Development of an Industry 4.0 Big Data Processing and Management System

Francisco Vidal Cabrita Carneiro
francisco.carneiro@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

December 2020

Abstract

Industry 4.0, also known as the fourth industrial revolution, refers to the enhancement of automation, connectivity and intelligence of machines within a production environment. Its objectives are increasing reactiveness, asset monitoring, decision making and value creation of manufacturing. Industry 4.0 solutions are however challenging to implement as they rely on data systems capable of handling massive amounts of data with low latency, high delivery guarantee, high fault tolerance and high information throughput. This generally implies satisfying Big Data, cyber physical system, Internet of Things and distributed computing system constraints. Understanding the high potential associated to Industry 4.0 solutions, AKKA Technologies, an Engineering and Technology consulting firm, decided to launch project ZORRO who's initial aim was to develop a data system capable of detecting manufacturing anomalies in real-time and to serve as a source of internal Industry 4.0 know-how. As a contribution to project ZORRO, in this work an Industry 4.0 Big Data solution, capable of satisfying Industry 4.0 and project ZORRO's objectives for automatic anomaly detection, is proposed providing end-point solutions consisting in machine learning platforms for automatic anomaly detection, data visualisation platforms for real-time dashboarding and monitoring, data science tools for extraction of important production insights, and batch and stream processing engines to implement virtual sensors, online predictions and other real-time calculations. The architecture was designed to respond to the various constraints associated with these types of Big Data systems whilst remaining modular, affordable using commodity hardware, scalable and microservice based.
Keywords: Industry 4.0, Big Data, Industrial Internet of Things (IIoT), Stream Processing, Real-time Anomaly Detection.

1. Introduction

In an increasingly digital world, most complex and large-scale production lines have adopted what is called an Industry 3.0 method, where machines, electronics and IT systems are aiding factories in increasing the speed, efficiency, capacity and flexibility of production through automation. Industry 3.0 gives way for monitoring and connecting these machines and systems into a network known as an Industrial Internet of Things (IIoT) environment or Industry 4.0. An Industry 4.0 solution consists in creating a connected network of machines, products and processes that make up the production context, monitoring and or controlling their status, and through this extract value and drive effective actions and decisions. Access to the information from this network allows for real-time production anomaly detection, equipment predictive maintenance and general forecasting, process optimisation, cost accounting and supply/value chain resilience.

Industry 4.0 promises manufacturers additional annual revenues of 2% to 3% on average [1]. Despite these clear advantages according to McKinsey Digital [2], only 30% of technology suppliers and 16% of manufacturers have an overall Industry 4.0 strategy. McKinsey suggested that the main implementation barriers were difficulties in coordinating actions across different organisational units, concerns about cybersecurity, lack of courage to push through a radical transformation and difficulties to adopt new technologies. These results are understandable since Industry 4.0 solutions require the implementation of complex data systems capable handling huge amounts of data (Big Data), complex network topologies (IoT) and large computational capacities (cloud computing) [3].

In this context, AKKA Technologies, an Engineering and Technology consulting firm, launched a project named ZORRO with the initial aim of creating a system capable of detecting manufacturing anomalies in real-time in an Industry 4.0 scenario. With this project, AKKA also wishes to equip itself

with the know-how required to advise customers on what an Industry 4.0 and Big Data solution consists of (in terms of architecture, technology and cost), but also understand what added value it can give manufacturers. The ZORRO system alone does not have the objective of being sold as a product. Nevertheless, ZORRO still aims in becoming a Meta solution, to be used as an Industry 4.0 internal reference for AKKA, capable of satisfying Industry 4.0 objectives and overcoming the constraints associated to these systems.

As a contribution to the ZORRO project, in this work, a Big Data system for the collection, storage and processing of real-time IIoT type data was implemented. The system was conceived with the following features:

- Data visualization / dashboarding tools supporting dynamic and interactive queries.
- Real-time (streaming) and batch data processing system.
- Machine Learning (ML) and Data Science platforms for data exploration, data analytics and general insight extraction.
- Real-time anomaly detection, and alerting system, through ML algorithms.

The solution is designed to address the Big Data constraints, data streaming constraints involving message latency and message throughput whilst guaranteeing the real-time analysis and reactiveness that Industry 4.0 promises. Figure 1 illustrates a high-level representation of the implemented system.

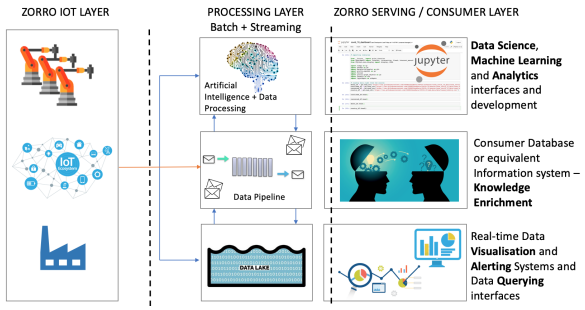


Figure 1: High-level representation of implemented system.

