

An Interest-based P2P RDF Query Architecture¹

Qian Gao, Zhihuan Qiu, Yu Wu, Jing Tian, Yafei Dai

CNDS Lab, Peking University, Beijing, China

qq@net.pku.edu.cn

{qzh, wuyu, tianjing, dyf}@net.pku.edu.cn

Abstract. Currently most centralized RDF storage and query architecture have problems in query performance and scalability. In this paper we present a distributed RDF query architecture to address these problems, which is built on Chord and Sesame. In our architecture, we construct semantic topology with distributed interest-index to route query and take advantage of semantic similarity between query and peer to select peers which are capable of answering the given query. Experimental results have showed that our approach has a much better performance while achieving a good recall rate.

1 Introduction

The next generation Web, Semantic Web, has recently been drawn considerable attention from both academy and industry. Through the structural and formal description of resource, semantic web makes the exchanging, analyzing and reasoning among different computer systems possible. By far, now that the W3C has put forward a series of semantic web specifications such as RDF(S)[1], OWL[2] and RDF storage and query systems also provide the fundamental platform, many kinds of semantic web applications have emerged.

At present most of RDF storage and query systems like Jena[3], Sesame[4] adopts centralized architecture to store information so as to build a local knowledge base, in which a great many of disadvantages exist. To begin with, if you want to query the remote knowledge source, you could not use unless you copy it from the remote to the local and merge it with local knowledge source. Furthermore, when you want to con-

¹ This work is supported by National Natural Science Foundation of China under Grant No.90412008, the National Grand Fundamental Research 973 program of China under Grant No.2004CB318204

struct a large application that requires the processing of megabytes of data, the query's response time will be unbearable. Finally, the scalability, efficiency and reasoning capability of the system would be great problems. Therefore, we need a distributed RDF storage and query architecture to support the semantic web application which will include a large dataset and can provide efficient query and high scalability.

P2P systems, like Chord[5], have become an important distributed architecture, which have many good characteristics, such as no centralized server, scalability, robustness against failure of any single component and so on. It would be a good idea to construct a distributed RDF query architecture that based on P2P architecture. By this way, we could not only solve the above problems but also make use of flexible and extendable resource-description method of semantic web to enable P2P system to support more complex metadata and query.

In this paper, we illustrate a distributed RDF query architecture based on P2P network and show how to take advantage of peer interest for routing query. In section 2 we will describe the design of distributed RDF query architecture based on Chord; section 3 will introduce how to compare semantic similarity between query and peer and how to route RDF queries by the way of interest-index; section 4 will give some experiment results; section 5 will discuss some questions related to our work, then concluding the paper in section 6.

2 Peer-to-peer based distributed RDF query Architecture

Before discussing the distributed RDF query architecture, we will recognize two basic components of the architecture: Sesame and Chord. Sesame is a RDF storage and query architecture, which supports RDF(S) inference and customized OWL inference. And Chord is a classic peer-to-peer system, which is a logical identifier ring space. Each node (peer) gets a random identifier when it joins Chord, and then all the nodes separate the identifier ring into chords. Given a key k , chord uses a consistent hash function (SHA-1) to get h_k , which is its position in identifier ring. Then the key k is assigned to the node whose identifier is equal to or follows (the identifier of) h_k , denoted by *successor*(h_k) in Chord.

In our architecture, we take the latest Sesame 1.2 as the knowledge storage architecture and apply the query language SeRQL for formulating semantic queries. As for

Chord, we don't take its protocol to distribute knowledge but to build a distributed interest index.

The paper describes the architecture in detail as bellow. Figure 1 shows the high-level design of the distributed RDF query architecture on a single peer. We will now briefly describe the individual components.

