J. Electrical Systems 20-11s (2024): 3175-3181

Ruby Dinakar[1]*
Dr. Vagdevi S[2]

Real-Time IoT Sensor Data Streaming and Processing with Apache Flink: A Scalable Solution for Smart Monitoring

Journal of Electrical Systems

Abstract
The Internet of Things (IoT) has revolutionized data-driven decision-making by enabling real-time data acquisition through an extensive network of sensors. However, the massive influx of continuous, high-velocity sensor data poses significant challenges for traditional data. The rapid explosion of sensor data, necessitating robust, scalable real-time data processing solutions. This paper presents a comprehensive solution for ingestion, processing, and analysis of large-scale sensor data streams in real time, based on Apache Kafka and Apache Flink. The framework handles continuous data flows with low latency and high throughput by leveraging Kafka's distributed streaming capabilities and Flink's advanced data processing features. This system is especially well-suited for applications that require immediate feedback, such as environmental monitoring, industrial automation, and smart city infrastructures. We provide a detailed case study that demonstrates the framework's efficiency in handling and processing high-velocity sensor data, emphasizing its potential to significantly improve decision-making processes in dynamic, data-driven environments. The findings highlight the framework's scalability and resilience, presenting a solid solution for organizations looking to use real-time analytics in their operations.

Keywords: Sensor data, Microservices, Real time streaming, Sensor Analytics, Apache Flink, Deep learning, Event driven Architecture

1.INTRODUCTION

The Internet of Things (IoT) has revolutionized the way we interact with the world, integrating digital and physical domains through an extensive network of interconnected devices. These devices, ranging from simple sensors to complex machinery, are embedded with electronics, software, and connectivity, enabling them to collect, exchange, and act upon data. The proliferation of IoT devices has been driven by advances in technology, reductions in sensor and connectivity costs, and the increasing demand for smart solutions across various sectors such as healthcare, manufacturing, transportation, and environmental monitoring. The growth trajectory of IoT devices has been exponential. Estimates suggest that the number of connected IoT devices worldwide is expected to surpass 30 billion by 2025. This surge is generating unprecedented volumes of data, often termed "big sensor data," which includes diverse data types such as temperature readings, motion detections, video feeds, and more. The sheer volume, velocity, and variety of this data pose significant challenges in terms of storage, processing, and real-time analysis.

1.1 Challenges in Processing Large-Scale Sensor Data in Real-Time

The rapid accumulation of big sensor data presents several critical challenges that need to be addressed to fully leverage the potential of IoT systems:

Data Volume and Velocity: The continuous and rapid generation of data from billions of devices creates massive datasets that traditional data processing systems struggle to handle. This data needs to be ingested and processed in real-time to provide timely insights and actions, especially in scenarios like emergency response, industrial automation, and smart city management.

Scalability: The infrastructure must scale dynamically to accommodate the growing number of devices and the corresponding increase in data flow. This requires scalable data ingestion pipelines and processing systems that can handle peak loads without degradation in performance.

Latency: For many applications, especially those involving real-time monitoring and control, low latency is crucial. Delays in processing data can lead to missed opportunities or even dangerous situations, such as in healthcare monitoring or autonomous driving.

Data Heterogeneity: IoT ecosystems involve a wide range of devices producing diverse data types (e.g., structured, unstructured, time-series, spatial). Processing systems must be versatile enough to integrate and analyse these varied data formats efficiently.

Fault Tolerance and Reliability: IoT systems must be robust, ensuring data integrity and continuous operation even in the face of hardware failures, network issues, or software errors. This is particularly important for critical applications like disaster management and public safety.

Data Security and Privacy: With the influx of data from numerous devices, ensuring the security and privacy of sensitive information becomes paramount. Effective data processing systems must include mechanisms to protect against data breaches and unauthorized access.

