RoXi: a Framework for Reactive Reasoning

Pieter Bonte[0000−0002−8931−8343] and Femke Ongenae[0000−0003−2529−5477]

Ghent University - imec, Belgium
Technologiepark-Zwijnaarde 126, B-9052 Ghent, Belgium
pieter.bonte@ugent.be

Abstract. The Stream Reasoning research paradigm aims to target application domains that need to solve both data variety and velocity at the same time. Many of these domains benefit from processing the data as close to the source as possible, e.g. in Internet of Things applications we see a paradigm shift towards Edge processing, and in the Web, we see more and more effort towards decentralization of the Web where applications run directly inside the browser. However, current Stream Reasoning engines are not able to reuse the same code-base to run applications in the cloud, edge, or browser.
In this paper, we present RoXi, a Reactive Reasoning framework that provides the needed building blocks to realize Stream Reasoning applications that can target cloud, edge, and browser environments.

Keywords: RDF Stream Processing · Edge Processing · Reasoning · Decentralized Web

1 Introduction

In recent years, the interest in streaming data has increased for application domains that combine data variety, i.e. data that require some form of data integration, with the requirement to process data in a reactive fashion, i.e., as soon as possible and before the data are no longer useful [1]. Examples of such application domains include Smart Cities, Industry 4.0, Web Analytics, etc. Stream Reasoning (SR) is a research initiative that combines Semantic Web with Stream Processing technologies to target both the data variety and velocity at the same time [8]. Semantic Reasoning allows to target the data variety, by providing means to integrate and abstract data from various sources. Furthermore, it provides a way to interpret any defined domain knowledge.

Many of the application domains that target both data variety and velocity can offload parts of the processing closer to where data is being produced in order to speed up computation. In Internet of Things (IoT) applications, this means to the Edge of the network, while in Web applications, this means to the browser of the user. Running computation in the browser has become more popular with the rise of the decentralized Web, as supported by the Solid project [6]. Both the Edge environment and browsers provide limited resources compared to processing all data in the cloud. However, many of the same building blocks

should be able to run both in the cloud and at the edge/browser. This calls for a Reasoning framework where the same code-base can be used for running applications both in the cloud or at the edge/browser.

We introduce RoXi, a framework for Reactive Reasoning that is fully written in Rust. RoXi allows to use the same code-base to develop applications that require Reactive Reasoning, disregardless of performing computation at the cloud, edge or browser. RoXi can run in the browser through the support of WebAssembly, at the edge through Rust's ability to optimize code for low-level devices and at the cloud.

2 Related Work

Various reasoners and Stream Reasoners have been proposed over the years [9], however, typically focusing on running high-performance hardware as found in the cloud. RDFox [4] is a highly-scalable reasoning-enabled RDF store and has shown to be able to run at the Edge, however, it does not support any browser run-time, nor stream processing capabilities. Fed4Edge [5] is an RDF Stream Processing engine optimized to run in an edge environment, it focuses on query answering on RDF data streams. However, it does not provide any reasoning capabilities or functionality to run inside the browser. DIVIDE [2] allows to optimize reasoning rules for the evaluation at the edge, however, it does not support any browser run-time or optimized code for running on low-level devices. Hylar [7] is a reasoner written in Javascript and designed to run in the browser. However, the Javascript run-time makes it an unsuited candidate to efficiently run at the Edge, in the cloud or efficiently handle high-velocity data streams.

3 Architecture

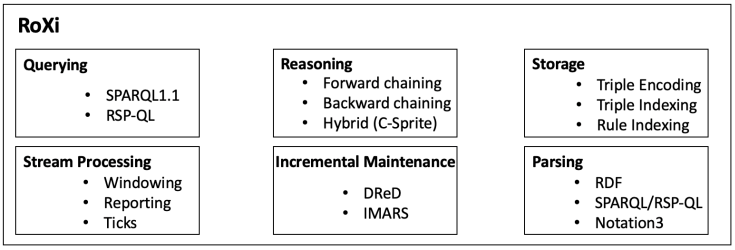

Fig. 1. Overview of the supported components in RoXi's architecture

RoXi is a framework for Reactive Reasoning, providing Querying, Reasoning and Stream Reasoning functionality. Fig. 1 visualizes RoXi's architecture, consisting of the following components:

1. **Querying**: RoXi provides support for both SPARQL1.1 queries on static data and RSP-QL queries on streaming data.

