International Journal of Computational Science
1992-6669 © Global Information Publisher
2009, Vol. x, No. x, xx-xx

A Framework for Efficient Query Answering on Semantically Heterogeneous Grids

Juan Li[1]* and Ying Su[2]

[1] Computer Science Department, North Dakota State University, USA
J.Li@ndsu.edu
[2] Computer Science Department, University of British Columbia, Canada
yingsu@cs.ubc.ca

Abstract. With the rapid growth of Grid computing, more and more data are generated and stored across the grid. To fully utilize the data, efficient data search and query answering mechanism is becoming a very important issue. However, the sheer amount of data and their heterogeneity nature pose challenges that current technology cannot cope with efficiently. In this paper, we propose an efficient query answering solution that integrates topology adaptation, semantic query routing, and view-based caching techniques to reduce bandwidth cost of distributed query processing while allowing efficient evaluation of complex semantic queries over large-scale, fully decentralized, and semantically heterogeneous grids. Simulated experimentations illustrate that our comprehensive query strategies effectively reduce the cost of query evaluation and improves the query performance.

Keywords: Grid Computing, Query Evaluation, Routing, Semantics, Ontology.

1 Introduction

The Grid is a distributed computing infrastructure that enables coordinated resource sharing within Virtual Organizations (VO) [12] consisting of individuals, institutions, and resources. Nowadays, large communities of people distributed around the world are increasingly using Grid to share resource and data at numerous levels: inside an organization, across organizations, and even worldwide. One of the key challenges in today's Grids is the need to deal with knowledge and data sources that are distributed, heterogeneous, and dynamic. In such systems, a complete global view

* Corresponding Author. Fax: 17012318255, Email: J.Li@ndsu.edu.

or understanding is impossible to achieve. Therefore information services need to go beyond the centralised strategy to a decentralized, distributed, semantics-based strategy. The Semantic Grid [34] aims to address this issue by adding meaning (ontologies, annotations and negotiation processes as studied in the Semantic Web and Software Agent paradigms) to the Grid. In this way, the Semantic Grid not only provides a general semantic-based computational network infrastructure, but a rich, seamless collection of intelligent, knowledge-based services for enabling the management and sharing of complex resources and reasoning mechanisms.

To locate desirable data and information from semantic grid in a timely and reliable manner, effective query and discovery mechanisms are required. Query evaluation in a large-scale semantic grid is challenging due to the considerable diversity of data schema, large number of data, dynamic behavior of the data source, and geographical distribution of the dataset. The recently proposed Data Grids [9] have evolved to tackle the challenges of large datasets and multiple data repositories at distributed locations by providing high capacity resources such as supercomputers, high bandwidth networks, and mass storage systems. However, increasing capacity does not fundamentally solve the problem. A single query may involve transferring of large amount of data among many geographically distributed nodes in the grid network. A query intensive system may easily be overwhelmed by large amount of query traffic. Therefore, scalable query answering and optimization are vital to query-intensive grids.

Peer-to-peer (P2P) technology has been used as a solution to distributed query evaluation, because it scales to very large networks, while ensuring high autonomy and fault-tolerance. The recently proposed structured P2P systems in the form of Distributed Hash Tables (DHTs) [26, 28, 29, 32] are a promising approach for building massively distributed data management platforms. However, they offer few data management facilities, limited to IR (Information Retrieval) -style keyword search. Keyword search is appropriate for simple file-sharing applications, but is unable to deal with complex semantic queries which have various properties and sophisticated relations with each other. More recently, a few studies [8, 24] extended the DHT-based P2P to support semantic queries. The basic idea is to map each keyword of a semantic entity to a key. For example, RDFPeer [8] indexes each RDF [7, 17] triple to support semantic RDF query. A query with multiple keywords then uses the DHT to lookup each keyword and returns the intersection. Systems, such as [26], avoid this multiple lookup and intersection by storing a complete keyword list of an object on each node. In this way, the DHTs can support multi-keywords queries. However, DHTs still have difficulty to support other semantically richer queries, such as wildcard queries, fuzzy queries, and proximity queries. In addition, most DHT-based applications require all peers in the system sharing a uniform schema, which is impractical in reality. These limitations restrict the deployment of DHTs to semantic data discovery.

In this paper, we propose a query answering mechanism that works in a fully decentralized and scalable way. Query evaluation in our system will address the challenges of diversity of schema, large amount of data, and geographical distribution of the dataset. As pointed out by Tatarinov et al. [30], it is neither practical to maintain a global schema for all nodes in a grid to map their local schemas with, nor is it feasible for a node to map its schema with all other nodes' schemas; a better approach is to let each node connect to and construct mapping with one or a few peer nodes, then queries can be translated between different peers. In our system, users can pose queries with their preferred local ontology/schema. Our query evaluation scheme will propagate queries to the network using pair-wise ontology mapping to translate the query between different ontologies and return a relatively complete set of relevant data in that preferred ontology. It thus achieves the ontology mediation capabilities of a data integration system, but in a more extensible, decentralized way. Moreover, we optimize the query evaluation system with novel and effective algorithms which make distributed query evaluation both bandwidth-efficient and cost-effective. In particular:

1. We propose effective topology adaptation schemes that organize the network according to semantic properties of participating nodes in order to improve system scalability and reduce information loss caused by transitive query reformulation.
2. We design a view-based ontology mapping scheme to facilitate information sharing among nodes with different ontology.
3. We propose a query routing algorithm based on semantic similarity and view-based query containment caching, which helps eliminate redundant queries, speedup query evaluation, and reduce network traffic.
4. We propose a mechanism for result collecting by efficiently pipelining and integrating results along querying path.

