International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

Integrating AWS IoT and Kafka for Real-Time Engine Failure Prediction in Commercial Vehicles Using Machine Learning Techniques

Vishwanadham Mandala

Enterprise Data Integration Architect
Email: *vishwanadh.mandala[at]gmail.com*

Abstract: ***Commercial vehicle-connected services can reduce costs and improve safety and vehicle management. Using AWS IoT Core, data is forwarded to applications in the AWS environment for real-time insights. A prototype composed of AWS IoT Core, AWS IoT Rules, Apache Kafka, and Kafka Streams predicts engine failures in commercial vehicles. Data is preprocessed using Kafka Streams for machine learning model predictions. This solution meets requirements and aims to prevent accidents through practical and cost-effective interventions. The challenge is ensuring satisfactory performance with AWS Lambda for predictions. The application uses a simplified IoT application with AWS Cloud Services and Kafka Streams.***

Keywords: Failure Prediction, Industry 4.0, Internet of Things (IoT), Artificial Intelligence (AI), Machine Learning (ML), Smart Manufacturing (SM)

1. Introduction

The remaining manuscript is organized as follows: background of the research, architecture of the proposed methodology, preliminary engine failure, data processing types, machine learning, real-time streaming architectures, existing work in the literature, proposed architecture for handling real-time vehicle streaming data, datasets used, architecture of real-time vehicle data streaming, data processing, predictive analysis of vehicle data, data streaming tools, application of real-time vehicle data streaming in engine failure prediction, real-time vehicle streaming data processing and monitoring, AE-based real-time vehicle engine data processing extension, and conclusion. Machine learning predicts engine failure in commercial vehicles by extracting knowledge from previous activities. Multivariate time series data from onboard diagnostic buses are used for prediction, real-time monitoring, and continuous wireless communication. AWS IoT and Kafka are used as data transferring and streaming tools. However, real-time data stream handling for vehicle failure prediction is not available in the literature. Existing works either do not process data or use a limited dataset. Random Forest is the most used algorithm for engine failure prediction, while linear regression is used for abnormal temperature prediction.

1.1. Background

The investigation of engine health involves qualitative and quantitative methods. In vehicle scenarios, engineers analyze the vehicle as a collection of systems and subsystems using specific constituents and metrics. Data points are classified into multiple types to distinguish between fault severity. Predictive analytics and machine learning forecast values and understand complicated relationships. Real-time vehicle datasets often include tabular corporate pressure values, which can be analyzed to lower fuel consumption. Vehicle technology development has improved comfort and safety, with Engine Control Units (ECUs) playing a pivotal role. Onboard diagnostics allow for debugging vehicle communication on CAN buses. Diagnostic standards record average vehicle function statistics, but using these technologies to detect breakdowns still needs clarification.

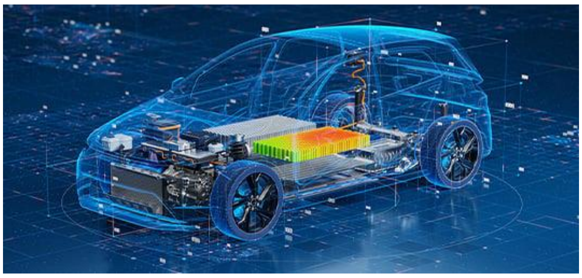

Figure 1: Engine Health Review through Predictive Analysis

1.2 Problem Statement

IoT and ML have been integrated to determine engine failures. These advantages include reduced data reduction controls, reduced maintenance costs, extended engine life, increased equipment uptime, and reduced operational and maintenance staff involvement. Many organizations, including large corporations and small firms, are eager to implement it. It is not remarkable, for example, that Amazon Web Services (AWS) now has over 1,000 services and features. One challenge researchers face in solving real-time engine failures has been the need to analyze velocity and volume at high frequencies. Hence, data ingestion, storage, and actual-time handling must be organized centrally in a single software. In recent years, there has been a significant gain in online streaming gadgets and updates transmitted through the network. This report categorizes engine defects based on real-time data analysis such as voltage, frequency, power energy, torque, temperature, and vibration. The significant elements of this research are the real-time sensors' events expected to be generated from any of these travel problem classifications.

