A Hybrid Approach to Linked Data Query Processing with Time Constraints

Steven Lynden, Isao Kojima, Akiyoshi Matono, Akihito Nakamura, Makoto Yui

National Institute of Advanced Industrial Science and Technology, Japan
Motivation

• Indexing systems, e.g. Sindice, can be used to query the Semantic Web, however:
  – *Hybrid SPARQL queries: fresh vs. fast results* - Umbrich et al.
    • Coherence
    • A significant proportion of data from Sindice etc. may not be up-to-date with sources.

• Existing distributed SPARQL query processing systems are often very unpredictable in terms of response time.

• Some applications may require a best effort in a fixed amount of time
  – e.g. a portal for browsing a Linked Data repository attempting to suggest related RDF data from other sources requiring answers from a query processing back-end within the average time a user stays on a page.
Proposed approach

• Execute two components in parallel
  – Active discovery
    • Investigate URIs, retrieve RDF data, match against triple patterns in the query applying FILTER predicates
  – Query SPARQL endpoints
    • Construct sub-queries from the federated query, execute them using available SPARQL endpoints

• Both components share a local graph data structure in which a temporary result is constructed

• After a set time period, both components terminated and the local graph transformed into a query result
Hybrid Query Processing with Time Constraints

- Compile Query
- Access SPARQL endpoints and documents containing RDF data
- Stop and evaluate
User’s SPARQL Query

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dbp: <http://dbpedia.org/resource/property>
SELECT * WHERE {
  ?x dc:subject dbp:FIFA World Cup-winning countries .
  ?p foaf:name "Luiz Felipe Scolari"@en .
}
```
Implementation

- Standard Java Libraries
- Active Discovery Manager
- Query Compilation
- Jena 2.7.1
- Endpoint Query Manager
- Evaluation
- ADERIS

Query Result
Endpoint Query Manager

- Prior to query execution the system is configured with a set of endpoints to be used
- Existence of triples with a given predicate assumed to be known, e.g:

  triple pattern matches exist in the data.semanticweb.org endpoint  
  (Predicates in query triple patterns are usually not variables)

- Objectives
  - Execute simple queries to provide results quickly that can be explored by the active discovery manager in parallel
  - Avoid placing excessive burden on endpoints and avoid fair-use restrictions
for all $tp_i \in TP$ do
  if predMatch($tp_i$, endpoint) then
    sq.addPattern($tp_i$)
  end if
end for

First Query?

Add all applicable triple patterns

Yes

No

For each query variable bound in the local graph, create a sub-query with bindings and add applicable triple patterns

For each query variable bound in the local graph, create a sub-query with bindings and add applicable triple patterns

Select sub-query with highest estimated value

$\text{value}(sq) = \sum_{\text{var} \in V} \frac{s}{\max(1, \text{localGraph.numBindings}(\text{var}))}$

Variables in the sub-query

Number of bound values in the FILTER

Send Query
Active Discovery Manager

• The active discover manager starts a thread for each Pay Level Domain (PLD) present in URIs in the query and as they are added to the local graph.
• Each thread is able to choose two URIs to investigate each second.

• Objective:
  – Match triple patterns in the query with RDF data retrieved via dereferencing the URIs
For all URIs in triple patterns in the query:

- If triple pattern variables are bound, add to S2
- If triple pattern contains non-bound variables, add to S1

DBpedia URIs investigated and the number of triples matching triple patterns in the query.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX dbp: <http://dbpedia.org/resource/property>
SELECT * WHERE {
    ?x dc:subject dbp:FIFA_World_Cup-winning_countries .
    ?p foaf:name "Luiz Felipe Scolari"@en .
}
```

/resource/England_national_football_team (25)
/resource/Spain_national_football_team (23)
/resource/Brazil_national_team (18)
/resource/FC_Bunyodkor (3)
/resource/Vicente_Feola (2)
/resource/Luiz_Felipe_Scolari (16)

Yes

Select bestRanking(S2)

No

Select bestRanking(S1)

$$\text{rank(uri)} = \sum_{u \in U} [\text{matched}(u) \times (1 - \text{distance}(u, uri))]$$

Levenshtein distance
Evaluation

• FedBench
  – Benchmark for testing the efficiency and effectiveness of federated query processing on semantic data.
• Multiple query sets, we used the Linked Data (LD) query set.
• 11 Queries, however some problems encountered with 2 of the queries.
• Remaining queries executed using the proposed approach with a limit of 10 seconds.
<table>
<thead>
<tr>
<th>Query</th>
<th># Triples retrieved</th>
<th># Eval time</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>17</td>
<td>36ms</td>
<td>297</td>
</tr>
<tr>
<td>Query 2</td>
<td>17</td>
<td>9ms</td>
<td>147</td>
</tr>
<tr>
<td>Query 3</td>
<td>274</td>
<td>18ms</td>
<td>304</td>
</tr>
<tr>
<td>Query 4</td>
<td>296</td>
<td>7ms</td>
<td>50</td>
</tr>
<tr>
<td>Query 5</td>
<td>4</td>
<td>13ms</td>
<td>241</td>
</tr>
<tr>
<td>Query 7</td>
<td>4</td>
<td>3ms</td>
<td>1</td>
</tr>
<tr>
<td>Query 9</td>
<td>65</td>
<td>3ms</td>
<td>1</td>
</tr>
<tr>
<td>Query 10</td>
<td>58</td>
<td>4ms</td>
<td>3</td>
</tr>
<tr>
<td>Query 11</td>
<td>189</td>
<td>36ms</td>
<td>892</td>
</tr>
</tbody>
</table>
Conclusions

• Answering the FedBench Linked Data queries in accordance with our objective of within 10 seconds was possible using the proposed technique.

• Advantages include:
  – Fault tolerance
  – Freshness
  – Increased coverage
  – Mitigation of fair-use restrictions

• Future work will investigate benefits with more dynamic data, e.g. RDFa etc and optimisation based on relevance /quality of data sources