The rest of the paper is organized as follows. Related work is presented in Section 2. Section 3 describes the detailed design of our proposed system. In Section 4, we evaluate the proposed methods and show the effectiveness of our proposed system with a comprehensive set of simulations. Concluding remarks are provided in Sections 5.

2 Related Work

The prevalence of the Internet and the boom of the P2P applications have brought an important shift in the way data is created, shared, stored, distributed, and manipulated. As such, the applications must manage data in a large-scale and distributed environment and resolve heterogeneities with respect to the schemas and their data. Some distributed data management systems have emerged, allowing users to query and retrieve data from each other. In this section, we present some important research works related to this issue.

Edutella[22] is a P2P network for searching semantic web metadata. Each Edutella peer can make its metadata information available as a set of RDF statements. The distributed individual RDF peers register the queries they may be asked through the query service (i.e., by specifying supported metadata schemas), and queries are sent through the Edutella network to the subset of peers who have registered with the service to be interested in this kind of query. The mediation process is done in two steps: when queries can be answered completely by one peer, it uses a simple mediation model. When query results are distributed on several peers, it relies on the integrating mediators or hubs. These mediators or hubs translate a query to a set of sub-queries which are forwarded to corresponding heterogeneous peers. To forward queries between nodes or hubs, Edutella uses JXTA to broadcast queries to a HyperCup topology. The simple P2P broadcast structure used by Edutella makes it very difficult to scale to large-scale networks.

In a contemporary work by Bernstein, Giunchiglia et al. [5], the authors argued that one cannot assume the existence of a global schema for all the peer databases. Instead, they treated the data being managed within the P2P network as an open collection of possibly overlapping and inconsistent databases. The inter-dependencies between local peer databases were described by the "coordination formula" and were defined by a declarative language. This model is hence called the "Local Relational Model". However several issues were not covered in their work including the protocol setting up acquaintances and exchange peer names, query optimizations and constraints on query propagation, etc.

Same as the Local Relational Model, the authors of Piazza [14] also based their work on the assumption that a global schema is not possible. They used two kinds of schema mappings: "peer descriptions" which map between each peer's "view of the world", and "storage descriptions" which map a peer's "view of the world" to the specific data at the peer. Queries are sent out in a flooding manner and translated along the path using the peer descriptions. This flooding-based query forwarding also has the scalability and efficiency problem. The Piazza project did not give any optimization suggestion on the query forwarding/routing.

While Piazza assumes static pair-wise mappings, the Chatty Web [1] manages the mappings in an evolutionary and decentralized process that relies on the initial pair-wise local interactions. During the life-time of the system, each peer has the potential to learn about existing mappings and add new ones. As a result, the network converges to a state where a query is only forwarded to peers that are most-likely to understand the query and where the correct mappings are increasingly reinforced.

Similar to Chatty Web, PeerDB [23] also tries to make the "best" peers topologically close to the query originator. However it does rely on some global names lookup servers to track the IP address and status of every peer. The query answering process is facilitated by agents: the relation matching agents would firstly use IR techniques to find out promising peers, then the data retrieval agents translate the submitted queries to those peers.

While all these solutions deal with the data heterogeneity in P2P networks, most of them do not aim at getting complete result set. The Edutella system [22], for example, would only send the query to a specific peer if it believes this peer can answer the query. While it is easy to tell if a peer is able to answer a query given its schema, it is far more difficult to tell that the data a peer provides serves the query issuer's need completely. Other issues that these work omitted are the system and scalability and query processing optimization, which is the main focus of our work.

3 System Design

In this section, we present the design of our query evaluation system, aiming at resolving the heterogeneity, scalability, and efficiency problems faced by query evaluation of large-scale semantic grids.

3.1 System Overview

For efficient sharing and collaborating over a large-scale network composed of nodes that are semantically heterogeneous and geographically distributed, our system provides a query answering mechanism in a fully decentralized and scalable way. As shown in Fig.1, the system consists of two layers: (a) a physical network layer and (b) a virtual semantic overlay layer. Nodes in the physical network layer are assumed to be connected physically. Nodes in the virtual semantic layer are connected by virtual semantic links and are referred to as neighbors. Each node has its own ontology. Once the neighbor-relationships are established, nodes can construct mapping with their neighbors. Queries can be translated between peers according to pair-wise ontololgy mapping. To improve the scalability and efficiency of query evaluation, our optimization is performed in the following aspects: (1) topology adaptation, (2) view-based ontology mapping, (3) query forwarding based on semantic containment and view-based caching, and (4) pipelining and combining the results along the path.

