PES UNIVERSITY
100 feet Ring Road, BSK 3rd Stage
Bengaluru 560085 INDIA

Department of Computer Science and Engineering
B. Tech. CSE – 6th Semester
Jan – May 2024

UE21CS343BB3
DATABASE TECHNOLOGIES (DBT)

PROJECT REPORT
on

Stream and Batch Processing of Twitter data using Kafka and Spark

Submitted by: Team # 208_207_825_583

Girish Y P	207	6D
Shreyas N	583	6J

Gopikrishna G	208	6D
Omkar Terdal	825	6F

Class of Prof. Raghu B. A.

Table of Contents

1) Introduction: -

This project is about using **Apache Spark** and **Kafka** to process streaming data. The data being considered here are **Twitter feeds** stored in a **MySQL** Database. This data is continuously stored at random intervals and includes a random number of tweets by Kafka.

The project involves executing multiple workloads, such as Spark SQL queries, to perform actions, transformations, or aggregations on the input data. Additionally, Twitter Data may be retrieved because of a certain SQL query.

The project focuses on four topics: **tweets**, **hashtags**, **top_tweets**, and **top_hashtags**.

There are subscribers, both streaming consumers and batch consumers, who subscribe to the topic on a random basis. Metrics are run on the same data in both **streaming mode and batch mode**, and the accuracy and performance of these two modes are compared.

The project also includes computation examples like counting tweets within a window and applying aggregation functions on numeric data with each tumbling window. The window size for this project is significant and suitable to the chosen domain problem. For example, one window might include **15-30 minutes** of tweets.

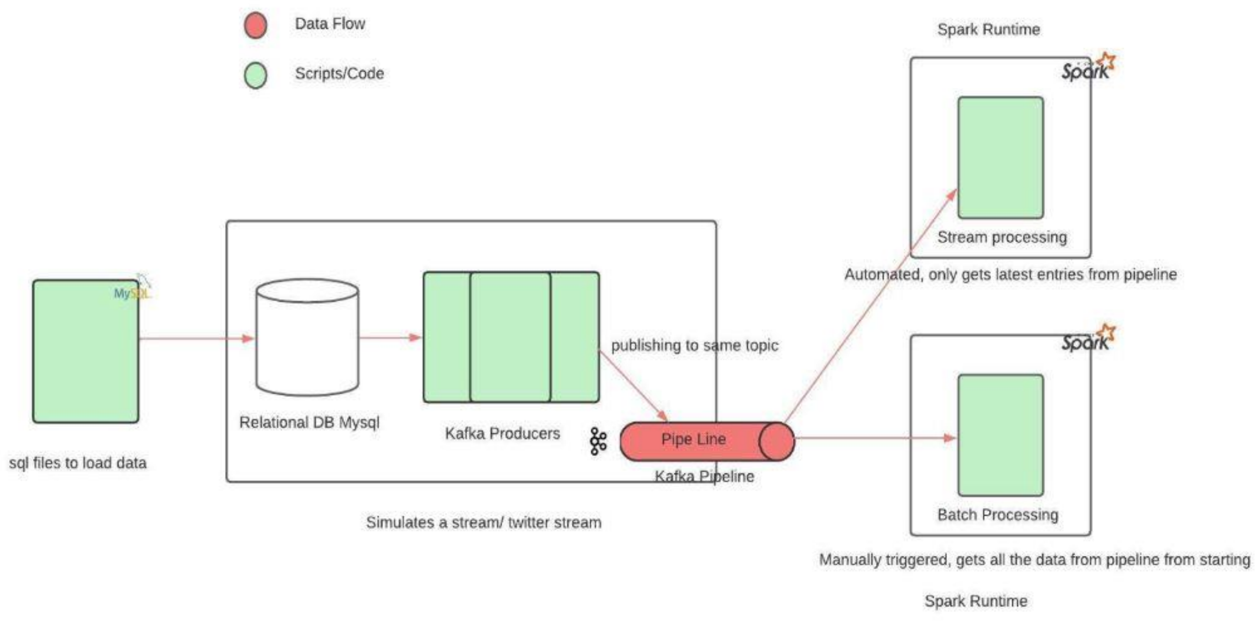
1.1) Problem Description:

This project primarily tackles the issue of handling the enormous amount of data produced by Twitter, and performing a variety of operations, transformations, and aggregations on it. The project strategy involves the use of Kafka to create multiple topics such as tweets, hashtags, top tweets, and top hashtags, among others. These topics pull data from a MySQL database, ensuring real-time data streaming as outlined in the project architecture.

The simulation is designed to **mimic** the performance of the **Twitter API**, with the overall goal of the project being to illustrate the efficient use of real-time streaming for processing and deriving valuable insights. Apache Spark, a subscriber in the overall model, is also utilized in this project.

Finally, a comparative topic is introduced that contrasts stream and batch processing applications across different windows. The problem statement and project intentions remain consistent throughout.

1.2) Architecture Diagram:



2) Installation of Software [include version #s and URLs]: -

Streaming tools used:

- **Apache Kafka (3.4.1):** A real-time distributed streaming platform that allows for the building of real-time data pipelines and streaming applications. We used this for streaming of data and as a publisher.

  LINK: Apache Kafka 3.7.0

- **Apache Spark (3.5.1):** A real-time processing engine built on top of the Apache Spark platform, allowing for processing of data streams in real-time: Also allows for batch processing.

  LINK: Apache Spark 3.5.1

- **PySpark (3.5.1):** A streaming engine like Apache Kafka but designed specifically for Python and its associated functionalities.

  To Install: pip3 install pyspark=="3.5.1"

DBMS & Tools used:

- **Mysql (8.0.36):** Used for storing the twitter data for processing.

  To install: sudo apt-get install mysql

3) Input data: -

Source: Twitter dataset randomly collected across multiple sources in the form of .CSV files from Kaggle, etc…

Description: This dataset contains the actual format of a general tweet message which includes the following metadata "id, username, id_created_at, date, time, timezone , user_id, conversation_id, place, tweet_language, mentions, urls, photos, replies_count, retweets_count, likes_count, hashtags, cashtags ,url_retweet ,url_videothumbnail, near_geo, source_link, id_user_rt_quote_url, id_reply_to, id_retweet_date, translate, _trans_src, _trans_dest".

But we choose only some data, that we need for our project. The following is the database and table, that is obtained from this dataset.

```
CREATE TABLE tweets (
tweet_id BIGINT PRIMARY KEY,
date_time VARCHAR(40),
tweet_text TEXT,
language VARCHAR(10)
);
CREATE TABLE hashtags (
hashtag_id BIGINT PRIMARY KEY,
hashtag VARCHAR(300) collate utf8mb4_bin
);
CREATE TABLE tweet_hashtags (
tweet_id BIGINT,
hashtag_id BIGINT,
CONSTRAINT pk_tweet_hashtag PRIMARY KEY(tweet_id, hashtag_id),
FOREIGN KEY(tweet_id) REFERENCES tweets(tweet_id) ON DELETE CASCADE,
FOREIGN KEY(hashtag_id) REFERENCES hashtags(hashtag_id) ON DELETE CASCADE
);
```

Kafka Topics (Streaming + Producer)