[1]*VTU Research Scholar, Faculty of CSE, PES University, Bangalore, Email: rubydinakar@gmail.com
[2]Professor & Head AIML dept, City Engineering College, Bangalore
***Corresponding Author:** Ruby Dinakar
**VTU Research Scholar, Faculty of CSE, PES University, Bangalore, Email: rubydinakar@gmail.com*

1.2 The Need for Advanced Processing Frameworks

The rapid expansion of IoT ecosystems and the consequent explosion in sensor data volume present unique challenges that conventional data processing systems are ill-equipped to handle. This paper addresses several specific challenges associated with real-time processing of large-scale sensor data. IoT devices continuously generate vast amounts of data, often in the form of high-frequency streams. Traditional batch processing systems are unable to cope with this deluge, resulting in data bottlenecks and processing delays. The proposed solution leverages Apache Kafka for its high-throughput data ingestion capabilities, which allows it to handle millions of data events per second. Kafka's distributed architecture enables horizontal scaling, ensuring that the system can accommodate increasing data loads without compromising performance. Many IoT applications require real-time or near-real-time data processing to facilitate immediate responses, such as in predictive maintenance, real-time monitoring, and emergency alert systems. However, achieving low-latency processing is difficult due to the computational complexity of analyzing large-scale data streams in real-time. Apache Flink, known for its real-time stream processing capabilities, is integrated into the framework to ensure low-latency data processing. Flink's event-driven processing model, coupled with its support for complex event processing, stateful computations, and windowing, enables the framework to deliver timely insights and actions. IoT environments generate heterogeneous data, including structured, semi-structured, and unstructured formats. This diversity complicates data integration, storage, and analysis. This solution is designed to handle various data types seamlessly. Kafka facilitates the ingestion of diverse data streams, while Flink's versatile data processing capabilities allow for the transformation, enrichment, and analysis of these heterogeneous datasets. This ensures that the system can effectively manage and utilize data from different sources and formats. Ensuring continuous operation and data integrity in the face of hardware failures, network issues, or software errors is critical, particularly in mission-critical applications like healthcare monitoring or industrial control systems. The framework employs Kafka's robust data replication and fault tolerance mechanisms to maintain data availability and reliability. Flink complements this by providing state management features that ensure consistency and recovery in case of failures, thus maintaining uninterrupted data processing. Optimizing the use of computational and storage resources is essential to manage costs and ensure system efficiency, especially when dealing with large-scale data. The framework utilizes Kafka's efficient data storage and streaming capabilities, along with Flink's adaptive resource management and task scheduling. This combination enables efficient resource allocation, reducing the overall computational and storage overhead. The collection and processing of sensitive data, especially in applications like healthcare or smart cities, raise significant concerns regarding data security and privacy. Unauthorized access or data breaches can lead to severe consequences. The framework integrates security measures, including encryption, access controls, and secure communication protocols, to protect sensitive data throughout its lifecycle. Kafka and Flink provide built-in features and extensions that support these security measures, ensuring compliance with data protection regulations and standards. This paper not only addresses these challenges but also demonstrates the practical application and effectiveness of the proposed Kafka-Flink framework through a detailed case study, showcasing its potential to transform how organizations manage and analyze real-time sensor data.

The primary objective of this study is to design and evaluate a robust framework for real-time processing of large-scale sensor data streams, leveraging the strengths of Apache Kafka and Apache Flink. The study aims to achieve the following specific goals. It aims to 1. develop a system capable of ingesting and processing continuous streams of sensor data in real-time, ensuring low-latency responses and high throughput. 2. Create a scalable architecture that can easily accommodate increasing data volumes and varied data types, adapting to the dynamic nature of IoT environments. 3. Ensure the system is resilient to failures, providing robust fault tolerance mechanisms to maintain data integrity and continuous operation. 4. Optimize the use of computational and storage resources, minimizing operational costs while maintaining high performance. 5. Implement comprehensive security measures to protect sensitive data and ensure compliance with data privacy regulations. The proposed framework makes several unique contributions to the field of real-time big data analytics and IoT data management. This framework suggest solution for efficient and scalable processing of continuous data streams. It incorporates complex event processing, stateful computations, and event-time windowing, enabling sophisticated analysis and decision-making based on real-time data. This capability is particularly crucial for applications requiring immediate responses, such as anomaly detection and predictive maintenance. This framework is designed to handle a wide variety of data formats, including structured, semi-structured, and unstructured data. This versatility ensures that the system can integrate data from diverse sources, providing a comprehensive view of the IoT ecosystem. 4. The study demonstrates how the combination of Kafka's distributed architecture and Flink's state management can be used to build a system that is both scalable and fault-tolerant. The framework's ability to handle large-scale data streams and recover from failures without data loss or downtime is a significant advancement over traditional data processing systems. 5. It includes mechanisms for dynamic resource allocation and efficient task scheduling, which optimize the use of computational and storage resources. This aspect is critical for reducing the total cost of ownership and ensuring the economic feasibility of large-scale deployments. 6. By incorporating data encryption, secure communication protocols, and access controls, the framework addresses the critical need for data security and privacy in IoT applications. This contribution is especially important for sectors like healthcare and finance, where data protection is paramount. 7. The paper includes a detailed case study that demonstrates the practical application and effectiveness of the framework in a real-world scenario. This case study not only validates the technical contributions of the framework but also provides insights into its deployment and operational benefits. Through these contributions, the proposed framework offers a comprehensive solution for real-time big data analytics in IoT environments, addressing key

