Received March 4, 2019, accepted March 27, 2019, date of publication April 18, 2019, date of current version April 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2911979

Implementation of a Large-Scale Platform for Cyber-Physical System Real-Time Monitoring

MIKEL CANIZO[1], ANGEL CONDE[1], SANTIAGO CHARRAMENDIETA[1], RAÚL MIÑÓN[2], RAUL G. CID-FUENTES[3], AND ENRIQUE ONIEVA[4]

[1]Ikerlan Technology Research Centre, 20500 Arrasate-Mondragón, Spain
[2]Tecnalia Research and Innovation, 01510 Vitoria-Gasteiz, Spain
[3]Global IoT and Eleven Paths & Telefónica Investigación y Desarrollo, 28050 Madrid, Spain
[4]Deusto Institute of Technology (DeustoTech), University of Deusto, 48007 Bilbao, Spain

Corresponding author: Mikel Canizo (mcanizo@ikerlan.es)

This work was supported by the Basque Government through the Elkartek Program under the TEKINTZE Project under Grant KK-2018/00104.

ABSTRACT The emergence of Industry 4.0 and the Internet of Things (IoT) has meant that the manufacturing industry has evolved from embedded systems to cyber-physical systems (CPSs). This transformation has provided manufacturers with the ability to measure the performance of industrial equipment by means of data gathered from on-board sensors. This allows the status of industrial systems to be monitored and can detect anomalies. However, the increased amount of measured data has prompted many companies to investigate innovative ways to manage these volumes of data. In recent years, cloud computing and big data technologies have emerged among the scientific communities as key enabling technologies to address the current needs of CPSs. This paper presents a large-scale platform for CPS real-time monitoring based on big data technologies, which aims to perform real-time analysis that targets the monitoring of industrial machines in a real work environment. This paper is validated by implementing the proposed solution on a real industrial use case that includes several industrial press machines. The formal experiments in a real scenario are conducted to demonstrate the effectiveness of this solution and also its adequacy and scalability for future demand requirements. As a result of the implantation of this solution, the overall equipment effectiveness has been improved.

INDEX TERMS Anomaly detection, big data, cyber-physical system, industry 4.0, real-time processing.

I. INTRODUCTION

In recent years, industrial manufacturing has advanced due to the fourth Industrial Revolution (Industry 4.0) and the Internet of Things (IoT) [1]. This evolution has been boosted by the specific needs of the industrial manufacturing sector [2]. Thus, companies have experienced a technological transformation by adopting Cyber-Physical Systems (CPSs) rather than traditional embedded systems [3]. Although the term CPS is applied to a wide variety of domains, it is assumed in this article to refer to an Industrial Cyber-Physical System (ICPS) because the scope of this work covers the manufacturing domain.

ICPSs enable new advanced strategies to be implemented to improve and optimize the manufacturing processes in the entire lifecycle of the manufacturing system and, therefore, of the product [4], [5]. This would lead to higher quality products and improvements in productivity and energy savings. This has encouraged the European Monitoring and Control (M&C) market to invest €143 billion in this area by 2020, making a total of €500 billion invested by the world wide M&C market [6].

An ICPS can be made up of many different devices. Some of these devices can communicate with each other to make decisions while the system is in operation, leading to smart manufacturing [7], [8]. Thus, it is necessary to capture all data coming from ICPSs to cost-effectively monitor the operation of these industrial systems to timely detect anomalies and avoid production shutdowns. The captured data allows anomalies in the system to be found. This anomaly detection helps to find errors at an early stage. However, to do so, all information received from an ICPS must be captured

The associate editor coordinating the review of this manuscript and approving it for publication was Yuedong Xu.

and processed. Note that an ICPS can be composed of many devices and, consequently, the data volume received is large. Hence, the data volumes that are currently generated are too large to be processed with traditional technologies [9], which often means delays and may cause non-functioning. This can be critical for decision-making processes, since obtaining correct information at the correct moment is a key issue [10]. Late detection of a fault can also be critical for an industrial machine and, consequently, for productivity. In an industrial scenario, the system needs to be in operation 24/7. Therefore, the devices, the network, and so on in a smart manufacturing system cannot be interrupted, since this would cause a drop in production and a loss of money. This is typically measured by the Overall Equipment Effectiveness (OEE), which identifies the percentage of manufacturing time that is truly productive.

The rapid growth and widespread use of a wide range of information technologies, from individual sensors to cloud computing and cloud services, has led to an increase in the volume of data that needs to be processed. As the data volume increases, the ICPS must be horizontally scaled up to add more computational resources to ingest, process and store the data. However, other issues can arise as more computational nodes are added to the existing server to spread the load across them, such as data partitioning [11] or the management of the computational resources, among others. Hence, scalability is one of the main challenges to these systems. In this context, Big Data frameworks and cloud computing are particularly important since they provide fast, scalable and fault-tolerant data processing capabilities for ICPSs [5]. Cloud-based approaches are especially suitable for small and medium enterprises since they provide on-demand services, which require lower barriers and initial investments [12].

This work presents a Big Data approach for ICPSs to perform a real-time analysis of the operational state of industrial systems in the manufacturing industry. This will enable the ICPS to take advantage of the benefits of Big Data in a cloud computing environment. This work is validated in a real industrial scenario where various press machines are used. The validation comprises the implementation of this solution to improve the performance and reliability of its monitoring and anomaly detection systems, since they required a new platform that is faster to process efficiently the data volume they currently generate and scalable to meet future needs. Consequently, this approach has helped to improve the OEE of their industrial systems. Moreover, the monitoring system developed here can be easily deployed on third-party cloud infrastructures such as Amazon EMR,[1] Microsoft HDInsight[2] or Cloud Dataproc.[3]

The main contribution of this work is the design and the implementation of a large-scale ICPS for monitoring industrial machines in a real work environment, where digitization is not yet very advanced. Specifically, the novelty of this work lies in: (i) the use of a Big Data solution to satisfy the needs of real industrial scenarios, both currently and in the future; and (ii) addressing the challenges identified in the literature review.

The rest of this article is structured as follows: Section II presents the state of the art. Section III describes the proposed Big Data architecture. Section IV details the configuration of the experimentation as well as the industrial case study. Section V analyzes the results of the scalability tests that we have performed. Section VI presents conclusions and future work. However, due to confidentiality, it is not possible to show specific details about the anomaly detection process or the OEE.

II. STATE OF THE ART

The term cyber-physical system was first coined by Helen Gill at the National Science Foundation. Briefly, it can be referred as a new generation of systems with integrated computational and physical capabilities that can interact with humans through many new modalities [13]. In fact, CPSs can be defined as "*physical, biological and engineered systems whose operations are monitored, coordinated, controlled and integrated by a computing and communication core*" [14], although more definitions can be found in the literature [15]–[18].

