Streaming Weather Data

BY:-
Redon N Roy
Rushikesh Lavate
Meenal Shree
Nirosha M
Neha Kumari

REVATURE

Our Project

Our project deals with the processing of the data from www.weatherbit.io using Kafka and Spark Streaming .

Here we are simulating the streaming data using previous days data and visualizing the outcome using matplotlib. The data processing is developed using Python.

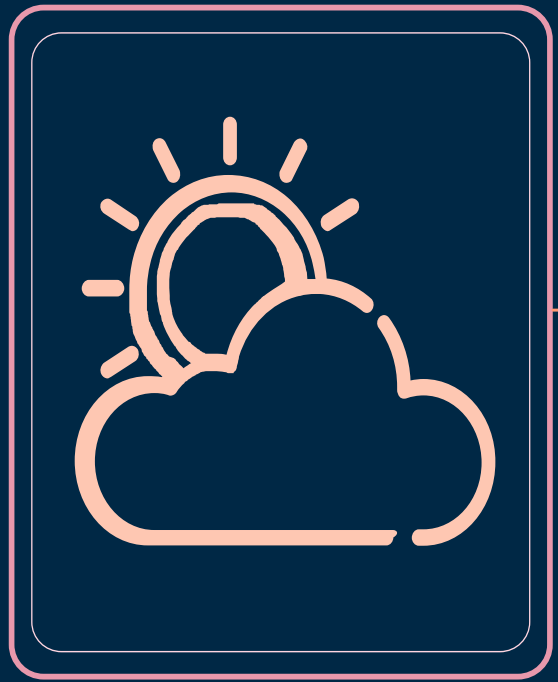
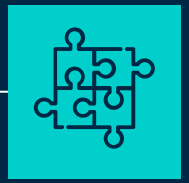


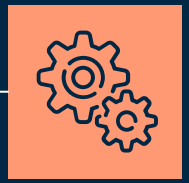
Table of Contents



01

PROBLEM & SOLUTION

Going over the problem statement and presenting our solution



02

CODE WALKTHROUGH

Explaining the code behind our solution and live demo

03

CONCLUSION

Discussing scope, future development and conclusion

REVATURE

PROBLEM & SOLUTION

Understanding the Problem

How to process raw stream data

One of the problem we are addressing in this project is the processing of raw stream data coming from multiple sources and getting only the required data from it.

Visualize the data at real-time

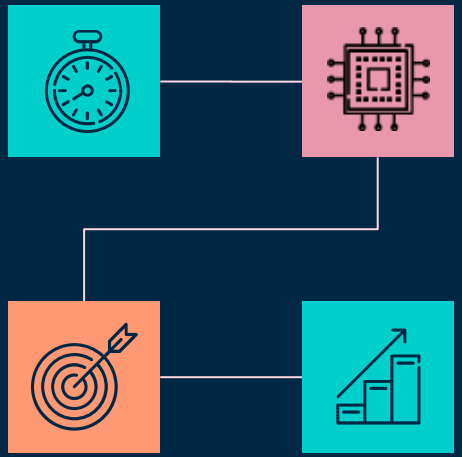
Another problem we will be addressing here is the vizualization of the processed data at near real-time to the user.