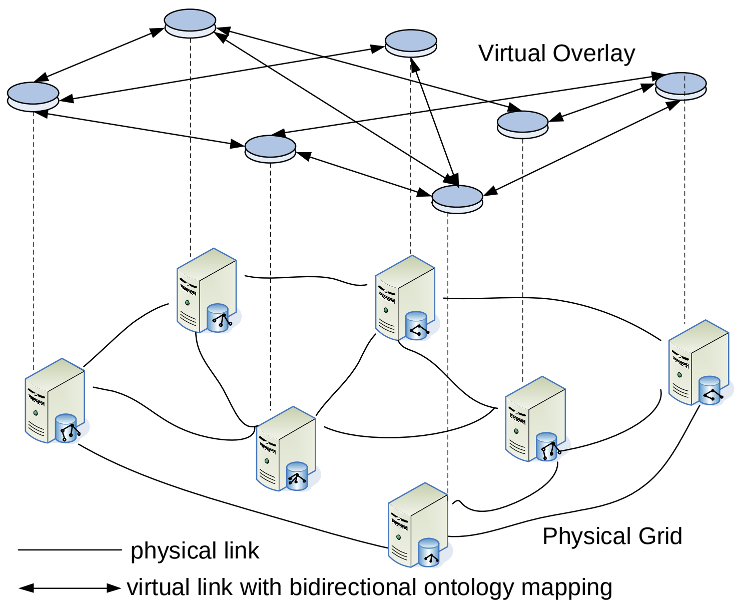


Fig. 1. Example of ontology mapping

3.2 Topology Adaptation

The objective of topology adaptation is to make the system's dynamic topology match the semantic clustering of peers. This would allow queries to quickly propagate among relevant peers. In addition, this adaptation allows semantically related peers to establish ontology mapping relationships to reduce information loss in the query reformulation process. The rationale behind the topology adaptation is based on two theories of the existing research (1) The most promising peers who are able to answer a particular query are those who are more similar with the querying peer [6]. (2) Connecting and constructing schema mapping with semantically similar nodes will reduce information loss caused by query reformulation [15]. We organize the network as an unstructured peer-to-peer (P2P) network which is capable of answering arbitrary types of queries. The network topology is adapted according to nodes semantic similarities. In our system, topology adaptation is performed at different states of a nodes' life time: (1) upon joining the network, a node chooses neighbors by their semantic similarity. (2) A node also gradually updates its links according to its query processing experiences, so that its topology always reflects its changing interest and data contents.

3.2.1 Ontology summarization

To configure the network topology according to nodes' semantic similarity, the first task is to measure the semantic similarity between nodes. There has been extensive research [16, 18, 27] focusing on measuring the semantic similarity between two objects in the field of information retrieval and information integration. However, their methods are very complex and computationally intensive. In this paper, we propose a light-weight method to compute the semantic similarity between two nodes. To compute the semantic similarity, we first summarize each node's semantic properties. We use a node's Terminology Box (TBox) [3] ontology to define its high-level concepts and their relationships. TBox is a good abstraction of the ontology's semantics and structure. We use keywords of a nodes' TBox ontology as its ontology summary. Because a semantic meaning may be represented by different labels in different ontologies, while it is also possible that the same literal label in different ontologies means totally different things. Ontology comparison based on TBox keywords may not yield satisfying results. Therefore, we extend each concept with its semantic meanings in WordNet [11], so that semantically related concepts would have overlaps. We use two most important relationships in WordNet – synonyms and hypernym – to expand concepts.

After extension, a node's ontology summary set may get a number of unrelated words, because each concept may have many senses (meanings), and not all of them are related to the ontology context. A problem causing the ambiguity of concepts is that the extension does not make use of any relations in the ontology. Relations between concepts are important clues to infer the semantic meanings of concepts, and they should be considered when creating the ontology summary. Therefore, we utilize relations between the concepts in an ontology to further refine the semantic meaning of a particular concept. Only words with the most appropriate senses are added to the summary set. Since the dominant semantic relation in an ontology is the subsumption relation, we use the subsumption relation and the sense disambiguation information provided by WordNet to refine the summary. It is based on a principle that a concept's semantic meaning should be consistent with its super-class's meaning. We use this principle to remove those inconsistent meanings. For every concept in an ontology, we check each of its senses; if a sense's hypernym overlaps with this concept's parent's senses, then we add this sense and the overlapped parent's sense to the ontology summary set. In this way we can refine the summary and reduce imprecision.

3.2.2 Semantic Similarity

To compare two ontologies, we define an ontology similarity function based on the refined ontology summary. The definition is based on Tversky's "Ratio Model" [31] which is evaluated by set operations and is in agreement with an information-theoretic definition of similarity [21]. Assume A and B are two nodes, and their ontology summary are S(A) and S(B) respectively. The semantic similarity between node A and node B is defined as:

$$sim(A, B) = \frac{|S(A) \cap S(B)|}{|S(A) \cap S(B)| + \alpha\, |S(A) - S(B)| + \beta\, |S(B) - S(A)|}$$

In the above equations, "∩" denotes set intersection, "–" is set difference, while "||" represents set cardinality, "α" and "β" are parameters that provide for differences in focus on the different components. The similarity *sim*, between *A* and *B*, is defined in terms of the semantic concepts common to *A* and *B*: $S(A) \cap S(B)$, the concepts that are distinctive to *A*: $S(A)-S(B)$, and the features that are distinctive to *B*: $S(B) - S(A)$. Two nodes, node A and node B are said to be semantically related if their semantic similarity measure, $sim(A,B)$ exceeds the user-defined similarity threshold t $(0<t\leq 1)$.

3.2.3 Topology adaptation

The construction of an ontology-based topology is a process of finding semantically related neighbors. A node joins the network by connecting to one or more bootstrapping neighbors. Then the joining node issues a neighbor-discovery query, and forwards the query to the network through its bootstrapping neighbors. The bootstrapping neighbors then use strategies, such as [20], to efficiently propagate the neighbor discovery query to the network to find semantically related neighbors for this new node. After this neighbor-discovery process, a new node is positioned to the right semantic virtual organization in the network, facilitating efficient query forwarding.

Because of the dynamic property of the large-scale grid network, and the evolution of nodes' ontology property, neighbor discovery for a joining node is not once and for all, but rather the first-step of our topology adaptation scheme. A node should keep updating its neighbor links according to its query experiences, including queries it received as a query forwarding router, and query result it collected as a query requestor. Based on the query experiences a node may add or delete neighbors according to the dynamic semantic environment. This way, the network topology is reconfigured with respect to peers' semantic properties, and peers with similar ontologies are close to each other.

3.3. View-based ontology mapping

As soon as a new node find its position in the network, it establishes ontology mappings with its neighbors, who posses similar ontologies with itself. In our system, ontology mappings between nodes are based on TBox ontology view. Fig. 2 illustrates the mapping between two neighbors: peer1 and peer2. Mappings are described by dashed arrows between the mapped ontology elements.