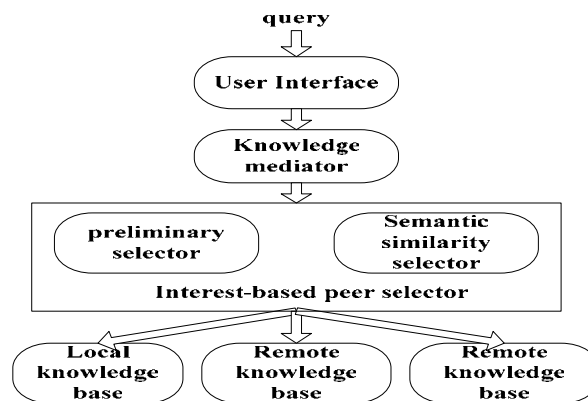


Figure 1 high-level design of the distributed RDF query architecture on a single peer

Every peer just has the same query structure as figure 1 shows .They could manage their own knowledge base formed by the database mode of Sesame and adopts SeRQL as semantic query language to query local and remote knowledge base uniformly.

Next, User interface component, on the one hand it receives query request from users and distributes to the lower level knowledge mediator, on the other hand it also take the responsibility of the customized display of query result.

By means of masking different knowledge bases on the lower level, knowledge mediator could offer a uniform knowledge view to users. And its function also covers analyzing the query to extract characteristics and distributing query request to the remote knowledge base after the mediator has received the query request delivered from user interface. In addition, knowledge mediator should take further processing on the results returned from different knowledge bases, for instance, to take semantic duplication detection, etc.

The interest-based peer selector consists of two parts: preliminary selector and semantic similarity selector. Preliminary selector uses distributed interest-index to obtain a peer set, and next semantic similarity selector will evaluate semantic similarity between query and peer from the preliminary peer set in order to get a final peer set.

In one word, this component is in charge of selecting peers, which are capable of answering the given query.

3 Interest-based distributed RDF query routing

In this section, we will consider a certain domain to construct the above architecture. Usually in the everyday work, researchers always look up and read a large number of documents, for instance, computer scholar often use the query system like Citeseer[6] and also collect some interested papers by themselves. So we could build a P2P based knowledge-exchanging platform in computer science domain to make researchers exchange information more freely and conveniently.

For constructing such a platform, we should build the semantic descriptions of various resources in this system at first. As everyone knows, papers are the most important resource in this application. Then with OWL we describe paper resources as ontology. Here the example bellow shows a fragment of the ontology,

```
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:allValuesFrom rdf:resource="#Publication"/>
    <owl:onProperty>
      <owl:ObjectProperty rdf:ID="refTo"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
```

Besides the paper resource, we also need ACM topic ontology to classify these papers. The ACM topic ontology is hierarchy classification architecture in computer science domain, which consists of 1287 topics and adopts the “SubTopic” property to represent the relationship between sub-topic and parent- topic.

After defining the semantic description of resources in the system, the knowledge base on each peer could be constructed according to the above ontology. In the following parts, we will describe how to deal with the distributed query under the situation above. Section 3.1 will show how to describe the peer interest and the query characteristics and how to compare semantic similarity between them. Section 3.2

will describe how to use Chord protocol to build distributed interest-index to shape semantic topology. Section 3.3 will introduce how to route query with the support of distributed interest-index and describe interest-based peer selection policy in detail.

3.1 Semantic descriptions of peer interest and query

In order to support the query on different knowledge bases, query should be sent to many peers. If just broadcasting, there will be many problems. An obvious solution is to send query to those peers which probably can answer a given query.

