Big Data-driven Architecture for Crowdsensing Systems in Smart Cities

Aleksa Miletić

Faculty of Organizational Sciences
University of Belgrade
Belgrade, Serbia
aleksa@elab.rs
[0000-0001-8940-9897]

Miloš Radenković

School of Computing
Union University
Belgrade, Serbia
mradenkovic@raf.edu.rs
[0000-0002-1708-9799]

Branislav Jovanić

Institute of Physics
University of Belgrade
Belgrade, Serbia
brana@elab.rs
[0000-0003-4130-1638]

Vladimir Vujin

Faculty of Organisational Sciences
University of Belgrade
Belgrade, Serbia
vladimir.vujin@fon.bg.ac.rs

Abstract—The subject of this paper is the development of a crowdsensing system with a big data architecture that aims to efficiently collect, process, and analyze data from various sensors deployed in smart cities. The primary goal of this research is to propose an architecture that enables real-time data collection, processing, and analysis for noise, vibrations, healthcare, and pollution monitoring. The proposed architecture is presented in detail, highlighting its components and their interactions. By leveraging asynchronous event-based communication and integrating Apache Kafka and Apache Spark, the proposed system offers improved decision-making capabilities and resource management for urban sustainability. This research contributes to the field of smart cities and crowdsensing by proposing a big data architecture that enables effective collection of data, processing, and analysis for noise, vibrations, healthcare, and pollution monitoring.

Keywords - Event driven architecture, Apache Kafka, Apache Spark, crowdsensing, smart city

I. INTRODUCTION

Smart cities are rapidly evolving to tackle environmental and health challenges posed by urbanization [1]. Smart cities integrate physical and virtual worlds, providing ordinary objects with intelligence and achieving high levels of integration, coordination, and cooperation to increase quality of life, minimize environmental impact, and optimize resource usage. The effects are more noticeable in urban areas and mega cities, and are achieved by integrating environmental sensing and automatic behaviour to objects to capture and analyse data from the real world for a better virtual operation [2]. In smart cities, crowdsourcing systems generate a vast amount of data that are crucial for providing quality services and improving citizens' lives. However, to make the best use of this data, it is necessary to develop appropriate architectures that can effectively manage large amounts of data and discover insights within them. To achieve this, a combination of Big Data, EDA, and microservice concepts are utilized for efficient data processing, analysis, and visualization. This combination enables fast and accurate execution of complex analyses over vast amounts of data, which provides better understanding of urban processes and improves citizens' lives. In recent years, big data-driven architecture has gained attention as a viable solution for building scalable and robust crowdsensing systems in smart cities. EDA is an architectural pattern where systems are designed to respond to events or messages asynchronously, rather than following a traditional request-response model [3]. This allows decoupling of components, enabling them to work independently and in parallel, resulting in improved scalability, flexibility, and responsiveness.

One of the key components of an big data-driven architecture based crowdsensing system is the data streaming platform, which is responsible for collecting and processing sensor data in real-time. Apache Kafka, a distributed data streaming platform, has gained popularity due to its scalability, fault-tolerance, and high-throughput capabilities, making it suitable for handling large volumes of sensor data in smart cities [4]. The use of Apache Kafka as the data streaming platform in this system enables efficient data collection from diverse devices, such as noise, vibrations, healthcare, and pollution sensors, and ensures reliable and timely delivery of data to Apache Spark for processing, thus contributing to the overall effectiveness and performance of the crowdsensing system [5][6].

The real-time collection and processing of sensor data is critical for addressing environmental and health challenges in urban areas [7]. For example, monitoring noise and vibrations can help in identifying areas with high noise pollution and taking appropriate measures to mitigate the impact on public health. Similarly, monitoring healthcare parameters such as heart rate and blood pressure can enable early detection of health risks and timely interventions [8]. Additionally, monitoring pollution levels can help in identifying pollution hotspots and implementing effective pollution control measures.

