A Compressed, Inference-enabled Scheme for RDF Stream Processing

Jérémy LHEZ, Xiangnan REN,Badre BELABBESS, Olivier CURÉ

LIGM, Marne-la-Vallée, FRANCE

OUTLINE

Agenda of the Presentation

CONTEXT

LITEMAT

PATBIN

CONCLUSION

CONTEXT

Importance of stream processing

DATA STREAMS EVERYWHERE

WAVES
ATOS SE

WAVES PROJECT

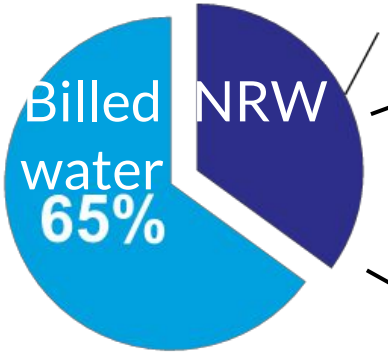
- **Smart water network management**
  - Data streams from sensors
  - Filtering errors in measures
  - Identify sources in external events

- **Main partner: Suez**
  - 650 collaborators in Europe
  - 73 billion $US in R&D

- **French project**
  - http://www.waves-rsp.org/

Why water management ?

Water SUPPLIED to the network – **Water BILLED to *customers* = NON-REVENUE WATER (NRW)**



