

Efficient Data Sharing over Large-Scale Distributed Communities

Juan Li, Samee Ullah Khan, Qingrui Li, Nasir Ghani
Nasro Min-Allah, Pascal Bouvry, and Weiyi Zhang

Abstract Data sharing in large-scale Peer Data Management Systems (PDMS) is challenging due to the excessive number of data sites, their autonomous nature, and the heterogeneity of their schema. Existing PDMS query applications have difficulty to simultaneously achieve high recall rate and scalability. In this chapter, we propose an ontology-based sharing framework to improve the quality of data sharing and querying over large-scale distributed communities. In particular, we add a semantic layer to the PDMSs, which alleviates the semantic heterogeneity and assists the system to adjust its topology so that semantically related data sources can be connected. Moreover, we propose a comprehensive query routing and optimization

Juan Li

Department of Computer Science, North Dakota State University, Fargo, USA, e-mail: j.li@ndsu.edu

Samee Ullah Khan

Department of Electrical and Computer Engineering, North Dakota State University, Fargo, USA, e-mail: samee.khan@ndsu.edu

Qingrui Li

Department of Computer Science, North Dakota State University, Fargo, USA, e-mail: qingrui.li@ndsu.edu

Nasir Ghani

Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, USA, e-mail: nghani@ece.unm.edu

Nasro Min-Allah

Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan, e-mail: nasar@comsats.edu.pk

Pascal Bouvry

Faculty of Sciences, Technology and Communications, University of Luxembourg, Luxembourg, e-mail: pascal.bouvry@uni.lu

Weiyi Zhang

Department of Computer Science, North Dakota State University, Fargo, USA, e-mail: weiyi.zhang@ndsu.edu

scheme to enable scalable and efficient query evaluation. Simulation studies reveal that our proposed ontology-based data sharing framework effectively improves the query performance in terms of recall and scalability.

1 Introduction

The continued growth of computational power and communication technologies is giving rise to a new information era. In this era, the broad availability of data coupled with increased capabilities and decreased costs of both storage and computing technologies enable users to share data and knowledge on an unprecedented scale. In a large distributed community, data are geographically distributed and owned by different organizations. The fact that users typically have little or no knowledge of data contributed by other participants in a large community poses a significant obstacle to their use. For this reason, distributed data sharing is a vital part of a distributed community, and an efficient data sharing infrastructure is crucial to make the distributed information available to users in a timely and reliable manner. However, data sharing in large-scale communities is very challenging due to the potential large amounts of data, diverseness, distributed arrangement, and dynamic nature. In addition, it is equally difficult to integrate the information sources with heterogeneous representation formats. This compels us to rethink how one must manage, i.e., store, retrieve, explore, and analyze this abundance of data. An effective data sharing solution for this environment must meet the following requirements:

Scalability: Because of the participation of Personal Computers (PCs) and devices, such as sensors at the periphery of the Internet, the number of nodes and data involved in a community may grow from hundreds to tens of thousands. The aforementioned raises the problem of potential performance degradation. Consequently, applications must be scalable in handling the vast number of nodes and data involved in the community deployment. The system must be capable of spreading load among the participants themselves and serving large number of users at a very low cost.

Interoperability: Different data may belong to different organizations and may have heterogeneous schema. The same data in different nodes can be defined differently. For example, one node may use the term processor to represent a computing resource, while another node may use CPU for description. On the other hand, the same term can also be used to represent different resources. For example, the term mouse might represent a computer mouse in one node, and a rodent in another. There are other relationships between concepts that also need to be considered. For instance, a query related to operating systems should be able to understand that Windows, UNIX, and Macintosh are different types of operating systems. It is also important for the sharing scheme to determine the relationships among concepts and locate all related resources; however they are represented. Therefore, the system must have a highly expressive logic with sophisticated reasoning capabilities.

Appropriate data schema integration strategies must be used to reconcile the semantic difference.

Robustness: PCs are not as stable as servers in classical clusters. With so many community data sources, the probability of some data failing is quite high. Therefore, node and data failures are the rule rather than the exception. In addition, data and their status change frequently. The system must adapt its behavior dynamically and should be able to provide reliable services in a highly dynamic environment, locating data when they are present and meeting the requirements of the application.

