Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 353

Big Data Architecture for the Semantic Analysis of Complex Events in Manufacturing

Marco F. Huber[1], Martin Voigt[2] and Axel-Cyrille Ngonga Ngomo[3]

Abstract: Today's production processes are monitored densely via a myriad of sensors. Appropriately processed, the data carries the potential to further increase automation, to early detect failures, and thus, to reduce costs. However, these opportunities can only be exploited if the storage and processing system is capable to deal with massive data in (near) real-time. The distributed and cloud-based architecture proposed in this paper addresses the needs of the manufacturing and plant industry in terms of the processing of a huge amount of complex event data from process monitoring. It combines machine learning and semantic technologies, which allows not only the automatic detection of process failures and their root causes, it also renders these findings in a human-interpretable way thanks to the semantification. By means of a real-world use case, where large-scale printing machines are monitored, we can demonstrate the capabilities of the proposed architecture.

Keywords: Industry 4.0, Semantic Web, Big Data, Machine Learning, Root-Cause Analysis

1 Introduction

Manufacturing and plant engineering is the economically strongest industry sector in Germany with an annual turnover of 212 billion Euros in 2014 and 218 billion Euros in 2015 and more than one million employees [VDM16]. Its success originates from increasingly sophisticated production machines that facilitate the execution of previously manual tasks in an automated and safe manner. To determine the current operational state with high precision, these machines are armed with a plurality of sensors. Similar to modern airliners state-of-the-art manufacturing plants like sheetfed printing machines or injection molding machines produce several gigabytes or even terabytes of sensor data per day. However, manufacturers are increasingly challenged by managing the incoming flood of data and by obtaining insights from the data in order to react timely on events like failures of the production processes. According to [MCB+11], the advent of methods and technologies for analyzing this so-called big data affords the opportunity to reduce production costs by up to 50%. Big data analytics are also considered as one of the key drivers of the forth industrial revolution, also known as Industry 4.0 [LKY14].

In order to gain these benefits, a data value chain needs to be applied. This chain commonly comprises the following steps: 1) data acquisition, 2) data analysis, 3) data curation, 4) data storage, and 5) data usage. For their implementation, very general architecture principles

[1] USU Software AG, Rüppurrer Str. 1, 76137 Karlsruhe, Germany, marco.huber@ieee.org
[2] Ontos GmbH, Wurzner Str. 154a, 04318 Leipzig, martin.voigt@ontos.com
[3] University of Leipzig, Augustusplatz 10, 04109 Leipzig, Germany, ngonga@informatik.uni-leipzig.de

such as the Lambda[4] and Kappa[5] architectures are available. However, none of them fits perfectly for the scenario of event processing from plants, which requires both data analytics being scalable in terms of data volume and velocity as well as human-interpretable results. The main reason besides the continuously evolving zoo of big data technologies[6] is the usage of standards and technologies from the Semantic Web domain, e. g., RDF[7], SPARQL[8], and triple stores. Thus, in order to setup our data value chain, we need to fulfill two objectives regarding the architecture: 1) Combine general big data technologies like Apache Spark[9] with the Semantic Web stack, and 2) enhance the Semantic Web-based tools to handle big data in their volume, velocity and variety.

We introduce a big data architecture for the analysis of event data from manufacturing processes. This architecture is currently being developed within the SAKE project[10], which is funded by the German Federal Ministry for Economic Affairs and Energy (BMWi). The main goals of the project and the architecture are to facilitate the timely detection and data-driven prediction of failures from event data. To cope with the potentially large amount of data, the architecture utilizes state-of-the-art distributed cloud-based big data technologies such as NoSQL databases for data storage and Apache Spark for data analysis and machine learning. In contrast to existing solutions, data storage and machine learning are compliant with Semantic Web standards. By these means, the following problems of processing big data identified in [vKRS+13] and being relevant for the manufacturing and plant industry can be solved: 1) ensuring unique data semantics, 2) scalable machine learning, and 3) generation of human-interpretable analytics.

The paper is structured as follows. In Sec. 2, we elaborate on the design goals being addressed by the proposed architecture. Furthermore, an architecture overview is given. The applicability of this architecture is demonstrated by means of a real-world manufacturing use case in Sec. 3. Conclusions and an outlook to future work complete the paper.

