Towards a semantic edge processing of sensor data. An incipient experiment

Paula-Georgiana Zălhan[1][0000−0002−4879−4394],
Gheorghe Cosmin Silaghi[1][0000−0002−3447−4736], and
Robert Andrei Buchmann[1][0000−0002−7385−1610]

Business Informatics Research Centre,
Babeş-Bolyai University, Cluj-Napoca, Romania
{paula.zalhan,gheorghe.silaghi,robert.buchmann}@econ.ubbcluj.ro

Abstract. This paper addresses a semantic stream processing pipeline, including data collection, semantic annotation, RDF data storage and query processing. We investigate whether the semantic annotation step could be moved on the edge, by designing and evaluating two alternative processing architectures. Experiments show that the edge processing fulfills the low-latency requirement, facilitating the parallel processing of the semantic enrichment for the sensor data.

Keywords: semantic stream processing · edge computing · sensor data.

1 Introduction

Industrial Wireless Networks (IWNs) have the potential to connect and integrate smart entities with traditional industries to provide greater flexibility, intelligence and adaptation [11]. These entities collect and transmit massive amounts of data regarding the working environment, which requires low latency processing capabilities to enable real-time monitoring of the industrial site. The stream data characteristics impose significant challenges to existing industrial systems. Thus, innovative solutions are demanded to overcome these issues by combining quantity-oriented methods with semantics-oriented techniques towards building processing pipelines that fulfill the needs of Industry 4.0 environments.

The main goal of this paper is to deliver faster insights of underlying large-scale sensor data generated in an industrial environment. Given a Semantic Stream Processing (SSP) [10] pipeline which includes [16] data collection, semantic annotation, RDF data storage and query processing, we investigate whether edge processing of semantic annotation could deliver a higher throughput of messages processed by the system. The pipeline's architecture is built with the help of various tools that include: the Apache Kafka [1] distributed streaming platform, the GraphDB [3] semantic database server and the Semantic Sensor Network (SSN) ontology [8].

To fulfill the low latency requirement of a smart factory use case, we evaluate two alternative pipeline architectures: (i) semantic annotation after the sensor

data is ingested into the real-time system, and (ii) semantic annotation on the "edge" – i.e., at the producer point. The experiment reported in this paper supports the decision on which approach should be considered in order to advance the proposed pipeline beyond the current, proof-of-concept status.

This paper is organized as follows. Section 2 reviews the existing SSP systems. Section 3 presents the adopted methodology and the architecture of our SSP pipeline. Section 4 describes the implemented alternatives, reports and discusses the results. Section 5 concludes the work.

2 Related work

The RDF Stream Processing (RSP) deals with fast changing data that can be semantically modeled by means of the RDF model [6]. Initially, centralized RSP engines such as CQELS [9] were proposed. However, these systems are unable to handle large-scale data originating from heterogeneous sources. Distributed RSP engines such as Strider [14] were designed to address the above-mentioned limitations. They perform parallel processing over the incoming data using a cluster computing infrastructure. Other solutions such as SEASOR [13] are based on a middleware component to deal with the real-time property of heterogeneous incoming data streams, delivered to various destination systems. In addition to SSP, Stream Reasoning [7] extends the traditional RSP engines with rule-based, logical, reasoning capabilities. Furthermore, stream reasoning could be combined with machine learning techniques [5].

3 Solution overview: a SSP pipeline

In order to build our SSP pipeline [16], we adopted the Design Science Methodology (DMS) [15]. We followed an iterative development cycle with the explicit intention of improving the pipeline's performance, which is empirically investigated considering a smart factory scenario. We are specifically interested in deciding the location in the pipeline where the semantic annotation should take place, for improving the system latency. Fig. 1 presents the architecture of our SSP pipeline, which is based on Apache Kafka [1] to collect and process the streaming data, GraphDB [3] to store the annotated data streams and the SPARQL query language to analyze the resulting RDF graph.

The data streams are generated by various sensors deployed in a smart factory ecosystem. The continuous data gathered from sensor sources is collected and processed by a distributed data ingestion system. Apache Kafka is employed for ingesting the sensor streams because it could handle large-scale data in a fault-tolerant manner [12]. Kafka works as a cluster connecting multiple producers and consumers to one or more servers known as brokers. Apache Zookeeper [2] is used to store metadata about the Kafka cluster.

