DIONYSUS:
Towards Query-aware Distributed Processing of RDF Graph Streams

Syed Gillani, Gauthier Picard, Frederique Laforest

Laboratoire Hubert Curien & Institute Mines St-Etienne, France

GraphQ 2016
[Outline]

• Stream processing in general
• Semantic-enabled stream processing (RDF stream processing)
• Issues and challenges for RDF stream processing
• Expectations from DIONYSUS
• Functional Layers of DIONYSUS
[The Data Deluge]

• More than 3000 Exabytes (billions GBs) created in 2015 alone
  - Increased from 150 Exabytes in 2005

• Many new sources of data become available
  - Sensors, mobile devices
  - Web feeds, social networks
  - Surveillance video and audio
  - Knowledge Bases
  - ....................

• **How can we make sense of all data**
  - Most of the data is not interesting
  - New data supersedes old data
  - Challenge is not only storage but processing
Stream Processing to the Rescue!

- **Process data streams on the fly without storage**

  - Stream data rates can be high
    - Volume, type, frequency can vary
    - High resource requirement for processing

  - Processing streams have real-time requirement
    - Latency of data processing matters
    - Limited amount of available memory
    - MUST be able to react to the events as they occur (Complex Event Processing)

- **Use cases**: Power management in Smart Grid, Traffic management, Social network analysis, fraud detection etc.,.
Stream Processing is it enough?

- Heterogeneity of sources, multiple Schemas, materialisation of implicit Knowledge
[RDF Stream Processing]

• Utilising Semantic Web Technologies
  - Facilitating data integration by using machine processable descriptions to reconcile heterogeneities (e.g., Semantic Sensor Web)
  - Handle diversity with **schema-less** Model
  - Graph-structured data model (RDF)

• Stream Reasoning
  - Performing materialisation of implicit knowledge (e.g., inference using ontologies)
  - Utilising static knowledge-bases (Linked-data Cloud) to extract contextual knowledge
[RDF Stream Processing]

Ontologies

Query

Query Instances

Ontologies

Knowledge-Bases (e.g., DBpedia)
RDF Stream Processing

- RDF Stream Processing is expensive
  - Graph Pattern Matching, an NP-complete problem

An RDF Graph Stream

\[ [\langle S_1, P_1, O_1 \rangle \ldots \langle S_i, P_i, O_i \rangle ] \tau_1 \]

\[ [\langle S_1, P_1, O_1 \rangle \ldots \langle S_i, P_i, O_i \rangle ] \tau_m \]

Query Pattern

\[ \langle ?x a b \rangle \land \langle ?x c ?y \rangle \land \langle ?y m n \rangle \]

Match!!
[RDF Stream Processing]

• Existing Solutions
  - Rely-on centralised processing
  - Triple stream model, which ignores the graph nature of RDF
  - Extended for DSMS models, black-box approach
  - Re-evaluation based query processing
  - No support to gather archives of streams

• Vision/Expectations (DIONYUSIS)
  - Scale-out solution
  - Handle the distributed nature of stream sources
  - Incremental indexing and incremental query processing
  - State-full operators to enable Semantic Complex Event Processing
Data Distribution

• Can we learn anything from static solutions
  - Hash-based clustering [Zeng et.al 2013], graph structure of RDF data?
  - Semantic Hash-based clustering [Lee et.al 2013], multivalued predicates?

• Distribution for RDF data is not trivial and requires extensive preprocessing

• Reverse paradigm for stream data distribution
  - Data is not known in advance for distribution analysis
  - New sources are added dynamically and old sources provide data at variable velocities
  - Variable query loads
• Relying on ontologies and use cases
  - Given a set of ontologies $O$ defined on a set of streams $S$, produce a set of common basic graph patterns (CBGPs)
  - Each CBGP contains information about a coherent subtopic within an ontology
CBGP Stores and Query Processing

- CBGP Stores
  - Aggregating common linked-concepts within a single CBGP store
  - Each CBGP store has customised optimisation, light-weight adaptive indexing
  - Reducing the network traffic and load at federation level
  - Acts as data filter, only relevant data is stored from a set of sources
  - Divided into three flavours: static, alive and deceased

- A collection of CBGP stores is abstracted under an Island, each island is assigned to a set of **query-conductors**
Query Conductors

- Determines the type of the registered query graphs: analytical query over archived data, streaming query, sequence-based query (CEP)

- Divide the query graph into a set of subgraph queries, share loads, and orchestrates the query execution

Archipelago of CBGP-Stores

Static Island of CBGP-Stores
Alive Island CBGP-Stores
Deceased island of CBGP-Stores

Data Stream Sources
[Functional Layers of DIONYUSIS]

1. Apps
2. Clients
3. Visualisation
4. Exact Query Graphs (EQGs)
5. Query Conductors
6. CEP Optimiser
7. Stream Optimiser
8. Analytical Optimiser
9. Archipelago of CBGP-Stores
   - Static Island of CBGP-Stores
   - Alive Island CBGP-Stores
   - Deceased Island CBGP-Stores
10. Data Stream Sources
Analytical Queries: Traditionally

- Compute each pattern against all the available data stores and the results are joined at the server
- Evaluating each pattern in nested-loop-join fashion: substituting the results from one pattern to another

**QUERY 1. Analytical query for Smart Grid use case**

```
SELECT ?area, ?house, AVG(?power)  (iii)

WHERE
{
?house :location ?l.
?house :powerSource ?source.  (i)
?source :value ?power.

?area :name ?areaName.
}
GROUP BY (?area)  (ii)
```

Deceased CBGP-store-1

Static CBGP-store-2

Query Conductor
• Streaming Queries
  - Query is distributed into set of subquery graphs each is hosted by a CBGP store
  - The matches are computed locally and window operators is executed at query conductor level

**QUERY 2. Streaming query for Smart-Grid use case**

```
WINDOW 2 HOURS
WHERE

{ STREAM <http://example.org/powersource> [Range 2s]
}

STREAM <http://example.org/weathersource> [Range 2s]
```
Query Optimisations for DIONYUSIS

• Sequence-based Query
  - Query is distributed into set of subquery graphs each is hosted by a CBGP store
  - The matches are computed locally and window operators is executed at query conductor level

```
QUERY 3. Sequence-based query for Smart Grid use case
SELECT ?house, ?1, ?power
WITHIN 24 hours
PARTITION BY (?house)
FROM STREAM S1 <http://example.org/powersource>
FROM STREAM S2 <http://example.org/weathersource>
WHERE
{ SEQ (A, B)
  A ON S1
  { ?house :location ?l.
  ?source :value ?power.
  FILTER (?power > 50)
  }
  B ON S2
  FILTER (?temp > 20 && ?wspeed > 10)
  }
}
```

SEQUENCE

Alive CBGP-store-1
Alive CBGP-store-2

Query Conductor
**Conclusion**

- **Addressing the requirements of RDF graph streams:**
  - Scalability, state management, distribution of data sources

- **One query interface to support:**
  - Continuous and distributed streaming queries
  - Queries over archived streams
  - Temporal sequential queries

- **Future Work:**
  - Integration of separate layers of the system.
  - Benchmarking distributed RDF graph streams
[Questions?]