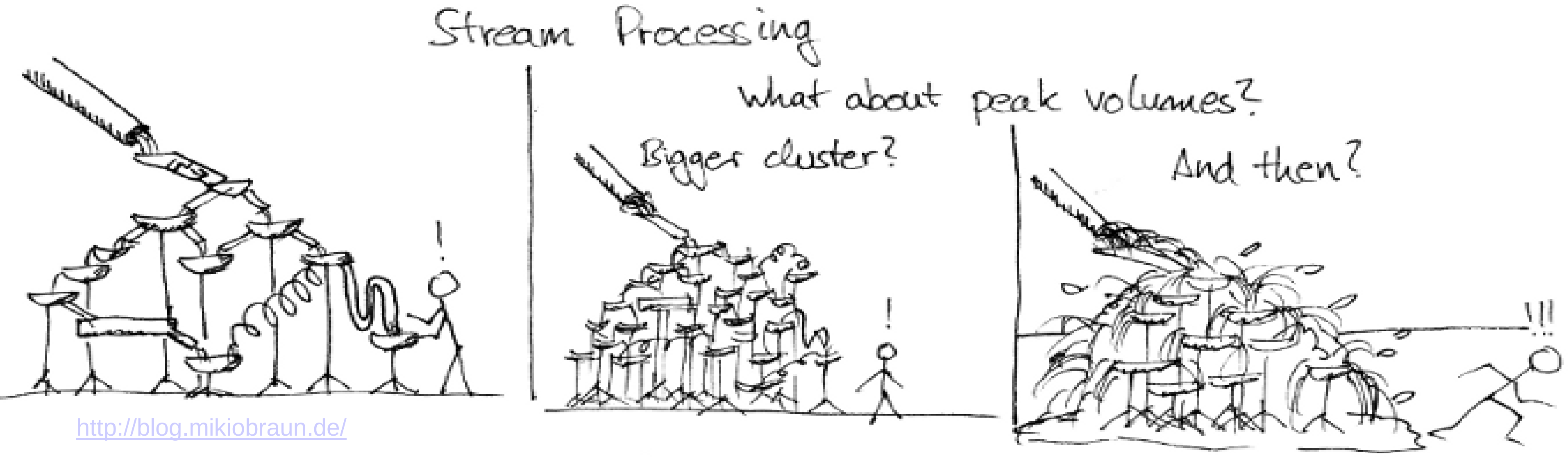
Towards efficient processing of RDF data streams

Alejandro Llaves
Javier D. Fernández
Oscar Corcho

Ontology Engineering Group
Universidad Politécnica de Madrid
Madrid, Spain
allaves@fi.upm.es

UPM, October 16th 2014



http://blog.mikiobraun.de/

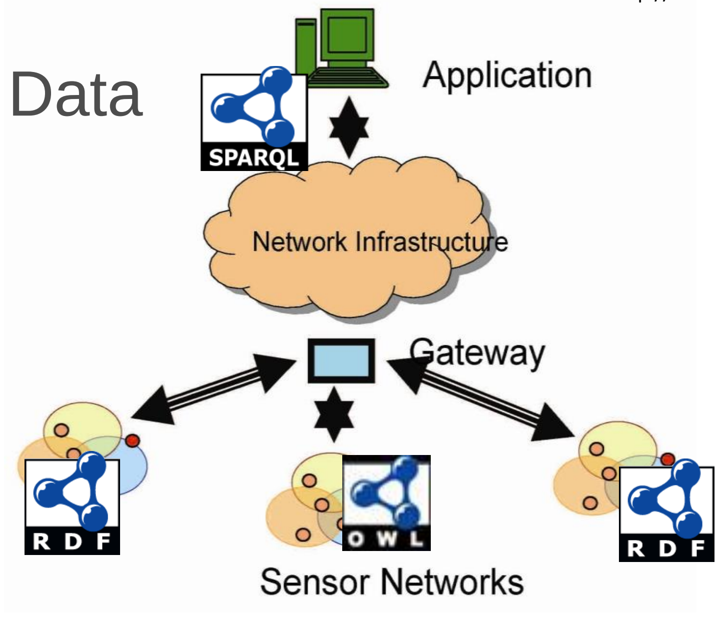
Outline

- Introduction
- Background: Storm and Lambda Architecture
- Approach
  - Architecture design
  - Storm-based operators for querying RDF streams
  - RDF stream compression
- Conclusions & future work
- SIMON Project

