Computing Continuous SPARQL Query over RDF Streams on Storm Platform

By Usman Younas and Sander Breukink

# Introduction

The amount of data on the web has massively increased over the years, and it doesn’t show any signs of stopping. Hence the demand for analyzing it in reasonable amount of time is also increasing. The Resource Description Framework (RDF) is a data model whose purpose is to form a comprehensive framework to integrate data from different fields. It is a flexible data model used in the Semantic Web (a Web of data) on which we can do querying or reasoning. In this report we discuss several ways of executing SPARQL queries on RDF data and compare their results based on several testing scenarios. For this we will use the Apache Storm Framework with different topologies. We will also run these topologies on a testing set and compare these results.

# Decomposing queries

The first part of our project consisted of decomposing the SPARQL queries on the RDF data. An RDF triple consists of a subject, a predicate and an object. The query decomposition means turning a selection query in SPARQL into Java code executed on RDF triples, note that we only focus on selection queries, in other words: we just look at the triples, we don’t change them.

We handle this tuple by tuple, so we loop through all the tuples and keep the ones that satisfy the query. We have 3 categories of queries:

* One variable joins: there’s only one input string, this string has to match the subject of the triples, the object and predicate can be anything.
* Two variable joins: There are 2 input strings, one has to match with the subject and one has to match with the object, if both of these match, the triple is passed on.
* Multi variable joins: There are 3 input strings. These strings are compared with the all 3 variables in the triple, if 2 of the input variables match with 2 of the variables in the triple, the triple is passed on.

RDF triples that match the query go on for further analysis, all the other triples are dropped and we no longer look at them.

# Spouts

In the RDF framework, we can create several spouts. In our program, a spout fetches the data and passes it on to a bolt, which we’ll discuss after this section. In our topologies, we choose to use only a single tuple every time because we have only one data source.

The spout calls a function nextTuple() every 100 milliseconds. This functions makes sure a new line from the input file get read, takes the data from this line and transforms it into a rdf triple which get’s passed on to the collector that passes it on to the bolts.

For more information on the architecture, we refer to the documentation on apache storm which can be found at [1].

# Bolts

Every time the spout emits a triple, it get’s passed on to the first bolt which is the next class that handles the triples. What the bolt does depends on the topology, we’ll discuss the different topologies in the next section.

We can have multiple bolts running in parallel, in most of our cases we have 3 bolts running in parallel. Bolts also handle the data and then pass them on. After that the outputted data is either the end result or is handled by another bolt.

# Topologies

A topology manages all the spouts and bolts. We have a number of topologies:

* RDFTopology
* IVJTopology
* TopologyCountBase
* TopologyGrid
* TopologyTimeBase
* TopologyWithThreeBF

TODO

# Testing scenarios

Several queries etc.

TODO

# Testing environment

Grid 5000 network

TODO

# Benchmark results

Results from the benchmarks

TODO

# Conclusion

TODO

# References

1. <http://storm.apache.org/releases/2.0.0-SNAPSHOT/index.html>
2. Todo
3. Todo