RDF Graph Processing on MapReduce

Data Processing in MapReduce based Systems (Apache Pig):
- Input is unformatted raw data but usually organized as tuples
- Declarative queries are expressed in terms of relational-like operators.
- Queries are ultimately translated into MapReduce workflow execution plan
- Each MapReduce (MR) cycle is used to execute single or a group of related operators e.g. JOIN operations on same key
- Intermediate results materialized and transferred within and across cycles

RDF Data Processing in MapReduce:
- RDF data is modeled as binary relations requiring several join operations for reassembling related data
- RDF files contain a heterogeneous non-delineated collection of data + metadata of different relations
- General processing strategy: spend one MR cycle to preprocess data into vertically partitioned relations, then follow with join processing

Example RDF data

<table>
<thead>
<tr>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>name</td>
<td>V1</td>
</tr>
<tr>
<td>V1</td>
<td>type</td>
<td>OFFER</td>
</tr>
<tr>
<td>V1</td>
<td>vendor1</td>
<td>V1</td>
</tr>
<tr>
<td>V1</td>
<td>homepage</td>
<td><a href="http://www.vendors.org/V1">www.vendors.org/V1</a></td>
</tr>
<tr>
<td>V1</td>
<td>delDays</td>
<td>3</td>
</tr>
</tbody>
</table>

Example SPARQL query

```
?o vendor .
?o price ?price .
?o review ?rev .
?o rating ?trait.
FILTER (?delDays < 3)
```

Query Plan

Star-joins S1, S2, S3 require 3 MR cycles!!!

Query Processing in Pig

**TripleGroup based operators are re-factored and coalesced to minimize MR cycles**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SJ1</td>
<td>(V1, price, 108, (V1, label, vendor1), (V1, delDays, 2))</td>
</tr>
<tr>
<td>SJ2</td>
<td>(V1, price, 108, (V1, label, vendor1), (V1, homepage, <a href="http://www.vendors.org/V1">www.vendors.org/V1</a>))</td>
</tr>
<tr>
<td>SJ3</td>
<td>(V1, price, 108, (V1, label, vendor1), (V1, delDays, 2))</td>
</tr>
</tbody>
</table>

Graph Pattern Matching using NTGA

Comparison and mapping between relational and NTGA for the example query plan

**RAPID+ is an extension to Apache Pig to enable SPARQL query processing on Hadoop. It supports:**
- SPARQL query interface using Jena’s ARQ
- NTGA operators for efficient processing of SPARQL graph patterns
- Hybrid plans using both Pig Latin and NTGA operators

Nested TripleGroup Algebra (NTGA)

An algebra for more efficient processing of RDF graph patterns based on a nested TripleGroup model (more graph-like processing)

Set of RDF Triples

<table>
<thead>
<tr>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>type</td>
<td>OFFER</td>
</tr>
<tr>
<td>V1</td>
<td>vendor1</td>
<td>V1</td>
</tr>
<tr>
<td>V1</td>
<td>homepage</td>
<td><a href="http://www.vendors.org/V1">www.vendors.org/V1</a></td>
</tr>
<tr>
<td>V1</td>
<td>delDays</td>
<td>3</td>
</tr>
</tbody>
</table>

**Set of RDF Triples**

\[ (\{S\} \rightarrow \{T\}) \]

**Set of RDF Triples**

\[ (\{S\} \rightarrow \{T\}) \]

References