Responsiveness: The response time includes network latency for propagating query over the network and the accumulated query processing time at each node. Response time is a key issue for data query applications. An effective data query system must respond to users quickly and efficiently with up-to-date information. In this chapter, we propose an ontology-based data sharing framework that aims to satisfy all of the aforementioned requirements. The proposed framework works in a fully decentralized and scalable way. It enables efficient sharing and collaborating over a large-scale community composed of geographically distributed data providers.

The rest of the chapter is organized as follows. Section 2 introduces the related work. Section 3 gives an overview of the system design. Section 4 presents the scheme to cluster nodes based on their semantics. Section 5 describes the query forwarding strategy. In Section 6 we evaluate the proposed method with a comprehensive set of simulations. Concluding remarks are provided in Section 7.

2 Related Work

Our proposed system is based on existing research on Peer Data Management Systems (PDMS) [3, 2]. PDMSs were proposed as the extension to distribute databases within the context of P2P systems. They are built of multiple peers, each of which provides a schema and accepts queries against the schema. Schema mappings are established between neighbor peers forming the basis for query answering. Queries are propagated and answered by all peers that can be reached through mappings over the neighbors. However, as mentioned, the schema mapping and neighbor forwarding strategy used by most PDMSs have serious problems such as inflexibility and low recall rate caused by strict schema mapping, information loss caused by query rewriting, long delay and large overhead caused by a large network population. Peer clustering [12, 21] has been proposed to reduce the query space and the semantic degradation caused by traversal of the semantic mappings. In [21], which is more similar to our approach, the authors propose a clustering process that extend classical clustering algorithm to distributed environment to group peers which are semantically related. Our proposed semantics-based topology adaptation, instead, incrementally cluster peers when peers join the system. The system also dynamically adjusts the topology to adapt to the evolving network. Unlike [21], our clustering does not need to broadcast in clusters and the computation is much simpler. There-

fore, it is easier to scale. Semantic web technologies are being used in PDMS to add intelligence and reconcile the semantic heterogeneity problem. In [5] the authors use the ontology mapping to facilitate schema mapping. However, how to locate semantically relevant peers to map is unspecified in the paper, and the scalability is still a problem. In [20] there is a distributed hash table (DHT) used [22, 19, 1, 4] approach to index peer's ontology URI to expedite the ontology discovery. However, indexing URI in fact does not consider the semantic meanings, therefore, cannot address the semantic similarity problem.

3 System Overview

3.1 Problem Description

A sharing community consists of autonomous nodes that store and manage structured relational data locally. Each data source has its own data schema. A node joins the community by connecting to one or more existing nodes (acquaintances) in the community. As pointed out by Tatarinov et al. [9], it is neither practical to maintain a global schema for all data sources to map their local schemas with, nor is it feasible for a data source to map its schema with all other sources. A realistic approach is to let each provider connect to and construct mapping with one or a few acquaintances, then queries can be translated between different schemas. Therefore, most of the current PDMSs [3, 2] use peer mappings and some suitable rewriting algorithms to propagate queries between acquainted peers. However, this strategy may cause some problems:

- The nature of structured data stored in the overlay enforces strict methods of querying and query rewriting. Frequently, the user intends to obtain information that is semantically relevant to the posed query, rather than information that strictly complies with structural constraints.
- Query rewriting based on multiple pair-wise schema mapping may cause information loss especially when the mapping pairs are not semantically similar enough.
- For a large-scale network, query has to be forwarded through a large number of peers, which is obviously unscalable.

3.2 A Multilayered Semantic Sharing Scheme

In this section, we address the aforementioned problems by exploiting the benefits provided by semantics through ontology. Ontology is defined as "an explicit specification of a conceptualization" [7]. Ontologies are (meta) data schemas, providing a controlled vocabulary of concepts, each with an explicitly defined and machine-

processable semantics. By defining shared and common domain theories, ontologies help both people and machines to communicate concisely, supporting the exchange of semantics and not only syntax. In general, an ontology of a domain consists of the four major components listed below:

- Concepts: Concepts of a domain are abstractions of concrete entities de-ri-ved from specific instances or occurrences.
- Attributes: Attributes are characteristics of the concepts that may or may not be concepts by themselves.
- Taxonomy: Taxonomy provides the hierarchical relationships between the concepts.
- Non-taxonomic relationships: Non-taxonomic relationships specify non-hierarchical semantic relationships between the concepts.