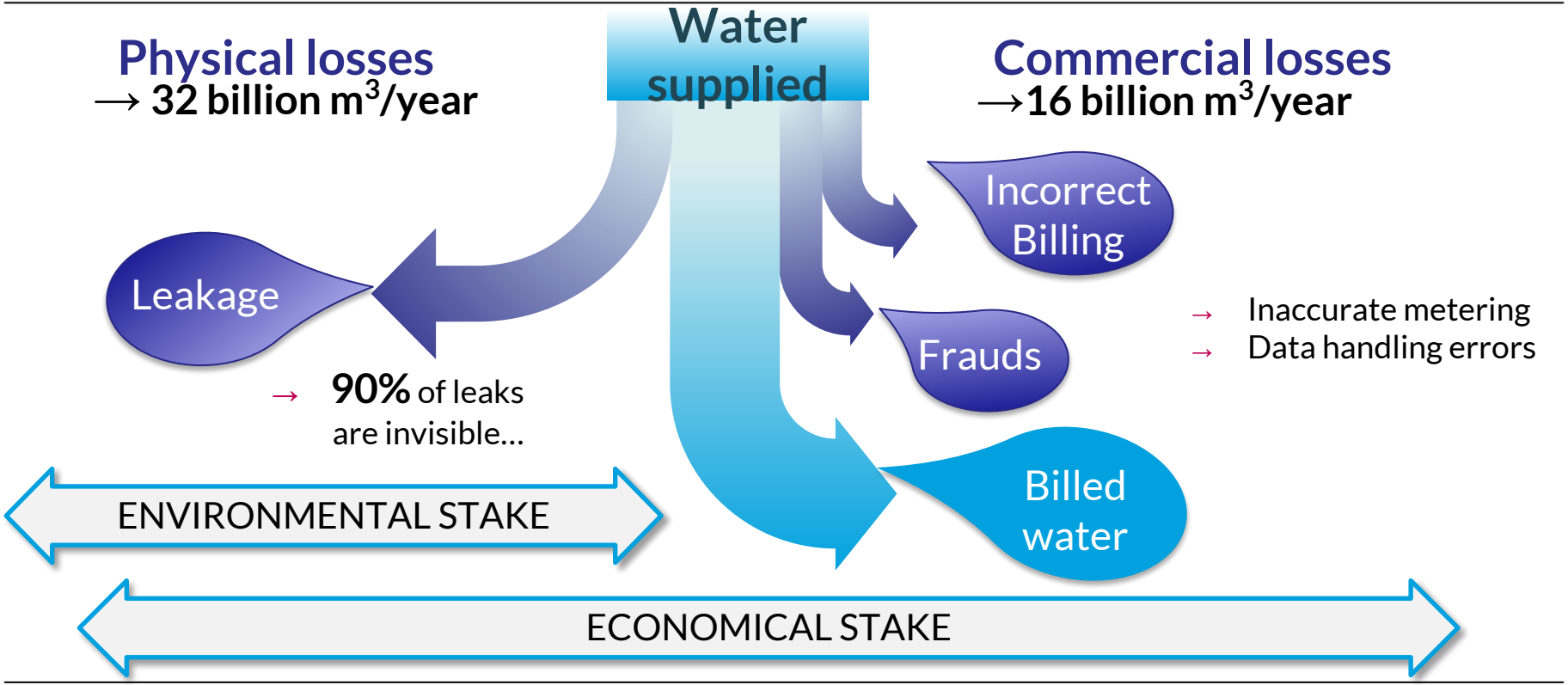
48.6 billion m^3/year

Loss of US$**14 billion**/year

=

x2 the annual domestic water consumption of the USA

Why water management ?

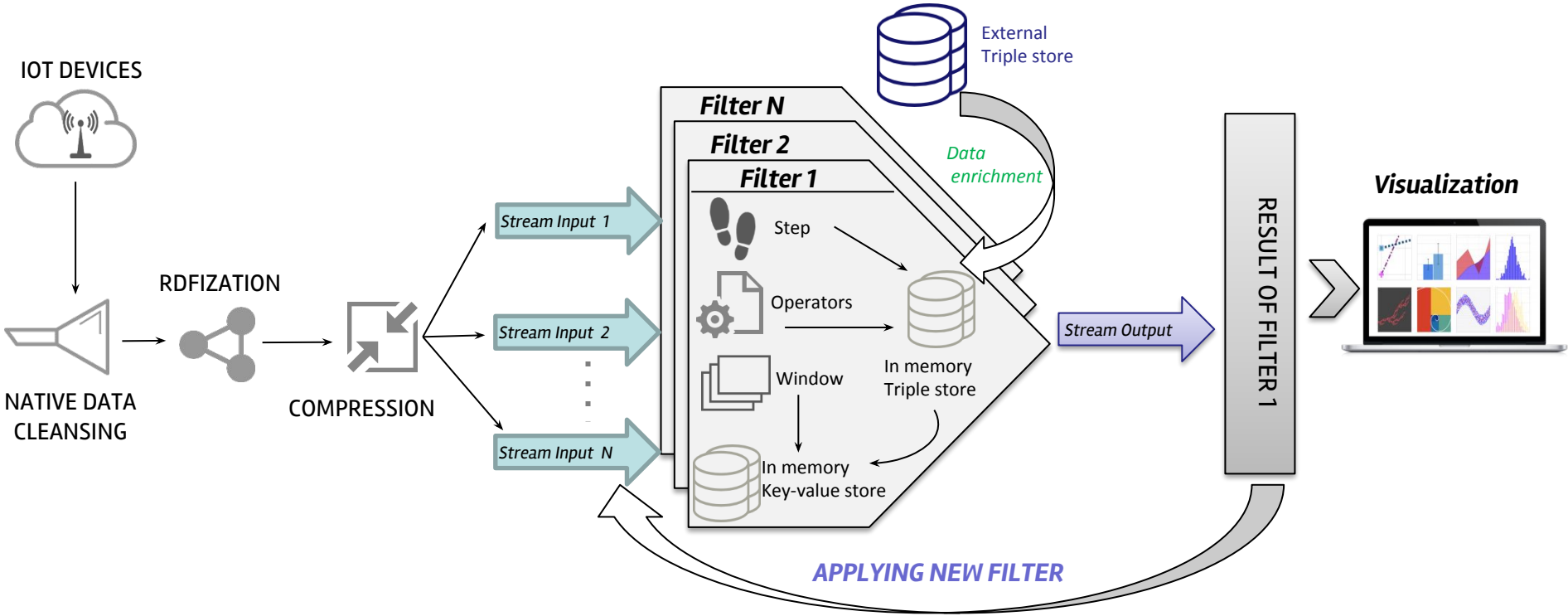


WAVES PROJECT

- **Main aspects:**
  - Real-Time processing
  - RDF data streams/SPARQL queries
  - Reasoning Capabilities/Inferences
- **Objectives:**
  - Robust RSP engine
  - Modularity and flexibility
  - Distribution – Industrial Cluster
- **Other applications:**
  - Banking/payments, climate, energy, power consumption, etc