challenges and paving the way for more intelligent, responsive, and secure IoT systems.

Microservices architecture has become a prevalent architectural pattern for building scalable and modular systems. It breaks down applications into smaller, loosely coupled services, each of which is responsible for a particular functionality. Event driven Micro services architecture is proposed as a solution for real-time streaming analytics on sensor data streams. Utilizing the benefits of microservices architecture, enables the efficient and scalable processing of high-velocity sensor data. It addresses the challenges of real-time data ingestion, processing, and storage by decomposing the analytics pipeline into smaller, autonomous microservices. The purpose of this paper is to provide a comprehensive understanding of micro services architecture, its design principles, and its application in real-time streaming analytics on sensor data streams. This paper contributes to the advancement of real-time analytics on sensor data by highlighting its unique capabilities, enabling organizations to make informed decisions and extract actionable insights in real-time.

1.3 Challenges

Performing analytics on sensor data streams in real time presents a number of challenges. The high volume, velocity, and variety of sensor-generated data, as well as the need for timely processing and analysing these data to generate actionable insights. Traditional batch processing methods are ineffective to dynamic event responses. Sensor data streams can generate a continuous data flow that must be ingested in real time. The conventional method of storing data in a database or data warehouse prior to processing is impractical for real-time analytics on sensor data streams. Consequently, there is a need for a scalable and efficient data ingestion mechanism that can handle a large volume of streaming data. It is crucial to design an architecture capable of processing and analysing high-velocity sensor data in real-time, ensuring low latency and accurate results. In addition, storing and managing sensor data for subsequent analysis and retrieval is a major concern. Robust data storage mechanisms are required for storing and organizing large volumes of streaming data while maintaining its integrity, availability, and scalability. For real-time analytics on sensor data streams, traditional relational databases may lack in performance and scalability.

In light of these obstacles, there is a need for an architecture capable of ingesting, processing, and storing real-time sensor data stream. Using the microservices pattern to provide scalability, fault tolerance, and modularity for real-time streaming analytics on sensor data, the Micro stream architecture aims to address these challenges. It enable organizations to extract valuable insights, make informed decisions, and optimize operations in real-time based on the continuous flow of sensor data by addressing these challenges. The purpose of this paper is to investigate and demonstrate the effectiveness of the Micro stream architecture for real-time streaming analytics on sensor data streams. It aims to provide a comprehensive understanding of the Micro services based streaming architecture, including its components and underlying design principles. This section will describe how the architecture employs microservices to manage the complexities of real-time streaming analytics on sensor data.

2. RELATED WORK

Streaming analytics has been the subject of numerous studies and innovations. Several of the most significant contributions to the field of streaming analytics are discussed here. Apache Kafka [1] is a widely adopted distributed streaming platform that employs a publish-subscribe model to manage high-throughput, fault-tolerant, and real-time data streams. It offers data replication, fault tolerance, and scalable message processing, making it a popular option for constructing streaming analytics solutions. Apache Spark Streaming is a framework extension that enables fault-tolerant, high-throughput stream processing. It provides an interface for processing real-time data streams and facilitates integration with diverse data sources and machine learning libraries. Apache Spark Streaming provides window-based computations, micro-batches, and exactly-once processing semantics. Apache Flink [2] is a powerful open-source framework for stream processing that offers low-latency and high-throughput capabilities. It supports multiple data sources and sinks and provides event-time processing and fault tolerance. The dataflow programming model of Apache Flink enables developers to efficiently express complex stream processing workflows. Apache Storm is a distributed real-time computation system designed to process high-velocity, large-scale data streams. It offers fault tolerance, assured message processing, and parallel processing support. The flexible architecture of Apache Storm enables the integration of diverse data sources and real-time analytics on streaming data.