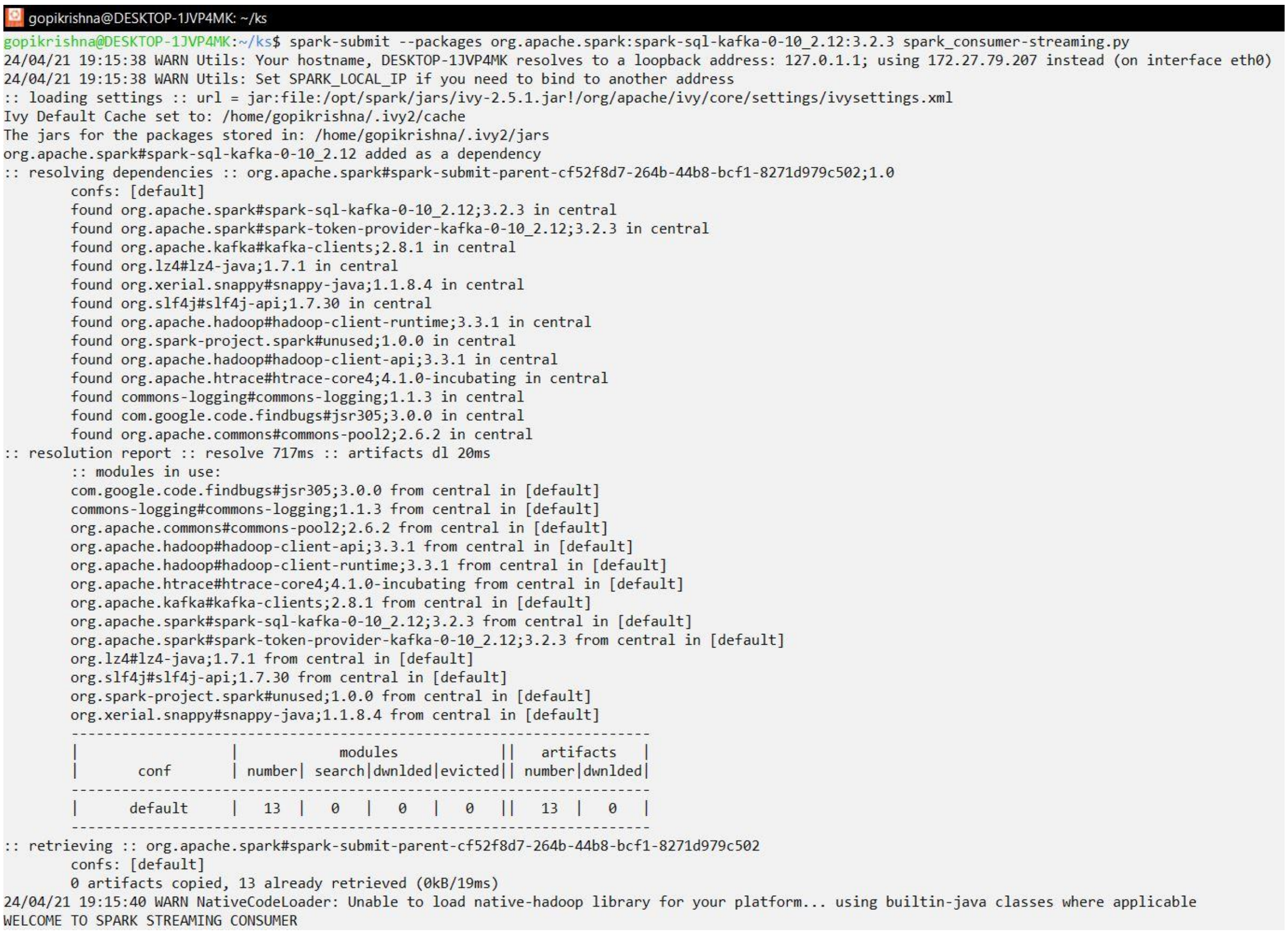
```
import random
from kafka import KafkaProducer
import mysql.connector
import time

# connect to the MySQL database
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="password",
    database="tweetDatabase"
)

# create a cursor to execute SQL queries
cursor = mydb.cursor()

# initialize Kafka producer
producer = KafkaProducer(bootstrap_servers=['localhost:9092'])

# continuously publish new tweets and hashtags to Kafka
while True:
    try:
        # select a random number of tweets to retrieve
        num_tweets = random.randint(1, 25)

        #-------------------------------------------------------------------
--------

        # select all tweets that were added since the last time the loop ran
        sql_tweets = "SELECT tweet_id, tweet, date_time, language FROM tweets WHERE
date_time > '2021-0>        cursor.execute(sql_tweets)
        # print the SQL query being executed
        print("Executing SQL query for tweets...")
        # publish each new tweet to the Kafka topic
        for tweet_id, tweet, date_time, language in cursor:
            message = bytes(f"{tweet_id},{tweet},{date_time},{language}",
encoding='utf-8')

            # print the data being processed
            print("Publishing tweet:", tweet)

            # publish the message to the Kafka topic
            producer.send('tweets', value=message)

        #----------------------------------------------------------------------------
--------

        # select a random number of hashtags to retrieve
        num_hashtags = random.randint(1, 10)

        # select all new hashtags that were added since the last time the loop ran
        sql_hashtags = "SELECT hashtag_id, hashtag FROM hashtags WHERE hashtag_id <
(SELECT MAX(hashtag>        cursor.execute(sql_hashtags)

        # print the SQL query being executed
        print("Executing SQL query for hashtags...")

        # publish each new hashtag to the Kafka topic
        for hashtag_id, hashtag in cursor:
           # serialize the row data to bytes
            message = bytes(f"{hashtag_id},{hashtag}", encoding='utf-8')

            # print the data being processed
            print("Publishing hashtag:", hashtag)
            # publish the message to the Kafka topic
            producer.send('hashtags', value=message)

        #----------------------------------------------------------------------------
--------

        # select the top 10 tweets with the most hashtags
        sql_top_tweets = "SELECT t.tweet_id, t.tweet, t.date_time, t.language,
COUNT(th.hashtag_id) as >        cursor.execute(sql_top_tweets)

        # print the SQL query being executed
        print("Executing SQL query for top tweets...")

        # publish each top tweet to the Kafka topic
        for tweet_id, tweet, date_time, language, hashtag_count in cursor:
             # serialize the row data to bytes
            message =
bytes(f"{tweet_id},{tweet},{date_time},{language},{hashtag_count}", encoding='utf>
            # print the data being processed
            print("Publishing top tweet:", tweet)
            # publish the message to the Kafka topic
            producer.send('top_tweets', value=message)

        #---------------------------------------------------------------------------
--------


        # select the hashtags that appear in the top 10 tweets
        sql_top_hashtags = "SELECT h.hashtag_id, h.hashtag, SUM(th.count) as count
FROM hashtags h JOIN>
        # execute the SQL query to retrieve the top hashtags
        cursor.execute(sql_top_hashtags)

        # print the SQL query being executed
        print("Executing SQL query for top hashtags...")

        # publish each top hashtag to the Kafka topic
        for hashtag_id, hashtag, count in cursor:
            # serialize the row data to bytes
            message = bytes(f"{hashtag_id},{hashtag},{count}", encoding='utf-8')

            # print the data being processed
            print("Publishing top hashtag:", hashtag)
            # publish the message to the Kafka topic
            producer.send('top_hashtags', value=message)

        #---------------------------------------------------------------------------
--------


        # sleep for a random amount of time between 5 and 15 seconds
        time.sleep(random.randint(5, 15))

    except Exception as e:
        print(e)
        # in case of any error, sleep for a shorter time and try again
        time.sleep(2)
```

4) Streaming Mode Experiment: -

Description: The code reads data in a streaming fashion (randomly selected), processes the data from Kafka topic, calculates the count of tweets per hashtag within a 30-minute window, sorts the result by window start time and hashtag, and writes the output to both the console and a CSV file. Here Pyspark is used to subscribe to the topic provided by Kafka.