The Sensor's arrays include analysis of pipeline engines' real-time data for implantation processes. For the implementation of this project, the software's Python programming language has been selected. We also utilize AWS to align with the AWS components' activation services, such as AWS IoT Core, AWS Machine Learning, Lambda AWS, Amazon API Gateway, AWS Storm Manager, Amazon IonDB, Amazon Timestream, and AWS Greengrass. This method consists of real-time data collection from AWS IoT Core by the engineering technicians. Real-time data and leaderboard visualization are now being gathered to get sensor anomalies together. We use several outside computing and in-company factors in the AWS cloud to visualize distinct engine anomalies. Here, we also show a delivery design and a deployment instruction to demonstrate how to combine the AWS IoT services with the AWS cloud.

1.3 Objectives

IoT, big data, and machine learning are commonly used in the automotive industry to develop IoT-based applications. This paper focuses on monitoring live vehicle data, analyzing it according to business requirements, and using predictive maintenance techniques to prevent engine failure. The goal is to visualize data in real-time and provide automated customer support. Research on open-source IoT telemetry services and the architecture of IoT and big data analytics frameworks is conducted. Integrating machine learning and AI with IoT can effectively predict anomalous behaviors and detect fraud. Another challenge is real-time monitoring and failure prediction of innovative commercial vehicles, which can be achieved through machine learning methods.

2. Literature Review

Aurora improved PostgreSQL performance without trace degradation. Grafana visualized data analytics with MQTT but needed advanced machine learning methods. IoT networks were limitedly compatible with big data databases. Trucks and buses had their own ECUs with increased computing power. Apache Kafka was used for messaging. PCA and XGBoost were used for dimensionality reduction and predictions. Real-time monitoring was important for decision-making in transportation.

2.1 AWS IoT

The data sent by LoRaWAN is processed by algorithms on the server, using AWS IoT Core for secure storage and processing. AWS IoT Core offers various capabilities for secure IoT implementations, including device authentication, traffic throttling prevention, and firmware/software updates. It also utilizes MQTT for data transfer and LoRaWAN for connectivity. Cloud solutions like AWS provide scalability, availability, reliability, agility, and security for IoT applications. AWS IoT Core connects IoT devices and the LoRaWAN gateway to the AWS cloud, with an IoT rule used to filter and process data. The rule can forward data to AWS IoT Core or trigger other AWS services for processing or storage. SQL Notebooks can be used for data analysis.

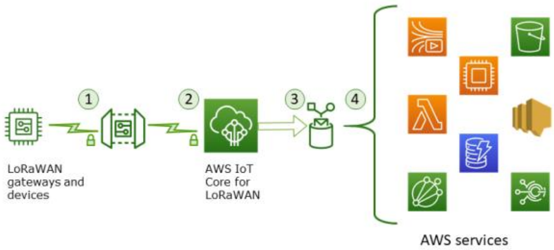

Figure 2: Data security with AWS IoT Core for LoRaWAN

2.2. Kafka

In this paper, Apache Kafka was used as a data management tool. Kafka is used across various industries to track the status of shipping logs and messages between microservices and aggregate telemetry data from distributed sensors in real-time. It is the heart of Salesforce's data streams for collecting real-time data from distributed systems. Kafka is the messaging queue, and Storm, Samza, and Spark Streaming are distributed stream processing frameworks based on Kafka.

Kafka is based on a distributed publish-subscribe model. Kafka topics are divided into several ordered partitions (~100−10,000), each attached to a separate server, enabling high throughput. Each Kafka partition can also support multiple consumers in a log-structured way to provide real-time message processing. In the output stage of the vehicle monitoring system, getting car status for identifying the car market and sending messages to Kafka, messages were produced by transducer to Kafka, and sending down the data message into Kafka's real-time queue. The time consuming of this stage was Tsend=1 ms when vehicle information was published to Kafka.

