Seminar: P2P Netzwerke

Topic: Semantic P2P Database, RDF

Patrick Tchakoute

29.07.2009
Overview

- Motivation
- RDF (Definition and example)
- RDFPEERS Architecture
- DHT (Chord)
- How to store?
- Query
- Evaluation
- References
Motivation

- Can we guarantee to find data ("query results")?
  - NO

- What happens in a highly dynamic environment where many data join and leave at any time?
  - Registration bottleneck

30.07.2009 P. Tchakoute: Semantic P2P Database, RDF
Motivation

- Overloaded system
Motivation

- No system failure by growing amounts of traffic

- Scalable and distributed RDF repository presented by M. Cai and M. Frank ("RDFPeers")
RDF Definition

- RDF = Resource Description Framework
- RDF represents resources on the web
- Special form so that computer can understand
RDF
Definition(2)

- RDF documents are composed of a set of RDF triples

  <subject, predicate, object>
RDF Definition (3)

- Subject is either an URI (Uniform Resource Identifier) or a blank node

- Object is either a resource or a literal

- Resource is identified by URI

- Literals are either plain or typed

- Literals have the lexical form of a unicode string
RDF Example

Information: „rdfpeers creator is mincai“

<info:rdfpeers> <dc:creator> <info:mincai>
RDF Example(2)

- `<info:mincai> <foaf:name> "Min Cai"`
RDF
Example (3)

Information: mincai is 28 years old

- `<info:mincai> <foaf:age> "28"^^<xmls:integer>`

  typed literal object
RDFPEERS Architecture

- RDFPeers is organized into a multi-
  attribute addressable network (MAAN)

- Each node consists of five components:
  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
RDFPEERS Architecture (2)

• **MAAN protocol → 3 classes of messages**
  - Topology maintenance
  - Storage
  - Search

• **RDF triple loader:**
  - read RDF document
  - parses it into RDF triples
  - Store triples into RDFPeers („STORE“ message)

• **Local RDF triple storage:**
  - Local storage of a RDPeer
RDFPEERS Architecture(2)

• Native query resolver:
  • Parses RDFPeers queries
  • Resolve queries ("SEARCH" message)

• RDQL-to-native-query translator:
  • Map high-level queries into native queries
RDFPeers Architecture(3)

Source: RDFPeers: A scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network; Min Cai and Martin Frank
DHT (Chord)

- System proposed by Ion Stoica (2001)
- Supports scalable <key, object> pairs registration
- Uses a one-dimensional circular identifier space (ring)
- Each node gets a unique $m$-bit identifier (ID)
- Each node has a successor list and a finger table
How to store?

- One of the three attribute values is designed as the destination of the routing
- Each triple is stored three times (once based on its subject, predicate, and object)
- Each triple will be stored at the successor node of the hash key of the value of the routing key attribute-value pair

<info: mincai> <foaf: name> "Min Cai" .
<info: mincai> <foaf: age> "28"^^<xmls: integer>
How to store?

Example

• Store the first triple above by subject

STORE {key, {("subject", <info:rdfpeers>),
  ("predicate", <dc:creator>),
  ("object", <info:mincai>))}

where key=SHA1Hash("<info:rdfpeers>")

• the first attribute-value pair ("subject",
  info:rdfpeers) is the routing key pair

• key is the SHA1 hash value of the subject value.

• This triple will be stored at the node which is the successor node of key
How to store?

Example

<table>
<thead>
<tr>
<th>URI / Literal</th>
<th>Hash Value in [0, 15]</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="">info:rdfpeers</a></td>
<td>13</td>
</tr>
<tr>
<td><a href="">info:mincai</a></td>
<td>1</td>
</tr>
<tr>
<td><a href="">dc:creator</a></td>
<td>5</td>
</tr>
<tr>
<td><a href="">foaf:name</a></td>
<td>4</td>
</tr>
<tr>
<td><a href="">foaf:age</a></td>
<td>10</td>
</tr>
<tr>
<td>“Min Cai”</td>
<td>7</td>
</tr>
<tr>
<td>“28”</td>
<td>2</td>
</tr>
</tbody>
</table>

