

CASE STUDY

Open Access



Enabling real time big data solutions for manufacturing at scale

Altan Cakir^{1,2,3,4*}, Özgün Akın^{3,4}, Halil Faruk Deniz^{3,4} and Ali Yılmaz⁴

*Correspondence:
altan.cakir@itu.edu.tr

¹ Istanbul Technical University
Artificial Intelligence,
Data Science Research
and Application Center, Istanbul
Technical University, Istanbul,
Turkey

² Department of Physics
Engineering, Faculty of Science
and Letters, Istanbul Technical
University, Istanbul, Turkey

³ Parton Big Data Analytics
and Consulting, ITU Teknokent,
Sarıyer, Istanbul, Turkey

⁴ Big Data and Business Analytics,
Department of Data Engineering
and Business Analytics, Istanbul
Technical University, Istanbul,
Turkey

Abstract

Today we create and collect more data than we have in the past. All this data comes from different sources, including social media platforms, our phones and computers, healthcare gadgets and wearable technology, scientific instruments, financial institutions, the manufacturing industry, news channels, and more. When these data are analyzed in a real-time nature, it offers businesses the opportunity to take quick action in business-development processes (B2B, B2C), gain a different perspective, and better understand applications, creating new opportunities. While changing their sales and marketing strategies, businesses are now able to manage the data they collect in real-time to transform themselves, to record them in a healthy way, to analyze and evaluate data-based processes, and to determine their digital transformation roadmaps, their interactions with their customers, sectoral diffraction, application, and analysis. They want to accelerate the transformation processes within the technology triangle. Thus, big data, recently called as small and wide data, is at the center of everything and becomes an important application for digital transformation. Digital transformation helps companies embrace change and stay competitive in an increasingly digital world. The value of big data in manufacturing, independent from sectoral variations, comes from its ability to combine both in an organization's efforts to both digitize and automate its end-to-end business operations. In this study, the current digitalization and automation applications of one of the plastic injection-based manufacturing companies at scale will be discussed. Presented open-source-based big data analytics platform, DataCone, that increases data processing efficiency, storage optimization, encourages innovation for real time monitorization and analytics, and support new business models in different industry segments will be demonstrated and discussed. Thus, development and applied ML solutions will be discussed providing important prospects for the future.

Keywords: Big data, Digital transformation, Real-time learning, Machine learning, Manufacturing, Open source, Plastics injection

Introduction

The modern business world requires using the information as an important resource in increasing competition. Organizational success can only be achieved by managing information, transforming it into knowledge, and using the extracted knowledge. In the digital transformation era, generating data-based strategies, data-driven business,

will provide optimizing the performance of the companies by collecting and analyzing data through the product life cycle [1].

Processing data is not a new sight for companies. Companies have systems for storing and processing data in order to accelerate business processes. On the other hand, most of them have traditional data warehouse technologies, which are not suitable for extremely high-speed data from various sources [2], for the current data flow in daily operations. IoT enables the enlargement of collectible data, big data technologies provide an opportunity for data to be processed and transformed into knowledge at scale. Big Data is generally described as technology that enables to obtain information from high-speed data from different sources by applying statistics, mathematics, econometric, simulations, optimizations, and/or other techniques to support decision-making processes [3] in real-time. Therefore, big data technologies and deployment of a data lake are considered in companies now in addition to data warehouse solutions. Managing all types of data, studying specific analytical studies by multiple users, providing the required data lineage back to source systems, and enabling users to access the analytical contents in a self-serve manner are the necessities of deployment of the data lake operations [4].

The manufacturing industry is one of the largest areas to create value from data. Smart manufacturing is the goal of companies in manufacturing. Smart manufacturing aims to convert data acquired across the product and process life cycle into manufacturing intelligence to get positive effects on all areas of manufacturing [5]. Machines, sensors, and other equipment are the various data sources in manufacturing that make essential big data. Data analysis can give meaning to patterns, trends, areas of inefficiency, and potential risk to manufacturers to provide improvement of process efficiencies [6]. Creating a real-time monitoring system provides accurate and timely decision-making in operational processes. All of the reasons and possible effects say it is expected that enabling big data solutions will contribute significantly to the advancement of smart manufacturing [7].

There are lots of areas that can gain points in manufacturing, on the other hand, big data usage is lower than other industries such as social media, e-commerce, and finance with respect to manufacturing due to software, middleware, and hardware challenges. There are five fundamental main challenges in manufacturing data sourcing and collection, data integrity, data processing, data storage and advanced data analytics. In the data acquisition stage, different types, heterogeneous structures, and various dimensions of data are the challenges that can be faced. In the data preprocessing stage, integration of data, data cleaning and matching, data redundancy and data compression are challenging tasks. In data storage, reliability of the database, scalability, and efficiency are the issues. Data security, data privacy, the efficiency of data mining, temporal and spatial correlation are the issues that can be encountered [8].

In this article, the big data management and operational analyses from the plastic injection processes of real applications are discussed. Infrastructure, components, and architecture of big data platform are mentioned, problems encountered and their solutions in implementation are explained. Time series integrated sequential machine learning and deep learning deployment are briefly presented.

Case description

As companies move deeper into digital technologies, production forecasting methodologies are more widely adopted, there are more opportunities than ever to transform routine production runs with data-driven decision that makes a difference. This is largely because of the maturation of big data—a prominent term for a solution of storage, organization, and analysis techniques developed for massive data sets in the context of various manufacturing processes. Our case study is a large scale common manufacturing company that produces automobile plastic components using injection molding techniques. Although injection molding based large scale production is considered a well-defined manufacturing technology, it requires to stay competitive and adapt to the dynamic market demands. Therefore, many companies are willing to move towards smart manufacturing processes, or Industry 4.0 applications [9]. Plastic injection molding is a significantly important process and the quality of the produced parts is mostly affected by the process conditions such as injection temperatures, injection pressures, injection speeds, ambient temperatures. Optimum injection parameters, environmental conditions, and other reasons that have effects on product quality and machines operations are not specified exactly. Therefore, to go beyond classical engineering studies, establishing a real time big data management system is the first step to integrating data driven applications for operational gains. The Big Data operation includes the generation, manipulation, ingestion, acquisition, storage, processing, and analysis of data. During injection molding, a large scale of data is generated and can be successfully collected by sensor sets installed in different units of the injection molding machine and other platforms. Although data is one of the most prominent assets of an intelligent solutions, many applications have difficulties selecting essential and useful analytical solutions from the manufacturing processes effectively. Therefore, many studies have been made to adopt machine learning (ML) techniques — especially for injection molding operations — for industrial application. This is essential for artificial intelligence studies in order to analyze the data obtained from production and environmental conditions in real-time nature [9].

Architecture design

In this section, DataCone's infrastructure and architecture are presented. Big data components that are used in DataCone, are explained briefly. The details of the system are discussed in the optional information section.

Infrastructure

DataCone's big data cluster is built on 11 virtual machines which have different technical features. The machines in the cluster totally have 128 CPUs, 512 GB ram, and 40 TB SSD disk space. The distribution of the hardware fits the requirements of the big data system tools to be installed and the amount of data to be processed at a certain time. In this context, a namenode and a secondary namenode are used primarily to ensure high availability in the Hadoop file system [10]. The task of the secondary namenode is to keep the replica of the system records and to ensure that the system continues without interruption in case a problem occurs in the namenode. After HDFS and mapreduce configurations are made, YARN [11] and Spark [12] are

