**Apache Kafka For Real Time Analysis of Wikimedia Data**

|  |  |
| --- | --- |
| **Sri Divya Meduri** smeduri@memphis.edu  University Of Memphis | **Vedvikas Reddy Magam** vmagam@memphis.edu  University Of Memphis |

# **ABSTRACT**

The rapid increase in real-time data on collaborative online platforms requires sophisticated and effective methods for managing and analyzing data. This project showcases the use of Apache Kafka, a distributed streaming platform, with OpenSearch to effectively handle and analyze the extensive real-time data produced by Wikimedia sites, such as Wikipedia. Apache Kafka enables the effortless recording and streaming of data modifications, while OpenSearch organizes this data for improved analytical processing. Utilizing Kafka Producer and Consumer guarantees continuous data transmission and rapid accessibility for analysis. This integration enhances the efficiency of managing real-time data and also enhances accessibility and analytical capabilities, hence facilitating a better understanding of collaborative content dynamics. The result is a strong framework that enables users to do advanced analysis and acquire a thorough comprehension of real-time Wikimedia data.

# **1 INTRODUCTION**

The world of electronic media is constantly evolving, especially on collaborative platforms such as Wikipedia. A wide range of users work together to create and update content, contributing to its ongoing evolution. The constant interaction between users produces a large amount of data, capturing a variety of information including content updates and interaction between users. This data presents immense possibilities for obtaining valuable insights and driving innovation in data management and analytics. However, these opportunities present significant challenges, as the vast amount and speed of data require efficient methods to effectively capture, store, and process information while maintaining its basic value.

Our project is motivated by the necessity to successfully utilize Wikimedia’s vast real-time data, in response to these difficulties. The main objective is to utilize this data in order to greatly improve user experiences on the platform by making information more easily available and analytically valuable. Enhancing data accessibility allows users and researchers to locate and utilize information, hence enhancing the entire interaction with quickly and dependably.

Moreover, by improving our analytical capabilities, we may reveal more profound understandings of user behavior and content patterns. This knowledge can then be used to make future improvements and guarantee that the platform develops according to the user requirements and expectations. Meeting these objectives is crucial not just for keeping up with fast technical progress but also for satisfying the growing requirements for timely and data driven insights in today’s digital-first environments.

## **1.1 Objective**

The primary objective of this project is to develop a robust framework using Apache Kafka, a renowned distributed streaming platform, to manage the high volume and velocity of Wikimedia data effectively. By integrating this data with OpenSearch, we aim to facilitate advanced analytics and search functionalities, making the data more accessible and actionable. Specific goals include:

1. Creating a dependable system for ongoing data collection and analysis by utilizing Kafka Producer and Consumer modules.
2. Improving the OpenSearch ecosystem by incorporating organized and significant Wikimedia data to enhance analytical processing.
3. Creating an adaptable solution capable of managing future growth in data volume and complexity.

## **1.2 Challenges to be Addressed**

1. Data Volume and Velocity: Effectively handling the vast amount of data produced in real-time and ensuring rapid processing while maintaining data integrity.
2. System Configuration and Implementation: Configuring and managing Apache Kafka to optimize both fault tolerance and data velocity.
3. Data Quality and Security: Ensuring the data’s integrity throughout transfer and storage and establishing strong security measures to protect the data.

# **2 RELATED WORK**

Managing and analyzing streaming data can be hard, but there are a number of well-known options that can help with real-time data streaming and processing. Some of these options are:

1. Kafka Connect SSE Source Connector: This application collects Server-Sent Events (SSE) from many sources, including Wikimedia, and inserts them into the Apache Kafka ecosystem. It serves as an interface that simplifies the introduction of streaming data into Kafka, allowing for subsequent processing and analysis.
2. Kafka Streams Counter Application: This application is frequently used in the Kafka Streams framework for the purpose of processing real-time data. The software conducts statistical computations on live data, providing strong capabilities for doing calculations quickly on the data stream. This is essential for real-time analytics.
3. Kafka Streams Counter Application: This application is frequently used in the Kafka Streams framework for the purpose of processing real-time data. The software conducts statistical computations on live data, providing strong capabilities for doing calculations quickly on the data stream. This is essential for real-time analytics.