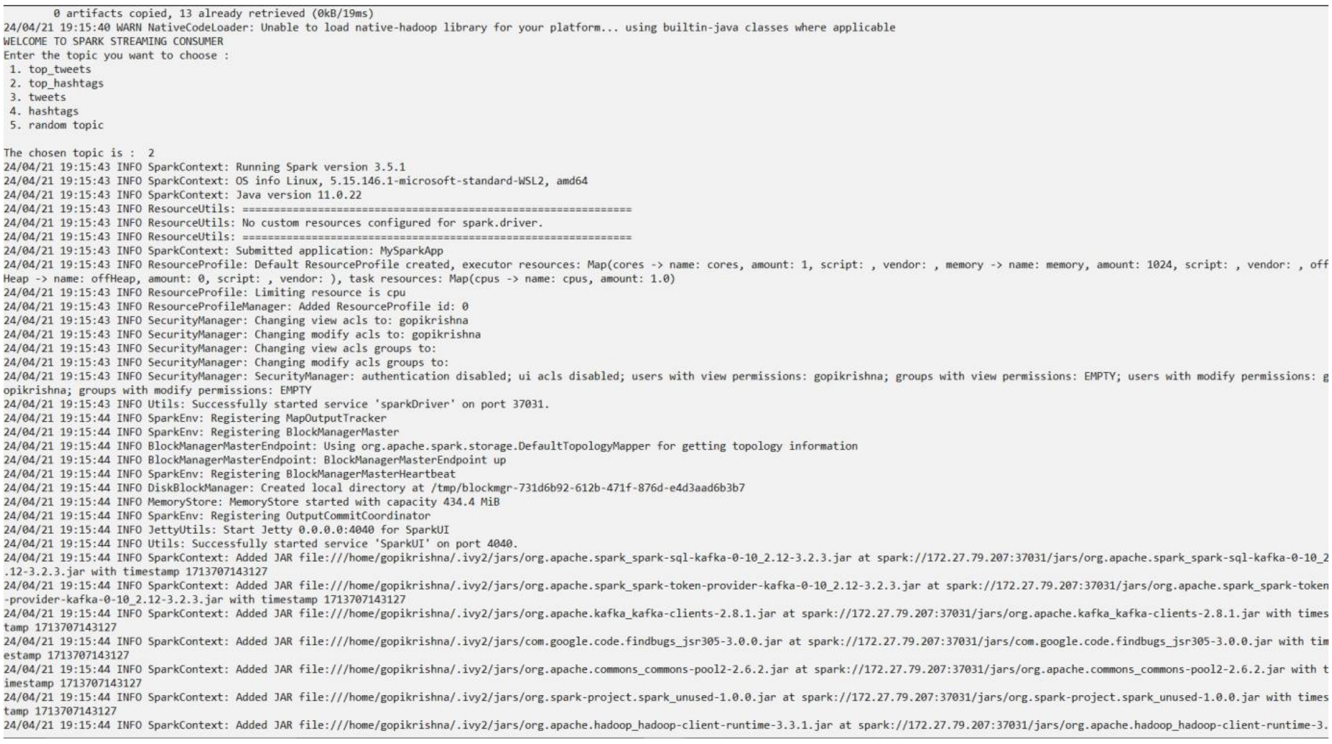
Windows: The code defines a sliding window of 15 minutes and groups the data within that, a query/data is provided. The 15 minutes tell us the fact that window based on topics maximum delayed up to 15 min, beyond which, it is discarded.

Results:

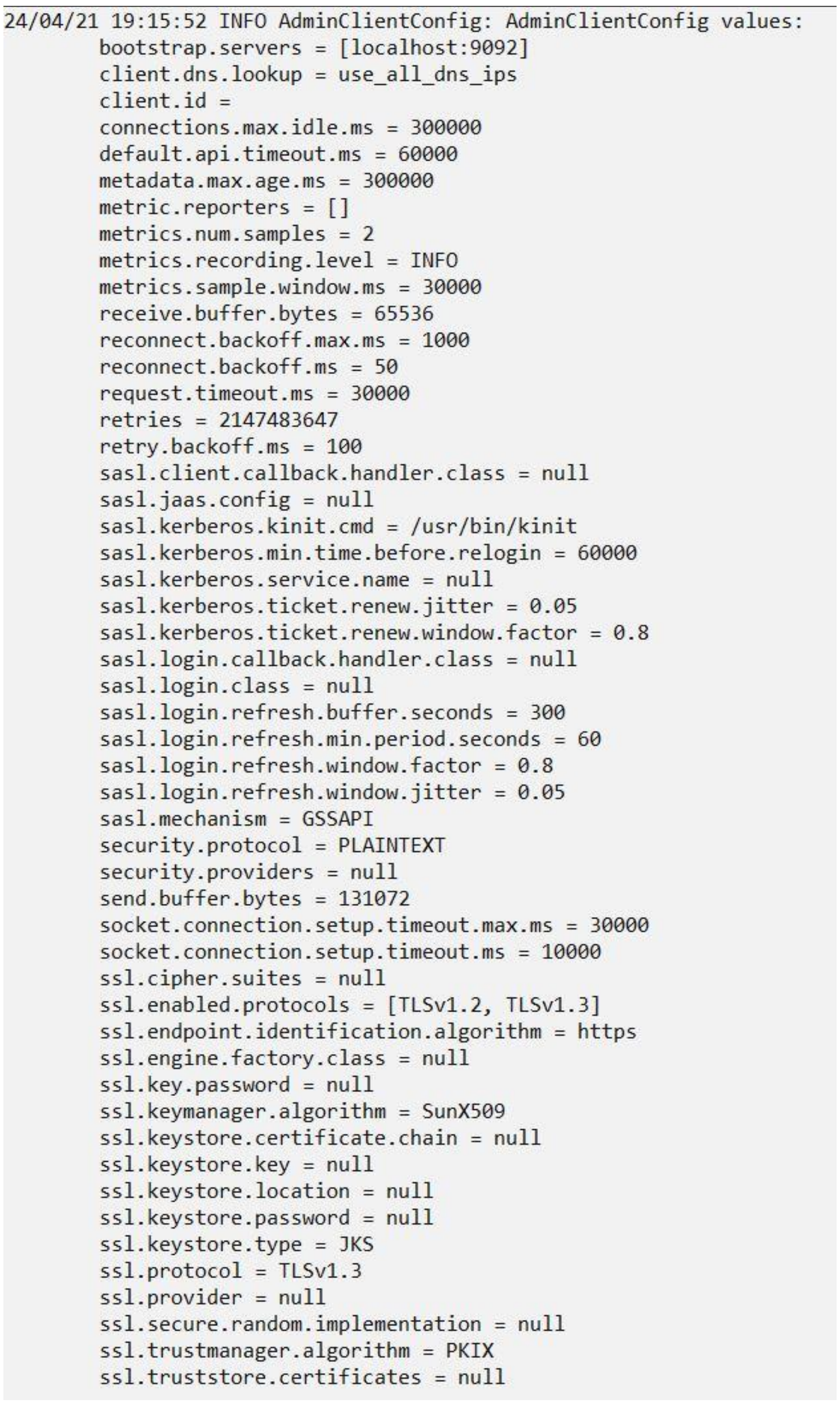


gopikrishna@DESKTOP-1JVP4MK: ~/ks

```
gopikrishna@DESKTOP-1JVP4MK:~/ks$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.2.3 spark_consumer-streaming.py
24/04/21 19:15:38 WARN Utils: Your hostname, DESKTOP-1JVP4MK resolves to a loopback address: 127.0.1.1; using 172.27.79.207 instead (on interface eth0)
24/04/21 19:15:38 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
:: loading settings :: url = jar:file:/opt/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/gopikrishna/.ivy2/cache
The jars for the packages stored in: /home/gopikrishna/.ivy2/jars
org.apache.spark#spark-sql-kafka-0-10_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-cf52f8d7-264b-44b8-bcf1-8271d979c502;1.0
        confs: [default]
        found org.apache.spark#spark-sql-kafka-0-10_2.12;3.2.3 in central
        found org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.2.3 in central
        found org.apache.kafka#kafka-clients;2.8.1 in central
        found org.lz4#lz4-java;1.7.1 in central
        found org.xerial.snappy#snappy-java;1.1.8.4 in central
        found org.slf4j#slf4j-api;1.7.30 in central
        found org.apache.hadoop#hadoop-client-runtime;3.3.1 in central
        found org.spark-project.spark#unused;1.0.0 in central
        found org.apache.hadoop#hadoop-client-api;3.3.1 in central
        found org.apache.htrace#htrace-core4;4.1.0-incubating in central
        found commons-logging#commons-logging;1.1.3 in central
        found com.google.code.findbugs#jsr305;3.0.0 in central
        found org.apache.commons#commons-pool2;2.6.2 in central
:: resolution report :: resolve 717ms :: artifacts dl 20ms
        :: modules in use:
        com.google.code.findbugs#jsr305;3.0.0 from central in [default]
        commons-logging#commons-logging;1.1.3 from central in [default]
        org.apache.commons#commons-pool2;2.6.2 from central in [default]
        org.apache.hadoop#hadoop-client-api;3.3.1 from central in [default]
        org.apache.hadoop#hadoop-client-runtime;3.3.1 from central in [default]
        org.apache.htrace#htrace-core4;4.1.0-incubating from central in [default]
        org.apache.kafka#kafka-clients;2.8.1 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.12;3.2.3 from central in [default]
        org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.2.3 from central in [default]
        org.lz4#lz4-java;1.7.1 from central in [default]
        org.slf4j#slf4j-api;1.7.30 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.8.4 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   13  |   0   |   0   |   0   ||   13  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-cf52f8d7-264b-44b8-bcf1-8271d979c502
        confs: [default]
        0 artifacts copied, 13 already retrieved (0kB/19ms)
24/04/21 19:15:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
WELCOME TO SPARK STREAMING CONSUMER
```



```
        0 artifacts copied, 13 already retrieved (0kB/19ms)
24/04/21 19:15:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
WELCOME TO SPARK STREAMING CONSUMER
Enter the topic you want to choose :
 1. top_tweets
 2. top_hashtags
 3. tweets
 4. hashtags
 5. random topic

The chosen topic is :  2
24/04/21 19:15:43 INFO SparkContext: Running Spark version 3.5.1
24/04/21 19:15:43 INFO SparkContext: OS info Linux, 5.15.146.1-microsoft-standard-WSL2, amd64
24/04/21 19:15:43 INFO SparkContext: Java version 11.0.22
24/04/21 19:15:43 INFO ResourceUtils: ==============================================================
24/04/21 19:15:43 INFO ResourceUtils: No custom resources configured for spark.driver.
24/04/21 19:15:43 INFO ResourceUtils: ==============================================================
24/04/21 19:15:43 INFO SparkContext: Submitted application: MySparkApp
24/04/21 19:15:43 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory -> name: memory, amount: 1024, script: , vendor: , off
Heap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)
24/04/21 19:15:43 INFO ResourceProfile: Limiting resource is cpu
24/04/21 19:15:43 INFO ResourceProfileManager: Added ResourceProfile id: 0
24/04/21 19:15:43 INFO SecurityManager: Changing view acls to: gopikrishna
24/04/21 19:15:43 INFO SecurityManager: Changing modify acls to: gopikrishna
24/04/21 19:15:43 INFO SecurityManager: Changing view acls groups to:
24/04/21 19:15:43 INFO SecurityManager: Changing modify acls groups to:
24/04/21 19:15:43 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: gopikrishna; groups with view permissions: EMPTY; users with modify permissions: g
opikrishna; groups with modify permissions: EMPTY
24/04/21 19:15:43 INFO Utils: Successfully started service 'sparkDriver' on port 37031.
24/04/21 19:15:44 INFO SparkEnv: Registering MapOutputTracker
24/04/21 19:15:44 INFO SparkEnv: Registering BlockManagerMaster
24/04/21 19:15:44 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
24/04/21 19:15:44 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
24/04/21 19:15:44 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/04/21 19:15:44 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-731d6b92-612b-471f-876d-e4d3aad6b3b7
24/04/21 19:15:44 INFO MemoryStore: MemoryStore started with capacity 434.4 MiB
24/04/21 19:15:44 INFO SparkEnv: Registering OutputCommitCoordinator
24/04/21 19:15:44 INFO JettyUtils: Start Jetty 0.0.0.0:4040 for SparkUI
24/04/21 19:15:44 INFO Utils: Successfully started service 'SparkUI' on port 4040.
24/04/21 19:15:44 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.apache.spark_spark-sql-kafka-0-10_2.12-3.2.3.jar at spark://172.27.79.207:37031/jars/org.apache.spark_spark-sql-kafka-0-10_2
.12-3.2.3.jar with timestamp 1713707143127
24/04/21 19:15:44 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.apache.spark_spark-token-provider-kafka-0-10_2.12-3.2.3.jar at spark://172.27.79.207:37031/jars/org.apache.spark_spark-token
-provider-kafka-0-10_2.12-3.2.3.jar with timestamp 1713707143127
24/04/21 19:15:44 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.apache.kafka_kafka-clients-2.8.1.jar at spark://172.27.79.207:37031/jars/org.apache.kafka_kafka-clients-2.8.1.jar with times
tamp 1713707143127
24/04/21 19:15:44 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/com.google.code.findbugs_jsr305-3.0.0.jar at spark://172.27.79.207:37031/jars/com.google.code.findbugs_jsr305-3.0.0.jar with tim
estamp 1713707143127
24/04/21 19:15:44 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.apache.commons_commons-pool2-2.6.2.jar at spark://172.27.79.207:37031/jars/org.apache.commons_commons-pool2-2.6.2.jar with t
imestamp 1713707143127
24/04/21 19:15:44 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.spark-project.spark_unused-1.0.0.jar at spark://172.27.79.207:37031/jars/org.spark-project.spark_unused-1.0.0.jar with times
tamp 1713707143127
24/04/21 19:15:44 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.apache.hadoop_hadoop-client-runtime-3.3.1.jar at spark://172.27.79.207:37031/jars/org.apache.hadoop_hadoop-client-runtime-3.
```