Introduction

Source: http://webnotations.com

- Origins of Linked Stream Data

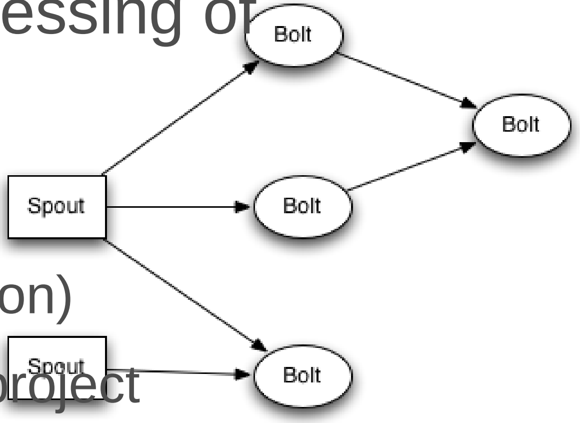


- Extracting information from data stream is complex: heterogeneity, rate of generation, volume, provenance,…
- Challenges
  - C1. Scalable processing of user queries over RDF streams
  - C2. Continuous transmission of data increases latency

Background

Storm - http://storm.incubator.apache.org/

- Distributed system for real-time processing of streams
- Why Storm?
  - Simple processing model (parallelization)
  - Open source community backing the project
  - Used by relevant companies, e.g. Twitter.



Lambda Architecture

- Batch layer: stores ALL the incoming data in an immutable master dataset and pre-computes batch views on historic data.
- Serving layer: indexes views on the master dataset.