The communication between physical and digital elements has come to play an important role in various industrial domains [16], [19], [20]. Within a manufacturing context, the use of ICPSs has led to smart manufacturing. According to the National Institute of Standards and Technology (NIST), these are "*fully-integrated, collaborative manufacturing systems that respond in real time to meet changing demands and conditions in the factory, in the supply network, and in customer needs*" [21]. However, an ICPS by itself is not sufficient for efficient monitoring of industrial systems. The gathered data must go through four IoT stages to gain enough knowledge to make an accurate decision [22]–[25]. These stages start from the device's connectivity to transmit the data, followed by real-time monitoring that enables the operational state of the systems to be visualized, and eventually leading to desired business outcomes. A data analytics stage then delivers insight, predictions, and optimization for the performance of the ICPS. Finally, an enhanced on-board intelligence that provides the maximum business benefit from the information obtained in the previous stages is required. This allows companies to gather data from their industrial systems and then process it to extract useful information to help make relevant decisions, failure diagnoses and to introduce predictive maintenance.

Given the necessity to analyze the data coming from the physical elements as fast as possible and the huge volume of captured data, Lee *et al.* [26] conclude that algorithms are required to draw conclusions and avoid anomalies. Consequently, data analysis can be more efficient than analyzing the data manually. As Niggemann *et al.* [27] show, humans are

[1] https://aws.amazon.com/es/emr/
[2] https://azure.microsoft.com/es-es/services/hdinsight/
[3] https://cloud.google.com/dataproc/

unable to draw conclusions in a fast and efficient way when a huge amount of data is involved. They also claim that systems managed by humans are hard to maintain, besides being incomplete.

However, acquiring data and processing it involves high levels of computational requirements. Colombo *et al.* [6] claim that these systems should be based on process control algorithms, architectures, and platforms that are scalable and modular (plug and play), and which are applicable across several sectors. Consequently, the application of cloud computing and Big Data technologies to ICPSs has attracted the interest of several researchers. Thus, the literature has identified the challenges of cloud-based ICPSs [28]–[31], and which agree on the need for an ICPS with the following characteristics:

- Cloud-based distributed file systems for ubiquitous access to data.
- Open-source programming frameworks to process and analyze Big Data.
- Large-scale, fast and fault-tolerant data processing.
- Real-time data collection from cyber-physical devices and storage in the cloud.
- Remote monitoring and control capabilities.
- Software as a Service (SaaS), Hardware as a Service (HaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).
- An intelligent search engine to answer queries.

Although researchers have recently focused on achieving some of these challenges, to the authors' knowledge, none have addressed these characteristics embedded in a single solution.

III. BIG DATA ARCHITECTURE FOR AN INDUSTRIAL CYBER-PHYSICAL SYSTEM MONITORING

This section details the proposed Big Data architecture for ICPS real-time monitoring. First, the components of the architecture are introduced (Section III-A). The technologies that are used to implement this architecture are then described (Section III-B). Finally, the workflow followed by the architecture to assess the monitoring of the ICPSs is detailed (Section III-C).

A. ARCHITECTURE

The goal of this architecture is to process the data generated by the industrial machines to monitor in real-time their operational state, providing key information to maximize the OEE. However, when many industrial machines are continuously sending data, a Big Data architecture is required to manage the large volume of data that is generated by the ICPSs. At this stage, a number of technological requirements arise, which must be analyzed in depth [27], as follows:

- **Data acquisition:** a system that is capable of gathering data from the industrial machines and sending it to the cloud is required. Consequently, a system that is capable of managing all of the data sent to the cloud is also required. These systems must efficiently manage thousands of messages per second without forming a bottleneck. They must also be scalable to be able to handle data volume increments. Finally, data loss must be prevented since the monitoring cannot be properly carried out without all of the relevant data. Thus, it must be fault-tolerant [32].
- **Data processing:** a data processing engine that is capable of processing the streaming data coming from the industrial machines is required. It must also enable batch processing for advanced analytics purposes. To detect anomalies as early as possible, the data processing engine must be fast in terms of processing data and executing calculations. It must also be able to manage data volume increments (scalability) and to handle system failures (fault-tolerant).
- **Data persistence:** there is a need to store huge volumes of data with high throughput. Consequently, storage flexibility is required as the data volume increases. Moreover, an efficient search engine is required to query the database without excessive delays. Fault-tolerance is also required to miss no data in case a system failure occurs.
- **Data serving:** a system that provides services to query/push information from/to a user interface is required to easily check in real-time the operational status of the industrial machines. This system must provide mechanisms to handle immediate information (i. e., anomaly alerts or current machine status), and medium to long-term information (i. e., advanced analytics). In other words, it manages the connections between the cloud and the user interface.

With these requirements in mind, as depicted in Figure 1, the architecture is divided into three main blocks: a local data acquisition, a cloud platform, and a front-end. The former is located physically on the manufacturing factory, that is, it is deployed in the servers of the company (on-premise). It is responsible for gathering data generated by the industrial machines and then sending it to the cloud. The local data acquisition system is composed of a database where the data coming from the industrial machines is stored and a data publisher pushes the new data to the cloud. This is part of the data acquisition system; concretely, the local side.

The cloud platform is the core block of the architecture since it is in charge of managing, processing, persisting, and serving all of the data sent to the cloud. To give support to all of the previously described requirements, according to lambda architecture [33], the cloud platform is divided into three layers: serving, speed, and batch layers. The serving layer encompasses all services related to the data acquisition in the cloud side, the data persistence, and the data serving. The speed layer includes real-time data processing services; that is, it is responsible for processing in real-time the data coming from the industrial machines. The batch layer provides batch processing services, which means that the data

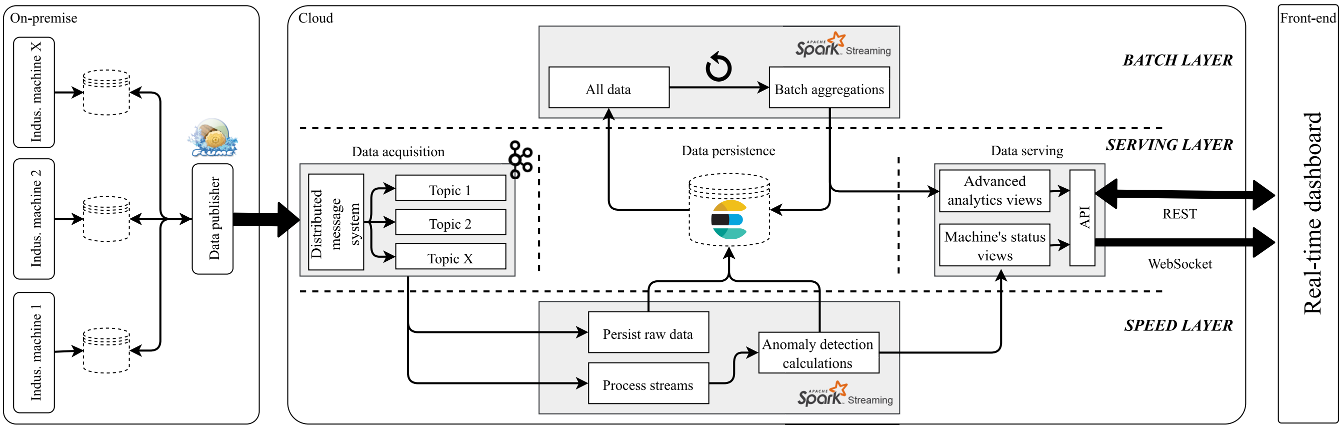


FIGURE 1. Architecture of the proposed ICPS real-time monitoring system.

is processed periodically with a long time interval. This is responsible for performing advanced analytics by aggregating historical data stored in the database.