Our project, which focuses on analyzing real-time data from Wikimedia using Apache Kafka, surpasses and develops upon these current frameworks by integrating multiple innovative methods:

Comprehensive Data Handling: Our project offers a more com-

prehensive approach to managing high-volume Wikimedia data sources compared to the Kafka Connect SSE Source Connector, which only focuses on SSE extraction. Our data pipeline is designed to effectively acquire, process, and prepare data from Wikimedia for advanced analysis, considering the intricate characteristics of real-time data.

Integration with OpenSearch: By utilizing the concepts of the Kafka Connect Elasticsearch Sink, our solution improves data integration by enabling the seamless transfer of processed Wikimedia data to OpenSearch. This encompasses not only the transfer of data, but also the enhancement of its efficiency enabling advanced data analysis and search functionalities under OpenSearch.

Complete Data Pipeline Processing: Our approach involves the full data pipeline, including efficient data streaming and interaction with Elasticsearch/OpenSearch. We prioritize the monitoring, measurement, and effective error management in the stream processing pipeline to guarantee the accuracy of data and the dependability of operations.

Monitoring and Error Handling:

Our project incorporates advanced monitoring and error control mechanisms that exceed the usual functionality found in conventional Kafka implementations. This guarantees optimal system performance and dependability, while maintaining the integrity and consistency of the processed data.

Customized solutions for managing data streams from Wiki-

media: Our project indicate highlights the effective integration of Kafka Connect, Kafka Streams, and OpenSearch to address the specific challenges presented by Wikimedia’s real-time data streams. This customized approach optimizes the importance and value obtained from Wikimedia data, enabling more informed decision making and improving user experiences on the platform. Our project offers a comprehensive customized solution to address the specific difficulties of handling Wikimedia’s real-time data, going beyond the capabilities of current solutions in terms of integration, reliability, and flexibility. This comprehensive method not only improves the accessibility and analytical capabilities of data, but also guarantees the ability to adjust and expand to meet the needs of changing digital environments.

# **3 SOLUTION PATH**

Overview

The project aims to create a thorough solution for the fast processing and analysis of Wikimedia data by utilizing Apache Kafka and OpenSearch. The solution design is divided into separate modules, with each module responsible for various components of the data flow, ranging from data ingestion to indexing. The utilization of a modular approach improves the ability to maintain, scale, and comprehend the code, hence permitting more efficient management and comprehension.

## **3.1 Producer Module**

Functionality: The Producer Module begins the data flow by establishing a connection to the Wikimedia real-time stream and generating events. Implementation

Producer Setup: Set up Kafka producer properties, such as bootstrap servers for connectivity, serializers for data format consistency, and compression settings to improve network and storage efficiency.

Topic Creation: Created a Kafka topic called "topic.wikimedia" specifically designed for managing a continuous stream of data from Wikimedia.

Event Handling: Created a class called ’producerhandler’ to efficiently handle different types of events from the Wikimedia stream.

Stream Connection: Established a connection to the Wikimedia event stream using an Event Source and activated the producer to start data ingestion.

Data Streaming: Incorporated a pause in the program to ensure stable and continuous data streaming, accommodating network and processing latencies.

## **3.2 Producer Handler Module**

Functionality: The system manages and executes actions related to events received from the Wikimedia stream, transmitting them to the specified Kafka topic.

Event Handler Interface: Implements the methods that were specified in the Event Handler interface to manage several categories of events, including open, close, message, comment, and error.

Data Extraction and Transmission: Extracted relevant data from message events and transmitted it to the "topic.wikimedia" Kafka topic using the configured Kafka producer.