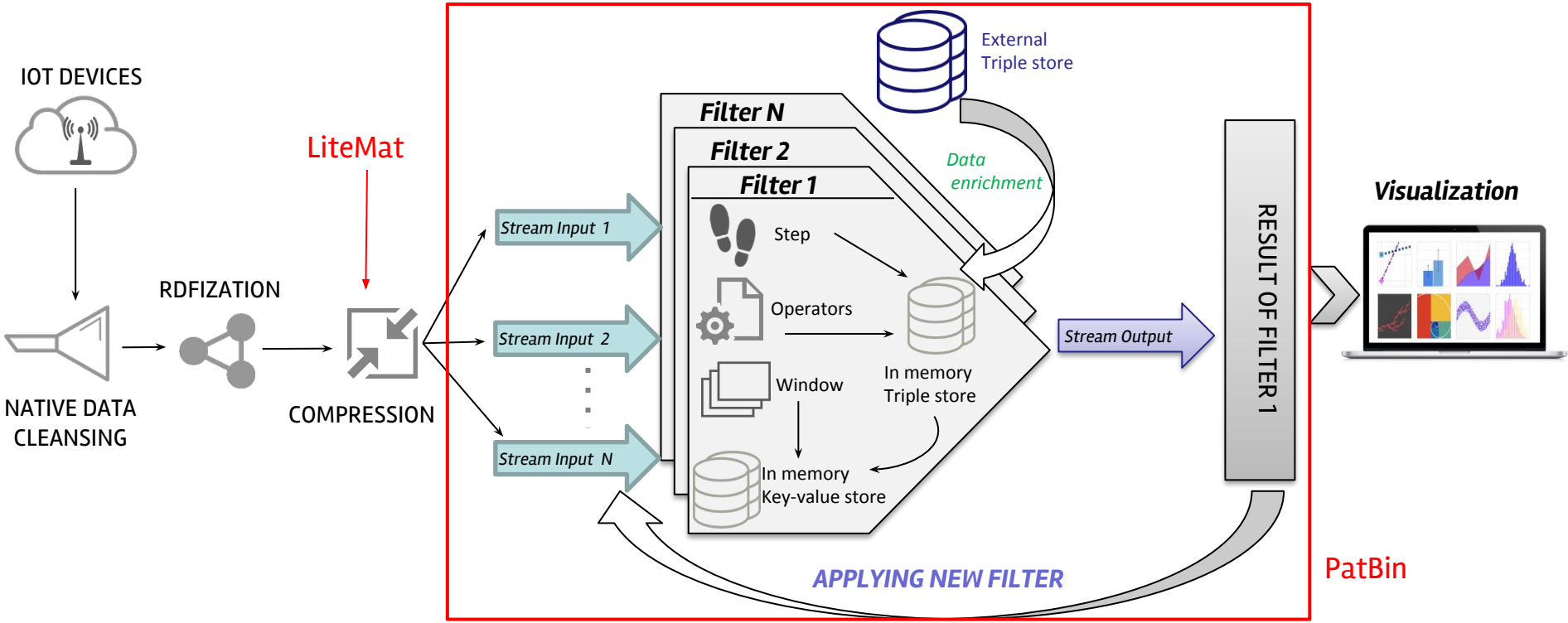
ARCHITECTURE

WAVES
ATOS SE



ARCHITECTURE

WAVES
ATOS SE



ISSUES OF STREAM PROCESSING

- **Compression efficiency**
  - No distribution
  - Decompression process

ISSUES OF STREAM PROCESSING

- **Compression efficiency**
  - No distribution
  - Decompression process

- **Solutions specific to a use case**
  - **Materialization:** huge amounts of data
  - **Query rewriting:** execution time

