Journal of Manufacturing Systems 54 (2020) 138–151

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

Technical Paper

Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case

Radhya Sahal*, John G. Breslin, Muhammad Intizar Ali

CONFIRM SFI Research Centre for Smart Manufacturing, National University of Ireland Galway, Ireland

ARTICLE INFO

Keywords:
Industry 4.0
Big Data
Stream processing
Predictive maintenance
Railway
Wind turbines

ABSTRACT

Industry 4.0 is considered to be the fourth industrial revolution introducing a new paradigm of digital, autonomous, and decentralized control for manufacturing systems. Two key objectives for Industry 4.0 applications are to guarantee maximum uptime throughout the production chain and to increase productivity while reducing production cost. As the data-driven economy evolves, enterprises have started to utilize big data techniques to achieve these objectives. Big data and IoT technologies are playing a pivotal role in building data-oriented applications such as predictive maintenance.

In this paper, we use a systematic methodology to review the strengths and weaknesses of existing open-source technologies for big data and stream processing to establish their usage for Industry 4.0 use cases. We identified a set of requirements for the two selected use cases of predictive maintenance in the areas of rail transportation and wind energy. We conducted a breadth-first mapping of predictive maintenance use-case requirements to the capabilities of big data streaming technologies focusing on open-source tools. Based on our research, we propose some optimal combinations of open-source big data technologies for our selected use cases.

1. Introduction

Industry 4.0 is relatively a new term being hailed as the fourth generation of the industrial revolution. It covers a broad range of technologies, processes, and systems mainly related to the digitalization of industry. In terms of data-related technologies, the main areas of Industry 4.0 are (i) *Cyber Physical Systems (CPS)*, (ii) *Industrial Internet of Things (IIoT)*, (iii) *Cloud Solutions & Decentralized Services*, and (iv) *Big Data & Stream Processing* technologies for processing large amounts of production data in real time [1,2]. In terms of use case scenarios, Industry 4.0 use cases are mainly categorized into the following three areas, namely; (i) *intelligent products*, (ii) *intelligent processes*, and (iii) *intelligent machines*. For intelligent products, Industry 4.0 applications manage all necessary information about the products and the production processes conducted on them. Intelligent processes focus on the results and consequences of product creation, including asset information management, etc. [3]. Intelligent machine scenarios focus on industrial machinery performance and applications such as predicting breakdowns, detecting any quality issues, and the need for conducting preemptive maintenance. However, the most prominent use case for intelligent machines is predictive maintenance (often referred to as PM or PM 4.0).

PM applications predict failure sufficiently ahead of time so that decision makers can take appropriate actions such as maintenance, replacement or even a planned shutdown. These applications facilitate savings on machine maintenance and increase productivity by ensuring the maximum uptime of machines. Mostly, the manufacturing processes follow assembly line production, thus any failure in the assembly line results in a domino effect, making it crucial to avoid any point of failure within the assembly line. By deploying predictive maintenance solutions, these failures can be avoided or at least reduced. However, for the most accurate and optimal prediction, it is of the utmost necessity to collect and analyze large amounts of relevant data within a reasonable time frame [4,5]. Consequently, big data analytics and stream processing technologies are a key requirement for predictive maintenance solutions [6,7].

Predictive maintenance applications are considered one of the crucial data-driven analytical applications for large-scale manufacturing industries. Considering use cases in the area of predictive maintenance, we identify the requirements for a big data processing pipeline in the different phases of data processing such as data collection, analytics, querying, and storage. We mapped these requirements to the capabilities of open-source technologies for big data and stream processing such as distributed queuing management, big data stream processing platforms, big data storage technologies and streaming SQL engines (see Fig. 1). Distributed queuing management technologies (e.g. Apache

* Corresponding author.
E-mail addresses: radhya.sahal@nuigalway.ie (R. Sahal), john.breslin@nuigalway.ie (J.G. Breslin), ali.intizar@nuigalway.ie (M.I. Ali).

https://doi.org/10.1016/j.jmsy.2019.11.004
Received 1 March 2019; Received in revised form 4 November 2019; Accepted 6 November 2019
Available online 19 December 2019

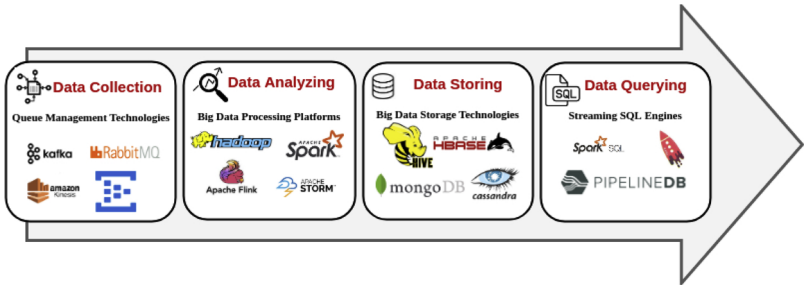

Fig. 1. Open-source big data pipeline analysis technologies.

Kafka, RabbitMQ, Amazon Kinesis, Microsoft Event Hubs and Google Pub/Sub) allow producers (i.e., data producers such as sensors) to push a high volume of messages to a queue and allow consumers (i.e., Industry 4.0 applications) to pull messages from the queue in real time with scalability and fault tolerance capabilities. The main big data stream processing platforms (e.g., Apache Storm, Apache Samza, and Apache Flink) handle distributed streams and batch data processing. Big data storage technologies including column-based storage (e.g., Cassandra and HBase), document stores (e.g., MongoDB) and Hadoop-based frameworks (e.g., Hive) are open-source technologies for storing, querying and analyzing large data sets. Streaming SQL engines are query languages that extend SQL with the ability to process real-time data streams (e.g., Spark SQL, Flink Table API, KSQL, SamzaSQL, StormSQL, and StormCQL, etc.).

In our study we produce a practical literature review combined with the functional requirements of predictive maintenance use cases. The main contributions in this paper are as follows:

- Explore open-source big data technologies, including big data streaming processing platforms, distributed message queue management systems, big data storage platforms, and streaming SQL engines.
- Identify big data technology related requirements for the PM use case to reduce operation and maintenance (O&M) costs.
- Identify functional requirements and operational characteristics for two Industry 4.0 sample use cases related to the railway transportation and wind turbine energy industries.
- Provide a mapping between the use cases' requirements and available technologies by combining different big data and stream processing technologies to design and deploy the selected use case.

The rest of this paper is organized as follows; The research motivation and methodology are presented in Section 2. The big data stream analytics technologies are described in Section 3. The predictive maintenance use cases and their requirements are presented in Section 4. The mapping between the functionalities of big data and stream processing technologies and the requirements of the selected predictive maintenance use cases is presented in Section 5. We further discuss existing work on big data and its suitability for Industry 4.0 by validating our proposed technologies in Section 6, and then we conclude our work in Section 7.

2. Research motivation and methodology

In this section, we provide our motivation for this work and present a set of research questions used to define our research methodology.

2.1. Research motivation

Our main motivation for conducting this research work is to bridge the gap between industry use case developers and open-source big data streaming technologies. We believe this work will act as a guideline for Industry 4.0 use case developers to make better decisions when choosing relevant open-source technologies before designing a predictive maintenance use case based on their requirements.

During our recent experience of working for a large research centre focused on the topic on smart manufacturing,[1] having over 42 industry partners including multinational corporations as well as SMEs, it was quickly realized that there is a strong need to provide a set of guidelines to decision-makers before selecting the most suitable technologies to address their requirements. Each industry partner has a different set of requirements, even for the same use case. Hence, this study provides a set of guidelines for our industry partners to help them choose the correct underlying technology before implementing any use case.

2.2. Research questions

Identifying a set of predictive maintenance requirements for a set of use cases can provide guidelines on how to use existing big data and streaming technologies for designing predictive maintenance use cases. Accordingly, this research explores the capabilities of open-source big data technologies that can improve the performance of decision-making processes within manufacturing. These open-source big data technologies provide an opportunity to fully explain the state of a manufacturing process and utilize the intelligence of data processing and predictive maintenance analytics algorithms.

In particular the goal of this paper is to assist data engineers in designing big data analysis pipelines for Industry 4.0. This is achieved by investigating the following research questions:

RQ1: What are the strengths and weaknesses of the available open-source big data and streaming data analysis technologies?

The purpose of this question is to explore the commonly used open-source big data and streaming data analysis technologies that are focusing on Industry 4.0. To address this question, a systematic methodology is provided to review the strengths and weaknesses of the existing technologies with a view to establish their usage for Industry 4.0 (see Section 3).

RQ2: What are the requirements for building a data analytics pipeline focusing on predictive maintenance use cases?

The purpose of this question is to identify the technical requirements for big data techniques, to understand how data is currently being generated from the factory floor, and then to find out how to process and analyze this data to produce insightful knowledge for decision making. To address this question, we categorize the requirements according to the typical data analysis pipeline application phases including data transmission (publishing and subscription), data processing, data querying, and data storage (see Section 4).