2 Architecture Overview

The guiding theme behind the development of the SAKE architecture can be phrased as:

> *Facilitate analytics for big data streams in such a way that the end user can understand the results and can reproduce their creation.*

Under consideration of the aforementioned data value chain, this theme resulted in the modular architecture depicted in Fig. 1. The architecture consists of the following three layers: The *acquisition layer* is mainly based on Semantic Web technologies and, thus, lays the foundation for human interpretability. Link discovery and structured machine learning are part of the *analytics layer* and allow for the automatic detection of patterns, relations, and failures in event data streams across data sources. The most visible layer for the end users, i. e., the *application layer*, provides intuitive user interfaces and dashboards for a straightforward and gradual exploration and analysis of data.

[4] http://lambda-architecture.net/ [5] https://www.oreilly.com/ideas/questioning-the-lambda-architecture
[6] https://www.linkedin.com/pulse/100-open-source-big-data-architecture-papers-anil-madan
[7] https://www.w3.org/TR/rdf11-concepts/ [8] https://www.w3.org/TR/sparql11-overview/
[9] http://spark.apache.org/ [10] www.sake-projekt.de

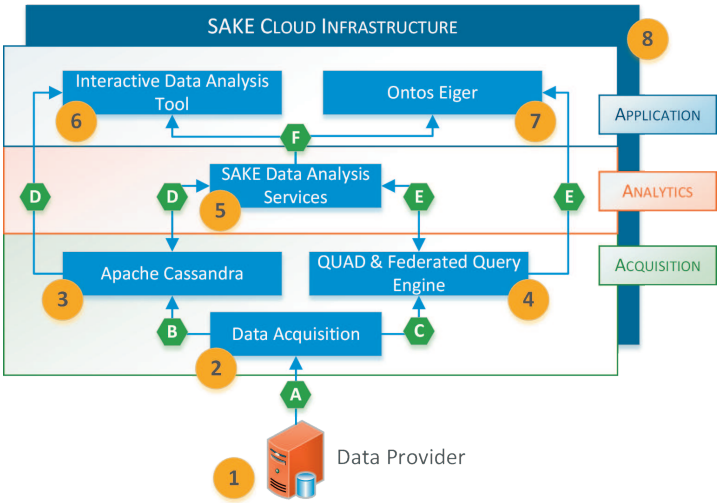

Fig. 1: High-level overview of the SAKE architecture.

These layers run in a distributed manner in the cloud (see Fig. 1-⑧). This is a necessary design decision given the large amount of data common in today's manufacturing processes. In order to stream the data into the platform (Fig. 1-Ⓐ), we rely on Apache Kafka[11]. In the platform, the communication between all components is in general flexible and loosely coupled by using the REST paradigm. The semantic RDF data is written and queried by using SPARQL (Fig. 1-Ⓒ, Ⓔ & Ⓕ) whilst the query language CQL[12] is used for Cassandra (Fig. 1-Ⓑ & Ⓓ). We also developed a novel SPARQL connector[13] for Apache Spark that eases the querying and usage of RDF data in the analytics layer.

In the following, we give a detailed description of the individual layers and their modules.

2.1 Acquisition Layer

The acquisition layer builds the interface between the data provider (Fig. 1-①) and the SAKE system. It carries out the explication of all required RDF for the further processing and ensures its availability to subsequent modules. To achieve this goal, it implements a three-step process: The *preprocessing* contains all data-cleansing steps to prepare the input before distributing it into the cluster. The *parsing* contains several steps that are required to extract the RDF statements required from the input. Finally, the *storage* stage allows to persist the generated model.

Preprocessing Plant machines commonly provide data in two different formats: 1) as compressed XML/CSV files and 2) as data streams. For the first case, tests revealed that

[11] http://kafka.apache.org/

[12] https://cassandra.apache.org/doc/cql3/CQL-2.2.html

[13] https://github.com/USU-Research/spark-sparql-connector

working with compressed and XML files in current distributed frameworks like Apache Spark is not straight-forward, unstable and inefficient because they generally work with line-based input files. For working with data streams, we rely on Apache Kafka to collect and prepare the data on-the-fly. The preprocessing of the data comprises the following tasks: 1) getting or receiving the data from the remote host, 2) uncompress the data if required, 3) transform the data to line-based formats, and 4) ensure UTF-8-conformity.

