Semantic Overlay Network for Grid Resource Discovery

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

Abstract

This paper presents a new approach to grid resource discovery by using semantic peer-to-peer (P2P) overlays. The framework is based on the RDF metadata infrastructure, allowing a rich and extensible description of resources and queries. To avoid flooding the network with a query, we propose a comprehensive semantics-based query forwarding strategy, which only forwards queries to semantically related nodes. After the related nodes have been located, the original RDF query is used to do the final query and retrieval. Results from simulation experiments demonstrate that this architecture is scalable and efficient.

1. Introduction

With the growing volume of information and resources stored in Grids, it is becoming increasingly difficult to search for desired resources in Grids.

Traditionally, resource discovery in Grids is based mainly on centralized or hierarchical models. For example, in the Globus Toolkit [4], users can get a node's resource information by directly querying a server application running on that node, or querying dedicated information servers that retrieve and publish the organization's resource information. Although interactions between these information servers are supported, the general-purpose decentralized service discovery mechanism is still absent.

To discover resources in more dynamic, large-scale, and distributed environments, P2P techniques have been used in Grids. For example, [14] and [16] use different P2P architectures for Grid information service. However, P2P technology brings new problems: The flooding-based unstructured P2P systems do not scale well in terms of message overhead. Random walks [7, 13], an alternative to flooding, can reduce the amount of network traffic, but it is at the cost of query latency. DHT-based systems [9–12] have been shown to be scalable and efficient. However, a missing feature is the inherent support for complex queries. Another hurdle to DHT deployment is its tight control of both data placement and network topology.

In this paper, we propose a semantics-based P2P model for grid resource discovery. It uses RDF [2, 5] to represent both resources and queries. In the framework, resource providers register their resource information to local information nodes. Information nodes connect with each other, forming a P2P overlay. Resource searching is carried out only on top of this P2P overlay. To support complex RDF queries without flooding the whole network, our system uses a Resource Distance Vector (RDV) routing algorithm. The basic idea is to extract the building blocks from RDF metadata and then summarize them to form a compact structure. Based on this summarization, we create a routing table to guide the query forwarding. When compared to unstructured P2P applications oblivious of the resource location, this routing strategy reduces both the query overhead and query latency, and guarantees a higher query hit ratio. Compared with DHTs, our approach inherently supports rich queries, and requires no explicit control over the network topology or data placement.

2. Resource representation and queries

Metadata plays an important role for complex queries that go beyond string matching. We use an RDF metadata representation to encode resources. The RDF metadata index not only provides improved query capabilities, but also supports more sophisticated query routing. Every peer maintains a resource index table, and peers exchange their indices. Queries can then be distributed by relaying based on these indices. However, exchanging RDF indices between nodes is almost impossible because each node may maintain a large number of resources. To reduce the overhead of propagating the index information, we must make the indices lightweight. Our strategy is to extract the

0-7803-9493-3/05/$20.00 © 2005 IEEE

building block of the RDF metadata (subject, predicate, object) and then summarize them in a compact structure: a triple filter. A triple filter includes three Bloom filters: the subject filter, the predicate filter, and the object filter. An RDF triple can be hashed to these three filters. Every node maintains a local triple filter and several aggregated neighbor triple filters. These filters form a routing table that directs query forwarding. Section 3.1 presents the creation and maintenance of this routing table.

We use RDQL [15] to query the metadata. We convert an RDQL query to a triple sequence and then match the triple sequence by the triple filters to determine where to forward the query. If the query can pass a node's filters, then it will be forwarded to that node. This filtering technique limits query routing traffic by forwarding queries only to a small number of related nodes. However, it cannot guarantee that the query can be answered through the forwarding path because the matched elements may belong to different resources. Matching the triple sequence with the filters relaxes the constraints of the original query. Nevertheless, an advantage of this scheme is that the filter can introduce only false positives but never false negatives – the correct nodes will not be excluded. Strict accuracy is not necessary for the routing process, since the local RDF database is checked by the original query in the end. The resource summarization works only as a hint for forwarding the query to related nodes.

3. Overlay routing

We propose a so-called resource-distance-vector (RDV) routing algorithm. It uses a distance vector approach to route the query to the nearest matching nodes. Every peer in the overlay network maintains a resource index table. This table uses the triple filters we mentioned above, and includes distance (in number of hops) information. Peers exchange the resource indices with their neighbors, and update relevant entries in their table. The distance information is updated whenever passing through a node. To reduce false positives brought by the result of resource information aggregation, we set a hop count limit, which we call *radius*, to limit the number of hops the resource information can travel. When a node receives a query request, the algorithm chooses the shortest route to forward the query. Therefore, if there is more than one provider supplying the same resource, then with high probability, the algorithm will forward the request to the nearest one. In addition, a "heuristic jump" method is used to expedite the searching process by skipping over the "barren" areas.