The ZORRO system was divided into three main layers: the IoT data production layer, the Processing layer and the serving layer. The IoT layer represents the physical data production layer. In the context of this project it was simulated by a Python coded factory simulator. The processing layer allows for real-time and batch data processing, ML, insight extraction and data storage. Anomaly detection algorithms, trained using ML, were implemented and deployed here. On the serving layer, a series of tools were developed to extract value from the developed platform. This layer includes real-time data visualisation and dynamic querying platforms, data science and ML development environments, and automatic knowledge enrichment features including alerts and creation of virtual sensors.

In this paper we shall present the work carried out as follows. Section 2, describes the background and context of this work and project ZORRO. The technologies chosen and the main conclusions drawn from the state-of-the-art study are described. Section 3 focuses on the engineering work done discussing in more detail how the work's mission and objectives were accomplished and what systems were added, improved, deployed and tested. The results that were obtained in terms of performance and system outputs are presented in section . Section 5 concludes with a review of the work done by analysing what was accomplished, what can be improved and what were gains that were obtained from this project.

2. Background

2.1. Project ZORRO

This work was carried out in the context of an internship at AKKA Technologies. AKKA Technologies is an engineering and technology consulting group founded in 1984. The firm is positioned in a series of technological sectors most notably: aeronautic, automotive, energy, information systems and telecommunications. Motivated by a clear shift in market demands, AKKA wishes to expand and diversify its operations towards the data and digital domains.

Having understood the high potential of Industry 4.0 for increasing efficiency of production environments and wanting to increase its expertise on digital transformation and data solutions, the ZORRO project was created at AKKA. This project emerged in the context of an European H2020 project with the objective of responding to the demand for European Industrial efficiency and competitiveness.

Project ZORRO's main goal is to develop an Industrial IoT (IIoT) data platform solution in a multi-production-line and multi-stage factory context, capable of detecting production anomalies, and other useful insights, in real-time and attaining zero-defect production. These production environments can produce large amounts of data from various streaming data sources and thus, ZORRO is an IIoT Big Data scale data processing platform, Figure 2.

2.2. Industry 4.0 Today

Although today there is no standard on how to perform Industry 4.0, robust architecture proposals are

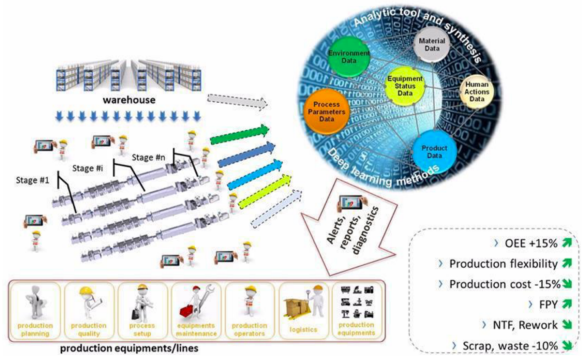


Figure 2: ZORRO industrial data platform solution.

starting to rise. One such proposal can be found in [4] where a 5 layer Industry 4.0 architecture is described as illustrated in Figure 3.

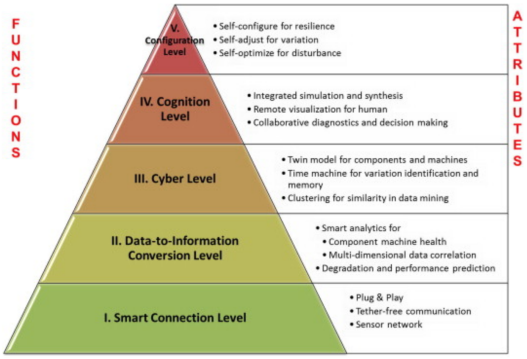


Figure 3: Industry 4.0 proposed architecture [4].

The layers can essentially be grouped into two main components, a Cyber Physical System (CPS) with advanced connectivity and real-time data acquisition system and an intelligent data management system who's objective is to leverage the data production and connectivity of the physical level to extract useful insights from the data and perform data-driven decisions. By Analysing Figure 3, we can already better understand why Industry 4.0 is difficult and complex to implement as it requires expertise on the physical and IoT domain, on the data engineering domain due to the complexity associated to Big Data, data streaming and distributed computing systems (cloud or edge computing), and on the data science domain as algorithms for machine learning, analytics and insight extraction need to be employed to be able to extract value from the data.

In spite of the difficulties associated to Industry 4.0, manufacturing companies require it to increase the quality and speed of production. In the Aeronautics sector for example, airbus has delivered in the past 40 years 10,000 aircraft's and aims in delivering 20,000 aircraft's in the next 20 years (2019 numbers before the COVID-19 pandemic). To achieve this goal, airbus is investing in Industry 4.0 technologies such as the mixed reality [5] and Big Data platforms [6], due to the fact that in Aerospace manufacturing, an increase in quality is an increase in safety.

2.3. Big Data and Cloud Computing Platforms

Data processing platforms are an essential component Industry 4.0 systems as they contribute to layers II, III and IV of Figure 3. In order to build these industrial Big Data processing systems in a scalable, felxible and affordable way, large amounts of on demand computing resources are necessary. In the context of Industry 4.0, cloud computing has become a promising solution [7].