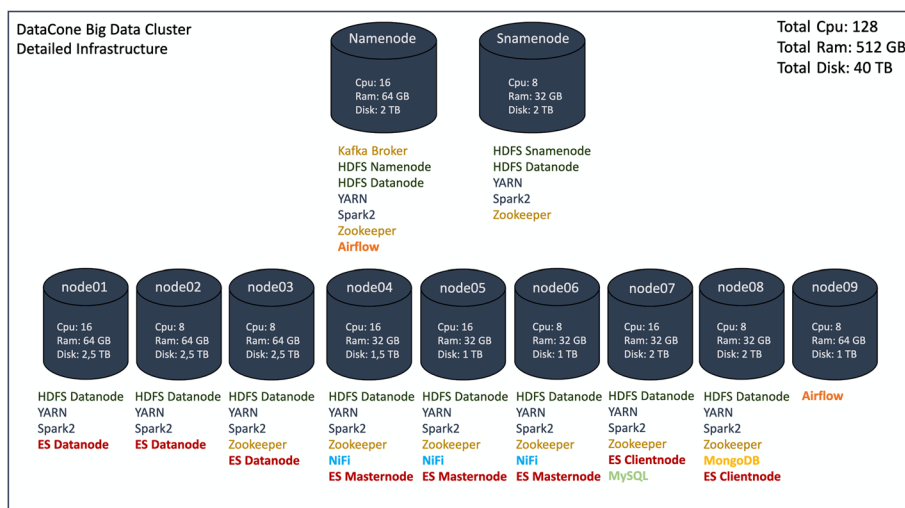


Fig. 1 Cluster design of DataCone with master node to manage the cluster, secondary master node to ensure high availability, and 9 slave nodes. The infrastructure of DataCone is designed for manufacturing companies considering data frequency, data volume, and sector-specific IoT protocols. Open-source big data software is distributed over the nodes to optimize hardware cost. The main logic of the infrastructure depends on scalable structure. When data volume is increased or new devices are needed to be added, infrastructure will be easily scaled by adding new nodes

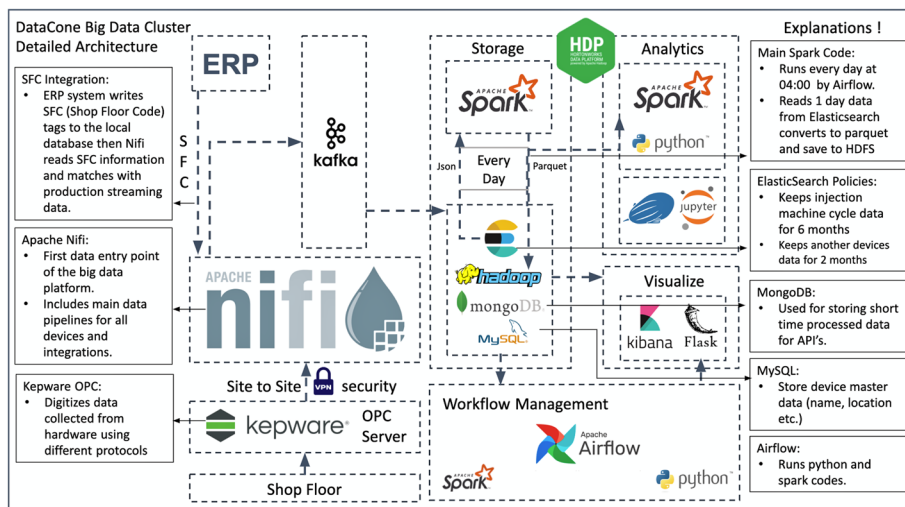


Fig. 2 Detailed architecture design of DataCone with a combination of proper open-source big data software and enterprise software which are already used by the company. DataCone's architecture ensures seamless interaction between existing manufacturing management systems and novel big data technologies. The data flows from left to right in this architecture. The data that is produced by IoT sensors and PLCs is consumed by Kepware OPC Server and published to the cluster via proper protocols. Apache Nifi consumes this data and enriches it by combining it with the data that is stored in ERP systems. After the enrichment process, the data is pushed to the storage and analytics layers for further analysis

installed on all nodes, which has active HDFS usage together with the in-memory computing model. In this way, it is aimed to use the power of memory-based processing, in contrast to disk-based processing in Hadoop technologies, in the entire cluster via Apache Spark. Elasticsearch [13] is added to the big data system in order to store data for a short time and run fast queries on the data for real-time query and reporting. All in all, considering data size and data flow frequency, Elasticsearch is installed in the cluster as 3 datanode, 3 masternode and 2 client nodes. Furthermore, Apache Nifi [14] is built on 3 nodes in cluster mode to set up streaming pipelines and define data streams. Corresponding heterogeneous components to be used in DataCone are positioned in the cluster as shown in the Figs. 1, 2.

Software components

Apache Nifi

Nifi was formerly developed by the National Security Agency (NSA) and is based on open source Niagara Files software as part of the technology transfer program in 2014. It is an open software for automating and monitoring network data movement between heterogeneous systems. It can analyze different data formats and protocols and operates on Java Virtual Machine [14]. NiFi implements the concept of FBP (Flow-Based Programming) as a real-time ETL for collecting large amounts of data in a distributed environment. Generally, ETL means to extract data from the same model or other models to a data warehouse or data lake in our case, transform the data to save the data in appropriate format or structure for inquiry or analysis [15].

In ETL or data in motion, the major concern that how to secure, track and manipulate the data flow. Therefore, Apache NiFi, which is our data automation platform, is capable of handling millions/billions of records to ingest and enrich data in near real-time in a distributed environment. In addition, Apache NiFi has been utilized as a user-friendly web UI with IoT ready-use processors. Processors are doing some combination of data routing, transformation, or mediation between systems by accessing the incoming flow files and their contents. The Processor can also operate independently without the flow files [16]. Apache NiFi supports dataflow management for both structured and unstructured data, powered by the separation of metadata and payload. Schema is not required but can be used as optional [17].

In DataCone, Nifi is used as a data entry point of the big data platform. All the production data is pulled using Apache Nifi from local servers and sensors. There are structured data and unstructured data which are coming from SQL Server and MQTT server. Also because of the constant change in the production area, the structure of data has a constantly changing schema. Nifi was chosen to easily manage both structured and unstructured data in a constantly changing production environments.

Apache Kafka

Kreps et al. (2011) introduced Apache Kafka, a distributed messaging system that is specially designed for a high volume of log events processing. It also provides integrated distributed support and can scale-out. Kafka achieves much higher throughput than conventional messaging systems (such as ActiveMQ and RabbitMQ) [18].

Apache Kafka can be deployed on bare-metal hardware, virtual machines, and containers, and on-premises as well as in the cloud systems [19]. It consists of several capabilities, therefore it can be implemented for end-to-end event streaming. These capabilities are to publish (write) and subscribe to (read) streams of events including continuous import/export of data from other systems, to store streams of events durably and reliably, to process streams of events as they occur, or retrospectively.

Apache Kafka is therefore placed right after the Apache Nifi in the DataCone pipeline. Near real-time production, data is subscribed to the Apache Kafka topics to be stored temporarily and many systems commune data from these Kafka topics with low latency and high security.

Elasticsearch

Elasticsearch is an open-source, real-time, and distributed search and analytics engine based on the Lucene library, is released in February 2010 [20, 21]. It is used to store, search and analyze big data with analytically created indices. Elasticsearch stores data as JSON(Javascript Object Notation) documents, and retrieves whatever document with fast query. It is generally used by big companies such as Netflix, LinkedIn for web search, log analysis, and big data analytics [22].

In order to understand the working principle of Elasticsearch, the terms index, shard, node, replica, cluster should be briefly discussed. The index is similar to a database in a relational fashion. Elasticsearch stores data in one or more indices and reads the data from these indices. Each index must have a unique name and must be created in lower-case letters. Every index is made up of one or more shards [13, 23].

In DataCone, production data is consumed to be indexed in Elasticsearch from Kafka topics using NiFi processors. Thanks to Elasticsearch, the production data become easily searchable. Data is stored in Elasticsearch for the indicated intervals in a hot phase. To be able to visualize the data near real-time, Elasticsearch is integrated with the Kibana dashboard.

Ambari

Big data platforms in general consist of many open-source software working synchronously on a distributed architecture. Everyone knows that it is not easy to install and maintain these open source components individually [24]. In other words, it is difficult to keep these systems alive in the production environment and to follow compatibility problems in the application. The former Hortonworks, the new Cloudera, data platform was developed in 2011 by the Yahoo team responsible for the Hadoop Project developments. [25] This platform provides many facilities for installing and managing big data systems over a single interface which is called Ambari. Thanks to Ambari, big data system components can be installed effectively on the cluster and it is possible to monitor the problems in the architecture with certain types of tracking algorithms.

In DataCone, modified version Ambari is used to install and manage HDFS, MapReduce, YARN, Spark, Zeppelin, Kafka, and Hive software's in the context of Lambda architecture [26]. The system is based on eleven nodes and a secondary master node is added to ensure sustainability. A Hadoop file system, which can be accessed through the Ambari interface, has been installed, allowing easy access to the folders stored in

HDFS. Thanks to the Ambari interface, upload and download operations can be performed on the file system established. Necessary configurations of the specified software were made through the interface. After the installation phase, the Ambari was also used to monitor the system status, and information about the number of nodes, storage space, network, and CPU usage, and memory usage were visualized via the system monitoring dashboard.