ISSUES OF STREAM PROCESSING

- **Compression efficiency**
  - No distribution
  - Decompression process
- **Solutions specific to a use case**
  - **Materialization:** huge amounts of data
  - **Query rewriting:** execution time
- **Lack of performance for heavy data load**

LITEMAT

Encoding the concepts and properties,
a compressed form conserving the hierarchy

ENCODING ALGORITHM

- **Principle:**
  - Binary structure conserving the semantics of the ontology
  - Each identifier is prefixed by its parent's
  - Conversion as integer identifiers
  - Supports RDFS

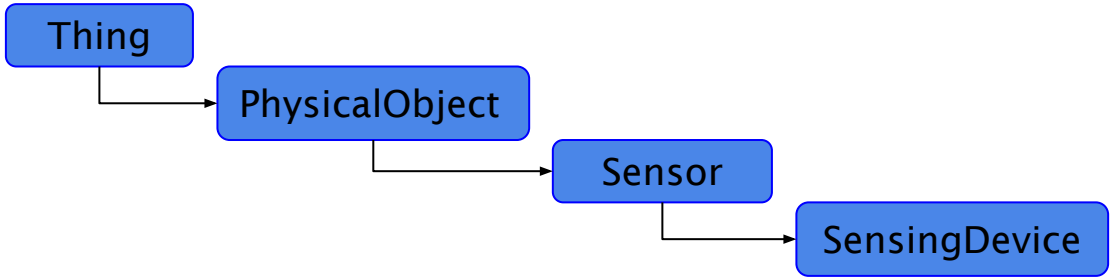
ENCODING ALGORITHM

- **Principle:**
  - Binary structure conserving the semantics of the ontology
  - Each identifier is prefixed by its parent's
  - Conversion as integer identifiers
  - Supports RDFS

- **Advantages:**
  - Compression maintaining hierarchy
    - ✓ No need for ontology at query runtime
    - ✓ Execution performance
  - Easy query rewriting

LITEMAT EXAMPLE

Extract from SSN



Transitivity

- waves:QBE04 rdf:type ssn:SensingDevice (explicit)
  - waves:QBE04 rdf:type ssn:Sensor
  - waves:QBE04 rdf:type DUL:PhysicalObject (implicit)

LITEMAT EXAMPLE

Concepts	Compression
owl:Thing	1
→ DUL:PhysicalObject	
→ ssn:Sensor	
→ ssn:SensingDevice	
→ dbo:Engine	
→ DUL:SocialObject	

LITEMAT EXAMPLE

Concepts	Compression
owl:Thing	100
→ DUL:PhysicalObject	101
→ ssn:Sensor	
→ ssn:SensingDevice	
→ dbo:Engine	
→ DUL:SocialObject	110

LITEMAT EXAMPLE

Concepts	Compression
owl:Thing	1
→ DUL:PhysicalObject	10100
→ ssn:Sensor	10101
→ ssn:SensingDevice	
→ dbo:Engine	10110
→ DUL:SocialObject	110

LITEMAT EXAMPLE

Concepts	Compression
owl:Thing	1
→ DUL:PhysicalObject	101
→ ssn:Sensor	101010
→ ssn:SensingDevice	101011
→ dbo:Engine	10110
→ DUL:SocialObject	110

LITEMAT EXAMPLE

Concepts	Compression
owl:Thing	100000 = 32
→ DUL:PhysicalObject	101000 = 40
→ ssn:Sensor	101010 = 42
→ ssn:SensingDevice	101011 = 43
→ dbo:Engine	101100 = 44
→ DUL:SocialObject	110000 = 48

- Identify the subclasses specific to a concept in a specific interval
  - e.g. [40, 48[= all physical objects

LITEMAT USAGE

- **Query reformulation:**

  x SELECT ?x WHERE { ?x rdf:type DUL:PhysicalObject . }

  ✓ SELECT ?x WHERE { ?x rdf:type ?v .
  FILTER (?v >= 40 && ?v < 48) }

- **Usage in WAVES:**
  - Encoding of streams and static knowledge bases
  - Integration in PatBin

PATBIN

Materialization and query rewriting over compressed forms

WHY PATBIN ?