Cloud computing refers to the availability on demand of computer system resources (data storage, servers, networks, applications). Big Data applications require a microservice based architecture with fast and flexible scalability. Cloud computing platforms allow for this as they manage large data storage and processing centres with flexible virtualisation tools.

Some of the Big Cloud providers such as Amazon Web Services, Microsoft Azure and Google Cloud Platform, also offer IoT, Big Data and Artificial Intelligence platform solutions for their customers, which are precisely the type of solutions that ZORRO aims to provide. These providers are therefore ideal candidates for a state-of-the-art reference for this project.

By studying the solutions developed by these cloud giants, the high-level architecture and building blocks of IoT and data platform solutions became clearer. From this study, an initial logical high-level architecture was proposed. It was decided that the system must be composed of the following generic sub-systems: IoT data measurement and production system; data collection and pipeline system (message broker); long term data storage system; data processing and analytics system; data visualisation and insight extraction system. This high-level architecture is illustrated in Figure 4.

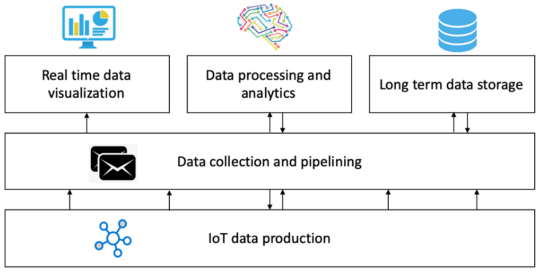
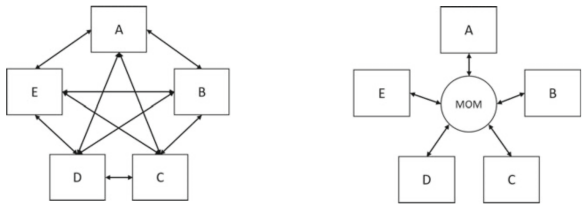


Figure 4: High level ZORRO architecture.

In the following sections we will described the individual technologies chosen to build each one of these sub-systems.

2.4. Data Pipeline - Message Broker

Message broker systems, or message orientated middleware (MOM), are systems that receive and temporarily retain data from various producers (publishers) and redirect the data to consumers (subscribers) that wish to consume it. The concept behind MOMs is illustrated in Figure 5.b. This type of interaction is known as publish/subscribe communication. It is an alternative to creating dedicated unique communication channels, like in Figure 5.a, where not only the number of communication links is significantly larger, but it will shift part of the communication burden to the consumers and producers abstracting communications while making interoperability possible [8]. Concerning Industry 4.0 cyber-physical systems, MOMs are communication protocol agnostic, making the adoption of different machine protocols easier [9]. They allow for entities in the production environment to produce data, therefore making their status known, and consume data allowing for them to be controlled.



(a) Communication without a MOM.

(b) Concept of MOM.

Figure 5: MOM interest.

There are many message brokers available today. Examples include Kafka, RabbitMQ, HiveMQ, Mosquito etc. With regards to message brokerig at Big Data scales, Apache Kafka is a reference [10]. Kafka was originally developed in LinkedIn to become the unified central data pipeline of the application. Kafka is a distributed publish-subscribe based durable messaging system. It is distributed because there is no central MOM (broker) entity, instead, it is composed of a cluster of brokers. In Kafka, messages are key-value pairs that are stored in topics. Topics are class like groups of messages managed by the brokers. Data is written to disk in Kafka, but only for a specified retention period. This data retention is what allows Kafka to provide data replication and therefore high availability and fault tolerance. Producers publish messages into specific topics. If a consumer is interested in data from a topic, it can subscribe to this topic and pull the data from it. Topics are divided into partitions allowing for data to be consumed in parallel.

2.5. Data Processing System

In the context of Big Data systems, a data processing system is a distributed processing system of Big Data sources. It is distributed in nature as it is not possible to efficiently process Big Data sources in a single node system. Examples of data processing systems include Hadoop (Hive), Spark, Flink and Storm. Traditionally these systems were used for batch processing only. Nevertheless, the world of Big Data has recently extended to stream processing as well.

One of the most popular go to options for distributed and large-scale streaming and batch data-processing is Apache Spark. Spark is a unified multi-purpose stack for carrying out distributed calculations and large scale data-processing. At its core, Spark is a computing engine (built on a clustered computer system). Spark offers rich libraries allowing for SQL, machine learning, graphing and streaming specific operations on data.

A Spark cluster consists of a master (cluster manager) and worker nodes. The master node is responsible for resource allocation and task scheduling on the workers. As a distributed system, Spark also guarantees fault tolerance in its executions. As of its second major release, Spark 2, Spark introduced streaming libraries allowing for the processing of data in real-time. Likewise, Spark's original ML library, MLlib, has evolved to Spark ML which is faster and more data science friendly. Overall, Spark's wide adoption, popularity, libraries and performance makes it an excellent choice for Big Data processing, analytics and machine learning.

