Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks

Zoi Kaoudi
Postdoctoral researcher

Work done in the context of my PhD thesis at the National and Kapodistrian University of Athens
Outline

• Introduction
• The system Atlas
• RDFS Reasoning in P2P networks
• SPARQL Query Processing and Optimization
• Conclusions
Web of Documents/Web of Data

Zoi Kaoudi

Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks

7 October 2011
Resource Description Framework

- RDF provides a simple and abstract knowledge representation for resources on the Web which are uniquely identified by *Universal Resource Identifiers* (URIs)

- RDF Schema (RDFS) gives meaning to RDF terms and groups them to concepts

- SPARQL: official W3C recommendation language for querying RDF data
Zoi Kaoudi

Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks

7 October 2011
RDF Data Management

• With the vast amount of available RDF data sources on the Web increasing rapidly, there is an urgent need for RDF data management

• RDF storage, query processing and reasoning have been at the center of attention during the last years
RDF Data Management

• Centralized solutions
  • Jena, Sesame, RSSDB, Oracle, RDF-3X, etc.

• Parallel and distributed solutions
  – P2P systems
    • RDFPeers, BabelPeers, GridVine, Edutella etc.
  – Powerful clusters
    • Virtuoso, YARS2, MaRVIN
  – Cloud computing
    • Mika08, Urbani09, Stein10
Peer-to-peer (P2P) systems

- P2P systems provide nice features for Internet-scale applications (e.g., content sharing, distributed digital libraries)

- Distributed Hash Tables (DHTs)
  - answer exact match queries efficiently
Challenges

• How to index RDF data and RDFS ontologies in a DHT?
  ➢ *Triple indexing scheme and handle RDF and RDFS uniformly*
  ➢ *Distributed mapping dictionary*

• How to answer SPARQL queries efficiently? What kinds of query optimization techniques to use?
  ➢ *Greedy optimization algorithms to minimize the size of the intermediate results*
  ➢ *Selectivity estimation techniques*
  ➢ *RDF statistics in a DHT system*

• How to provide reasoning mechanisms for RDFS in a DHT?
  ➢ *Distributed forward chaining (FC)*
  ➢ *Distributed backward chaining (BC)*
  ➢ *Distributed magic sets transformation (MS)*
  ➢ *Comparative study (analytically and experimentally)*
  ➢ *Theoretical proofs of correctness*
Outline

• Introduction

• The system Atlas

• RDFS Reasoning in P2P networks

• SPARQL Query Processing and Optimization

• Conclusions
RDF Data Management in P2P networks: the system Atlas
The system Atlas

- **Atlas**: a full-blown open source P2P system for the distributed processing of RDF and RDFS data stored on top of DHTs ([http://atlas.di.uoa.gr](http://atlas.di.uoa.gr))

- Atlas has been used in EU projects OntoGrid and SemsorGrid4Env as a distributed registry of metadata
Indexing in Atlas

RDF triple \( t = (\text{zoi}, \text{advisor}, \text{manolis}) \)

Index identifier = (Hash(\text{zoi}))

Index identifier = (Hash(\text{advisor}))

Index identifier = (Hash(\text{manolis}))

RDF data and RDFS ontologies are handled **uniformly**

No global knowledge about the schema
Outline

• Introduction

• The system Atlas

• **RDFS Reasoning in P2P networks**

• **SPARQL Query Processing and Optimization**

• Conclusions
RDFS Reasoning

Query: Find all artists
q = (?x, rdf:type, artist)

RDFS entailment rules

Answer

<table>
<thead>
<tr>
<th>?x</th>
</tr>
</thead>
<tbody>
<tr>
<td>rembrandt</td>
</tr>
<tr>
<td>picasso</td>
</tr>
<tr>
<td>michelangelo</td>
</tr>
</tbody>
</table>
RDFS reasoning techniques

• Bottom-up approach
  – compute all inferences a priori (RDFS closure)

• Top-down approach
  – compute inferences at query run time

• Optimized bottom-up approach
  – compute inferences a priori given a specific query
Outline

• Introduction

• The system Atlas

• RDFS Reasoning in P2P networks
  – Bottom-up approach
  – Top-down approach
  – Optimized bottom-up approach
  – Evaluation

• SPARQL Query Processing and Optimization

• Conclusions
Bottom-up approach

- Distributed *forward chaining* algorithm
  - compute all inferences a priori (RDFS closure)
  - many redundant triples
Distributed Forward Chaining (FC)
Distributed Forward Chaining (FC)