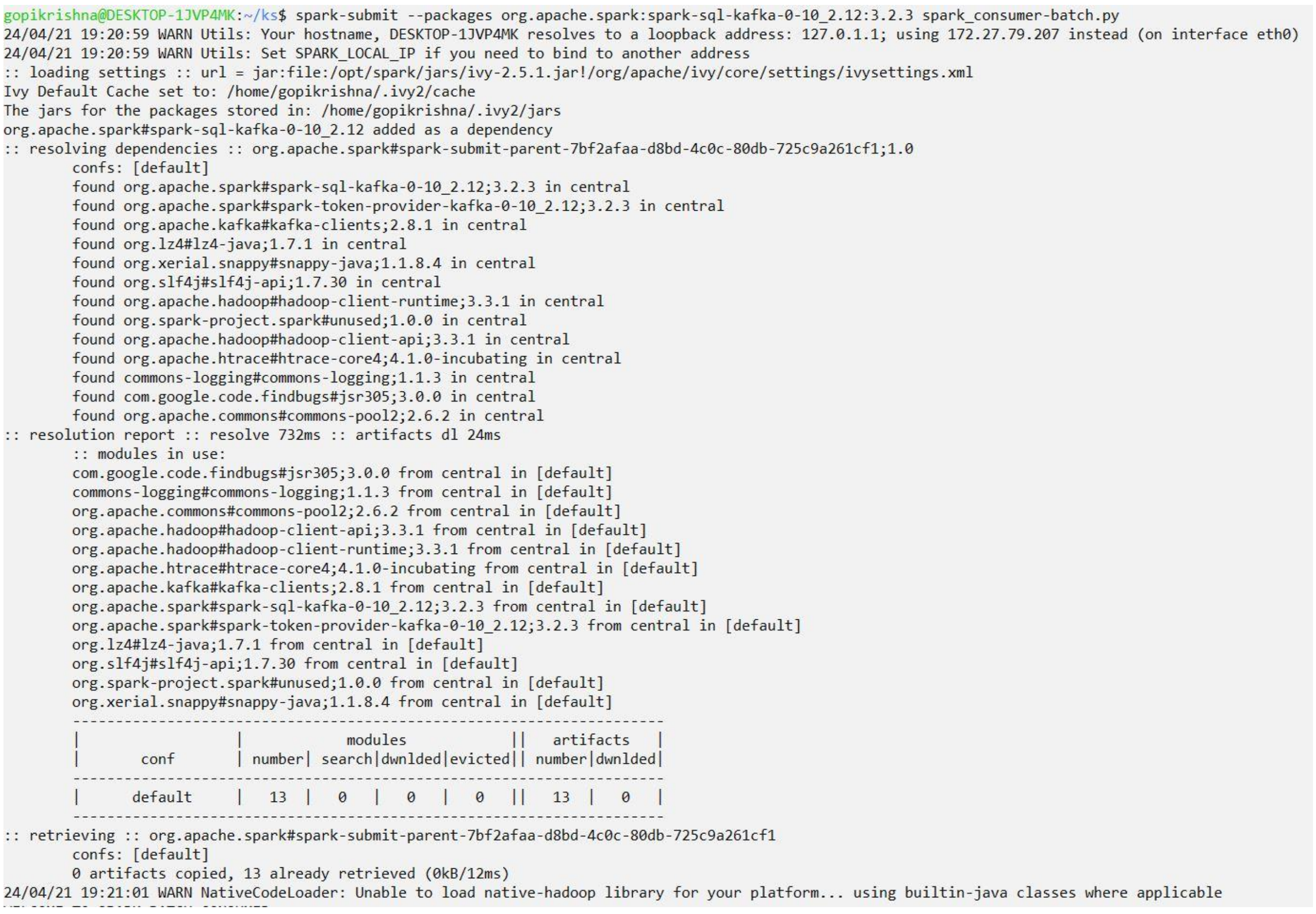
```
24/04/21 19:15:52 INFO AdminClientConfig: AdminClientConfig values:
        bootstrap.servers = [localhost:9092]
        client.dns.lookup = use_all_dns_ips
        client.id =
        connections.max.idle.ms = 300000
        default.api.timeout.ms = 60000
        metadata.max.age.ms = 300000
        metric.reporters = []
        metrics.num.samples = 2
        metrics.recording.level = INFO
        metrics.sample.window.ms = 30000
        receive.buffer.bytes = 65536
        reconnect.backoff.max.ms = 1000
        reconnect.backoff.ms = 50
        request.timeout.ms = 30000
        retries = 2147483647
        retry.backoff.ms = 100
        sasl.client.callback.handler.class = null
        sasl.jaas.config = null
        sasl.kerberos.kinit.cmd = /usr/bin/kinit
        sasl.kerberos.min.time.before.relogin = 60000
        sasl.kerberos.service.name = null
        sasl.kerberos.ticket.renew.jitter = 0.05
        sasl.kerberos.ticket.renew.window.factor = 0.8
        sasl.login.callback.handler.class = null
        sasl.login.class = null
        sasl.login.refresh.buffer.seconds = 300
        sasl.login.refresh.min.period.seconds = 60
        sasl.login.refresh.window.factor = 0.8
        sasl.login.refresh.window.jitter = 0.05
        sasl.mechanism = GSSAPI
        security.protocol = PLAINTEXT
        security.providers = null
        send.buffer.bytes = 131072
        socket.connection.setup.timeout.max.ms = 30000
        socket.connection.setup.timeout.ms = 10000
        ssl.cipher.suites = null
        ssl.enabled.protocols = [TLSv1.2, TLSv1.3]
        ssl.endpoint.identification.algorithm = https
        ssl.engine.factory.class = null
        ssl.key.password = null
        ssl.keymanager.algorithm = SunX509
        ssl.keystore.certificate.chain = null
        ssl.keystore.key = null
        ssl.keystore.location = null
        ssl.keystore.password = null
        ssl.keystore.type = JKS
        ssl.protocol = TLSv1.3
        ssl.provider = null
        ssl.secure.random.implementation = null
        ssl.trustmanager.algorithm = PKIX
        ssl.truststore.certificates = null
```

5) Batch Mode Experiment: -

Description: This program reads data from a Kafka topic named "tweets" in a streaming fashion using Spark Structured Streaming. The data is parsed using a predefined schema and processed to count the number of tweets based on their language within a sliding window of 5 minutes. The processed results are then output to the console using the complete mode of output.

Data Size: The size of the data is 20 rows per batch. It depends on the size of the data produced and stored in the Kafka topic.

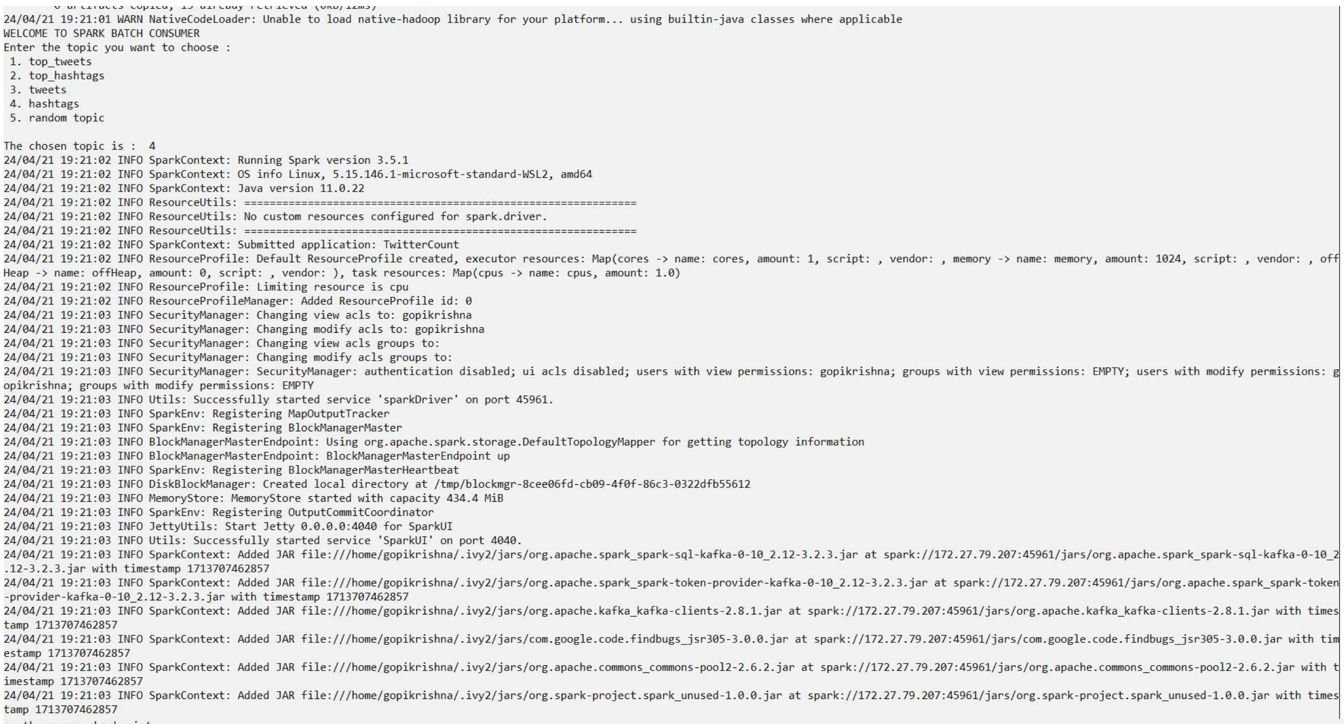
Results:



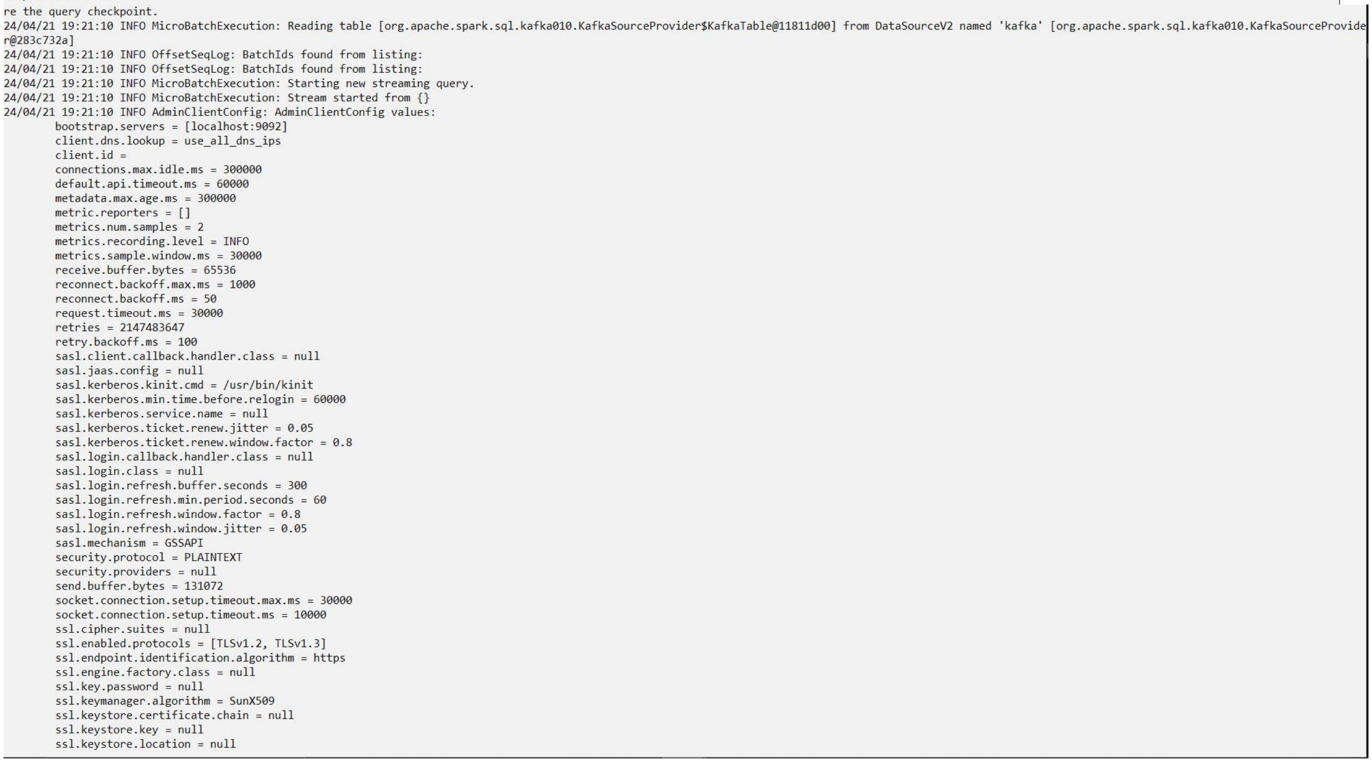
```
gopikrishna@DESKTOP-1JVP4MK:~/ks$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.2.3 spark_consumer-batch.py
24/04/21 19:20:59 WARN Utils: Your hostname, DESKTOP-1JVP4MK resolves to a loopback address: 127.0.1.1; using 172.27.79.207 instead (on interface eth0)
24/04/21 19:20:59 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
:: loading settings :: url = jar:file:/opt/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /home/gopikrishna/.ivy2/cache
The jars for the packages stored in: /home/gopikrishna/.ivy2/jars
org.apache.spark#spark-sql-kafka-0-10_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-7bf2afaa-d8bd-4c0c-80db-725c9a261cf1;1.0
        confs: [default]
        found org.apache.spark#spark-sql-kafka-0-10_2.12;3.2.3 in central
        found org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.2.3 in central
        found org.apache.kafka#kafka-clients;2.8.1 in central
        found org.lz4#lz4-java;1.7.1 in central
        found org.xerial.snappy#snappy-java;1.1.8.4 in central
        found org.slf4j#slf4j-api;1.7.30 in central
        found org.apache.hadoop#hadoop-client-runtime;3.3.1 in central
        found org.spark-project.spark#unused;1.0.0 in central
        found org.apache.hadoop#hadoop-client-api;3.3.1 in central
        found org.apache.htrace#htrace-core4;4.1.0-incubating in central
        found commons-logging#commons-logging;1.1.3 in central
        found com.google.code.findbugs#jsr305;3.0.0 in central
        found org.apache.commons#commons-pool2;2.6.2 in central
:: resolution report :: resolve 732ms :: artifacts dl 24ms
        :: modules in use:
        com.google.code.findbugs#jsr305;3.0.0 from central in [default]
        commons-logging#commons-logging;1.1.3 from central in [default]
        org.apache.commons#commons-pool2;2.6.2 from central in [default]
        org.apache.hadoop#hadoop-client-api;3.3.1 from central in [default]
        org.apache.hadoop#hadoop-client-runtime;3.3.1 from central in [default]
        org.apache.htrace#htrace-core4;4.1.0-incubating from central in [default]
        org.apache.kafka#kafka-clients;2.8.1 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.12;3.2.3 from central in [default]
        org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.2.3 from central in [default]
        org.lz4#lz4-java;1.7.1 from central in [default]
        org.slf4j#slf4j-api;1.7.30 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.8.4 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   13  |   0   |   0   |   0   ||   13  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-7bf2afaa-d8bd-4c0c-80db-725c9a261cf1
        confs: [default]
        0 artifacts copied, 13 already retrieved (0kB/12ms)
24/04/21 19:21:01 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

•

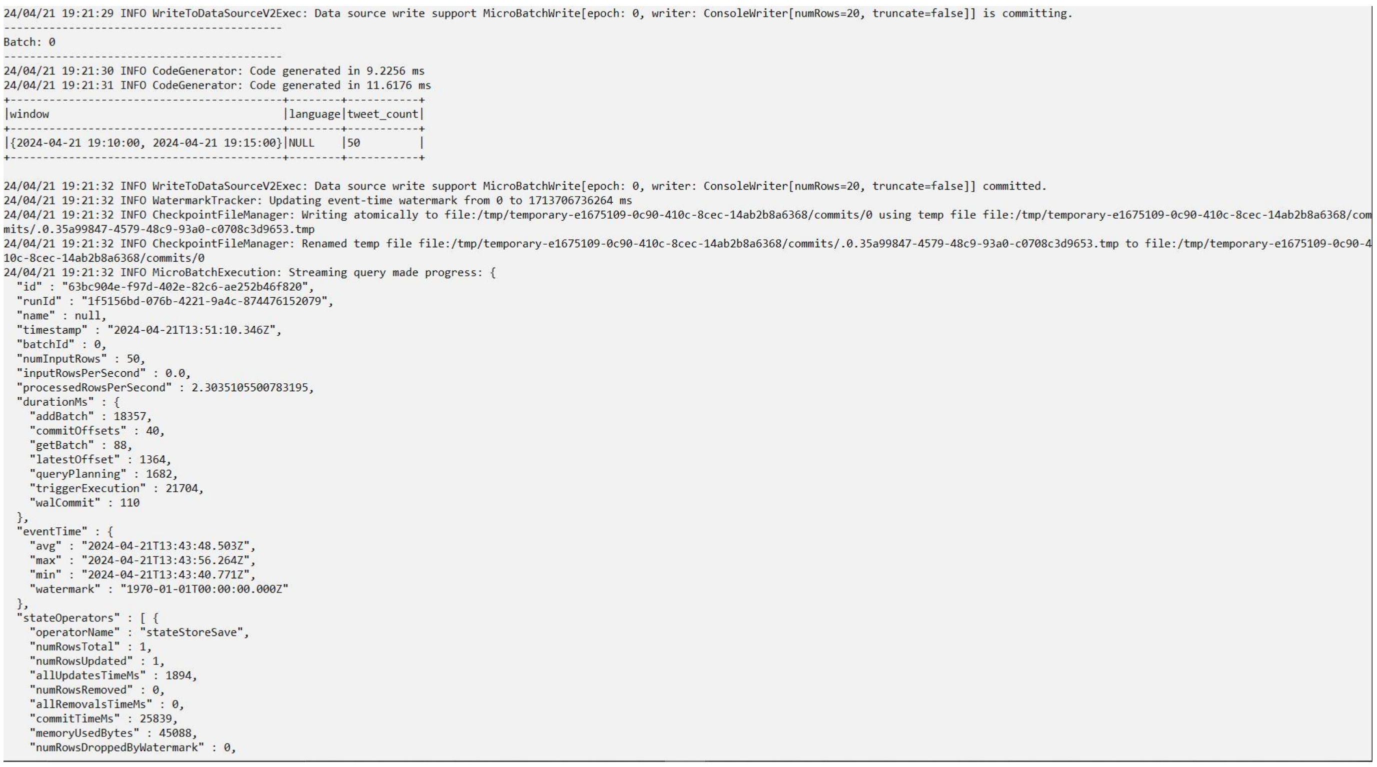
•



