Projet WAVES : Des flux de données brutes et hétérogènes à l'information qualifiée

N° du contrat F1411006 Q
Date de début 2 juin 2014
Durée 36 mois

Livrable D1.2 b

Prototype V0: Configuration and Results

30.03.2016

Status

Niveau dissémination	Publique
Date d'échéance	Mois 20, 30/03/2016
Date de soumission	29/03/2016
Work Package	1
Tâche T	1.2
statut d'approbation	Final
Version	2
Nombre de Pages	13
Nom du fichier	D1.2-waves-tutoriel-prototypeV0

Authors

Organisation	Nom	Contact
ATOS	Niantso Tendry RANDRIAMALALA	niantso-tendry.randriamalala@atos.net
ATOS	Houda KHROUF	houda.khrouf@atos.net

Outline

- System configuration
- Query configuration
- Use-case Scenario
- Scenario 1:
  - Sector by sector comparison
  - Results
- Scenario 2:
  - Threshold-based Reasoning
  - Results

System configuration

- installation.name : To set the global name of the application
- zookeeper.hosts : To specify the connection to Zookeeper, the host and the port
- redis.host : To give Redis server host and port to the program

```
# Waves global configuration parameters
#
installation.name       = first-tests
#
# ZooKeeper configuration (not to use Storm default ZooKeeper server)
# Note: This property is required when using Storm Kafka spout that discovers
#       Kafka servers through their ZooKeeper registration.
zookeeper.hosts         = localhost:2181
#
# Global Redis database
redis.host              = localhost:6379
```

- Windowing rules : To determine temporal sliding window characteristics
  - [streamId].stepRate : the period for execution of the query
  - [streamId]. windowSpan : the size of the temporal window

```
S-3f.stepRate           = 900
S-3f.windowSpan         = 900
```

Query configuration

- The static query is given with sparql1Feed.sparqlQuery
  - sparql1Feed.endpoint : to give the SPARQL endpoint of the triplestore containing data
  - sparql1Feed.type : to figure out query language syntax
  - sparql1Feed.refresh : to fix the period to update static data

```
sparql1Feed.type        = SPARQL
sparql1Feed.endpoint    = http://localhost:8181/openrdf-sesame/repositories/wavesRepo
sparql1Feed.sparqlQuery = \
    PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> \
    PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#> \
    PREFIX cuahsi:<http://his.cuahsi.org/ontology/cuahsi#> \
    PREFIX wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#> \
    PREFIX qudt:<http://data.nasa.gov/qudt/owl/qudt#> \
    CONSTRUCT {  \
        ?sector rdf:type ssn:Platform . \
        ?sector rdfs:label ?labelSector . \
        ?inputFlowSensor ssn:onPlatform ?sector . \
    WHERE { \
        ?sector rdf:type ssn:Platform . \
        ?sector rdfs:label ?labelSector . \
        ?inputFlowSensor ssn:onPlatform ?sector . \
    }
sparql1Feed.refresh     = 120
```