The proposed architecture aims to effectively monitor

the environment for air pollution, noise, and vibrations in order to derive insights and conclusions about the impact of these factors on different parts of the city during specific times of the day. This architecture is designed to handle the massive amounts of data generated by crowdsensing applications and leverage it to derive meaningful insights about the parts of the city that may be problematic during specific times of the day due to air pollution, noise, or vibrations. The architecture will focus on developing efficient mechanisms to collect, analyse, and store large amounts of data from crowdsensing applications. Additionally, efforts are made to notify people in a timely and relevant manner about information that is pertinent to their specific needs, such as individuals with health concerns related to air pollution, noise, or vibrations.

The main goal of the proposed architecture is to effectively monitor the environment for air pollution, noise, and vibrations in order to derive insights and conclusions about the impact of these factors on different parts of the city during specific times of the day.

II. RELATED WORK

Data-Driven Knowledge Management Systems [9] focus on the knowledge management process, including knowledge exploration and exploitation, and the capture and organization of explicit and tacit knowledge in organizational memory. Knowledge components which are critical to the workings of such systems, can be stratified into different levels and shared through various knowledge conversion processes.

The proposed architecture is the further refinement of the research done in [10], and is focused on the development of a mobile crowdsensing system for monitoring noise pollution in smart cities. The system is comprised of several elements such as a crowdsensing mobile application, cloud and big data infrastructures, a web application for monitoring and data analysis, and REST web services for communication between components. The system supports three scenarios for mobile device calibration: full calibration using certified sound calibrators in a laboratory, calibration based on the model of the mobile phone using calibration data of other devices of the same model in the database, and no calibration which is the most common approach in crowdsensing contexts but provides lower accuracy of individual measurements. Overall, the developed mobile crowdsensing system offers a comprehensive approach for monitoring noise pollution in smart cities, leveraging mobile devices and cloud-based infrastructure for real-time data collection, analysis, and decision-making, with flexibility in calibration methods based on accuracy requirements and practical considerations.

Apache Kafka is a distributed streaming platform designed to handle high-volume, real-time data streams. It provides a publish-subscribe model where producers write data to topics, and consumers subscribe to those topics to receive and process the data. Kafka is known for its high throughput, fault-tolerance, and scalability, making it a popular choice for building data pipelines, event-driven architectures, and real-time streaming applications [4].

Apache Spark, an open-source distributed data processing engine, is another crucial component that can be integrated with Apache Kafka for processing and analysing sensor data in real-time. Apache Spark provides various powerful features such as batch processing, machine learning, and graph processing, making it suitable for advanced data analytics tasks in smart cities [11]. The use of big data-driven architecture can enable efficient data collection and processing in large-scale smart city environments, where data from a diverse range of sensors need to be integrated and processed in real-time. Big data-driven architecture allows for decoupling of components, making it possible to add or remove sensors without disrupting the entire system [12]. This provides scalability and flexibility, allowing the system to adapt to changing sensor deployments and data processing requirements in a dynamic urban environment.

III. BIG DATA-DRIVEN ARCHITECTURE FOR CROWDSENSING SYSTEMS IN SMART CITIES

In this part of article will be presented architecture for crowdsensing systems in smart cities designed to efficiently collect, process, and analyse diverse data from multiple sources in real-time. The architecture consists of five layers: data sources, data streaming, data structure, data analytics, and monitoring processes.

On the Fig. 1, first layer includes IoT sensors and mobile phone sensors, existing databases, and files. Data collection works similarly to edge computing and can perform some data modification and analysis on the device itself before sending it to the database. Mobile phone sensors can include a wide range of sensors, such as GPS, accelerometer, gyroscope, magnetometer, ambient light sensor, proximity sensor, and microphone, among others. These sensors can provide valuable data on location, motion, orientation, light levels, proximity to objects, and ambient sound, among others.