2.6. Time Series System

One of the priorities of the ZORRO data storage system is to be optimised for large scale timestamped data. Traditional relational databases are not adapted for time series data mainly due to scale, as time series data piles up fast, but also in terms of timestamp orientated queries (data stored in time-ascending order). Examples of time series databases include TimescaleDB, OpenTSDB, TICK stack, Graphite and others. After investigating these different solutions the TICK stack was found to be the leader in terms of performance when it came to time series databases [11]. The central piece of the TICK stack is the time series database InfluxDB (I). InfluxDB is a NoSQL database purpose built for time series data. Data is ingested into InfluxDB through their metrics collection agent, Telegraf (T). Data visualisation capabilities through dynamic dashboards and interactive queries is provided by Chronograf (C). Finally, Kapacitor (K) is the real-time data processing engine for time series data.

2.7. Persistent Big Data Storage System

In order to be able to store large amounts of data and process it efficiently using data processing systems (like Spark), Big Data information systems require special types of databases. When it comes to Big Data systems, traditional relational databases are built in such a way that they have limitations in terms of scalability, fault tolerance and query speed. This is mainly due to the fact that relational databases are designed to have data stored in one node (no partitioning of data).

When investigating popular Big Data storage systems, many options are possible: Cassandra, MongoDB, HBase and Neo4j. In this work, Apache Cassandra was chosen as the main persistent storage database solution because it significantly leads in Big Data storage performance both in terms of throughput and operation speed [12].

Cassandra is a distributed NoSQL database system, originally developed at Facebook. It was designed to satisfy: full scalable multi-master database replication and global availability at low latency. All data in Cassandra is associated to a Token. The nodes in a Cassandra cluster share the responsibility of data storage and occupy themselves with data from multiple ranges of token values. When new nodes are added, the token ranges are simply further partitioned and free token ranges are given to the new nodes.

2.8. Deployment and Virtualisation

Having chosen the main technologies to be used in each one of the sub-systems, the next question that must be answered is how to deploy these technologies. When it comes to deployment of Big Data information systems, or most modern applications, containerised deployments are the most popular. Containers are essentially isolated processes that contain a namespace (Bins/libs) and access to a limited amount of hardware resources (control group). Containers allow for a microservice and scalable based architecture as they are lightweight (easy to bring up and down) and stateless (fault tolerant). To create, deploy and manage containers, the Docker is the go-to container engine technology as they are the original creators of containers as we know them today.

2.9. Automatic Anomaly Detection

As seen so far, Industry 4.0 systems process data and detect events to perform a series of actions: anomaly detection, predictive maintenance, supply chain resilience, machine control etc. A majority of these events are inferred from numerous sensors and their complex relationships. In order to process this data and act at the desired speed, automatic and intelligent algorithms are required.

In the ZORRO project, the first mission is to be able to detect production anomalies in real-time. This was done using Machine Learning (ML), in particular supervised learning. In supervised learning, the algorithm uses data from a dataset (training dataset) containing input-label pairs $(x_1, y_1), ...(x_n, y_n)$, with input $x_n \in X$, and observed labels (outputs) $y_n \in Y$, also quantitative or qualitative, to create a model $\hat{f}$ such that:

$$Y = \hat{f}(X) + \epsilon \tag{1}$$

Where ϵ represents the measurement noise or error made by the model. The objective of the model $\hat{f}$ is therefore to attempt to explain the variable Y based on the observed features, X. The quality of the model $\hat{f}$ is determined by measuring the prediction error ϵ. In the context of the ZORRO project, X is the various sensor features of the machines on the production environment and the value we wish to predict, Y is the status of the machine (faulty 1 or not faulty 0).

Focusing on the algorithms offered directly by Apache Spark's ML library, we chose to study the following algorithms: **Decision Tree**, **Random Forest**, **Logistic Regression** and **Support Vector Machine** (SVM). All these algorithms are methods used for defining the function $\hat{f}$ described above.

3. Implementation

In this section we will describe the engineering work done in terms of the implementations developed.

3.1. Deployment Strategy

Before building the system based on Figure 1, and the chosen technologies, the deployment and virtualisation strategy was defined so that a consistent deployment model was follow throughout the project. As described in section 2.8, the desired deployment methodology is containers. In order to be able to define deployment states docker-compose was used. The idea behind docker-compose is to define a desired container deployment (state) for one host OS through a configuration .yaml file. This file can be used to define containers, the configurations and the environment variables to be passed to them as well as the networks the containers are in and the (data) volumes they are connected to. A docker-compose file was therefore defined for each sub-system in ZORRO. All machines used in this internship were OpenStack virtual machines, from AKKA's internal cloud. These machines contained 2 CPUs, an Ubuntu operating system and 4 or 6 GB of RAM. We label these machines as either type 1 or 2 respectively. The machines belonged all to the same network with 1.5 GB/s of bandwidth.

3.2. System Requirements Analysis

The adopted project methodology was the V-cycle development. As a crucial element of this methodology, the requirements of the system to be conceived were defined. The requirements formed an extensive list, of which the main system requirements were the following: (1) The system must be horizontally scalable on commodity hardware, meaning that if the amount of data to process from one day to another doubles, then the solution can roughly double the instances/machines. (2) The system must be fault tolerant. This implies that if any sub-component or entity of each system fails, then the whole system should continue to work. With respect to data, this means that if one of these subcomponents or entities fails, no data is lost. (3) The system architecture should be based on microservices. (4) The architecture must be generic so that it can easily be adapted to fit future AKKA customer requirements and use cases.