- **Motivation**
  - Exploit repetitive information in streams & queries
  - Querying compressed data

- **Principle**
  - Combination of materialization and query rewriting
  - Decomposition of the triples in Pattern - Binding:
    - Patterns compressed, using Litemat
    - Bindings holding the objects

PATBIN PRINCIPLE

- **Pattern**
  - Use the predicates' identifiers from LiteMat
  - Format preserving the triples interlinking
    - example: 16:21:37:(29)
- **Binding**
  - Storage of the objects
    - example: "Q250";"Hubies haut"; ;"Pascal"
- **Advantages**
  - Pattern repetitive for streams (no regeneration)
  - No need for decompression

PATBIN EXAMPLE: STREAM

- **Stream (in blue):**

  _:x1 id "Q250"

  _:x1 date 30/03/2017

  _:x1 pressureMeasure _:x2

  _:x2 value 4.5

- **Pattern:**

  Ps = 38:40:56:(42)

- **Binding:**

  Bs = 30/03/2017;"Q250"; ;4.5

LiteMat dictionnary		
date	=	38
id	=	40
value	=	42
hLocation	=	54
pMeasure	=	56

PATBIN EXAMPLE: QUERY

- **Query (in red):**

```
SELECT ?x ?v
FROM   ...
WHERE {
        ?s id ?x .
        ?s hasLocation “Platform1” .
        ?s pressureMeasure ?y .
        ?y value ?v .
}
```

- **Pattern:**

Pq = 40:54:56:(42)

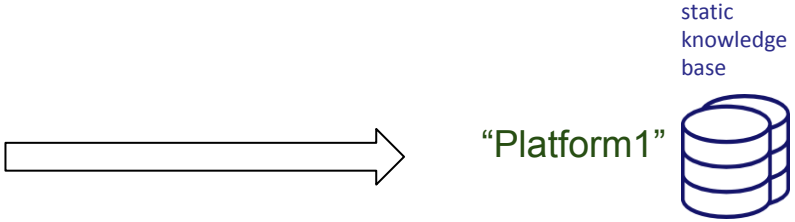
- **Binding:**

Bq = ;”Platform1”; ;

LiteMat dictionnary		
date	=	38
id	=	40
value	=	42
hLocation	=	54
pMeasure	=	56

PATBIN EXAMPLE

- **Need of materialization (in green):**
  - Ps ∩ Pq = Pi
    - 38:40:56:(42) ∩ 40:54:56:(42) = 40:56:(42)
  - Pq - Pi = Pm
    - 40:54:56:(42) - 40:56:(42) = 54
- **Fetch the missing information:**
  - SELECT ?y
    WHERE {
    ?x id "Q250" .
    ?x hasLocation ?y .
    ?x pressureMeasure ?z .
    }



PATBIN EXAMPLE

- **Enrich the stream triples (in light blue):**
  - 38:40:56:(42) + 54 = 38:40:54:56:(42) ⇨ P's
  - 30/03/2017;"Q250"; ;4.5 + "Platform1" = 30/03/2017;"Q250";"Platform1"; ;4.5 ⇨ B's
- **Verify the compatibility:**
  - Pq ⊂ P's = true
  - Bq ⊂ B's == true ?
    - ;"Platform1"; ; ⊂ 30/03/2017;"Q250";"Platform1"; ;4.5 = true
- **Give the result:**
  - "Q250" 4.5

CONCLUSION

CONCLUSION

- **Waves project**
  - RSP engine
  - Use case: drinkable water network management

- **LiteMat**
  - Compression with identifiers
  - Entity identifiers represent the ontology semantics

- **PatBin**
  - Efficient reasoning scheme
  - Combines materialization, query rewriting and compression based on LiteMat

FUTURE WORK

- **LiteMat**
  - Support of RDFS++
    - x *owl:sameAs*, transitive and inverse properties