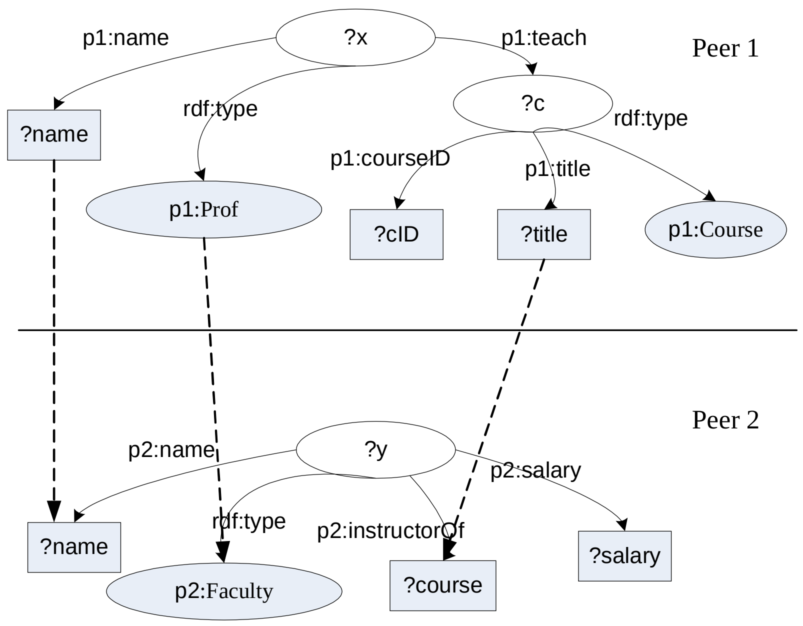


Fig. 2. Example of ontology mapping

Fig. 3 illustrates the RDF [17] views corresponding to semantic mappings in Fig.2, in which peer2 advertises its ontology to peer1. Mappings are defined as views in the form of datalog. It can be viewed as a query over the ontology. Peer2's advertisement based on this view is capable to answer query patterns using peer1's ontology.

```
V: p2(?name, ?course) :-
          (?x rdf:type p1:Prof).
          (?x  p1:name ?name).
          (?x  p1:teach ?c).
          (?c  rdf:type p1:Course).
          (?c  p1:title ?course).
```

Fig. 3. View example

With the view-based mapping constructed, a node can determine where to forward a query and rewrite the query accordingly.

3.4. Cost-efficient query evaluation

Efficient query forwarding/routing is another critical factor determining the system performance and usability. Our query forwarding strategy tries to optimize query forwarding paths and eliminate redundant queries. In particular, we utilize the semantic similarity metric proposed in Section 3.2.1 to guide query forwarding; and we propose a containment-based caching strategy to reduce unnecessary query traffic.

In a distributed and unstructured network, queries are evaluated by forwarding between neighbors. Our semantic similarity function can be used as an effective metric to direct queries only to related neighbors. However, there is still chance of large number of duplicated messages flooding in the network. For an ontologically homogenous network, this flooding-based query forwarding can be optimized by existing approaches such as [10] and [19]. These approaches can reduce or eliminate redundancy by cutting flooding paths, since as long as one copy of the query message reaches the target nodes, the query can be evaluated correctly and get a complete result set. However, in a semantically heterogeneous grid, optimization cannot be done by simply cutting the flooding paths. The reason is that, in such environments, ontology mappings have to be established between neighboring nodes. Queries are forwarded and translated according to the mappings between neighbors. Therefore, query rewrites can come to the same nodes by different paths. Any two rewrites of the same query may differ from each other to any extent, from being identical to one containing the other, or having no containment relationship at all. Arbitrarily cutting flooding paths may lose some query rewrites thus causing result loss. On the other hand, answering all variants of the query rewrites may not be necessary, because some rewrites may be the same, and some may be contained by others. Generating answers for redundant queries will produce redundant results, which will strain the network and waste the bandwidth.

In this section, we propose a comprehensive query routing and optimization scheme to address the aforementioned challenges of semantic query evaluation and enable efficient query evaluation.

3.4.1 Motivation example

In order to illustrate the problem, consider an example scenario based on Fig.4. This example illustrates a grid network consisting of universities and organizations sharing data and information resources. Mappings have been constructed between neighboring data sources. Nodes do not have complete knowledge of the grid but every node has a view of its neighbors' ontology summary.

Suppose a query Q_{NDSU} is posed over node *NDSU*, asking for information about universities' research personnel and publication. To get more results from the network, after checking its local dataset, node *NDSU* forwards the query to its neighbors. It is obvious that NDSU should not forward the query to City_Fargo, because its ontology is not related to the query. This can be determined by measure the semantic similarity (if the similarity is beyond a predefined the threshold). After excluding unrelated neighbors, the query will be forwarded to all related neighbors: *DBLP* and *UBC*. These two nodes will then use the ontology mappings with *NDSU* to reformulate the query Q_{NDSU}. Let us denote the resulting queries as Q_{DBLP} and Q_{UBC}. Q_{DBLP} and Q_{UBC} are then further reformulated over UW, resulting in a pair of queries Q_{UW} and $Q_{UW'}$. Before reformulating any of these queries further, it is important to make sure that these two queries are not redundant with each other. A query is redundant with respect to another query if the query always returns a subset of the results of the other query. Reformulating a redundant query leads to wasted work. As a result, if either Q_{UW} or $Q_{UW'}$ is redundant, the redundant query should be discarded without further processing.