Key

<table>
<thead>
<tr>
<th>Key</th>
<th>Responsible peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>n6</td>
</tr>
<tr>
<td>artist</td>
<td>n1</td>
</tr>
<tr>
<td>painter</td>
<td>n3</td>
</tr>
<tr>
<td>flemish</td>
<td>n5</td>
</tr>
</tbody>
</table>
Distributed Forward Chaining (FC)

Key Responsible peer
----------------------
person  n6
artist   n1
painter n3
flemish n5

Zoi Kaoudi Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks 7 October 2011
Distributed Forward Chaining (FC)

Key Responsible peer

<table>
<thead>
<tr>
<th>Key</th>
<th>Responsible peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>n6</td>
</tr>
<tr>
<td>artist</td>
<td>n1</td>
</tr>
<tr>
<td>painter</td>
<td>n3</td>
</tr>
<tr>
<td>flemish</td>
<td>n5</td>
</tr>
</tbody>
</table>

Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks

Zoi Kaoudi

7 October 2011
Distributed Forward Chaining (FC)

The same triple is generated in two nodes

<table>
<thead>
<tr>
<th>Responsible peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
</tr>
<tr>
<td>artist</td>
</tr>
<tr>
<td>painter</td>
</tr>
<tr>
<td>flemish</td>
</tr>
</tbody>
</table>

Zoi Kaoudi
Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks
7 October 2011
Distributed Forward Chaining (FC)

The same triple is sent to be stored twice

```
rdfs11
uuu rdfs:subClassOf vvv
vvv rdfs:subClassOf xxx
→ uuu rdfs:subClassOf xxx
```
RDFS Entailment Rules

(Only the RDFS rules proposed in the minimal deductive system mrdf of [Munoz et al.2009])

- `subClass(X,Y) :- triple(X, rdfs:subClassOf, Y).`
- `subClass(X,Y) :- triple(X, rdfs:subClassOf, Z), subClass(Z, Y).`
  
  **edb relation: triple**

- `subProperty(X, Y) :- triple(X, rdfs:subPropertyOf, Y).`
- `subProperty(X, Y) :- triple(X, rdfs:subPropertyOf, Z), subProperty(Z, Y).`
  
  **idb relations: subClass, subProperty, newTriple, type**

- `newTriple(X, P, Y) :- triple(X, P, Y).`
- `newTriple(X, P, Y) :- triple(X, P1, Z), subProperty(P1, P).`

- `type(X, Y) :- triple(X, rdfs:domain, Y).`
- `type(X, Y) :- triple(X, P, Z), triple(P, rdfs:domain, Y).`
- `type(X, Y) :- triple(Z, P, X), triple(P, rdfs:range, Y).`
Distributed Forward Chaining (FC*)

subClass(X,Y) :- triple(X, rdfs:subClassOf, Y).
subClass(X,Y) :- triple(X, rdfs:subClassOf, Z), subClass(Z, Y).

Key | Responsible peer
---|------------------
person | n6
artist | n1
painter | n3
flemish | n5
Outline

• Introduction

• The system Atlas

• RDFS Reasoning in P2P networks
  – Bottom-up approach
  – Top-down approach
  – Optimized bottom-up approach
  – Evaluation

• SPARQL Query Processing and Optimization

• Conclusions
Top-down approach

- Distributed *backward chaining* algorithm
  - store only the given triples
  - compute necessary inferences at query run time
Distributed Backward Chaining (BC)

Query: Find all subclasses of person

\[ q = (X, sc, \text{person}) \]

<table>
<thead>
<tr>
<th>Key</th>
<th>Responsible peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>n6</td>
</tr>
<tr>
<td>artist</td>
<td>n1</td>
</tr>
<tr>
<td>painter</td>
<td>n3</td>
</tr>
<tr>
<td>sculptor</td>
<td>n4</td>
</tr>
</tbody>
</table>

```
(painter, sc, artist)
(sculptor, sc, artist)
(artisit, sc, person)
```
Distributed Backward Chaining (BC)

Query: Find all subclasses of person
q = (X, sc, person)

Key Responsible peer
person n6
artist n1
painter n3
sculptor n4

Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks

Zoi Kaoudi
7 October 2011
Distributed Backward Chaining (BC)

subClass (X, person)

Which predicate should we evaluate first?

subClass (X,Y) :- triple (X, rdfs:subClassOf, Y).
subClass (X,Y) :- triple(X, rdfs:subClassOf, Z), subClass(Z, Y).