3.3. Deployed System

Figure 6 illustrates the final deployment state of this work. Each box represents a machine that was used (either of Type 1 of Type 2). A docker symbol is included in the top left corner when all the deployed technologies and tools were containerised for that machine.

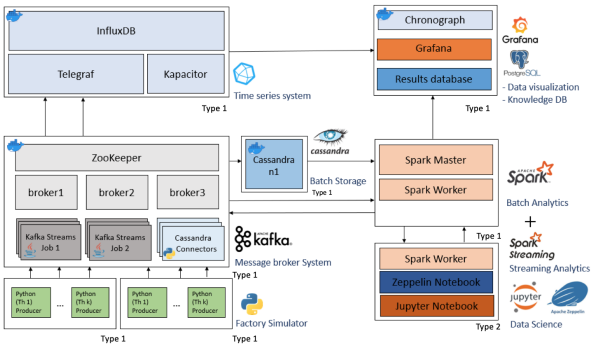


Figure 6: Final ZORRO system architecture.

The central piece of the deployment is Apache Kafka. Kafka was deployed with 3 message brokers. The Kafka deployment was complemented with Kafka streams applications, which performed quick online data processing jobs such as the implementation of virtual sensors. It receives data from the factory simulator, which as shown in Figure 6 was optimised and distributed across multiple machines. For time series specific and data visualisation purposes, the TICK stack was deployed and made to ingest data from Kafka in real-time (top left). For persistent storage, Kafka connectors were coded to send data to Cassandra that was deployed as a persistent storage data lake. A Zeppelin notebook was installed on the serving layer to interact with the raw data in Cassandra. With regards to data processing, a two worker Spark cluster distributed across two machines was deployed. Spark was connected both to Cassandra, in batch processing mode, and to Kafka in stream-processing mode. This is known as a lambda architecture. A Jupyter notebook client was deployed on the serving layer allowing for the interaction with the ZORRO data processing system through the development of data processing jobs such as table-to-table joins, data labelling for machine learning, data exploration, and analytics for insight extraction.

All elements illustrated in the Figure were deployed, used and tested. The developed system was designed to reflect similar Big Data, stream processing and IoT data systems analysed in the state-of-the-art study. All technologies used to build the core of the system are open source industry used Big Data technologies guaranteeing scalability and fault-tolerance. The system is modular as the technologies chosen are easily integrated with other technologies different from the ones used in this work. This marked the satisfaction of the requirements 1, 2, 3 and 4 described in section 3.2. Allied to each architectural or technological feature, documentation was produced for AKKA in order to justify the design choices made. All files used to build the developed solution were stored in the functioning deployment environment and on a GitLab code repository for future use.

4. Results

The results of the developed work in this dissertation can be divided into two main contributions. First of all, the creation of know-how for AKKA Technologies regarding Big Data Industry 4.0 systems. Secondly, the development of a meta architecture and solution with quantitative and qualitative proof of concept outputs and performance indicators. In this section we will describe the results obtained for both these contributions focusing mainly on the results obtained from the developed Meta solution.

4.1. Real-time Data Visualisation

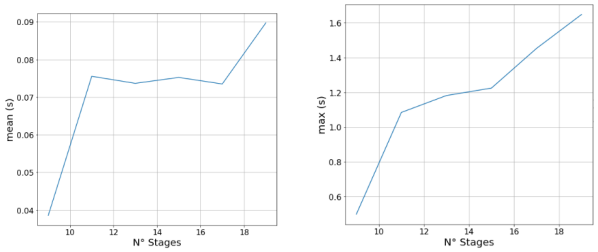
The first end-point system that was deployed was the real-time data visualisation and asset monitoring tool. Figure 7 is an example of one of the dashboards that was produced.

Dashboards like these mark the completion of the real-time data visualisation and asset monitoring objectives which can clearly be applied in use cases such as remote analysis (maintenance) and visual insight extraction. A latency analysis was performed on this service to understand the extent to which this visualisation was real-time. The latency here measured refers to the time taken for the data to be transferred from the simulator to Kafka to the

Figure 7: Example of a ZORRO Chronograf Dashboard produced.

TICK stack.

The latency test carried out consisted in progressively increasing the data load on the system. The load was increased by increasing the number of production stages being simulated. Each stage contained two equipments. For each test, an extra factory stage containing two equipments was added. Each equipment contained 4 data producers (on equipment, equipment surrounding on product and product surrounding), each producing 5 messages per second (mes/s). Each group of 5 messages holds around 6 KB of data. Therefore, this corresponds to around $6 \times 4 \times 2 \times nb_{stages}$ KB/s of data. For 19 stages for example, the amount of data being transferred is 912 KB/s and 760 mes/s. Figure 8 illustrates the mean and maximum latency's obtained.



(a) Mean latency. (b) Maximum Latency.

Figure 8: Kafka latency obtained after optimisations.