Along with the above four components, ontologies may also consist of instances of each of the concepts, and inference rules of the domain. Fig. 1 illustrates a portion of the ontology for a Computer Science Department. It indicates some of the major concepts and their hierarchical relationships only. In general, the ontology includes non-taxonomic relationships as well, for example: *Student - take - Course*, *Professor - teach - Course*, *Worker - work - Department*.

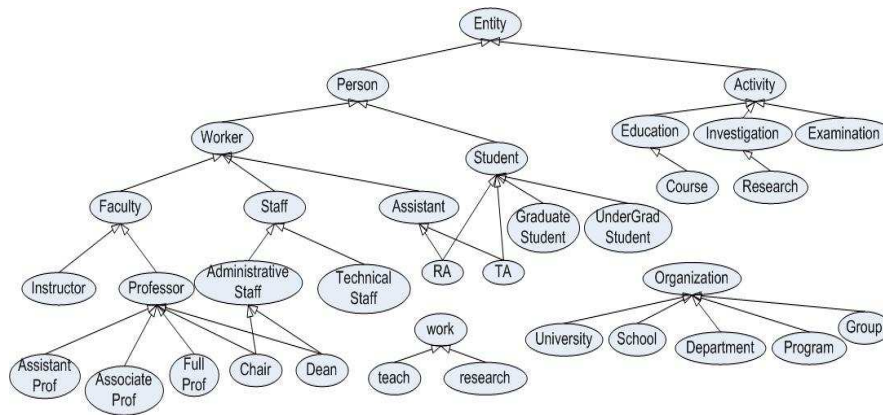


Fig. 1 An example of an ontology snippet

By defining shared and common domain theories, ontologies help both people and machines to communicate concisely, supporting the exchange of semantics and not only syntax. We assume that there exist domain ontologies shared in the community.

As shown in Fig. 2, we extend the traditional PDMS structure with a two-level semantic dimension. The ontology layer utilizes the understanding of domain knowledge to model the schema of data sources, which enables more flexible peer mapping at the semantic-level. Semantic-level mapping overcomes the limitation of relational schema mapping and is able to map with more semantically related peers.

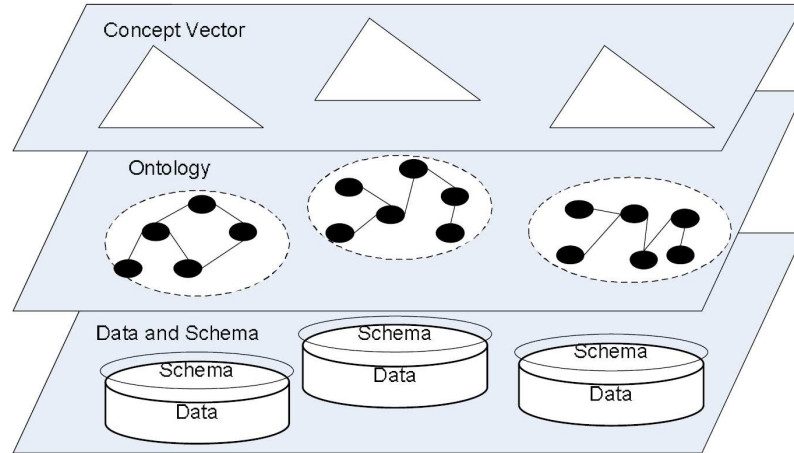


Fig. 2 Layered architecture of data sharing

The concept layer is a summarization of ontology. The concept layer simplifies the computation of semantic closeness between peers. This makes clustering semantically related peers much easier. Connecting semantically related nodes can reduce information loss caused by peer-wise query rewriting. Last but not least, query forwarding with semantic guidance may intelligently propagate queries to the right direction, thus reducing bandwidth used for forwarding queries and increasing recall rate. We present how semantics improves the system performance from these aspects in the following sections.

