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

Raghava Mutharaju[1], Sherif Sakr[2], Alessandra Sala[3], and Pascal Hitzler[1]
[1]Kno.e.sis Center, Wright State University, Dayton, OH, USA.
[2]University of Dammam, Saudi Arabia and University of New South Wales, Australia.
[3]Alcatel-Lucent Bell Labs, Dublin, Ireland.

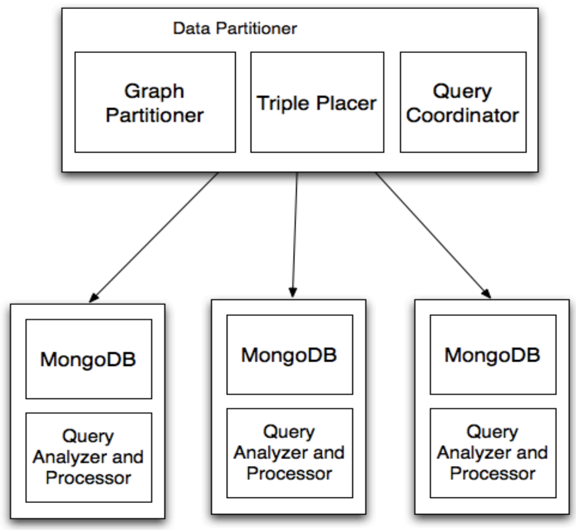
Why?

- 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.

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.

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.

References

[1] Huang, J., Abadi, D, J., Ren, K.: Scalable SPARQL Querying of Large RDF Graphs. PVLDB 4(11), 1123-1134 (2011).

[2] Kim, H., Ravindra, P., Anyanwu, K.: From SPARQL to MapReduce: The Journey Using a Nested TripleGroup Algebra. PVLDB 4(12), 1426-1429 (2011).

[3] Neumann, T., Weikum, G.: The RDF-3X engine for scalable management of RDF data. VLDB J. 19(1), 91-113 (2010).

[4] Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP² Bench: A SPARQL Performance Benchmark. In: ICDE. 222-233 (2009).

Preliminary Results

#Triples	Query2		Query3		Query4	
	RDF-3X	D-SPARQL	RDF-3X	D-SPARQ	RDF-3X	D-SPARQ
77 million	217	192.5	80	69.43	Out Of Memory	319.87
163 million	1537	398	434	166	Out Of Memory	671

- RDF data is generated using SP² Bench [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 SP² Bench. 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.