Evaluating Conjunctive Triple Pattern Queries over Large Structured Overlay Networks

Erietta Liarou, Stratos Idreos, and Manolis Koubarakis 2006.

Waled Almukawi.
Outline

- Motivation
- Terminologies
  - Conjunctive Queries
  - RDF
  - DHT
- System Model & Data Model
- Query Chain Algorithm
- Spread By Value Algorithm
- Experiments
- Future Works
- References
Motivation

- One of the most challenging and interesting open problems in the frontiers of P2P and Semantic Web is how to evaluate queries expressed in RDF query languages on top of P2P networks.

- Previous RDFpeers used only atomic formula or disjoint RDF triple pattern.

- The conjunctive queries are interesting, since they provide a flexible method to deal effectively with a full functionality of existing RDF query.
Conjunctive Queries

- The conjunctive queries are simply the fragment of first order logic given by the set of atomic formula using conjunction $\land$.

- A conjunctive query $q$ formula:
  
  $\forall x_1, \ldots, x_n : (s_1, p_1, o_1) \land (s_2, p_2, o_2) \land \cdots \land (s_n, p_n, o_n)$

  where $x_1, \ldots, x_n$ are answer variables, each $(s_i, p_i, o_i)$ is a triple pattern, and each variable $x_i$ appears in at least one triple pattern $(s_i, p_i, o_i)$. 
Conjunctive queries (2)

Example:

Q: \text{ans}(x,y) :- \text{country}(\text{Id},x), \text{capital}(y,x), \text{continent}(x, \text{"Europa"})

Q': \text{ans}(x,y) :- (?x, "type", "country"), (?x, "has-capital", ?y), (?x, \text{continent}, \text{"Europa"}).
Resource Description Framework (RDF)

- RDF is a framework for describing Web resources, such as the title, author, content ... e.t.c.

- RDF Statements is The combination of a Resource, a Property, and a Property value forms a Statement (known as the subject, predicate and object of a Statement).
Distriuted Hash Tables (DHT)

- DHTs are an important class of P2P networks that offer distributed hash table functionality, and allow one to develop scalable, robust and fault-tolerant distributed applications.
- Distribute data over a large P2P network
- Can Quickly find any given item in Log N nodes.

Look up for an Item in the DHT is performed in O(Log N) nodes.
System Model (ref:[4])

Finger table:

<table>
<thead>
<tr>
<th>distan</th>
<th>succ.</th>
<th>keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>N11+0</td>
<td>N11</td>
<td>11</td>
</tr>
<tr>
<td>N11+1</td>
<td>N17</td>
<td>12</td>
</tr>
<tr>
<td>N11+2</td>
<td>N17</td>
<td>13</td>
</tr>
<tr>
<td>N11+4</td>
<td>N17</td>
<td>15</td>
</tr>
<tr>
<td>N11+8</td>
<td>N27</td>
<td>19</td>
</tr>
<tr>
<td>N11+16</td>
<td>N27</td>
<td>27</td>
</tr>
<tr>
<td>N11+32</td>
<td>N48</td>
<td>43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>distan</th>
<th>succ.</th>
<th>keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>N27+0</td>
<td>N27</td>
<td>27</td>
</tr>
<tr>
<td>N27+1</td>
<td>N33</td>
<td>28</td>
</tr>
<tr>
<td>N27+2</td>
<td>N33</td>
<td>29</td>
</tr>
<tr>
<td>N27+4</td>
<td>N33</td>
<td>31</td>
</tr>
<tr>
<td>N27+8</td>
<td>N42</td>
<td>35</td>
</tr>
<tr>
<td>N27+16</td>
<td>N48</td>
<td>43</td>
</tr>
<tr>
<td>N27+32</td>
<td>N63</td>
<td>59</td>
</tr>
</tbody>
</table>
Data Model

- Each network node is able to describe in RDF the resources that it wants to make available to the rest of the network.
- It creates and inserts the RDF triples \((\text{Subject, Predicate, Object})\).

  - The subject of a triple identifies the resource that the statement is about.
  - The predicate identifies a property or a characteristic of the subject.
  - The object identifies the value of the property.
Query Chain Algorithm

- The main idea of QC is that the query is evaluated by a chain of nodes.
- Intermediate results flow through the nodes of this chain.
- Finally the last node in the chain delivers the result back to the node that submitted the query.
- Distribute the Load Balance by
  - Split the query into the triple patterns that it consists of.
  - Evaluate each triple pattern separately to a different node.
  - Intermediate result flow through these nodes as triple and partially satisfy the query.
Query Chain Algorithm

- Indexing a new triple

```
Put (S, P, O)
```

```
T1=Hash(S)  
T2=Hash(P)  
T3=Hash(O)  
```
Query Chain Algorithm

- Indexing a new triple

Put (s1, P1, s2), (s2,p2,s3), (s3,p3,o3).