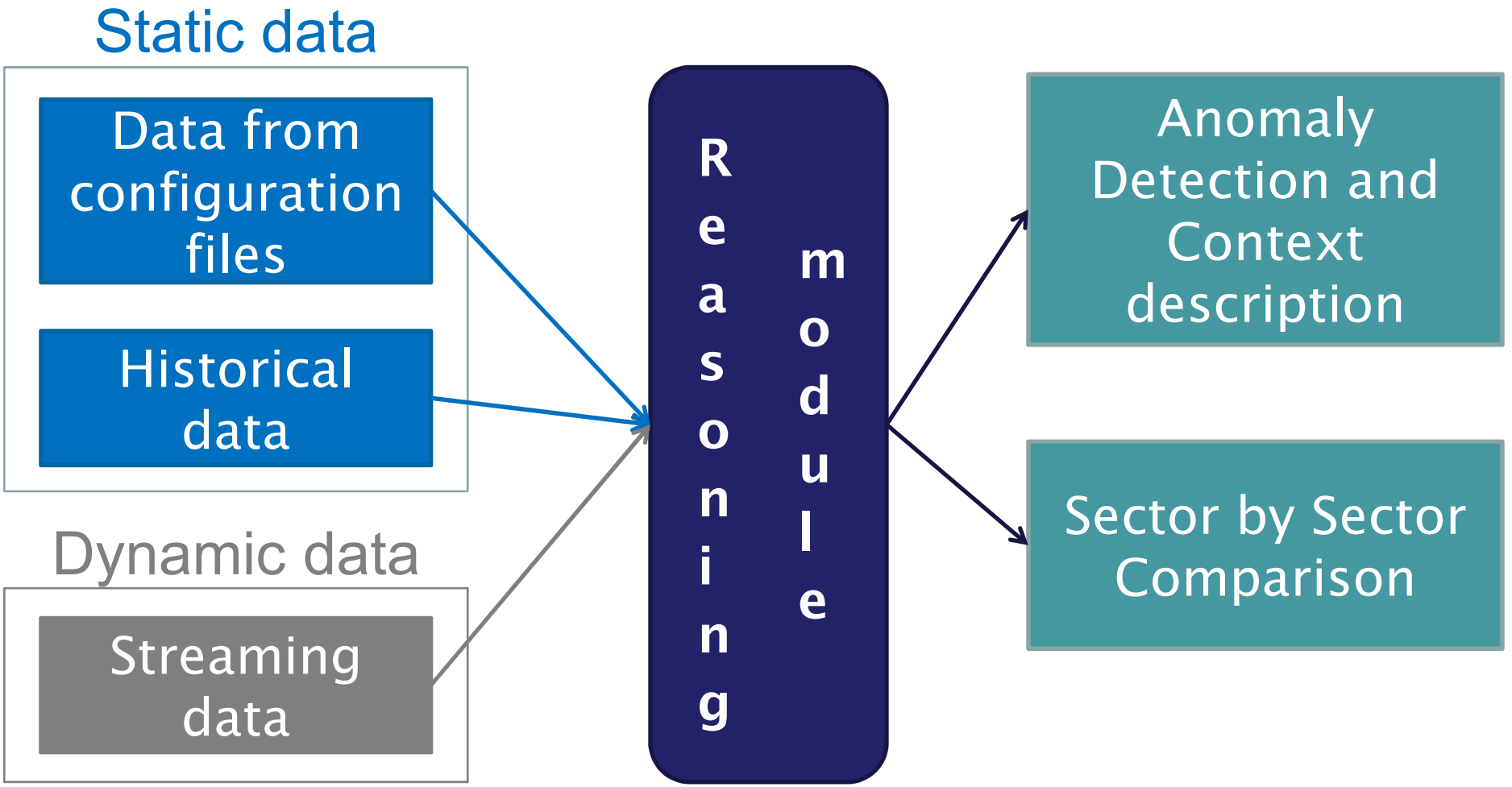
Query configuration

- The dynamic query is defined by [streamId].sparqlQuery

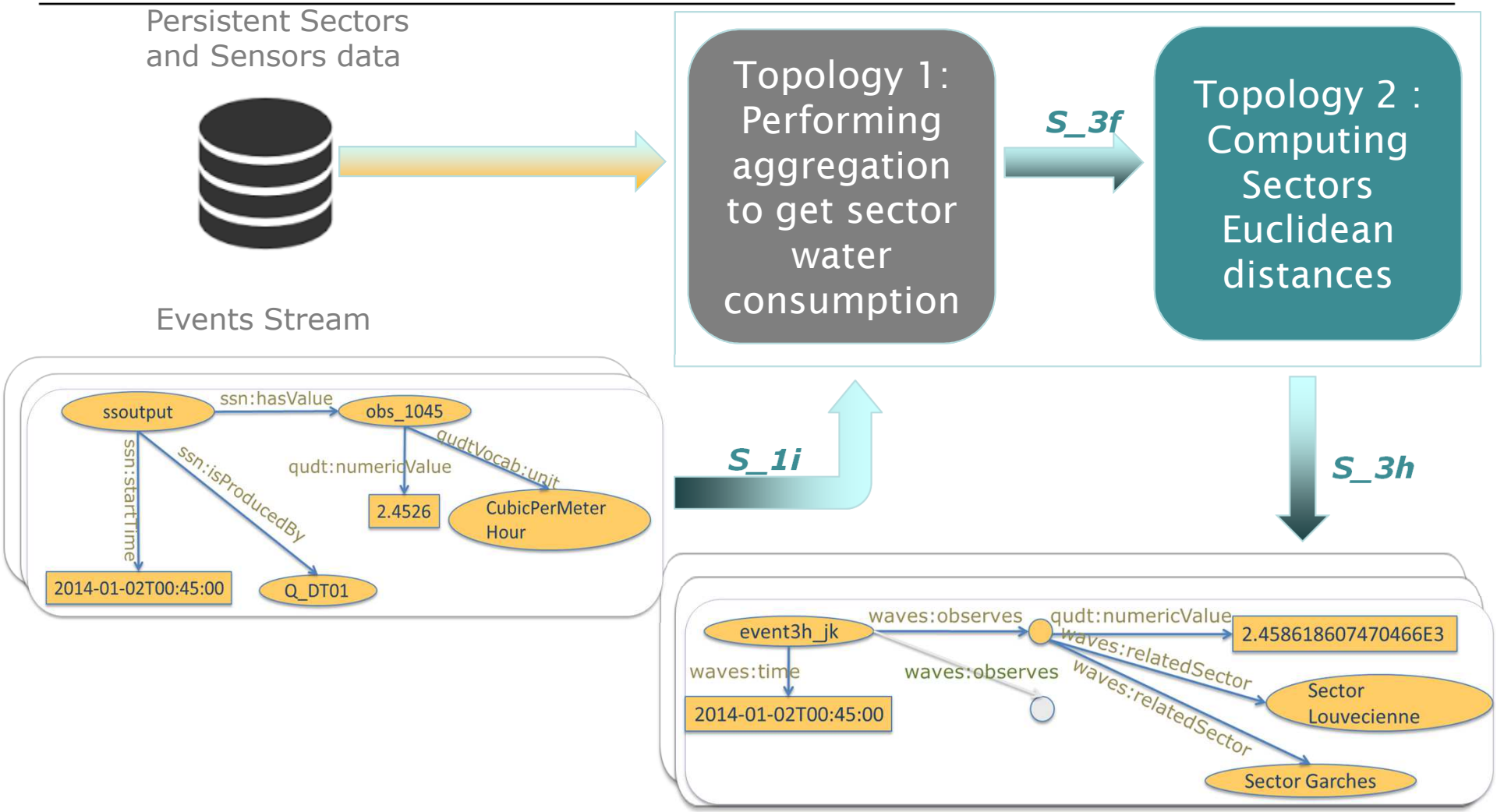
```
S-3f.sparqlQuery        = \
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> \
PREFIX qudt:<http://data.nasa.gov/qudt/owl/qudt#> \
PREFIX waves:<http://waves-rsp.org/resource#> \
PREFIX ssn:<http://purl.oclc.org/NET/ssnx/ssn#> \
    CONSTRUCT { \
            ?sector rdfs:label ?labelSector . \
            ?inputFlowSensor ssn:onPlatform ?sector . \
            ?inputEvent ssn:isProducedBy ?inputFlowSensor . \
            ?inputEvent ssn:hasValue ?inputObservationValue . \
            ?inputEvent ssn:startTime ?time . \
            ?inputObservationValue a ssn:ObservationValue. \
            ?inputObservationValue qudt:numericValue ?inputValue . \
    } \
        WHERE { \
            ?sector rdfs:label ?labelSector . \
            ?inputFlowSensor ssn:onPlatform ?sector . \
            ?inputEvent ssn:isProducedBy ?inputFlowSensor . \
            ?inputEvent ssn:hasValue ?inputObservationValue . \
            ?inputEvent ssn:startTime ?time . \
            ?inputObservationValue a ssn:ObservationValue. \
            ?inputObservationValue qudt:numericValue ?inputValue . }
```