The front-end is the visual component of the architecture. It shows in real-time the status of the industrial machines and it enables advanced analytics to be displayed to support decision making. The front-end is also deployed within the cloud platform.

B. CLOUD MANAGEMENT AND USED TECHNOLOGIES

Of the requirements that we have described, the management of volume, velocity, scalability, and fault-tolerance are the most important issues. The first two are already addressed by adopting a Big Data paradigm since they are implicit [34]. In addition, the multi-node design of Big Data frameworks provides scalability [35] and fault-tolerant features. Therefore, the selection of these technologies was made in view of the requirements and issues described above.

Since data persistence is required, Elasticsearch[4] was selected Elasticsearch is a distributed, document-oriented, RESTful search and analytics engine that is capable of persisting data and fulfilling the established requirements [36]. Besides satisfying our requirements, Elasticsearch is also a mature and robust technology that has been successfully adopted in other domains [37]–[40]. InfluxDB was also suitable for this use case although it was discarded as it must be paid in case more than one node is used.

Apache Flume[5] and Apache Kafka[6] were selected to meet the data acquisition requirements. Flume is a distributed, reliable and available service that can efficiently collect, aggregate and move large amounts of logged data. It is used as a data publisher to send the data generated by the industrial machines to the cloud. Its potential is demonstrated in [41]. Kafka is a distributed messaging system that uses the publisher/subscriber communication pattern [42]. It is in charge of passing the data sent by the data publisher into the cloud. Furthermore, Kafka minimizes the loss of messages by means of its fault-tolerant design. Apache Kafka has also been adopted in similar use cases [43]–[45].

Apache Spark Streaming[7] was adopted to process the data. Spark Streaming is a scalable, high-throughput, fault-tolerant stream processing for live data streams. For streaming data processing, a short batch interval was defined while a long batch interval was defined for batch processing. The main advantage of Spark Streaming is its in-memory data processing, which provides a faster engine than those using disk I/O. This allows data to be processed 100 times faster than traditional Big Data technologies [46]. In addition, it is supported by a huge developer community and is powered by companies such as IBM, Hortonworks or Cloudera, which means that the framework is mature and resilient over time.

Two components are used to manage messages between the application and the dashboard: a generic REST API to query the database from the dashboard, and a WebSocket that enables direct messages to be sent to the dashboard. These are responsible for data serving, which is located in the cloud within the serving layer. To provide fault-tolerant and scalability capabilities to the dashboard, a microservice solution based on the 12-factor app approach[8] was adopted.

To carry out the management of cloud resources Apache Mesos[9] and Apache Zookeeper[10] are used. Among other computational resources, Apache Mesos abstracts CPU, memory, and storage away from the machines (physical or virtual), enabling fault-tolerant and elastic distributed systems [47]. Mesos is in charge of dynamically managing the resources used by the frameworks within the cloud, specifying where and how they have to be executed. In the same way, it allows to dynamically add or reduce the resources

[4] https://www.elastic.co/products/elasticsearch
[5] https://flume.apache.org/
[6] https://kafka.apache.org/
[7] https://spark.apache.org/streaming/
[8] https://12factor.net/
[9] https://mesos.apache.org/
[10] https://zookeeper.apache.org/

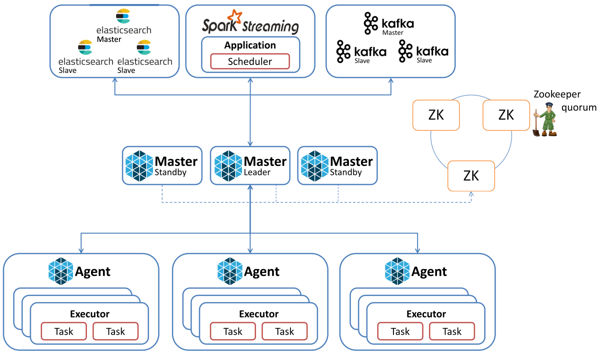


FIGURE 2. Configuration of the cloud platform.

that a framework can use at run-time. Apache Zookeeper is a centralized service that is used to maintain the configuration information and naming, to provide distributed synchronization and group services [48].

The design of the cloud management is shown in Figure 2. Mesos has three master nodes and three agent nodes. The former are responsible for managing the cloud and the latter for executing the scheduled tasks. Only one of the master nodes is active (the leader) while the others are in standby mode as a replica in case the leader fails.

Zookeeper, Mesos, Elasticsearch, and Kafka have a master node and two slave nodes. This allows a threefold replication of the services. Therefore, if an error were to occur in one of the nodes, then the data would still be available in the remaining live nodes. Furthermore, this avoids the single point of failure problem [49]; that is, there is no point at which if something fails, the entire system stops working. In addition, if any master node fails, then Zookeeper would be responsible for selecting a new master node. Thus, the application could continue working properly.

It is worth pointing out that Flume is deployed locally. Thus, if network problems were to arise, the gathering of new data would continue and submission of the data to the cloud would take place when the network was able to recover its normal behavior. Despite not being deployed on the cloud, Flume can restart itself at the point it had reached before failure occurred. To do this, it uses a checkpointing mechanism to ensure that no events are lost. Moreover, Flume itself is a scalable framework. This guarantees scalability and fault-tolerant services.

C. REAL-TIME MONITORING

This section describes the data-flow followed to perform the real-time monitoring of the industrial systems and to early detect anomalies. This process starts when data is gathered from industrial machines through a data acquisition system. At this stage, data is provided by a Programmable Logic Controllers (PLCs) installed on each industrial machine; this data is then persisted in a local database. Concurrently, the data publisher periodically queries[11] the local database to check whether new data is available. If so, then it publishes new data within Kafka topics (Kafka publisher role). Note that various types of data are gathered from industrial machines and thus, there is one topic for each data type.

Once the local side of the data acquisition system publishes data on Kafka, the real-time processing service subscribes to the corresponding topics to read the messages. It uses as many data streams as topics, which means that it can process the sent data concurrently through these topics. It is noteworthy that if the data volume increases in the future, then the number of partitions per topic can be increased. Therefore, more data streams can be created for each topic and, as a consequence, the data ingestion throughput can be increased. Subsequently, the tasks of Apache Spark are twofold and are executed concurrently: 1) to persist the received data into Elasticsearch, and 2) to process data and perform calculations to detect anomalies.

Based on expert advice and experience, three flags were defined to model the different states resulting from the calculations: green, yellow and red. These colors indicate the criticality of the anomaly, with the red flag representing the most critical state and the green flag indicating normal behavior. Each industrial machine has its own flag. Two boundaries were defined to measure the criticality of the anomaly: a low and a high boundary. These boundaries are static and were set by experts. Hence, if the result of a calculation is higher than the low boundary, then a yellow flag is generated. If it is higher than the high boundary, then a red flag is generated. Otherwise, the flag is set to green. The color of the flags can only be modified to increase the criticality of the anomaly, and an alert is generated only in this case. The state of the alarms is reset to green by executing a specific mechanism for resetting the alarms.

Within this use case, there are two types of calculations to detect anomalies: in the first, cataloged as Single Data Anomaly Detection (SDAD), a calculation is performed for each received measurement. In the second, classified as Multiple Data Anomaly Detection (MDAD), multiple measurements are required to execute a single calculation. Following the computations, the criticality of each is verified using the boundaries described above. At this point, critical anomalies are detected and an alert is sent to the dashboard.