3.3 From Schema to Ontology

As mentioned, we assume that there exist domain ontologies shared by members of the community. A data source can map its local schema to one or more ontologies. These mappings are peer-specific and may be constructed manually or semi-automatically. Our principles of mapping the relational schema to ontology are based on relational schema design. Specifically, we analyze Entity-Relationship (ER) models that support entities with attributes, relationships, and inheritance hierarchies between entities. We reverse the process of translating ER model to relational model. In this way, we convert entities, keys, foreign keys to entities and relationships (including hierarchical relationship) in ontology.

Fig. 3 depicts a simplified example of this mapping, in which a relational schema presenting information about "professor" is mapped to an ontology that contains domain concepts about university, course and faculty. For simplicity, the ontology graph only shows part of the ontology that is related to the schema.

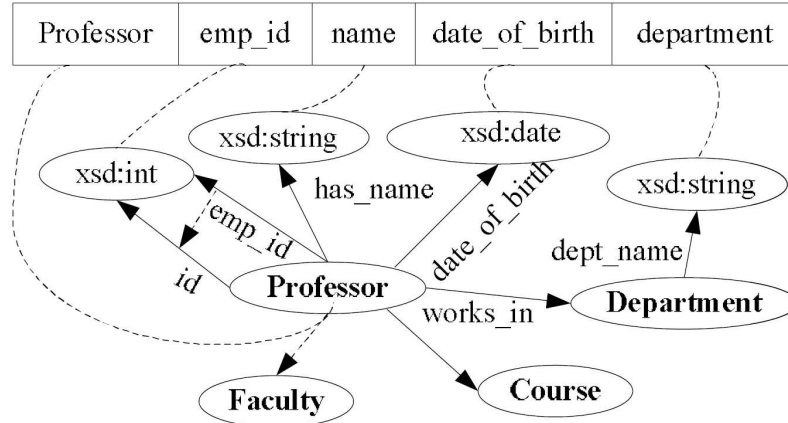


Fig. 3 Mapping from schema to ontology

3.4 Semantic Similarity

With the schema to ontology mapping defined, measuring the similarity between data sources can be converted to a problem of evaluating the similarity (or distance) between (sub-)graphs of the ontology graphs.

We adopt the idea of distance-based approach [15, 23, 18, 10] to measuring the semantic similarity. The basic idea behind the distance-based approach is to identify the shortest path between two concepts in terms of the number of edges in the ontology graph and then translate that distance into semantic distance. Our approach improves the accuracy by integrating factors, such as the depth of a node in the ontology hierarchy and the type of links. The semantic distance between two concepts is defined as a "relative" distance between nodes and their common ancestors in an ontology graph, thus it integrates the edge weight with the depth and the length of the shortest path. Given two ontologies that are summarized as collections of concepts, the similarity between these two ontologies is the normalized similarity of all related pairs of concepts. The similarity defined this way is not a symmetric relation. If the semantic similarity from A to B is larger than a user-defined similarity threshold t ($0 < t \leq 1$), ontology A is said to be semantically related to ontology B . Note that the measurement defined above can be extended to evaluate the similarity of multiple ontologies. The idea is to place the concepts of the ontologies into a global ontology - WordNet [6], then the problem turns to a similarity measurement of same ontology again. Due to the lack of space, we are restricted to provide only surface-level information pertaining to semantic similarity measurement. We encourage the readers to find more details about our pro-posed solution by referring to [10].

4 Semantics-based Self-Clustering

The establishment of schema mappings and query answering in PDMSs can be improved if semantically related peers are grouped together in the community network. The objective of semantic clustering is to make the system's dynamic topology match the semantic clustering of peers. This would allow queries to be quickly propagated among relevant peers. In addition, this adaptation allows semantically related peers to establish ontology mapping relationships to reduce information loss. 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 [16]. (2) Connecting and constructing schema mapping with semantically similar nodes will reduce information loss caused by query reformulation [13]. In our system, semantic clustering 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 experiences, so that its topology always reflects its changing interest and data contents.