Amazon Kinesis [3] is a fully managed service for ingesting and processing streaming data in real time. It provides capabilities for large-scale data collection, processing, and analysis. Amazon Kinesis supports a variety of data sources, including IoT devices and application logs, and integrates with other AWS services for advanced analytics and visualization. Google Cloud Dataflow [4] is a serverless data processing service that enables the construction of streaming and batch data processing pipelines. It offers a unified programming model for batch and stream processing, as well as autoscaling, fault tolerance, and integration with other Google Cloud services. Google Cloud Dataflow simplifies the creation and deployment of applications for streaming analytics. Microsoft Azure Stream Analytics [5] is a cloud-based streaming analytics service that enables insights and actions on streaming data in real time. It provides a query language similar to SQL for analyzing data streams and supports the integration of multiple data sources and outputs. Microsoft Azure Stream Analytics offers scalability, low latency, and user-friendliness for the development of streaming analytics solutions.

Numerous studies have been conducted to investigate the concepts, advantages, and applications of microservices architecture over the past few years. Here are some important works on microservices architecture: Taibi et al. [6] provided an overview of the research landscape, including microservices' key concepts, advantages, challenges, and architectural principles. Additionally, the study identifies research gaps and offers recommendations for future research in the field. The authors [7] examined the evolution, characteristics, design principles, and adoption challenges of microservices architecture. They also presented industry case studies and identified research challenges in service composition, deployment, and management of microservices. Hasselbring et al. [8] provided a comprehensive analysis of microservices architecture, with a focus on its compatibility with agile practices, DevOps, and organizational culture. In terms of scalability, fault tolerance, team structure, and continuous delivery, the challenges and benefits of adopting microservices are examined. These works contribute to the comprehension, implementation, and adoption of microservices architecture. They offer insight into microservices' fundamental principles, design considerations, and challenges.

Sensor data processing is a prominent research field that focuses on deriving insights and knowledge from sensor-generated data. Numerous studies have been conducted to investigate various sensor data processing techniques and methods. Here are some significant works on sensor data processing: Shi et.al [9] in his work, provided an overview of sensor network data mining techniques. They also discussed the application of various data mining tasks, such as classification, clustering, anomaly detection, and prediction, to sensor data. The paper also examines the obstacles and opportunities associated with sensor data mining. Zhu et al. [10] examined the use of big data analytics in IoT-enabled smart transportation systems that rely heavily on sensor data. It discussed the challenges and various techniques for analyzing sensor data in the context of transportation systems, including data collection, data preprocessing, data analytics, and visualization. Mohammad et.al [11] This review focuses on the application of machine learning techniques to the analysis of sensor data. It discusses the application of machine learning algorithms such as supervised, unsupervised, and reinforcement learning to sensor data for classification, regression, and anomaly detection. Yu et.al [12] examines the techniques for real-time sensor data analytics. It discusses different aspects of real-time analytics, such as data preprocessing, streaming analytics, complex event processing, and visualization. In addition, challenges and future directions in real-time sensor data analytics are presented. Gama, et.al [13] focused on algorithms for efficiently processing sensor data streams. Techniques such as sliding window aggregation, approximate query processing, and adaptive sampling are discussed for managing high-velocity sensor data streams in real time.

These works contribute to the comprehension of techniques, algorithms, and applications for sensor data processing. They offer insight into various techniques for analyzing sensor data, such as data mining, fusion, machine learning, and real-time analytics. Utilizing this knowledge, researchers and practitioners can create effective techniques and systems for processing sensor data, enabling the extraction of valuable insights from sensor-generated data.

3. METHODOLOGY

System Architecture Overview:

Micro stream architecture is a contemporary method for real-time streaming analytics on sensor data streams. It employs microservices principles to manage the complexities of processing and analyzing high-velocity data streams in real-time. The architecture offers a scalable, modular, and efficient framework for ingesting, processing, and storing sensor data, allowing organizations to gain valuable insights and make informed decisions based on real-time analytics.

Microservices are small, independent, and loosely coupled components. Each microservice is dedicated to a particular functionality, such as data ingestion, data processing, analytics and storage. Fig 1 shows the simplified layer representation.