Apache Spark

Spark was developed as a unified engine that can work at various workloads for the analysis of big data. Apache spark was adapted to industry and academia in a short time and became one of the most popular projects of Apache Software Foundation [27]. Compared to previous technologies, spark is much faster and has a much more compact structure. For example, in the study on logistic regression, it was concluded that spark operates 100 times faster than Hadoop map-reduce [28]. Since spark provides APIs in many programming languages such as Python, Java, Scala, R, it makes complex distributed big data analysis operations easier.

Spark provides many impressive high-level tools that can be used in cluster computing such as spark SQL for running SQL queries on spark, MLlib to build machine learning models, Spark Streaming to handle streaming data, GraphX for graph processing [28]. Having these libraries, which have separate very important functions, made spark a general big data processing engine. These tools are developed and reproduced in each version of spark to keep it up-to-date and functional.

Spark Streaming is used to process and compress near real-time production data in DataCone's architecture. The data is consumed from Kafka topics using Spark Streaming code which is written in python. This code is processed all production data and saves it to the Hadoop distributed file system as a parquet file in a 10 minutes time interval. The data size is decreased ninety-five percent thanks to using the Parquet data format.

Apache Hadoop

Manufacturing data consists of structured, semi-structured, and unstructured data. The high volume of data is stored in HDFS for batch analysis. Hadoop Distributed File System (HDFS) is one of the key elements in the Hadoop Framework and it is an open-source implementation of the Google File System (GFS) for storing a vast amount of data. HDFS is a large distributed file system designed to be deployed on low-cost hardware [29] and provides high throughput with fault-tolerance. HDFS is widely used in research for its open-source and advanced architecture. It primarily settles the massive capacity issue and data consistency.

HDFS stores the records as a series of blocks and as duplicated for fault tolerance. Data partitioning, processing, columnar formats, replication, and placement of data blocks will increment the performance of Hadoop. The performant columnar format in the Hadoop Ecosystem is Parquet which is an open-source file format. Apache Parquet is designed for efficiency as well as the performant flat columnar storage format of data compared to row-based files like CSV or TSV files [30].

Hadoop is used as a long-term distributed data storage in DataCone. After the data is processed with Spark, it is saved to Hadoop as a parquet file. Data are classified

according to date. The file system is designed in a layered structure. In the first layer, data is kept on a daily basis, and in the next layers, files are grouped by month and year.

MySQL

SQL (Structured Query Language) is a relational database management standard for accessing and manipulating databases. SQL is used for executing queries, retrieving data, inserting, updating, and deleting records, etc. There are lots of RDBMS (Relational Database Management System) for using SQL such as MS SQL Server, Oracle, IBM DB2, MySQL, and Microsoft Access.

MySQL is an open-source RDBMS. Traditional databases are not effective when using and processing large volumes of data. On the other hand, when master data tables, relational entities, basic attributes are considered, SQL cannot be ignored with its easy-to-design structure and high efficiency. Therefore, MySQL is an open-source and powerful system that was selected to store master data and determine relationships between them.

In DataCone's architecture, MySQL is the place where the master data of the devices are stored. There are tables that store the descriptive information and the DataCone standardized naming of devices.

MongoDB

MongoDB is the most popular NoSQL database that is open-source document-oriented [31]. MongoDB was developed in C++. Because MongoDB is a document-oriented database, the system can have several instances of the database. Storing data is flexible with MongoDB because it uses a syntax similar to JSON called BSON(Binary JSON). This eases the change of data structures and documents. The document model determines keys and values in JSON-like application code, this makes more intelligible and readable data. Indexing is important for databases, it matches queries efficiently without scanning every document of a collection. MongoDB uses indexing to process enormous data rapidly. MongoDB is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability [32].

As a part of DataCone's architecture, APIs are used to feed data to mobile applications and web dashboards. These APIs read the data from MongoDB. JSON documents are written to Mongo via custom python codes and these documents are updated at specified intervals.

Apache Airflow

Apache Airflow is an Apache Software Foundation project, originally developed by Airbnb and released in 2016. Apache Airflow is a python-based fast-running, easy-to-integrate and customizable platform to programmatically author, schedule, and monitor workflows as Directed Acyclic Graphs (DAGs) of tasks [33]. In this way, both graphically observing and creating workflows create a facilitating effect for processes. At the same time, the created tasks do not need to share resources with each other, so it is possible to run independently from each other. It supports the establishment of workable, repeatable structures of workflow studies. The general architecture of Airflow is shown

below and Airflow comprises three main components which are webserver, scheduler, and worker.

Apache Airflow is used to manage code flows in DataCone's architecture. Spark and python codes are scheduled via Airflow. When the code gets an error, Airflow is sent an e-mail to the code responsible. Therefore the problems can be solved in a short time period.

Kibana

Kibana is an open-source interactive visualization interface and analysis tool for data in Elasticsearch. It is an Elastic plugin. Kibana offers the opportunity to visualize data with different types of graphs such as maps, charts, bars, tables, pies etc. [22]. Kibana allows real-time monitoring with the power of Elasticsearch. Users can create dashboards by editing the generated graphics [34]. Kibana presents all created visualizations, analyzes, and time series to the user with dashboards. It shows the whole picture to the user along with the filters [34, 35].

Kibana is used for real-time visualizations in DataCone. Thanks to the seamless connection between Elasticsearch and Kibana, the data which flows to the Elasticsearch are easily visualized on Kibana.

Flask

Flask is a micro-framework for Python with a small core and easy-to-extend philosophy. Depends on the Jinja template engine and the Werkzeug WSGI toolkit. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. Flask supports extensions. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies, and more [36]. Flask aims to keep the core framework simple but highly expansible means it is compatible with all Python libraries. It includes all the necessary features on its micro-framework like development server, debugger, secure cookie management, RESTful API compliance that can save time and effort for programmers.

Flask is used for developing custom dashboards and API's in DataCone. After data is processed with python, it is visualized in custom dashboards which are developed using HTML and javascript. Thanks to Flask, these dashboards can be easily managed by Python.

Operational information

In this section, operational details are discussed for each framework that is used in DataCone.

Data digitalization

Euromap protocol

EUROMAP 63 is a file-based communication protocol for data exchange between Injection Molding Machines(IMMs) and the central computers. It was created by EUROMAP (Europe's Association for Plastics and Rubber Machinery Manufacturers) founded in

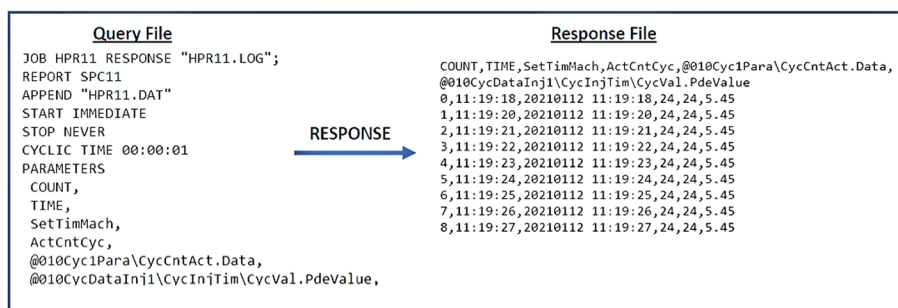


Fig. 3 A sample of file transfer via Euromap 63, that is plastic injection molding machines specific protocol written by Euromap Organization to establish a connection between the machines and the digital systems


1964 [37]. EUROMAP provides technical recommendations and protocols for plastics and rubber machinery manufacturers [38]. The protocols it creates are control, communication, and data digitization protocols suitable for machines of many brands such as Engel, Krauss Maffei. Euromap 63 is one of the data digitization protocols. With the Euromap 63 plug-in, part process data, machine status data, machine health data, etc. makes it collectible by digitizing. The data that want to be collected from the plastic injection machine is written in Euromap 63 standards into the text file created in the local folder where the machine communicates in the central computer. Euromap 63 plug-in reads this file and retrieves the data from the machine and delivers the desired data in the same folder as a text file. In this way, the data generated in the machine is collected. The text files that the protocol reads and writes are shown in the Fig. 3. This is the working logic of Euromap 63. It is possible to use the Euromap 77 version instead of 63 in some of the new injection molding machines. Euromap 77 is able to communicate over OPC UA protocol. Real-time data can be moved to the big data systems directly over OPC UA protocol without needing any OPC servers.

