Efficient Query Routing in RDF/S schema based P2P Systems

HDMS 2005

Lefteris Sidirourgos, George Kokkinidis
Institute of Computer Science - FORTH
{lsidir,kokkinid}@ics.forth.gr

Theodore Dalamagas
School of Electr. and Comp. Engineering
National Technical University of Athens
dalag@dblab.ece.ntua.gr

Schema-Based P2P Systems

In schema-based P2P systems, joining peers advertise their actual data using the schema of their bases.

Capturing explicitly the semantics of data bases available in the network using a schema enables us to:
- support expressive queries on (semi-)structured data
- deploy effective methods for locating remote peers that can answer these queries
- build efficient distributed query processing mechanisms

The main challenge in this setting is to build an effective and efficient lookup service for identifying, in a complete decentralized fashion, which peer views can actually contribute to the answer of a specific query.
Our Framework

Chord Ring: DHT Index of view advertisements

- peer1 advertise its view V1
- peerN

Build on top:
- large scale collaborative applications:
  - no centralized warehouse
  - no unlimited data migration
- sophisticated data management services:
  - query processing
  - load balance
  - data replication

RDF/S Query/View Subsumption

Queries are formulated using an RDF/S query language (e.g., RQL)

query pattern:
(x, p3, y)

a pattern specifies the fragment of the RDF/S graph which is involved in the evaluation of the query

query fragment:

Views specify fragments of the RDF/S graph too

view fragment:

Views that are subsumed by the query contribute to the answer:

Query:

How views are placed on the Chord Ring in order to efficiently decide view/query subsumption?
The AdjSub Cube is a succinct representation of RDF/S schema graphs allowing to check whether a graph fragment is subsumed by another.

- Each cell represents a schema triple; each schema triple is assigned a number through the function $\text{pos()}$, according to the position of the corresponding cell in the cube; the encoding of an RDF/S schema fragment of $n$ schema triples $s_{n}, \ldots, s_{1}$ is the unique number $N_{2^{\text{pos}(s_{n})}+\ldots+2^{\text{pos}(s_{1})}}$ or equivalent the set $\{\text{pos}(s_{n}), \ldots, \text{pos}(s_{1})\}$.

- Does not need to be implemented, instead it is used to derive an encoding function of RDF/S schema fragments.

AdjSub Cube relies on an interval-based encoding of subsumption hierarchies.

- Imposes a total ordering of classes and properties on each dimension based on the post-order enumeration.
- Places subsuming views on the Chord Ring sequentially to facilitate the lookup service.
Query Routing

- The sequence in which the subsumed views are looked up is given by the AdjSub Cube
  - this sequence is important since there must be no lookups that address preceding peers in order to avoid costly jumps over the Chord ring

example:
- Query Pattern: 
  \((x, p_3, y)\)
- RDF/S graph fragment:

  ![AdjSub Cube traversal for the RDF/S fragment]

  \(N(C_2, p_3, C_8)\)
  \(N(C_7, p_3, C_8)\)
  \(N(C_7, p_3, C_3)\)

Experimental Evaluation – DHT Index

- views are stored uniformly over the Chord Ring
- as the network grows, views are distributed in more peers thus there is no introduction of hot-spots and the index scales gracefully
Experimental Evaluation - Lookup Service

**Increasing the number of schema triples of a view implies more routing hops, because:**
- there exist additional subsumed views, and
- for large views, subsumed views are distant in the Chord ring

**Less than** $S \times \log(N)$ hops
- ex. for views with 3 schema triples:
  $S = 60$, $N = 20000$, $\log(N) = 14$ : $60 \times 14 = 840$
  but, experimental evaluation shows no more than 350 hops

Conclusions & Future Work

- a DHT-based framework to efficiently route expressive RDF/S queries
- introduced a succinct representation of RDF/S schema graphs that ensures fast query/view subsumption checking
- designed a distributed lookup service
- experimentally evaluated our framework to demonstrate that it scales gracefully

**Future Work:**
- extend with distributed query planning capabilities
- interleaved execution of routing/planning phases
- distributed trees?
Thank you!