4.1 *Joining the Right Semantic Cluster*

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 neighbor-discovery query routing is a process of greedily forwarding the query to neighbors with highest similarity. A neighbor-discovery query message includes several parts: (1) the querying node's compressed concept vector, (2) a similarity threshold which is a criterion to determine if a node is semantically related to the query (optional), (3) a query Time To Live (TTL) to gauge how far the query should be propagated.

When a node N receives a neighbor-discovery query Q which tries to find neighbors for a new joining node X , N computes the semantic similarity between X and itself. If N is semantically related to X , N will send a Neighbor Found reply to X . If the query's TTL has not expired, N computes the semantic similarity between X and each of N 's neighbors, and forwards the query to semantically related neighbors. If no semantically related neighbors are found, the query will be forwarded to a few "outsiders" of the cluster. The query will propagate with this strategy until enough matches have been located or TTL expires.

4.2 *Dynamic Self-adjusting*

Because of the dynamic property of the large-scale network and the evolution of nodes' ontology property, neighbor discovery for a node is not once and for all,

but rather the first-step of the topology adaptation scheme. A node should keep updating its neighbor links according to its query experiences and other peer's recommendations. A peer obtains experiences by accumulating queries received as a query forwarding router, and query results collected as a query requestor. Based on its experiences and peer recommendations, 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.

Once a node joins the right cluster, i.e., it connects to the semantically related nodes, it can establish appropriate ontology mappings with its neighbors manually, semi-automatically or automatically. The ontology mapping is beyond the scope of this chapter. Readers can refer the state of the art in this field [17, 14] for more information.

5 Query Evaluation

With the semantics-based topology constructed, query can be evaluated within the scope of querying node's local cluster most of the time, because queries reflect the querying node's ontology interest, and semantically related nodes are within the neighborhood of the querying node. The semantic clusters reduce the search time and decrease the network traffic by minimizing the number of messages circulating among clusters. However, how to minimize query cost inside clusters is still a challenging issue. In this section, we propose a comprehensive query routing and optimization scheme to address the challenges of semantic query evaluation.

5.1 Problems of Query Evaluation

In a distributed and unstructured network, queries are evaluated by forwarding between neighbors. Because there are multiple paths between nodes, there are a large number of duplicated messages propagated in the network. There are many technologies [24, 11] to control duplicated messages by cutting flooding paths. However, in a schema-heterogeneous PDMS network, optimization cannot be done by simply cutting the flooding paths. In such environments, queries are forwarded and translated according to the mappings between neighbors. Query rewrites based on different mappings can come to the same nodes by different paths. 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 order to solve this problem, we utilize the semantic similarity metric proposed in Section 3.4 to guide query forwarding.

Moreover, we propose a containment-based caching strategy to reduce unnecessary query traffic.

5.2 Semantics-based Forwarding

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 ($q.originator$) and the ID of the previous hop ($q.lastHop$) that passes the query to the current node, and the SQL 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 semantically related neighbors (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 [3, 8]. Our system adopts the algorithm in [3] which produces maximally contained rewritings. We extend this algorithm to support semantic data defined with ontologies by adding the ontological containment, such as subsumption relationship of classes and properties, equivalent class and property, inverse and transitive property.

Since the ontology mapping between two nodes rarely maps all concepts in one node to all concepts in the other, mappings typically lose some information and can be partial or incomplete; the reformulated query may deviate from the original query's intention, and the query result should be evaluated at the querying node. Feedback on query results can be used to improve the quality of mappings. Moreover, nodes can learn from query results to update their neighbors. Therefore, when a node updates its semantic interests, the system is able to adjust that node's links accordingly.

5.3 Containment-based Caching

To further reduce the query transmission cost, 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 Alg. 1.