3.1. Routing table

As mentioned, each node maintains a RDV routing table (RDVT). The RDVT contains both local and neighbor triple filters. Besides resource information, the triple filters also record the distance to the resource. Figure 1 shows part of the network with the associated RDVT for each node. For brevity, only one of the three filters is shown here. Each element in the filter is associated with a distance number: the minimum distance to a matching resource. The first row of the RDVT is the local filter containing local resource index. For example, node *A*'s local filter contains a local resource *a,* which is mapped to two positions *(2, 4)* in the filter. We set the distance number of a local resource as *0*. The rest of the rows represent resources accessible from neighbors. For example, *A*'s second row contains resources that can be reached through the neighbor *B* (e.g., resource *b(4,0)* with *1* hop).

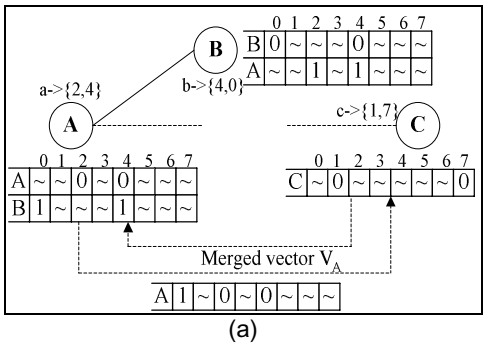

Figure 1. Maintaining routing indices

Figure 1 also illustrates the RDVT update process when a new node *C* joins the network. Node *C* joins the network by connecting to an existing node *A* in the network. After the connection is established, node *C* sends its resource indices to *A*. Similarly, *A* informs *C* of all the resources *A* has knowledge of. Specifically, *A* merges its local and neighbor vectors to one vector and sends it to *C*. The aggregation is done by comparing

every element of the vectors, and selecting their minimum value. The merged vector of *A* represents resources accessible from *A* and their shortest distances to *A*. After *C* receives the merged vector from *A*, it adds *1* hop to each element of the vector, and adds an additional row in its RDVT (as shown in Figure 1(b)). After *A* receives *C*'s resource information and updates its routing table, it informs its neighbors (in this case, node *B*) of the update. In the merged vector, if an element's value equals *radius*, we reset the value to infinity ("~" in the figure), representing "not available."

Each node sends updates to and receives updates from its directly connected neighbors. Eventually the table stabilizes, and all resources within the range determined by *radius* are known. Nodes need to periodically "ping" their neighbors to make sure that they are still alive. To reduce the overhead of transmitting routing information, a soft-state update mechanism is used, in which routing information is exchanged periodically. At any given time, the resource routing information may potentially be stale or inconsistent, but as mentioned, this approximation will not affect the system's fidelity.

3.2. Query forwarding

Figure 2 illustrates a query routing example. In this example, the *radius* is set to *3*, so nodes are only aware of resources within 3 hops. Node *A* receives a query for resource *e*, which is mapped to two positions: *3* and *6* in the filter. It checks its routing table and finds two matches: through *C* with *2* hops *(C_3=2, C_6=2)* and through *D* with *3* hops *(D_3=3, D_6=3)*. So the shortest distance to the resource is *2* through neighbor *C*. Therefore, the query is forwarded to *C*. Similarly, *C* forwards the query to *E*. *E* finds a match in its local vector, and then it checks the RDF database with the original RDQL query.

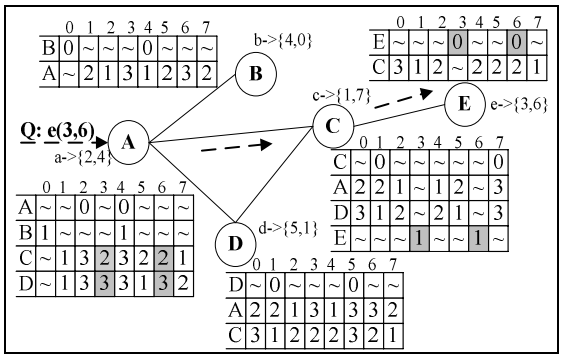

Figure 2. Query routing

Our routing algorithm works fine with networks containing cycles. Because of cycles, a node may receive a query multiple times. To avoid processing queries more than once, every query has a unique query ID and every node keeps a list of recently received query IDs. If a query has been received before, it will be discarded.

3.3. Heuristic jump and caching

By setting a *radius*, we limit the distance a node's resource information can travel. This reduces false positives, but at the same time, a node does not have global knowledge of the network but only a local view of the neighborhood. Because of this, a node may not find enough matches from its RDVT to forward queries. To address this problem, we introduce a forwarding method called "heuristic jump."