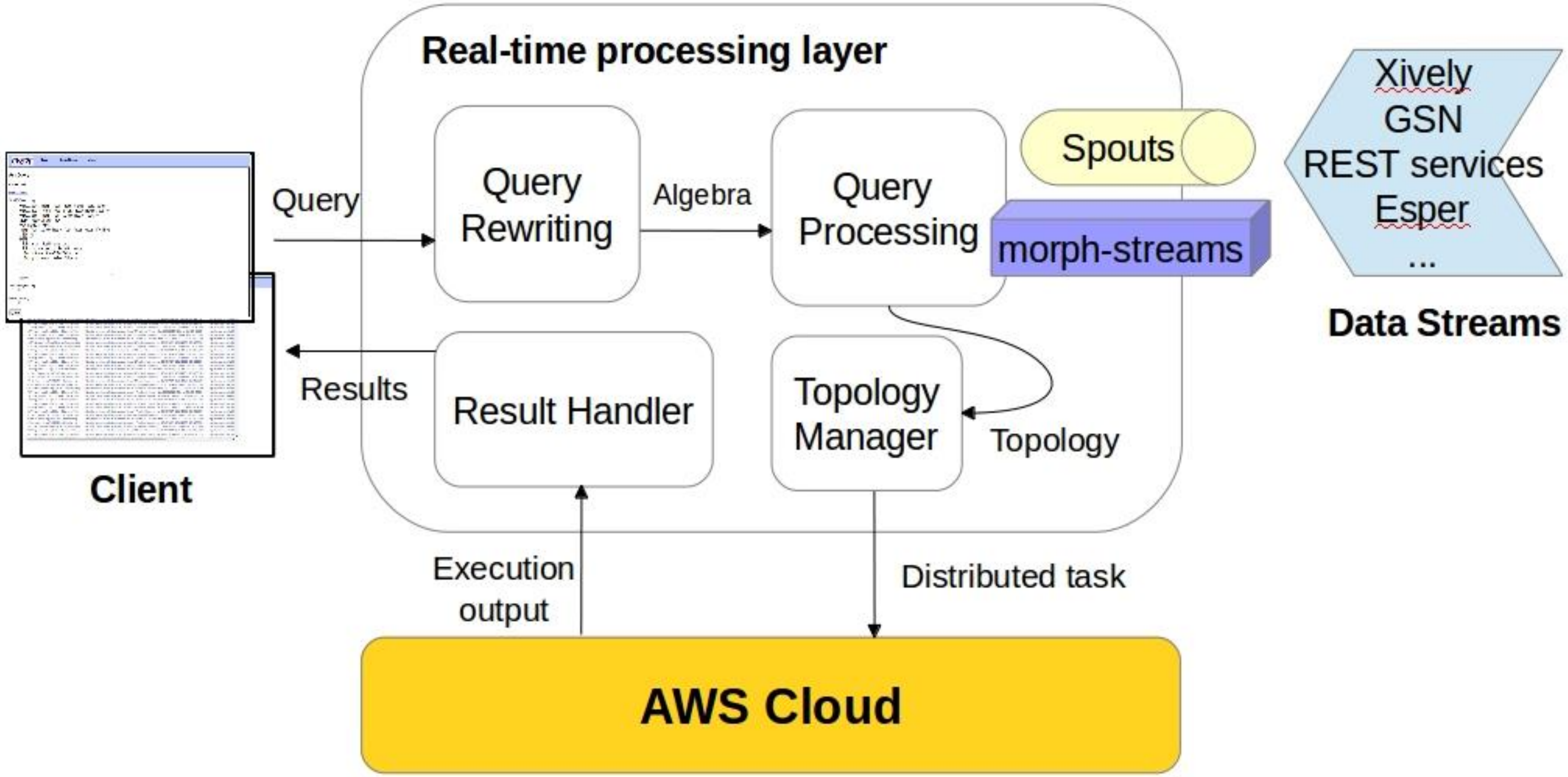
Approach: morph-streams++

Goal: to develop a stream processing engine capable of adapting to changing conditions, such as changing rates of input data, failure of processing nodes, or distribution of workload, while serving complex continuous queries.

Methodology

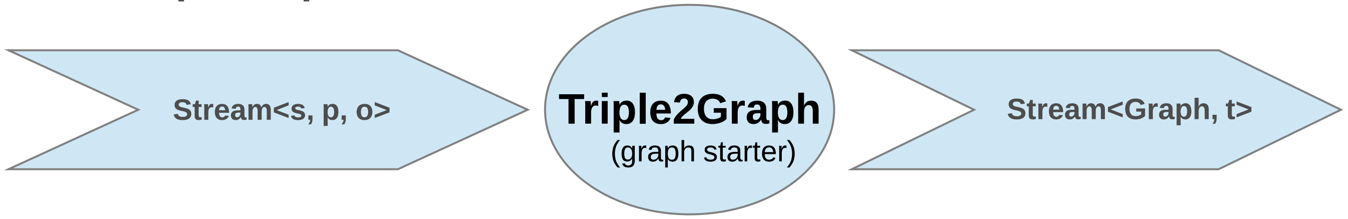
- State of the art of (RDF) stream processing
- Evaluate how to parallelize SPARQLStream queries
- Implementation of RDF query operators
- Optimize parallelization for common queries
- Design self-adaptive strategies that allow the engine to react in front of changes