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 evalu-

Algorithm 1 Algorithm of optimized query evaluation

```

1: if  $q$  can be found in active_id_cache, i.e., it is an active query then
2:   if  $q$  is in the query_cache or  $q$  is contained by a query in the query_cache then
3:     ignore the query and return ;
4:   else if  $q$  contains or overlaps with queries in the query_cache then
5:     determine if  $q$  should be decomposed, and decompose  $q$  accordingly;
6:   end if
7: else
8:   put  $q.id$  to active_id_cache;
9:   if  $q$  is in the query_cache then
10:    return the results cached;
11:   end if
12:   if  $q$  is contained by a query in the query_cache then
13:    get the result by using constraint on the cached result;
14:    return;
15:   else if  $q$  contains or overlaps with queries in the query_cache then
16:    determine if  $q$  should be decomposed, and decompose  $q$  accordingly;
17:   end if
18: end if
19: find results from local dataset for query  $q$  (or  $q$ 's sub-queries);
20: forward  $q$  (or  $q$ 's sub-queries) to semantically related neighbors;
21: put  $q$  into the query_cache, replacement policy may be used

```

ation 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 Alg. 1. We can see that the query evaluation process is integrated with caching. Our later experiments demonstrate that this query optimization strategy dramatically reduces the query traffic.

6 Experiment

In this section, we present the experimental results which demonstrate the performance of our searching scheme.

We generated 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. In our experiments, the total number of distinguished schema is 100. We assume each data sites uses 1 to 3 schemas. The simulation is initialized by injecting nodes one by one into the network until a certain network size has been reached. After the initial topology is created, queries are injected into the network based on certain ratios. Inserted nodes are assumed to have one or more acquaintances.

We expect that our self-clustering scheme will transform the topology into semantics-based communities. To verify this transformation, we examine an important network statistic, the *clustering coefficient*. The *clustering coefficient* (CC) is a measure of how well connected a node's neighbors are with each other. According to one commonly used formula for computing the *clustering coefficient* of a graph (Eq. 1), the *clustering coefficient* of a node is the ratio of the number of existing edges and the maximum number of possible edges connecting its neighbors. The average over all $|V|$ nodes gives the *clustering coefficient* of a graph (Eq. 2).

$$CC_v = \frac{\# \text{ of edges between } v\text{'s neighbors}}{\text{maximum } \# \text{ of edges between } v\text{'s neighbors}} \quad (1)$$

$$CC = \frac{1}{|V|} \sum_v CC_v \quad (2)$$

Fig. 4 shows plots of the *clustering coefficient* as a function of the number of nodes in the network. We observe that the *clustering coefficient* of our semantic clustering is much larger than that of the random power-law network. This indicates the emergence of clusters in the network.

Semantic topology of the network is a crucial factor determining the efficiency of the query system. We measure the query evaluation system by recording the effectiveness of our semantic cluster-based topology. We compare the query performance of a semantic cluster 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.

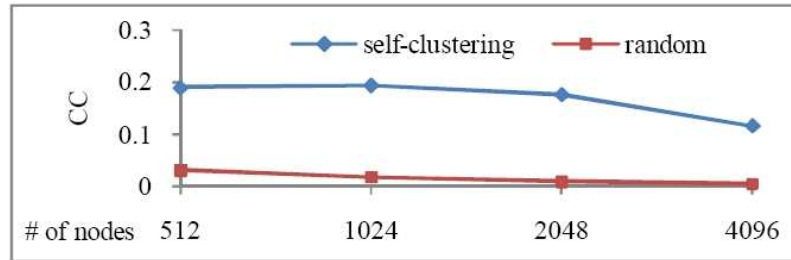


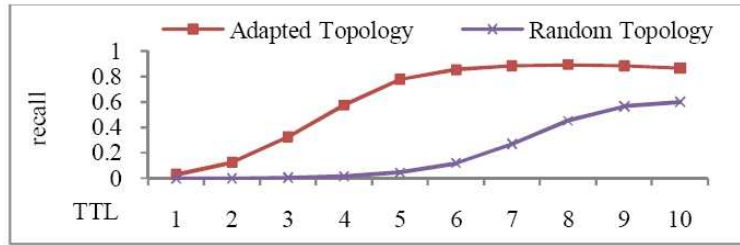
Fig. 4 Comparison of clustering coefficient

