On the need for adaptivity in RDF Stream Processing

Zia Ush Shamszaman, Muhammad Intizar Ali, and Alessandra Mileo

Insight Centre for Data Analytics,
National University of Ireland
Galway, Ireland
{zia.shamszaman,ali.intizar,alessandra.mileo}@insight-centre.org

RDF streams are generated through numerous modalities and different levels of granularity and refers to the application of Linked Data principles to streaming data published on the (Semantic) Web. RDF stream Processing (RSP) combines advantages from two different domains, i.e., Linked Data processing and traditional Stream Processing (SP). RSP ensures data freshness over conituous data like SP and inherits unique data identification (URI) concept form Linked Data, moreover RSP is capable to process combinly static background data along with stream data which is not considered in traditional SP. Over the last few years several stream processing systems have been proposed for efficient processing of RDF stream, just to name a few C-SPARQL[2], CQELS[6], SPARQLstream[3] and EP-SPARQL[1]. Though the existing RSP systems are in their cradle, it is important to identify key features and requirements of RSP engines in order to built more robust and scalable solutions.

Despite initiatives such as the W3C RSP Working Group [1] aim at defining a common model for producing, transmitting and continuously querying RDF Streams, we still believe there is a wide range of features which might limit the applicability of RSP solutions. We classify these features into two categories:

i) *design-time features* include aspects such as input data model, language to define processing rules, operational semantics, and supported streaming operators, and
ii) *run-time features* include aspects such as execution time, processing techniques, quality of service (QoS), privacy, target domain of applications and more.

Existing directions in RSP research focus on reaching a consensus on how to process RDF streams, rather than on the ability to adapt and react to changing requirements of the applications and properties of the underlying data streams. It is not trivial to find a configuration of such features that can work independently of the data and the application domains. Therefore, we believe it is important to design flexible RSP solutions which can adapt to the application requirements and let the application discover and select the underlying RSP engine (or a configuration of their features) on the fly through dynamic adaptation.

[1] https://www.w3.org/community/rsp/

Consider a use-case scenario of home surveillance system. Mike's house is equipped with various sensors, which are capable of producing data streams and sending it over a connected network. Mike has installed a home surveillance system to continuously monitor his house remotely. Home surveillance system can be configured to produce various notifications e.g. any burglar attempt, fire or smoke detection and periodic monitoring over children activities. Notifications are generated using two surveillance approaches, (i) continuous or (ii) periodic, where former approach is time efficient but will consume more resources, while later approach is cost effective with a compromise over time efficiency and accuracy. It is worth mentioning here that this scenario demands RSP as data freshness and background data both are required for application, e.g., childrens profile and mike's profile is static background data and sensor observations are stream data. Hence RSP is suitable for this scenario as it allows to process static data and background data in a single engine.

Existing RSP engines are designed to have two types of input data models for query execution, (i) *data driven* - query is executed whenever data is received by RSP engine, or (ii) *time driven* - query is executed periodically. *Data driven* approach is ideal for continuous monitoring while *time driven* approach is good for periodic monitoring. C-SPARQL follows time driven strategy where result may get stale if the re-execution frequency is lower than the frequency of the updates [5] and thus not suitable for application where delay can be crucial (e.g. burglar attempt notification). Another RSP engine, CQELS follows data driven approach which is ideal for time efficient but can be resource expensive for periodic notification system (e.g. periodic surveillance over childrens activities). Input data models are selected at design time, which limits the adaptation capability of applications built using any of the existing RSP engines to switch the input data model depending on the changing requirements of the scenario.

Another important design time feature of RSP engines is output streaming operator. Work in [5] has showed that C-SPARQL suffers from duplicate results for simple queries and misses some certain output in complex queries. This is due to implementing the R-Stream operator as an output operator in which old triples does not get removed from the time window. However, results of another evaluation, conducted in [7, 4], have shown that C-SPARQL provide more correct results among other RSP engines with different benchmark queries. Diversity in output results produced by various RSP engines is a known phenomenon.

Input data models and output streaming operators are mere two examples to showcase how some design-time features may affect correctness and performance of the RSP engine for query execution, and therefore would benefit from being adapted to input data and application requirements, rather than being fixed at design-time within the specific engine. We are interested in investigating and identifying various design and run time features of RSP engines which can play a pivotal role for designing adaptive RSP solutions. Adaptation can be implemented either by extending the existing RSP engines or by designing a configuration layer on top of the existing RSP engines.

Authors Expertise The authors of this paper have established expertise in designing and implementing RDF stream processing solutions. Zia Ush Shamszaman is a PhD student working on adaptive solutions for robust applications that rely on consuming linked data streams. Dr. Muhammad Intizar Ali is a postdoctoral researcher involved in research activities for designing and implementing query language syntax and semantics, query parsers, compilers and processors and he is leading research and development activities in the area of Smart Cities and Smart Enterprise. Dr. Alessandra Mileo is a senior Research Fellow heavily involved in research and development activities that are relevant to the RSP group, including stream reasoning, query optimisation and event processing, and she is PI in EU funded smart city projects and industry funded projects in the area of the Internet of Everything.

Relation to the RSP community and relevance to the broader ESWC audience The W3C RSP Working Group aims at a definition of a common model to represent and process RDF streams. Such a common and extensible core model for RDF stream processing can serve as a starting point for RSP engines to be able to talk to each other and interoperate. We believe our research interest is complementary and relevant to RSP since it can help identifying crucial features that can affect and provide insigths on how such model should be defined. Independently of the existence of one singe RSP engine or several ones, the ability to adapt to changing environment can serve as a heuristic-based layer for more robust and flexible RSP solutions. In the broader scope of the European Semantic Web Conference (ESWC), our ideas will contribute to new discussions for those that rely on semantic technologies to address the velocity and variety challenges of Big Data, which requires interoperable and flexible tools to adapt to changing environments in a variety of applications around the Internet of Things and the Internet of People, including Smart Cities, Health Monitoring, and the media industry.

References

1. F. P. R. S. S. N. Anicic, D. EP-SPARQL: a unified language for event processing and stream reasoning. In *Proc. of the 20th WWW Conf.* ACM, 2011.
2. D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. C-sparql: Sparql for continuous querying. In *Proc. of the 18th*, WWW Conf. ACM, 2009.
3. J.-P. Calbimonte, H. Jeung, O. Corcho, and K. Aberer. Enabling query technologies for the semantic sensor web. *IJSWIS*, (1), 2012.
4. D. Dell'Aglio, J.-P. Calbimonte, M. Balduini, O. Corcho, and E. D. Valle. On correctness in rdf stream processor benchmarking. In *ISWC*, LNCS, 2013.
5. D. Le-Phuoc, M. Dao-Tran, M. Pham, P. Boncz, T. Eiter, and M. Fink. Linked stream data processing engines: Facts and figures. *ISWC 2012*.
6. D. Le-Phuoc, M. Dao-Tran, J. X. Parreira, and M. Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. *ISWC 2011*.
7. Y. Zhang, M.-D. Pham, Ó. Corcho, and J.-P. Calbimonte. Srbench: A streaming rdf/sparql benchmark. In *ISWC (1)*, 2012.