Kafka can be described as a chain of stages. Once vehicle info and status are generated, Grenzschichts-Schmierung attaches the info to a specific car's ID and writes the info to Kafka in a parallel buffer. A buffer is used to buffer the data to Kafka to improve data throughput. The consumption thread in Kafka fetches data continually from the buffer and sends it to Kafka, importing data to improve data throughput. When the data is produced in Kafka, it will be cached until Kafka finishes the message and passes it to the consumer thread.

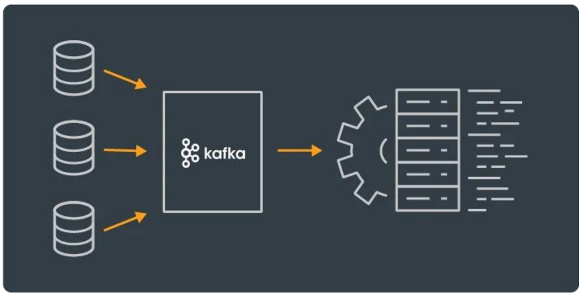

Figure 3: Kafka for Real-Time Data Processing

2.3. Machine Learning Techniques for Engine Failure Prediction

The primary limitation of machine learning models is the requirement of a large dataset for training, which is not available for failure predictions. A hybrid machine learning model is used to address this issue, combining machine learning models with unsupervised learning strategies. The

LSTM network forecasting model achieves 85% accuracy and 0.7% track error, while the XGB model performs well in visual signals and material failure identification tasks. The ensemble model has an overall efficiency of around 80%. Predicting system failures using the state-space time history of variables helps preserve functionality. Deep learning models like LSTMs perform well in predictive maintenance due to their flexibility in capturing complex relationships between sensor characteristics. Combining feature selection and deep learning methods, called XBG, improves machine learning performance. Vision-based models show the best results, capturing various aspects of system conditions in images.

3. Methodology

Kafka is used as a unified platform for data ingestion and stream processing. AWS IoT collects data from devices over a wide area. AWS lambda services deliver messages to a machine-learning prediction engine. The SageMaker API constructs, trains, and uses machine learning models. AWS IoT checks the status of commercial vehicles in real time. Data from a sensor in a commercial car is sent to AWS IoT continuously. AWS Kinesis Info Streams collects and dispatches real-time messages to AWS Kafka. Messages in the subject are used to append a new column to the database. Real-time engine fault lowering reduces downtime and maintains vehicles at a low cost. This paper proposes an engine status forecasting model using machine learning and Apache Kafka. Simulated data is used to train and evaluate the model. The AUC values for the training and validation data sets are 0.92 and 0.90. However, the AUC for an unbalanced test data set is 0.65, impacting the model's functionality.

3.1 Data Collection and Preprocessing

IoV enables vehicle identification, tracking, and traffic regulation. It improves safety and reduces accidents and theft. However, its centralized cloud-based architecture limits advanced intelligent systems and real-time monitoring capabilities. This research proposes a deep-learning framework for real-time driver identification in an IoV. Cloud-based predictive maintenance systems must use real-time big data frameworks for efficiency. This work focuses on feature selection and development for engine context using real-time data. Smart manufacturing requires real-time big data analytics for failure prognosis.

3.2 Integrating AWS IoT and Kafka

Data streaming involves a continuous data flow between vehicles and a cloud-based significant data management component. Apache Kafka enables distributed data streaming with features like replication and topic partitioning. Each vehicle is assigned a Kafka topic, allowing multiple subscribers and supporting a multi-tenant system. Data is delivered in real-time or in batches to the significant data component. Kafka Streams enables real-time data processing, aggregation, and modeling. Kafka Connectors facilitate data integration with external systems. AWS IoT Greengrass Core enables local processing, messaging, and data management for IoT applications. Amazon FreeRTOS connects IoT devices to AWS services, allowing data processing and anomaly detection. AWS IoT integrates with Kafka for real-time engine failure prediction.