Fig. 5 depicts the findings. As it can be seen, query evaluation based on our adapted topology performs better as measured by recall rate. Semantic clusters effectively reduce 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

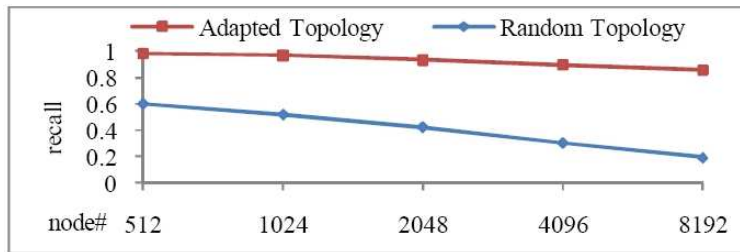
is able to locate semantically related results that cannot be located by the random topology. Because of the semantic heterogeneity of our experimental setup, relevant resources may be represented with different ontologies. The system using the semantic topology may use its ontology summary to find semantically 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.

Moreover, we show that our containment-based caching improves the query performance by reducing not only query traffic but also query latency. Fig. 6 and Fig. 7 demonstrate these two aspects respectively. In this experiment, we increase the skew degree of the query distribution from random to Zipf [25] with $\alpha=1.25$. From Fig. 6 and Fig. 7 we can get two conclusions:

- Caching, especially containment-based caching significantly reduces the query traffic and query latency.
- 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.



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



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

Fig. 5 Effect of topology adaptation on query performance

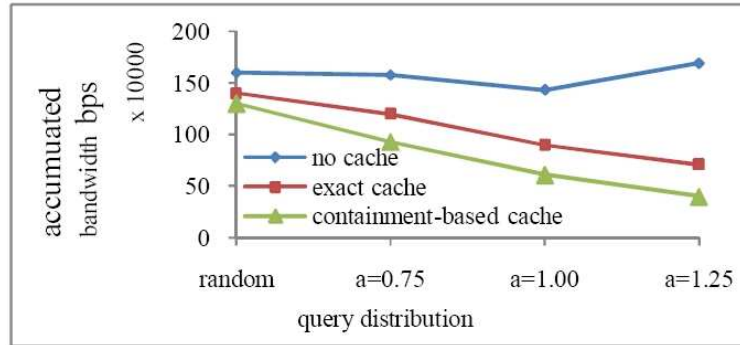


Fig. 6 Performance of caching (accumulated bandwidth vs. query distribution)

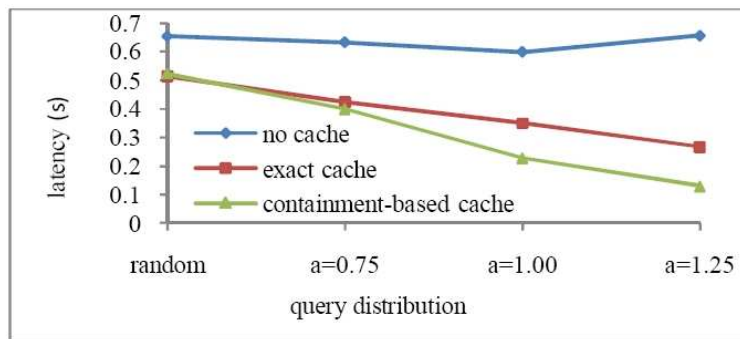


Fig. 7 Performance of caching (query latency vs. query distribution)

7 Conclusion

In this chapter, we presented strategies for efficient sharing and querying over large-scale distributed data sites. In particular, we extended the traditional PDMS with a semantic layer. This layer allows peers to locate semantically related data sources that are not reachable by traditional schema mappings. Based on this semantic layer, we proposed an efficient clustering mechanism and an intelligent query routing scheme, which dramatically improves the query recall rate and the scalability of the system.

As future work, we plan to further improve the aforementioned strategy by simplify the semantic similarity measurement for heterogeneous ontologies. Moreover, we plan to evaluate our system with real distributed data sets and deploy the system on large-scale distributed data sharing environments, such as a distributed medical community.