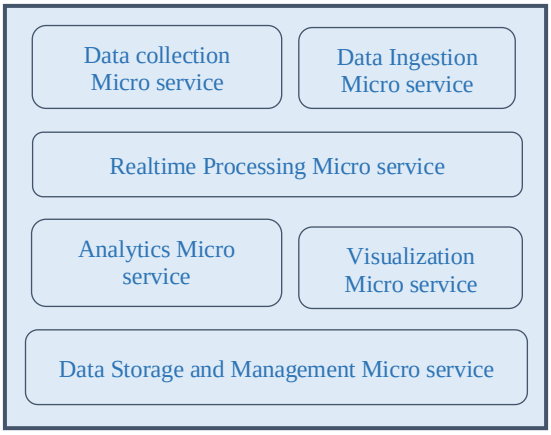

Fig 1. Layered Architecture.

Data Collection layer: Data can be generated from sensor mote using Arduino uno with different sensors and send it to

the raspberry pi gateway to publish the data to the cloud.
Data integration layer: The data integration layer is responsible for ingesting sensor data efficiently in real-time. The data ingestion layer ensures that sensor data streams are seamlessly integrated into the stream processing pipeline.
Stream Processing Layer: The stream processing layer processes and analyses sensor data streams as they arrive. In this stage data are segregated and transformed to extract insightful information from the data.

The analytics and visualization layer applies different deep learning algorithms for performing analytics on the processed sensor data. It uses different data visualization techniques to identify patterns, trends, anomalies, and correlations in the data. The layer provides interactive dashboards and visualizations to effectively present the analyzed data.
Data Storage and Management Layer: This layer is responsible for the storage, organization, and retrieval of sensor data. It offers scalable and dependable storage mechanisms to manage the massive amounts of streaming data to cloud. This layer ensures data accessibility, integrity, and availability for subsequent analysis and retrieval.

Algorithm:
Step1. Initialize serial communication with the specified baud rate.
Step2. Read the sensor values.
Step3. Filter the data to remove noise or outliers and calibrate the sensor readings.
Step4. Format the data into a suitable format for transmission.
Step5. Send the formatted data to the Raspberry Pi via serial communication.
Step6. Receive the data and parse the received data and format the data for ThingSpeak API.
Step7. Send the formatted data to ThingSpeak using the API.
Step8. Start the Kafka server and publish records to Kafka topics.
Step9. Setup the Flink Environment and subscribe to the sensor topics using Kafka Flink consumer.
Step10. Computes average metrics over defined windows.
Step11. Detect anomalies.
Step12. Store the processed data to MongoDB as JSON object for long term analysis.
Step13. Visualize using charts and Graphs.

4. IMPLEMENTATION

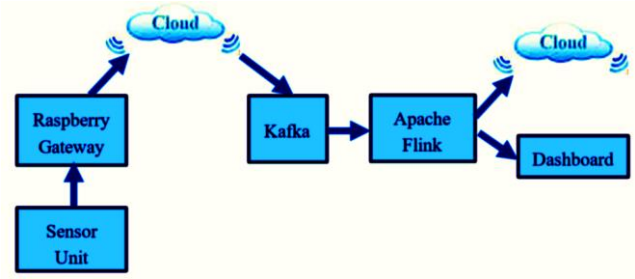

Fig 2: Block Diagram

The data collection layer contains a Microcontroller Unit (MCU) interface with MQ131, MQ135 and DHT11 sensors and conducts telemetry operations. The system employs a Wi-Fi transceiver to transmit the sensed parameters to the cloud using MQTT protocol and data collection microservice. In the data ingestion layer, The MQTT broker seamlessly integrates with Apache Kafka and stream the data to multiple topics in Kafka cluster using the data ingestion microservice. Apache Kafka creates multiple topics with several partitions to produce data through the brokers. The data are streamed with timestamp and sensor id. The real time processing layer has Apache Flink consumer service which subscribe to the selected topics and receive the data stream. For reliability the data is replicated on three data nodes. If one node fails, it redoes the same task in the nearest one. Apache Flink is a streaming platform which supports event-based streaming. It is well connected with microservices architecture. The Analytics layer contains analytics micro service which cleans, transforms and normalize the data. Apache Flink consolidates the data according to the sensor id, timestamp using rolling sum which uses moving average for a specific time window. During this task the redundant data are eliminated and only the required data from different sensors are combined together as a single stream using Apache Flink's aggregate and join operations. The streamed data are stored on the cloud storage as a JSON object which supports heterogeneous data. The real time data is fed to the analytics service which applies a deep learning algorithm to perform the forecasting on the given data according to the user requirement. The data is presented to the user using a dashboard to get key insights on the data to improve business decision making for enterprises. The data stored in the cloud storage can be used as a historical data for batch processing to analyze patterns and trends in the data.

