D-SPARQ: Distributed, Scalable and Efficient RDF Query Engine

Raghava Mutharaju¹, Sherif Sakr², Alessandra Sala³, and Pascal Hitzler¹
¹Kno.e.sis Center, Wright State University, Dayton, OH, USA.
²University of Dammam, Saudi Arabia and University of New South Wales, Australia.
³Alcatel-Lucent Bell Labs, Dublin, Ireland.

Why?
- There is an exponential increase in the amount of RDF data available.
- Even simple SPARQL queries involve multiple triple patterns.
- Joins of multiple triple patterns across large data is slow.
- Aim of this work is to efficiently handle join patterns at a scale.

What?
- Architecture of D-SPARQ is shown in the picture above.
- A graph is constructed from RDF data.
- Graph partitioner is used to spread the data across the cluster.
- MapReduce job helps in importing data into MongoDB.
- Triples on the partition boundary are replicated [1].
- We chose MongoDB, a document store, because a variety of compound indexes can be built, has good read/write performance and supports complex querying.

How?
- Many SPARQL queries have triple patterns joined on either subject or object. These triple patterns form a star [2].
- We take advantage of this by grouping triples with the same subject into one document (equivalent to row in RDBMS) in MongoDB.
- This ensures that we retrieve subject based star patterns in one read call.
- Compound indexes are created on subject-predicate and predicate-object pairs.
- Using these compound indexes, MongoDB can also answer queries on any prefix of the index.
- The given SPARQL query is analyzed to identify the following patterns.
  - Triple patterns which are independent.
  - Star patterns, i.e., patterns with the same subject.
  - Pipeline patterns, i.e., patterns which depend on the result of other patterns (subject of one is the object of another pattern).
- Identifying these patterns enable us to run different parts of the query in parallel.
- Another optimization is to use selectivity of triple patterns within a star pattern to reorder their execution.

Preliminary Results

<table>
<thead>
<tr>
<th>#Triples</th>
<th>Query 2</th>
<th>Query 3</th>
<th>Query 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RDF-3X</td>
<td>D-SPARQ</td>
<td>RDF-3X</td>
</tr>
<tr>
<td>77 million</td>
<td>217</td>
<td>192.5</td>
<td>80</td>
</tr>
<tr>
<td>163 million</td>
<td>1537</td>
<td>398</td>
<td>434</td>
</tr>
</tbody>
</table>

- RDF data is generated using SP²Bench [4]. Cluster consists of 3 nodes with 16GB RAM each. All the runtimes given are in seconds.
- The given triples are the average number of triples loaded into RDF-3X and MongoDB of each node in the cluster.
- So, the total number of triples in the first case is around 230 million and second case is around 490 million.
- We compared our system with RDF-3X [3], which runs on each node of the cluster.
- The three queries are from SP²Bench. We did not consider queries involving OPTIONAL, FILTER, ORDER.
- These SPARQL features are not supported by our system. Here we focus on efficient join operations.
- The query runtimes of D-SPARQ are significantly better than that of RDF-3X especially for large number of triples.
- We observed that as the number of triples increases, performance of RDF-3X decreases.

References