Cluster Computing

Course- INT 315

Twitter Stream Analysis (Using Kafka)

Submitted To : Dr. Amrit Pal Singh

Submitted By:

Name: Fanish Pandey

Reg No.- 12102636

Section: K21FZ

Roll No. 24

Acknowledgement

I would like to express my heartfelt gratitude to my respected mentor, **Dr. AmritPal Singh**, for his continuous support, guidance, and encouragement throughout the completion of this project titled **"Real-Time Twitter Hashtag Monitoring using Apache Kafka"**. His deep insights into distributed systems and expert teaching in the course **Cluster Computing (INT 315)** greatly contributed to my understanding and execution of this project.

The objective of this project was to develop a real-time data streaming pipeline that captures Twitter hashtag traffic and counts the frequency of hashtags using **Apache Kafka**. A Kafka **Producer** simulates hashtag traffic using custom-generated Twitter-like data, while a Kafka **Consumer** processes and aggregates hashtag counts in real time. The processed data is then visualized using **JavaFreeChart** to showcase dynamic trends and spikes in hashtag usage.

This hands-on project significantly enhanced my knowledge of big data streaming, message brokers, and real-time data analytics. It also helped me develop a deeper appreciation for the scalability and efficiency of distributed data processing frameworks.

I am truly thankful for the opportunity to work on such a meaningful and technically enriching project under the expert mentorship of Dr. AmritPal Singh.

Introduction

In the era of social media, platforms like Twitter generate massive amounts of real-time data every second. Among this data, **hashtags** play a vital role in identifying trending topics and gauging public interest. Monitoring and analyzing hashtags in real time can offer valuable insights for businesses, media, researchers, and public sentiment analysis.

This project, **"Real-Time Twitter Hashtag Monitoring using Apache Kafka,"** focuses on building a data streaming pipeline that captures, processes, and visualizes hashtag activity. By simulating Twitter data using custom inputs, a **Kafka Producer** sends streaming data to a **Kafka Topic**, which is then consumed by a **Kafka Consumer** for processing and counting hashtag frequencies. The output is dynamically visualized using **JFreeChart**, allowing users to observe live trends and popular tags.

The project not only demonstrates the power of real-time stream processing but also provides practical experience in working with **Apache Kafka**, one of the most widely-used distributed event streaming platforms in the industry.

Objectives

1. **Simulate real-time Twitter hashtag data** using a Kafka Producer.
2. **Stream the simulated data** using Apache Kafka topics.
3. **Consume and process the data** in real time using a Kafka Consumer.
4. **Count the frequency of each hashtag** as it appears in the stream.
5. **Visualize the live results** using JFreeChart in Java to reflect the trending hashtags dynamically.
6. Gain hands-on experience in **stream processing**, **distributed messaging**, and **real-time analytics**.

Tools & Technologies Used

Tool / Technology	**Purpose**
Apache Kafka	Real-time data streaming and messaging
Java	Backend development for Kafka Producer and Consumer
JFreeChart	Visualization of real-time hashtag trends
Nmap (simulated)	Used to mimic traffic injection with embedded hashtags
IntelliJ IDEA / Eclipse	Development environment
Kafka Topics & Zookeeper	Data coordination and message brokering

<u>CODE for PRODUCER (in ECLIPSE IDE) :</u>

```
import org.apache.kafka.clients.producer.*;


import java.io.BufferedReader;

import java.io.FileReader;

import java.util.Properties;

import java.util.regex.Matcher;

import java.util.regex.Pattern;


public class Producer {

  private static final String TOPIC = "hashtags_topic";


  public static void main(String[] args) {

    String csvFile = "C:/Users/DELL/Downloads/sample.csv"; // Adjust path
if needed


    // Kafka Producer configuration

    Properties props = new Properties();

    props.put("bootstrap.servers", "localhost:9092");

    props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");

    props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");


    KafkaProducer<String, String> producer = new
KafkaProducer<>(props);
```

```
      try (BufferedReader br = new BufferedReader(new FileReader(csvFile)))
{
        String line;
        br.readLine(); // Skip header

        while ((line = br.readLine()) != null) {
          // CSV is comma-separated. Handle commas inside quotes if
needed.
          String[] values = line.split(",(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)");

          if (values.length < 4) continue; // Skip malformed lines

          String tweetContent = values[3];

          // Extract hashtags using regex
          Pattern pattern = Pattern.compile("#\\w+");
          Matcher matcher = pattern.matcher(tweetContent);

          while (matcher.find()) {
            String hashtag = matcher.group();
            ProducerRecord<String, String> record = new
ProducerRecord<>(TOPIC, hashtag);
            producer.send(record);
            System.out.println("Sent hashtag: " + hashtag);
          }

          Thread.sleep(100); // Simulate real-time streaming
        }
      } catch (Exception e) {
        e.printStackTrace();
      } finally {
        producer.close();
      }
  }
}
```