The current state of the alarm is compared with a newly detected anomaly since an alert is only sent to the dashboard if the anomaly is more critical than the current state. However, Spark does not support the persistence of values between batches as a default. Consequently, a stateful method was adopted. In this way, the previous states are available within each batch, for comparison purposes. This method takes into consideration a key, a value, and a state as input parameters. The *key* is a unique identifier for classifying the data; the *value* is the data received within a batch, classified by key; and the *state* is the parameter through which data can be persisted in memory between batches.

Algorithm 1 shows the method followed for SDAD. First, the data is received by means of data streams and it is then

[11] https://github.com/keedio/flume-ng-sql-source

Algorithm 1 Single Data Anomaly Detection algorithm

```
receive data from kafka
group received data by key          ▷ key=machineNumber
execute statefulMethod;

procedure statefulMethod(key, value, state)
    calculations = doCalculations(value)
    currentState = checkCurrentState(calculations)
    previousState = getPreviousState(state)
    if currentState > previousState then
        update the current state
        send an alert to the dashboard
    end if
end procedure
```

Algorithm 2 Multiple Data Anomaly Detection Algorithm

```
receive data from kafka;
group received data by key;         ▷ key=machineNumber
execute statefulMethod1;
if statefulMethod1 returns some value then
    execute statefulMethod2;
end if

function statefulMethod1(key, value, state)
    arrayData = getPreviousState(state);
    fill arrayData with new received data;
    check if all data is received;
    if all data received then
        remove state;
        return arrayData;
    else
        update the current state;
    end if
end function

procedure statefulMethod2(key, value, state)
    calculations = doCalculations(value);
    currentState = checkCurrentState(calculations);
    previousState = getPreviousState(state);
    if currentState > previousState then
        update the current state;
        send an alert to the dashboard;
    end if
end procedure
```

grouped by key. Subsequently, a stateful method is applied. In this case, the *key* is the machine identifier, the *value* is the data received in the actual batch, grouped by key, and the *state* is a string containing the current color of the flag for each machine. Following this, calculations are performed to check whether a more critical anomaly is found. If so, then the value of the state is updated with the corresponding color, and an alert is generated and sent to the dashboard for visualization. Finally, it returns to receive new data.

Algorithm 2 shows the method followed for MDAD. In this case, two stateful methods are used: the first waits until all required data is received, and the second measures the criticality of the calculations. In the first method, data received within a batch is grouped by key. The *value* is the data received within a batch grouped by key, while the *state* refers to an array whose size is determined by the amount of data required. Thus, this array is filled each time that a batch is processed. If not all of the data is received, then it returns to receive new data. The calculation can be performed when all of the required data is received. At this point, the state is removed to free memory space, since this data is not used again in this context. The result is then sent to the second stateful method, where the criticality of the given result is measured. This second method performs the same process as that described for SDAD. Finally, it returns to receive new data.

To correct the maintenance process, alerts are also sent to the Technical Assistance department where a maintenance assistant analyzes the failure and plans the corresponding corrective actions to be made, if needed. If the failure can be remotely fixed, then the assistant will start the process using a Virtual Private Network (VPN). Otherwise, the assistant will launch a maintenance order, in which the assistant will have to physically fix the fault.

This solution can be implemented in any industrial domain as it is composed of generic frameworks. Regarding the architecture, it must be equal for any domain. However, since industrial systems in each domain have their own characteristics and requirements, the way in which ICPSs gathers data from the physical machines and how they send the data to the local database must be changed. This is an ad-hoc process. As the data flow is regarded, the data ingestion, the data processing, and the data persistence are also ad-hoc processes and, therefore, they must be adapted to the corresponding requirements of the specific domain. Consequently, the data structure and how the data is processed and modeled must be modified.

IV. EXPERIMENTAL FRAMEWORK

This section describes the configuration and properties related to the experimentation followed in this article. Note that the experimentation only covers the performance and scalability of the real-time processing side of the architecture. This is due to the fact that it is the most demanding and critical part of the entire system. We first describe the industrial case study (Section IV-A). We then define a hypothesis in order to rigorously define the objective (Section IV-B). Next, we describe the used evaluation metrics (Section IV-C), and the conducted scalability tests (Section IV-D).

A. INDUSTRIAL CASE STUDY

This work is validated in a real industrial scenario where press machines are used. Although this work can be applied to other domains (see Section III), the rest of the article is focused on this particular use case. Press machines are industrial systems

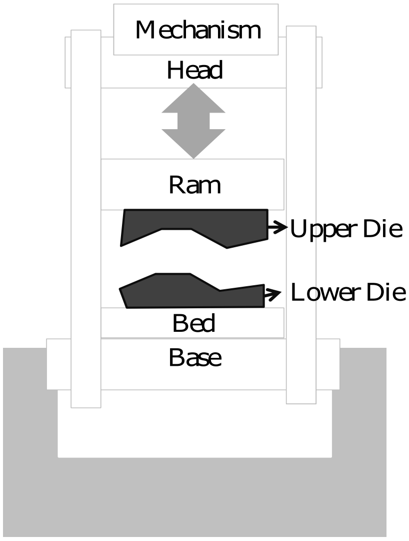


FIGURE 3. **Composition of a press machine (main components).**

that are capable of changing the shape of a workpiece by applying pressure on it. The main components of a press machine are shown in Figure 3. A press is composed of a mechanism, two rigid platforms (head and base), a bed, a ram and two dies (upper and lower). The die gives shape to the workpiece. During this process, a workpiece is introduced between two dies to mold it into the corresponding shape by applying a specific pressure. The mechanism is responsible for moving the head and, in turn, the ram and the upper die to apply the required pressure to the workpiece. The complete action of pushing down the mechanism to change the shape of the workpiece is called a stroke.

Press machines have to work 24/7 and they must withstand huge amounts of pressure at each stroke. Furthermore, their components continuously grind against each other. Therefore, structural failure can be critical for both the press machine and the product. This can lead to unplanned downtimes, and consequently to expensive repair work. There are three main indicators that can cause critical operation for press machines [50]: (i) mechanism misalignment, which can lead to friction between components and thus to malformations of components that can cause imperfections in the final workpiece; (ii) oil degradation is another indicator because poor lubrication can indicate friction between the components, which can also increase the temperature of the press; and (iii) temperature is another key indicator that should be taken into account. Any problem from these indicators may cause a significant impact on any of the OEE scores (Availability, Performance, and Quality).

These industrial systems are equipped with a number of sensors that offer relevant measures related to the working performance (i.e., temperature, pressure, inductive or flowmeter sensors), which can be useful in monitoring the previously described indicators. However, processing the data gathered from these sensors raises three issues. First, the data volume generated may be too large to be processed in real time. Second, the calculations required to detect anomalies are generally expensive in terms of computational cost. Finally, the industrial context is prone to failures (i.e., network or power downtimes), which can lead to several unexpected errors that must be managed effectively.