RQ3: How can the technical requirements for the use cases be mapped onto existing open-source big data technologies?

The

[1] http://confirm.ie/.

purpose of this question is to elaborate on the matchmaking between the requirements of the use cases and the capabilities of open-source big data technologies. This mapping will help us to recommend a concrete design/implementation for predictive maintenance applications. We conducted a breadth-first mapping of predictive maintenance requirements for two real-world use cases (i.e., railways and wind turbines) to the strengths and weaknesses of open-source big data streaming technologies (see Section 5).

3. Open source big data technologies and their suitability for Industry 4.0

In this section, the first research question, RQ1, is addressed.

RQ1: What are the strengths and weaknesses of the available open-source big data and streaming data analysis technologies?

Technological developments in automation systems for Industry 4.0 have brought many advantages and opportunities in terms of flexibility, economic production, and the optimization of production processes [8,9]. In particular, manufacturers have started to combine data from different collaborative systems to drive productivity in the design, production, and delivery of products [10]. For example, manufacturing firms such as Raytheon Corp. have developed smart factories which are based on the powerful capacity of handling big data from different sources, e.g., instruments, sensors, Internet transactions, CAD models, and digital records to enable the real-time control of multiple elements within the production process [11].

Indeed, handling manufacturing data is a big data handling problem [1]. Consequently, a deeper understanding of the strengths and weaknesses of the state-of-the-art big data processing technologies is necessary to truly realize fully automated large-scale manufacturing systems. In this paper, a technical overview of the state-of-the-art in big data processing technologies is given under four main headings: distributed queuing management technologies; big data stream processing platforms; big data storage technologies; and streaming SQL engines.

3.1. Distributed queuing management technologies

Distributed queuing management technologies such as Kafka,[2] RabbitMQ,[3] Amazon Kinesis,[4] Microsoft Event Hubs[5] and Google Pub/Sub [6] have matured in the last few years to support publish/subscribe messaging. These technologies have added some useful new forms of solutions when moving large-scale data around for real-time applications. While distributed queuing management technologies may seem very similar to traditional message queuing technologies, they differ significantly in their architecture and therefore have very different performance and behavioral characteristics. For example, traditional queuing systems remove processed messages from the queue and cannot scale out with multiple consumers taking multiple independent actions on the same event. In contrast, distributed queuing technologies are suitable for both offline and online message consumption by supporting a group of consumers and preventing data loss by using persistent disks over replicated clusters. The messages are persisted immediately to the distributed queues to guarantee the delivery of messages for a period of time. In particular, each distributed queuing management technology shards its topics (i.e., a topic is where data (messages) gets published by a producer and will be pulled by consumer(s)) into one or more partitions. Then, each consumer group consumes the messages of a given topic (its partitions) where each partition cannot be consumed by more than one consumer of the same consumer group. The consumer groups feature is very important to perform re-balancing whenever partitions and/or consumers change.

Table 1 shows a number of features that should be considered when choosing a distributed queuing technology including messaging guarantee, ordering guarantee, consumer groups, disaster recovery, replication, federated queues (i.e., providing a way of balancing the load of a single queue across nodes or clusters), and a list of supported query languages. For the messaging *delivering guarantee* feature, three types of configurations are defined:

1 **At most once**: some messages may be lost, and no message is delivered more than once.
2 **Precisely once**: each message is guaranteed to be delivered once only, not more or less.
3 **At least once**: each message is guaranteed to be delivered, but may in some cases be delivered multiple times.

The *ordering guarantee* feature defines five possible options including:

Table 1
List of distributed queuing management technologies.

Feature	Technology				
	Apache Kafka	RabbitMQ	Amazon Kinesis	Ms Azure Event Hub	Google pub/sub
Delivering guarantees	Yes At least once	Yes At least once	Yes At least once	Yes At least once	Yes At least once
Ordering guarantees	Guaranteed within a partition	Guaranteed using AMQP channel	Guaranteed within a partition	Guaranteed within a partition	No order guarantees
Consumer groups	Yes	Yes	Yes	Yes	Yes
Disaster recovery	Yes	Yes	Yes	Yes	Yes
Replication	Configurable replicas	Configurable replicas	Hidden (across three zones)	Configurable replicas	Hidden
Federated queues	No	Yes	No	No	No
Language supported	Java, Go, C++, Python, .NET, Node.js, PHP, Ruby	Java, Go, C/C++, Python, .NET, Node.js, PHP, Ruby	Java, Go, C++, Python, .NET, Node.js, PHP, Ruby	Java, C++, Python, .NET, Node.js, PHP, Ruby	Java, C++, Python, .NET, Node.js, PHP, Ruby

[2] http://kafka.apache.org.
[3] http://www.rabbitmq.com.
[4] http://aws.amazon.com/kinesis.
[5] http://azure.microsoft.com/en-us/services/event-hubs.
[6] http://cloud.google.com/pubsub.

1 **None**: no guarantee of any incoming message.
2 **Within a partition**: guarantee order within any given partition, but across partitions, the messages may be out of order.
3 **Across partitions**: guarantee order across all partitions which slows things down and is expensive to scale.
4 **Within AMQP channel**: guarantee order by using the Advanced Message Queuing Protocol (AMQP) channel which is used for message-oriented middleware to support sending and receiving messages between distributed systems.
5 **Within a shard**: guarantee order within a shard, where sharding is a method to distribute data/messages across multiple different servers to increase the scalability of distributed queuing systems.

The *federated queues* feature provides a mechanism for balancing the load of a single queue across several nodes or clusters to support migration among clusters without stopping all producers and consumers. The federation features are of the utmost necessity for most data applications, particularly, for applications that aggregate data from disparate data sources and make it accessible to data consumers as one integrated data store. Also, many data sources within big data applications in different companies, including finance, healthcare and manufacturing, are producing a large sequence of events. Such data needs to be captured and analyzed in real time to give insights for decision making without any delay. Consequently, distributed queuing management technologies have been adopted to collect large amounts of streaming data, feeding it to real-time systems.

3.2. Big data stream processing platforms

Over the years, traditional relational database management systems and more recently batch processing technologies, such as Hadoop[7] and Spark,[8] have been used to manage and analyze data. These technologies are now well matured and well-suited for a broad range of applications, but are not an ideal choice for building real-time applications. Consequently, a new set of technologies are introduced which are capable of handling large amounts of streaming data, processing and analyzing it on the fly to meet the needs of real-time applications such as Apache Storm,[9] Apache Samza[10] and Apache Flink.[11] . These technologies target the essence of time for real-time analytics, streaming analytics, and Complex Event Processing. They allow organizations to build real-time solutions using IoT and extract information from their big data sources to derive insights from millions of events in minimal time.

Starting with the state-of-art big data platforms, Apache Hadoop is inspired by Google's MapReduce design which is based on batch processing [12]. The current Apache Hadoop ecosystem consists of a Hadoop kernel, MapReduce, Hadoop Distributed File System (HDFS) and several other related projects such as Apache Hive, HBase, and Zookeeper. Apache Spark was developed in 2012 to overcome the MapReduce cluster computing paradigm. It uses the micro batching procedure to perform stream processing by dividing the incoming stream of events into a group of small batches and keeps the latency of stream processing under control. Apache Spark claims to be faster than Hadoop by achieving better performance due to its micro-batch processing. However, collecting events together for processing in batches using Spark streaming processing is still a limiting factor for real-time data analysis. Beyond this, a great variety of other streaming platforms such as Apache Storm, Apache Smaza, and Apache Flink have emerged to introduce the notion of streaming-first systems claiming to be truthful stream processing platforms. These platforms treat batch processing as a special case and do not use micro batching to overcome small-batch problems, and hence are ideally designed for streaming applications [13].

Table 2 depicts a comparison of different big data platforms comparing their capabilities and characteristics. Although, the big data platforms are mainly based on the principle of either batch processing or stream processing model, they also further differ in terms of their architectural components. For instance, Hadoop has three core components which are HDFS, YARN, and MapReduce while Spark has

Table 2
List of big data processing platforms.

Feature	Technology				
	Hadoop	Storm	Samza	Spark	Flink
Components	HDFS YARN MapReduce	Streams Spouts Bolts	Samza API YARN Kafka	Resilient distributed datasets (RDD)	Operators Sources Sinks
Processing model	Batch	Hybrid Stream and Batch	Stream	Hybrid Stream and Batch	Hybrid Stream and Batch
Memory management	Configurable memory management	Configurable memory management	Configurable memory management	Configurable memory management	Automatic memory management
Scalability	Highly scalable	Highly scalable	Highly scalable	Highly scalable	Highly scalable
Latency	High	Low	Low	Low	Low
Recovery	Highly fault-tolerant	Highly fault-tolerant	Checkpointing	Checkpointing RDDs	Checkpointing data sources
Interactive mode	No	No	Interactive SQL shell	Interactive Spark shell	Interactive Scala shell
SQL supported	Apache Hive	StormSQL	SamzaSQL	Spark SQL	Table API
Queuing management systems	Kafka RabbitMQ Kinesis	Kafka RabbitMQ Kinesis Event Hubs	Kafka RabbitMQ Kinesis Event Hubs	Kafka RabbitMQ Kinesis Event Hubs Google Pub/Sub	Kafka RabbitMQ Kinesis Event Hubs

[7] http://hadoop.apache.org.
[8] http://spark.apache.org.
[9] http://storm.apache.org/releases/1.1.2/storm-sql.html.
[10] http://samza.apache.org.
[11] http://flink.apache.org.