Parsing In the next step, the data from preprocessing is parsed into a generic event model. It is based on the SAKE event ontology, which formalizes generic concepts, e. g., *Event*, *Timeline* and their time-related relations, as well as domain specific context information, e. g., modules of the printing machines. The workload could be easily parallelized and thus, distributed in a data processing cluster like Apache Spark. The order of the events is kept by their timestamps and provenance ID. Each node in the cluster parses a subset of the input into an in-memory model ,which is finally used to create the required output. SAKE provide different parsers, e. g., for XML or CSV, which can be extended for specific use cases through an extensible plug-in mechanism. This core step of the data acquisition requires the most knowledge from domain experts.

Storage In the last step, the generated data model is stored in the data sinks. As shown in Fig. 1, we currently support two different storage solutions: Apache Cassandra ③ is especially used to store numeric data whereas an RDF store (QUAD) ④ is our premiere data sink. The latter is used in order to store all data alongside the defined SAKE ontology. Due to the enormous data size (multiple terabytes) the QUAD instances [PPD+13] are distributed in the cluster. For an efficient access to the data, we developed a concept of federated query engine (Fig. 1-④) with a cost-based query optimization.

2.2 Analytics Layer

The analytics layer aims to detect failures and anomalies in the event data. For this purpose, machine learning is employed, where training data comprising events of a regular execution of the manufacturing process are used for learning a probabilistic "normal model" for instance via kernel density estimation. A significant deviation of events of the incoming data stream from this model allows identifying an anomalous situation.

Detecting anomalous behavior is merely a first step. To avoid future failures, it is important to understand the causes of failures for effective counter measures. Accordingly, methods for the provision of drill-downs to root cause become necessary. Such analyses employ structured machine learning, where we learn classifiers from labeled event sequences. The steps involved to achieve this goal are:

Link Discovery First, we link events within and across machine data streams. Therewith, we aim to ensure that the subsequent machine learning process can learn complex OWL class expressions. Given that events can be regarded as entities that exist within a certain time interval, we rely on 1) the functionality offered by the LIMES framework for link discovery [Ngo12], 2) a reduction of Allen's algebra [All83] to atomic relations to achieve

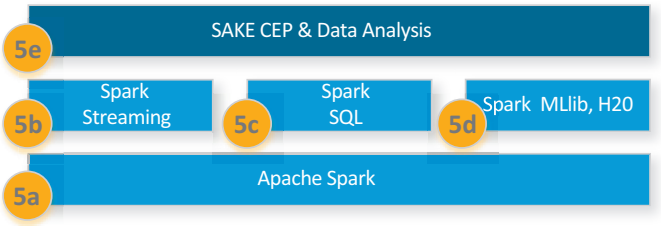

Fig. 2: Detailed view of the SAKE Data Analysis Services based on Apache Spark.

highly scalable link discovery and 3) an efficient planner based on [Ngo14] to scale up to linking tens of thousands of events per minute and processor kernel.

Structured Machine Learning The results of the acquisition and the link discovery are a network of linked events. We learn event descriptions by using a refinement-operator-based approach [Leh09]. The idea here is to start with positive and negative examples for anomalous events and to explore the RDF network around these events (including related events) to find the most generic network patterns that describe the positive examples and do not describe the negative examples.

The SAKE Data Analysis Services Fig. 1-(5), which form the core module of the analytics layer, are depicted in more detail in Fig. 2. The aforementioned anomaly detection and root-cause analysis capabilities are part of the the SAKE CEP & Data Analysis module (5e), where CEP stands for complex event processing and facilitates the application of analytics like anomaly detection on streaming event data. This module sits on top of an Apache Spark stack (5a)–(5d). Apache Spark is an open-source cluster computing framework. Similar to MapReduce [DG08], it provides a simplified programming interface for performing distributed computing. In contrast to MapReduce, Spark is especially well suited for iterative algorithms, which are common in machine learning.

While the Spark core (5a) is designed for batch processing, our analytics also need to process event streams. Thus, we utilize the Spark Streaming module (5b) in addition. Spark SQL (5c) supports processing of structured data via a table-like data format called DataFrame, which is preferred by several implementations of machine learning algorithms. These are part of module (5d), which comprises the machine learning libraries Spark MLlib and H2O[14].