IoT sensors, on the other hand, can be deployed throughout the city and can be both static and mobile in nature. Static IoT sensors can be installed at fixed locations, such as on street lamps, buildings, or other infrastructure, and can include sensors for monitoring environmental parameters such as temperature, humidity, air quality, noise levels, and pollution levels. Mobile IoT sensors, on the other hand, can be attached to moving vehicles, drones, or other mobile devices, and can provide real-time data on road conditions, weather conditions, and other dynamic parameters. Additionally, data can also be collected from mobile phones using specialized applications. In some cases, citizens can install Raspberry Pi stations in their homes to collect data, while in other cases, professional stations can be installed by citizens or companies to collect data in more specific locations such as yards or parking lots.

In addition to mobile phone and IoT sensors, the data sources layer can also include data from existing databases and files. These can be data from municipal databases, open data sources, social media, or other relevant sources. Such data can provide additional context and metadata for the crowdsensing data, such as location information, historical data, or other relevant information that can enrich the overall understanding of the urban environment.

Overall, the data sources layer of the proposed crowdsensing architecture encompasses a wide range of sensors and data sources, including mobile phone sensors, IoT sensors (both static and mobile), and existing databases and files, all of which contribute to the collection of diverse and rich crowdsensing data for analysis and insights in smart city applications.

The second layer, data streaming, plays a crucial role in transmitting the crowdsensing data to a private cloud for further processing and analysis. This layer utilizes the MQTT (Message Queuing Telemetry Transport) protocol, a lightweight messaging protocol designed specifically for efficient and reliable communication between devices in IoT applications [13]. MQTT ensures that the data is transmitted in real-time, allowing for timely updates and insights. Likewise, it provides secure and efficient communication, making it suitable for handling the large volumes of crowdsensing data generated from IoT sensors.

Real-time data processing in Apache Kafka is performed using Kafka Streams, a powerful and lightweight stream processing library that is part of the Kafka ecosystem. This allows for processing and decoupling of sensitive user/mobile data and sensor values, ensuring that they are separated and processed independently to maintain data privacy and security.

To achieve this, data from different sources, such as user/mobile data and sensor values, are ingested into separate Kafka topics using Kafka producers. Kafka topics act as channels that hold the data streams in a distributed and scalable manner, allowing for parallel processing. Kafka Streams then consumes these topics and performs real-time data processing operations, such as filtering, aggregation, transformation, and enrichment, using a stream processing topology defined by the application logic.

Data processing layer, after data is processed in Apache Kafka using Kafka Streams, it will be sent to Apache Spark for further analysis in real-time or batch mode, depending on data structure. Apache Spark is a popular distributed data processing framework that provides advanced analytics capabilities for large-scale data processing [14]. The processed data from Apache Kafka will be transformed and organized into a unified and structured format using data structuring techniques.