References

1. Chord: A scalable peer-to-peer lookup service for internet applications. In *In ACM SIGCOMM*, pages 149–160, 2001.
2. Data management in large-scale p2p systems. In *In Proc. of the International Conference on High Performance Computing for Computational Science*, pages 104–118, 2004.
3. J. Madhavan P. Mork D. Suci I. Tatarinov A. Halevy, Z. Ives. The piazza peer data management system. *IEEE Transactions on Knowledge & Data Engineering (TKDE)*, 16(7):787–798, 2004.
4. J. D. Kubiawicz B. Y. Zhao and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, UCB/CSD-01-1141, 2000.
5. T. Pachêco C. E. Pires, D. Souza and A. C. Salgado. Semantic-based ontology matching process for pdms. In *In Proc. of the in Globe, Vol. 5697, Springer LNCS*, pages 124–135, 2009.
6. C. Fellbaum D. Gross G. A. Miller, R. Beckwith and K. J. Miller. Introduction to wordnet: an on-line lexical database. In *In Proc. of the International Journal of Lexicography*, 1990.
7. T. R. Gruber. Principles for the design of ontologies used for knowledge sharing. *International Journal Human-Computer Studies*, 43(3-4):907–928, 1995.
8. A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4).
9. J. Madhavan A. Halevy D. Suci I. Tatarinov, Z. Ives.
10. S. U. Khan J. Li. Mobisn: Semantics-based mobile ad hoc social network framework. In *In Proc. of the IEEE Global Communications Conference (Globecom 2009), Honolulu, HI, USA*, pages 1–6, 2009.
11. S. Vuong J. Li. Efa: an efficient content routing algorithm in large peer-to-peer overlay networks. In *In Proc. of the Third IEEE International Conference on Peer-to-Peer Computing*, 2003.
12. K.U. Sattler K. Hose, C. Lemke. Processing relaxed skylines in pdms using distributed data summaries. In *In Proc. of the 15th ACM international conference on Information and knowledge management*, pages 425–434, 2006.
13. A. Zeitz K. Sattler and F. Naumann K. Hose, A. Roth. A research agenda for query processing in large-scale peer data management systems. *Information Systems*, 33(7-8):797–610, 2008.
14. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18:1–31, 2003.
15. J. Li and S. Vuong. Soon: A scalable self-organized overlay network for distributed information retrieval. In *In Proc. of the 19th IFIP/IEEE International Workshop on Distributed Systems (DSOM)*, pages 1–13, 2008.
16. M. Kacimi S. Mîche J. Xavier Parreira M. Bender, T. Crecelius 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.
17. H. Han N. Choi, I. Y. Song. A survey on ontology mapping. *ACM Sigmod Record*, 35(3):34–41, 2006.
18. E. Bicknell M. Blettner R. Rada, H. Mili. Development and application of a metric on semantic nets. *IEEE Transaction on Systems, Man, and Cybernetics*, 19(1):17–30, 1989.
19. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *In Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms*, 2001.
20. R. Heese S. Herschel. Humboldt discoverer: A semantic p2p index for pdms. In *In Proc. of the Workshop on Data Integration and the Semantic Web*, 2005.
21. F. Mandreoli R. Martoglia S. Lodi, W. Penzo and S. Sassatelli. Semantic peer, here are the neighbors you want! In *In Proc. of the 11th Extending Database Technology*, pages 26–37, 2008.
22. M. Handley R. Karp S. Ratnasamy, P. Francis and S. Shenker. A scalable content-addressable network. In *In Proc. of the ACM SIGCOMM*, pages 161–172, 2001.

23. J. Michelizzi T. Pedersen, S. Patwardhan. Wordnet: Similarity-measuring the relatedness of concepts. In *In Proc. of the 19th National Conference on Artificial Intelligence (AAAI)*, 2004.
24. L. Breslau N. Lanhan S. Shenker Y. Chawathe, S. Ratnasam. Making gnu-tella-like p2p systems scalable. In *In Proceedings of ACM SIGCOMM'03*, 2003.
25. G. K. Zipf. *Human Behaviour and the Principle of Least-Effort*. Addison-Wesley, Cambridge MA, 1949.