Table 3
List of big data storage technologies.

Feature	Technology			
	Hadoop Hive	MongoDB	Cassandra	HBase
Description	Data warehouse software for querying and managing large distributed datasets, built on Hadoop	One of the most popular document stores	Distributed database for managing large amounts structured data	Open-source, distributed, versioned, column-oriented store
Data model for storing	File system	Document-based	Column-based	Column-based
Data scheme	Relational DBMS Schema-on-Reading	Schema-free	Relational DBMS uses Amazon DynamoDB	Relational DBMS uses Google Bigtable
MapReduce key-value	Yes	Yes	Yes	Yes
APIs and other access methods	JDBC ODBC Thrift	Proprietary protocol using JSON	JDBC ODBC	JDBC ODBC
In-memory capabilities	N/A	Yes	Yes	Yes
Partitioning methods	Sharding	Sharding	Key partitioning	Key partitioning
Concurrency	Yes	Yes	Yes	Yes
Architecture model	Master-Slave	Master-Slave	Peer-Peer	Master-Slave
CAP Theorem	Consistency Partition tolerance	Availability Partition tolerance	Consistency Partition tolerance	Consistency Availability

Resilient Distributed Datasets. On the other hand, Flink has different layers which are the deploying layer include YARN, the core layer and API&Libraries such as Table API and FlinkML. The Flink core layer is also known as the distributed streaming dataflow which consists of streaming operators, sources, and sinks. Another important factor is memory management and an increasing number of big data projects are choosing big data platforms according to their memory management mechanism i.e. managing JVM memory whether manually configurable or automatic management style. The fast growth of real-time data makes a key challenge for big data in terms of performing low-latency analysis which means systems respond quickly to actions [14]. For Hadoop, data are physically stored first usually on HDFS and then analyzed. For stream processing, data are not stored but rather directly processed to decrease response latency at the sub-second or even at the millisecond level. In tandem with the low-latency advantage of the big data platforms for real-time applications, the fault tolerance mechanism plays a critical role in streaming performance allowing programs to recover from failure with minimal disruption. The big data platforms are equipped with different recovery mechanisms to guarantee fault tolerance such as HDFS fault tolerance, check-pointing RDDs, and data sources for Hadoop, Spark, and Flink respectively.

Today most applications are designed based on users' experience by using an interactive mode (a.k.a. shell mode) to give users more flexibility. In contrast, the data centers adopt machine language programs based on the batch mode (a.k.a. script mode), which executes operations that are not interactive. Both Apache Spark and Apache Flink support interactive mode using in-memory data processing which probably is limited in case of processing hundreds of terabytes. A very close look on big data platforms indicates that the programming APIs for developing big data applications are often low-level and require substantial customized code and have a substantial initial learning curve and additional maintenance overhead. Consequently, there is a strong need to enhance the capabilities of streaming platforms to support concurrent querying languages allowing developers/users to use multiple querying streams depending on application requirements effectively. Furthermore, big data platforms are capable of dealing with distributed queuing management technologies to subscribe to generated data from different streaming sources.

3.3. Big data storage technologies

Recently, big data analysis systems have been improved to deal with the challenge of large data volume, which grows exponentially. These big data analysis systems typically address the volume challenge by allowing scaling out by adding new nodes to the distributed environment to provide processing units and storage. On the software side, new technologies emerged such as columnar stores e.g., Cassandra[12] and HBase,[13] clever combinations of different storage systems, e.g., Hadoop Distributed File System (HDFS),[14] and documented store such as MongoDB[15] are often more efficient and less expensive. These big data storage technologies use shared-nothing architectures to address storage limitation by horizontally scaling out to new nodes providing extra storage for such massive data growth.

Aforementioned big data storage technologies are compared using characterized criteria such as database model, schema type, processing, transaction, partitioning methods, in-memory capabilities, and concurrency (see Table 3). Data model for storing can be broadly divided into three types: (i) *File System*, e.g., HDFS for Hive Hadoop; data are stored schemaless using HDFS and read in a structured manner at processing time based on the requirements of the processing application which is known as Schema-on-Reading; (ii) *Document-based*, e.g., MongoDB; (iii) *Column-based schema*, e.g., Cassandra, and Hbase. By considering the data schema feature, two types of schema have been identified: (i) *Relational DBMS* which fits with structured data and (ii) *Schema-free* which fits with semi-structured and unstructured data. Typically, classifying big data storage technologies, according to the used schema, gives a clear view to big data application developers to choose the proper technologies according to the nature of their data.

Recently, in-memory data processing is denominating in emerging technologies where the slower disks are replaced by RAM and flash memory. Consequently, we can vary big data storage technologies

[12] http://cassandra.apache.org.
[13] http://hbase.apache.org.
[14] http://hive.apache.org.
[15] http://www.mongodb.com.

according to their capabilities to process data using the in-memory mechanism, especially for critical real-time applications. MongoDB, Cassandra, and HBase are representatives of this mechanism. As per Eric Brewer, father of *CAP theorem* (i.e., Consistency, Availability, and Partition Tolerance), the choice of storage technology is limited to two of three characteristics, which is up to data application requirements [15]. For instance, if the big data application needs data to be consistent among all nodes, columnar relational store such as Cassandra, and HBase is the appropriate big data storage. In consequence, the proper choice of an appropriate big data storage technology depends on the requirement of the application and identified the feature of storage technology.

3.4. Streaming SQL engines

As the data-driven economy evolves, enterprises have come to realize a competitive advantage in being able to act on high volume and high-velocity streams of data. However, the programming API provided by these technologies is often low-level, requiring substantial custom code that adds to the programmer learning curve and maintenance overhead. Particularity, these technologies often lack SQL querying capabilities that have proven popular on big data systems like Hive, Impala, or Presto. On the other hand, some streaming platforms leverage windowing operations by repeatedly iterating over a series of micro-batches, in much the same way as static queries operate over stored data such as Spark. Also, some of the streaming platforms have developed streaming processing mechanisms i.e., Event Stream Processing and Complex Event Processing using different windowing types to execute continuous SQL queries. In consequence, different open-source stream query processing engines are proposed and developed based on windowing concept to support big data technologies

Table 4
List of streaming SQL engines.

Feature	Technology						
	Description	Window types	Query types	Platforms	Processing model	Queuing technology	Storage technology
Spark SQL	Apache Spark's module for working with structured data	Tumbling Sliding	Filter Aggregation Join	Spark	Streams Batch	Kafka RabbitMQ Kinesis Azure Event Hubs Google Pub/Sub	HDFS
Table API	Unified relational API for stream and batch processing	Tumbling Sliding Hopping	Filter Aggregation Join	Flink	Streams Batch	Kafka RabbitMQ Kinesis Azure Event Hubs	HDFS Amazon S3 MapR and Alluxio
KSQL	Open Source Streaming SQL for Apache Kafka	Tumbling Session Hopping	Filter Aggregation Join	N/A	Streams Batch	Kafka	HDFS
PipelineDB	Open-source relational streaming SQL database	Sliding	Filter Aggregation Join	N/A	Streams Batch	Kafka	PostgreSQL
Squall	Scalable online query engine that runs complex analytic	Tumbling Sliding	Filter Aggregation Join	Storm	Streams	Kafka	HDFS
StreamCQL	Continuous query language on real time system	Tumbling Sliding	Filter Aggregation Join	Storm	Streams	Kafka	HDFS
SamzaSQL	Scalable fault-tolerant SQL streaming engine on Samza	Tumbling Sliding Hopping	Filter Aggregation Join	Samza	Streams Batch	Kafka	HDFS
StormSQL	StormSQL leverages Apache Calcite to implement SQL standard on Storm	Tumbling Sliding Hopping	Filter	Storm	Streams	Kafka	HDFS
Siddhi	Java library for streaming queries	Sliding	Filter Aggregation Join	N/A	Streams Batch	Kafka RabitMQ	HDFS MongoDB Hbase
Athenax	Built on top of Apache Calcite and Apache Flink	Tumbling Sliding Hopping	Filter Aggregation Join	Flink	Streams Batch	Kafka	LevelDB

such Spark SQL,[16] Flink Table API,[17] KSQL,[18] SamzaSQL,[19] [16], StromSQL,[20] and Siddhi,[21] ,[22] etc. Indeed most of these engines are built on top of Apache Calcite[23] which is an open-source framework for building databases and data management systems. It is considered as an industry-standard SQL that consists of a parser, validator, and JDBC driver to support heterogeneous data models (i.e., relational, semi-structured, streaming, and geospatial).