In the Smart Factory scenario, the incoming data is categorized into topics and analyzed on the fly using the Kafka core API. To provide machine-readable

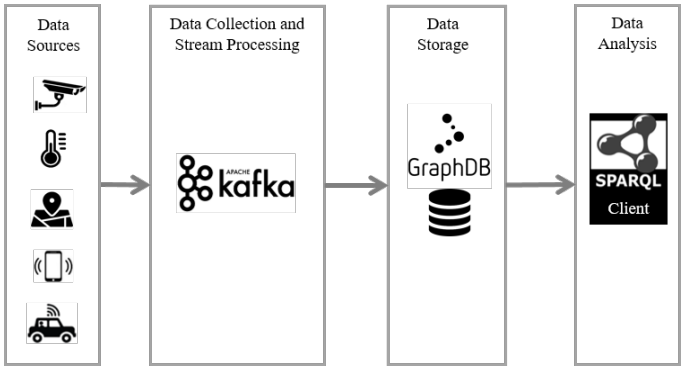

Fig. 1. The SSP pipeline [16]

descriptions of the ingested data, the streaming data previously collected is continuously annotated using the SSN ontology [8]. The annotated sensor streams are then persisted in a triplestore for later analysis. We chose GraphDB to store the RDF streams because it is one of the popular RDF stores, delivering good performance for triplestore initialization, loading and indexing [4]. The RDF streams are then retrieved and manipulated using SPARQL query language.

4 Introductory results

The performance of the deployed SSP pipeline depends heavily on the software and hardware settings. All the tests were carried out on Ubuntu 16.04.4 LTS x64-based PC with Intel(R) Core(TM) i7-7500U, 2.70 GHz CPU, 8 GB RAM and Java 1.8. We deployed the Confluent streaming platform 4.0.0 with Apache Kafka 1.0.0 and Apache Zookeeper 3.4.10, which were the latest versions available at the time of building the SSP. For RDF parsing of the JSON streams we used RDFLlib 4.2.2 package with SPARQL 1.1 implementation. To store the annotated streams we used GraphDB SE 8.7.

To identify the proper place for semantic annotation of the raw sensor data, we implemented and evaluated two different architectures - as indicated in Fig. 2:

- *Semantic annotation on consumer side* (S1). The semantic annotation task is performed after sensor streams are written into topics.
- *Semantic annotation on producer side* (S2). The semantic enrichment task is performed directly on edge sensors.

In both scenarios, the Kafka cluster was configured on a single machine, with multiple brokers and one Zookeeper instance. The producers run in their own thread during 10 minutes and simultaneously publish the data streams to multi-partition topics. The timing of the produced data streams follows a Poisson process with a data rate that varies depending on the experiment.

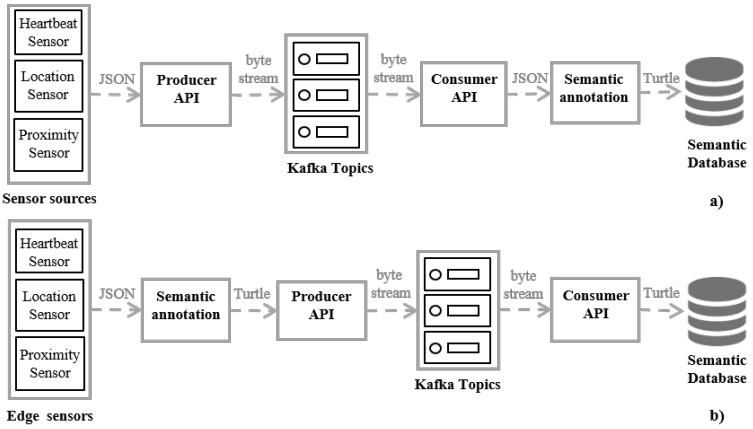

Fig. 2. a) Semantic annotation on consumer side (S1) vs. b) semantic annotation on the producer side (S2)