Overall, the values obtained for the latency are admissible, having obtained mean latency values always inferior to 0.1 s and consistent with some of the state-of-the-art [13]. Despite the admissible latencies obtained, it was discovered that the dynamics of the latencies were highly effected by the factory simulator. Indeed, large changes in the latency were seen precisely when there was a significant change in the load on a machine. Therefore, in order to truly measure the latency more accurately, a more efficient factory simulator is required.

4.2. Data Science and Exploration Tool

As seen in Figure 6, a Jupyter notebook was deployed and connected to Spark, to allow for Big Data analytics and data exploration. To test this tool and demonstrate its potential, a series of data exploration methods were used. Python statistics and machine learning libraries were used to analyse the data stored in Cassandra. Some of the outputs from the data exploration obtained can be seen in Figure 9.

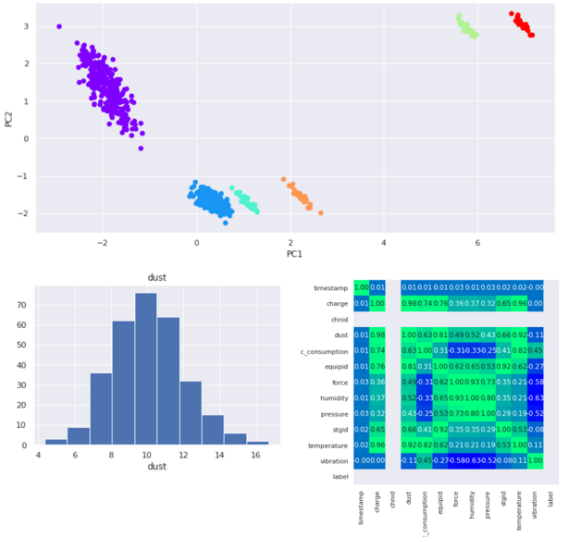


Figure 9: Results from data visualisation - PCA, histogram and correlation matrix.

Illustrated in this figure are the produced output of 2D (2 primary components) Primary Component Analysis (PCA) showing colour coded clusters corresponding to six different equipments, histogram of the dust measurement on one of the equipments and correlation matrix between measured variables from one equipment.

All the results obtained from these diagrams allow us to understand, in a data driven fashion, what nominal and non-nominal factory behaviour physically looks like. We can also use these outputs to understand where optimisations can be carried out. For example, using the correlation matrix, if we find that temperature for a machine is highly correlated with electric consumption, we can attempt to move this machine to a cooler area or reduce the temperature of the section it is in. Essentially, our platform is capable of using and manipulating tools which allow for data driven insights to be extracted and data driven decisions to be made.

4.3. Automatic Anomaly Detection

Once the data sources were prepared, the next step was to train models to detect equipment anomalies. Four supervised learning models referenced in section 2.9 were used. The problem was therefore ap-

proached as a binary classification problem (faulty 1 vs non-faulty 0). In order to train supervised learning models to detect equipment anomalies, batch jobs were done beforehand. The relevant data was joined into a single table and labelled by associating to each row, a column indicating an equipment status of faulty or non-faulty.

In this work, two anomaly criteria were chosen. In the first criterion, a reading was considered an anomaly if, at the same time both values of two different sensors were above or below a certain threshold. The thresholds were defined depending on the mean, the standard deviation and the distribution of the sensor reading. For normally distributed sensor readings, for example, the thresholds were defined at three sigmas (3-sigma) to the left and to the right of the mean (99.7% rule). An is defined by a combination of two anomaly sensor measurements. The idea behind this criterion is that anomalies are related not to individual features but to a combination and relationship between features. The models were trained in a data set where approximately 5% of the points were labelled as anomalies. The data corresponded to 20 minutes worth of simulation.

Following this data labelling criterion, the classical machine learning process was followed. The data was divided into training and testing data sets (75% to 25% respectively). The results obtained for the model scores are available in Figure 10.

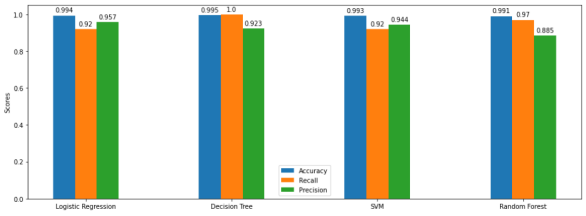


Figure 10: First anomaly criterion - model scores.

Three score metrics were chosen: Precision, Recall and overall accuracy. These are typical metrics used in binary classification based on the confusion matrix. Recall refers to the ability of finding anomalies when they exist (true positives) [14], and is given by the ratio between true positives and the total number of positives. Notice that in the case of this paper, a positive refers to an equipment anomaly. Precision refers to the quality of the model in finding the positive class. It is thus also known as the positive predictive value [14] and is calculated by dividing the true positives by the sum of the true positives and the false positives.

In order to obtain a more precise evaluation of the models, precision vs recall and ROC curves were plotted. The models studied, apart from SVM, can predict a probability that a certain measurement is an anomaly, rather than the class itself. This provides model tuning flexibility, as the threshold of probability with which a measurement is considered an anomaly can be varied. The ROC curve plots the false positive rate against the true positive rate (recall) as a function of this threshold. Figure 11 illustrates the ROC curves plotted for this anomaly scenario.