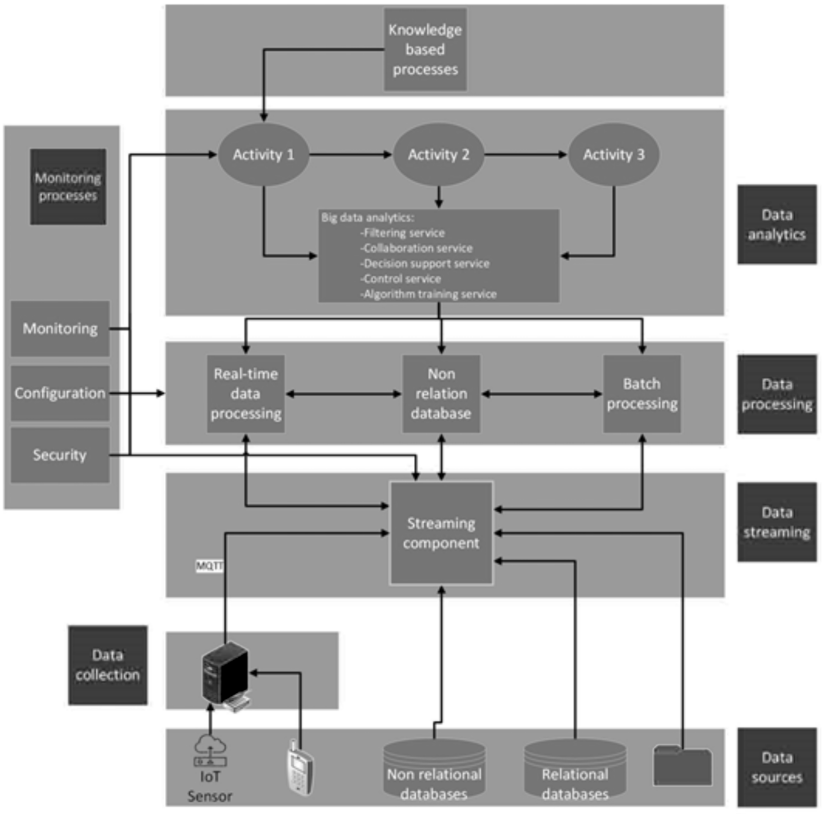

Fig. 1 - Event-driven architecture for crowdsensing systems in smart cities

In the proposed architecture, Apache Spark can consume the processed data from Kafka topics and perform various data analytics operations, such as data aggregation, machine learning, and graph processing, to gain insights and extract valuable information from the data. This enables efficient data analysis in Apache Spark and ensures that the data is stored in a suitable database for further retrieval and analysis.

The fourth layer of the proposed architecture involves interactive services that allow users to further interact with the processed data and take appropriate actions. These services include:

1. The Filtering service enables users to apply filters and settings to the data for tailored analysis and decision-making. Filters can be set by location or time.
2. The Collaboration service promotes real-time collaboration among users, facilitating information sharing and feedback exchange.
3. The Decision support service provides tools for data visualization, analytics, and predictive modeling to support evidence-based decision-making.
4. The Control service allows remote management and control of devices or systems based on processed data, enhancing system responsiveness.
5. The Algorithm training service supports continuous training and refinement of algorithms for improved system performance.

These interactive services empower users to further interact with the processed data, collaborate, make informed decisions, take actions, and continuously improve the system's performance, leading to a more effective and user-centric solution for monitoring environmental pollution and healthcare in smart cities.

The final layer, monitoring processes, involves monitoring and tracking the performance and effectiveness of the entire architecture. For that will be utlized tools such as Prometheus and Grafana. Also thay will ensure the timely and accurate processing of crowdsensing data. Configuration involves managing the various components of the architecture, such as sensors and databases, to ensure they are working together smoothly and efficiently. Security involves implementing measures to protect the data being collected and processed, as well as ensuring the overall security of the architecture itself.

IV. IMPLEMENTATION REMARKS

In this part will be presented technologies and why are they chosen for implementation of the proposed architecture.

Kafka is an asynchronous messaging system that provides a messaging system with a broker for broadcasting messages and storing them for as long as needed, making it suitable for streaming and fire-and-forget messaging [3]. Unlike traditional messaging systems, Kafka is distributed and provides high availability, storage, and linear scale-out across a cluster. Some people compare Kafka to a database due to its storage capabilities, support for large data volumes, SQL interface for querying data, and transaction support. Kafka's Connect interface allows for data integration with various interfaces and datastores, and its streaming APIs enable data manipulation in-flight [3].

Apache Kafka is capable of accepting data from various sources through Kafka Connect, which allows data to be pulled from IoT sensors through the MQTT protocol, mobile applications, as well as databases and files.

Apache Kafka and Apache Spark are an ideal combination for processing and analysing large-scale data in real-time.[15] Spark, with its fast and general-purpose cluster computing framework, can process data in parallel across a cluster of machines, enabling scalable and high-performance data processing. The integration between Kafka and Spark allows for easy data ingestion and real-time data processing in Spark Streaming applications [15]–[17]. Kafka's ability to store and retain data for a configurable period of time enables Spark Streaming to perform analytics on both historical and real-time data. This makes the combination suitable for a wide range of use cases, from real-time data analytics to processing of large-scale data streams.

Grafana and Prometheus are likewise a powerful combination for monitoring distributed systems. Prometheus provides reliable and scalable monitoring and alerting capabilities, while Grafana offers rich visualization and dashboarding features [18], [19]. Together, they enable users to gain insights into the performance and health of their systems through customizable visualizations and interactive dashboards.