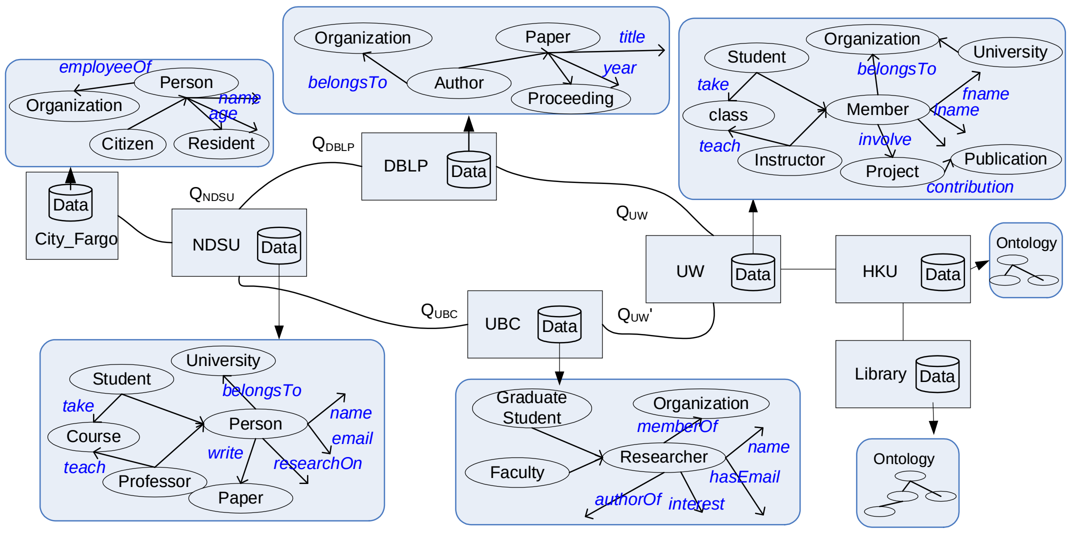


Fig. 4. Example of a semantic grid for sharing semantic data. Each peer's ontology is shown.

From another aspect, there are two paths between *NDSU* and *UW*. Since *DBLP* does not model University (including student and professor) information, data about University in *UW* cannot be used for answering a query on *NDSU*, if we use the path that goes through DBLP. In contrast, the path through *UBC* does enable that data flow. Therefore, finding the right semantic path is critical to query answering in the network. Based on this observation, we try to optimize the process by eliminating redundant queries as early/much as possible. In this paper, we focus on conjunctive SPARQL [25] queries formed by class and properties as well as projected variables. Other SQL-Like queries can be processed similarly.

3.4.2 Query evaluation process

A query *q* is uniquely identified by its query ID (*q*.id) issued by the originator. It also includes the ID of the query originator (*q*.orininator) and the ID of the previous hop (*q*.lastHop) that passes the query to the current node, and the SPARQL query string (*q*.query). When a node receives a query from a neighbor, it first tries to answer the query with its local data. Then it will measure the semantic similarity between the query and the ontology summaries of its neighbors. Based on the similarity measure, the query will only be forwarded to neighbors that are semantically related to the query (i.e., the similarity value is beyond the predefined threshold *t*). Before forwarding the query to a neighbor, the query will be reformulated to the neighbor's ontology according to the ontology mapping. The translated query *q'* is called a rewriting of the originate query. The query reformulation algorithms have been extensively studied in [4, 13, 14]. Our system adopts the algorithm in [14] which produces maximally contained rewritings. We extend this algorithm to support semantic data defined with ontology by adding the ontological containment defined in the TBox, such as *subsumption* relationship of classes and properties, *equivalent* class and property, *inverse* and *transitive* property.

The semantic similarity measure helps to reduce the query transmission cost by avoiding query be forwarded to unrelated nodes. However, there is still unnecessary query traffic due to the existence of multiple paths between nodes. As mentioned, we cannot prune the paths due to the semantic heterogeneity. To solve this problem, we propose an efficient caching mechanism based on the semantic query containment checking. In this approach, each node maintains two caches: (1) *active_id_cache* which stores the IDs of active queries (query's Time to Live (TTL) is not expired yet), (2) *query_cache* that caches the best rewrites of all historically popular queries and their corresponding results. The algorithm of query processing with containment-based caching is illustrated with the pseudocode in Fig. 5.

The original query ID is passed along the query reformulation paths together with the rewrites. Each node remembers the current active queries (in *active_id_cache*) and a set of best rewrites for a distinct original query (in *query_cache).* The evaluation of the new incoming query is based on the status of the query: if it is a new query or an existing active query, if it can be found in the *query_cache*, or it is contained by queries in the *query_cache*, or it contains or overlaps with queries in the *query_cache*. The operations we take may vary from ignoring the query, returning cached results, processing and then returning the cached results, decomposing the query and evaluating the sub-queries, and evaluating the query from scratch. The detail of the evaluation is presented in Fig. 5. We can see that the query evaluation process is integrated with caching. Our later experiments will demonstrate that this query optimization strategy dramatically reduces the query traffic.

```
query_processing (q)
    {
    if q can be found in active_id_cache, i.e., it is an active query
        if q is in the query_cache or q is contained by a query in the query_cache
            ignore the query, return
        else if     q contains or overlaps with queries in the query_cache
            determine if q should be decomposed, and decompose q accordingly
    else
        put q.id to active_id_cache
        if q is in the query_cache
            return the results cached
        if q is contained by a query in the query cache then
            get the result by using constraint on the cached result, return
        else if     q contains or overlaps with queries in the query_cache
            determine if q should be decomposed, and decompose q accordingly
    find results from local dataset for query q(or q's sub-queries)
    forward q(or q's sub-queries) to semantically related neighbors
    put q into the query_cache, replacement policy may be used
    }
```