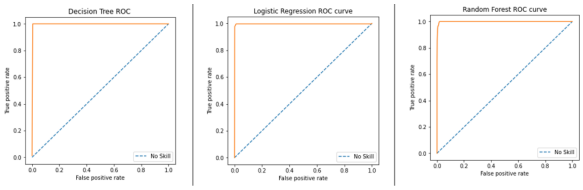


Figure 11: Sensor combination anomaly - model ROC curves.

The area under the ROC curve (AUC) measures how well a binary classification model separates the two classes. The ROC curves confirm the satisfactory model results obtained.

In the second anomaly criterion, we identify an anomaly every time one of its features displays a value that is above or below the defined thresholds. In this case, the proportion of anomalies to non-anomalies changed significantly. Around 33% of points were anomalies. Figure 12 shows the model scores obtained following the same process and using the same model parameters.

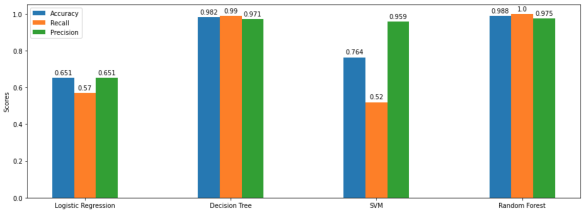


Figure 12: Second anomaly criterion - model scores.

In this second anomaly scenario, tree-based algorithms shine (Random Forest and Decision Tree). From this initial approach, we can already conclude that tree based algorithms seem to behave better in data sets where the ratio between classes approaches 50%. Also, it is clear that supervised learning models are sensitive to the type of anomaly.

All the curves, results and notebooks in which they were obtained, can serve as an important tool for ZORRO PoC purposes but also for future work to be done on the machine learning and anomaly detection aspects of ZORRO. This contribution paves the way for a new ZORRO ML Toolbox.

4.4. Anomaly Pipeline and Visualisation

Once the models were trained, they were deployed in Spark for online real-time anomaly detection. The detection pipeline was built so that when a

model identified an anomaly, it would signal that anomaly by sending a message to the Kafka broker's Anomaly topic. Given that the predictions are written into Kafka, it is then easy to then insert them into the TICK stack via Telegraf and visualise them in real-time. In Figure 13, the Chronograf dashboard on the right peaks every time an anomaly is identified. The graph on the left shows the probability with which the decision of labelling the anomaly was made.

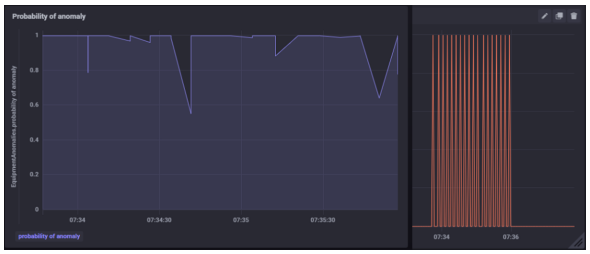


Figure 13: Visualisation of anomaly detection with a Chronograf dashboard.

This was an important milestone in the ZORRO project as it is a real-time visual representation of the machine learning algorithms working to adding insights on the production status. Most importantly, the anomaly information and alerts are display in a human interpretable fashion. The architecture can now also be considered event-driven.

With the anomaly detection streaming pipeline tested and functioning correctly, the next aspect studied was the latency of this pipeline. Given that anomalies need to be identified in real-time so that the responses to these events can also be done in real-time, the levels of latency of this pipeline must be understood and minimised.

The simulator was run, for each model separately, with one equipment only (as each model should be trained for one equipment only). The objective was to be able to study the latency associated with each individual model. For each model, the simulator was run for 20 minutes and the results were calculated on the final 15-minute window (to make sure the system latency could stabilise at a certain value). Given that only one equipment was run, the amount of data the system was exposed to was only around 32 KB/s or 20 mes/s. The simulator was run using the exact same anomaly configuration that the model was trained with. The model scenario chosen for the latency study was the sensor combination model. The results of the mean latencies obtained are illustrated in Figure 14.

Each column represents a latency when compared with the moment the timestamp of the reading was created in the factory simulator. The orange column represents the time it took for the data to arrive to Spark from the factory. This therefore

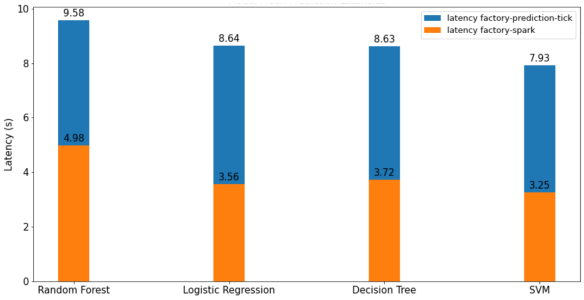


Figure 14: Mean latencies obtained for each model.

involves communication from the factory to Kafka and then from Kafka to Spark. The blue column represents the time it took for an anomaly to be written into the TICK stack (displayed in the dashboard). This is therefore the total time that it takes to identify an anomaly and display it in a human readable fashion.