We have conducted a systematic review to provide useful insights into state-of-the-art streaming query technologies by identifying a key list of features in query language processing for comparison. Table 4 lists the analysis of ten open-source big data stream query processing engines in terms of their query language. Additionally, for the time-streaming context, four types of windows are defined to perform operations on a finite size of time-stamped data and enable developers to perform complex stream processing jobs with minimal effort, described as follows[24] :

1 **Tumbling window**: It is a series of fixed-sized, non-overlapping, and contiguous time intervals. It segments a data stream into distinct time segments and performs a function against them such as aggregations and joins.
2 **Sliding window**: It produces an output only when an event occurs, which is required for specific applications. The window is not distinct and triggers the stream per defined intervals, for example, an application might require smoothed aggregates.
3 **Hopping window**: Unlike tumbling windows, hopping window model schedules overlapping windows by hopping forward in time within a fixed period; it is the tumbling window that can overlap.
4 **Session windows**: It groups events that arrive at similar times. It does not overlap, unlike hopping windows and does not have a fixed start and end time.

As the windows are considered heart of processing infinite streams, choosing the proper SQL engine is based on its supported windows types to fit it within a specific application scenario. For example, Spark SQL does not support hopping while KSQL, Table API, SamzaSQL, StormSQL, and Athenax support it, which gives us a variety of options to amalgamate different SQL engines and big data platforms for a given scenario. In contrast, StormSQL is limited to supporting join queries, so it could not be proposed for data applications which perform aggregation and join queries.[25] As another example, if a given use case needs a set of requirements which could be fitted with Storm, then three SQL engines could be proposed, i.e., Squall, StreamCQL, and StormSQL. Furthermore, if a use case requires a column-based or document-based store, Siddhi could be the proper SQL streaming engine that supports both HBase and MongoDB to store and query column-based or document-based historical data respectively. Moreover, for the data processing model (in terms of streaming, batch processing or both), the supported queuing technologies and big data storage technologies are identified for the reviewed open-source SQL streaming engines. We can conclude that all streaming SQL engines which are built on top of big data platforms can utilize their capabilities. Also, they can give a set of selections for the proper SQL engine depending on the use case requirements, capabilities of the platform, and data application scenario.

4. Predictive maintenance 4.0 use cases

In this section, the second research question,RQ2, is addressed.

RQ2: What are the requirements for building a data analytics pipeline focusing on predictive maintenance use cases?

We define the high-level technical requirements of predictive maintenance use cases. Then, the selected use cases from two different domains and their technical requirements are presented.

4.1. High-level technical requirements

Maintenance is one of the application areas in Industry 4.0; it is referred to as predictive maintenance 4.0 or PM 4.0. It enables systems to self-learn, predict failures, make their diagnoses, and trigger maintenance flows by using historical data, domain knowledge, and real-time data collected through IoT devices [17,18]. PM 4.0 has multiple attributes in various domains where each domain has different requirements. This study aims to introduce the dimensions of PM 4.0 analytics and associated technical requirements from a big data analytic techniques perspective (which are suitable for decision making). The outcomes of the presented study are identifying the characteristics of PM 4.0 in the era of big data and the discussion of these characteristics in the condition monitoring for Industry 4.0 use cases. It contributes to the ongoing discussion focusing on Industry 4.0, i.e., PM within both the academic sector and industry. Table 5 shows the PM 4.0 use cases' requirements using 10 main requirements and 29 sub requirements. Also, it presents the classification of the technical requirements in terms of queuing management, platform, storage, and SQL engines.

We analytically demonstrate these classified categories based on the big data technology stack, starting with distributed queuing management (**R6**) which collects the sensor data from the factory shop floor. Delivering data in an orderly manner from different locations is an important requirement which needs a scalable and efficient queuing system to accurately collect data from multiple machines generators. These heterogeneous data instances require representation in different schemas (i.e., **R2** such as structured, semi-structured, and unstructured). Then it could be stored based on various data storage models (i.e., **R1** such as distributed file system, document-based, and column-based) according to the analysis aspects. Furthermore, the **CAP** theorem needs to identify a choice for big data storage that may be important when retrieving historical data in PM solutions (**R5**). Typically, an efficient in-memory data processing is required to speed up data processing for decision-making, whether historical data or streaming data (**R4**). On the other hands, big data stack technology provides different big data platforms which support batch processing, stream processing, and both of them (**R3**). According to the capital of industrial markets, the big data platform is picked based on the needs of the PM use case.

Fundamentally, data comes from different sources in different formats (i.e. structured, semi-structured and unstructured) based on the complexity of the data contained within manufacturing processes such as process signals, text, multimedia including images, videos, audio, graphics and time series sequence data. So, the data complexity in terms of its size, variety, and uncertainty is difficult to be analyzed using traditional techniques. Consequently, knowledge processing approaches including machine learning, data mining, and deep learning techniques have extensively been used in many industry 4.0 applications (e.g., pattern recognition, product identification, product steering, predictive maintenance, scheduling, material flow control, and predictive analytics in supply chains) [19]. In particular, via the automation and intelligent feature in Industry 4.0 and its applications, the PM

[16] http://spark.apache.org/sql.
[17] http://ci.apache.org/projects/flink/flink-docs-master/dev/table/tableApi.html.
[18] http://www.confluent.io/product/ksql.
[19] http://github.com/milinda/samza-sql.
[20] http://storm.apache.org/releases/1.1.2/storm-sql.html.
[21] http://github.com/wso2/siddhi.
[22] http://docs.wso2.com/display/CEP400/SiddhiQL+Guide+3.0.
[23] http://calcite.apache.org.
[24] http://ci.apache.org/projects/flink/flink-docs-stable/dev/stream/operators/windows.html%23window-assigners.
[25] http://storm.apache.org/releases/1.1.2/storm-sql.html.

Table 5
High-level technical requirements.

Main requirements		Sub requirements		Description	Technology
R1	Data storage model	**R1.1**	File system	a method of stored data structure on the disk e.g., HDFS	Storage
		R1.2	Document-based	one of the main categories of NoSQL databases to store and managing semi-structured data	
		R1.3	Column-based	stores data tables by column rather than by row	
R2	Data schema	**R2.1**	Structured	e.g., RDBMs	Storage & Platform
		R2.2	Semi-structured	e.g., JSON, XML	
		R2.3	Unstructured	a.k.a schema-free or schema-less	
R3	Processing model	**R3.1**	Batch	to execute a series of non-interactive jobs all at one time	Platform
		R3.2	Streaming	to perform computing on data directly as it is produced or received	
		R3.3	Hybrid	use both of batch and hybrid	
R4	In-memory processing	**R4.1**	Real-time data	platform can process streaming data using memory	Storage & Platform
		R4.2	Historical data	storage can process stored data using memory	
R5	CAP Theorem	**R5.1**	Consistency	every read would get the most recent write	Storage
		R5.2	Availability	every node (if not failed) always executes queries	
		R5.3	Partition tolerance	the system continues to work despite message loss or partial failure	
R6	Distributed queuing management	**R6.1**	Delivering massage grantees	grantee of message delivered	Queuing management
		R6.2	Ordering massage grantees	grantee of message orders	
		R6.3	Federated queuing	to support data migration in distributed system	
R7	Knowledge processing	**R7.1**	Machine learning	to predicate failure of the system and aid the first-floor managers to make quick and accurate decisions on the fly e.g, regression, binary classification and multi-class classification	Platform
		R7.2	Deep learning	for very large data size and lack of domain understanding for feature selection	
		R7.3	Data mining	to discover the properties of datasets	
R8	Query types	**R8.1**	Complex join	to support join streams or joining streams with historical data	SQL engine
		R8.2	Aggregation	to draw conclusions for decision makers using stream or historical data analysis	
		R8.3	Filtration	to skip noisy or useless data	
R9	Window types	**R9.1**	Tumbling	perform analysis of non-overlapping continuous events	SQL engine
		R9.2	Sliding	perform analysis per identified intervals	
		R9.3	Hopping	perform analysis of overlapping continuous events	
		R9.4	Session	perform analysis of a group of events that have the same time	
R10	Computing paradigm	**R10.1**	Cloud computing	to perform analysis on the cloud	
		R10.2	Edge computing	to perform analysis using edge computing	

needs to utilize knowledge processing approaches for accurate failure predictions to enhance decision-making and maximize profit in the production system chains (**R7**). Mostly, these knowledge processing approaches require high-dimensional time-series data. The time-series data could be assembled from historical data using window-based queries such as join and aggregation queries to test the machine learning model (**R8**). Particularly, a kind of queries is described based on the behavior of oncoming events such as contiguous time intervals, defined intervals which could be pinpointed as the required window type for the PM use case (**R9**). The IoT-based predictive maintenance solutions could be deployed on different computing paradigms such as public cloud and edge computing in multiple industrial plants to leverage extra cloud capabilities for real-time analysis (**R10**).