Kepware

Kepware OPC server software is installed in the local environment is used to query and collect data from different PLCs, injection machines, and sensors. Devices that are connected to the company network are registered in the OPC server via their IP addresses. Devices are registered to the system using Kepware’s manufacturing plug-in. During the registration, attention is paid to selecting data connection protocols suitable for the device. For example, while the Euromap63 protocol is selected for injection machines, the Modbus TCP-IP protocol is selected to activate the connection of temperature and humidity sensors. After the device definitions are made and the connection is established, the addresses of each data in the device are defined on Kepware. Pre-determined standardization rules were applied in defining data names. In this way, both name complexities are prevented and identification is provided for devices to be defined in the future. Two different methods are followed in publishing the data defined on Kepware to other systems. Digitalization layers of the data are shown in the Fig. 4.

The first of these methods is the database connection. Kepware’s datalogger plug is used for database connection. Thanks to this plugin, it is ensured that the data flowing

Publish	INTERFACES	ODBC	MQTT, REST
	PLUG-IN	Kepware - KepserverEX	Kepware - KepserverEX
Ingest	SERVER	Kepware - KepserverEX	
Connect	DRIVERS	PLC, Euromap63, OPC Client, RTU, Database, Custom	
	DEVICES	Analyzer, Sensor, Injection Molding Machine, PLC, Robot,	



Data Flow Direction

Fig. 4 Digitalization layers of the data and communication protocols of the devices are shown on the diagram. There are 5 types of devices that produce data in different formats and frequencies on the shop floor. This data is digitalized using an OPC server and published via different methods such as MQTT, REST, ODBC to the big data platform











	Number of Devices	Protocol	Frequency	Total Data Volume for Devices per year	Compression Rate	Storage Volume per year
	13		4 second 1 cycle	606 GB	55:1	11,01 GB
	22		2 second 1 cycle	1400 GB	55:1	25,45 GB
	35		1 second	117 GB	74:1	1,58 GB
	80		1 second	266 GB	78:1	3,4 GB
	10		1 second	12 GB	60:1	0,2 GB
TOTAL	132			2400 GB	57:1	41,64 GB

Fig. 5 Data transmission frequencies, streamed data volume, data transfer protocols, and device types are described in the table. There are two types of injection molding machines as KraussMaffei and Engel. Data frequencies are different because of machine specifications. Data compression rates differ according to data types and average of the compression rate is 59:1. Thanks to the compression process, 19GB storage resource is used instead of 1168GB

through Kepware can be written to the desired relational database. Most of the data collected by the injection machines are recorded in registers once in each cycle (the completion of the production of each part is called the cycle and the average cycle time of the factory is 30 seconds). A database connection is provided over Kepware for the data recorded in the cycle, and it is ensured that these data are written on the database at the end of each cycle. The reason for choosing this method is that the data frequency is small and short-term buffer areas are desired to be created on the database on local servers. These buffer fields are important to prevent data loss when the connection with remote servers is lost.

The second method used to broadcast data over Kepware is streaming over the MQTT protocol. For data flowing at high frequency (current, voltage, pressure), the IoT Gateway plug-in in the Kepware OPC server was used. The data defined on this plug-in can

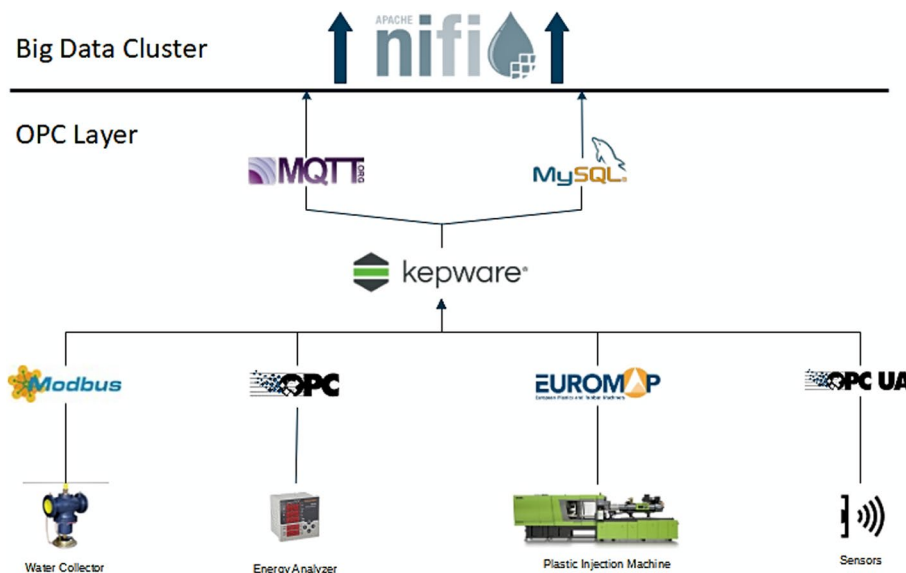


Fig. 6 Data digitalization schema of DataCone. The data is collected from 4 different types of devices, and it is published to the big data platform over two different methods. MQTT protocol is used to transfer streaming data such as voltage, current, pressure, flow, etc. Batch data is inserted into a MYSQL database after each machine cycle is finished. The shop floor data is transferred to the big data platform using those two methods via Apache Nifi

be broadcast instantly over the MQTT protocol. In this way, the data that will be used for the analysis to be made on the flowing data can be accessed in near real-time. Data transmission frequencies are shown in the Fig. 5.

Data pipeline

PLC and OPC intergation

This layer is the beginning of the journey of data. The data generated in the production layer must be read from machines and sensors. There are different types of protocols for collecting data from each data source. The Kepware OPC server, which can collect data from various protocols, is used for this. In this layer, data is collected from machine-specific protocols, PLCs, and sensors. In plastic injection machines, data is collected from the Euromap63 module. The data collected from Euromap63 is collected on a cycle basis and per second. Data from sensors such as environment and vibration are collected every second with OPC UA. Energy analyzers, water collectors, and other equipment data are collected from the PLC of the devices every second. In the production layer, all the machines, devices, and sensors in the cell are connected to a common main panel, therefore all data sources are connected to the network.

Standardized names are used to describe the data correctly. Data names are determined for each device, all definitions are made on the OPC server according to these names. After each data/tag is defined to the OPC server, it is published to the digital layer according to its frequency and the characteristics of the data source. Due to the features of the Euromap63 module, some data come once a cycle and some data come

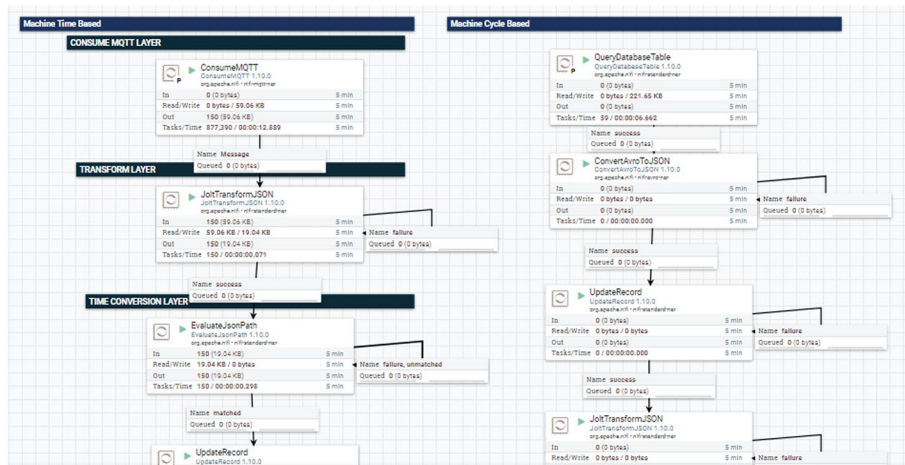


Fig. 7 Two types of data pipelines that are developed in Apache NiFi are used to collect data from injection molding machines with different frequencies are shown in the figure. While the pipeline on the left side is used to gather MQTT data every second, the other one collects the data produced every cycle from the MySQL database