This method allows the system to keep additional long-distance links as an addendum to the RDVT. When the RDVT cannot resolve the query, the query will "jump" to remote nodes the links point to. To discover those long-distance links, the system employs an aggressive caching technique. After finding the result of a query, the result travels along the reverse path to the requester. Whenever it is passed through a node, it is cached in that location. Every internal node caches the query, the destination node, and the distance to that node. During the query-forwarding process, when a node cannot find enough matches in its routing table, it chooses appropriate long-distance links from its cache and forwards the query accordingly. This expedites the searching process by jumping over barren areas. Candidate long-distance nodes should be located outside the neighborhood area; i.e., the distance should be greater than *radius*.

4. Experiments

Via simulations, we compared the performance of our routing algorithm with Simple Random Walk (SRW) and Advanced Random Walk (ARW). SRW randomly chooses neighbors to forward queries. ARW forwards to nodes according to past experience. We deploy 5 walkers for these three routing algorithms.

Figures 3 and 4 compare the three routing algorithms in terms of query recall rate and query hop consumption. As expected, our RDV routing algorithm outperforms either SRW or ARW on both metrics. Compared with Random Walks, RDV always forwards the query to the right direction, so it can find more results and thus enjoy a higher recall. In addition, RDVT records the resource distance information, so it

can forward queries to the nearest resource providers. That's why RDV needs fewer hops to resolve a query.

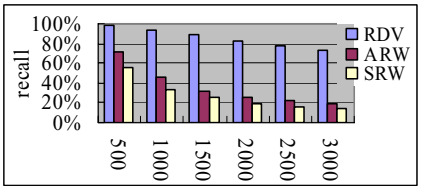

Figure 3. Recall versus network size

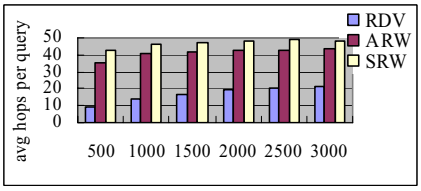

Figure 4. Hops versus network size

5. Related work

Many recent P2P searching techniques relate to our research. Due to space limitation, we list only some of the most relevant ones: Edutella [3] and RDFPeers [6] have used RDF to represent resources and queries. Bloom filters have been used as a succinct summary technique for query filtering and routing. For example, OceanStore [17] uses attenuated Bloom filters to store objects information. PlanetP [8] also uses Bloom filters to distribute a summary of the contents of each peer. Paper [1] discusses routing indices based on the various aggregation strategies of content indices.

6. 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 present a novel design for resource discovery in large-scale Grids. The system is based on the P2P model and provides a complex query interface. It supports rich resource descriptions and complex queries by encoding resources and queries with RDF. To avoid flooding queries to irrelevant nodes, a semantics-based routing scheme is proposed to route queries only to related nodes. This system has been evaluated by a group of simulations, which show that the proposed routing schemes are both efficient and scalable.

7. References

[1] A. Crespo and H. Garcia-Molina. "Routing indices for peer-to-peer systems." In *Proceedings International Conference on Distributed Computing Systems*, July 2002.

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

[3] W. Nejdl, M. Wolpers, W. Siberski, A. Loser, I. Bruckhorst, M. Schlosser, and C. Schmitz. "Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-To-Peer Networks." *In Proceedings of the Twelfth International World Wide Web Conference (WWW2003)*, Budapest, Hungary, May 2003.

[4] Globus Toolkit: http://www.globus.org/toolkit/

[5] Dan Brickley and R.V.Guha. "W3C Resource Description Framework (RDF) Schema Specification." http://www.w3.org/TR/1998/WD-rdf-schema/

[6] M. Cai and M. Frank. RDFPeers: "A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network." *In International World Wide Web Conference (WWW)*, 2004.

[7] Lv, C., Cao, P., Cohen, E., Li, K., Shenker, S. "Search and replication in unstructured peer-to-peer networks." In: *ACM, SIGMETRICS* 2002.

[8] F. M. Cuenca-Acuna, C. Peery,R. P. Martin, andT. D. Nguyen. "PlanetP: Infrastructure Support for P2P Information Sharing," *Technical Report Department of Computer Science, Rutgers University, Nov. 2001.*

[10] 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.

[11] 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*, 2001

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

[13] Adamic, L., Huberman, B., Lukose, R., Puniyani, A.: "Search in power law networks." *Physical Review* (2001)

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

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

[17] S. C. Rhea and J. Kubiatowicz, "Probabilistic location and routing," *in Proc. INFOCOM, vol. 3,* New York, NY, June 2002, pp. 1248–1257.