Evaluate first the predicate that can be evaluated locally!
RDFS Entailment Rules - revisited

• Recursive rules
• Rule adornment from recursive query processing
  – Good orderings for evaluating predicates
  – e.g. subClass(X, artist) $\rightarrow$ subClass$^{fb}$ (X,Y)
• Extended adornment
  – Ordered string of $f$, $b$, $k$
    • $k$ : an argument that is bound and the key
    • $b$ : bound argument (not the key)
    • $f$ : free argument
  – e.g. At node responsible for key artist:
    triple(X, rdf:type, artist) $\rightarrow$ triple$^{fbk}$ (X, rdf:type, Y).

  – Good ordering for evaluating predicates in a distributed environment
RDFS Entailment Rules - revisited

- subClass_{kf} (X,Y) :- triple_{kbf} (X, rdfs:subClassOf, Y).
- subClass_{kf} (X,Y) :- triple_{kbf} (X, rdfs:subClassOf, Z), subClass_{ff} (Z, Y).
- subClass_{fk} (X,Y) :- triple_{fbk} (X, rdfs:subClassOf, Y).
- subClass_{fk} (X,Y) :- subClass_{ff}(X, Z), triple_{fbk} (Z, rdfs:subClassOf, Y).

- subProperty_{kf} (X,Y) :- triple_{kbf} (X, rdfs:subPropertyOf, Y).
- subProperty_{kf} (X,Y) :- triple_{kbf} (X, rdfs:subPropertyOf, Z), subProperty_{ff} (Z, Y).
- subProperty_{fk} (X,Y) :- triple_{fbk} (X, rdfs:subPropertyOf, Y).
- subProperty_{fk} (X,Y) :- subProperty_{ff}(X, Z), triple_{fbk} (Z, rdfs:subPropertyOf, Y).

- type_{kf} (X, Y) :- triple_{kbf} (X, rdf:type, Y).
- type_{kf} (X, Y) :- triple_{kff} (X, P, Z), triple_{fbf} (P, rdfs:domain, Y).
- type_{kf} (X, Y) :- triple_{ffk} (Z, P, X), triple_{fbf} (P, rdfs:range, Y).
- type_{kf} (X, Y) :- triple_{kbf} (X, rdf:type, Z), subClass_{ff} (Z, Y).
- type_{fk} (X, Y) :- triple_{fbk} (X, rdf:type, Y).
- type_{fk} (X, Y) :- triple_{ff} (X, P, Z), triple_{fbk} (P, rdfs:domain, Y).
- type_{fk} (X, Y) :- triple_{ff} (Z, P, X), triple_{fbk} (P, rdfs:range, Y).
- type_{fk} (X, Y) :- type_{ff} (X, Z), triple_{fbk} (Z, rdfs:subClassOf, Y).
Distributed Backward Chaining (BC)

subClass $^k$ (X, person)

<table>
<thead>
<tr>
<th>Key</th>
<th>Responsible peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>n6</td>
</tr>
<tr>
<td>artist</td>
<td>n1</td>
</tr>
<tr>
<td>painter</td>
<td>n3</td>
</tr>
<tr>
<td>sculptor</td>
<td>n4</td>
</tr>
</tbody>
</table>
Distributed Backward Chaining (BC)

\[
\text{subClass}^f_k (X, \text{person})
\]

\[
\text{triple}^f_b (X, \text{rdfs:subClassOf}, \text{person})
\]

\[
\text{triple}^f_b (Z, \text{rdfs:subClassOf}, \text{person})
\]

\[
\text{subClass}^f (X, Z)
\]

\[
\text{subClass}^f_k (X, Y) :- \text{triple}^f_b (X, \text{rdfs:subClassOf}, Y). \quad (r1)
\]

\[
\text{subClass}^f_k (X, Y) :- \text{subClass}^f (X, Z), \text{triple}^f_b (Z, \text{rdfs:subClassOf}, Y). \quad (r2)
\]