4.2. Selective use cases from different domains

The aim of the study of PM 4.0 use cases is to discuss the characteristics of their requirements to reduce O&M costs. These requirements are based on the input data of use case and need of data-analytic-algorithms of PM solutions, from the viewpoint of big data technologies. Two Industry 4.0 use cases are discussed; railway maintenance for transportation industry and wind turbines maintenance for the energy industry.

4.2.1. Transportation industry – railway predictive maintenance

In this subsection, an overview of railway maintenance is presented together with detailed requirements for this use case.

Overview.

The transportation sector, especially railways, has largely adopted Industry 4.0 for improving quality of services, new savings, and enhanced resource utilization. The new trend of automation and the emerging big data technologies which deal with cyber-physical systems, IoT and cloud computing can achieve high levels of effective and efficient services for many passengers across the transport networks [20,18]. With the strong competition of regional and long-distance trains, providing an attractive service to meet passengers' requirements in terms of maturity and safety has become critical for many rail operators today. Consequently, the predictive maintenance applications could be deployed in the railway industry to make diagnoses and trigger maintenance actions.

The PM applications can perform predictive analytics to make decisions based on the analysis of huge amounts of data. For instance, the rail companies should be aware of the sudden rail change e.g., temperature variation, which expands rails. Another example, the failures of train wheels cause train deterioration and derailments. The failures contribute to the high cost of the global rail industry. Concerning the railway transportation industry, this problem signifies the need to monitor the performance of wheels and replace them in a preventive manner. The PM solutions of wheels will help with the just-in-time replacement of wheels. Also, the door failures of train cars are the major reason for delays in subway operations. So, the door failures predictions are needed to optimize door servicing schedules by identifying the remaining number of days until door failure.

Requirements.

Taking the big data technologies potentials into account and PM solutions can establish the health of the infrastructure and can contribute to strategic decisions about the railways [21]. Rail cars, locomotives, wayside, signal equipment, and track testing processes generate massive amounts of machine data (i.e., sensor data such as temperature, light, vibration, and GPS, etc.). These machine data contain valuable information to identify future railway failures (**R2.2**). On the other hands, multimedia data (e.g., audio, video, and surveillance data) and unstructured text data (e.g., document management which could be generated by railway technicians and managers are other new sources of unstructured data. These unstructured data could be leveraged into PM solutions for analysis purpose (**R2.3**).

Railway industries use big data technologies to perform predictive algorithms on heterogeneous data sources, scalable data structures, and real-time communications (**R3.2** and **R4.1**). For example, the multiplicity of sensors data will generate very large flows and volume of data in real-time such as the location of trains, speed, passengers on board, door status. Furthermore, failure could be early predicted by detecting abnormal conditions of use such as vibrations, energy consumption, including adjusting lighting, and air conditioning. For accurate failure prediction, the sensors data need high guarantees in terms of delivering and ordering consuming data using scalable queuing management technologies (**R6.1** and **R6.2**). The consumed timestamp-based data which are the enrolled windows within the train timetable could be typically analyzed according to discrete intervals such as tumbling windows or certain intervals such as sliding window (**R9.1** and **R9.2**). Then, the streamed data could be incorporated first by performing complex join queries to prepare the whole training data for real-time analysis purpose. For example, join operations could be performed for monitoring data (e.g., temperature, light, use of a seat), status monitoring of trains equipment (e.g., doors, load per axle, gear temperature, vibrations), localization data (e.g., GPS, and accelerometer), infrastructure equipment (e.g., switch position, number of trains having passed a point), and external conditions (weather, temperature, etc.) (**R8.1**, **R8.2** and **R8.3**).

Besides the online analysis of rail's data, the offline analysis could be performed to predict future failures using past failures information extracted from historical data. The historical railway data could be some static data such as train timetable (**R4.2**). Also, it is worth mentioning that the historical data should be consistently stored according to the recent trains timetables and conditions of use (**R5.1**). The historical data should be possible to easily restore and retrieve from big data storage (**R5.2**). Appropriately, the columnar database technology is efficient to write and read data to and from big data storage. In particular, the columnar storage technology is used to speed up the time it takes to return analytic queries by ignoring unwanted data within rows (**R1.3**). It would consist of a set of selected correlated columns which could be identified by using feature engineering from data mining techniques.

In the end, creating real-time predictive algorithms from heterogeneous data sources (i.e., sensor data and historical data) need to cope strong machine learning and data mining techniques such as feature and instance selection, discretization, data compression, ensemble classifiers, and regression models [22] (**R7.1** and **R7.2**). Meanwhile, the offline analysis which deals with non-continuous data could be performed in the batch manner. In addition, the online analysis needs to be performed using scalable fast streaming big data platforms (**R3.1**, **R3.2** and **R3.3**). This will provide new PM solutions that combine new database capabilities to integrate heterogeneous data sources in a high-performance accessing system based on cloud computing (**R10.1**). Table 6 lists the requirements of railway maintenance 4.0 based on the aforementioned detailed railway industry investigation.

4.2.2. Energy industry – wind turbines predictive maintenance

In this subsection, an overview of wind turbine maintenance is introduced and then the wind turbine maintenance requirements details are provided.

Overview.

Regarding the Energy Roadmap 2050, the wind energy supplies between 31.6% and 48.7% of Europe's electricity [23]. It has strong ties to Industry 4.0 and manufacturing. For example, automating wind turbines takes the ideas and methods of Industry 4.0 to implement concrete scalable systems in terms of vertical and horizontal communication, big data, and consistent engineering. On the other hands, the environmental benefits arising from the use of wind energy make a high competition in the energy markets [24,25].

Table 6
Predicative maintenance use cases requirements specifications.

Requirements		Railway	Wind turbines
Distributed queuing management			
R6	**R6.1**	✓	✓
	R6.2	✓	✓
	R6.3	X	✓
Platform			
R3	**R3.1**	✓	✓
	R3.2	✓	✓
	R3.3	✓	✓
R4	**R4.1**	✓	✓
R7	**R7.1**	✓	✓
	R7.2	X	✓
	R7.3	✓	X
Storage			
R1	**R1.1**	X	✓
	R1.2	X	X
	R1.3	✓	X
R2	**R2.1**	X	X
	R2.2	✓	✓
	R2.3	✓	✓
R4	**R4.2**	✓	✓
R5	**R5.1**	✓	✓
	R5.2	✓	X
	R5.3	X	✓
SQL engine			
R8	**R8.1**	✓	✓
	R8.2	✓	✓
	R8.3	✓	✓
R9	**R9.1**	✓	✓
	R9.2	✓	✓
	R9.3	X	X
	R9.4	X	X
Computing paradigm	**R10.1**	✓	✓
R10	**R10.2**	X	✓

According to the energy market, there is a constant need for reducing the cost of energy. Offshore wind farms, in particular, are under constant pressure for cost optimization [1]. Consequently, it is essential for operating companies that implement predictive maintenance strategies for increasing the life cycle of their wind systems [26]. Substantially, it is necessary to monitor turbine operational behavior, including operational sensors data of different machines and context information such as maintenance. Consequently, the turbine failure could be predicted and repaired during low usage periods of the wind turbine and before failure occurs. We aim to identify the wind turbine maintenance requirements based on aspects of big data technologies including real-time processing, storage, analytic queries in the condition monitoring wind turbines for providing capable PM 4.0 to the energy markets.

Requirements.

Concerning Energy 4.0, i.e., wind turbines, choosing the right big data technology stack for energy marketing is no different than for any other type of applications. Finding the proper technology that maps most economically and efficiently to the wind turbines maintenance requirements can minimize O&M costs. There are a lot of factors like, rough environmental conditions and installations in remote geographical locations which are the major concerns to be involved within monitoring and remote surveillance of wind turbines. Two main typical elements within a turbine which receive information from sensors are; Programmable Logic Controller (PLC) and Supervisory Control and Data Acquisition (SCADA). PLCs receive information from sensors transforming electrical signals to digital data while SCADA systems are physically connected to the sensors/PLCs collecting signal and other data.