- T1=Hash(p1)
- T2=Hash(p2)
- T3=Hash(o3)
Query Chain Algorithm

- Evaluating a query

Query:

\[ q \text{ is } ?x:(?x,p_1,?y), (\text {?y},p_2,?z), (\text {?z},p_3,o_3). \]

R1 searches in its TT and evaluates t1

\[ L = \{ s_1, s_2 \} \]

\[ R' = \{ s_1, s_2 \} \]

QEval(q,2,R',ip(x))
Query Chain Algorithm

- Evaluating a query

\[ S_2,p_2,s_3 \]

\[ r_2 \]

R2 searches in its TT and evaluates \( t_2 = (\text{?y},p_2,\text{?z}) \)

\[ S_3,p_3,s_3 \]

\[ r_3 \]

R3 searches in its TT and evaluates \( t_3 = (\text{?z},p_3,o_3) \)

\[ X \]

\[ \text{IP}(x) \]

L = \{s_2,s_3\}, R' = \{s_1,s_3\}

QEval (q,3,R',ip(x))

L = \{s_3\}, R' = \{s_1\}

Delivers answer answer (q,R')
Query chain algorithms

- Local processing at each chain node

Query evaluation at node $n$:

$$QEval(q, i, R, ip(x))$$

$$L = \pi X (\sigma F (TT))$$

Example: $q_1 = (?s_1,p_1,o_1)$

$$L = \pi 1(\sigma 2=p_1 \land 3=o_1(TT))$$

$$R' = \pi Y(R \bowtie L)$$

Query evaluation at node $n'$:

$$QEval(q, i+1, R', ip(x))$$

Query evaluation at node $n_k$:

$$QEval(q, k, R', ip(x))$$

$$R' = \pi Y(R \bowtie \pi X (\sigma F (TT)))$$

Answer to query $q$:

$$Answer(q, R')$$
Spread By Value Algorithm

- In SBV the triple pattern \( t \) will be stored at the successor nodes of the identifiers \( \text{Hash}(s), \text{Hash}(p), \text{Hash}(o), \text{Hash}(s+p), \text{Hash}(s+o), \text{Hash}(p+o) \) and \( \text{Hash}(s+p+o) \).

- SBV exploits the values of matching triples found while processing the query incrementally.
SBV algorithm

- Indexing a new triple

Put (s₁, p₁, s₂), (s₂, p₂, s₃), (s₃, p₃, o₃), (s₁, p₁, o₁).

T₁ = Hash(p₁)
T₂ = Hash(s₂ + p₂)
T₃ = Hash(s₃ + o₃)
T₄ = Hash(p₁)
SBV Algorithm

- Evaluating a query

\[
q \text{ is } \langle ?x : (?x, p1, ?y),
\langle ?y, p2, ?z),
\langle ?z, p3, o3).\rangle
\]

\[
\text{QEval}(q, V, u, IP(x)),
\text{r1 evaluates } \langle ?x, p1, ?y \rangle
\]
SBV Algorithm

Evaluating a query

R1 searches in its TT and finds

v’a={?x=s1,y=s2}, v’b={?x=s1,y=o1}

q’a=(s2p2,?z)(?z,p3,o3)
q’b=(o1p2,?z)(?z,p3,o3)

(?x,p1,?y),(?y,p2,z3)
t1=(s1,p1,s2)
t4=(s1,p1,o1)

QEval(q’a,V,v’a,ip(x))

QEval(q’b,V,v’b,ip(x))

r1

r2

r4

S(H(s2+p2))
S(H(p2×o1))
SBV Algorithm

- Evaluating a query

Each node searches in its TT for matching triples \((?z,p_3,o_3)\)

\[ v''a = \{?x=s_1,?y=s_2,?z=s_3\} \quad q''a = (s_3,p_3,o_3) \]

QEval \((q''a,V,v''a,ip(x))\)

Delivers answer \(\text{Answer} (q,V)\)
Comparing the query chains in QC and SBV

q is \( ?x : (s1,p1,?x),(?x,p2,?y),(?y,p3,o3) \).

\(?x\) has four matching values for \((s1,p1,?x)\).

\(?y\) has three matching values for each value of \(?x\).
Optimizing network traffic

- IP cache (IPC) used by our algorithms to significantly reduce **network traffic**.

- Another effect of IPC, is that we reduce the **routing load** incurred by nodes in the network.
Experiments

- A simulator for chord network with java code was done.

- Our metrics are:
  1) The amount of network traffic that is created.
  2) how well the query processing load and storage load are distribution among the network nodes.
Experiments

- Some Experimental evaluating of the two algorithms, and the result are showed in the following figures

(a) Traffic cost  
(b) IPC effect
Experiments

- Query processing and storage load distribution

![Graphs showing query processing and storage load distribution](image-url)
Future work

- Order of triple chosen to evaluate the query.

- RDFpeers reasoning.
Reference


2) E. Liarou, S. Idreos, and M. Koubarakis. Publish-Subscribe with RDF Data over Large Structured Overlay Networks. In DBISP2P ’05.


Thank you for your attention.