In the proposed architecture, the Kafka cluster will be deployed on a private cloud infrastructure and managed using Kubernetes, a popular container orchestration platform. If a Kafka broker pod fails, Kubernetes will automatically detect the failure and initiate a rescheduling process to restart the failed pod on a healthy node.

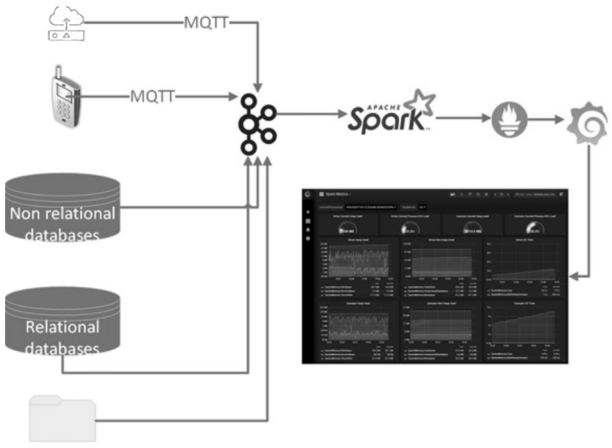

Fig. 2 Event-driven architecture for crowdsensing systems in smart cities

V. CONCLUSION

The proposed architecture, implemented in smart cities with a mobile app for monitoring noise, vibration, air pollution, and healthcare, offers several benefits. The most notable benefit lies in its enables real-time data processing, which allows citizens to receive timely and accurate information about pollution levels in their vicinity. This empowers citizens to make informed decisions regarding their health and well-being. The system also offers the ability to decouple sensitive data, ensuring the privacy and security of user information. Additionally, the use of Prometheus and Grafana for monitoring provides real-time insights into the performance and effectiveness of the system, enabling proactive management and troubleshooting. Furthermore, deploying the system on a private cloud with Kubernetes ensures high availability and scalability, making it suitable for large-scale deployments in smart cities. Overall, this system offers a valuable tool for citizens to monitor environmental pollution and healthcare in their localities, leading to improved quality of life and informed decision-making.

Lastly, the integration of Apache Kafka and Apache Spark as data streaming and processing platforms, respectively, can provide powerful capabilities for real-time data analytics in smart cities. Apache Kafka can efficiently collect and stream data from sensors to Apache Spark, where data processing tasks such as filtering, aggregation, and analysis can be performed on the collected data, enabling real-time insights and decision-making in smart cities. The combination of Apache Kafka's high-throughput capabilities for data collection and Apache Spark's robust data processing capabilities can contribute to the effectiveness and performance of crowdsensing systems that deal with diverse data types.

REFERENCES

[1] M. P. Rodríguez-Bolívar, “Transforming city governments for successful smart cities,” Transform. City Gov. Success. Smart Cities, pp. 1–185, Jan. 2015, doi: 10.1007/978-3-319-03167-5/COVER.

[2] O. Alvear, C. T. Calafate, J. C. Cano, and P. Manzoni, “Crowdsensing in Smart Cities: Overview, Platforms, and Environment Sensing Issues,” Sensors 2018, Vol. 18, Page 460, vol. 18, no. 2, p. 460, Feb. 2018, doi: 10.3390/S18020460.

[3] B. Stopford and S. Newman, “Concepts and Patterns for Streaming Services with Apache Kafka Designing Event-Driven Systems,” 2018, Accessed: Apr. 23, 2023. [Online]. Available: http://oreilly.com/safari

[4] A. Zelenin and A. Kropp, “Apache Kafka,” in Apache Kafka, 2021, pp. I–XVII. doi: 10.3139/9783446470460.fm.