For SCADA system within one turbine which contains 20–30 sensors, the amount of generated sensor data resulting in 60–100 different SCADA signals for 1 s is 8-byte values and 1.8 GB raw data per month [27]. Beyond this, a typical wind farm contains 10-100 turbines where zones or geographical regions incorporate 5–50 wind farms. Consequently, it is a big challenge to extract valuable knowledge and enable storage of raw data, especially in the case of PLCs which collect high-frequency data within some tens of milliseconds. Furthermore, these collected data need a guarantee to receive data from sensors in the correct order (**R6.1** and **R6.2**). As stated earlier, wind turbines are geographically distributed across wind farms which need to link different upstreams to satisfy the demand for messages from local consumers i.e., wind industry applications. It needs to aggregate data from turbines, give them to a common data storage model which could be done by using federated queues that geographically consumes data among turbines within different wind farms (**R6.3**). For the data chain of the wind industry, no data has to be dropped. For instance, both SCADA and PLC data with the resolution of some seconds and high-frequency condition monitoring signals can be stored centrally and distributively according to the wind farms locations using HDFS technology (**R1.1**). Moreover, most traditional data storage (i.e. SQL-based data architectures) are limited in scalability, so using the No-SQL database and column-based technologies are preferable to increase reliability in case of node failure (**R1.3**, **R5.2** and **R5.3**) [24]. Also, in-memory processing for historical data is a strong requirement for automated wind turbine maintenance (**R4.1**).

The wind turbine is a physical system which contains a combination of historical data, sensor data, and unstructured content (**R2.2** and **R2.3**). These data need to be analyzed whether in batch processing, stream processing, and both of them for accurate failure prediction in wind park (**R3.1**, **R3.2**, and **R3.3**). The historical data is considered as a traditional data for the predictive maintenance such as loads from wind, waves, electricity grid (i.e., to determine the behavior of each wind turbine in the farm), failure data (i.e., status code logs), service and maintenance activity list and system health management logs. The

Table 7
Predictive maintenance use cases requirements with respect to queue management technologies.

Use case	Req.	Apache Kafka	RabbitMQ	Amazon Kinesis	Ms Azure Event Hub	Google pub/sub	Proposed technologies
Railway	**R6.1**	✓	✓	✓	✓	✓	Kafka, RabbitMQ, Kinesis, Azure Event Hub
	R6.2	✓	✓	✓	✓	X	
Wind turbines	**R6.1**	✓	✓	✓	✓	✓	RabbitMQ
	R6.2	✓	✓	✓	✓	✓	
	R6.3	X	✓	X	X	X	

Table 8
Predictive maintenance use cases requirements with respect to big data stream platforms.

Use case	Req.	Hadoop	Storm	Samza	Spark	Flink	Proposed platforms
Railway	**R3.1**	✓	✓	✓	✓	✓	Storm
	R3.2	X	✓	✓	✓	✓	Spark
	R3.3	X	✓	X	✓	✓	Flink
	R4.1	X	✓	✓	✓	✓	
	R7.1	✓	✓	✓	✓	✓	
	R7.3	✓	✓	✓	✓	✓	
Wind turbines	**R3.1**	X	✓	✓	✓	✓	Storm
	R3.3	X	✓	X	✓	✓	Spark
	R4.1	X	✓	✓	✓	✓	Flink
	R7.1	✓	✓	✓	✓	✓	
	R7.2	✓	✓	✓	✓	✓	

recent type of data combination consists of a sensor measurement, the records of wind parameters (e.g., stress and acceleration), weather condition (e.g., the sea state, the wave heights based on the season of the year, the salt in the water) are being taken in to account [24,1]. Also, the unstructured content, including the technical reports (e.g., event annotations in control rooms) and multimedia data, could add significant contributions to the real analysis for PM of wind farms. These captured heterogeneous data sources, within discrete and certain intervals such as tumble and sliding windows (i.e., it could be hours, days, etc., based on the business needs), and historical fault prognostic data could be joined to be exposed within PM solution (**R8.1**, **R9.1**, and **R9.2**). Also, the bunch of summarizing of the wind farm during real-time analytics could be needed for PM solution using sensors to monitor turbine conditions such as temperature, wind direction, the power generated, generator speed, etc. (**R8.2** and **R8.3**). Ideally, data is gathered from multiple wind turbines from wind farms located in various regions where each turbine has multiple sensor readings relaying measurements at a fixed time interval.

Wind data is one of the high-dimensional time-series data which needs a precise prediction failure method such as numerical weather predictions, machine learning algorithms [28]. The other class of prediction methods is formed by machine learning algorithms that are implemented to predicate failures in time horizon from seconds to hours (**R7.1**). The predication requirement for wind data is formulated as a regression problem by employing the simple regression methods linear regression and the state-of-the-art technique support vector regression (SVR) for individual turbines and then for entire wind parks [29]. On the other hands, the wind industry adopts deep learning to predicate failure by progressively monitoring the engine degradation over its lifetime (**R7.2**). The degradation can be detected in engine sensor measurements. PM solutions try to model the relationship between the changes in the sensor values within turbines and the historical failures. Then, the model can predict when the turbine may fail in the future based on the current state of sensor measurements which can adopt Recurrent Neural Networks (RNNs). Currently, a different platform could perform deep learning techniques to cut down the training time to a matter of hours by using powerful GPU cluster. Also, different cloud providers have supported deep learning frameworks for industry and science, such as TensorFlow, by allowing deep learning models to scale efficiently at lower costs using GPU processing power. Ultimately, these geographically produced streamed wind data and distributed stored data help companies to perform their analysis by deploying PM solutions using scalable big data platforms which launched on cloud or edge (**R10.1** and **R10.2**). The summary of the wind turbine use case requirements from a big data perspective is analytically listed in Table 6.

5. Mapping stream processing technologies to predictive maintenance 4.0 use cases requirements

In this section, the below mentioned research question (**RQ3**) is addressed.

RQ3: How can the technical requirements for the use cases be mapped onto existing open-source big data technologies?

A minimal set of use cases from industry 4.0 (i.e., predictive maintenance for the railway transportation industry and wind turbines energy industry) are discussed based on their requirements in terms of some architectural composition of big data technologies. Then, the comparison parameters based on the use case requirements are mapped to the selected big data technologies. Finally, a few ideal combinations of open-source big data stream processing technologies are proposed for the selected maintenance 4.0 use cases.

Table 6 shows the specifications of the two use case requirements; railway and wind power. The selected industrial use cases need a guarantee to receive data from sensors in the correct orders. Particularly, the wind turbines industry needs to link different upstreams using federated queues that geographically consume data across multiple wind parks. According to the identified characteristics of queuing management systems listed in Table 1, Kafka, RabbitMQ, Kinesis and Ms Azure Event Hub could be used in railway maintenance while RabbitMQ is a perfect queuing technology for wind industry maintenance. Because, RabbitMQ supports federated queues which are feasible for wind farms geographically located at various sites (see Table 7). Regarding big data platforms, both use cases need batch as well as stream processing to perform offline analysis and online analysis for historical and real-time data respectively. The offline analysis usually requires in-memory computing for better performance. Consequently, Table 8 leads to finding the appropriate platform based on its capabilities to meet the use case requirements regarding streaming process model and knowledge processing approaches. More specifically, Apache Storm, Apache Spark and Apache Flink can be adopted in

Table 9
Predictive maintenance use cases requirements with respect to storage technologies.

Use case	Req.	Hadoop Hive	MongoDB	Cassandra	HBase	Proposed technologies
Railway	**R1.3**	X	X	✓	✓	HBase
	R2.2	X	✓	✓	✓	
	R4.2	X	✓	✓	✓	
	R5.1	✓	X	✓	✓	
	R5.2	X	✓	X	✓	
Wind turbines	**R1.1**	✓	X	✓	X	Cassandra
	R2.2	X	✓	✓	✓	
	R4.2	X	✓	✓	✓	
	R5.1	✓	X	✓	✓	
	R5.3	✓	✓	✓	X	

Table 10
Predictive maintenance use cases requirements with respect to SQL engines.

Use case	Req.	Spark SQL	Table API	KSQL	PipelineDB	Squall	StreamCQL	SamzaSQL	StormSQL	Siddhi	Athenax	Proposed SQL engine
Railway	**R8.1**	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	Spark SQL
	R8.2	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	Table API
	R8.3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	StreamCQL
	R9.1	✓	✓	✓	X	✓	✓	✓	✓	X	✓	SamzaSQL
	R9.2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	Squall and Athenax
Wind turbines	**R8.1**	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	Spark SQL,
	R8.2	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	Table API
	R8.3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	StreamCQL
	R9.1	✓	✓	✓	X	✓	✓	✓	✓	X	✓	SamzaSQL
												Squall and Athenax

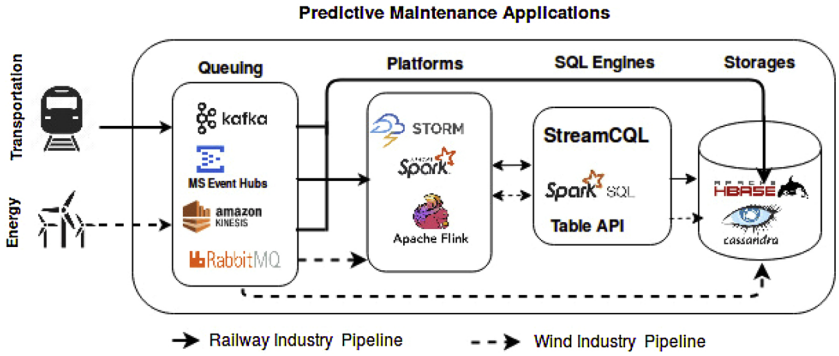

Fig. 2. The stream processing technologies for predicative maintenance 4.0 use cases: railway and wind turbines.