## **3.3 Consumer Module**

Functionality: Consumes messages from the Kafka topic and indexes them into OpenSearch for further analysis. Consumer Configuration: Configure Kafka consumer properties, such as the bootstrap servers, deserializers for data interpretation, and a group ID for managing the consumer.

Topic Subscription: Subscribed to the "topic.wikimedia" to receive data.

Index Creation in OpenSearch: Used a CreateIndexRequest to create an index in OpenSearch if it is not already present.

Continuous Polling: Implemented an infinite loop to continuously poll for new records from the Kafka topic.

Data Indexing: Processed the incoming records by extracting the message values and storing them in OpenSearch. Logged the response for the purpose of monitoring and debugging.

## **3.4 OpenSearch Client Module**

Functionality: Manages connectivity and operations related to the OpenSearch cluster.

Cluster Connectivity: Extracted the host, port, and credentials from the OpenSearch cluster URL.

Authentication Setup: Implemented authentication by utilizing the BasicCredentialsProvider and established the RestHighLevelClient with the provided credentials.

Client Configuration: Created and returned a fully configured RestHighLevelClient for interfacing with OpenSearch.

## **3.5 Rationale Behind Solution Design**

The design of this solution is aimed at managing the complete data processing pipeline effectively:

Modular Design: Every module has particular functions, which makes the system architecture simpler and improves the maintainability of the code.

Efficiency and Throughput: The configurations of the Kafka producer and consumer are adjusted to provide the highest possible data throughput and processing efficiency.

Comprehensive Data Management: The solution includes the entire process of processing data, starting from streaming data ingestion, and extending to indexing in OpenSearch.

Robustness and Monitoring: Ensuring the robustness and quality of the data streaming and processing pipeline is essential for managing the real-time data created by Wikimedia. This involves placing a primary focus on error handling, monitoring, and analytics.

# **4 EVALUATION**

# **4.1 Benchmarks Used**

**4.1.1 Kafka.**

1. Throughput: Messages per second.
2. Latency: Time taken for a message to travel from Wikipedia changes to Kafka topic.
3. Resource Utilization: CPU and memory usage.
4. Efficiency: Increased batch size and compressed the incoming data from Wikimedia through producer

**4.1.2 OpenSearch.**

1. Indexing Rate: Rate at which documents from Kafka topic areindexed.
2. Creation of OpenSearch cluster
3. Resource Utilization: CPU, memory, network usage.

## **4.2 Hardware/Software Configuration**

1. First, we installed Apache Kafka on our Mac using "home brew". Then, we set the path variable to the necessary files.
2. After that, we proceeded to initiate the Kafka Server. ImplementedKraft mode. (Note: It is advisable to start before running the producer and consumer classes.
3. Our code has been implemented using the IntelliJ platform.

**4.2.1 Producer.**

1. Added Kafka clients 3.1.0 version as a dependency.
2. Added okhttp3 and okhttp-eventsource as dependencies for getting and handling the streaming data.
3. Used the compressor type “snappy” for better efficiency.

**4.2.2 Consumer.**

1. Added opensearch-client 1.2.4 version dependency as It provides a simple API for different logging frameworks.
2. Added opensearch-rest-high-level-client dependency as it provides a more convenient and idiomatic way of interacting with an OpenSearch cluster compared to the low-level client.
3. Added google gson dependency as it provides an easy way to convert Java objects to JSON and vice versa.

## **4.3 Comparison with Baselines**

1. One comparison would be with Apache Kafka Producer result when dependency is not set to latest version and after setting it.

* Old Version

A screenshot of a computer

Description automatically generated

* New Version

A screenshot of a computer

Description automatically generated

From the above results we could see acks=1 and acks= -1, batch size =16Mb and batch size = 32Mb, max in flight per connection = none and 5 respectively. These are the some of the differences with old and latest version of Kafka. Latest version will automatically ensure there are enough replicas, high throughput, fault tolerant and so on. Even without latest version and some dependencies, we got the result, but we tried to achieve efficiency so that there won’t be any discrepancies.

