that our algorithm greatly balanced the load of each peer. We also observe that, as we increase the query frequency, the variance becomes larger. This is because directory lookups are highly skewed, introducing more query will result in more imbalanced distribution of directory accesses.



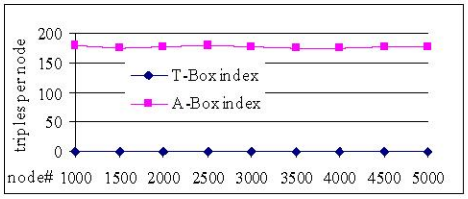
(a) Without load balancing



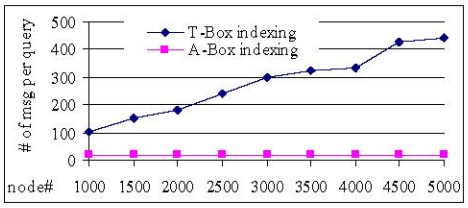
(b) With load balancing adjustment

Figure 2. Mean, 10th and 90th percentiles of the ratio of load/capacity.

Figure 3 compares the performance of the T-Box and the A-Box indexing in terms of query overhead and indexing overhead. There are totally 30 different ontologies. In the simulation, each node can randomly pick one ontology. From the figures we can see that, the A-Box indexing overhead is high, while the searching overhead based on it is low. On the contrary, the T-Box indexing overhead is low but the searching overhead could be high.



(a)Indexing overhead: triples per node

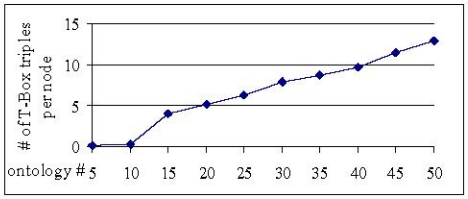


(b) Query overhead: messages per query

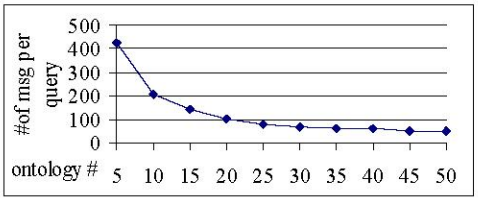
Figure 3. Comparison of the indexing and query overhead between T-Box indexing and A-Box indexing

Figure 4 illustrates the effect of ontology heterogeneity on the performance of T-Box indexing. We noticed that the searching performance based on T-Box indexing is related to the system's ontology heterogeneity: the more ontologies in the network, the

better the searching performs. This is easy to understand: when nodes have homogeneous ontologies, most nodes have the same T-Box knowledge; then indexing T-Box cannot effectively distinguish nodes, thus query forwarding performs poorly. When the system has highly heterogeneous ontologies, T-Box indexing can distinguish nodes' ontologies well; therefore query routing is more efficient.



(a) Number of triples per node



(b) Number of messages created per query

Figure 4. Effect of ontology heterogeneity on T-Box indexing

7. Conclusion

As more and more resources appear in grids, there is a compelling need to find an effective and efficient way to discover and query these resources. In this paper, we presented GOIDS, an ontological framework for resource integration and discovery in large-scale grids. The system provides a distributed ontological directory service to help nodes form communities according to their semantic properties. As a result, searching cost is reduced by sending discovery requests only to appropriate semantic communities. Inside a community, nodes may use different ontologies to represent their resource knowledge, and a distributed integration mechanism was proposed to cope with the mediation between different ontologies. To efficiently locate the desirable resources in the community, a flexible DHT indexing scheme is provided. The main algorithms of the system were evaluated with simulation experiments.

8. Reference

[1] Watts, D.J., Strogztz,S,: Collective dynamics of 'small-world' networks. Nature 393 (1998).

[2] Zhiyong Xu, Rui Min, Yiming Hu. "HIERAS: A DHT Based Hierarchical P2P Routing Algorithm". ICPP 2003

[3] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," Technical Report, UCB/CSD-01-1141, April 2000.

[4] A. Rowstron and P. Druschel. "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, Middleware, November 2001.

[5] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H.Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," ACM SIGCOMM, August 2001, pp. 149-160.

[6] S. Ratnasamy, P.Francis, M.Handley, R.Karp, and S. Shenker. "A Scalable Content-Addressable Network," ACM SIGCOMM, August 2001, pp. 161-172.

[7] Berners-Lee T, Hendler J, Lassila O (2001) The semantic web . Sci Am 284(5):34–43

[8] Stevens,R.D., Robinson,A.J. and Goble,C.A. (2003) myGrid: personalized bioinformatics on the information grid. Bioinformatics.

[9] OntoGrid project: http://www.ontogrid.net/

[10] M. Arumugam, A. Sheth, and I. B. Arpinar. Towards peer-to-peer semantic web: A distributed environment for sharing semantic knowledge on the web. In Proc. of the International World Wide Web Conference 2002.

[11] S. Castano, A. Ferrara, S. Montanelli, and D. Zucchelli. Helios: a general framework for ontology-based knowledge sharing and evolution in P2P systems. In IEEE Proc. of DEXA WEBS 2003 Workshop, Prague, Czech Republic.

[12] Ora Lassila and Ralph R. Swick, "W3C Resource Description framework (RDF) Model and Syntax Specification".

[13] G. Miller, "WordNet: A lexical database for English", Communications of the ACM, vol. 38, no. 11, 1995.

[14] M. Cai, M. Frank, J. Chen and P. Szekely, "MAAN: A Multi-Attribute Addressable Network for Grid Information Services." The 4th International Workshop on Grid Computing, 2003.

[15] Iamnitchi A, Foster I, "On Fully Decentralized Resource Discovery in Grid Environments," Proc. The 2nd IEEE/ACM International Workshop on Grid Computing 2001, Denver.

[16] P. Ganesan, K. Gummadi and H. Garcia-Molina, "Canon in G major designing DHTs with hierarchical structure", Proceedings of the 24th ICDCS, Tokyo,2004.

[17] I. Horrocks, F. van Harmelen, and P. Patel-Schneider. DAML+OIL.
http://www.daml.org/2001/03/daml+oil-index.html

[18] W3C. Web-ontology (webont) working group.
http://www.w3.org/2001/sw/WebOnt/.

[19] A. Seaborne. "RDQL: A Data Oriented Query Language for RDF Models."
www-uk.hpl.hp.com/people/afs/RDQL/, 2001.