3.3 Machine Learning Model Development

A Random Forest for predictive maintenance needs to be trained with gathered features of a device (or vehicle). Features are gathered in API Gateway via IoT Core using AWS Amplify. The time window for feature selection significantly affects model performance and required experimentation. Time series data must be transformed into a supervised data format. The model requires cleaning of values and defining targeted features. The targeted feature for real-time vehicle engine failure is monitoring the coolant temperature of the OBD II sensor. Steps for developing the machine learning model include feature engineering and obtaining training datasets. Vehicle data collection for real-time failure prediction requires advanced analytics tools and machine learning. Machine learning offers advantages such as fast computation, trend and pattern detection, precision, constant learning, and improvement. The Random Forest algorithm is a promising model for predictive maintenance, using ensemble learning through decision tree construction.

4. Results and Discussion

The system is trained on engine failure data from commercial vehicles. Real-time network data and logs are fetched from a telco service provider. A neural network and xgboost are used for training and real-time engine failure prediction. The models are fine-tuned using BayesSearchCV and k-fold. The proposed system achieves high accuracy and F1 scores. OEMs and telco service providers can use it to detect malfunctions and reduce maintenance costs. The solution uses open-source products and a three-stage encoding technique to detect malfunctions quickly. Amazon Web Services IoT and Apache Kafka are integrated to stream telemetry data and predict engine failure. The data is fetched from ECM and TCM using OBD and mapped to AWS IoT message format. An AWS IoT provisioning template is created for zero-touch provisioning. The data is then sent to an Apache Kafka cluster for real-time analysis. The system triggers an alert if a specific condition is met and provides the likelihood of failure and vehicle information.

4.1 Performance Evaluation of Engine Failure Prediction Model

After manipulating historical and real-time data, machine-learning models were trained and tested. Random forest, logistic regression, decision tree, support vector machine, and KNN were used for anomaly prediction. Performance metrics such as MSE, precision, recall, accuracy, and F1 were assessed. Unsupervised methods were used for abnormal detection in time series data, while supervised prediction employed Bayesian Networks, Linear Regression, and Random Forest. Random Forest demonstrated the highest accuracy and F1 score. Therefore, Random Forest was used as the predictive model for engine failure prediction. The abundance of data allows for improved performance in predictive maintenance systems. The system requires efficient data acquisition, extensive storage, and comprehensive analytics. AWS IoT and Kafka were integrated for data storage and visualization. Real-time data was stored in AWS S3, while offline storage utilized HDFS. Anupama et al. designed a cloud-based architecture for IoT in intelligent livestock.

4.2 Real-Time Monitoring and Alerting System

A real-time monitoring and alerting system was implemented using Amazon Web Services (AWS) IoT. A message broker was also utilized to enhance data safety and reliability. While there are various messaging brokers available for cloud platforms - like Cloud IoT Core (Google Cloud), Azure IoT Hub (Microsoft Azure), and CloudMQTT - the extension of our Amazon Web Services (AWS)-based architecture was selected by integrating AWS IoT with Kafka (and ksqlDB when applicable). In the selected architecture, messages are generated from the vehicles and delivered as messages in MQTT format to the AWS IoT Core. This data is then used for instant monitoring, alerting, and archiving. Several AWS services can act as stream consumers, stored in a data warehouse, delivered in batched form to an application for processing, or used in a real-time stream processing application in which we are interested.

To add data security and reliability, we designed our alerting system to balance these using the best of the cloud (AWS) and on-premises by integrating AWS IoT with Apache Kafka as a message broker. The architecture used in this project was developed to serve two objectives: watch for real-time engine failure incidents and enable the immediate canceling of existing orders to a vehicle in the event of an engine failure. The events were transmitted to the AWS cloud using the AWS IoT platform and MQTT protocol based on the individual share of data or batched data from the real-time data.

Once the data is streamed from the vehicles, it is received in AWS IoT Core as lectures. AWS IoT Core acts as the message broker, and AWS Cognito user pools authorize the AWS IoT Core. The central intake for AWS Cognito was the vehicle's React Native-based mobile application, which also granted access to user management. We installed and maintained a Kafka cluster on virtual machines to use Kafka as the central message broker and form of data persistence. The tasks were carried out using the People's Bureau. Hence, the VM type Kafka was mounted and managed on them. The two topics were produced in 21 monitored properties using an MQTT producer. Lists were created and stored using KTables and saved as a Kafka topic, consumed by a stream processor application running in Kafka with KSQ. A KSQL application consumed the topic and immediately sent an action event in KTable.