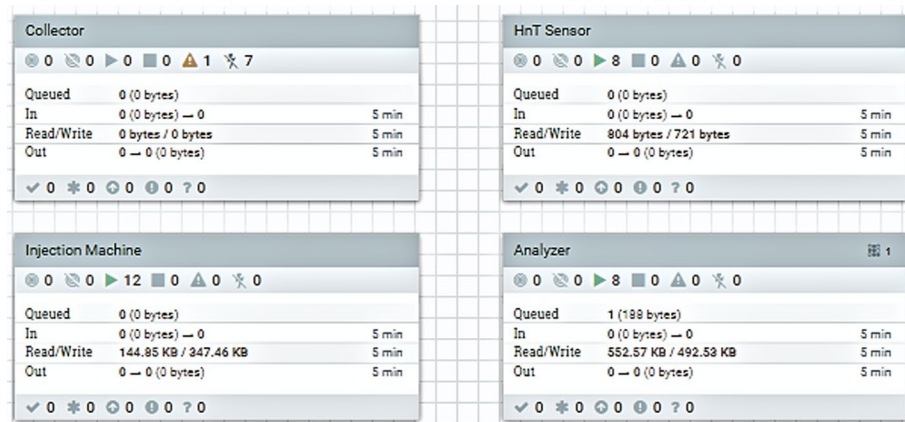


Fig. 8 Pipeline categorization of the devices in a production cell. All equipment in the production cells is grouped by device categories in Apache NiFi to ensure ease of use. The data lineage can be captured from group objects which are seen in the figure

every second. Cycle-based incoming data is written to MySQL, and incoming data every second is published with the MQTT protocol. All data from other devices are tagged with the cycle number data received from the plastic injection machine and published with the MQTT protocol every second. You can see this layer in Fig. 6

Data ingestion layer

For the workflow design, Apache NiFi is used as the medium for routing the data as shown in Fig. 7, by dividing the operation into 2 parts which are time-based and cycle-based workflows. Using NiFi processors, we were able to fetch IOT data from

the manufacturing side with MQTT protocol and ERP data from the ERP server with DBCPService into our data platform. In DataCone, NiFi workflows were partitioned by machine ID and all data was sent to Kafka every second in the first architecture. However, in many cases, these workflows did not fit ML Layer and cycle-based data was duplicated. Since a few data is changed for each cycle which is between 30 and 100 seconds regard to mold and machine and other data is changed every second. Hence workflow which is separated into device-based architecture consists of energy collector, environment sensors, analyzer, Injection machine, and ERP data that is shown in Fig. 8.

Ni-Fi has several processors that give real-time control that performs ETL basic to make available connectors for the file system in the Hadoop Cluster, Elasticsearch, and distributed messaging technologies such as Kafka. In DataCone, the proposed NiFi Custom Processor consists of three processes: The first Extract Custom Processor is responsible for delivering raw data (JSON format) information by using MQTT Consumer which receives messages from MQTT broker and processed data from ERP (CSV, JSON format) information with HTTP requests. The second Transform Custom Processor alters dynamic raw data structure to desired data structure using Jolt Transformation library that written in Java that allows a developer to convert one JSON structure to another [39]. The last third Load Processor is loaded streaming structured data using Kafka as a messaging system that publishes messages to Elasticsearch and Spark Streaming.

Streaming data processing layer

In DataCone, Apache Spark Structured Streaming is utilized to transform JSON file format to Parquet file format and save these files to HDFS which is the data lake of DataCone. By way of parquet, we compress data by 15 times roughly and through the columnar format of Parquet, we read fast OLAP query for batch processing. To read data lying HDFS quicker, we utilize partitioning and repartition on Apache Spark, and data is partitioned by day of a timestamp during saving Parquet data to HDFS. Spark Structured Streaming read streams from Apache Kafka topics and read JSON data with predefined JSON schema. Thus we will approach streaming more structured way and we can manipulate data uncomplicatedly. For fault tolerance, Apache Spark uses checkpoint to read data from source and by way of Apache Kafka on the off chance that any error happens, Spark is able to proceed from the last correct Kafka offset.

Workflow management layer

In DataCone, Airflow has been used for several applications to schedule cron jobs in place of crontab, execute the tasks, and monitor workflows on the Airflow web server. The user interface makes it simple to visualize DataCone pipelines running in production, displays the states of currently active and past tasks, shows diagnostic information, error logs about task execution, and troubleshoot issues when needed.

In DataCone, object-oriented programming in Python is used in Spark, ETL jobs, Machine Learning Products, and scripting. In this manner, the Python API structure in Airflow matched our pipeline exceptionally well. Moreover, workflow implementation in

Airflow is extremely flexible, and the Python API structure utilized to write workflows is simple and learn and understand [40] Machine learning deployment workflow with Airflow is shown. This workflow approach is utilized for our all machine learning use-cases in production.

Data messaging layer

The data flow is designed according to the production technique of the manufacturing company. Using Kafka is aimed to collect valuable process data without loss. In the design created, the cell containing each plastic injection machine and all the devices connected to the machine is considered as a factory. Therefore, separate topics were created for each cell in Kafka. These topics are named according to the cell names and the device the data is received, eg e117-anl. In DataCone, Kafka is managed through Nifi. A topic is created with Nifi in Kafka. To write to the created topic, a PublishKafka Processor is created and a Producer writes data on the topic. Then, Consumer is created with ConsumeKafka Processor to read the data from the topic. In this way, data flow is provided in real-time and without losing cell data. By creating a topic, consumer, and producer over Nifi, Kafka's management is managed from an interactive screen.

Data storage

Long term data storage

In DataCone, HDFS is our parallel manner data lake which stored any kind of warm and cold data in long term. Manufacturing data which can be divided into streaming data and batch analysis results, ERP data, specific dictionaries, and other unstructured data are stored in HDFS. Most of the data will be transformed to Parquet format because we use mostly OLAP queries on HDFS for analytic results and read data lying on HDFS faster. Apache Parquet is the performant columnar format and open-source format. It is more efficient compared to row-based files like CSV files or semi-structured JSON which is the main IOT data transferring format in our data platform. For the streaming process, by the way of converting JSON file format to parquet format, we obtain space in the disk by 15 times around. We also utilized data partitioning to increment the performance of HDFS. Through Spark Streaming, before writing Parquet data to HDFS, we use partitioning and repartition on Spark, and data is partitioned by day of the date. By the way of partitioning, we can read faster with filtered data by timestamp.

Short term data storage

Elasticsearch is a database where we keep hot and warm data and access data with fast queries. Data are taken to Elasticsearch, together with indexes named for each device according to device category, device number, and pipeline number, and month-year. These indexes are designed as 1 shard without replicas. The frequencies of the data in the indexes are different. It has been designed so that production teams can quickly access data. By easily accessing the data of each device, anyone authorized can see the data. For example, quality engineers can easily access the process data of a part produced 2 months ago with a part tag. It also has a design where data teams can easily extract data

Table 1 Naming conventions and device data sample table captured from DataCone's master database

buss_unit	device_category	device_number	pipeline_number	es_index
m14	anl	10001	11	anl1000111
m14	imm	10001	11	imm1000111
m14	tnh	10001	11	tnh1000111
m59	imm	10002	11	imm1000211

Unique naming conventions are created for the manufacturing sector. Device category names are described with three letters for example, anl stands for analyzer and imm stands for injection molding machine. The Device number is started from 10001 for each device category and increases sequentially. The pipeline number describes specific pipeline methods for example 11 is used for real-time data pipelines. es_index stands for index name in Elasticsearch of device and pipeline combination. It consists of device category, device number, and pipeline number

for fast analysis. Elasticsearch in the DataCone is not designed to store historical data, so indexes must be deleted at certain times in order for Elasticsearch to always be fast. The retention time of the indexes is designed according to the data frequencies. Delete policies that work every 6 months were created for cycle-based data, and a 2-monthly delete policy was created for data received every second.

Document database

MongoDB database is installed on the servers in the cluster and is used to store the JSON objects generated by the system. APIs of mobile applications and web-based dashboards read the data they need from the MongoDB database. The codes that periodically calculate the data to be used in the established structure to prevent delays in the interface and applications are managed by airflow. The outputs of these codes are saved in the MongoDB database and are kept ready for APIs to read. For example, the data required to be displayed on the production monitoring screens, which are custom dashboards developed on flask for production, are calculated every 5 minutes by the python code triggered by airflow and saved in the MongoDB database. When users want to access information, it is queried from the MongoDB database by the API connected to the flask interface and presented to the user quickly. In this way, delays are prevented and a constant load is guaranteed in the system for the specified time period.