The productivity of these press machines is measured by the OEE, which is a relevant metric used in this domain to identify the percentage of planned production time that is truly productive [51]. The OEE is calculated as the multiplication of availability, performance and quality scores. Availability takes into account unplanned and planned stops. An availability score of 100% means that the process is constantly running during planned production time. Performance represents the percentage of the speed at which the industrial system is running considering the speed for which it was designed to run in optimal conditions, and takes into account slow cycles and small stops. A performance score of 100% means that the process is running as quickly as possible. Quality takes the manufactured pieces that do not meet quality standards into account, including pieces that are later reworked. A quality score of 100% means there are no defects (i.e., only good parts are being produced).

Therefore, Big Data and cloud computing can help to minimize the gap between the current situation of the companies and the ideal production scenario; that is, manufacturing only good parts (quality), as fast as possible (performance), with no unplanned stop time (availability). In fact, some studies have already demonstrated that a company can increase their productivity by using Big Data frameworks [46].

B. HYPOTHESIS

To measure the suitability of the proposed monitoring system and to determine whether it successfully passes the established tests, we defined the following hypothesis:

"*The developed real-time monitoring system is capable of detecting anomalies by processing data generated by press machines in a stable way, when the data volume is equal to the data generated in the current scenario and under conditions considered for future scenarios.*"

In this context, stable means that the application needs less time to process data than the duration of a batch (i.e., five seconds). Therefore, as the data received from the press machines increases in volume, the monitoring system would have to maintain stable by scaling its computational resources and, consequently, processing more data within the same period of time.

C. EVALUATION METRICS

In this section, we analyze the metrics used to measure the performance and the scalability of the developed real-time monitoring system. Different parameters provided by Spark Streaming have been used to validate the hypothesis:

- **Input rate:** the number of messages received per second.
- **Scheduling delay:** the time for which a batch waits in a queue until the processing of previous batches

is finished. For example, assuming that the monitoring agent reads from the source with a frequency of five seconds, and that the given batch took seven seconds to compute, then means the agent is two seconds behind ($7 - 5 = 2$), thus making the scheduling delay two seconds long.

- **Processing time:** the time to process each batch of data.
- **Delay time:** the time spent to complete all the streaming jobs of a batch. This is calculated by summing the scheduling delay and the processing time. Following the same example, if the agent is already two seconds behind and the processing of the next batch takes a further seven seconds, then the data will be processed with a total delay of nine seconds ($2 + 7 = 9$). These metrics are calculated for each batch. Therefore, since this parameter is calculated from the previous ones, it is used to measure the success of the test.

D. SCALABILITY TEST

To verify our hypothesis, three tests were conducted. Each test varied from the others in terms of the input rate. In addition, in the last test, the computational resources provided were also modified. The details of each test follow:

- **Test 1:** simulates the current scenario. This means that the same data volume as that generated by the sensors of the press machines in a normal scenario was used as the input rate. This implies a data ingestion of 40 messages per second.
- **Test 2:** currently, the number of press machines is low, although it is expected to grow significantly. Thus, data volume was increased to bring the monitoring system to its limit. The input rate was progressively increased until the processing time reached the duration of a batch and the total delay started to increase. The objective of this test was to find the maximum input rate supported by the application.
- **Test 3:** the same input rate as used as for Test 2 was established. However, the computational resources of the application were increased. Therefore, verification could be made as to whether or not the application is scalable and, consequently, whether it could process more data in a stable manner solely by adding computational resources.

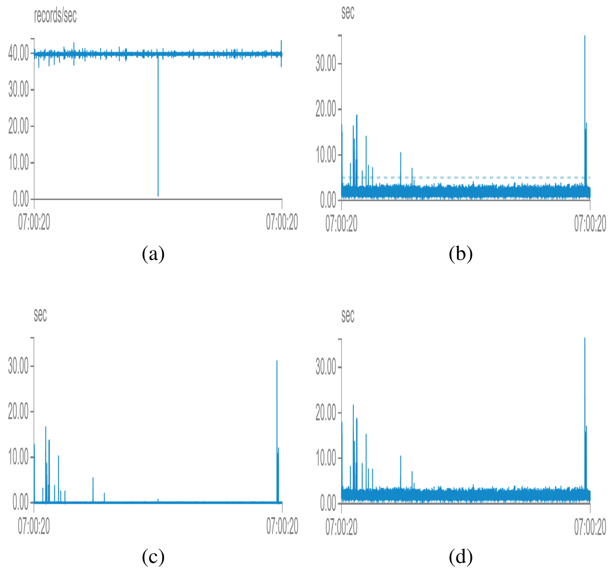
For Tests 1 and 3, the total delay must be lower than a threshold of 20 s to be considered successful. The duration of each test was one day; that is, 17.280 batches. Table 1 shows the configuration used for executing the tests. More specifically, the resources used in the drivers and executors are as follows:

TABLE 1. Computational resources used for Tests 1, 2 and 3.

	Driver		Executors		
	# cores	Memory	Amount	# cores	Memory
Test 1 & 2	2	4 GB	1	4	11 GB
Test 3	3	8 GB	3	6	11 GB

TABLE 2. Results of Tests 1, 2 and 3.

		Test 1	Test 2	Test 3
Input Rate		40 msg/s	4,000 msg/s	4,000 msg/s
Scheduling Delay	Min	0.00 s	174.00 s (2.90 min)	0.00 s
	Avg	0.03 s	387.60 s (6.50 h)	0.01 s
	Max	31.49 s	36,000.00 s (10 h)	4.15 s
Processing Time	Min	0.89 s	4.23 s	0.47 s
	Avg	1.68 s	5.56 s	2.19 s
	Max	31.25 s	27.47 s	13.53 s
Total Daly	Min	0.89 s	174.00 s (2.90 min)	1.56 s
	Avg	1.71 s	390.60 s (6.50 h)	2.19 s
	Max	32.13 s	36,000.00 s (10 h)	13.53 s



(a)

(b)

(c)

(d)

FIGURE 4. Test 1: a) input rate, b) scheduling delay, c) processing time, d) total delay.

V. RESULTS AND DISCUSSION

The results of Test 1 are shown in Figure 4 and summarized in Table 2. These results demonstrate that the monitoring system is stable during the computation for the gathered data (Figure 4a). It is worth pointing out that the used system to push data to the cloud was not designed for real-time purposes. Therefore, the input rate was not stable throughout time due to system overheads. As shown in Figure 4b, the processing time remains lower than the batch period. Moreover, as depicted in Figure 4c, the scheduling delay is almost zero, which means that almost no batches are enqueued before being processed. This implies that data processing is performed in real time and that the total delay (Figure 4d) is made up of processing time. Overall, the monitoring system requires 1.71 s, on average, between gathering the data coming from press machines and providing a result indicating their status. Taking into account the condition defining success and the obtained results, Test 1 was passed satisfactorily.