<table>
<thead>
<tr>
<th>Key</th>
<th>Responsible peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>n6</td>
</tr>
<tr>
<td>artist</td>
<td>n1</td>
</tr>
<tr>
<td>painter</td>
<td>n3</td>
</tr>
<tr>
<td>sculptor</td>
<td>n4</td>
</tr>
</tbody>
</table>
Distributed Backward Chaining (BC)

\[ \text{subClass}^k(X, \text{person}) \]

\[ \text{triple}^{fbk}(X, \text{rdfs:subClassOf}, \text{person}) \]

\[ \text{triple}^{fbk}(Z, \text{rdfs:subClassOf}, \text{person}) \]

\[ \text{subClass}^f(X, Z) \]

\[ \text{subClass}^k(X, \text{artist}) \]

\[ \text{triple}^{fbk}(X, \text{rdfs:subClassOf}, \text{artist}) \]

\[ \text{triple}^{fbk}(Z, \text{rdfs:subClassOf}, \text{artist}) \]

\[ \text{subClass}^f(X, Z) \]

\[ \text{subClass}^k(X, \text{sculptor}) \]

\[ \text{triple}^{fbk}(X, \text{rdfs:subClassOf}, \text{sculptor}) \]

\[ \text{subClass}^f(X, Z) \]

\[ \text{Z / artist} \]

\[ \text{Z / painter} \]

\[ \text{Z / sculptor} \]

Key

<table>
<thead>
<tr>
<th>Key</th>
<th>Responsible peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>n6</td>
</tr>
<tr>
<td>artist</td>
<td>n1</td>
</tr>
<tr>
<td>painter</td>
<td>n3</td>
</tr>
<tr>
<td>sculptor</td>
<td>n4</td>
</tr>
</tbody>
</table>

Zoi Kaoudi
Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks
7 October 2011
Distributed Backward Chaining (BC)

subClass ^P (X, person)

triple ^P (X, rdfs:subClassOf, person) triple ^P (Z, rdfs:subClassOf, person) subClass ^P (X, Z)

Z / artist

subClass ^P (X, artist)

triple ^P (X, rdfs:subClassOf, artist) triple ^P (Z, rdfs:subClassOf, artist) subClass ^P (X, Z)

Z / painter

subClass ^P (X, painter)

triple ^P (X, rdfs:subClassOf, painter) triple ^P (Z, rdfs:subClassOf, painter) subClass ^P (X, Z)

Z / sculptor

subClass ^P (X, sculptor)

triple ^P (X, rdfs:subClassOf, sculptor) triple ^P (Z, rdfs:subClassOf, sculptor) subClass ^P (X, Z)

Key

<table>
<thead>
<tr>
<th>Responsible peer</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
</tr>
<tr>
<td>artist</td>
</tr>
<tr>
<td>painter</td>
</tr>
<tr>
<td>sculptor</td>
</tr>
</tbody>
</table>

Zoi Kaoudi

Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks

7 October 2011
Outline

• Introduction

• The system Atlas

• **RDFS Reasoning in P2P networks**
  – Bottom-up approach
  – Top-down approach
  – **Optimized bottom-up approach**
  – Evaluation

• SPARQL Query Processing and Optimization

• Conclusions
Forward chaining for magic rules (MS)

- Magic sets transformation technique
  - Rewrite rules given a specific query
  - Bottom-up evaluation (forward chaining)
  - No unnecessary information is inferred

<table>
<thead>
<tr>
<th>Rule</th>
<th>Head</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>_a owl:tri(s(F), P, T)</td>
<td></td>
</tr>
</tbody>
</table>

Query: Find all painters
q = (?x, rdfs:characteristic, painter)
Outline

• Introduction

• The system Atlas

• **RDFS Reasoning in P2P networks**
  – Bottom-up approach
  – Top-down approach
  – Optimized bottom-up approach
  – **Evaluation**

• SPARQL Query Processing and Optimization

• Conclusions
Experimental setup

- Algorithms have been implemented in Atlas system using Bamboo DHT [Rhea04]

- Testbeds
  - PlanetLab (~210 nodes available at the time of the experiments)
  - local cluster (30 machines – 4 Atlas nodes per machine)

- Datasets
  - Synthetic data from RBench generator [Theoharis05]
    - Number of instances: 10,000 to 1M
    - RDFS class hierarchy tree depth: 2-6 (7 to 128 classes)
    - Query: Give me the instances of the root class
  - LUBM benchmark
    - Number of triples: ~110,000 to ~2,700,00
    - Query: Give me the instances of a class
Store time and network traffic

PlanetLab, RBench with 10,000 instances

Time (min) vs. Depth

Bandwidth (MB) vs. Depth
Query response time

PlanetLab, RBench with 10,000 instances

- BC
- BC cache
- FC

Depth vs. time (sec)