- [QOutId].outputFile : To be specified if we need to write output stream from the module into a file, if not, output will be redirected to kafka.
- [QOutId]. outputFormat: To change the output flow format

Use-case Scenario



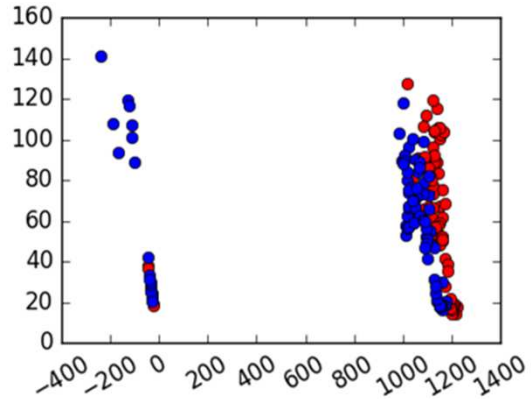
Scenario 1: Sector by Sector Comparison

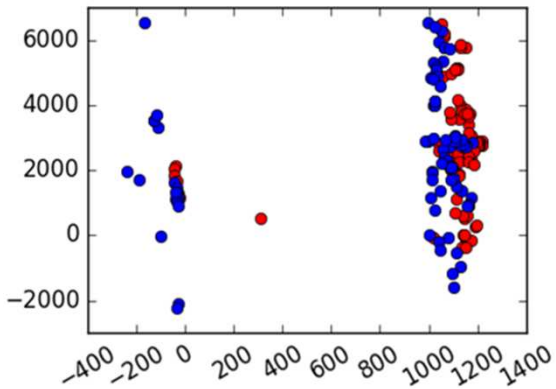


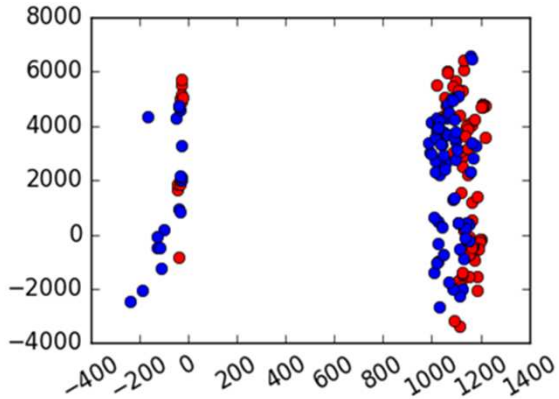
Scenario 1:
Sector by Sector Comparison - Results

- -> ([Consumption **Hubies**],02/01/2014, time) , ([Consumption **Other**],02/01/2014, time)
- -> ([Consumption **Hubies**],03/01/2014, time) , ([Consumption **Other**],03/01/2014, time)

