# Supporting Real-Time Monitoring in Criminal Investigations

Robin Keskisärkkä(✉) and Eva Blomqvist

Linköping University, Linköping, Sweden
{robin.keskisarkka,eva.blomqvist}@liu.se

**Abstract.** Being able to analyze information collected from streams of data, generated by different types of sensors, is becoming increasingly important in many domains. This paper presents an approach for creating a decoupled semantically enabled event processing system, which leverages existing Semantic Web technologies. By implementing the actor model, we show how we can create flexible and robust event processing systems, which can leverage different technologies in the same general workflow. We argue that in this context RSP systems can be viewed as generic systems for creating semantically enabled event processing agents. In the demonstration scenario we show how real-time monitoring can be used to support criminal intelligence analysis, and describe how the actor model can be leveraged further to support scalability.

**Keywords:** Semantic event processing · Event processing · RDF stream processing · Actor model · Criminal intelligence

## 1 Introduction

Semantic Web (SW) technologies provide flexible tools for working with heterogeneous data, and Linked Data principles enable information to be shared by explicitly articulating the underlying schemas and ontologies.

Traditional SW technologies have been developed to support slowly evolving (or static) data, and scale quite poorly when data is highly dynamic. In recent years, a number of RDF Stream Processing (RSP) systems have therefore been developed to support streaming Linked Data, focusing on timely execution of continuous queries over streams.

Unlike most types of event processing approaches, such as Drools fusion[1] and ESPER[2], RSP systems use the Linked Data principles to leverage the semantics in the streaming data. The available RSP systems, however, provide only a limited set of features out-of-the-box. For example, in the available versions of C-SPARQL [3], CQELS [5], INSTANS [8], and ETALIS/EP-SPARQL [1], streams, queries, and result listeners are closely coupled with their respective engine. This can make them difficult to use in settings where streams are not

[1] http://www.drools.org/.
[2] http://esper.codehaus.org/.

DOI: 10.1007/978-3-319-25639-9_16

under the direct control of the system itself, or when other technologies need to be included in the event processing pipeline.

In this paper we present an approach for creating decoupled semantically enabled event processing systems by leveraging existing technologies, and demonstrate its applicability in a criminal intelligence scenario.

## 2 Related Work

The Streaming Linked Data framework, based on C-SPARQL, allows publishers to stream data to a central server, where the data can be queried, stored, replayed, decorated, and republished as new streams [2]. This drastically improves the flexibility of the RSP system, making it possible to provide APIs and add functionality to the standard C-SPARQL system.

The Super Stream Collider is platform for combining semantically annotated Linked Stream and Linked Data sources [7]. It was constructed around the CQELS engine and supports registering of streams and queries in a web-based interface. This simplifies the querying of static and dynamic resources, and allows rapid development Linked Stream mashups that can be used by applications.

Both approaches significantly increase the flexibility of their respective RSP systems, but the frameworks are still closely coupled with the structure of the underlying engines. There are several issues related with closely coupled systems. For example, a close coupling with the data streams limits the ability to handle stream overload. Most engines naively attempt to handle the full set of streaming data, regardless of rate, volume, and number of registered queries, which can create bottlenecks that deteriorate performance across the entire system.

CQELS Cloud uses scalable parallel algorithms to support elastic parallelising of query execution [6]. This approach helps to scale processing, both in terms of parallel queries and stream rates, but the underlying assumption is still that it will be possible to scale up the processing to handle all the incoming data.

## 3 Architecture Overview

In a decoupled event processing system the states of event producers, events, and event consumers are independent of each other [4]. This demo implements the actor model to handle message based communication, which completely separates the states of the different parts of the event processing system.

We implemented the actor model using the Akka[3] toolkit and runtime environment, which supports efficient, lightweight, and scalable, asynchronous message communication between its actors. This approach supports a robust and possibly distributed system, which avoids slow-downs that may result from individual components.

*Event producers* generate internal event streams from event sources, which are typically outside the control of the system, and these event streams are then

[3] http://akka.io/.

fed to *event consumers*. Event Processing Agents (EPAs) are special in that they are both event consumers and event producers [4]. To handle the communication between producers and consumers we created an event distribution mechanism, which pushes data from event producers (identified by URIs) to the listening event consumers. Sequential processing of event streams is made possible by pipelining several EPAs, thus enabling event processing requiring multiple steps.

The novelty of this approach in the RSP context is that it allows us to abstract from the implementation specific aspects of individual RSP engines, facilitating the use of multiple different engines within the same system. We represent events as RDF graphs, which requires RSP engines that support only RDF triples streams to decompose the events into triples internally. To demonstrate our approach we integrated the CQELS engine by creating wrappers for its internal RDF stream and query listener.

We can view each registered RSP query together with its output stream as an EPA. This allows us to seamlessly integrate different RSP engines in the overall workflow, and we can use optimized EPA components for such things as text analysis, stream decoration, and statistical analysis.

## 4 Demonstration Scenario

Criminal investigations involve a wide range of data sources, ranging from criminal records, modus operandi descriptions, case files, criminal reports, videos, images and more. When large volumes of data need to be interpreted, analysts have to rely heavily on their own domain expertise and skill to detect potential patterns in the data. However, pure manual work scales poorly as the amount of accumulated and continuously delivered data increases. Although many of the tasks of the analysts are difficult to articulate some can be formalized as rules, for example, to filter, aggregate, or decorate events.

The task in the demonstration scenario is inspired by real-world investigative tasks of the police. Based on a stream of Automatic Number Plate Recognition (ANPR) observations, originating from CCTV footage, the task is to monitor vehicles to detect when two "persons of interest" are possibly meeting up.