Hence, we need to compare similarity between peer and query. First, we define the method of describing peer interest and query characteristic. Here we denote peer interest as P and query characteristic as Q and ACM topic set as T . Due to the topics of papers reflecting interest on which peer focuses, we can use paper topics produced by ACM ontology to describe peer interest. Here, we count the topics and the corresponding number in the knowledge base to describe it. Supposing a peer contains these topics $t1, t2...tm$, and the corresponding number are $n1, n2...nm$, then P is $\{(t1, n1), (t2, n2), \dots, (tm, nm)\}$. Similar to describing peer interest, by analyzing the query, we could obtain the topic set to define query characteristic Q . Take the following query as the example:

```
select distinct id, title, abstract, keywords  
from {id} cnds:hasTitle {title};  
cnds:hasAbstract {abstract};  
cnds:hasKeywords {keywords};  
cnds:isAbout {<acm: ACMTopic/Computer_Systems_Organization>};  
using namespace  
cnds = <"http://ais.grids.cn/ontology/20041104/bibonto#">  
acm = <"http://ais.grids.cn/ontology/20041104/classification#">
```

So this query characteristic Q is {"ACMTopic/Computer_Systems_Organization"}. Sometimes, we cannot extract any topic from the query, and then we define Q as the topics of the peer that sends the query.

Next, we define semantic similarity between query characteristic and peer interest, the higher the similarity value is, the more similar they are, and more possibly the

peer answers the query. Now many semantic similarity measuring methods have been developed. In our case, the similarity degree of two topics depends on the semantic distance between them, that is, the position they stay in ACM classification system, the nearer the distance is, the more similar they are. For example, topic ACM-Topic/Software/Software_Engineering is closer to the topic ACMTopic/Software/Software_Engineering/Management than the topic ACMTopic/Software/Software_Engineering/Metrics/Complexity_Measures.

[7] have raised some methods to compare the semantic similarity degree of the individual words in a hierarchical structure., which are also applicable for comparing the similarity degree of ACM classifications. Here is the similarity function between topic s_1 and s_2 :

$$Sim(s_1, s_2) = e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \quad \alpha=0.2, \beta=0.6$$

l is the shortest path length between s_1 and s_2 , h is the depth of the subsumer which is derived by counting the levels from the subsumer to the top of the lexical hierarchy.

Now we define the similarity function between query Q and peer P_i .

$$Sim(Q, P_j) = \sum_{i=1}^n (Sim(s_i, P_j) \times n_j^{j_0} / \sum_{k=1}^m n_k^{j_0})$$

$$Sim(s_i, P_j) = Sim(s_i, s_j^{j_0}) \quad (Sim(s_i, s_j^{j_0}) = \max_{l \in J_j} \{ Sim(s_i, s_l^l) \})$$

$$Q = \{s_1, s_2, \dots, s_n\}, P = \{P_1, P_2, \dots, P_m\}, P_i = \{ \langle s_i^j, n_i^j \rangle \mid j \in J_i \}$$

Here s_i^j represents a topic appearing in P_i , and its number is n_i^j . J_i is the topic set in P_i .

3.2 Distributed interest-index

Every peer has its own interest and different capacity of answering different queries, therefore peer interest is an important factor in the process of conducting query.

However, peer interests disperse on different peers respectively, so for taking advantage of peer interest we take Chord protocol to build distributed interest-index

Every peer should register their own interest into the system. For each topic in peer interest and all its parent topics, there is a peer for maintaining a corresponding interest table. Supposing a certain topic t , the corresponding node id $n_i=successor(t)$ and its interest table holds the interests and the IDs of the peers which cover this topic. Figure 2 is an example of registering interest.