```
        0 artifacts copied, 15 already retrieved (0kB/12ms)
24/04/21 19:21:01 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
WELCOME TO SPARK BATCH CONSUMER
Enter the topic you want to choose :
 1. top_tweets
 2. top_hashtags
 3. tweets
 4. hashtags
 5. random topic

The chosen topic is :  4
24/04/21 19:21:02 INFO SparkContext: Running Spark version 3.5.1
24/04/21 19:21:02 INFO SparkContext: OS info Linux, 5.15.146.1-microsoft-standard-WSL2, amd64
24/04/21 19:21:02 INFO SparkContext: Java version 11.0.22
24/04/21 19:21:02 INFO ResourceUtils: ==============================================================
24/04/21 19:21:02 INFO ResourceUtils: No custom resources configured for spark.driver.
24/04/21 19:21:02 INFO ResourceUtils: ==============================================================
24/04/21 19:21:02 INFO SparkContext: Submitted application: TwitterCount
24/04/21 19:21:02 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory -> name: memory, amount: 1024, script: , vendor: , off
Heap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)
24/04/21 19:21:02 INFO ResourceProfile: Limiting resource is cpu
24/04/21 19:21:02 INFO ResourceProfileManager: Added ResourceProfile id: 0
24/04/21 19:21:03 INFO SecurityManager: Changing view acls to: gopikrishna
24/04/21 19:21:03 INFO SecurityManager: Changing modify acls to: gopikrishna
24/04/21 19:21:03 INFO SecurityManager: Changing view acls groups to:
24/04/21 19:21:03 INFO SecurityManager: Changing modify acls groups to:
24/04/21 19:21:03 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: gopikrishna; groups with view permissions: EMPTY; users with modify permissions: g
opikrishna; groups with modify permissions: EMPTY
24/04/21 19:21:03 INFO Utils: Successfully started service 'sparkDriver' on port 45961.
24/04/21 19:21:03 INFO SparkEnv: Registering MapOutputTracker
24/04/21 19:21:03 INFO SparkEnv: Registering BlockManagerMaster
24/04/21 19:21:03 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
24/04/21 19:21:03 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
24/04/21 19:21:03 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
24/04/21 19:21:03 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-8cee06fd-cb09-4f0f-86c3-0322dfb55612
24/04/21 19:21:03 INFO MemoryStore: MemoryStore started with capacity 434.4 MiB
24/04/21 19:21:03 INFO SparkEnv: Registering OutputCommitCoordinator
24/04/21 19:21:03 INFO JettyUtils: Start Jetty 0.0.0.0:4040 for SparkUI
24/04/21 19:21:03 INFO Utils: Successfully started service 'SparkUI' on port 4040.
24/04/21 19:21:03 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.apache.spark_spark-sql-kafka-0-10_2.12-3.2.3.jar at spark://172.27.79.207:45961/jars/org.apache.spark_spark-sql-kafka-0-10_2
.12-3.2.3.jar with timestamp 1713707462857
24/04/21 19:21:03 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.apache.spark_spark-token-provider-kafka-0-10_2.12-3.2.3.jar at spark://172.27.79.207:45961/jars/org.apache.spark_spark-token
-provider-kafka-0-10_2.12-3.2.3.jar with timestamp 1713707462857
24/04/21 19:21:03 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.apache.kafka_kafka-clients-2.8.1.jar at spark://172.27.79.207:45961/jars/org.apache.kafka_kafka-clients-2.8.1.jar with times
tamp 1713707462857
24/04/21 19:21:03 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/com.google.code.findbugs_jsr305-3.0.0.jar at spark://172.27.79.207:45961/jars/com.google.code.findbugs_jsr305-3.0.0.jar with tim
estamp 1713707462857
24/04/21 19:21:03 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.apache.commons_commons-pool2-2.6.2.jar at spark://172.27.79.207:45961/jars/org.apache.commons_commons-pool2-2.6.2.jar with t
imestamp 1713707462857
24/04/21 19:21:03 INFO SparkContext: Added JAR file:///home/gopikrishna/.ivy2/jars/org.spark-project.spark_unused-1.0.0.jar at spark://172.27.79.207:45961/jars/org.spark-project.spark_unused-1.0.0.jar with times
tamp 1713707462857
```