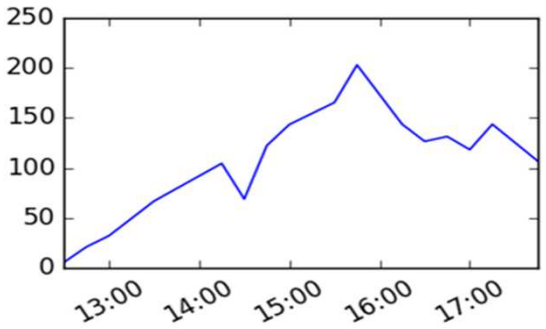
Suresnes	Louveciennes	Ville Nouvelle

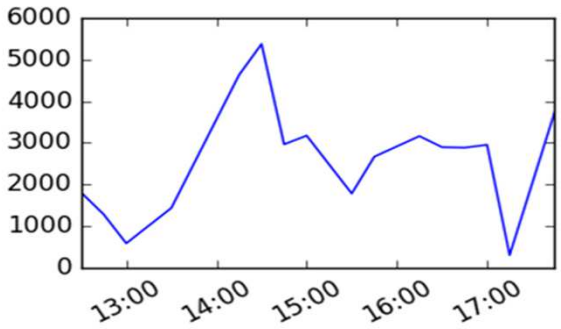


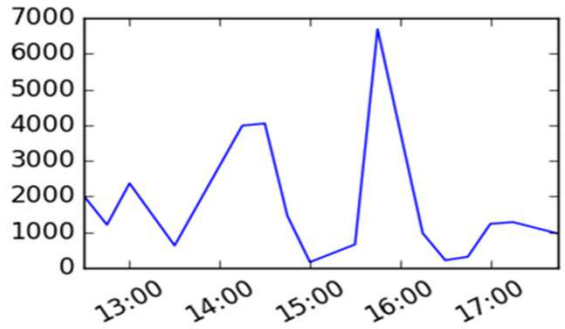




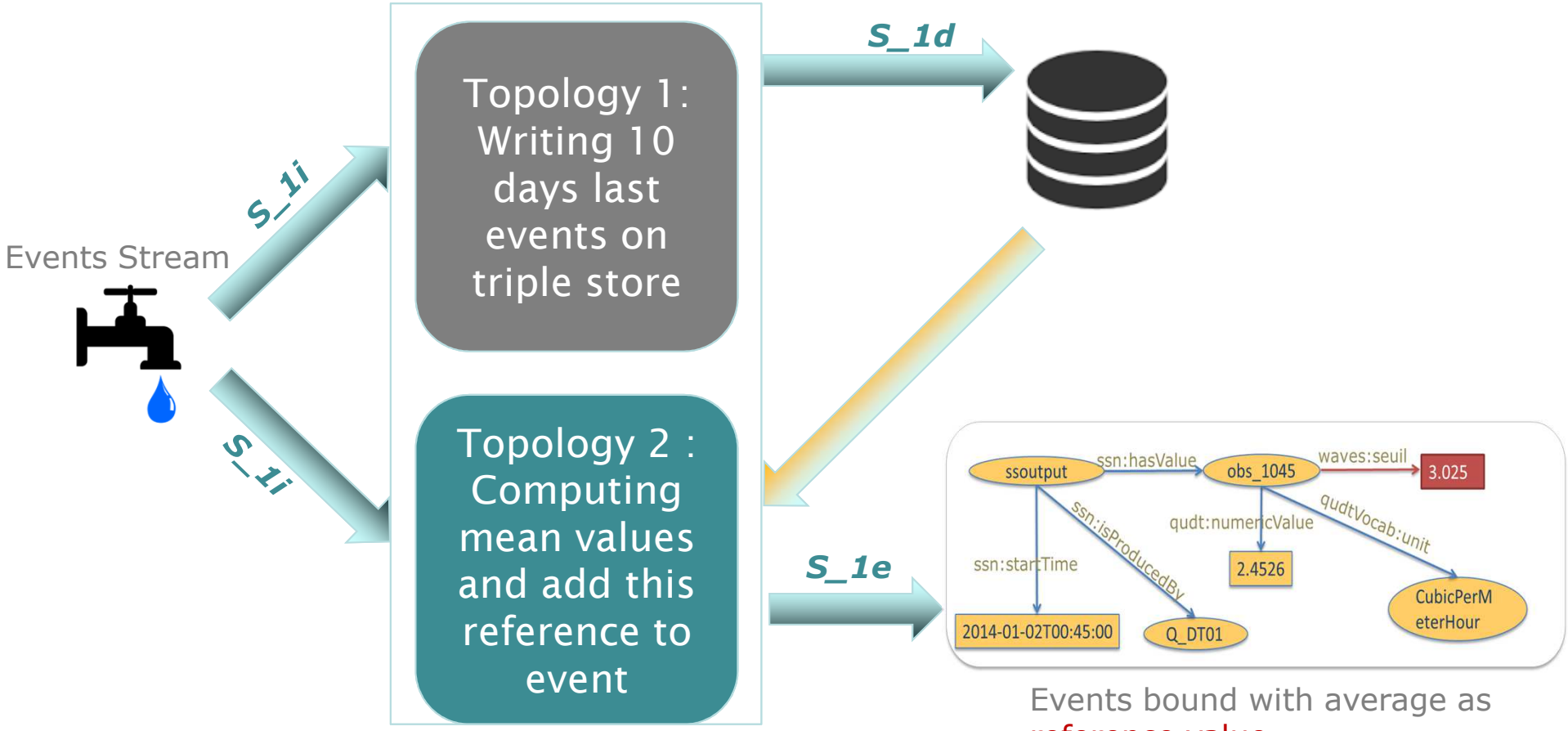
Curve presentation of distances to get time information