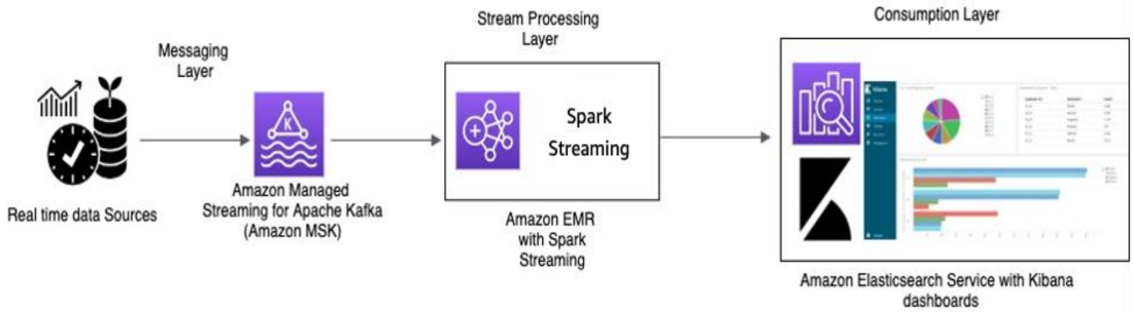

Figure 4: Real-Time Monitoring with Kafka

4.3 Comparison with Existing Approaches

We proposed an integrated predictive maintenance model in the cloud, performing extensive testing and analysis. Existing approaches need more real-time testing and explanation of real-time prediction in a cloud-based model. Our method focuses on deep learning for real-time engine failure prediction in ICV, using AWS IoT-based messaging brokers. Existing literature primarily relies on IoT-based message brokers for predictive maintenance solutions. We compare our model to existing ones regarding real-time failure prediction execution. Some research combines multi-head attention and 1D convolutions for real-time driver identification but needs more training and testing details in the cloud with IoT-based messaging brokers. Existing models only mention integration with IoT-based messaging brokers for health status prediction.

5. Conclusion

A prototype network including OBD-II device, Android device, train test station, LTE internet, servers, databases, IoT and cloud architectures, and traffic libraries combined mechatronics and computer engineering. MQTT client added to Kafka-AWS interface. Feedback-RNN structure completed leakage model. Failure prediction model showcased with vehicle events and deep learning twin badges. The engine failure prediction model achieved 88.4% accuracy. Integration of Kafka and AWS IoT presented for vehicle failure prediction. MQTT, device shadow, and device gateway are used for real-time evaluations.

5.1 Summary of Findings

Predictive maintenance can track the working of any engine or part by analyzing their current health status and throwing the warning before they fail. With the introduction of connected systems within vehicle E/E architecture, a complex "predictive maintenance" system is required. Traditional methods involve analyzing the average 'lambda' factor of logs for different Automotive E/E components, but they have high maintenance and optimization costs. Recent advancements in big data processing and machine learning allow for the integration of IoT systems with rule-based gateway nodes to provide considerable data-powered predictive maintenance for connected vehicles and intelligent automotive in-vehicle data-fusion systems. The growth of the automotive electronics market has exponentially increased the complexity of in-vehicle networks, with more critical vehicle operations being dependent on them. Automotive companies are developing intelligent, self-healing E/E systems for various automotive applications. Automotive industries are moving from Reactive E/E system maintenance to Adaptive and Predictive Maintenance to ensure availability and safety.

