FedX:
Optimization Techniques for Federated Query Processing on Linked Data

ISWC 2011 – October 26th

Andreas Schwarte\textsuperscript{1}, Peter Haase\textsuperscript{1}, Katja Hose\textsuperscript{2}, Ralf Schenkel\textsuperscript{2}, and Michael Schmidt\textsuperscript{1}

\textsuperscript{1}fluid Operations AG, Walldorf
\textsuperscript{2}Max-Planck Institute for Informatics, Saarbrücken
Outline

- Motivation
- Federated Query Processing
- FedEx Query Processing Model
- Optimization Techniques
- Evaluation
- Conclusion & Outlook
Motivation

Query processing involving multiple distributed data sources, e.g. Linked Open Data cloud

Query both data collections in an integrated way
Federated Query Processing

Federation mediator at the server

- Virtual integration of (remote) data sources
- Communication via SPARQL protocol
Federated Query Processing

Example Query
Find US presidents and associated news articles

```sparql
SELECT ?President ?Party ?TopicPage WHERE {
  ?President rdf:type dbpedia-yago:PresidentsOfTheUnitedStates .
  ?nytPresident owl:sameAs ?President .
}
```
Federated Query Processing

Query:

SELECT ?President ?Party ?TopicPage WHERE {
  ?President rdf:type dbpedia-yago:PresidentsOfTheUnitedStates .
  ?nytPresident owl:sameAs ?President .
  ...
}

Federation Mediator

?President rdf:type dbpedia-yago:PresidentsOfTheUnitedStates .

“Barack Obama”
“George W. Bush”
...
Federated Query Processing

Query:

SELECT ?President ?Party ?TopicPage WHERE {
  ?President rdf:type dbpedia-yago:PresidentsOfTheUnitedStates .
  ?nytPresident owl:sameAs ?President .
  ...
}

Input:

“Barack Obama”
“George W. Bush”
...

Output:

“Barack Obama”, yago:Obama
“Barack Obama”, nyt:Obama
Federated Query Processing

Query:

```sparql
SELECT ?President ?Party ?TopicPage WHERE {
?President rdf:type dbpedia-yago:PresidentsOfTheUnitedStates .
...
}
```

?nytPresident owl:sameAs “George W. Bush” .

Input:

“Barack Obama”, yago:Obama
“George W. Bush”, nyt:Bush
...

Output:

“Barack Obama”, yago:Obama
“Barack Obama”, nyt:Obama
“George W. Bush”, nyt:Bush

... and so on for the other intermediate mappings and triple patterns ...
FedX Query Processing Model

Scenario:

- Efficient SPARQL query processing on multiple distributed sources
- Data sources are known and accessible as SPARQL endpoints
  - FedX is designed to be fully compatible with SPARQL 1.0
- No a-priori knowledge about data sources
  - No local preprocessing of the data sources required
  - No need for pre-computed statistics
- On-demand federation setup
Challenges to Federated Query Processing

1) Involve only relevant sources in the evaluation
   
   **Avoid:** Subqueries are sent to all sources, although potentially irrelevant

2) Compute joins close to the data
   
   **Avoid:** All joins are executed locally in a nested loop fashion

3) Reduce remote communication
   
   **Avoid:** Nested loop join that causes many remote requests
Optimization Techniques

1. Source Selection:
   Idea:
   Triple patterns are annotated with relevant sources
   - Sources that can contribute information for a particular triple pattern
   - SPARQL ASK requests in conjunction with a local cache
     - After a warm-up period the cache learns the capabilities of the data sources
       ➔ During Source Selection remote requests can be avoided

2. Exclusive Groups:
   Idea:
   Group triple patterns with the same single relevant source
   - Evaluation in a single (remote) subquery
   - Push join to the endpoint
Optimization Techniques (2)

Example: Source Selection + Exclusive Groups

```
SELECT ?President ?Party ?TopicPage WHERE {
  ?President rdf:type dbpedia-yago:PresidentsOfTheUnitedStates .
  ?nytPresident owl:sameAs ?President .
}
```

Advantages:

- Avoid sending subqueries to sources that are not relevant
- Delegate joins to the endpoint by forming exclusive groups (i.e. executing the respective patterns in a single subquery)
Optimization Techniques (3)

3. Join Order:

Idea:

Iteratively determine the join order based on count-heuristic:

- Count free variables of triple patterns and groups
- Consider "resolved" variable mappings from earlier iteration

4. Bound Joins:

Idea:

Compute joins in a block nested loop fashion:

- Reduce the number of requests by "vectored" evaluation of a set of input bindings
- Renaming and Post-Processing technique for SPARQL 1.0
Optimization Techniques (4)

Example: Bound Joins

SELECT ?President ?Party ?TopicPage WHERE {
    ?President rdf:type dbpedia:PresidentsOfTheUnitedStates .
    ?nytPresident owl:sameAs ?President .
}

Assume that the following intermediate results have been computed as input for the last triple pattern

Block Input
“Barack Obama”
“George W. Bush”
...

Before (NLJ)
...