PM solution for railway and wind industries. In contrast, Apache Hadoop and Apache Samza are not likely to be a great fit for these use cases due to their limitations to support streaming and batch processing respectively (see Table 2).

Similarly, two use cases need to analyze varied data structure types which are generated from different data sources (such as operational machines, environmental conditions, and technical reports, etc.) for future failure prediction and then store this data for future analysis. For example, the railway industry requires column-based big data storage technology to store the correlated columns in order to accelerate query processing. While the wind industry needs to store data geographically in multiple partitions according to winds parks locations. Besides the scalable persistent data storage, fast response times are also required for decision-making in the real-time context. So, storage technology that supports in-memory data processing is required for both railway and wind industries. These storage requirements are identified in Table 9 and the results show that HBase fits with the railway industry because of its capabilities to consistently store data in column-based with high availability. In contrast, Cassandra fits with wind industry regarding its ability to consistently store data across partitions.

Regarding stream querying, railway and wind industries need to perform window-based queries such as join different data sources, provide summarisation by using aggregations and do a kind of filtration which could be needed for pre-processing data according to analysis purpose. Comparing these two use cases requirements for the capabilities of Big SQL stream engines, Table 10 presents that Spark SQL, Table API, StreamCQL, SamzaSQL, Squall, and Athenax can accomplish complex join, aggregations and filtration queries on streaming data based on the required tumble and sliding windows. It is worth mentioning that the PipelineDB and Siddhi are not considered because of their limitations to execute queries using thumbing windows while StormSQL is excluded because of its limitation to perform join and aggregation stream queries [30].

Ultimately, we can conclude that this study provides an extensive survey of state-of-the-art big data streaming technologies. Typically, Fig. 2 depicts the pipeline of industrial big data stream processing technologies for different predictive maintenance 4.0 use cases. Accordingly, the machine-data (i.e., maintenance data) is published by sensors, managed by queuing systems and then consumed by online analyzer using big data stream technologies or pulled up and then stored in big data storage. Meanwhile, some of the queries could be raised on this data, such as stream queries for real-time analysis and batch queries for retrieving historical data from big data storage. Furthermore, this study discusses how a combination of different

Table 11
The proposed combinations of big data technologies for the selected PM use cases.

Use case	Queuing system	Platform	Storage	SQL engine
Railway	Kafka	Storm	HBase	StreamCQL
	Kafka	Spark	HBase	SparkSQL
	Kafka	Flink	HBase	Table API
	RabbitMQ	Storm	HBase	StreamCQL
	RabbitMQ	Spark	HBase	SparkSQL
	RabbitMQ	Flink	HBase	Table API
	Kinesis	Storm	HBase	StreamCQL
	Kinesis	Spark	HBase	SparkSQL
	Kinesis	Flink	HBase	Table API
	Azure Event Hub	Storm	HBase	StreamCQL
	Azure Event Hub	Spark	HBase	SparkSQL
	Azure Event Hub	Flink	HBase	Table API
Wind turbines	RabbitMQ	Storm	Cassandra	StreamCQL
	RabbitMQ	Spark	Cassandra	SparkSQL
	RabbitMQ	Flink	Cassandra	Table API

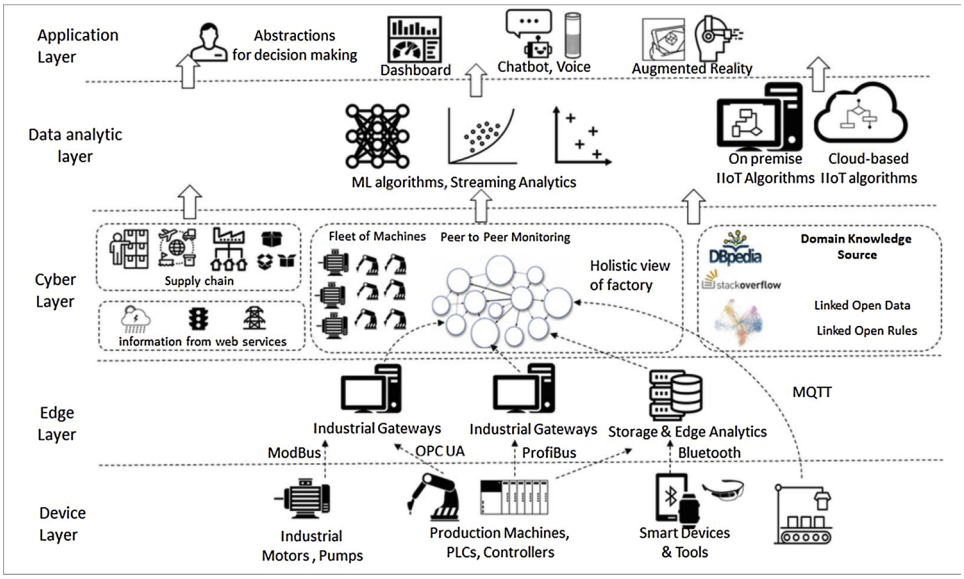

Fig. 3. A layered view of the Semantic Web of Things for Industry 4.0 (SWeTI) platform [34].

technologies could work for a certain PM 4.0 use case such as building a PM use case requiring a set of requirements which could be mapped to categorized requirements (i.e., main requirements including ***R1***, ***R2***, …, ***R10*** and their sub-requirements including ***R1.1***, ***R1.2***, …, ***R10.2***), the ideal combination can be such as queuing management technology ***W***, platform ***X***, persistent storage ***Y***, and SQL engine ***Z*** as described in Table 11.

Finally, a few ideal combinations of open-source big data stream processing technologies are proposed for the selected industry 4.0 use cases. As can be seen regarding railway transportation PM use case, 12 combinations of big data technologies can be used to build the whole pipeline starting from receiving raw data within factory till supporting intelligent decision-making, for example a combination of Kafka, Spark, HBase, MongoDB, and SparkSQL can be used. Also, another combination for railway transportation PM use case using open-source big data technologies could be Amazon Kines, Flink, HBase, and Flink APIs. Similarly, the wind turbines industry has 4 proposed combinations to perform the PM 4.0, but based on our analysis an ideal combinations could be using RabbitMQ, Storm, Cassandra, and StreamCQL.

6. Discussion and validation of the proposed technologies

In this section, we compare the contributions from our work with state-of-the-art studies conducted in a similar fashion, particularly surveys on the use of big data technologies for Industry 4.0.

6.1. Comparison with existing studies on big data requirements for Industry 4.0

Industry 4.0 has recently received a substantial attention from both research community and industry. More and more efforts are being conducted to truly realize the vision of Industry 4.0. Particularly, for the open-source community developing big data analytics solution, Industry 4.0 provides a set of use cases, while for the industry there is a great interest to use existing open-source state-of-the-art solutions to build intelligent applications for Industry 4.0. Below, we discuss literature review of the research work conducted on the use of big data and IoT for Industry 4.0.

The authors in [31] have identified requirements of Industry 4.0 to process large amount of data. Authors have provided a sophisticated categorisation of different tools and techniques required for data processing to develop Industry 4.0 applications. Authors have divided application requirements into two categories, namely, data requirements and processing requirements. The data requirements are further divided into three sub requirements, i.e., data model, data integration, and data content. The processing requirements are also divided into three sub-categories, namely, decision processing, knowledge processing, and real-time processing. Furthermore, the authors have mapped their categorization of requirements using two generic use cases to emphasize adoption of big data techniques for Industry 4.0.

In [32], the authors have reviewed state-of-the-art structure of a smart factory and then identified requirements of a smart factory. Their work is focused on providing an itemized list of requirements for a smart factory including its ability to process large amount of data and then use these requirements to define the characteristics of a smart factory. This work lays down a set of steps and principles in order to build a new smart factory or upgrade any existing traditional factory to make it smart. These principles are defined as modularity, interoperability, decentralization, virtualization, service orientation, and real-time capability. The identified requirements are then analyzed in comparison with existing research to provide a state-of-the-art review. This works aims to further narrow down the broad definition of a smart factory or Industry 4.0 and identifies the gaps which can be filled to truly achieve the vision of Industry 4.0.

Recently, Ismail et al. have shown the importance of data analytics for smart factory and proposed a few solutions for building a data processing pipeline to facilitate smart manufacturing [33]. The authors have identified requirements to build a big data analytics pipeline for manufacturing processes and then discussed available big data tools to show how these requirements can be met from the existing state-of-the-art tools and techniques. Authors have characterized the set of requirements into functional and non-functional. The non-functional requirements include functionality, usability, performance, and supportability, while the functional requirements are data ingestion, communication, storage, analysis, and visualization.