- **PatBin**
  - Support of SPARQL operators
    - ✓ aggregates (MIN/MAX, SUM, AVG), ORDER, GROUP, HAVING
    - x OPTIONAL, FILTER, …

THANK YOU

LITEMAT PERFORMANCES

WAVES
ATOS SE

Dataset	Duration (sec.)	Added triples (%)	Deleted triples (%)	throughput (triples/sec.)
LUBM1K	172.7	0	0	466,552
LUBM10K	1,755.8	0	0	307,139
DBPedia	29.2	0	0	478,129
Wikidata	859.2	0	0.04	77,240

ONTOLOGY-BASED ENCODING WITH LITE MATERIALIZATION

Dataset	Duration (sec.)	Added triples (%)	throughput (triples/sec.)
LUBM1K	176.6	38.15	460,917
LUBM10K	3,136.8	38.2	272,796
DBPedia	57	13.6	356,348
Wikidata	1,482	57.9	123,571

ONTOLOGY-BASED ENCODING WITH FULL MATERIALIZATION

Query	Lite mat.	Full mat.	No mat.
Q1	15.6	20.2	21.8
Q2	15.6	17.2	21.6
Q3	72.2	99.2	81.4
Q4	50.2	79.8	27.2

QUERY TIMES (IN SEC.) FOR LITE/FULL/NO MATERIALIZATION

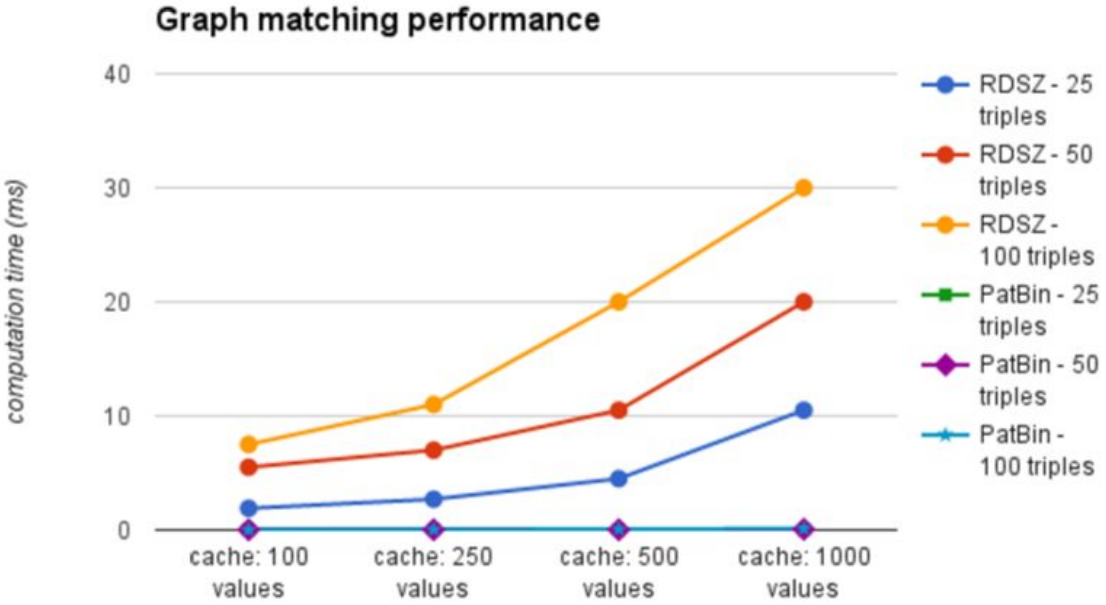
PATBIN PERFORMANCES

	RDSZ time	PatBin time	RDSZ size	PatBin size
5 triples	383	1	312	13
10 triples	387	2	370	29
25 triples	394	3	425	89
50 triples	397	6	523	184
100 triples	401	10	750	382

Signature generation performance for RDSZ and PatBin, time of compression in microseconds, size of he signature in bytes

PATBIN PERFORMANCES

WAVES
ATOS SE



Materialized RDF Stream and reformulated continuous query