Yasper 1.0: Towards an RSP-QL Engine

Riccardo Tommasini[1] and Emanuele Della Valle[1]

Politecnico di Milano, DEIB, Milan, Italy
{riccardo.tommasini,emanuele.dellavalle}@polimi.it

Abstract. In the Stream Reasoning (SR) research, working prototypes often came along with foundational investigations. For RDF Stream Processing (RSP) in particular, RSP engines empirically proved the approach feasibility and paved the road to application design and comparative analyses. Observing these real systems highlighted their heterogeneity and fosters new foundational achievements: RSP-QL, i.e. a reference a model that explains, unifies and can be used for correctness checking and optimization of RSP approaches. In this paper, we present Yasper 1.0 a brand new RSP engine that implements RSP-QL semantics and, we hope, will foster new empirical research on RSP.

1 Introduction

Since 2008, the Stream Reasoning (SR) community proposed SPARQL extensions to address the problem of continuous processing streams of RDF data (RSP) [4]. C-SPARQL, CEQLS and SPARQL$_{stream}$ emerged among many proposals for RSP and also thanks to the availability of working prototypes.

These prototypes, called RSP Engines, enabled to design stream reasoning applications, i.e. they empirically proved SR feasibility. They paved the road to comparative researches and benchmarks [5](cfr. § 2). Last but not least, RSP engines fostered more foundational research. Indeed, Dell'Aglio et al. [3] observed heterogeneities of existing prototypes and proposed RSP-QL, i.e. a unifying model for continuous SPARQL extensions. The application scope of RSP-QL comprises, but is not limited to, correctness checking, query planning and optimization. Although RSP-QL is more than a query language, the community is designing a syntax [2][1]. We agree with this decision and we are also convinced that an RSP engine will encourage RSP-QL adoption and foster empirical analysis as it happened with the aforementioned approaches.

In this paper, we present Yasper 1.0 (Yet Another RSP Engine), a brand new RSP engine that implements RSP-QL semantics. Yasper can answer queries using the syntax proposed in [2] and incorporates some lessons learned as well as new ideas on RSP, which can be studied empirically. Yasper is released open-source[2] under Apache 2.0 License and we welcome researchers and practitioners to test, study, benchmark and enhance it.

[1] https://github.com/streamreasoning/rsp-ql
[2] https://github.com/streamreasoning/yasper

2 Background

In this section, we summarize the notions of RSP-QL [3] that are required to understand how Yasper works.

An RDF Stream is a sequence of pairs (O_i, t_i), where t_i is a non-decreasing timestamp and O_i is either an RDF Graph or an RDF Triple.

The time-based sliding window operator $\mathbb{W}$ is a Stream-to-Relation (S2R) operator [1]. It is defined as a triple (α, β, t^0) that, starting at the timestamp t^0, defines a series of windows of width (α) and that slide of (β).

A Time-Varying Graph is a function that takes a time instant as input and produces as output RDF Graph which is called Instantaneous RDF Graph. The application of $\mathbb{W}$ on a RDF Stream S produces a Time-Varying Graph $\mathrm{TVG}_{\mathbb{W},S}$ that for any given time instant t at which $\mathbb{W}$ is defined outputs an Instantaneous RDF Graph, which is the result of coalescing all the RDF Graphs or triples that the current window contains[3].

An streaming dataset (SDS) is an extension of SPARQL dataset[4] that is composed by: an optional default graph A_0, n ($n \geq 0$) named Time-Varying Graphs, and m ($m \geq 0$) named sliding windows over k ($k \leq m$) data streams.

An RSP-QL query is continuously evaluated against an SDS by an RSP engine. The set ET of all the instants at which the evaluation occurs is determined by the reporting policy of the RSP engine that executes the query. RSP-QL defines a reporting policy for an RSP as a combination of one or more of the following strategies: **CC** Content Change – the engine reports if the content of the current window changes–, **WC** Window Close – the engine reports if the current window closes –, **NC** Non-empty Content – the engine reports if the the current window is not empty –, and **P** Periodic – the engine reports periodically.

The evaluation of a RSP-QL query outputs an instantaneous multiset of solution mappings for each evaluation time instant in "ET". Relation-to-Stream (R2S) [1] operators are required to transform the instantaneous multiset into a stream. RSP-QL comprises the following R2S operators: the **RStream** that emits each solution mappings; the **IStream** that emits the difference between the current solution mappings and previous ones, and; the **DStream** that emits the difference between the previous solution mappings and the current ones.

3 Yet Another RDF Stream Processing Engine

In this section, we introduce Yasper's architecture and how it implements RSP-QL concepts. Figure 1 shows the following Yasper's modules: Streams, Windowing, SDS, Querying and Reasoning. Modules that expose Yasper as a REST service are available[5] but not discussed.