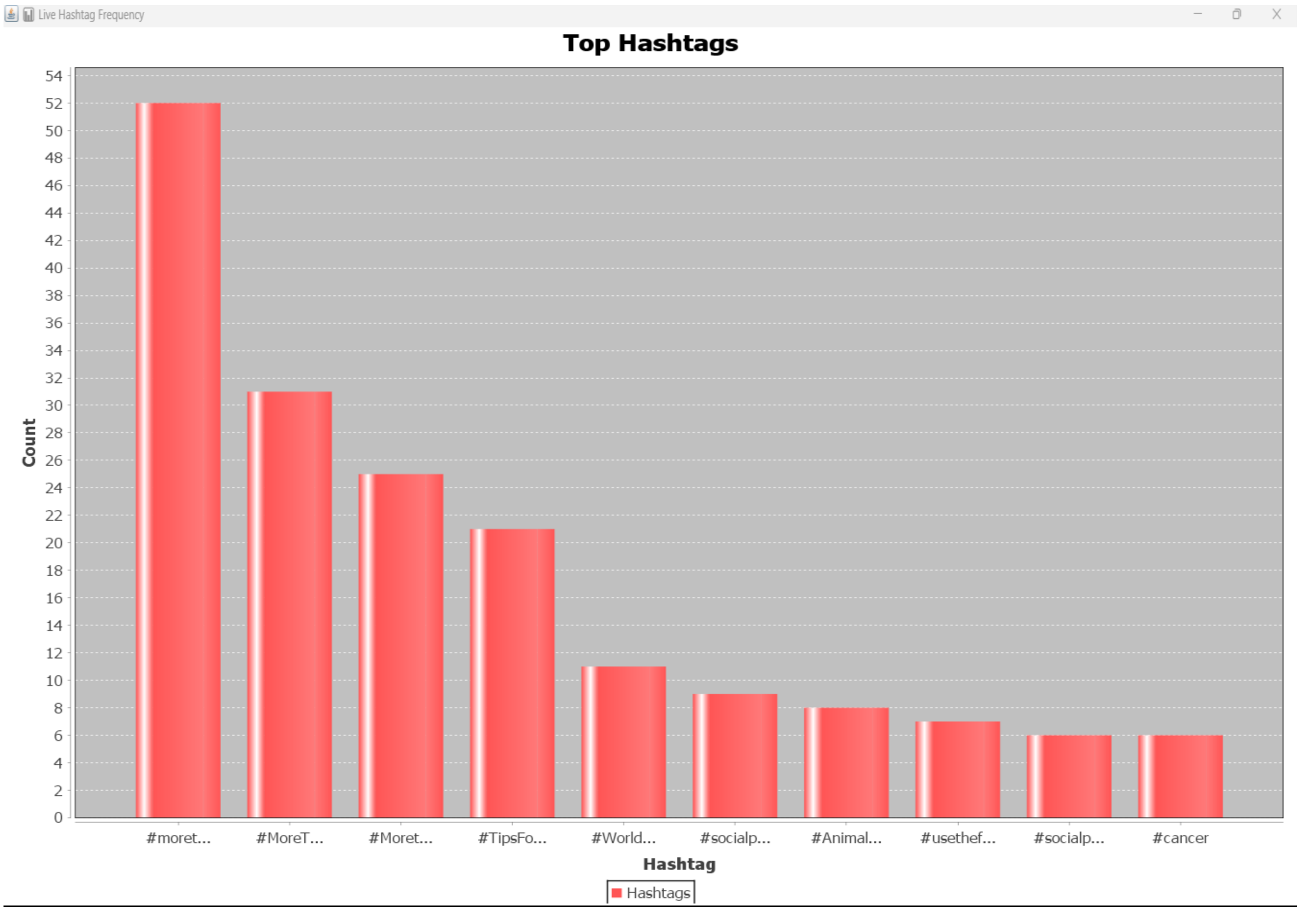
CODE for CONSUMER (in ECLIPSE IDE) :