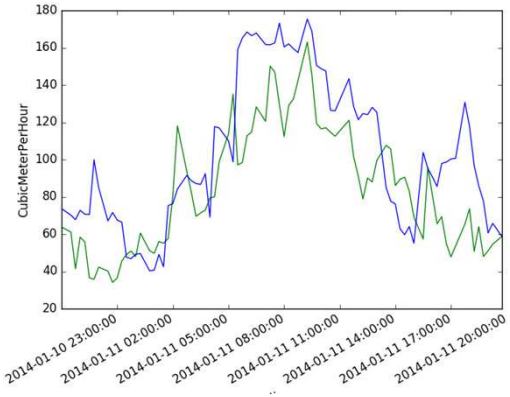
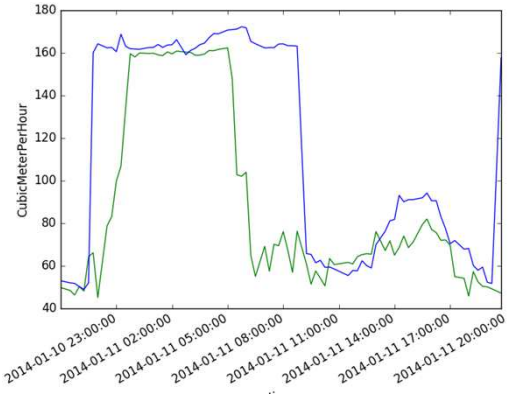
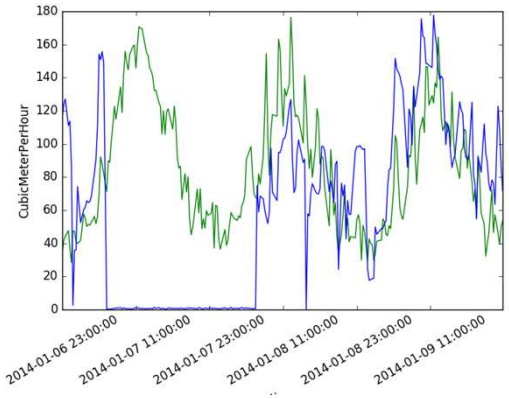
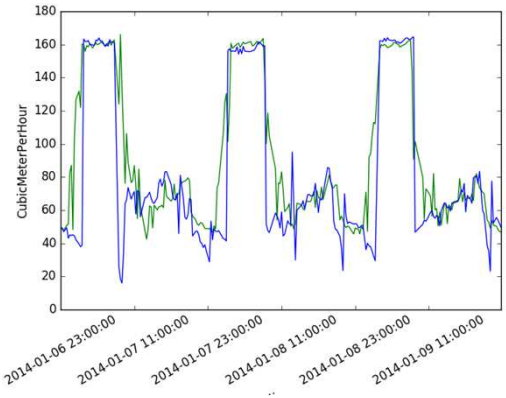


Scenario2: Threshold-based Reasoning



Events bound with average as reference value

Scenario2: Threshold-based Reasoning - Results

	Sensor Q 400N [Sector Louvecienne]	Sensor Q EN01[Sector Ville Nouvelle]
One day observation 11/01/2014		
Three days observation 07/01/2014 to 09/01/2014		

- Average of values measured on 10 past days

- Real time values given by stream

Merci

Atos

30.03.2016