Now: Evaluation in a single remote request using a SPARQL UNION construct + local post processing (SPARQL 1.0)

SPARQL 1.1: BINDINGS clause
Optimization Example

1. **SPARQL Query**

   Compute *Micronutrients* using Drugbank and KEGG

   ```sparql
   SELECT ?drug ?title WHERE {
     ?drug drugbank:casRegistryNumber ?id .
     ?keggDrug bio2rdf:xRef ?id .
   }
   ```

2. **Unoptimized Internal Representation**

   - 4x Local Join
   - 4x NLJ

3. **Optimized Internal Representation**

   **Exclusive Group**

   - Remote Join
Evaluation

Based on FedBench benchmark suite

- 14 queries from the Cross Domain (CD) and Life Science (LS) collections
- Real-World Data from the Linked Open Data cloud
- Federation with 5 (CD) and 4 (LS) data sources
- Queries vary in complexity, size, structure, and sources involved

Benchmark environment

- HP Proliant 2GHz 4Core, 32GB RAM
- 20GB RAM for server (federation mediator)
- Local copies of the SPARQL endpoint to ensure reproducibility and reliability of the service
  - Provided by the FedBench Framework
Evaluation (2)

a) Evaluation times of Cross Domain (CD) and Life Science (LS) queries

<table>
<thead>
<tr>
<th></th>
<th>AliBaba</th>
<th>DARQ</th>
<th>FedX</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD1</td>
<td>0.125</td>
<td>x</td>
<td>0.015</td>
</tr>
<tr>
<td>CD2</td>
<td>0.807</td>
<td>0.019</td>
<td>0.330</td>
</tr>
<tr>
<td>CD3</td>
<td>&gt;600</td>
<td>&gt;600</td>
<td>0.109</td>
</tr>
<tr>
<td>CD4</td>
<td>&gt;600</td>
<td>19.641</td>
<td>0.100</td>
</tr>
<tr>
<td>CD5</td>
<td>#</td>
<td>294.890</td>
<td>0.097</td>
</tr>
<tr>
<td>CD6</td>
<td>17.499</td>
<td>x</td>
<td>0.281</td>
</tr>
<tr>
<td>CD7</td>
<td>3.623</td>
<td>x</td>
<td>0.324</td>
</tr>
<tr>
<td>LS1</td>
<td>1.303</td>
<td>0.053</td>
<td>0.047</td>
</tr>
<tr>
<td>LS2</td>
<td>0.441</td>
<td>x</td>
<td>0.016</td>
</tr>
<tr>
<td>LS3</td>
<td>&gt;600</td>
<td>133.414</td>
<td>1.470</td>
</tr>
<tr>
<td>LS4</td>
<td>20.370</td>
<td>0.025</td>
<td>0.001</td>
</tr>
<tr>
<td>LS5</td>
<td>12.504</td>
<td>55.327</td>
<td>0.480</td>
</tr>
<tr>
<td>LS6</td>
<td>#</td>
<td>3.236</td>
<td>0.034</td>
</tr>
<tr>
<td>LS7</td>
<td>#</td>
<td>&gt;600</td>
<td>0.481</td>
</tr>
</tbody>
</table>
### Evaluation (3)

#### b) Number of requests sent to the endpoints

<table>
<thead>
<tr>
<th></th>
<th>AliBaba</th>
<th>DARQ</th>
<th>FedX</th>
<th>CBJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD1</td>
<td>27</td>
<td>x</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>CD2</td>
<td>22</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>CD3</td>
<td>(93,248)</td>
<td>(170,579)</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>CD4</td>
<td>(372,339)</td>
<td>22,331</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>CD5</td>
<td>(117,047)</td>
<td>247,343</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>CD6</td>
<td>6,183</td>
<td>x</td>
<td>185</td>
<td></td>
</tr>
<tr>
<td>CD7</td>
<td>1,883</td>
<td>x</td>
<td>138</td>
<td></td>
</tr>
<tr>
<td>LS1</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>LS2</td>
<td>61</td>
<td>x</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>LS3</td>
<td>(410)</td>
<td>101,386</td>
<td>2059</td>
<td></td>
</tr>
<tr>
<td>LS4</td>
<td>21,281</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LS5</td>
<td>16,621</td>
<td>2,666</td>
<td>458</td>
<td></td>
</tr>
<tr>
<td>LS6</td>
<td>(130)</td>
<td>98</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>LS7</td>
<td>(876)</td>
<td>(576,089)</td>
<td>485</td>
<td></td>
</tr>
</tbody>
</table>

**Runtimes**
- AliBaba: >600s
- DARQ: >600s
- FedX: 0.109s

**Runtimes**
- AliBaba: >600s
- DARQ: 133s
- FedX: 1.4s
FedX – The Bigger Picture

Information Workbench:
Integration of Virtualized Data Sources as a Service

Application Layer

Virtualization Layer

Data Layer

Semantic Wiki
Collaboration
Reporting & Analytics
Visual Exploration

Transparent & On-Demand
Integration of Data Sources

Data Registries
CKAN, data.gov, etc.
+ Enterprise Data

Metadata Registry
Conclusion and Outlook

FedX: Efficient SPARQL query processing in federated setting

- Available as open source software: http://www.fluidops.com/fedx
  - Implementation as a Sesame SAIL
- Novel join processing strategies, grouping techniques, source selection
  - Significant improvement of query performance compared to state-of-the-art systems
- On-demand federation setup without any preprocessing of data sources
- Compatible with existing SPARQL 1.0 endpoints

Outlook

- SPARQL 1.1 Federation Extension in Sesame 2.6
  - Ongoing integration of FedX’ optimization techniques into Sesame
- Federation Extensions in FedX
  - SERVICE keyword for optional user input to improve source selection
  - BINDINGS clauses for bound joins (for SPARQL 1.1 endpoints)
- (Remote) statistics to improve join order (e.g. VoID)
Visit our exhibitor booth or arrange a private live demo!

Further information on FedX
http://www.fluidops.com/fedx
Query Cross Domain 3 (CD3)

Find US presidents, their party and associated news

SELECT ?president ?party ?page WHERE {
?president <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/President> .
?x <http://www.w3.org/2002/07/owl#sameAs> ?president .
}
Query Life Science 3 (LS3)

Find drugs and interaction drugs (+ the effect)

}