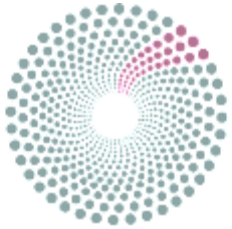

BIG DATA EUROPE

Empowering Communities
with Data Technologies

INTEGRATING SEMAGROW AND BIG DATA STORES

Antonis Troumpoukis
NCSR “Demokritos”

24-25 May 2016

UoA Technical Workshop

Motivation

- Semagrow is a SPARQL endpoint federation engine
- Offers the basis for big data integration beyond triple stores
- RTD work during FP7 ICT SemaGrow:
  - Modular architecture allows for different *executors* to connect to different stores
  - Extremely efficient, minimal overheads over the federated stores' latency and throughput
  - Optimizes execution plan to account for different stores' characteristics
  - Applies vocabulary transformations to integrate data on the fly

Work in BDE/WP4

- Develop executors for BDE storage solutions
  - Started with Cassandra
- A new dimension of heterogeneity to handle:
  - Different QL syntax
  - Weaker QL than SPARQL, so Semagrow must often carry out substantial computation to fill the gap
- A new opportunity to make Semagrow more efficient
  - Distribute the execution of the Semagrow engine itself

Cassandra Data model and CQL

- Hybrid between key-value pair and tabular database
- 3 types of columns w.r.t. primary key
  - *Partition columns* (data distribution across the nodes)
  - *Clustering columns* (efficient retrieval of data belonging to the same partition)
  - *Regular columns* (the remaining columns)
- CQL – Cassandra Query Language
  - SQL-like syntax
  - No JOIN, UNION, ORDER BY operators
  - WHERE clause:
    - only AND operator between *column restrictions*
    - some of the operators IN,=,<,<=,>=,>, based on the column type.

Example CQL queries

author (partition)	year (clustering)	title (regular)	rating (regular)	publisher (regular)
George R.R. Martin	1996	A Game of Thrones	5	Bantam Spectra

- `SELECT author, title FROM books WHERE author='George R.R. Martin';`
- `SELECT title, year FROM books WHERE year=1996 ALLOW FILTERING;`
  - year is a clustering column, partition columns are not restricted => keyword ALLOW FILTERING is needed
- `SELECT author, title FROM books WHERE title='A Game of Thrones';`
  - Not a valid CQL query, since title is a regular column
  - User has to issue a query that returns all books and filter the result set

Cassandra Rows to RDF Tuples

author (partition)	year (clustering)	title (regular)	rating (regular)	publisher (regular)
George R.R. Martin	1996	A Game of Thrones	5	Bantam Spectra

- The above row is mapped to the following set of tuples:
  - `_:node1aj25hg65x22 <http://example.org/books#author> “George R.R. Martin”.`
  - `_:node1aj25hg65x22 <http://example.org/books#year> “1996”^^xsd:int.`
  - `_:node1aj25hg65x22 <http://example.org/books#title> “A Game of Thrones”.`
  - `_:node1aj25hg65x22 <http://example.org/books#rating> “5”^^xsd:int.`
  - `_:node1aj25hg65x22 <http://example.org/books#publisher> “Bantam Spectra”.`

Accessing Cassandra via Semagrow (1/3)

- Example query:

```
SELECT ?title ?year WHERE {
        ?s <http://example.org/books#author> “George R.R. Martin”.
        ?s <http://example.org/books#year> ?year .
        ?s <http://example.org/books#title> ?title }
```

- It is transformed by the Query Executor of the Cassandra Connector into the equivalent CQL query

```
SELECT author, year, title WHERE author=‘George R.R. Martin’;
```

- This query is a valid CQL query

Accessing Cassandra via Semagrow (2/3)

- Semagrow can perform queries that cannot run directly on Cassandra.
  - SELECT ?author WHERE {
    ?s <http://example.org/books#author> ?author .
    ?s <http://example.org/books#title> “A Game of Thrones” }
- Cassandra Connector assists the Decomposition process by consulting the Cassandra schema and analyzing the initial query.
- This query is transformed as follows:
  - SELECT ?author WHERE {
    { ?s <http://example.org/books#author> ?author .
    ?s <http://example.org/books#title> ?temp } @Cassandra
    FILTER (?temp = “A Game of Thrones”) }

Accessing Cassandra via Semagrow (3/3)

- Semagrow can join results coming from SPARQL endpoints and a Cassandra store.
- Example: all authors liked by Joe are contained in a triple store.
  - SELECT ?title WHERE {
    <http://example.org/people/joe> <http://example.org/likes> ?author .
    ?s <http://example.org/books#author> ?author .
    ?s <http://example.org/books#title> ?title }
- The query is transformed as follows:
  - SELECT ?title WHERE {
    { <http://example.org/people/joe> <http://example.org/likes> ?author } @4s
    ?s <http://example.org/books#author> ?author .
    ?s <http://example.org/books#title> ?title } @Cassandra }

Tests

- We performed two simple tests against a Cassandra Store that contains NetCDF file metadata.
- We compare it with a custom-tailored Java code that performs the same operation.
  - Test 1.
    - **Federates:** Cassandra.
    - **Query:** a simple term search on the attributes of the NetCDF headers.
  - Test 2.
    - **Federates:** 4store/Cassandra.
    - **Query:** Returns all dataset names that contain dimension-names that are retrieved from the 4store endpoint.

Term	#Results	SemaGrow	Java
time	16	18150	14478
whoknows	115	14026	13818
yield	2	13975	13858
Time	120	14102	13819

	#Results	SemaGrow	Java
Run 1 (cold)	7	6498	2722
Run 2	7	2434	2653
Run 3	7	2446	2630

Conclusions

- Current state
  - Cassandra connector developed and tested for triple store/Cassandra federations
  - Hot-runs are time-equivalent to custom-tailored Java code
  - No show-stopper in our TODO list, the essentials are there and pilots can already deploy triple store/Cassandra federations
- Next steps
  - Optimizations
  - DISCUSS with JJ: Semagrow over Lucene/Solr