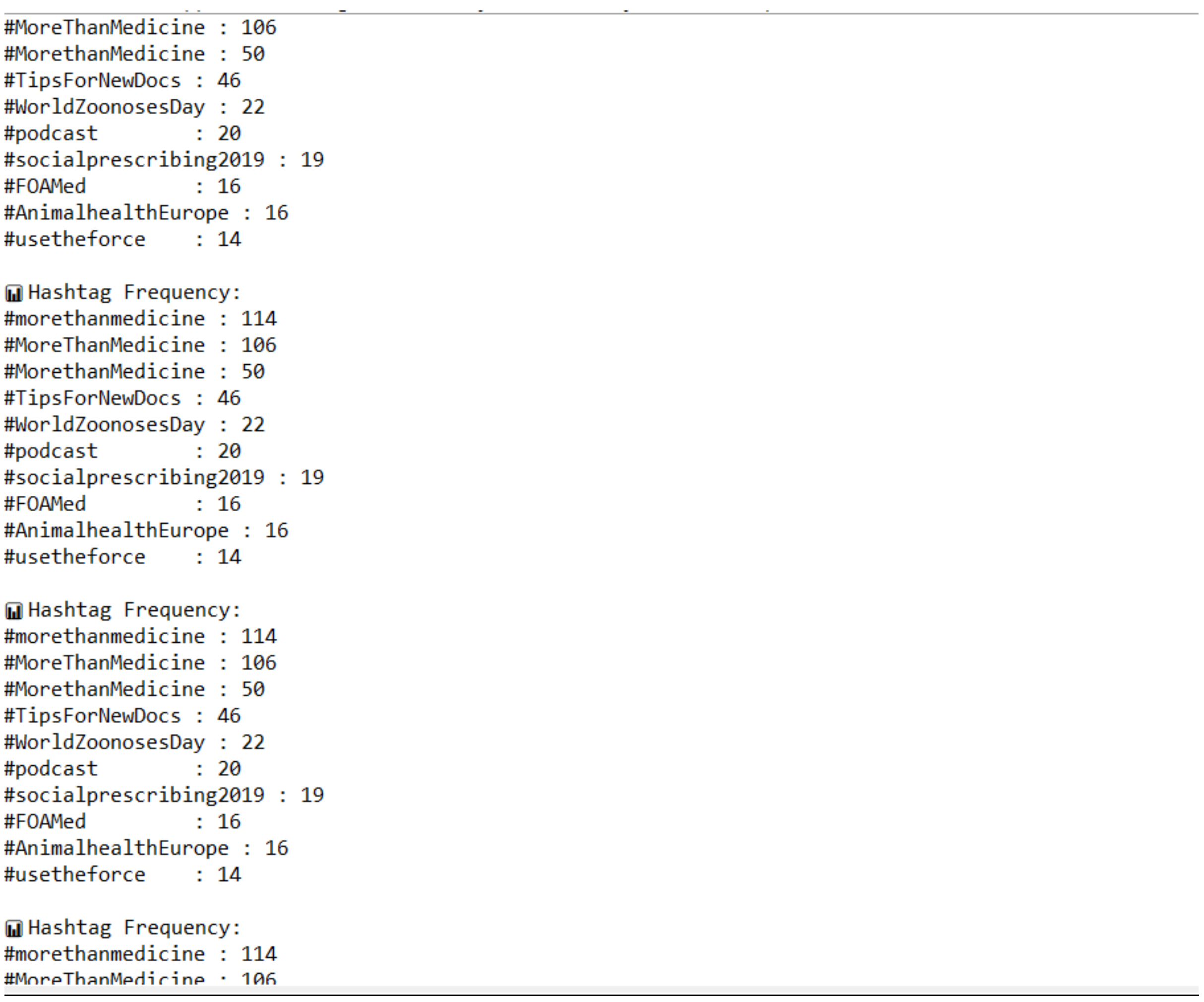
```
import org.apache.kafka.clients.consumer.*;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.data.category.DefaultCategoryDataset;

import javax.swing.*;
import java.time.Duration;
import java.util.*;

public class Consumer {
    private static final String TOPIC = "hashtags_topic";
    private static Map<String, Integer> hashtagCounts = new HashMap<>();
    private static DefaultCategoryDataset dataset = new
DefaultCategoryDataset();
```

```
  public static void main(String[] args) {
    // Set up Kafka consumer
    Properties props = new Properties();
    props.put("bootstrap.servers", "localhost:9092");
    props.put("group.id", "chart-consumer-group");
    props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
    props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");

    KafkaConsumer<String, String> consumer = new
KafkaConsumer<>(props);
    consumer.subscribe(Collections.singletonList(TOPIC));

    // Create chart UI
    SwingUtilities.invokeLater(() -> createChartWindow());

    System.out.println("📈 Chart consumer started...");

    while (true) {
      ConsumerRecords<String, String> records =
consumer.poll(Duration.ofMillis(1000));
      for (ConsumerRecord<String, String> record : records) {
        String hashtag = record.value();
        hashtagCounts.put(hashtag,
hashtagCounts.getOrDefault(hashtag, 0) + 1);
      }
```

```
        updateChart();
    }
}

private static void createChartWindow() {
    JFrame frame = new JFrame("📊 Live Hashtag Frequency");
    JFreeChart barChart = ChartFactory.createBarChart(
            "Top Hashtags",
            "Hashtag",
            "Count",
            dataset
    );

    ChartPanel chartPanel = new ChartPanel(barChart);
    chartPanel.setPreferredSize(new java.awt.Dimension(800, 600));
    frame.setContentPane(chartPanel);
    frame.pack();
    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

private static void updateChart() {
    dataset.clear();
    // Sort and show top 10 hashtags
    hashtagCounts.entrySet().stream()
            .sorted((a, b) -> b.getValue() - a.getValue())
            .limit(10)
            .forEach(e -> dataset.addValue(e.getValue(), "Hashtags",
e.getKey()));
   }
}
```