Architecture design

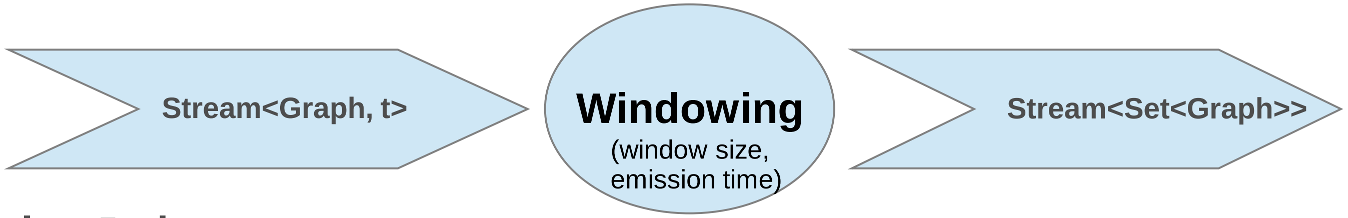


Storm-based operators for querying RDF streams

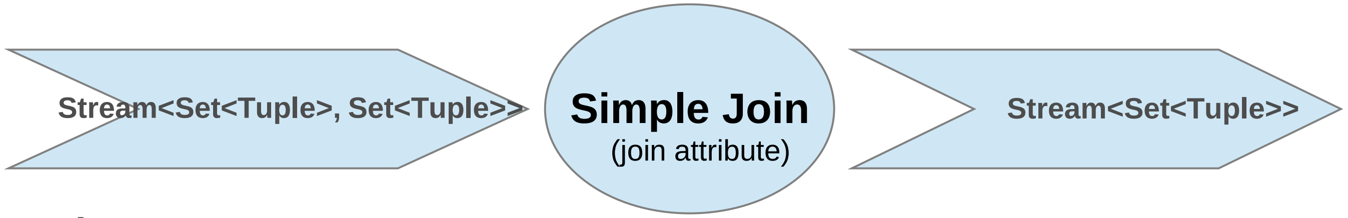
- Triple2Graph operator



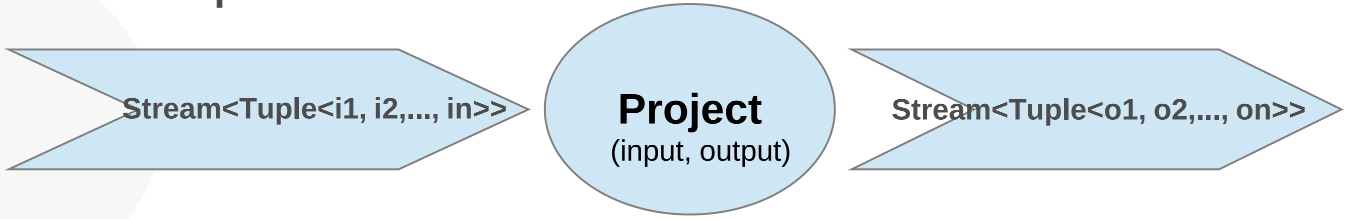
- Time Window operator



- Simple Join operator



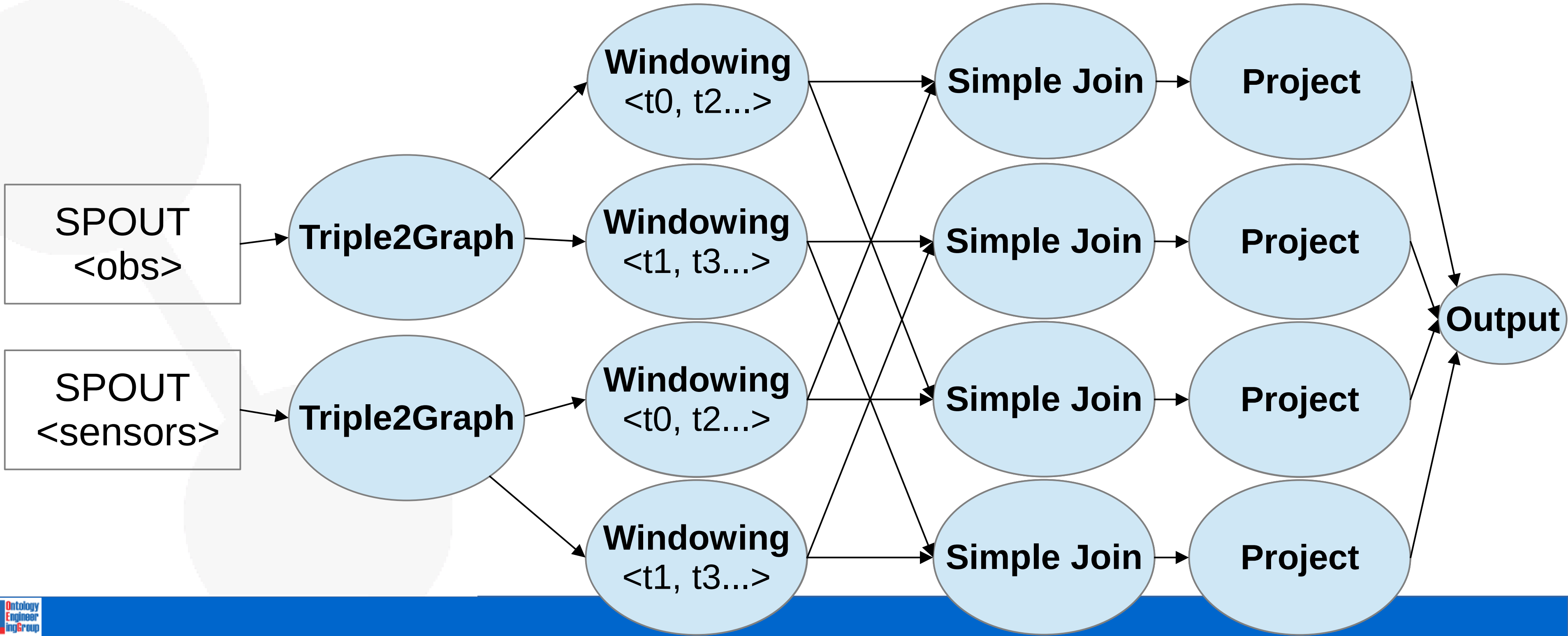