[5] A. Miletić, P. Lukovac, B. Radenković, and B. Jovanić, “Designing a data streaming infrastructure for a smart city crowdsensing platform,” E-bus. Technol. Conf. Proc., vol. 2, no. 1, pp. 61–64, Jun. 2022, Accessed: Nov. 19, 2022. [Online]. Available: https://ebt.rs/journals/index.php/conf-proc/article/view/128

[6] A. Miletić, M. Despotović-Zrakić, Z. Bogdanović, M. Radenković, and T. Naumović, “WorldCIST’23 - OpenConf Abstract Submission, Peer Review, and Event Management System.” http://itmasoc.org/wcist23/modules/request.php?module=oc_program&action=summary.php&id=240 (accessed May 10, 2023).

[7] M. G. Alvarez, J. Morales, and M. J. Kraak, “Integration and Exploitation of Sensor Data in Smart Cities through Event-Driven Applications,” Sensors 2019, Vol. 19, Page 1372, vol. 19, no. 6, p. 1372, Mar. 2019, doi: 10.3390/S19061372.

[8] E. Roth, “The Internet of Medical Things: What is it and What is its Role in Healthcare.” https://www.productiveedge.com/blog/the-internet-of-medical-things-what-is-it-and-what-is-its-role-in-healthcare (accessed May 03, 2023).

[9] T. Le Dinh, T.-C. Phan, and T. H. Bui, “Towards an Architecture for Big Data-Driven Knowledge Management Systems”, Accessed: Apr. 12, 2023. [Online]. Available: https://www.researchgate.net/publication/314047241

[10] I. Jezdović, S. Popović, M. Radenković, A. Labus, and Z. Bogdanović, “A crowdsensing platform for real-time monitoring and analysis of noise pollution in smart cities,” Sustain. Comput. Informatics Syst., vol. 31, Sep. 2021, doi: 10.1016/J.SUSCOM.2021.100588.

[11] A. Bansal, R. Jain, and K. Modi, “Big Data Streaming with Spark,” Stud. Big Data, vol. 43, pp. 23–50, 2019, doi: 10.1007/978-981-13-0550-4_2/COVER.

[12] F. Ten Sythoff, “The Advantages of Event-Driven Architecture (EDA),” May 11, 2022. https://www.greenbird.com/news/event-driven-architecture-eda (accessed Apr. 23, 2023).

[13] D. Soni and A. Makwana, “A survey on MQTT: a protocol of internet of things (IOT) Prediction of Telemetry data using Machine Learning Techniques View project Analysis and Survey on String Matching Algorithms for Ontology Matching View project a survey on MQTT: a protocol of inter,” 2017, Accessed: Apr. 23, 2023. [Online]. Available: https://www.researchgate.net/publication/316018571

[14] I. Pointer, “What is Apache Spark? The big data platform that crushed Hadoop | InfoWorld.” https://www.infoworld.com/article/3236869/what-is-apache-spark-the-big-data-platform-that-crushed-hadoop.html (accessed Sep. 17, 2022).

[15] G. M. D’Silva, A. Khan, Gaurav, and S. Bari, “Real-time processing of IoT events with historic data using Apache Kafka and Apache Spark with dashing framework,” RTEICT 2017 - 2nd IEEE Int. Conf. Recent Trends Electron. Inf. Commun. Technol. Proc., vol. 2018-January, pp. 1804–1809, Jul. 2017, doi: 10.1109/RTEICT.2017.8256910.

[16] L. Hall, “How to process streams of data with Apache Kafka and Spark.” https://cloudblogs.microsoft.com/opensource/2018/07/09/how-to-data-processing-apache-kafka-spark/ (accessed Apr. 24, 2023).

[17] “Apache Kafka - Integration With Spark.” https://www.tutorialspoint.com/apache_kafka/apache_kafka_integration_spark.htm (accessed Apr. 24, 2023).

[18] M. Chakraborty and A. P. Kundan, “Grafana,” Monit. Cloud-Native Appl., pp. 187–240, 2021, doi: 10.1007/978-1-4842-6888-9_6.

[19] “Get started with Grafana and Prometheus | Grafana documentation.” https://grafana.com/docs/grafana/latest/getting-started/get-started-grafana-prometheus/ (accessed Apr. 24, 2023).