Master data storage

Descriptive master data such as the names, categories, and locations of the devices in the big data system are kept in the MySQL database installed in the cluster. There are three main tables in the database where device information is kept. The first of these tables is the device category table. This table contains the categories of the devices used in production and the descriptions of these categories. The second and most important table is the device register table. This table contains business unit, device number, pipeline number device category, and Elasticsearch index information for each device. Elasticsearch indexes are created from a combination of device category, device number, and pipeline number. Device numbers are given in each category in increments of the order of newly added devices. The pipeline number indicates the

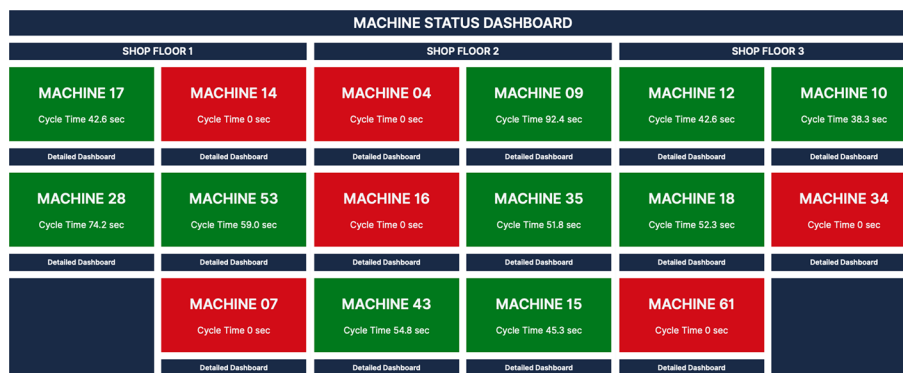


Fig. 9 Machine status dashboard shows whether machines are working or not. If it is working, the color of the box is green. If it is not working, then the color is red. The average cycle time for each machine is shown on the dashboard. Users can reach the detailed dashboards over the links on this dashboard

data transfer pipeline category as shown in Table 1. The last table in the database is the table that specifies the locations of the business units. In this table, the country, category, and facility information of each business unit number is kept. Newly added business units can be followed from this table, and at the same time, this table is used for country and facility-based reporting.

Visualization

Real time dashboards

Kibana is used for real-time monitoring of manufacturing fields and machine statuses. Since Kibana is an Elasticsearch product, the same indexes are used to make graphics. Therefore, it is quite easy to draw graphics with drafts in Kibana. Production monitoring screens that show each field's machines' working status and cycle times are designed for manufacturing engineers and responsible. The screen named production monitoring screen can be seen in Fig. 9.

Kibana contains draft graphs and charts like line graphs, bar charts, radar charts, heat maps, region maps, pie charts, gauge,s, etc. To monitor real-time data in every injection molding manufacturing cell, machine-specific screens are designed. The screen shows actual machine parameters like cylinder temperatures, pressures, volumes, etc. This screen shows not only data generated on the machine but also environmental data, mold data, and ERP production management data. Machine status, number of parts produced, last cycle time, real-time machine parameters, mold water temperature, ambient temperature, and pressure are real-time data that can be viewed in this screen shown in Fig. 10.

Reporting tool

When visualizing data with JSON data type we use the ajax method to get data in javascript. After creating a template with HTML and CSS, inside javascript code, we use plotly.js which is an interactive web graphics via javascript open-source graphing library. To make charts interactive with constantly updated data from put-up dash API URL we turn ajax into a function to collaborate plotly data section. After changing other values



Fig. 10 Detailed machine dashboard shows near real-time process data such as pressure, time, volume, speed, and temperature. Plots on the dashboard are mostly grouped by part number. Users can compare different parts in terms of the process parameters by selecting part numbers using the drop-down menu on the left corner. This dashboard is refreshed automatically every 5 s

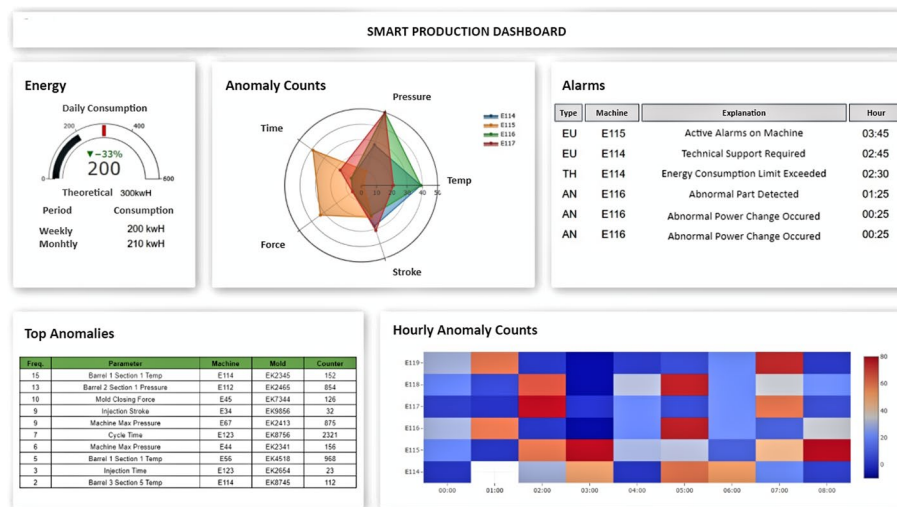


Fig. 11 The outputs of the artificial intelligence model which is responsible for detecting anomalies in the production are shown on the smart production dashboard. The sum of the anomalies that occurred in each hour on each machine is calculated and shown on a matrix on the right corner of the dashboard. This plot helps manufacturing engineers to realize if there is a major problem in one of the hours of the day or one of the machines for several hours

our charts are ready. One condition to active plotly library is that add extension inside javascript code.

On the flask side, we merged dashboards' HTML and js code with python to work with API and much other stuff such as displaying on the website. Flask is very useful for displaying constantly updated data, especially when working with HTML and js. Template Inheritance in Flask permits developers to build a base template that can be overridden

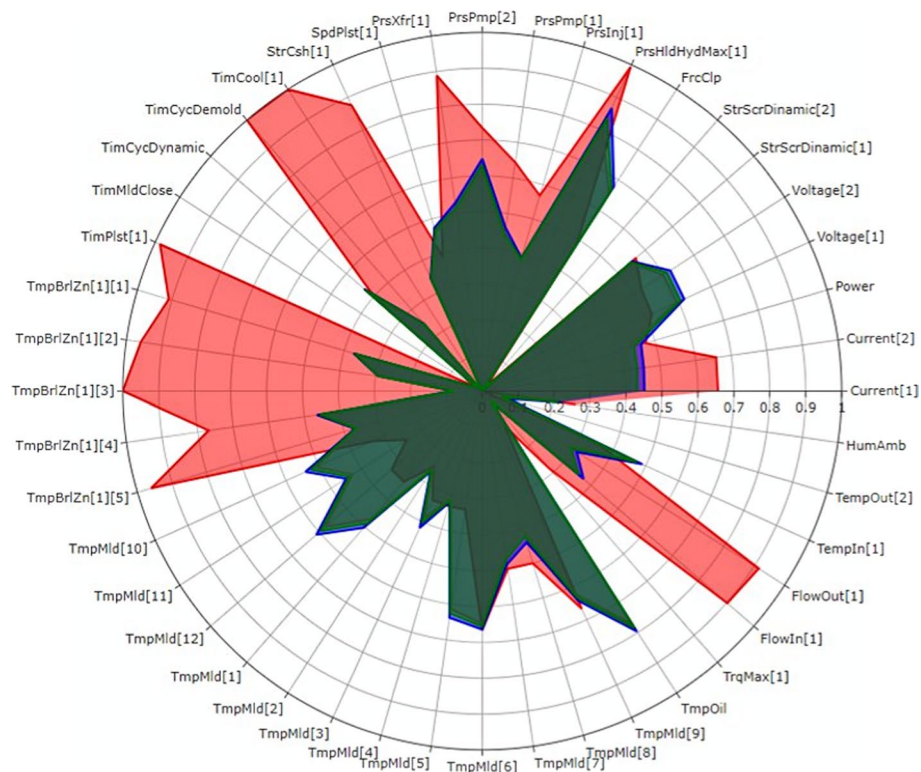


Fig. 12 The radar map compares the quality product with the abnormal product in terms of the process parameters. The scaled process parameters are placed around the circle. While the green area indicates the quality product, the red area shows the abnormal one. The abnormal product can be analyzed in detail using this radar map