- Projection operator



Storm-based operators for querying RDF streams

Storm topology example

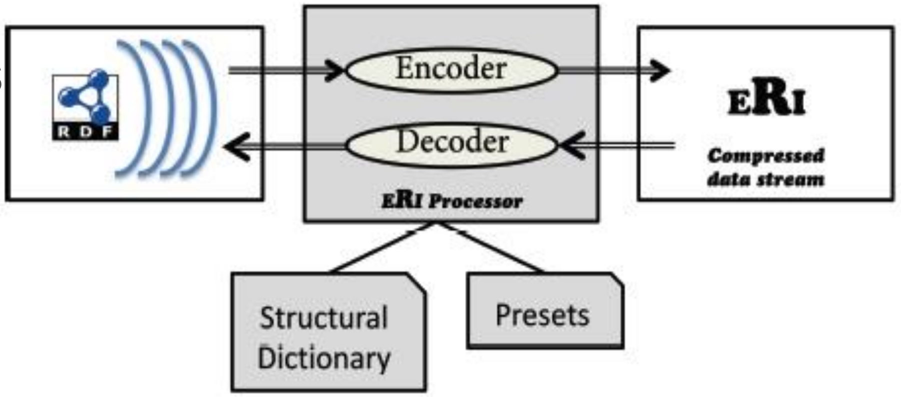
```
SELECT ?obs.value ?sensors.location
FROM NAMED STREAM <obs> [60 SEC TO NOW]
FROM NAMED STREAM <sensors> [60 SEC TO NOW]
WHERE obs.sensorId = SENSORS.id ;
```