Having obtained an average latency per model of about 8s, it is clear that stream-to-stream joins along with machine learning predictions take time. We are no longer at the small communication latencies we identified initially with the TICK stack in section 4.1. The latency is also clearly affected by the model chosen. As expected, the Random Forest model is the one that takes the most time given that it must consult the predictions of multiple trees (300 were used). This latency could be reduced by attempting three things: scaling the Spark cluster thus adding processing capacity, Scaling the Kafka cluster or reducing the frequency (load) of the data production in the factory. Even though the obtained latencies are far from ZORRO's objective of 5s, a platform and architecture is setup and ready to be optimised so that the objectives can be achieved.

In conclusion, Spark structured streaming applications capable of detecting anomalies in real-time and displaying them as alerts in dashboards were successfully produced. A lambda architecture capable of using machine learning to detect production anomalies in real-time was built. This was the original objective of project ZORRO.

5. Conclusions

In this work we have developed an Industrial IoT data collection and processing system, respecting Big Data constraints and allowing for both batch and streaming data processing. We used the developed system to visualise manufacturing data in real-time, perform data exploration and analytics, and to detect manufacturing anomalies in real-time using machine learning algorithms trained and deployed on the same platform. Tests were performed on the platform both to quantify its performance but also to demonstrate its potential. Allied to the

developed IIoT Meta solution, an extensive documentation was done in order to create an internal known-how for AKKA regarding Big Data and IIoT systems.

Despite the advances made to the system, there is a significant amount of work to be done for the ZORRO system to be fully validated and ready to be demonstrated as a Proof of Concept to potential future clients and partners of AKKA. One of the most important aspects to work on in this project is the integration of the developed stack with a real IoT layer. Given that the factory simulator was coded using Python scripts, many aspects inherent to a real IoT layer were not at all simulated. In a real IoT layer, network connection may be limited, and sensors may have unexpected behaviour that was not simulated in our anomalies. On the deployment side of things, most of the developed architecture was containerised, however, deploying it with a cluster manager such as Kubernetes would make the containerised deployment more micro-service, fault tolerant and scalable. Regarding automatic anomaly detection, new models, such as the auto-encoder artificial neural network could be explored, to obtain more robust models.

With this work, AKKA is better equipped both from a technological and a know-how perspective in aiding their manufacturing customers in their Industry 4.0 strategies and implementations.

Acknowledgements

The author would like to thank Dr Absadeq Zougari for the tutoring received in AKKA throughout this project and Prof. João Nuno de Oliveira e Silva for his supervision in the completion of this master thesis.

References

[1] V. Koch R. Geissbauer, S. Schrauf and S. Kuge. Industry 4.0 – opportunities and challenges of the industrial internet. PricewaterhouseCoopers, December 2014.

[2] McKinsey. Industry 4.0 after the initial hype. McKinsey Digital, 2016.

[3] Nancy Velásquez, Elsa Estevez, and Patricia Pesado. Cloud computing, big data and the industry 4.0 reference architectures. *Journal of Computer Science and Technology*, 18(03):e29–e29, 2018.

[4] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23, 2015.

[5] Microsoft. Airbus reaches new heights. [Online] https://news.microsoft.com/en-my/2019/06/17/airbus-reaches-new-heights-with-the-help-of-microsoft-mixed-reality-technology/. Accessed: 10 12 2020.

[6] Airbus. Skywise, the leading data platform for the aviation industry. [Online] https://skywise.airbus.com/. Accessed: 13 12 2020.

[7] Waqas Ali Khan, Lukasz Wisniewski, Dorota Lang, and Jürgen Jasperneite. Analysis of the requirements for offering industrie 4.0 applications as a cloud service. In *2017 IEEE 26th international symposium on industrial electronics (ISIE)*, pages 1181–1188. IEEE, 2017.

[8] P Sommer, F Schellroth, M Fischer, and Jan Schlechtendahl. Message-oriented middleware for industrial production systems. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 1217–1223. IEEE, 2018.

[9] Emanuel Trunzer, Pedro Prata, Susana Vieira, and Birgit Vogel-Heuser. Concept and evaluation of a technology-independent data collection architecture for industrial automation. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 2830–2836. IEEE, 2019.

[10] Peter Heller, Dee Piziak, and Jeff Knudse. An enterprise architect's guide to big data: Reference architecture overview. *Unpublished paper. Oracle Enterprise Architecture White Paper*, 2015.

[11] Rui Liu and Jun Yuan. Benchmarking time series databases with iotdb-benchmark for iot scenarios. *arXiv preprint arXiv:1901.08304*, 2019.

[12] DataStax. Benchmarking top nosql databases apache cassandra, couchbase, hbase, and mongodb. eBook, April 2014.

[13] P Sommer, F Schellroth, M Fischer, and Jan Schlechtendahl. Message-oriented middleware for industrial production systems. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 1217–1223. IEEE, 2018.

[14] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.