The microservice handles complex event processing, which includes identifying and correlating events across multiple data streams in real-time. It identifies critical events by detecting and responding to complex patterns defined by business rules. The microservice aggregates and summarizes the data to provide higher-level information to facilitate sensor data monitoring, reporting, and visualization. It also provides storage scalability to accommodate the large amount of sensor

data. Data partitioning and sharding optimizes data storage and retrieval. Data are replicated for fault tolerance and high availability.

5. RESULTS AND DISCUSSION

Various pollutants like CO, CO2, NOx, O_3, PM 2.5, PM 10 etc., are predicted for next 1 hour by considering previous 6 lags using moving average and also for next 24 hours based on previous 24 lags using different deep learning models such as CNN, RNN, LSTM, ARLSTM and compared the results. RMSE is taken as a loss function with ADAM optimizer and MAE as evaluation metric to identify the accuracy of different models by comparing it with linear regression model to identify the efficient model. Many pollutants were predicted as part of the case study. Here for sample only Carbon Monoxide is discussed with experimental results outcome.

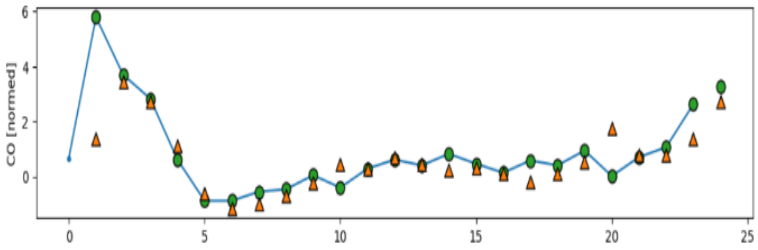

Fig. 3. Single step prediction for pollutant CO in LSTM model

The figure 3. shows every next hour prediction of CO pollutant using LSTM model which performs well compared to the other models. The performance comparison of each model with its MAE factor is given in the figure 4.