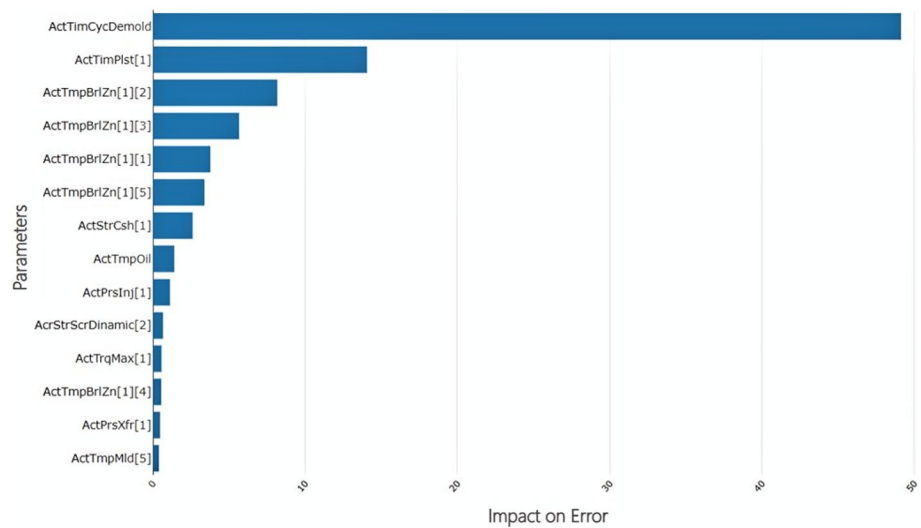


Fig. 13 The ranking of the impact of the parameters on the anomaly. 14 parameters that have more effect than other process parameters on the anomaly of a sample product are listed on the bar chart

by child template thus in dashboard1 we inherited a base_dashboard.html. Render_template function helps us to use @app.route.

As can be seen on Fig. 11 there are 5 charts in dashboard2. They represent 5 data from linked machines; Energy Track, Anomaly Situations, Alarms, Anomaly Situation List, and Anomaly Situation Number Track. An indicator chart to show daily consumption and its rate to weekly consumption. For anomaly situations in each machine, radar charts have been used. With javascript capability, the tables have been created. And lastly, a Heatmap with Categorical Axis Labels has been created to show the Anomaly Situation Number Track with rising importance colors. Dashboard1 graph data shows the most troubled parameters and the last 10 cycles. Each production machine has its own data to be examined. If it's functioning well the color will be seen green if not it will turn yellow or red according to problems. The tables section demonstrates Cycle time, anomaly parameter, and when they have been analyzed. Also, we can deduce meanly and momentary consumption from tables.

AI based real time anomaly detection

It is very important to track the production process in production lines, for producing defect-free parts, and to detect errors. Despite the warnings given by the machines at specified thresholds and the operators' knowledge of the process, it is not possible to understand the events taking place in the capillaries of the machine with a human eye. In the study, with the data collected from different sources on a per-second and cycle basis, effects of small process variable differences on the part quality are aimed to be detected. In addition, parameters that have the most effect on the process anomaly will be analyzed.

For each cycle, the anomalies that occur during the production of the part are detected. For the cycles that have anomalies, the parameters are compared with the golden cycle. In the study, process parameters such as temperatures, pressures, forces, strokes, and cycle time are collected from the injection molding machine, ambient temperature and

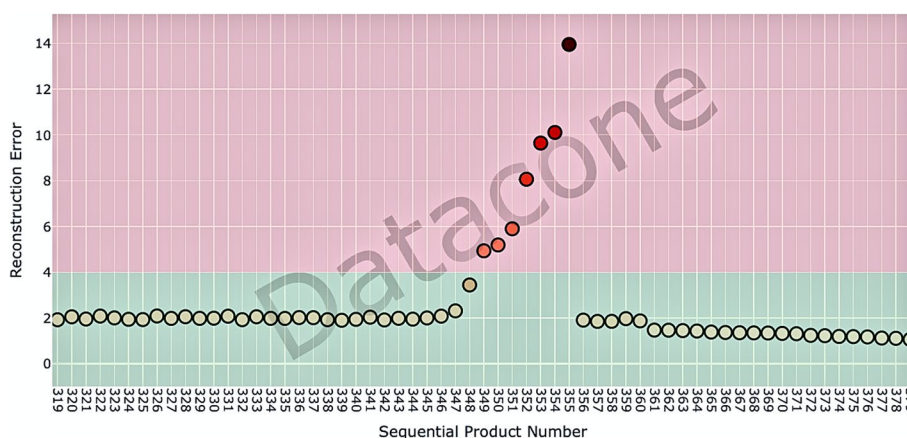


Fig. 14 This figure shows autoencoder algorithm based reconstruction error of 60 product that produced sequentially. The reconstruction error threshold is determined as 4. The products which has reconstruction error more than 4, are labeled as abnormal products. Its clearly seen on the plot, there is some issue between 348. and 356. products

ambient humidity data are collected using the ambient sensor. An artificial neural network algorithm called Autoencoder is chosen to be used for unsupervised learning. With the anomaly detection algorithm developed based on Autoencoder, the pattern of the process data is continuously followed and comparisons are made for each cycle. If there is a cycle that progresses differently from the general pattern, it is recorded as an abnormal cycle. With the detected anomalies, the further downstream of the process is predicted.

Figure 12 shows how the abnormal cycle differs from the general pattern. This analysis is the first indication of the abnormality that started in the process. Figure 13 shows which parameters caused this abnormality the most. With this developed algorithm, the heartbeat of the machine is controlled and warnings are given on the screens. In Fig. 14, it is observed how the process starts to change after the abnormality is detected and the process goes abnormally within a certain range according to reconstruction error.

The study shows how small parameter differences in process parameters affect production. While warning systems can only be created with sharp increases in the parameters determined in classical systems, with this approach all parameters can be evaluated together in real-time with process data and differences can be determined. With this more holistic approach, it is aimed to prevent the process that will cause errors, to provide energy efficiency, and to increase efficiency by reducing both quality and cost.

Discussion and evaluation

With this case study, a big data platform was developed for the manufacturing company, which the company did not have before. Thanks to this platform, large volume and high frequency data were collected from many equipments in the production area. With the platform, the traceability of the process in the field has been increased and the collected data has provided input to many analytical studies. By making the data manageable and traceable, productivity gains have been achieved in many different areas such as error detection, root cause analysis and process design. At the same time, the collection and accumulation of data plays an important role for future artificial intelligence studies. Data analysis software has been integrated to be used in studies to analyze the data collected within the platform. Thanks to these software technologies, it is aimed to shorten the project durations by creating a collaborative working platform for data analysts, data scientists and data engineers. New devices and data sources can be integrated into the created platform quickly and easily. In this way, the integration stages, which are the most critical stages of data projects, can be completed in a short time. With the artificial intelligence-based anomaly detection application developed and implemented entirely on the platform, the quality control process, which is a very critical and manual process for production, has been automated. In this way, machine time and man-hour savings are achieved.

Conclusion

In conclusion, the growing need to data management at manufacturing industry and demand forecasting is derived by globalization and increasing digital transformation approaches. In this study, we performed a through local open-source methodologies for effective-low cost applications of predictive big data analytics including anomaly detection forecasting example. The study overviewed the real time big data design and analytics methods applied to manufacturing processes and provided a comparative categorization of them. We collected and analyzed these studies with respect to methods and techniques used in production prediction. Several open-source lambda architecture technologies are identified and studied with their pros and cons. The corresponding choices are observed as the common low-cost and easy-applicable techniques, among others. The study also pointed to the fact that data management and real-time optimization models can be used to improve the accuracy of time-dependent analytics, sequential time-series prediction, through formulating and optimizing a cost function for the fitting of the predictions to data in real time nature. One key finding from our study the existing literature was that there is a very limited research conducted on the applications of open-source designed, a real-time predictive big data analytics in large scale plastic injection moulding machines in manufacturing at scale and reverse applications for production, maintenance, and other deployments. As a result, we propose a big data architecture design, DataCone, using low-cost effective on-premise open source big data technologies to meet the needs of manufacturing companies in many operations. All in all, this work explored how operational detection approaches alongside different types of predictive maintenance algorithms, autoencoder, can be used for the abnormal cycle differs from the general pattern. This alone shows how small parameter differences in process parameters affect production. While warning systems can only be created with sharp increases in the parameters determined in classical systems, with this approach all parameters can be evaluated together in real-time without latency in the application.