Source: RDFPeers: A scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network; Min Cai and Martin Frank
Queries

- Atomic Triple Patterns
- Disjunctive and Range Queries
- Conjunctive Multi-Predicate Queries
- RDQL Queries (Jena Java RDF toolkit)
  - Inefficient for join queries
Queries

- **Atomic Triple Patterns**
  - triple pattern in which the subject, predicate, or object can each either be a variable or an exact value
  - eight resulting possible queries
<table>
<thead>
<tr>
<th>No.</th>
<th>Query Pattern</th>
<th>Cost</th>
<th>Query Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>(?s, ?p, ?o)</td>
<td>O(N)</td>
<td>find all possible triples</td>
</tr>
<tr>
<td>Q2</td>
<td>(?s, ?p, o_i)</td>
<td>logN</td>
<td>given object o_i of any predicate, find the subjects and predicates of matched triples</td>
</tr>
<tr>
<td>Q3</td>
<td>(?s, p_i, ?o)</td>
<td>logN</td>
<td>given predicate p_i, find the subjects and objects of the triples having this predicate</td>
</tr>
<tr>
<td>Q4</td>
<td>(?s, p_i, o_i)</td>
<td>logN</td>
<td>...</td>
</tr>
<tr>
<td>Q5</td>
<td>(s_i, ?p, ?o)</td>
<td>logN</td>
<td>...</td>
</tr>
<tr>
<td>Q6</td>
<td>(s_i, ?p, o_i)</td>
<td>logN</td>
<td>...</td>
</tr>
<tr>
<td>Q7</td>
<td>(s_i, p_i, ?o)</td>
<td>logN</td>
<td>...</td>
</tr>
<tr>
<td>Q8</td>
<td>(s_i, p_i, o_i)</td>
<td>logN</td>
<td>...</td>
</tr>
</tbody>
</table>

Source: RDFPeers: A scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network; Min Cai and Martin Frank
Queries

- **Disjunctive and Range Queries**
  - Domain of the variables is limited
    - Atomic queries with constraints
  - Constraint
    - OrExpression
    - an interval for a numeric value
Queries

- Disjunctive and Range Queries (2)

No: Q9

- (a) (?s, dc:creator, ?c) AND (?c="Tom" || ?c="John")

- (b) (?s, foaf:age, ?age) AND (?age > 10 && ?age < 20)
Queries

- Conjunctive Multi-Predicate Queries

- Conjunction of atomic queries patterns or disjunctive range queries for the same subject variable.
Queries

• Conjunctive Multi-Predicate Queries

Example of a subject variable with a list of restricting predicate, object or predicate, object-range pairs.

No: Q10

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

- RDQL Queries
  - Query language for RDF
  - (1) SELECT ?x
        WHERE (?x, <vcard:FN>, "John Smith")
        \rightarrow Q4
  - (2) SELECT ?x, ?fname
        WHERE (?x, <vcard:FN>, ?fname)
        \rightarrow Q3
Queries

• RDQL Queries

(3) SELECT ?givenName
    WHERE (?y, <vcard:Family>, "Smith"),
    (?y, <vcard:Given>, ?givenName)

→ Q9 with Q3
How to store?

Example

By subject:
<info:rdfpeers> <dc:creator> <info:mincai>

By object:
<info:rdfpeers> <dc:creator> <info:mincai>

By subject:
<info:rdfpeers> <foaf:name> “Min Cai”
<info:mincai> <foaf:age> “28”

By object:
<info:rdfpeers> <dc:creator> <info:mincai>

By predicate:
<info:mincai> <foaf:age> “28”

By object:
<info:mincai> <foaf:name> “Min Cai”

Source: RDFPeers: A scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network; Min Cai and Martin Frank
Evaluation

- Implementation in Java
- Performance measurement on a real-world network (128 nodes)
- Number of neighbors at each node increases logarithmically with the network size
  - node state in MAAN scales well to a large number of nodes
- For exact-match queries, the number of routing hops in the worst case is $O(\log N)$ and the average routing hops is $(\log N)/2$
References

- M. Cai, M. Frank. RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network


Thank you for listening.

Questions?