Similarly, the results of Test 2 are represented in Figure 5 and summarized in Table 2. These results show that the limit of the monitoring system, for the given computational resources, is 4,000 messages per second, as can be seen

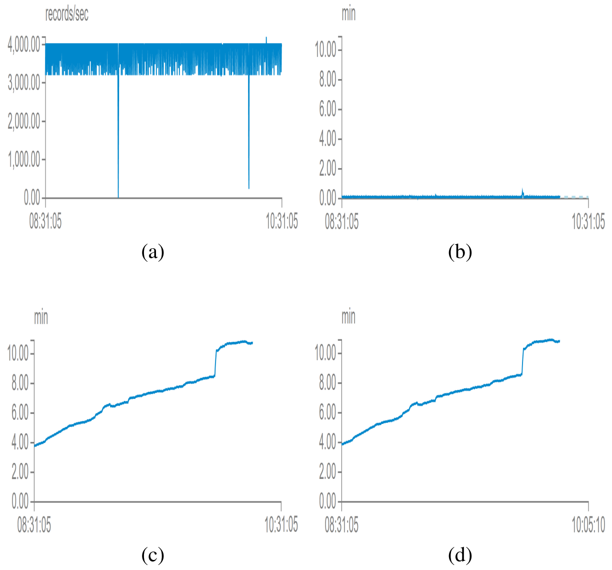


FIGURE 5. Test 2: a) input rate, b) scheduling delay, c) processing time, d) total delay.

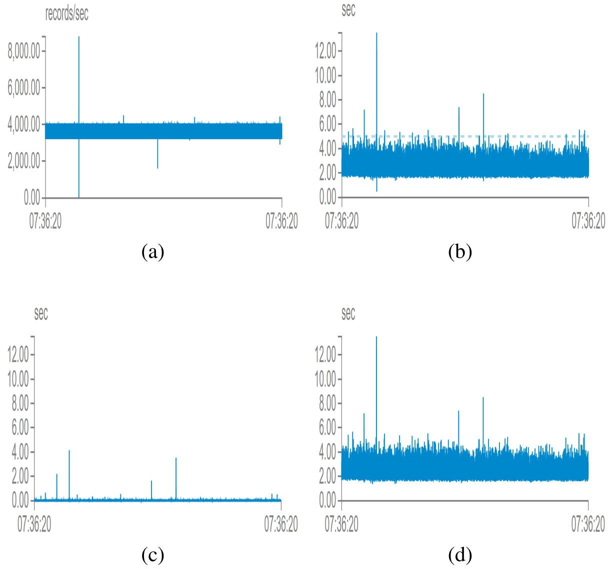


FIGURE 6. Test 3: a) input rate, b) scheduling delay, c) processing time, d) total delay.

in Figure 5a. As shown in Figure 5b, the processing time borders on the batch time, as its average is 5.56 s. Each time the processing time exceeds the batch time threshold, this implies a delay time. Moreover, the scheduling delay (Figure 5c) increases due to the execution of other data processing tasks, such as persisting data. This combination makes the total delay (Figure 5d) too large to be considered to be a fast response, since detecting an anomaly so late would be critical. Otherwise, the scale of the graph would have been too large to observe these values. The rest of the representation follows the same pattern.

Once the maximum input rate for the given computational resources was known, Test 3 was executed. The results are presented in Figure 6 and summarized in Table 2. These results confirm the scalability of the developed monitoring system, as it was able to process the volume of data forming the limit in the previous test (Figure 6a) in a stable way. As shown in Figure 6b, the increase of computational resources implied faster data processing than in Test 2. Therefore, almost no batches are enqueued (Figure 6c) and, consequently, the total delay is almost equal to the processing time, as shown in Figure 6d. Thus, according to the condition defined for success, Test 3 was passed successfully.

These tests show that the monitoring system satisfies all of the requirements of the hypothesis: first, the system is capable of detecting anomalies; second, the data processing is stable under the current and future scenarios, as shown in Tests 1 and 2; and finally, the monitoring system is scalable, as it can handle future demand for data volume by scaling its computational resources, as shown in Tests 2 and 3.

The implementation of this solution, for this particular use case, has led to several enhancements in the maintenance service. The combination of a fast, scalable, and fault-tolerant real-time monitoring system with an effective feedback system to manage the anomalies has improved the OEE. However, as it has only been a short time since this solution was implemented, there is an absence of qualitative and quantitative results from an empirical application and/or validation. Therefore, it is difficult to evaluate the potential of the proposed solution with respect to its usability and/or usefulness for industry adoption. This is a general problem when implementing this type of solution in the industry [52]. Preliminary studies made by the clients show the adequacy and the correctness of this implementation. Nonetheless, an exhaustive analysis of the monitoring system will be done once the system has been in production long enough to obtain sufficient quantitative data to measure the real gain.

VI. CONCLUSIONS AND FUTURE WORK

This work presents a Big Data solution for the real-time monitoring of for ICPSs which is validated on a real industrial scenario where several press machines are monitored. The proposed solution demonstrates the potential of Big Data technologies in an industrial scenario where the volumes of data generated are very large, and unexpected failures must be managed without affecting the proper operation of the monitoring system. Therefore, this work uses fast, scalable and fault-tolerant data acquisition and data processing systems. In addition, a dashboard is developed to visualize the performance of industrial machines and detected anomalies.

The experimental results obtained in the industrial use case show that the application exceeds the current needs of the monitoring system, since the data processing remains stable for the data volume generated in the current scenario. Although the limit of this application was reached in Test 2, its scalability is demonstrated in Test 3, since it allows this volume (or more) of data to be processed simply by adding more computational resources. Furthermore, this implementation has improved the OEE.

In this work, a platform is developed where data related to the performance of industrial machines is processed. Hence, machines can be monitored to effectively detect anomalies. However, this platform opens new possibilities to improve the maintenance strategy, from fault diagnosis to failure prognosis [53], [54]. This is made possible by applying data mining algorithms to the historical data that is already stored on the database. Thus, instead of detecting an imminent failure, it can be predicted and early repair work can be done. This will increase the lifespan of the company's systems, improve their availability and reliability, and this will directly affect productivity [55]. In addition, these strategies will reduce operational and maintenance costs.

ACKNOWLEDGEMENTS

The authors are grateful to Goizper and Fagor Arrasate for providing the industrial case study, and specifically Jon Rodriguez and David Chico (Fagor Arrasate) for their help and support. Any opinions, findings and conclusions expressed in this article are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] I. Yaqoob *et al.*, "Internet of Things architecture: Recent advances, taxonomy, requirements, and open challenges," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 10–16, Jun. 2017.

[2] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," *Int. J. Adv. Manuf. Technol.*, vol. 94, nos. 9–12, pp. 3563–3576, Feb. 2018.

[3] G.-J. Cheng, L.-T. Liu, X.-J. Qiang, and Y. Liu, "Industry 4.0 development and application of intelligent manufacturing," in *Proc. Int. Conf. Inf. Syst. Artif. Intell. (ISAI)*, Jun. 2016, pp. 407–410.

[4] F. Tao, Q. Qi, A. Liu, and A. Kusiak, "Data-driven smart manufacturing," *J. Manuf. Syst.*, vol. 48, pp. 157–169, Jul. 2018.

[5] F. Tao and Q. Qi, "New IT driven service-oriented smart manufacturing: Framework and characteristics," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 1, pp. 81–91, Jan. 2019.

[6] A. W. Colombo, S. Karnouskos, and T. Bangemann, *Towards the Next Generation of Industrial Cyber-Physical Systems*. Cham, Switzerland: Springer, 2014, pp. 1–22.

[7] X. Yue, H. Cai, H. Yan, C. Zou, and K. Zhou, "Cloud-assisted industrial cyber-physical systems: An insight," *Microprocess. Microsyst.*, vol. 39, no. 8, pp. 1262–1270, Nov. 2015.

