Albert-Ludwigs-Universität Freiburg SS 2009
Institut für Informatik
Lehrstuhl für Rechnernetze und Telematik

Seminararbeit

Semantic P2P Database, RDF

Patrick Tchakoute

27. Juli 2009

Betreut durch Liaquat Ali

Inhaltsverzeichnis

1 Motivation

In Peer-to-Peer network, there is a large traffic of data. If we consider a system where many machines try to get some data from a server at the same time, this system could become overloaded. Thus, it will not be possible to guarantee to find the data, even if this have been stored on the server. In fact, in an environment which is highly dynamic[1], the server can become a registration bottleneck [2].

We want to have no failure by a growing amount of traffic. Min Cai an Martin Frank have presented a scalable distributed RDF repository called '*RDFPeers*' [1]. This will permit to find query results if the exist.

2 Introduction

On the Web, resources are represent by RDF data. We will define RDF on section 3. As explain above, it is not easy to find them just because they have been put on the Web. As example, Google takes many days to show a Web page that have newly created.

With RDFPeers, we will see that it is efficient to find data which have been stored before. In this repository, data are stored three time and queries results will be find by routing to the nodes in the network where the RDF data are stored if it exist.

We will first of all define a RDF data and have a look on the architecture of the RDFPeers. Then to understand how this data are stored and searched, we will see how Distributed Hash Tables(DHT) work.

3 RDF

3.1 Definition

RDF(Resource Description Framework) represents resources on the web. It is a method for modeling information that is implemented in web resources. This resources are in a special form so that machines can understand their meaning.

RDF documents are composed of a set of RDF triples $< subject, predicate, object >$ The Subject is the resource which is described. The predicate is the relationship between the subject and the object. It is the property that is apply to the subject. The object is

[1]Data are joining and leaving at any time

the property value of the predicate.

> *Example:*
>
> The information '*rdfpeers creator is mincai*' is represent by the triple:
> $< info : rdfpeers >< dc : creator >< info : mincai >$

We can define some properties to the component of a RDF triple as follow:

- Subject is either an URI[2](Uniform Resource Identifier) or a blank node
- Object is either a resource or a literal
- Resource is identified by URI
- Literals are either plain[3] or typed[4]
- Literals have the lexical form of a Unicode string

3.2 Architecture of RDFPeers

In the distributed triple repository, the are many nodes called RDFPeers. The RDFPeers use a multi-attribute addressable network (MAAN) [2] as the underlying network layer. Each node has five component [1]:

1. The MAAN network layer
2. The RDF triple loader
3. The local RDF triple storage
4. The native query resolver
5. The RDQL-to-native-query translator

The MAAN has three classes of messages for maintaining the topology of the network(JOIN/LEAVE, KEEPALIVE), storing(STORE) and searching(SEARCH) triples. The RDF triple loader transform data into RDF triples and store them in the RDFPeers network using the store message of MAAN. Each RDFPeer store triples in its local RDF triple storage when it receives a STORE message. The native query resolver analyses queries and find their results by using the SEARCH message of MAAN. In case of a high-level(complex) query like RDQL, it will be simplify into a native query. This is done by the RDQL-to-native-query translator.

[2]String of characters to identify or name a resource on the Internet
[3]example: "$MinCai$"
[4]"28"^^$< xmls : integer >$

4 Distributed Hash Tables

We will take here the example of the Chord DHT system proposed by Ion Stoica al el [3]. Chord supports scalable $< key, object >$ pairs registration and lookup operations. It uses a one-dimensional circular identifier space with modulo 2^m [2]. m is the number of bits in node identifiers and object keys(Data). Each node has an ID(m-bit identifier). Hash functions are used to assign ID to nodes.

$h : P \longrightarrow Z_{2^m}$ where P is the set of nodes

m should be high enough to avoid collusion.(for example $m = 128$)

4.1 Routing

In Chord, each node has a *successor list* and a *finger table.*The nodes in the successor list follow the node in the identifier space immediately. In the finger table, there are at most m entires. The i-th entry in the table for the node with ID p contains the identity of the first node s, that succeeds p by at least $2^i - 1$ on the identifier circle.(all arithmetics are modulo 2^m). That mean that the first finger is the immediate successor of p.

When node p look for the object with key k, it will route a search request to the successor node x of key k, x = *successor(k).* If node x is far away from p,p can forward the request to a far node in its finger table, which is much closer to x than p. The Process is repeated until the request arrive to the node p' which ID is immediately before k in the circle. The successor of p' is responsible for the key k.