2.3 Application Layer

At the top layer of the SAKE architecture we combine automatic verbalization techniques with application specific dashboards and reporting tools. The Interactive Data Analysis Tools Fig. 1-(6) essentially provide web applications that are customized to the needs of the end user. They provide functionality for data exploration, dashboards for live process monitoring, and reporting. Therefore, we employ and extend notebook environments like

[14] http://www.h2o.ai

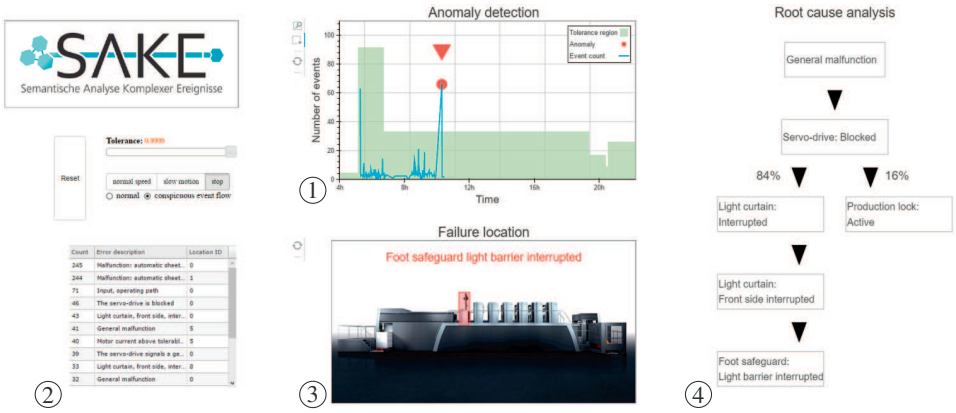

Fig. 3: Web application for analyzing anomalies of printing machines. ① Anomaly detection in event streams, ② event statistics, ③ visualization of failure location, ④ root-cause analysis.

Jupyter[15] and JavaScript libraries like D3[16] or Bokeh[17]. This combination turned out to be especially suitable for supporting explorative analysis and interactive visualization. Furthermore, we use the extendable, web-based Linked Data suite Ontos Eiger (Fig. 1-⑦) in order to create user-driver dashboards for the semantic data. Depending on the given use case, domain experts but also managers can easily add and configure required widgets to visualize and understand the outcome of the analytics services.

One of the outcomes of these services, especially the structured machine learning, are OWL Class expressions[18]. Given that most users of the SAKE platform are not versed in Semantic Web technologies, we provide a verbalization framework for OWL Class expressions based SPARQL2NL framework [NNBU+13]: SemWeb2NL[19] uses a bottom-up approach to verbalize expressions. Atomic expressions are verbalized using verbalization patterns. Complex expressions are verbalized by combining the results of the verbalization of the corresponding atoms via linguistic rules. These verbalizations can then be used in the user dashboards.

3 Use Case: Processing Events from Printing Machines

Besides the design and development of a big data architecture, the SAKE project also aims for applying the architecture to real-world manufacturing processes. In this section, we describe the current status of a use case, in which the SAKE architecture is used for monitoring event data of a sheetfed printing machine of Heidelberger Druckmaschinen AG (HDM)[20]. HDM is a project member and the considered printing machine depicted in Fig. 3-③ is running at the printing facility of one of HDM's customers.

[15] http://jupyter.org [16] http://d3js.org [17] http://bokeh.pydata.org/ [18] https://www.w3.org/TR/owl2-syntax/
[19] https://github.com/AKSW/SemWeb2NL/ [20] http://www.heidelberg.com/

On average, printing machines of this type produce approximately four million events per day, where events are represented in XML and comprise for instance the status of the current printing job, safety issues, sensor readings, alerts, etc. Based on these events, we are interested in detecting anomalies leading to an unplanned reboot of the printing machine. Given this use case, the layers of the SAKE architecture and its modules are customized as follows:

Acquisition layer The XML-based event data is pushed into the SAKE architecture and processed as explained in Sec. 2.1. Since the causes for the anomalies is unknown, all available events are extracted and "semantified". By also obtaining explicit information from plain text log messages, the final data size grows by a factor of eight compared to the size of the plain log files.

Analytics layer The first step is the link discovery across the events generated by the machine. We generate links according to the first seven relations of Allen's algebra (all other relations can be derived from them). The resulting network of RDF resources and a set of examples (positive and negative) are then forwarded to the DL-Learner. The refinement operator underlying CELOE [LABT11] is then used to derive the corresponding OWL Class expression. For anomaly detection historic events form a training set for learning a "normal model" of the printing machine. Metrics on the frequency of the incoming event stream are then compared against this model.