```
re the query checkpoint.
24/04/21 19:21:10 INFO MicroBatchExecution: Reading table [org.apache.spark.sql.kafka010.KafkaSourceProvider$KafkaTable@11811d00] from DataSourceV2 named 'kafka' [org.apache.spark.sql.kafka010.KafkaSourceProvide
r@283c732a]
24/04/21 19:21:10 INFO OffsetSeqLog: BatchIds found from listing:
24/04/21 19:21:10 INFO OffsetSeqLog: BatchIds found from listing:
24/04/21 19:21:10 INFO MicroBatchExecution: Starting new streaming query.
24/04/21 19:21:10 INFO MicroBatchExecution: Stream started from {}
24/04/21 19:21:10 INFO AdminClientConfig: AdminClientConfig values:
        bootstrap.servers = [localhost:9092]
        client.dns.lookup = use_all_dns_ips
        client.id =
        connections.max.idle.ms = 300000
        default.api.timeout.ms = 60000
        metadata.max.age.ms = 300000
        metric.reporters = []
        metrics.num.samples = 2
        metrics.recording.level = INFO
        metrics.sample.window.ms = 30000
        receive.buffer.bytes = 65536
        reconnect.backoff.max.ms = 1000
        reconnect.backoff.ms = 50
        request.timeout.ms = 30000
        retries = 2147483647
        retry.backoff.ms = 100
        sasl.client.callback.handler.class = null
        sasl.jaas.config = null
        sasl.kerberos.kinit.cmd = /usr/bin/kinit
        sasl.kerberos.min.time.before.relogin = 60000
        sasl.kerberos.service.name = null
        sasl.kerberos.ticket.renew.jitter = 0.05
        sasl.kerberos.ticket.renew.window.factor = 0.8
        sasl.login.callback.handler.class = null
        sasl.login.class = null
        sasl.login.refresh.buffer.seconds = 300
        sasl.login.refresh.min.period.seconds = 60
        sasl.login.refresh.window.factor = 0.8
        sasl.login.refresh.window.jitter = 0.05
        sasl.mechanism = GSSAPI
        security.protocol = PLAINTEXT
        security.providers = null
        send.buffer.bytes = 131072
        socket.connection.setup.timeout.max.ms = 30000
        socket.connection.setup.timeout.ms = 10000
        ssl.cipher.suites = null
        ssl.enabled.protocols = [TLSv1.2, TLSv1.3]
        ssl.endpoint.identification.algorithm = https
        ssl.engine.factory.class = null
        ssl.key.password = null
        ssl.keymanager.algorithm = SunX509
        ssl.keystore.certificate.chain = null
        ssl.keystore.key = null
        ssl.keystore.location = null
```



```
24/04/21 19:21:29 INFO WriteToDataSourceV2Exec: Data source write support MicroBatchWrite[epoch: 0, writer: ConsoleWriter[numRows=20, truncate=false]] is committing.
-------------------------------------------
Batch: 0
-------------------------------------------
24/04/21 19:21:30 INFO CodeGenerator: Code generated in 9.2256 ms
24/04/21 19:21:31 INFO CodeGenerator: Code generated in 11.6176 ms
+------------------------------------------+--------+-----------+
|window                                    |language|tweet_count|
+------------------------------------------+--------+-----------+
|{2024-04-21 19:10:00, 2024-04-21 19:15:00}|NULL    |50         |
+------------------------------------------+--------+-----------+

24/04/21 19:21:32 INFO WriteToDataSourceV2Exec: Data source write support MicroBatchWrite[epoch: 0, writer: ConsoleWriter[numRows=20, truncate=false]] committed.
24/04/21 19:21:32 INFO WatermarkTracker: Updating event-time watermark from 0 to 1713706736264 ms
24/04/21 19:21:32 INFO CheckpointFileManager: Writing atomically to file:/tmp/temporary-e1675109-0c90-410c-8cec-14ab2b8a6368/commits/0 using temp file file:/tmp/temporary-e1675109-0c90-410c-8cec-14ab2b8a6368/com
mits/.0.35a99847-4579-48c9-93a0-c0708c3d9653.tmp
24/04/21 19:21:32 INFO CheckpointFileManager: Renamed temp file file:/tmp/temporary-e1675109-0c90-410c-8cec-14ab2b8a6368/commits/.0.35a99847-4579-48c9-93a0-c0708c3d9653.tmp to file:/tmp/temporary-e1675109-0c90-4
10c-8cec-14ab2b8a6368/commits/0
24/04/21 19:21:32 INFO MicroBatchExecution: Streaming query made progress: {
  "id" : "63bc904e-f97d-402e-82c6-ae252b46f820",
  "runId" : "1f5156bd-076b-4221-9a4c-874476152079",
  "name" : null,
  "timestamp" : "2024-04-21T13:51:10.346Z",
  "batchId" : 0,
  "numInputRows" : 50,
  "inputRowsPerSecond" : 0.0,
  "processedRowsPerSecond" : 2.3035105500783195,
  "durationMs" : {
    "addBatch" : 18357,
    "commitOffsets" : 40,
    "getBatch" : 88,
    "latestOffset" : 1364,
    "queryPlanning" : 1682,
    "triggerExecution" : 21704,
    "walCommit" : 110
  },
  "eventTime" : {
    "avg" : "2024-04-21T13:43:48.503Z",
    "max" : "2024-04-21T13:43:56.264Z",
    "min" : "2024-04-21T13:43:40.771Z",
    "watermark" : "1970-01-01T00:00:00.000Z"
  },
  "stateOperators" : [ {
    "operatorName" : "stateStoreSave",
    "numRowsTotal" : 1,
    "numRowsUpdated" : 1,
    "allUpdatesTimeMs" : 1894,
    "numRowsRemoved" : 0,
    "allRemovalsTimeMs" : 0,
    "commitTimeMs" : 25839,
    "memoryUsedBytes" : 45088,
    "numRowsDroppedByWatermark" : 0,
```

6) Comparison of Streaming & Batch Modes: -

Results and Discussion: For this case we used both the streaming and batch process to subscribe to only Topic-1 i.e. top_tweets. We take all the statistics of computation, aggregation, count by window, etc…
When observed, we see that processed #row in seconds is near 0 to 0 in case of streaming as the processing taking place in a very near time which is the actual real time, whereas #row in batch processing it is over 100s after some time which indicates that we are processing the data only after some time. This is the case with number of rows input into process model. Also, memory and amount of commit time is less in stream processing, compared to batch processing, which account for huge data processed into both processes, not applicable to small dataset.
These are some of the statistics available when we compare the result of both processes. There are many more parameters on which we can evaluate on.

7) Conclusion: -

In conclusion, streaming and batch processing are both important concepts in data processing. Streaming is suitable for handling data in real-time or near real-time, where data is processed as it arrives, whereas batch processing is more suitable for processing large volumes of data at once.

In this code example, batch processing was used to process a large volume of tweets received from a Kafka topic over a specific time (window duration), grouped by language and timestamp windows and counted to show the total number of tweets. The results were printed to the console in the "complete" output mode.

Ultimately, the choice between streaming and batch processing depends on the nature of the data being processed and the specific requirements of the use case. Both have their own advantages and disadvantages, and it's up to data engineer or scientist to choose the appropriate processing mode for their specific needs. The results metrics as discussed above will show difference between as mentioned above.

8) References: -

URL's:

- [1] CSV to SQL converter
- [2] Kafka Producer in Python
- [3] Structured Streaming Guide