REVATURE

Our Solution

Data Source

The stream data coming from the source



Processing

The raw data is filtered to make it structured and usable

Filtering

Getting only the data that we to work on from the processed data

Visualizing

The filtered data is visualized at near real-time

Major Tech Stacks

Python
High level programming language

Spark Streaming
Processing real time data

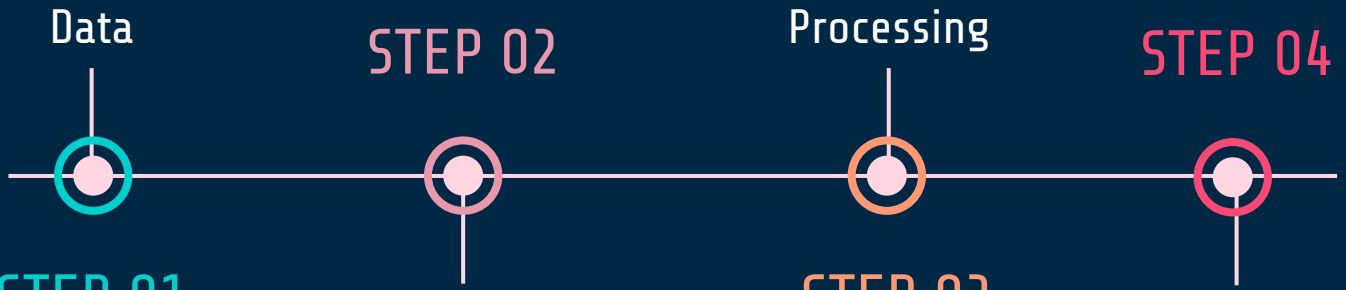
Kafka
Framework for event streaming data

Matplotlib
For plotting graph

Getting the project up and running

- Frist of all install all the tech stack for the project.
- Then start the Zookeeper server using the zookeeper-server-start.bat file its configuration file.
- Then start the Kafka server using the kafka-server-start.bat file its configuration file.
- After that we will execute the producer program.
- Once that's done, do spark-submit of the consumer program.
- In the end, run the output program.

Our Process



STEP 01

Data

Searched for the data source and finalized on weather data from www.weatherbit.io

STEP 02

Data ingestion

Took historical data and simulated into streaming data and published into Kafka weather topic and performed debugging for the errors

STEP 03

Processing

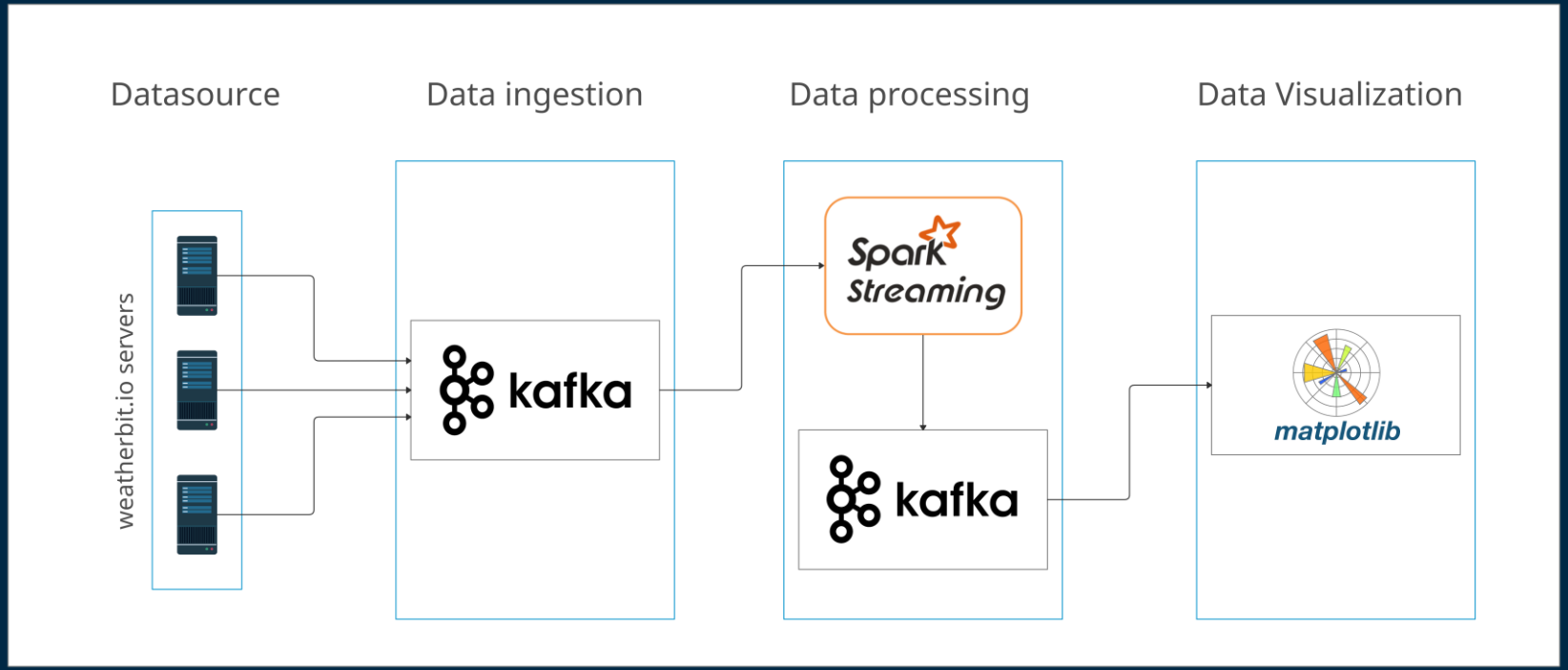
Process data using spark structure streaming and put it in the output topic and performed debugging for the errors