2. **Reasoning**: RoXi has at this point support for rule-based reasoning, with Datalog expressivity. We support a subset of *Notation3* rules[1] that relate to the expressivity of *Datalog*. RoXi supports both forward- and backward-chaining reasoning algorithms. For efficient reasoning over RDF data streams, RoXi implements the C-Sprite algorithm [1], i.e. a hybrid approach that optimizes the reasoning based on the registered queries.
3. **RDF Stream Processing**: RoXi aims to provide the same RDF Stream Processing interfaces and functionality as RSP4J [8], which aims to unify RDF Stream Processing. RoXi provides *Time-based Windows* that allow to cut the unbounded streams in processable chunks. *Different Reporting* dimensions are supported that allow to configure when the window triggers: *(a) Content Change*: reports when the content of the current window changes, *(b) Window Close*: reports when the current window closes, *(c) Non-empty Content*: reports when the current window is not empty, and *(d) Periodic* reports periodically. Various *Ticks* are supported that can configure how the evaluation of the reporting should be triggered: time-driven, tuple-driven, or batch-driven. Once the window triggers, different components can be called, such as *Querying* and *Reasoning*.
4. **Incremental Maintenance**: incremental maintenance programs allow to keep an updated view on all inferred triples through forward-chaining. These programs update the view when triples are added or removed. At this point, DReD [3] and IMARS incremental maintenance programs are supported in RoXi. The latter focus specifically on Window-based maintenance. In the future, the more optimized Forward-Backward-Forward [3] and the naive counting-based [3] approaches will be provided as well.
5. **Storage**: To reduce memory usage, RoXi employs dictionary encoding on the consumed triples. To speed up query evaluation, RoXi supports Indexed Datastores for all triples and rules.
6. **Parsing**: RoXi supports parsing of RDF, SPARQL, RSP-QL and Notation3.

RoXi is implemented in Rust and can be compiled to run at the edge, cloud or browser. The code is available on our Github-page[2]. RoXi's functionality can be tested directly in the browser using the Web UI[3].

4 Conclusion & Future Work

This paper presents RoXi, a framework for Reactive Reasoning. RoXi provides components for querying, reasoning, incremental maintenance and RDF Stream Processing. RoXi can be compiled to run in the cloud, edge and inside the browser, allowing the same code-base to be used while targeting IoT, Edge and decentralized Web applications.

In our future work, we aim to support more implementations of different technologies for each component in RoXi's hierarchy. For *Querying*, we wish to

[1] https://w3c.github.io/N3/spec/
[2] https://github.com/pbonte/roxi
[3] https://pbonte.github.io/roxi/index.html

support SHACL and research its applicability in combination with the *RDF Stream Processing* component. In terms of *Reasoning*, we wish to investigate the integration of more expressive reasoning algorithms both for reasoning over static data, e.g. to support OWL2 DL, and streaming data e.g. using temporal reasoning such as datalogMTL. For *RDF Stream Processing*, we aim to support more advanced types of windowing, such as session windows. We also aim to formalize the connection between reasoning and RDF Stream processing while using RoXi as a prototype. For *Incremental Maintenance*, we aim to integrate the more optimized Forward-Backward-Forward [3] and the naive counting-based [3]. In terms of *Storage*, we wish to experiment with different indexing techniques and their relation with streaming data, i.e. researching data structures that optimize trade-offs between the added value of the indexes and the cost of performing the indexing on data that changes frequently.

RoXi brings us one step closer to realize the Stream Reasoning vision by providing a Reactive Reasoning framework that allows the same code-base to be run in the cloud, edge or even in the browser.

Acknowledgement This work is funded by Pieter Bonte's postdoctoral fellowship of Research Foundation Flanders (FWO) (1266521N) and by SolidLab Vlaanderen (Flemish Government, EWI, and RRF project VV023/10).

References

1. Pieter Bonte, Riccardo Tommasini, Filip De Turck, Femke Ongenae, and Emanuele Della Valle. C-sprite: efficient hierarchical reasoning for rapid rdf stream processing. In *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems*, pages 103–114, 2019.
2. Mathias De Brouwer and et al. Context-aware & privacy-preserving homecare monitoring through adaptive query derivation for iot data streams with divide. *Semantic Web Journal*.
3. Boris Motik, Yavor Nenov, Robert Piro, and Ian Horrocks. Maintenance of datalog materialisations revisited. *Artificial Intelligence*, 269:76–136, 2019.
4. Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. Rdfox: A highly-scalable rdf store. In *ISWC*, pages 3–20. Springer, 2015.
5. Manh Nguyen-Duc, Anh Le-Tuan, Jean-Paul Calbimonte, Manfred Hauswirth, and Danh Le-Phuoc. Autonomous rdf stream processing for iot edge devices. In *JIST*, pages 304–319. Springer, 2020.
6. Andrei Vlad Sambra, Essam Mansour, Sandro Hawke, Maged Zereba, Nicola Greco, Abdurrahman Ghanem, Dmitri Zagidulin, Ashraf Aboulnaga, and Tim Berners-Lee. Solid: a platform for decentralized social applications based on linked data. *MIT CSAIL & Qatar Computing Research Institute, Tech. Rep.*, 2016.
7. Mehdi Terdjimi, Lionel Médini, and Michael Mrissa. Hylar+ improving hybrid location-agnostic reasoning with incremental rule-based update. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 259–262, 2016.
8. Riccardo Tommasini, Pieter Bonte, Femke Ongenae, and Emanuele Della Valle. Rsp4j: An api for rdf stream processing. In *ESWC*, pages 565–581. Springer, 2021.
9. Riccardo Tommasini, Pieter Bonte, Fabiano Spiga, and Emanuele Della Valle. *Web Stream Processing Systems and Benchmarks*, pages 109–138. Springer International Publishing, Cham, 2023.