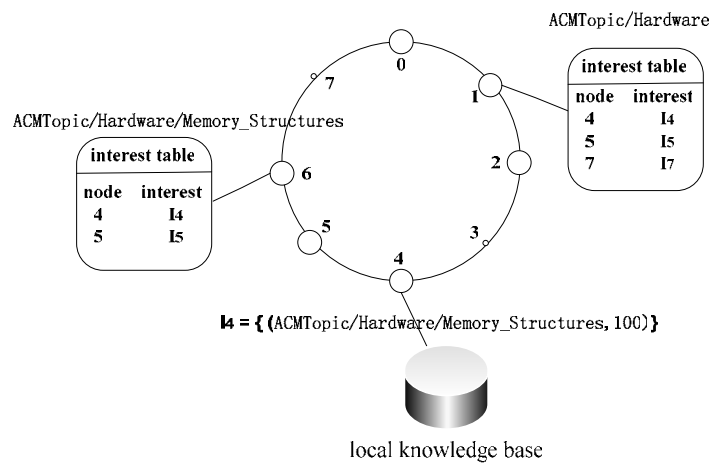


Figure 2 An example of registering interest

As showed in Figure 2, the interest of node 4 $I_4=\{(ACMTopic/Hardware/Memory_Structures,100)\}$, so it should register its own interest on $successor(“ACMTopic/Hardware/Memory_Structures”)$ and $successor(“ACMTopic/Hardware”)$, which is node 6 and node 1 respectively.

When a new node joins the system, it should distribute its own interests following the way mentioned above, so the interest tables on all peers will shape a semantic topology. What should be pointed out is that we assume peer interest changes slowly or even constantly, so it is in the real life because science workers always pay close attention to some certain areas for a long time. While local knowledge changes, it only needs to refresh interest table periodically rather than immediately.

The distributed interest-index shapes the semantic topology which is independent from the underlying network and offer foundation for routing RDF query.

3.3 Routing RDF query with distributed interest-index

Knowledge mediator provides a unified knowledge view to make the distribution of knowledge transparent for users, and analyzes query to extract query characteristic Q where we can get the corresponding peer to each topic by the $successor(t)$. From these peers, preliminary selector could get back its needed interest tables.

As indicated in figure 2, if peer 2 send a query and its characteristic is {"ACM-Topic/Hardware/Memory_Structures"}, then the corresponding peer id is 6 where we could obtain the interest table containing peer 4 and peer 5. When the corresponding interest table of a topic is empty, it has to get back the corresponding interest table of its parent-topic. After preliminary set is selected, it will be cached on this peer, so when there is another similar query, it could use the cached peer set directly. As peer interest changes slowly, the cached peer set would have high hit rate.

Next, the semantic similarity selector will calculate semantic similarity value between query characteristic and each peer interest in preliminary set, the query ranks higher with the value increasing. So we could select the top n peers or those ones whose similarity value is higher than a certain threshold value. After that, the query will be sent to these peers at the same time.

4 Experiment result

In the distributed RDF query system, the factors bellow should be taken into account. 1) Number of peers and resources, representing the size of P2P system and reflecting the scalability of the system. 2) Resource distribution, rarely completely random but may have certain properties. 3) Network topology, what we applied is built on Chord. 4) Peer selection policy, it may be simply broadcasting without any selection, or interest-based peer selection we described before.

We build the simulation system to evaluate our architecture. Our dataset comes from Citeseer which consists of metadata for 574084 articles in the computer science domain and is classified according to ACM classification ontology. These articles are distributed to different peers and we assume that the number of topics on one peer follows standard normal distribution, and the number of copies of an article follows Zipf distribution.

Now we give the experiment results. First we compare the query time in centralized architecture and distributed architecture. As Figure 3 shows, distributed architecture reduces query time obviously.