[8] Y. Lu and X. Xu, "Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services," *Robot. Comput.-Integr. Manuf.*, vol. 57, pp. 92–102, Jun. 2019.

[9] R. Atat, L. Liu, J. Wu, G. Li, C. Ye, and Y. Yang, "Big data meet cyber-physical systems: A panoramic survey," *IEEE Access*, vol. 6, pp. 73603–73636, 2018.

[10] R. B. Faiz and E. A. Edirisinghe, "Decision making for predictive maintenance in asset information management," *Interdiscipl. J. Inf. Knowl. Manage.*, vol. 4, pp. 23–36, Jan. 2009.

[11] Y. Xun, J. Zhang, X. Qin, and X. Zhao, "FiDoop-DP: Data partitioning in frequent itemset mining on Hadoop clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 101–114, Jan. 2017.

[12] Y. Cheng, Y. Zhang, P. Ji, W. Xu, Z. Zhou, and F. Tao, "Cyber-physical integration for moving digital factories forward towards smart manufacturing: A survey," *Int. J. Adv. Manuf. Technol.*, vol. 97, nos. 1–4, pp. 1209–1221, Jul. 2018.

[13] R. Baheti and H. Gill, "Cyber-physical systems," *Impact Control Technol.*, vol. 12, pp. 161–166, Mar. 2011.

[14] P. J. Antsaklis and V. Gupta, "Control of cyber-physical systems workshop," Tech. Rep., Oct. 2012.

[15] L. Monostori *et al.*, "Cyber-physical systems in manufacturing," *CIRP Ann.*, vol. 65, no. 2, pp. 621–641, 2016.

[16] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Comput. Ind.*, vol. 81, pp. 11–25, Sep. 2016.

[17] R. Harrison, D. Vera, and B. Ahmad, "Engineering methods and tools for cyber-physical automation systems," *Proc. IEEE*, vol. 104, no. 5, pp. 973–985, May 2016.

[18] I. Gerostathopoulos *et al.*, "Self-adaptation in software-intensive cyber-physical systems: From system goals to architecture configurations," *J. Syst. Softw.*, vol. 122, pp. 378–397, Dec. 2016.

[19] T. Yue, S. Ali, and B. Selic, "Cyber-physical system product line engineering: Comprehensive domain analysis and experience report," in *Proc. 19th Int. Conf. Softw. Product Line*, 2015, pp. 338–347.

[20] J. C. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and J. Zou, "Distributed real-time software for cyber-physical systems," *Proc. IEEE*, vol. 100, no. 1, pp. 45–59, Jan. 2012.

[21] A. N. Narayanan, R. Ak, Y.-T. T. Lee, R. Ghosh, and S. Rachuri, "Summary of the symposium on data analytics for advanced manufacturing," NIST, Gaithersburg, MD, USA, Appl. Note 100-7, 2017.

[22] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. P. van der Aalst, "Compliance monitoring in business processes: Functionalities, application, and tool-support," *Inf. Syst.*, vol. 54, pp. 209–234, Dec. 2015.

[23] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[24] L. Wilber, "A practical guide to big data: Opportunities, challenges & tools," Tech. Rep., 2012. [Online]. Available: https://www.3ds.com/products-services/exalead/resources/woc=%7B%22brand%?22%3A%5B%22brand%2Fexalead%22%5D%7D&wocset=5

[25] Z. Chen, X. Zhang, and K. He, "Research on the technical architecture for building CPS and its application on a mobile phone factory," in *Proc. 5th Int. Conf. Enterprise Syst. (ES)*, Beijing, China, Sep. 2017, pp. 76–84.

[26] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manuf. Lett.*, vol. 3, pp. 18–23, Jan. 2015.

[27] O. Niggemann, G. Biswas, J. S. Kinnebrew, H. Khorasgani, S. Volgmann, and A. Bunte, "Data-driven monitoring of cyber-physical systems leveraging on big data and the Internet-of-Things for diagnosis and control," in *Proc. 26th Int. Workshop Princ. Diagnosis*, 2015, pp. 185–192.

[28] R. F. Babiceanu and R. Seker, "Big data and virtualization for manufacturing cyber-physical systems: A survey of the current status and future outlook," *Comput. Ind.*, vol. 81, pp. 128–137, Sep. 2016.

[29] O. Battaïa, A. Otto, F. Sgarbossa, and E. Pesch, "Future trends in management and operation of assembly systems: From customized assembly systems to cyber-physical systems," *Omega*, vol. 78, pp. 1–4, Jul. 2018.

[30] D. Serpanos, "The cyber-physical systems revolution," *Computer*, vol. 51, no. 3, pp. 70–73, Mar. 2018.

[31] P. Zheng *et al.*, "Smart manufacturing systems for industry 4.0: Conceptual framework, scenarios, and future perspectives," *Frontiers Mech. Eng.*, vol. 13, no. 2, pp. 137–150, 2018.

[32] I. P. Egwutuoha, D. Levy, B. Selic, and S. Chen, "A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems," *J. Supercomput.*, vol. 65, no. 3, pp. 1302–1326, 2013.

[33] T. S. J. Darwish and K. A. Bakar, "Fog based intelligent transportation big data analytics in the Internet of vehicles environment: Motivations, architecture, challenges, and critical issues," *IEEE Access*, vol. 6, pp. 15679–15701, 2018.

[34] M. V. Chavan and R. N. Phursule, "Survey paper on big data," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 6, pp. 7932–7939, 2014.

[35] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of 'big data' on cloud computing: Review and open research issues," *Inf. Syst.*, vol. 47, pp. 98–115, Jan. 2015.

[36] C. Gormley and Z. Tong, *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*. Newton, MA, USA: O'Reilly, 2015.

[37] D. Chen *et al.*, "Real-time or near real-time persisting daily healthcare data into HDFS and elasticsearch index inside a big data platform," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 595–606, Apr. 2017.

[38] A. Balalaie, A. Heydarnoori, and P. Jamshidi, ''Microservices architecture enables devOps: Migration to a cloud-native architecture,'' *IEEE Softw.*, vol. 33, no. 3, pp. 42–52, May /Jun. 2016.

[39] P. P. I. Langi, Widyawan, W. Najib, and T. B. Aji, ''An evaluation of Twitter river and Logstash performances as elasticsearch inputs for social media analysis of Twitter,'' in *Proc. Int. Conf. Inf. Commun. Technol. Syst. (ICTS)*, Sep. 2015, pp. 181–186. doi: 10.1109/ICTS.2015.7379895.

[40] A. De Dios Fuente, O. Ø. Andreassen, and C. Charrondière, ''Monitoring mixed-language applications with elastic search logstash and kibana (ELK),'' in *Proc. ICALEPCS*, Dec. 2015, pp. 9–12.

[41] P. B. Makeshwar, A. Kalra, N. S. Rajput, and K. P. Singh, ''Computational scalability with apache flume and mahout for large scale round the clock analysis of sensor network data,'' in *Proc. Nat. Conf. Recent Adv. Electron. Comput. Eng. (RAECE)*, Feb. 2015, pp. 306–311.