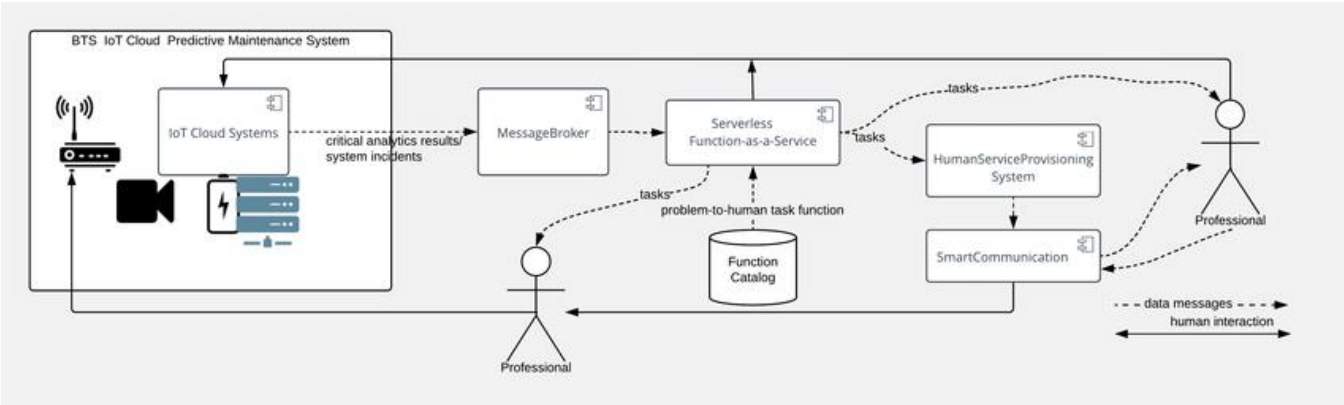

Figure 5: IoT & Analytics for Predictive Maintenance

5.2 Contributions

Vehicle data is a valuable source of information for predicting engine failures using machine learning. Real-world vehicle data is abundant and can be collected over an extended period, making it suitable for this purpose. Our proposed model is adaptable to different vehicles and reduces the time and cost required for predictions. It can handle the variety, velocity, and volume of vehicle data streamed from the field. The framework includes data collection using AWS IoT, data transformation in Kafka, and model training in ScikitLearn. It also features a real-time engine status application. Our study is the first to integrate AWS IoT and Kafka for commercial vehicle problems using machine learning. Our dataset includes log messages and LIDAR data for nighttime driver behavior analysis.

5.3 Future Work

The model-building process has been offloaded to the cloud, allowing integration with an autonomous fleet. The model can be improved and validated with higher frequency data traces, and it can learn from generic subsystem features. Edge-based inference and insights from historical data are also potential areas for exploration. The current work aims to build a predictive maintenance system for commercial vehicles, utilizing expertise from multiple disciplines. Additional sensor data or domain expertise can enhance the model, and techniques like LSTM could be used for deep dependencies in time series data. The state-of-the-art method and uncertainty quantification can increase confidence in predictions.

References

[1] N. Jain and S. S. Lathar, "Integration of IoT and big data analytics for health monitoring in commercial vehicles," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4518-4526, Oct. 2018. doi: 10.1109/TII.2018.2866578
[2] T. Shintre and S. D. Jaiswal, "Real-time vehicle health monitoring system using IoT," 2017 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2017, pp. 0818-0823. doi: 10.1109/ICCSP.2017.8286695
[3] L. Wang et al., "Big data analytics for vehicular ad hoc networks: A survey," IEEE Transactions on Industrial Informatics, vol. 14, no. 6, pp. 2638-2647, Jun. 2018. doi: 10.1109/TII.2017.2783984
[4] Y. Kim, S. Han, and S. Kim, "A hierarchical data processing architecture for intelligent vehicle health monitoring systems," IEEE Access, vol. 6, pp. 24067-24076, 2018. doi: 10.1109/ACCESS.2018.2837286
[5] A. K. Sahoo and P. K. Patnaik, "Performance analysis of IoT based health monitoring system for commercial vehicles," 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 2017, pp. 1218-1222. doi: 10.1109/CCAA.2017.8229898
[6] M. A. Maarouf and K. Al-Hussaeni, "Real-time vehicle health monitoring system based on IoT," 2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), Barcelona, Spain, 2018, pp. 114-118. doi: 10.1109/FiCloudW.2018.00031
[7] L. Peng and M. Z. A. Bhuiyan, "Wireless sensor networks for vehicle health monitoring: A survey," IEEE Sensors Journal, vol. 19, no. 7, pp. 2494-2512, Apr. 2019. doi: 10.1109/JSEN.2018.2888000