1. The other comparison could be with Kafka Topic name.

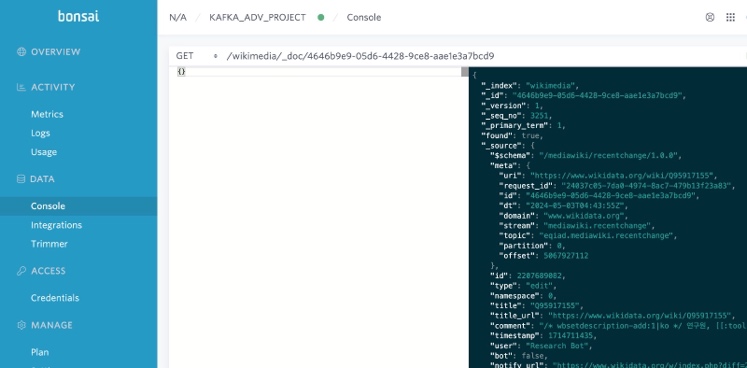
The name of the Kafka topic holds significant importance, as it is crucial to ensure that we produce messages using the correct topic name and subsequently consume messages from the same topic name.

It has been observed that an error occurs when there is a discrepancy in the topic names used in the producer and consumer files. However, in cases where there is no discrepancy, the code executes without any issues, and the data is effectively transmitted to the OpenSearch platform.

**4.4 Real Time Data Management and monitoring with Bonsai.io**

**4.4.1 Data Querying**

By utilizing Bonsai.io, we can perform accurate queries on our OpenSearch database, as demonstrated by fetching particular documents associated with Wikimedia events. This feature guarantees rapid retrieval of indexed data and enables accurate real-time analysis.



**4.4.2 Monitoring of Request Logs**

The following view displays the "Recent Requests" that have been handled by the system. It includes information such as the request type (PUT), the document ID, and the response time. Every request is recorded with its completion status, denoted by the HTTP status code 201, which signifies successful data indexing. The system's efficiency in handling data transactions is illustrated by its short reaction times, which are approximately 5 ms.

A screenshot of a computer

Description automatically generated

**4.4.3 Performance Metrics**

Bonsai.io provides necessary measures, such the number of requests and the duration, which provide valuable insights into the performance of our data processing system. These metrics aid in evaluating the effectiveness of a system and provide insight for making improvements to enhance its responsiveness and stability.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# **5 CONCLUSION**

During this project, we effectively developed and executed an advanced solution for handling and analyzing real-time data from Wikimedia. We utilized Apache Kafka and OpenSearch and integrated them through the Bonsai.io platform. This approach sought to utilize the large amounts of data produced on collaborative platforms such as Wikipedia, with the goal of greatly improving user experience by expanding data accessibility and providing advanced analytical capabilities. The architectural solution we developed included a thorough data processing pipeline that incorporated specialized modules for gathering, analyzing, and indexing real-time events. Each module was specifically designed to effectively manage the large volume and rapid flow of Wikimedia data.

The integration of Bonsai.io was crucial, as it provided robust tools for data management, monitoring of performance, and operational analysis, guaranteeing optimal data processing speed and system efficiency. This configuration not only achieved our project goals but also showcased the revolutionary capacity of modern data streaming technology. Our initiative enhanced decision-making processes and user engagement on digital platforms by analyzing massive amounts of real-time data. This has established solid foundations for future expansions and applications in various real-world scenarios.

# **6 REFERENCES**

[1] For installing Kafka on mac: <https://www.conduktor.io/kafka/how-to-install-apache-kafka-on-mac-with-homebrew/>

[2] For getting Kraft mode: <https://www.conduktor.io/kafka/how-to-install-apache-kafka-on-linux-without-zookeeper-kraft-mode/>

[3] For Kafka download: <https://kafka.apache.org/downloads>

[4] Jdk 11 download: <https://docs.aws.amazon.com/corretto/latest/corretto-11-ug/downloads-list.html>.

[5] To login to Bonsai.io: https://app.bonsai.io/login