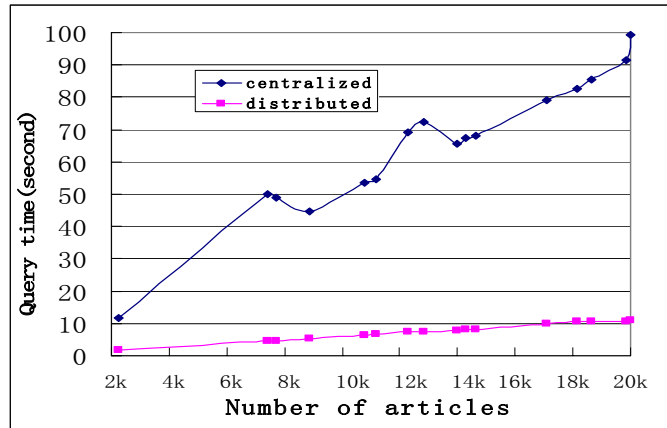


Figure 3

Next, we compare the recall of query with three peer selection policies which are random selection, exact match and interest-based similarity selection respectively. In Figure 4a, we just pick out 4 peers to answer the query with each method, as it illustrates, with the total number of peers increasing, the recall goes down, and interest-based similarity selection always has much better recall than the others. Figure 4b shows that with a fixed total number of peers (400 peers), interest-based similarity selection requires fewer peers to get a good recall.

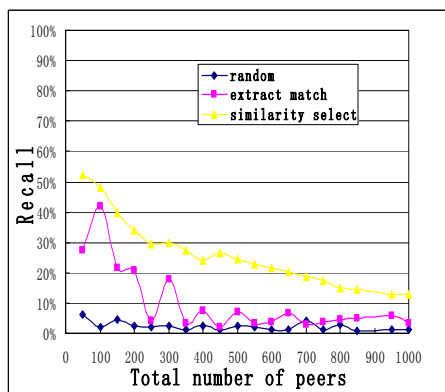


Figure 4a

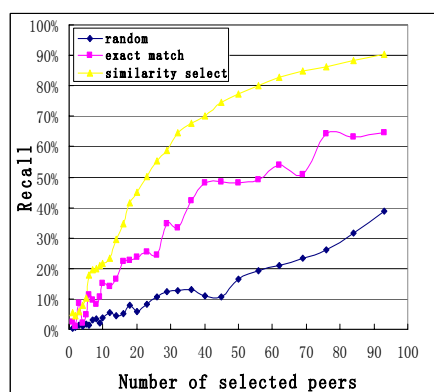


Figure 4b

5 Discussions

There are some questions related to our work. First, we just use paper classification to describe peer interest, but [8] have pointed out that the history queries are also an important factor to peer interest, so we should take further study on how to describe peer interest more exactly. Secondly, for building distributed interest-index, we register all parent topics of every topic, which produce too many messages. That is also a problem we have to solve in the future. In addition, [9] proposes that sometimes we have to decompose a query into sub-queries which will be dispersed to different peers and then join the results to get the final answer. Hence, we should study how to decompose the query, how to distribute sub-query and how to join temporary results.

6 Conclusions

At present, the RDF storage and query architecture always adopts centralized architecture. It will cause many problems such as the performance and the scalability. In this paper, we put forward a method of constructing distributed RDF query architecture built on P2P, in which we route query with the semantic topology formed by distributed interest-index and exploit interest-based peer selection policy to reduce peer set that query is sent to. The experiment results indicate that our architecture could resolve the problem of centralized RDF query architecture and improve the performance and scalability of the system.

References

1. O. Lassila and R. R. Swick: W3C Resource Description framework (RDF) Model and Syntax Specification. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>
2. Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>
3. Jena – A Semantic Web Framework for Java <http://jena.sourceforge.net/>
4. Broekstra, J. and Kampman: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In Proceedings of ISWC, 2002.
5. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In Proceedings of the ACM SIGCOMM, 2001.

6. Steve Lawrence, C. Lee Giles, Kurt Bollacker: Digital Libraries and Autonomous Citation Indexing, IEEE Computer, 1999
7. Yuhua Li, Zuhair A. Bandar, and David McLean: An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. IEEE Transactions on knowledge and data engineering, 2003.
8. Peter Haase, Marc Ehrig, Andreas Hotho, Björn Schnizler: Personalized Information Access in a Bibliographic Peer-to-Peer System, In Proceedings of the AAAI Workshop on Semantic Web Personalization, 2004
9. Stuckenschmidt, Heiner and Vdovjak, Richard and Broekstra, Jeen: Index Structures and Algorithms for Querying Distributed RDF Repositories. In Proceedings International WWW Conference, 2004