The Stream module, Fig 1 (a), contains the classes to represent a Stream, which is identified by an URI and its content. A StreamItem is identified by

[3] The current window identified by $\mathbb{W}$ with the oldest closing time instant at t

[4] `https://www.w3.org/TR/rdf-sparql-query/#specifyingDataset`

[5] https://github.com/streamreasoning/rspservices

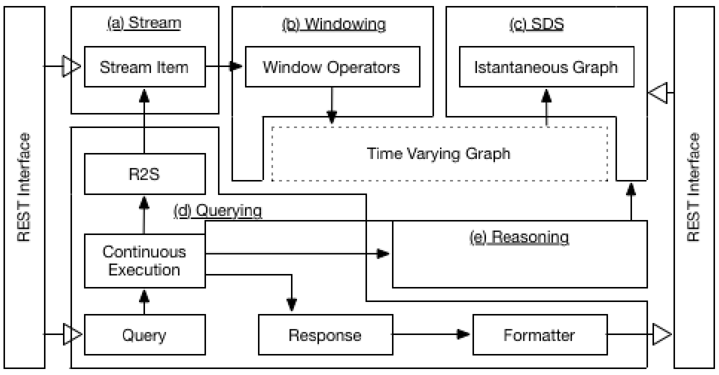

Fig. 1: Yasper's Modules: (a) Streams, (b) Windowing, (c) SDS, (d) Querying and (e) Reasoning. Only relevant architectural details are represented.

the triple $< t_i, t_e, O >$ where t_i is the time when s entered the system (*Ingestion Time*); t_e is the time when s occurred (*Event Time*) and O is a generic representation of the data in s, i.e. for RSP either a RDF Graph or a Triple[6].

The Windowing module, Fig 1 (b), is based on Esper[7], i.e. an open-source Data Stream Management Systems that relies on the Event Processing Language (EPL) and special objects called *listeners* that continuously receive the EPL queries outputs. Each EPL statement together with a dedicated listener represents a (named) time-based sliding window operator (henceforth referred as just window operator) on a RDF stream. The windowing is performed by means of temporal annotations of the StreamItems. Yasper works by default using Event Time, but it can be configured to work with Ingestion Time[8].

An Esper-based window operator maintains one time-varying graph. A window operator has two alternative ways to deliver content to Time-Varying Graph: (i) as a *Snapshot* – i.e. the window operator pushes to the Time-Varying Graph the whole window content – or (ii) as *Deltas* – i.e. window operator pushes to the Time-Varying Graph only the differences between the current window and the previous one in terms of additions and deletions. By default Yasper works in Snapshot mode, but it can be configured to work with Deltas. In both the ways, the Time-Varying Graphs reactively generates a Instantaneous RDF Graph any time the engine reports. Yasper reports the results on window close (**WC**) without empty content (**NC**).

Since the set ET is determined reading the StreamItems, at each evaluation time instant we can identify an Instantaneous SDS to evaluate a RSP-QL query against. A streaming dataset SDS, Fig 1 (c), can be reactively consolidated into a set of (named) Instantaneous Graphs[9] at the time t at which a Time-Varying

[6] We implement O using Apache Jena 3.

[7] www.espertech.com

[8] Event Time does not guarantee total ordering of StreamItems

[9] Slowly evolving RDF graph are represented as a (named) Time-Varying Graph too.

```
REGISTER STREAM <example> AS CONSTRUCT ISTREAM {?s ?p ?o}
FROM NAMED WINDOW  :win1  [RANGE 5 s, SLIDE 2s] ON STREAM :stream1
FROM NAMED WINDOW  :win2  [RANGE 5 s, SLIDE 5s] ON STREAM :stream2
WHERE { WINDOW ?w1 {?s ?p ?o}
        WINDOW ?w2 {?s ?p ?o}  FILTER (?w1 != ?w2) }
```

Listing 1.1: An example of RSP-QL Query.

Graph is updated. We implemented the SDS and this mechanism by extending Apache Jena 3 in-memory dataset.

The Querying module, Fig 1 (d), contains the elements for query instantiation and continuous execution. At this stage of development, Yasper accepts SELECT and CONSTRUCT queries written in RSP-QL syntax (e.g. Listing 1.1). Moreover, as the reader can observe in Listing 1.1, Yasper supports multi-streams query, despite [3] does not discuss how to handle them. The following Relation-To-Stream (R2S) operators are available R-, I- and DStream.

Finally, the reasoning module, Fig 1 (e), allows to answer RSP-QL queries under OWL entailment by means of Jena Rule reasoner. In the future we want to integrate RDFox, Ontop and DL reasoners e.g. Pellet or Hermit.

4 Conclusion

In this paper, we presented Yasper, an RSP engine for RSP-QL queries. Yasper adopts a generic stream representation, it can work with event time or ingestion time and it implements multi-stream queries evaluation. At the moment of writing, Yasper supports SELECT and CONSTRUCT queries. ASK support is a work in progress, while DESCRIBE requires further investigations for RSP. We plan to support all the reporting Policies in Yasper, either by configuration or as an extension of the proposed syntax.

Finally, we plan to perform an empirical evaluation [5] that compares Yasper architectural configurations and the existing RSP engine implementations.

References

1. Arasu, A., Babu, S., Widom, J.: The CQL continuous query language: semantic foundations and query execution. VLDB J. 15(2), 121–142 (2006)
2. Dell'Aglio, D., Calbimonte, J., Della Valle, E., Corcho, Ó.: Towards a unified language for RDF stream query processing. In: ESWC 2015 Satellite Events Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers. pp. 353–363 (2015)
3. Dell'Aglio, D., Della Valle, E., Calbimonte, J., Corcho, Ó.: RSP-QL semantics: A unifying query model to explain heterogeneity of RDF stream processing systems. Int. J. Semantic Web Inf. Syst. 10(4), 17–44 (2014)
4. DellAglio, D., Della Valle, E., van Harmelen, F., Bernstein, A.: Stream reasoning: A survey and outlook. Data Science (Preprint), 1–24
5. Tommasini, R., Della Valle, E., Balduini, M., Dell'Aglio, D.: Heaven: A framework for systematic comparative research approach for RSP engines. In: The 13th International ESWC, Heraklion, Crete, Greece, 2016, Proceedings. pp. 250–265 (2016)