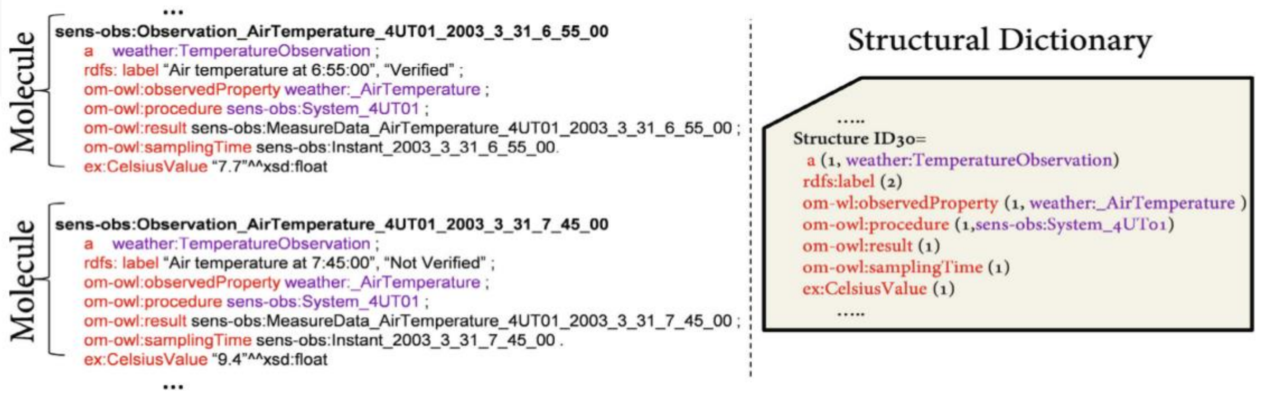


RDF stream compression (1/2)

Efficient RDF Interchange (ERI) format

- Based on Efficient XML Interchange (EXI)
- Main assumption: RDF streams have regular structure and are redundant
- Information encoded at 2 levels
  - Structural dictionary
  - Presets (values)
- Example: SSN observations




Molecule
Molecule
Structural Dictionary

```
...
sens-obs:Observation_AirTemperature_4UT01_2003_3_31_6_55_00
    a    weather:TemperatureObservation ;
    rdfs: label "Air temperature at 6:55:00", "Verified" ;
    om-owl:observedProperty weather:_AirTemperature ;
    om-owl:procedure sens-obs:System_4UT01 ;
    om-owl:result sens-obs:MeasureData_AirTemperature_4UT01_2003_3_31_6_55_00 ;
    om-owl:samplingTime sens-obs:Instant_2003_3_31_6_55_00.
    ex:CelsiusValue "7.7"^^xsd:float

sens-obs:Observation_AirTemperature_4UT01_2003_3_31_7_45_00
    a    weather:TemperatureObservation ;
    rdfs: label "Air temperature at 7:45:00", "Not Verified" ;
    om-owl:observedProperty weather:_AirTemperature ;
    om-owl:procedure sens-obs:System_4UT01 ;
    om-owl:result sens-obs:MeasureData_AirTemperature_4UT01_2003_3_31_7_45_00 ;
    om-owl:samplingTime sens-obs:Instant_2003_3_31_7_45_00 .
    ex:CelsiusValue "9.4"^^xsd:float
...
```

```
.....
Structure ID30=
  a (1, weather:TemperatureObservation)
 rdfs:label (2)
 om-wl:observedProperty (1, weather:_AirTemperature )
 om-owl:procedure (1,sens-obs:System_4UT01)
 om-owl:result (1)
 om-owl:samplingTime (1)
 ex:CelsiusValue (1)
.....
```

RDF stream compression (2/2)

Evaluation

- *Datasets:* streaming, statistical, and general static.
- Compression ratio, compression time, and parsing throughput (transmission + decompression)
- Comparison to other formats, such as N-Triples, Turtle, RDSZ, HDT, with different configurations of ERI w.r.t. transmitted data block (1K – 4K) and the presence of dictionary.
- *Conclusion:* ERI produces state-of-the-art compression for RDF streams and excels for regularly-structured static RDF datasets. ERI compression ratios remain competitive in general datasets and the time overheads for ERI processing are relatively low.

ISWC 2014
Riva del Garda - Trentino, Italy

Conclusions and future work

Conclusions

- We have addressed challenges C1 (scalability) and C2 (transmission)
  - Catalogue of Storm-based operators to parallelize query processing over RDF streams.
  - New format for RDF stream compression called ERI.
- Challenge C3 (integration) involves storage of historical data and the deployment of batch and serving layers OR the migration to a more general system, e.g. Apache Spark.