Fig. 5. Optimized query evaluation

3.5 Result collecting and cache management

When returning results, our goal is to effectively integrate internal results and eliminate duplicated results from different sources while preserving good response latency to query requestors. If a query request is successfully evaluated, the results will be passed back to the upstream requestor. A node, when receiving the results for a certain query rewrite *q*, would update its cache, so that future queries can take advantage of it. The newly discovered results are added into the corresponding record of the resulting view cache. The cache content is maintained by certain replacement policies, taking the query frequency, temporal locality, evaluation cost and view sizes into consideration. The node also remembers quality routing paths, which is a determined by metrics such as the amount of results returned and the time used. The routing cache is updated dynamically according to the path quality, so that the system can forward the queries only to the best paths. Our simulations show that the network traffic can be dramatically reduced while preserving good response latencies for the query issuers.

In addition, we propose a batch process scheme, in which, when a node receives a result, it does not transmit it immediately, but waits for a brief period to see whether it receives results for the same query again. If it does get multiple results from different paths, it will integrate the results and forward the intermediate results to its requesting neighbors. Determining a suitable waiting period is complex: it needs to incorporate neighborhood knowledge and catching statistics.

4 Experiment

In this section, we will explain the experiment setup, and then present the simulation results.

4.1. Setup

As it is difficult to find representative real world ontology data, we have chosen to generate test data artificially. Our data does not claim to model real data, but shall rather provide reasonable approximation to evaluate the performance of the system. Ontology data can be characterized by many factors such as the number of classes, properties, and individuals; thus we have generated the test data in multiple steps. The algorithm starts with generating the ontology schema (TBox). Each schema includes the definition of a number of classes and properties. The classes and properties may form a multilevel hierarchy. Then the classes are instantiated by creating a number of individuals of the classes. To generate an RDF instance triple *t*, we first randomly choose an instance of a class *C* among the classes to be the subject: *sub(t).* A property *p* of *C* is chosen as the predicate *pre(t)*, and a value from the range of *p* to be the object: *obj(t).* If the range of the selected property *p* are instances of a class *C'*, then *obj(t)* is a resource; otherwise, it is a literal.

The queries are generated by randomly replacing parts of the created triples with variables. For our experiments, we use single-triple-queries and conjunctive-triple-queries. To create the conjunctive-queries, we randomly choose a property p_1 of class C_1. Property p_1 leads us to a class C_2 which is the range of p_1. Then we randomly choose a property p_2 of class C_2. This procedure is repeated until the range or the property is a literal value or we have created *n* ($n \leq 3$) triple patterns.

To control the ontology heterogeneity, we use a small-sized vocabulary set to generate the ontology data; we fix the mapping relation to the *equivalentClass* relation and ignore all other mapping relations. We do not do knowledge reasoning. In other words, we do not augment the RDF graph by inference (forward chaining). The total number of distinguished ontology schema is 20. We assume each node uses 1 to 3 ontologies. Each ontology includes at most 10 classes. The number of properties that each class has is at most k=3. The number of instances of each class at each peer is less than 10. Finally, the number of triple patterns in each query we create is either 1 or 3.

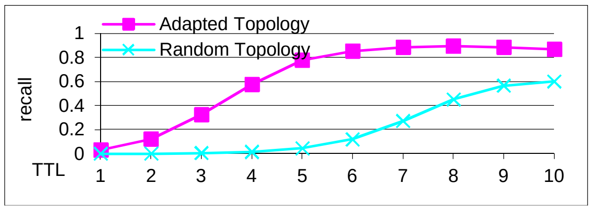
The simulation is initialized by injecting nodes one by one into the network until a certain network size has been reached (default network size is 1000). The network topology created this way has power-law properties; nodes inserted earlier have more links than those inserted later. This property is consistent with the real world situation, in which nodes with longer session time have more neighbors. After the initial topology is created, a mixture of *joins*, *leaves*, and queries are injected into the network based on certain ratios. We can adjust the query intensity by changing the ratio of query to join and leave (i.e., churn). In a query intensive system, the query frequency is higher. For example, in our experiment, at any time, the possibility of a node to issue a query is 20% and the possibility of a node to change its status (from on to off or vice versa) is 2% which represents a dynamic and query intensive scenario. For example, in a network with 1000 nodes, at any given time, there are on average 200 queries issued and 40 nodes change their status. The proportion of join to leave operations is kept the same to maintain the network at approximately the same size. Inserted nodes start functioning without any prior knowledge.

4.2. Result

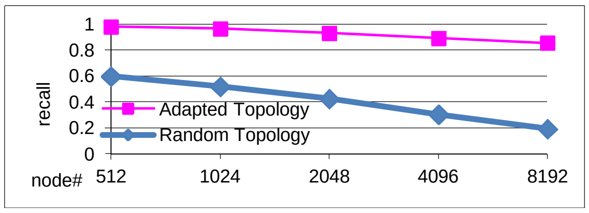
As discussed, the topology of the network is a crucial factor determining the efficiency of the query system. We start the evaluation of the query evaluation system by first recording the effectiveness of our semantics-based topology adaptation algorithm. We compare the query performance of a semantics-based topology with that of a semantics-free random topology. The performance measurement is based on the metric of recall rate which is defined as the number of results returned divided by the number of results actually available in the network.