[42] J. Kreps, N. Narkhede, and J. Rao, ''Kafka: A distributed messaging system for log processing,'' in *Proc. NetDB*, 2011, pp. 1–7.

[43] H. Yoon, S. Kim, T. Nam, and J. Kim, ''Dynamic flow steering for IoT monitoring data in SDN-coordinated IoT-cloud services,'' in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 625–627.

[44] J. Park and S.-Y. Chi, ''An implementation of a high throughput data ingestion system for machine logs in manufacturing industry,'' in *Proc. 8th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2016, pp. 117–120.

[45] S. Zhao, M. Chandrashekar, Y. Lee, and D. Medhi, ''Real-time network anomaly detection system using machine learning,'' in *Proc. 11th Int. Conf. Des. Reliable Commun. Netw. (DRCN)*, Mar. 2015, pp. 267–270.

[46] F. Provost and T. Fawcett, ''Data science and its relationship to big data and data-driven decision making,'' *Big Data*, vol. 1, no. 1, pp. 51–59, 2013.

[47] B. Hindman *et al.*, ''Mesos: A platform for fine-grained resource sharing in the data center,'' in *Proc. 8th USENIX Conf. Netw. Syst. Design Implement.*, 2011, pp. 295–308.

[48] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, ''ZooKeeper: Wait-free coordination for Internet-scale systems,'' in *Proc. USENIX Annu. Tech. Conf.*, vol. 8, 2010, p. 11.

[49] K. Ranjithprabhu and D. Sasirega, ''Eliminating single point of failure and data loss in cloud computing,'' *Int. J. Sci. Res.*, vol. 3, no. 4, pp. 2319–7064, 2014.

[50] A. Iglesias, H. Lu, C. Arellano, T. Yue, S. Ali, and G. Sagardui, ''Product line engineering of monitoring functionality in industrial cyber-physical systems: A domain analysis,'' in *Proc. 21st Int. Syst. Softw. Product Line Conf.*, Spain, Sep. 2017, pp. 195–204.

[51] G. Hwang, J. Lee, J. Park, and T.-W. Chang, ''Developing performance measurement system for Internet of Things and smart factory environment,'' *Int. J. Prod. Res.*, vol. 55, no. 9, pp. 2590–2602, 2017.

[52] M. Khurum and T. Gorschek, ''A systematic review of domain analysis solutions for product lines,'' *J. Syst. Softw.*, vol. 82, no. 12, pp. 1982–2003, 2009.

[53] J. Yan, Y. Meng, L. Lu, and L. Li, ''Industrial big data in an industry 4.0 environment: Challenges, schemes, and applications for predictive maintenance,'' *IEEE Access*, vol. 5, pp. 23484–23491, 2017.

[54] M. Marjani *et al.*, ''Big IoT data analytics: Architecture, opportunities, and open research challenges,'' *IEEE Access*, vol. 5, pp. 5247–5261, 2017.

[55] H. M. Hashemian, ''State-of-the-art predictive maintenance techniques,'' *IEEE Trans. Instrum. Meas.*, vol. 60, no. 1, pp. 226–236, Jan. 2011.

MIKEL CANIZO graduated in computer engineering from Mondragon University, in 2014, and the master's degree in computer engineering from the University of Deusto, Bilbao, in collaboration with the Ikerlan Research Center, in 2016. He is currently pursuing the Ph.D. degree with the IK4-Ikerlan Research Center, Spain. His research interests include big data analytics and machine learning, concretely in deep learning.

ANGEL CONDE received the Ph.D. degree in computer science from the University of the Basque Country, in 2016. He is currently a Researcher with IK4-Ikerlan, and collaborates with the GaLan Research Group, the Group for Adaptive Teaching-Learning Environment, and the University of the Basque Country UPV/EHU.

SANTIAGO CHARRAMENDIETA has been a Researcher with the IK4-Ikerlan Research Center, since 1997. He has complemented his work with the use of these techniques for smart maintenance systems in a cloud computing environment. Prior to that, he has focused his research activities on the engineering of railway control embedded systems based on a model-driven product-line approach. His research interests include distributed systems, cloud computing, big data, workflows, and machine learning, with a special emphasis on new manufacturing (Industry 4.0) and cyber-physical systems. He has led several R&D projects funded by the Spanish/Basque Governments and has participated in European H2020 projects as well. He is currently coordinating a research line on big data architectures for manufacturing scenarios.

RAÚL MIÑÓN received the M.Eng. degree in computer science from the University of the Basque Country (UPV/EHU), in 2008, and the Ph.D. degree from the Egokituz Laboratory, in 2015. He worked in a computer science consultancy in the area of Web development as a Programmer, an Analyst, and an Assistant in training courses for two years in Bilbao. He was granted with a Doctoral Fellowship from the Basque Government. He also did research stay collaboration in the Laboratory on Human Interfaces in Information Systems, in 2013. His thesis research topic was the automatic generation of accessible user interfaces taking people specific needs into account. Subsequently, he worked as an entrepreneur founding a company, called MalguTech, aimed to personalize automatically other companies' Web sites to the needs of their users. Next, he was a Researcher in IK4-Ikerlan in the Big Data Architecture Team for two years. Currently, he continues conducting big data projects in Tecnalia Research and Innovation.

RAUL G. CID-FUENTES received the dual M.Sc. degrees in electrical engineering and in electrical and computer engineering and the Ph.D. degree *(cum laude)* in electrical engineering from the Universitat Politècnica de Catalunya, in 2011 and 2016, respectively. He was a Visiting Scholar with the Georgia Institute of Technology, in 2011, under the guidance of Prof. Dr. I. F. Akyldiz, and with Northeastern University, in 2015. He was a Postdoctoral Researcher and a Project Manager with the Universitat Politècnica de Catalunya, in 2016, and a Researcher and Project Manager with IK4-IKERLAN, from 2016 to 2018. He has been with Telefónica I+D, since 2018, where he is the Technical Leader and Product Manager of a new initiative in IoT Security. He has coauthored more than 20 papers in international conferences and journals. His research interests include the Internet of Things, energy harvesting, and wireless networks. He received numerous awards, such as the Special Doctorate Award, in 2018, the Best In-Session Presentation Award at the IEEE INFOCOM 2016, and the Best Student of the M.Sc. in EE, in 2011.

ENRIQUE ONIEVA is currently a Professor in artificial intelligence, machine learning, and big data with the University of Deusto, as well as a Researcher of intelligent transportation systems applications related to data processing, mobility, and smart city solutions in the DeustoTech-Mobility Research Unit. He is also the Director of the Big Data Executive Program and the Ph.D. Program in engineering for the Information Society and Sustainable Development of the Faculty of Engineering. He has participated in more than 25 research projects, including, CYBERCARS-2 (FP6), ICSI (FP7), and PostLowCit (CEF-Transport). His research is responsible for the Artificial Intelligence Work Package of the Project TIMON (H2020) and a Project Coordinator of the LOGISTAR Project (H2020). He has authored more than 100 scientific articles. Among them, more than 40 are published in journals of the highest level. His research has been recognized and awarded several times in international conferences, and his publications got an H-index of 17 (scopus.com). Currently, he is one of the most prolific researchers in his area. His research interest includes the application of artificial intelligence to intelligent transportation systems, including fuzzy-logic-based decision, evolutionary optimization, and machine learning.