Future work

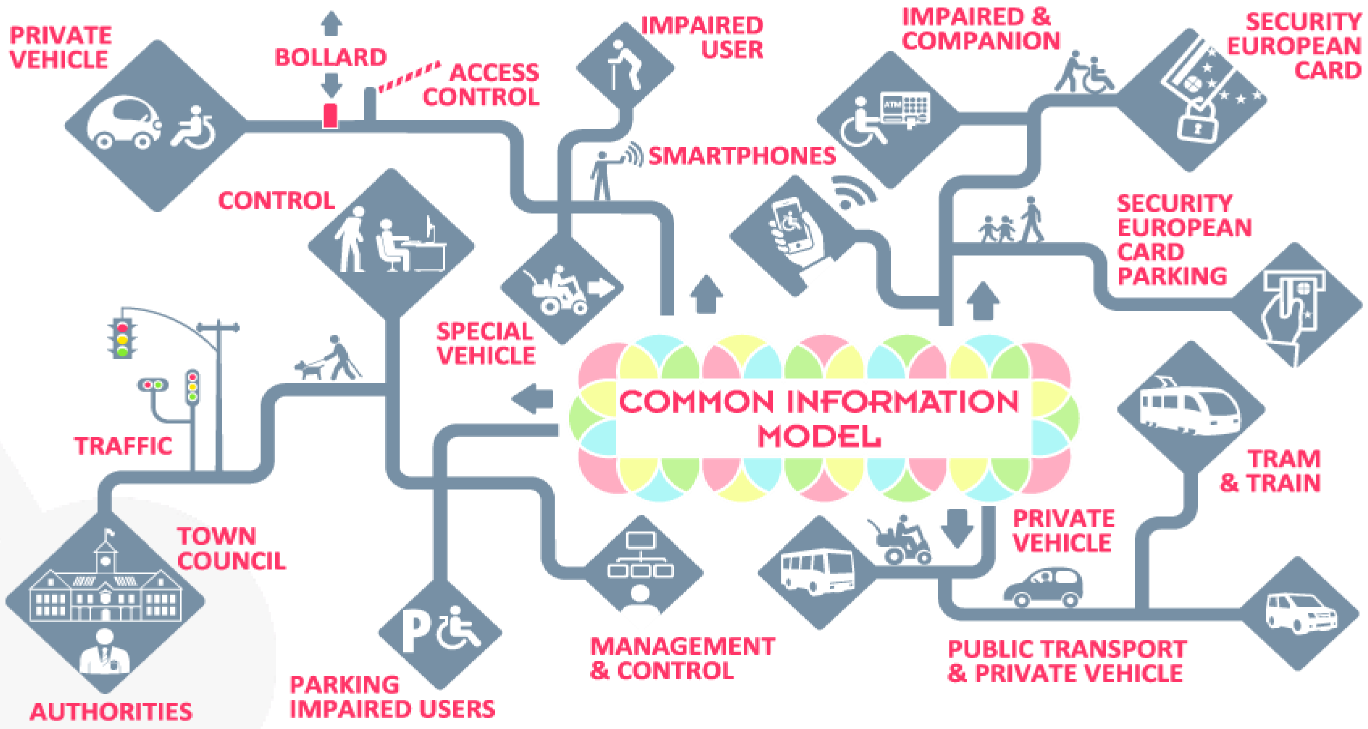
- Finish the implementation of RDF query operators
- Test the parallelization of a set of common queries

SIMON Project (1/2)

http://simon-project.eu/

Time frame: Jan 2014 – Dec 2016

Description: demonstration-oriented project with 3 large scale pilots in Lisbon, Parma, and Madrid aiming to use ICT services to promote the independent living and societal participation of mobility impaired people in the context of public parking areas and multiple transport modes.



SIMON Project (2/2)

SIMON
ASSISTED MOBILITY
FOR OLDER AND IMPAIRED USERS

Goals

- To reduce fraud in the use of the European Disable Badge for public parking areas.
- To improve navigation solutions for elderly and people with disabilities.

Partners: ETRA I+D (Spain), Madrid City Council, Institute of Biomechanics of Valencia, Consorcio Regional de Transportes de Madrid, Locoslab GmbH (Germany), EMEL (Portugal), and Infomobility SpA (Italy).

OEG role

[illegible]

Ontology Engineering Group

Thanks!

The presented research has been supported by an AWS in Education Research Grant award.

Alejandro Llaves
allaves@fi.upm.es