STEP 04

Map Plotting

With the help of matplotlib ,plotted the graph at near real time and performed debugging for the errors

Working of the Project



CODE WALKTHROUGH

REVATURE

Data injestion and Publishing to Kafka Topic

```
producer = KafkaProducer(value_serializer = lambda x: str(x).encode("utf-8"), bootstrap_servers =
['localhost:9092'])


def send_Mumbai():
    url = 'https://api.weatherbit.io/v2.0/history/subhourly?&city=Mumbai&country=IN&start_date=2021-09-
18&end_date=2021-09-22&key={}'.format(key)
    data = requests.get(url).json()
    city_name = data['city_name']

    for dataval in data['data']:
        dataval["city"] = city_name
        del dataval["weather"]
        del dataval["timestamp_utc"]
        del dataval["timestamp_local"]
        output = json.dumps(dataval)
        producer.send('weather', output)
        producer.flush()
        time.sleep(4)

threading.Thread(target=send_Mumbai).start()
```

Creating Spark Session and consuming data from Kafka

```
spark = SparkSession \
    .builder \
    .appName("weatherData") \
    .getOrCreate()

incoming_df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe","weather") \
    .option("startingOffsets", "latest") \
    .load()
```

Converting raw data to structured data with schema

```
schema = StructType().add("precip_rate", IntegerType()).add("rh", IntegerType()) \
    .add("wind_spd", FloatType()).add("snow_rate", IntegerType()).add("app_temp", FloatType()) \
    .add("pres", FloatType()).add("azimuth", FloatType()).add("dewpt", FloatType()).add("uv",
FloatType()).add("elev_angle", FloatType()) \
    .add("wind_dir", IntegerType()).add("ghi", FloatType()).add("dhi", FloatType()) \
    .add("solar_rad", FloatType()).add("vis", IntegerType()).add("dni", FloatType()).add("temp",
FloatType()).add("slp", FloatType()) \
    .add("clouds", IntegerType()).add("ts", IntegerType()).add("city", StringType())

raw_df = incoming_df.selectExpr("CAST(value AS STRING)")

s_df = raw_df.select(from_json(raw_df.value, schema).alias("data"))
```

Filtering data and publishing output to kafka topic

```
weather_df =
s_df.select(col("data.temp"),col("data.app_temp"),col("data.pres"),col("data.rh"),col("data.uv"),col("d
ata.city"),col("data.ts"))

output_df = weather_df.select(to_json(struct(col("*"))).alias("value"))

query = output_df.writeStream.format("kafka").option("kafka.bootstrap.servers",
"localhost:9092").option("checkpointLocation", "/tmp/spark_checkpoint").option("topic",
"output").outputMode("update").start()

query.awaitTermination()
```

Consuming from Kafka topic and preparing data for plotting

```
def plot():
    count = 0
    for message in consumer:
        if(message.value['city'] == city_name):
            plt_val_x.append(int(count))
            count += 5
            plt_val_y.append(float(message.value["temp"]))
            plt_val_y2.append(float(message.value["app_temp"]))
            plt_val_y3.append(float(message.value["pres"]))
            plt_val_y4.append(float(message.value["rh"]))

consumer = KafkaConsumer('output',bootstrap_servers=['localhost:9092'], value_deserializer=lambda x:
json.loads(x.decode('utf-8')))

plot_thread = threading.Thread(target=plot)
plot_thread.start()
```