OUTPUT:





```
#MoreThanMedicine : 106
#MorethanMedicine : 50
#TipsForNewDocs : 46
#WorldZoonosesDay : 22
#podcast         : 20
#socialprescribing2019 : 19
#FOAMed          : 16
#AnimalhealthEurope : 16
#usetheforce     : 14

📊 Hashtag Frequency:
#morethanmedicine : 114
#MoreThanMedicine : 106
#MorethanMedicine : 50
#TipsForNewDocs : 46
#WorldZoonosesDay : 22
#podcast         : 20
#socialprescribing2019 : 19
#FOAMed          : 16
#AnimalhealthEurope : 16
#usetheforce     : 14

📊 Hashtag Frequency:
#morethanmedicine : 114
#MoreThanMedicine : 106
#MorethanMedicine : 50
#TipsForNewDocs : 46
#WorldZoonosesDay : 22
#podcast         : 20
#socialprescribing2019 : 19
#FOAMed          : 16
#AnimalhealthEurope : 16
#usetheforce     : 14

📊 Hashtag Frequency:
#morethanmedicine : 114
#MoreThanMedicine : 106
```

STEP BY STEP GUIDE TO RUN THE PROJECT:

1. Paste the code of PRODUCER and CONSUMER into eclipse IDE.
2. Open the PowerShell terminal in administrator mode.
3. Run zookeeper (.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties)
4. Again open the second PowerShell terminal in administrator mode and run server. (.\bin\windows\kafka-server-start.bat .\config\server.properties)

5. Again open the PowerShell terminal in administrator mode and create the topic (.\bin\windows\kafka-topics.bat --bootstrap-server 127.0.0.1:9092 --create --replication-factor 1 --partitions 1 --topic hashtags_topic) then press enter

6. Now run the consumer code in eciplse and run producer code after consumer

7. Output is ready.

DATSET LINK:

https://www.kaggle.com/datasets/ahmedshahriarsakib/tweet-sample