Zoi Kaoudi
Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks
7 October 2011
Backward chaining vs. magic sets

Cluster: RBench with 1M instances
Outline

• Introduction

• The system Atlas

• RDFS Reasoning in P2P networks

• **SPARQL Query Processing and Optimization**

• Conclusions
• SPARQL queries of basic graph patterns

SELECT ?x ?y ?z
WHERE {
  ?x rdf:type ub:Student . (tp1)
  ?x ub:takesCourse ?z . (tp2)
  ?x ub:advisor ?y . (tp3)
  ?y ub:teacherOf ?z . (tp4)
}

– conjunctive triple patterns

tp1 ^ tp2 ^ tp3 ^ tp4
SPARQL query processing

- SPARQL queries of basic graph patterns

```sql
SELECT ?x ?y ?z
WHERE {
  ?x rdf:type ub:Student . (tp1)
  ?x ub:takesCourse ?z . (tp2)
  ?x ub:advisor ?y . (tp3)
  ?y ub:teacherOf ?z (tp4)
}
```
Query Optimization

• Find a query plan that optimizes the performance of a system with respect to a metric of interest
  
  Query response time  Bandwidth consumption

• Find a good ordering of the triple patterns

Minimize the size of intermediate results

Lower bandwidth consumption

Joins with smaller intermediate relations
Query Optimization

- Greedy optimization algorithms
  - selectivity-based heuristics
  - minimize the size of intermediate results
Outline

• Introduction

• The system Atlas

• RDFS Reasoning in P2P networks

• SPARQL Query Processing and Optimization
  – Selectivity estimation
  – Statistics for RDF
  – Optimization algorithms
  – Evaluation

• Conclusions
Selectivity estimation

• Single triple pattern
  – Bound-is-easier heuristic
  – Analytical estimation: \( sel(tp) = sel(s) \times sel(p) \times sel(o) \)

\[
sel(v) = \frac{freq_c(v)}{T} \quad \text{frequency of value } v \text{ as a component } c
\]

\[
T \quad \text{total triples stored in the network}
\]

• Joins of triple patterns: \( sel(tp_1 \land tp_2) = \frac{join_card(tp_1, tp_2)}{T^2} \)

\[
join_card(tp_1, tp_2) = \frac{T_{tp_1} \times T_{tp_2}}{\max(I_{tp_1(?x)}, I_{tp_2(?x)})} \quad \text{number of triples matching } tp_1, tp_2
\]

\[
\text{size of the domain of } ?x \text{ in } tp_i
\]
Outline

• Introduction

• The system Atlas

• RDFS Reasoning in P2P networks

• **SPARQL Query Processing and Optimization**
  – Selectivity estimation
  – **Statistics for RDF**
  – Optimization algorithms
  – Evaluation

• Conclusions
Required statistics

• Frequency of a triple component $c$ with value $v$ ($freq_c(v)$)

• Size of the domain of a variable in a triple pattern ($I_{tpi(?x)}$)
Frequency and variable domain size - example

(\(?x, \text{advisor}, ?y\))
frequency of advisor \(\rightarrow\) \# occurrences of advisor as a predicate \(\text{freq}_p(\text{advisor})\)
domain size of \(?x\) \(\rightarrow\) \# distinct subject values of predicate advisor \(\text{ds}_p(\text{advisor})\)
domain size of \(?y\) \(\rightarrow\) \# distinct object values of predicate advisor \(\text{do}_p(\text{advisor})\)

RDF triples

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>zoi</td>
<td>advisor</td>
<td>manolis</td>
</tr>
<tr>
<td>iris</td>
<td>advisor</td>
<td>manolis</td>
</tr>
<tr>
<td>maria</td>
<td>advisor</td>
<td>mike</td>
</tr>
<tr>
<td>zoi</td>
<td>takesCourse</td>
<td>db</td>
</tr>
<tr>
<td>zoi</td>
<td>takesCourse</td>
<td>sw</td>
</tr>
<tr>
<td>manolis</td>
<td>teacherOf</td>
<td>sw</td>
</tr>
</tbody>
</table>

Statistics for predicates

<table>
<thead>
<tr>
<th>Value</th>
<th>freq(_p)</th>
<th>ds(_p)</th>
<th>do(_p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>advisor</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>takesCourse</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>teacherOf</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Statistics in DHTs