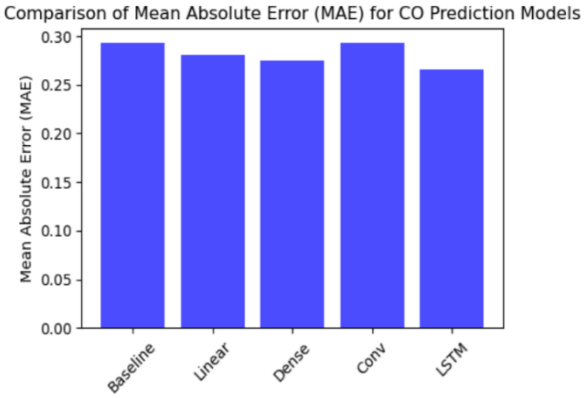

Fig. 4. Performance comparison of MAE for pollutant CO

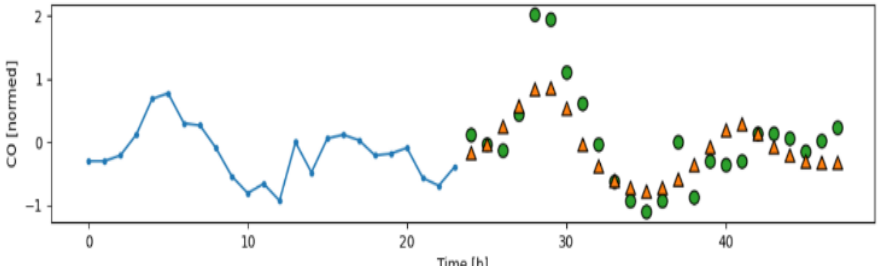

Fig. 5. Multi step prediction for pollutant CO in AR LSTM model

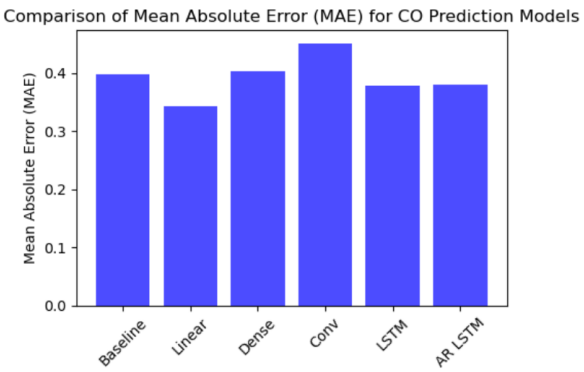

Fig 6. Performance comparison of MAE for pollutant CO

6. CONCLUSION

With the advancement in IoT, sensor devices and big data technologies pave way for researchers to perform real time analytics on sensor data in an efficient manner. Our work suggests a cost-effective mechanism for streaming and storing sensor data on cloud using Apache Kafka and Apache Flink from IoT devices based on microservices model. Since this approach store the data as object complex heterogeneous data can be stored without transforming the data to the common format. The stored data can be used as historical data to prepare the model for analytics. The new approach is tested on the case study of only one city data it can be further extended to more cities to increase the application complexity. The microservices architecture helps both vertical and horizontal scalability, thus we can add more functionalities.

REFERENCES

[1] Shree, R., Choudhury, T., Gupta, S. C., & Kumar, P. (2017, August). KAFKA: The modern platform for data management and analysis in big data domain. In 2017 2nd international conference on telecommunication and networks (TEL-NET) (pp. 1-5). IEEE.

[2] Nazari, Elham, Mohammad Hasan Shahriari, and Hamed Tabesh. "BigData analysis in healthcare: apache hadoop, apache spark and apache flink." Frontiers in Health Informatics 8.1 (2019): 14.

[3] Quadri, N. S., & Yadav, K. (2018, April). Efficient data classification for IoT devices using AWS Kinesis platform. In 2018 21st Saudi Computer Society National Computer Conference (NCC) (pp. 1-5). IEEE.

[4] Krishnan, S. P. T., Gonzalez, J. L. U., Krishnan, S. P. T., & Gonzalez, J. L. U. (2015). Google cloud dataflow. Building Your Next Big Thing with Google Cloud Platform: A Guide for Developers and Enterprise Architects, 255-275.

[5] Barnes, Jeff. "Azure machine learning." Microsoft Azure Essentials. 1st ed, Microsoft 2015.

[6] Taibi, Davide, Valentina Lenarduzzi, and Claus Pahl. "Continuous architecting with microservices and devops: A systematic mapping study." Cloud Computing and Services Science: 8th International Conference, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018, Springer International Publishing, 2019.

[7] Söylemez, M.; Tekinerdogan, B.; Kolukısa Tarhan, A. Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review. Appl. Sci. 2022, 12, 5507. https://doi.org/10.3390/app12115507

[8] W. Hasselbring and G. Steinacker, "Microservice Architectures for Scalability, Agility and Reliability in E-Commerce," 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), Gothenburg, Sweden, 2017, pp. 243-246, doi: 10.1109/ICSAW.2017.11.

[9] Shi, Ke. (2013). Data Mining Techniques for Wireless Sensor Networks: A Survey. International Journal of Distributed Sensor Networks. 2013.

[10] L. Zhu, F. R. Yu, Y. Wang, B. Ning and T. Tang, "Big Data Analytics in Intelligent Transportation Systems: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 1, pp. 383-398, Jan. 2019, doi: 10.1109/TITS.2018.2815678.

[11] Mohammad Saeid Mahdavinejad, Mohammadreza Rezvan, Mohammadamin Barekatain, Peyman Adibi, Payam Barnaghi, Amit P. Sheth, Machine learning for internet of things data analysis: a survey, Digital Communications and Networks,Volume 4, Issue 3,2018, pp 161-175,https://doi.org/10.1016/j.dcan.2017.10.002.

[12] Yu, T., Wang, X. Real-Time Data Analytics in Internet of Things Systems. In: Tian, YC., Levy, D. Handbook of Real-Time Computing. Springer, Singapore. https://doi.org/10.1007/978-981-4585-87-3_38-1

[13] Gama, J., Rodrigues, P.P. Data Stream Processing. In: Gama, J., Gaber, M.M.Learning from Data Streams. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-73679-4