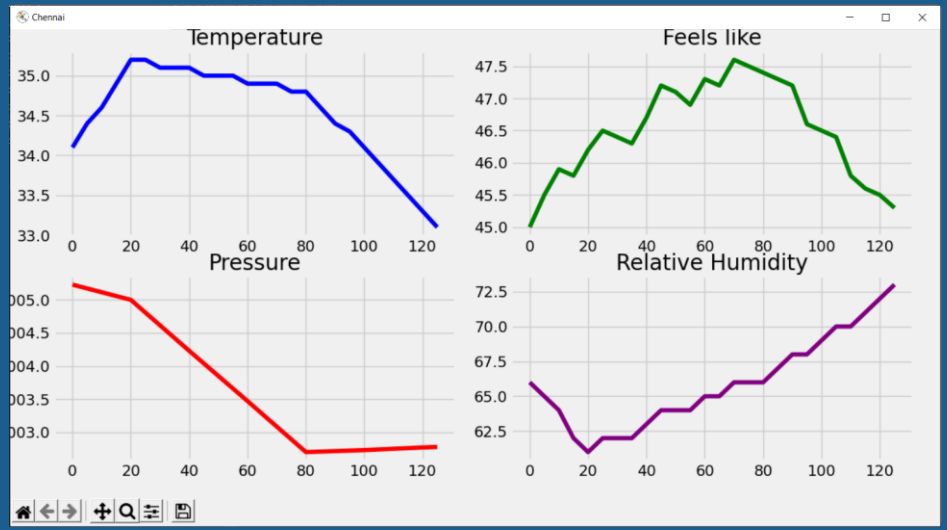
Plotting data as graph at near real-time with matplotlib

```
def animate(i):
    global plt_val_x, plt_val_y

    axs[0, 0].cla()
    axs[0, 1].cla()
    axs[1, 0].cla()
    axs[1, 1].cla()
    axs[0, 0].plot(plt_val_x, plt_val_y,color='blue')
    axs[0, 0].set_title('Temperature')
    axs[0, 1].plot(plt_val_x, plt_val_y2, color='green')
    axs[0, 1].set_title('Feels like')
    axs[1, 0].plot(plt_val_x, plt_val_y3, color = 'red')
    axs[1, 0].set_title('Pressure')
    axs[1, 1].plot(plt_val_x, plt_val_y4, color = 'purple')
    axs[1, 1].set_title('Relative Humidity')

fig, axs = plt.subplots(2, 2, figsize=(12,6), num = city_name)
ani = FuncAnimation(plt.gcf(), animate, interval=1000)
plt.tight_layout()
plt.show()
```

Prerequisites for running the project

```
zookeeper-server-start.bat C:\kafka\config\zookeeper.properties

kafka-server-start.bat C:\kafka\config\server.properties

kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic
weather

kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic
output

python producer.py

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2 consumer.py

python output.py
```

Output (Graph)



OUTCOMES & CONCLUSION

REVATURE

Challenges

- Difficulties in finding the real time data source.
- Faced problems while setting up the dependencies of the project like Kafka and Spark
- Was unable to get the spark structure streaming to perform the operations in the proper way.
- Spark submit failed due to missing package to integrate Kafka with Spark.
- The graph was not visible while plotting .
- The graph didn’t update with the real time data.

Scope

IoT Data

We can use this project to process the stream data coming from IoT devices like sensors.

Other real time processing

With minor modification we can use for processing real time data like stock market data

Future Developments

Change data source

Change the data source to an actual real time data rather than simulation.

Create dashboard

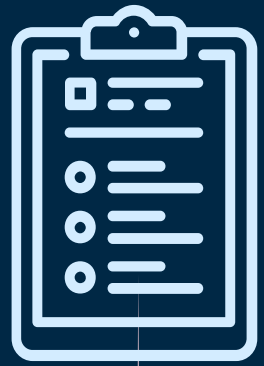
Create the dashboard to display the real time data.

REVATURE

Conclusion

- The raw data can be inconsistent which cannot be used directly for processing. So we need to filter the raw data by applying the necessary operations on the raw data. This is were Spark Structured Streaming come into play which can help with processing of the raw data.

- Displaying the processed data in near real-time is a task that can be achieved in multiple ways. The basic idea is to continuously listen to the incoming data and update the visualization accordingly. This is were the animation module of Matplotlib was used for real-time visualization.



Thank You !