Each peer is responsible for creating and maintaining statistics of its **locally** stored data and only for triple component values which are **key** at the specified peer.
Statistics at each DHT peer

• Each peer creates statistics for each triple component separately
  – distinguish resource objects from class objects

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
<th>object-class</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq_s</td>
<td>freq_p</td>
<td>freq_o</td>
<td>freq_c</td>
</tr>
<tr>
<td>dp_s</td>
<td>ds_p</td>
<td>ds_o</td>
<td>-</td>
</tr>
<tr>
<td>do_s</td>
<td>dp_o</td>
<td>dp_o</td>
<td>-</td>
</tr>
</tbody>
</table>

• Given a space budget, each peer can keep the exact frequency distribution or an estimation by using v-optimal histograms
Outline

• Introduction

• The system Atlas

• RDFS Reasoning in P2P networks

• SPARQL Query Processing and Optimization
  – Selectivity estimation
  – Statistics for RDF
  – Optimization algorithms
  – Evaluation

• Conclusions
Optimization algorithms

• Static optimization
  – Before the query evaluation begins, at the peer that receives the query request
    • Naïve optimization algorithm (NA): Orders triple patterns based on their selectivity, from the most selective to the least selective
    • Semi-naïve optimization algorithm (SNA): Orders triple patterns based on the join selectivity between pairs of triple patterns
Optimization algorithms

- **Dynamic optimization**
  - During the query evaluation, at each peer participating in the query processing
  - **Dynamic optimization algorithm (DA):** join selectivity is computed at each peer participating in the query processing and estimated between the real intermediate results so far and a triple pattern

- All algorithms (static and dynamic) use a query graph (QG) representation to avoid Cartesian products
Outline

• Introduction

• The system Atlas

• RDFS Reasoning in P2P networks

• **SPARQL Query Processing and Optimization**
  – Selectivity estimation
  – Statistics for RDF
  – Optimization algorithms
  – **Evaluation**

• Conclusions
Experimental setup

• All algorithms and techniques have been implemented in Atlas
  – Mapping dictionary

• We used as a testbed PlanetLab and a local cluster server blade machines with two processors at 2.6GHz and 4GB memory 30 available machines – up to 4 nodes per machine → up to 120 DHT peers

• LUBM Benchmark (queries with more than 4 triple patterns)
Query response time (LUBM-50)

?X ub:takesCourse ?Y}

Q2 query plan space (LUBM-10)

Query response time of Q2 for increasing dataset size

![Graph showing query response time (QRT) vs. triples stored (x1M) for different methods: QG, NA, NA, SNA, DA. The graph indicates an upward trend in QRT as the number of triples increases, with QG consistently having the lowest QRT.]
Optimization time (LUBM-50)

- Q2: NA, SNA, DA
- Q4: NA, SNA, DA
- Q7: NA, SNA, DA
- Q8: NA, SNA, DA
- Q9: NA, SNA, DA

LUBM Query

Optimization (msec)
Mapping dictionary

![Graph showing QRT (sec) vs triples stored (x1M)]

- **With dictionary**
- **No dictionary**

Zoi Kaoudi

Distributed RDF Query Processing and Reasoning in Peer-to-Peer Networks

7 October 2011
Outline

• Introduction

• The system Atlas

• RDFS Reasoning in P2P networks

• SPARQL Query Processing and Optimization

• Conclusions
Conclusions

• **Atlas**: a P2P system for the distributed query processing and reasoning of RDF and RDFS data which is built on top of DHTs

• RDFS Reasoning on top of DHTs
  – Distributed forward chaining
  – Distributed backward chaining
  – Distributed magic sets transformation
  – Comparative study (analytically and experimentally)
  – Theoretical proofs of correctness

• SPARQL Query Processing and Optimization on top of DHTs
  – Selectivity estimation techniques
  – Statistics for RDF
  – Optimization algorithms
  – Distributed mapping dictionary
Future directions

• Distributed recursive queries using Datalog

• RDF in the cloud!
Thank you

Questions?

References

Z. Kaoudi, K. Kyzirakos and M. Koubarakis. SPARQL Query Optimization on Top of DHTs. In 9th International Semantic Web Conference (ISWC 2010), Shanghai, China, November 7-11, 2010


Z. Kaoudi and M. Koubarakis. Distributed RDFS Reasoning over Structured Overlay Networks. Extended version of ISWC 2008 submitted to ACM TWEB