Table 1. Annotated messages and corresponding RDF triples in both architectures

	S1		S2		
Producers	**Messages**	**RDF triples**	**Messages**	**RDF triples**	**Throughput multiplication**[1]
10	529	8993	5703	96951	10.78
20	558	9486	6253	106301	11.21
30	696	11832	6207	105519	8.92
40	579	9843	6023	102391	10.40
50	357	6069	6043	102731	16.93
60	652	11084	5714	97138	8.76
70	707	12019	5902	100334	8.35
80	851	14467	5904	100368	6.94
90	751	12767	5911	100487	7.87
100	705	11985	6091	103547	8.64

[1] Throughput multiplication of the number of annotated messages in S2 vs. S1

Table 1 summarizes the number of annotated messages and their corresponding RDF triples, in the alternative architectures. We notice that when the semantic processing is moved at the edge of the network (S2), even ten times more annotated messages are generated and implicitly, the number of corresponding RDF triples it is about ten times larger. In the semantic edge architecture (S2), the data does not waste time to travel through the pipeline. Thus, the edge computing architecture for semantic modeling of sensor streams provides advantages in terms of data processing power and network bandwidth reduction.

5 Conclusions

When building the low-latency pipeline for streaming analysis of sensor data, we adopted the Design Science for investigating the proper place to execute the

semantic modeling of sensor data using the SSN ontology. We implemented two alternative architectures of the semantic stream processing system in the context of a smart factory. Experiments focus on assessing the performance of the alternative architectures and results show that the semantic edge architecture is capable of producing an increased number of annotated sensor streams. Performing the semantic enrichment of sensor data locally, not only reduces the physical distance the data travel through the semantic pipeline, but has the potential to improve the IWN's performance by reducing the latency.

References

1. Apache Kafka, https://kafka.apache.org, last accessed 22 May 2020
2. Apache Zookeeper, https://zookeeper.apache.org/, last accessed 22 May 2020
3. GraphDB, graphdb.ontotext.com/, last accessed 22 May 2020
4. Bellini, P., Nesi, P.: Performance assessment of rdf graph databases for smart city services. Journal of Visual Languages & Computing **45**, 24–38 (2018)
5. Chen, J., Lécué, F., Pan, J.Z., Chen, H.: Learning from ontology streams with semantic concept drift. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 957–963. IJCAI'17, AAAI Press (2017)
6. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, 25 February 2014, https://www.w3.org/TR/rdf11-concepts/, last accessed 22 May 2020
7. Dell'Aglio, D., Della Valle, E., van Harmelen, F., Bernstein, A.: Stream reasoning: A survey and outlook. Data Science **1**(1-2), 59–83 (2017)
8. Haller, A., Janowicz, K., Cox, S., Le-Phuoc, D., Taylor, K., Lefrançois, M.: Semantic Sensor Network Ontology. W3C Recommendation, 19 October 2017, https://www.w3.org/TR//vocab-ssn/, last accessed 22 May 2020
9. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Aroyo, L. et al. (ed.) The Semantic Web – ISWC 2011. pp. 370–388. Springer (2011)
10. Le-Phuoc, D., Manfred, H.: Semantic stream processing. In: Sakr, S., Zomaya, A. (eds.) Encyclopedia of Big Data Technologies. Springer (2019)
11. Li, X., Li, D., Wan, J., Vasilakos, A.V., Lai, C.F., Wang, S.: A review of industrial wireless networks in the context of industry 4.0. Wireless networks **23**(1), 23–41 (2017)
12. Narkhede, N., Shapira, G., Palino, T.: Kafka: the definitive guide: real-time data and stream processing at scale. O'Reilly Media, Inc. (2017)
13. Pacha, S., Murugan, S., Sethukarasi, R.: Semantic annotation of summarized sensor data stream for effective query processing. Journal of Supercomputing **76**(6), 4017–4039 (2020)
14. Ren, X., Curé, O.: Strider: A hybrid adaptive distributed rdf stream processing engine. In: d'Amato, Claudia et al. (ed.) The Semantic Web – ISWC 2017. pp. 559–576. Springer (2017)
15. Wieringa, R.J.: Design science methodology for information systems and software engineering. Springer (2014)
16. Zalhan, P.G., Silaghi, G.C., Buchmann, R.A.: Marrying big data with smart data in sensor stream processing. In: Siarheyeva, A. et al. (ed.) 28th Intl. Conf. on Information Systems Development (ISD2019). AIS eLibrary (2019)