Application layer We developed a Web application based on Bokeh, which among others visualizes the incoming event stream and detected anomalies (Fig. 3-①). For anomalous situations, the web frontend provides information about all events in a certain time window (Fig. 3-②), the location of the machine part probably causing the anomaly (Fig. 3-③), and linked events that are considered likely to trace back to the root-cause (Fig. 3-④).

4 Conclusion and Future Work

The proposed SAKE architecture reflects the needs of manufacturing and plant engineering in the Industry 4.0 era by combining the latest technology for analyzing big data streams with Semantic Web standards. In doing so, it enables scalable machine learning together with human comprehensible results. Thanks to its modular structure, the architecture can be customized easily to a given manufacturing process as was shown exemplary for the monitoring of a large-scale printing machine.

While the basic design of the architecture is considered complete, there is still room for further developments on the module level. For instance, we plan to exploit unsupervised or at least semi-supervised machine learning for root-cause analysis in order to minimize the effort of event labeling. We also plan to extend our works on large-scale link discovery [NH16], planning [Ngo14] and machine learning [Leh09] to big data processing frameworks such as Spark. Finally, we plan to apply the SAKE architecture also to use cases aside the manufacturing industry like in IT monitoring in order to identify and close gaps towards a more universal architecture for the semantics analysis of complex events.

Acknowledgement

This work was partially supported by the BMWi project SAKE (Grant No. 01MD15006).

References

[All83] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[DG08] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Magazine Communications of the ACM*, 51(1):107–113, January 2008.

[LABT11] Jens Lehmann, Sören Auer, Lorenz Bühmann, and Sebastian Tramp. Class expression learning for ontology engineering. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(1):71–81, 2011.

[Leh09] Jens Lehmann. DL-Learner: learning concepts in description logics. *The Journal of Machine Learning Research*, 10:2639–2642, 2009.

[LKY14] Jay Lee, Hung-An Kao, and Shanhu Yang. Service Innovation and Smart Analytics for Industry 4.0 and Big Data Environment. *Procedia CIRP*, 16:3–8, 2014.

[MCB+11] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. Big data: The next frontier for innovation, competition, and productivity. Report, McKinsey Global Institute, June 2011.

[Ngo12] Axel-Cyrille Ngonga Ngomo. On link discovery using a hybrid approach. *Journal on Data Semantics*, 1(4):203–217, 2012.

[Ngo14] Axel-Cyrille Ngonga Ngomo. Helios–execution optimization for link discovery. In *The Semantic Web–ISWC 2014*, pages 17–32. Springer, 2014.

[NH16] Axel-Cyrille Ngonga Ngomo and Mofeed Hassan. The Lazy Traveling Salesman – Memory Management for Large-Scale Link Discovery. In *Proceedings of the Extended Semantic Web Conference*, 2016.

[NNBU+13] Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. Sorry, i don't speak SPARQL: translating SPARQL queries into natural language. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 977–988, 2013.

[PPD+13] Alexander Potocki, Anton Polukhin, Grigory Drobyazko, Daniel Hladky, Victor Klintsov, and Jörg Unbehauen. OntoQuad: Native High-Speed RDF DBMS for Semantic Web. In *Knowledge Engineering and the Semantic Web*, volume 394, pages 117–131. Springer Berlin Heidelberg, 2013.

[VDM16] Maschinenbau in Zahl und Bild. Annual report, VDMA, 2016.

[vKRS+13] Tim van Kasteren, Herman Ravkin, Martin Strohbach, Mario Lischka, Miguel Tinte, Tomas Pariente, Tilman Becker, Axel Ngonga, Klaus Lyko, Sebastian Hellmann, Mohamed Morsey, Philipp Frischmuth, Ivan Ermilov, Michael Martin, Amrapali Zaveri, Sarven Capadisli, Edward Curry, Andre Freitas, Nur Aini Rakhmawati, Umair ul Hassan, Aftab Iqbal, Anna Karpinska, Syzymon Danielczyk, Pablo Mendes, John Domingue, Anna Fensel, and Andreas Thalhammer. Consolidated Technical Whie Papers. Project report, Big Data Public Private Forum (BIG), 2013.