The system generates an alert when two vehicles, belonging to persons of interest in an investigation, are observed in close proximity of each other within a small time window. All observations are visualized on a map, while detected events are persisted in a separate tab.

The available data is: (1) a stream of ANPR observations, (2) a dataset containing the locations of the ANPR cameras, and pre-calculated distances between them, (3) a dataset containing registered owners of vehicles, and (4) "persons of interest" within specific investigations. The static datasets (2–4) are represented as RDF, while the ANPR stream is converted into RDF (from csv) in real-time via a direct mapping by a designated event producer. A diagram of the demonstration setup can be seen in Fig. 1.

The data used in the demonstration was created artificially (by the authors), but care was taken to follow the format of the ANPR data available within

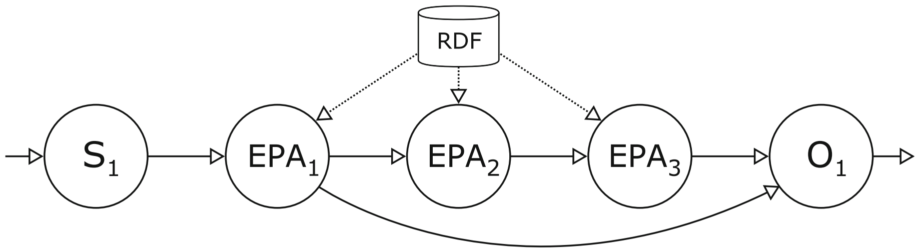

**Fig. 1.** An event producer, $S_1$, converts a stream of csv-strings into an RDF stream. The generated event stream is decorated with camera location information, $EPA_1$. Vehicle owner information is added to the stream, $EPA_2$, and finally, vehicles belonging to the same cases that are potentially meeting up are detected, $EPA_3$. The event streams generated by $EPA_1$ and $EPA_3$ are consumed by the output formatter, $O_1$, which converts them for use in the web-interface.

the VALCRI project[4], which in turn reflects the formatting of actual ANPR data used by the UK police. The locations of the ANPR cameras were set to street crossings in small section of London (UK), and possible paths were generated within and through this area. Observations of randomized registration plate numbers were assigned to paths, and the average delay between observations was approximated based on the distance between cameras. The number of observations were balanced against time of day to roughly correspond to the distribution of ANPR data in the VALCRI project. The queries and datasets used in the demo, and a recording of the running demo is available at http://valcri.ida.liu.se:8080/eswc2015/.

## 5 Scalability

The demonstration scenario was run on standard PC with a 1.7 GHz dual-core processor and 4 Gb RAM. In the scenario, the CQELS engine runs three parallel queries and processes up to 85 events per second, which is equivalent to approximately 285 triples per second. When benchmarking the system the same setup gives acceptable performance even when increasing the application time by up to 15 times, thereby processing more than 1200 events per second (corresponding to more than 4000 triples per second).

The Akka framework supports communication between different virtual machines running on the same machine, as well as communication in a peer-to-peer fashion. This means that the approach could be used to tackle several scalability issues, for example, to support more parallel queries by running and interlinking several instances of RSP engines on separate machines.

## 6 Conclusions

We have shown the potential of leveraging SW technologies and RSP engines in the context of semantic event processing. Our demo application demonstrates

[4] http://www.valcri.org/.

how an actor system can be used as a way of setting up a decoupled event processing system, where RSP engines can be viewed as a generic means for creating semantically enabled EPAs. We also describe how the same architecture can be used to distribute processing and leverage more than a single RSP system, for example, to take advantage of engine specific features.

In criminal investigations manually processing continuously delivered messages is often not possible. Limited resources means that data is often logged until need for analysis arises, thereby missing out on the potential benefit of detecting events in real-time. The demo scenario shows how event processing can be used to support some investigative tasks, and how RSP systems can be used to support semantically enabled event processing. The demo visualization shows how this has the potential to be used to develop tools for criminal investigations.

**Acknowledgments.** This work was supported by the EU FP7 project Visual Analytics for Sense-making in Criminal Intelligence Analysis (VALCRI) under grant number FP7-SEC-2013-608142.

## References

1. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: Proceedings of the 20th International Conference on World Wide Web (2011)
2. Balduini, M., Della Valle, E., Dell'Aglio, D., Tsytsarau, M., Palpanas, T., Confalonieri, C.: Social listening of city scale events using the streaming linked data framework. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., et al. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 1–16. Springer, Heidelberg (2013)
3. Barbieri, D.F., Braga, D., Ceri, S., Valle, E.D., Grossniklaus, M.: Querying RDF streams with C-SPARQL. SIGMOD Rec. **39**(1), 20–26 (2010)
4. Etzion, O., Niblett, P.: Event Processing in Action, 1st edn. Manning Publications Co., Greenwich (2010)
5. Le-Phuoc, D., Dao-Tran, M., Parreira, J.X., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 370–388. Springer, Heidelberg (2011)
6. Le-Phuoc, D., Quoc, H.N.M., Le Van, C., Hauswirth, M.: Elastic and scalable processing of linked stream data in the cloud. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 280–297. Springer, Heidelberg (2013)
7. Quoc, H.N.M., Serrano, M., Le-Phuoc, D., Hauswirth, M.: Super stream collider-linked stream mashups for everyone. In: Proceedings of the Semantic Web Challenge Co-located with the 11th International Semantic Web Conference, Boston, MA, USA, November 2012
8. Rinne, M., Nuutila, E., Törmä, S.: INSTANS: high-performance event processing with standard RDF and SPARQL. In: Proceedings of the ISWC 2012 Posters and Demonstrations Track, Boston, US (2012)