Fig. 6 depicts the findings of our first set of simulations. As it can be seen, query evaluation based on our adapted topology performs better as measured by recall rate. Semantics-based topology effectively reduces the search space, and its ontology summary guides the query in the right direction. Therefore, the query system can locate results faster and more accurately with this topology. This explains why our topology scales to large network size and why it achieves higher recall with smaller TTL. Besides all these reasons, another factor contributing the overall better recall rate of our topology is that it is able to locate semantically related results that cannot be located by the random topology. Because of the semantic heterogeneity of our experimental setup, relevant resources may be represented with different ontologies. System based on the semantic topology may use its ontology summary to find semantically related nodes and use the mapping defined to translate the query. Therefore, it can locate most of the relevant results. However, for unstructured random topology, they have no way to find semantically related resources. Therefore, they can only locate resources represented in the same ontology as the ontology of the querying node.

Simulations also were carried out to validate and characterize the performance of the proposed query routing scheme. First, we demonstrate the performance of our semantics-based greedy query forwarding strategy by comparing it with semantics-free query forwarding scheme. In this experiment, we vary the average number of ontology schemas a node can pick, and then we evaluate the bandwidth of solving all the queries. As shown in Fig. 7, our semantics-based query forwarding reduces the bandwidth dramatically, especially when the grid is more semantically heterogeneous. This is because we only forward the query to related neighbors, thus saving unnecessary traffic.



(a) Comparison of query efficiency (recall vs. TTL)



(b) Comparison of system scalability (recall vs. network size)

Fig.6. Effect of topology adaptation on query performance.

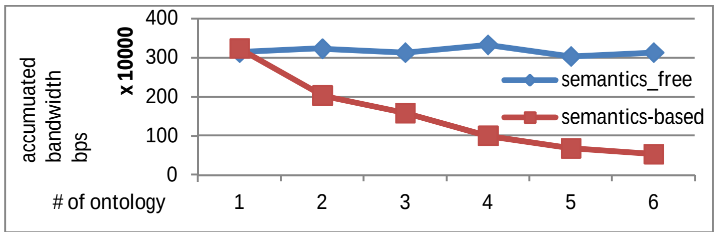


Fig.7. Comparison of query overhead (accumulated bandwidth vs. # of ontology schema per node)

Moreover, we show that our containment-based caching improves the query performance by reducing not only query traffic but also query latency. Fig. 8 and Fig. 9 demonstrate these two aspects respectively. In this experiment, we increase the skew degree of the query distribution from random to Zipf [33] with α=1.25. From Fig. 8 and Fig. 9 we can get two conclusions: (1) Caching, especially containment-based caching significantly reduces the query traffic and query latency. (2) Query distribution has a significant impact on the performance of caching. The more skewed the query distribution, the more effective the caching performs. According to [2], in an open and live distributed environment, query distribution is skewed and follows a Zipf distribution. Therefore, our caching scheme would be an effective strategy to improve the system performance.

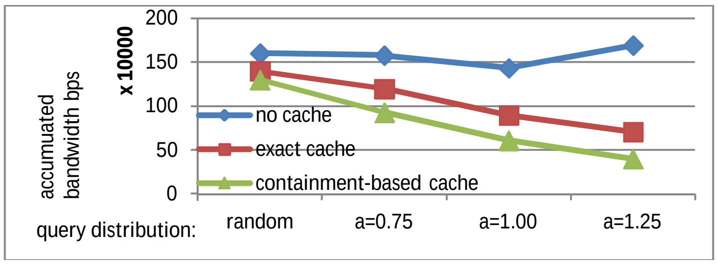


Fig.8. Performance of caching (accumulated bandwidth vs. query distribution)

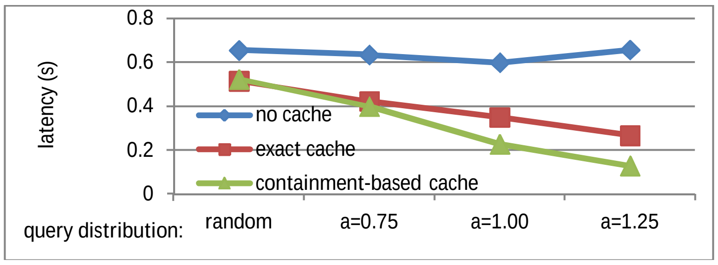
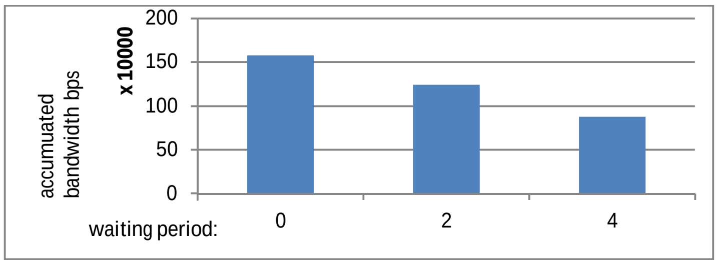
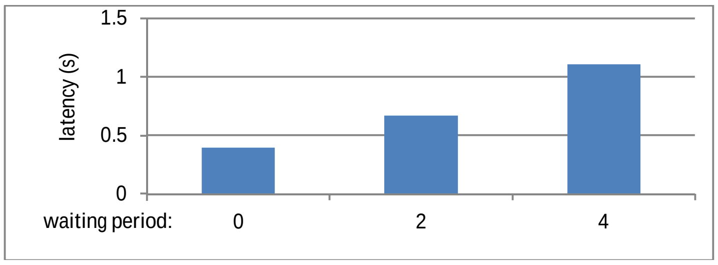


Fig.9. Performance of caching (query latency vs. query distribution)



(a) Query overhead vs. waiting period



(b) Query latency vs. waiting period

Fig.10. Effect of batch query collecting

Finally, we evaluate the performance of the batch processing scheme on result collecting. From Fig. 10, we can see that adding a waiting period for result collecting does reduce the result collecting overhead; however, on the other hand, it increases the response time. An application should choose an appropriate waiting period according to its special need.