Future works

In our next publications, big data architectures and machine learning models, which are implemented to the continuous production line, will be explored. Continuous and cellular production will be compared in terms of architectures, pipelines, and analytics of big data. In addition, operational cost optimization of the big data platforms will be studied.

Acknowledgements

It is dedicated to the memory of my beloved father, Süleyman Turhan Çakır, who has always supported me throughout my life.

Author contributions

AÇ designed the novel big data architecture of DataCone which is suitable for manufacturing and he calculated the hardware requirements of infrastructure. ÖA and HFD installed and integrated the software which is indicated in big data architecture design and they developed data pipelines. AY analyzed the production data and contributed to developing anomaly detection model and design of the smart production dashboards. All authors read and approved the final manuscript.

Funding

Not applicable.

Availability of data and materials

The datasets generated and/or analysed during the current study are not publicly available due to the confidentiality of company data but an anonymized sample of the data are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Received: 10 March 2022 Accepted: 29 November 2022

Published online: 14 December 2022

References

- Mourtzis D, Vlachou E, Milas N. Industrial big data as a result of iot adoption in manufacturing. *Procedia CIRP*. 2016;55:290–5. <https://doi.org/10.1016/j.procir.2016.07.038>.
- Liu R, Isah H, Zulkernine F. A big data lake for multilevel streaming analytics. 2020 1st International Conference on Big Data Analytics and Practices (IBDAP). 2020 1st International Conference on Big Data Analytics and Practices (IBDAP) (2009). <https://doi.org/10.1109/IBDAP50342.2020.9245460>.
- Belhadi A, Zkik K, Cherrafi A, Yusof SM, Fezazi SE. Understanding the capabilities of big data analytics for manufacturing process: insights from literature review and multiple case study. *Comput Ind Eng*. 2019. <https://doi.org/10.1016/j.cie.2019.106099>.
- IBM Analytics: IBM Industry Model support for a data lake architecture (2016). <https://www.ibm.com/downloads/cas/DNKPJ80Q> Accessed 26 Apr 2021.
- Tao F, Qi Q, Liu A, Kusiak A. Data-driven smart manufacturing. *J Manuf Syst*. 2018;48:157–69. <https://doi.org/10.1016/j.jmsy.2018.01.006>.
- Shao G, Jain S, Shin S-J. Data analytics using simulation for smart manufacturing. *Proceedings of the Winter Simulation Conference*. 2014. <https://doi.org/10.1109/WSC.2014.7020063>.
- Syafrudin M, Fitriyani NL, Li D, Alfian G, Rhee J, Kang Y-S. An open source-based real-time data processing architecture framework for manufacturing sustainability. *Sustainability*. 2017. <https://doi.org/10.3390/su9112139>.
- Dai H-N, Wang H, Xu G, Wan J. Big data analytics for manufacturing internet of things: opportunities, challenges and enabling technologies. *Enterprise Inf Syst*. 2019. <https://doi.org/10.1080/17517575.2019.1633689>.
- Wilcox T, Jin N, Flach P, Thumim J. A big data platform for smart meter data analytics. *Comput Ind*. 2019;105:250–9. <https://doi.org/10.1016/j.compind.2018.12.010>.
- White T. *Hadoop: The Definitive Guide*. Sebastopol: O'Reilly Media Inc; 2009.
- Murthy A, Vavilapalli VK. *Apache Hadoop YARN*. Upper Saddle River: Addison-Wesley; 2014.
- Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A, Meng X, Rosen J, Venkataraman S, Franklin MJ, Ghodsi A, Gonzalez J, Shenker S, Stoica I. *Apache spark: a unified engine for big data processing*. *Commun ACM*. 2016. <https://doi.org/10.1145/2934664>.
- Kuč R, Rogoziński M. *Mastering Elasticsearch*. Birmingham: Packt Publishing; 2015.
- https://en.wikipedia.org/wiki/Apache_NiFi Accessed 7 Feb 2021.
- Pandya A, Kostakos P, Mehmood H, Cortes M. Privacy preserving sentiment analysis on multiple edge data streams with apache nifi. In: *Proceedings of European Intelligence and Security Informatics Conference (EISIC) (2019)*. <https://doi.org/10.1109/EISIC49498.2019.9108851>.
- Samal B, Panda M. Real time product feedback review and analysis using apache technologies and nosql database. *Int J Eng Comput Sci*. 2017. <https://doi.org/10.18535/ijecs/v6i10.04>.
- Soner K, Upadhyay H. A survey: Ddos attack on internet of things. *Int J Eng Res Dev*. 2014;10(11):58–63.
- Kreps J, Narkhede N, Rao J. *Kafka: a distributed messaging system for log processing*. In: *Proceedings of the NetDB, Athens, Greece 2011*. <https://kafka.apache.org/intro> Accessed 24 Mar 2021.
- What is Elasticsearch. <https://www.elastic.co/guide/en/elasticsearch/reference/master/elasticsearch-intro.html> Accessed 2 Apr 2021.
- Mu C, Zhao J, Yang G, Zhang J, Yan Z. Towards practical visual search engine within elasticsearch. 2018. [arxiv:1806.08896](https://arxiv.org/abs/1806.08896).
- Srivastava A, Miller D. *Elasticsearch 7 Quick Start Guide*. Birmingham: Packt Publishing; 2019.
- Kuc R, Rogoziński M. *Elasticsearch Server*. Birmingham: Packt Publishing; 2014.
- Chiary MR, Anand R. Hadoop cluster on linode using ambari for improving task assignment scheme running in the clouds. *Int J Comput Sci Inf Technol*. 2015;6(1):586–9.
- Erraissi A, Belangour A, Tragha A. A big data hadoop building blocks comparative study. *Int J Comput Trends Technol*. 2017;48(1):36–40. <https://doi.org/10.14445/22312803/IJCTT-V48P109>.
- John T, Misra P. *Data Lake for Enterprises*. Birmingham: Packt; 2017.
- Salloum S, Dautov R, Chen X, Peng PX, Huang JZ. Big data analytics on apache spark. *Int J Data Sci Anal*. 2016;1:145–64.
- Shoro AG, Soomro TR. Big data analysis: Ap spark perspective. *Global J Comput Sci Technol* 2015;15(1).
- Above the clouds: A berkeley view of cloud computing. Technical report, University of California at Berkeley. 2009. <https://databricks.com/glossary/what-is-parquet> Accessed 17 Mar 2021.
- <https://www.geeksforgeeks.org/mongodb-an-introduction/> Accessed 4 Jul 2021.
- Mongodb—a comparison with nosql databases. *Int J Sci Eng Res*. 2016.

33. Beauchemin M. Airflow: a workflow management platform. <https://medium.com/airbnb-engineering/airflow-workflow-management-platform-46318b977fd8> Accessed 12 Mar 2021.
34. Srivastava A, Azarmi B. Learning Kibana 7: Build Powerful Elastic Dashboards with Kibana's Data. Birmingham: Packt Publishing; 2019.
35. Build visualizations simply and intuitively. <https://www.elastic.co/kibana> Accessed 11 Mar 2021.
36. Flask Web Development, One Drop At A Time. <https://readthedocs.org/projects/flask/> Accessed 16 Apr 2021.
37. About EUROMAP. <https://www.euromap.org/about-us/about-euromap> Accessed 13 Jan 2021.
38. Plastics and Rubber Machinery. <https://opcfoundation.org/markets-collaboration/plastics-and-rubber-machinery/> Accessed 13 Jan 2021.
39. Jolt. <https://github.com/bazaarvoice/jolt> Accessed 16 Feb 2021.
40. Mitchell R, Loic Pottier SJ, da Silva RF. Exploration of workflow management systems emerging features from users perspectives. In: IEEE International Conference on Big Data. 2019. <https://doi.org/10.1109/BigData47090.2019.9005494>.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- ▶ Convenient online submission
- ▶ Rigorous peer review
- ▶ Open access: articles freely available online
- ▶ High visibility within the field
- ▶ Retaining the copyright to your article

Submit your next manuscript at ▶ [springeropen.com](https://www.springeropen.com)