These aforementioned research works have mostly covered the requirements of the industry 4.0/smart factory and data analysis. However, to the best of our knowledge, there is not enough work on the requirements mapping for Industry 4.0 and stream processing. In this paper, our aim is to provide a similar study to map the requirements of industry 4.0 use cases in respect to streaming data platforms to cover Industrial Internet of Things (IIoT) related applications. We further did a fine-grained mapping between the capabilities of tools and requirements of use cases for industry 4.0/maintenance 4.0.

6.2. Validation of the proposed technologies

There have been varying perspectives with regards to validation (classifying research, including statistical analysis, comparison, simulation, surveys/questionnaire, and implementation). In this work (which belongs to research in engineering design), we have validated our proposed technologies using pre- and post- questionnaires as part of our industry engagement for the CONFIRM SFI Research Centre. We have had discussions with different industry partners for our requirements acquisition. An interesting observation was that the manufacturing industry within Ireland has in general shown a strong motivation towards modernizing their production processes using Industry 4.0. However, it was not clear what could and could not be achieved using these technologies or where businesses could see an immediate impact. This work was helpful for us as we built our survey on "*How ready are you for Industry 4.0?*" for dissemination amongst industry partners to better understand their requirements.

For this survey, we used one of our previous works called the SWeTI platform [34] as shown in Fig. 3. This platform proposes a layered architecture for using open source tools to build Industry 4.0 applications. Each layer within the SWeTI platform contains a variety of tools and techniques to build smart applications that can process raw sensor information and support smart manufacturing using Semantic Web and AI techniques. Similar to the designed SWeTI platform layers, we identified industrial requirements and proposed relevant big data technologies appropriate to the application layer, the data analytics layer, and the edge layer.

7. Conclusion

This paper bridges the technological gap between the requirements of Industry 4.0 applications and the capabilities of available big data and stream analytics technologies for the use case of predictive maintenance. We discussed the requirements of predictive maintenance use cases for railway transportation and wind energy, and showed the use of existing big data technologies to serve the requirements of these use cases. We provided a set of combinations of different data processing techniques and tool kits by discussing their strengths and weaknesses in different scenarios. We then mapped these technological combinations with the requirements of the selected use cases. The outcome of this work is a comprehensive set of guidelines and technology combinations with a focus on open source tools. We believe that this study will become a reference document for decision makers, data scientists/analysts, and developers to choose the most appropriate technology based on their application requirements. As a logical next step, we plan to share this document with a wider community working in the Industry 4.0 space and collaborate towards building Industry 4.0 solutions for our industry partners at the CONFIRM SFI Research Centre for Smart Manufacturing.

Acknowledgement

This publication has emanated from research supported by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/16/RC/3918 (CONFIRM), co-funded by the European Regional Development Fund.

References

[1] Nabati EG, Thoben KD. Big data analytics in the maintenance of off-shore wind turbines: a study on data characteristics. Dynamics in logistics. Springer; 2017. p. 131–40.
[2] Tao F, Qi Q, Liu A, Kusiak A. Data-driven smart manufacturing. J Manuf Syst 2018;48:157–69 special issue on Smart Manufacturing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0278612518300062.
[3] Wang J, Zhang L, Duan L, Gao RX. A new paradigm of cloud-based predictive maintenance for intelligent manufacturing. J Intell Manuf 2017;28(June (5)):1125–37. https://doi.org/10.1007/s10845-015-1066-0.
[4] Patel P, Ali MI, Sheth A. On using the intelligent edge for IoT analytics. IEEE Intell Syst 2017;32(5):64–9.
[5] Ali MI, Patel P, Breslin JG. Middleware for real-time event detection and predictive analytics in smart manufacturing. 2019 15th international conference on distributed computing in sensor systems (DCOSS). 2019. p. 370–6.
[6] Ferreiro S, Konde E, Fernández S, Prado A. Industry 4.0: predictive intelligent maintenance for production equipment. European conference of the Prognostics and Health Management Society 2016:1–8.
[7] Wang K. Intelligent Predictive Maintenance (IPDM) system – Industry 4.0 scenario. WIT Trans Eng Sci 2016;113:259–68.
[8] Oztemel E, Gursev S. Literature review of industry 4.0 and related technologies. J Intell Manuf, Jul 2018. [Online]. Available: https://doi.org/10.1007/s10845-018-1433-8.
[9] Mittal S, Khan MA, Romero D, Wuest T. A critical review of smart manufacturing & industry 4.0 maturity models: implications for small and medium-sized enterprises (SMEs). J Manuf Syst 2018;49:194–214 [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0278612518301341.
[10] Nedelcu B, et al. About big data and its challenges and benefits in manufacturing. Database Syst J 2013;4(3):10–9.
[11] Tiwari S, Wee H, Daryanto Y. Big data analytics in supply chain management between 2010 and 2016: insights to industries. Comput Ind Eng 2018;115:319–30.
[12] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Commun ACM 2008;51(1):107–13.
[13] Spangenberg N, Roth M, Franczyk B. Evaluating new approaches of big data analytics frameworks. International conference on business information systems. 2015. p. 28–37.
[14] Tian X, Han R, Wang L, Lu G, Zhan J. Latency critical big data computing in finance. J Finance Data Sci 2015;1(1):33–41 [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405918815000045.
[15] Simon S. Brewer's CAP theorem. CS341 Distributed information systems. University of Basel (HS2012); 2000.
[16] Pathirage M, Hyde J, Pan Y, Plale B. SamzaSQL: scalable fast data management with streaming SQL. 2016 IEEE international parallel and distributed processing symposium workshops (IPDPSW). 2016. p. 1627–36.
[17] Ji W, Wang L. Big data analytics based fault prediction for shop floor scheduling. J Manuf Syst 2017;43:187–94 [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0278612517300389.
[18] Mourtzis D, Vlachou E. A cloud-based cyber-physical system for adaptive shop-floor scheduling and condition-based maintenance. J Manuf Syst 2018;47:179–98 [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0278612518300700.
[19] Wang J, Ma Y, Zhang L, Gao RX, Wu D. Deep learning for smart manufacturing: methods and applications. J Manuf Syst 2018;48:144–56 special issue on Smart Manufacturing. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0278612518300037.
[20] Kans M, Galar D, Thaduri A. Maintenance 4.0 in railway transportation industry. Proceedings of the 10th world congress on engineering asset management (WCEAM 2015). 2016. p. 317–31.
[21] Thaduri A, Galar D, Kumar U. Railway assets: a potential domain for big data analytics. Procedia Comput Sci 2015;53:457–67.
[22] Li H, Parikh D, He Q, Qian B, Li Z, Fang D, et al. Improving rail network velocity: a machine learning approach to predictive maintenance. Transport Res C: Emerg Technol 2014;45:17–26.
[23] Márquez FPG, Tobias AM, Pérez JMP, Papaelias M. Condition monitoring of wind turbines: techniques and methods. Renew Energy 2012;46:169–78.
[24] Helsen J, Peeters C, Doro P, Ververs E, Jordaens PJ. Wind farm operation and maintenance optimization using big data. 2017 IEEE Third International Conference on big data computing service and applications (BigDataService). 2017. p. 179–84.
[25] Canizo M, Onieva E, Conde A, Charramendieta S, Trujillo S. Real-time predictive maintenance for wind turbines using big data frameworks. 2017. arXiv preprint arXiv:1709.07250.
[26] Dahane M, Sahnoun M, Bettayeb B, Baudry D, Boudhar H. Impact of spare parts remanufacturing on the operation and maintenance performance of offshore wind turbines: a multi-agent approach. J Intell Manuf 2017;28(October (7)):1531–49. https://doi.org/10.1007/s10845-015-1154-1.
[27] Viharos ZJ, Sidló CI, Benczúr A, Csempesz J, Kis KB, Petrás I, et al. "Big data" initiative as an IT solution for improved operation and maintenance of wind turbines. 2013.
[28] Heinermann J, Kramer O. Machine learning ensembles for wind power prediction. Renew Energy 2016;89:671–9.
[29] Treiber NA, Heinermann J, Kramer O. Wind power prediction with machine learning. Computational sustainability. Springer; 2016. p. 13–29.
[30] Storm sql integration. [Online]. Available: http://storm.apache.org/releases/1.1.2/storm-sql.html.
[31] Gölzer P, Cato P, Amberg M. Data processing requirements of industry 4.0 – use cases for big data applications. ECIS 2015.
[32] Mabkhot M, Al-Ahmari A, Salah B, Alkhalefah H. Requirements of the smart factory system: a survey and perspective. Machines 2018;6(2):23.
[33] Ismail A, Truong H-L, Kastner W. Manufacturing process data analysis pipelines: a requirements analysis and survey. J Big Data 2019;6(1):1.
[34] Patel P, Ali MI, Sheth A. From raw data to smart manufacturing: AI and semantic web of things for industry 4.0. IEEE Intell Syst 2018;33(July (4)):79–86.