5 Conclusions

The main contribution of this paper is to present a bandwidth-efficient framework for query evaluation in a large-scale and fully decentralized heterogeneous grid network. In this framework, we organize nodes' topology according to their semantic distances, so that queries can be focused in semantically related regions only. To reduce redundant queries and results from different semantic paths, a view-based semantic caching mechanism is proposed. Query results and routes are cached for future use. Semantic containment and rewriting are introduced to match semantically similar queries with cached views. Our simulation results show that these optimization techniques dramatically reduce bandwidth cost and improve query response time.

Important problems in large-scale distributed query evaluation remain to be solved. We identify several limitations of our work and research directions for future work. First, the overhead of constructing the semantics-based topology is relatively high. It involves searching the WordNet dictionary. If the users' devices have limited computing capacity or power, this overhead can be prohibitively high. Therefore, one important issue we are going to address is to further simplify the semantic similarity measure to minimize the computation cost. Moreover, we are going to improve the query evaluation process by integrating indexing and caching prediction. Furthermore, in the current system, query results are returned to requesters without using any ranking mechanisms. We plan to investigate the result-ranking problem, so that query results can be ordered based on relevance and importance for users.

References

[1] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, "The Chatty Web: Emergent Semantics Through Gossiping,"of the 12th international conference on World Wide Web, 2003

[2] L. Adamic, B. Huberman, "Zipf's law and the Internet" Glottometrics, 2002

[3] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider: The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press, Cambridge, UK, 2003. ISBN 0-521-78176-0

[4] C. Beeri, A. Halevy, and M.C. Rousset. Rewriting Queries Using Views in Description Logics. In PODS'97.

[5] P. A. Bernstein, F.Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L.Serafiniand I. Zaihrayeu, "Data Management for peer-to-peer computing" A vision." "Proceedings of the fifth International Workshop on the Web and Database, 2002.

[6] M. Bender, T. Crecelius, M. Kacimi, S. Miche, J. Xavier Parreira, and G.Weikum. Peer-to-peer information search: Semantic, social, or spiritual? IEEE Bulletin of Computer Society Technical Committee on Data Engineering, 30(2):51--60, 2007.

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

[8] 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.

[9] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury and S. Tuecke, The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. J. Network and Computer Applications, 2001.

[10] Y. Chawathe, S. Ratnasam, L. Breslau, N. Lanhan, S. Shenker, "Making Gnutella-like P2P Systems Scalable", In Proceedings of ACM SIGCOMM'03, 2003.

[11] C. Fellbaum. "WordNet: An Electronic Lexical Database" In Fellbaum, Christiane, MIT Press, 1998.

[12] I. Foster, C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers.

[13] A.Y. Halevy, Answering queries using views: A survey. The VLDB Journal 10, 4 (2001).

[14] A. Halevy, Z. Ives, J. Madhavan, P. Mork, D. Suciu. "The Piazza Peer Data Management System"

[15] K. Hosea, A. Rothb,A. Zeitzc,K. U. Sattlera,and F. Naumannb, A research agenda for query processing in large-scale peer data management systems, Information Systems, Volume 33, Issues 7-8, November-December 2008, Pages 597-610

[16] J. Jiang and D. Conrath, "Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy," in Proceeding of the Int'l Conf. Computational Linguistics, 1997.

[17] O. Lassila and Ralph R. Swick, "W3C Resource Description framework (RDF) Model and Syntax Specification", World Wide Web Consortium, 1999.

[18] J. Lee, M. Kim, and Y. Lee, "Information Retrieval Based on Conceptual Distance in IS-A Hierarchies," J. Documentation, vol. 49, pp. 188-207, 1993.

[19] J. Li, S. Vuong, "Efa: an Efficient Content Routing Algorithm in Large Peer-to-Peer Overlay Networks", in Proceedings of the Third IEEE International Conference on Peer-to-Peer Computing", 2003.

[20] J. Li and S. Vuong, "SOON: A Scalable Self-Organized Overlay Network for Distributed Information Retrieval", in Proceedings of the 19th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management Managing Large Scale Service Deployment 2008.

[21] D. Lin. An information-theoretic definition of similarity. In Proc. 15th International Conf. on Machine Learning, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[22] E. W. Nejdl, W. Siberski an M. Sintek (2003), " Design Issues and Challenges for RDF an schema-based peer-to-peer systems." ACM SIGMOD Record 32(3):41-46.

[23] W.S. Ng, B.C. Ooi, K.L. Tan, A Zhou, "Peerdb: A p2p-based system for distributed data sharing", Proceedings of the International Conference on Data Engineering, 2003.

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

[25] J. Perez, M. Arenas, C. Gutierrez: "The semantics and complexity of SPARQL." In Proceedings of the 5th International Semantic Web Conference, 2006.

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

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

[27] M. A. Rodriguez, M. J. Egenhofer, "Determining Semantic Similarity Among Entity Classes from Different Ontologies". IEEE Transactions on Knowledge and Data Engineering, 2003.

[28] A. Rowstron and P. Druschel. "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," in Proc. of the IFIP/ACM CDSP, Middleware, 2001.

[34] D De Roure, NR Jennings, NR Shadbolt, "The semantic grid: A future e-science infrastructure", Grid Computing-Making the Global Infrastructure a Reality, 2003.

[29] 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

[30] Z. I. Tatarinov, J. Madhavan, A. Halevy, D. Suciu, The Piazza Peer Data Management System, ACM Sigmod Record, 2003

[31] A. Tversky. Features of similarity. Psychological Review, 84(4):327–352, 1977.

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

[33] G. K. Zipf, "Human Behaviour and the Principle of Least-Effort", Addison-Wesley, Cambridge MA, 1949