5 Storing RDF triples

Each triple is stored three times. In fact the hash function is applied to the subject, predicate, and object to find the key that will be use for the storage. Each triple will be stored at the successor node of the hash key of the value of the routing key attribute-value pair. To store for example the triple on the example in the subsection 3.1 by its predicate, the hash function will be apply to the predicate ($key = Hash('< dc : creator >')$).
RDFPeers would send the following message:
STORE $\{key, \{('subject', < info : rdfpeers >), ('predicate', < dc : creator >), ('object', < info : mincai >)\}\}$
where $key = Hash('< dc : creator >')$

In this example, the triple will be stored at the successor node of key. We do this for the three component of the triple. And the end we have three node which is responsible for that triple.

6 Searching RDF triples

Min Cai and Martin Frank define different sets of queries. These are all resilved with MAAN´s multi-attribute range queries.

6.1 Atomic queries Patterns

In this query, the components of the triple $< subject, predicate, object >$ can each either be a variable or an exact value. Because we have three component, there are $2^3 = 8$ different triples.
For example $(?s, pi, ?o)$, $(?s, pi, oi)$: in the first query, the predicate is given(exact value).We have to find the subjects and objects of the triples having this predicate.
The most general and most expensive[5] query is the query where all the components are variable $(?s, ?p, ?o)$. MAAN´s routing algorithm (based on DHT as seen in subsection 4.1) is used to resolve these queries. If we don´t consider the most general query, there is always at least one component which is an exact value. That mean, the query can be routed to the node responsible for storing the exact value. that responsible node matches all the triples with the value and send them to the requesting node.

6.2 Disjuctive and Range Queries

In this type of query, the domain of the variables is limited. These query are atomic query with constraint. This constraint can be $OrExpression('||')$ or an interval for a numeric value. There also constraint for stirng value. For example a set of string value can be define. Examples of Disjuctive and Range Queries are the following:
$(?s, dc : creator, ?c)$ AND $?c =' Min'||?c =' Jinbo'$
$(?s, foaf : age, ?age)$ AND $?age > 35$ && $?age < 55$
In the first query, a node search triples with creator´s name Min or Jinbo. In the second query, there is a restriction on the age of the subject.

To resolve the query, the underlying network layer of RDFPeers(MAAN) uses locality preserving hashing. Hash functions are used to assign m-bits identifier to each node. A

[5]query matches all triples

locality preserving hashing function H is defined as follow:
$H(v_i) < H(v_j)$ iff $v_i < v_j$, and if an interval $[v_i, v_j]$ is split into $[v_i, v_k]$ and $[v_k, v_j]$, the corresponding interval $[H(v_i), H(v_j)]$ must be split into $[H(v_i), H(v_k)]$ and $[H(v_k), H(v_j)]$.

6.3 Conjuctive Multi-Predicate Queries

Conjuctive Multi-Predicate Queries can be resumed as a conjunction of atomic queries patterns or disjunctive range queries for the same subject variable.
Example:

(?x, <rdf:type>, <foaf:Person>)
(?x, <foaf:name>, 'John')
(?x, <foaf:age>, ?age) AND ?age > 35

To resolve these conjunctive multi-predicate queries, the authors[6] use a recursive query resolution algorithm which searches candidate subjects on each predicate recursively and intersects the candidate subjects inside the network, before returning the search results to the query originator [1]

7 Conclusion

In this paper,we have seen the RDF repository proposed by Min Cai and Frank Martin(RDFPeers). As many other RDF repository, RDFPeers makes possible to store, index and search RDF data. It guarantee that RDF data will be found if they exist. The particularity in this repository is first of all the fact that RDF data are stored three time. Then, the system support distributed hash table. That makes the resolution of the queries(*searching for a RDF data*) to become efficient. The authors also have a look of the perfomance of the implementation of the system(MAAN). The results are consistent with their theory [2].

[6]M. Cai, M. Frank

Literatur

[1] Min Cai and Martin Frank. Rdfpeers: A scalable distributed rdf repository based on a structured peer-to-peer network. pages 650–657. ACM, May 2004.

[2] Min Cai; Martin Frank; Jinbo Chen and Pedro Szekely. Maan: A multi-attribute addressable network for grid information services. IEEE Computer Society, 2003.

[3] Peter Mahlmann and Christian Schindelhauer. *Peer-to-Peer Netzwerke*, pages 81–88. ISBN: 978-3-540-33991-5. Springer, 2007.