Experiment 5 – 5. Design and Deploy simple strom topology

import random

import time

class SentenceSpout:

    def \_\_init\_\_(self):

        self.sentences = [

            "Storm is great for real-time processing",

            "Apache Storm processes unbounded streams"

        ]

    def next\_tuple(self):

        time.sleep(0.1)

        sentence = random.choice(self.sentences)

        return [sentence]  # Emitting the sentence as a tuple

class WordCountBolt:

    def \_\_init\_\_(self):

        self.counts = {}

    def process(self, tup):

        sentence = tup[0]  # Since we're emitting the sentence as a tuple

        words = sentence.split(" ")

        for word in words:

            self.counts[word] = self.counts.get(word, 0) + 1

        print("Current Word Counts:", self.counts)

class ManualTopology:

    def \_\_init\_\_(self):

        self.spout = SentenceSpout()

        self.bolt = WordCountBolt()

    def run(self, iterations=10):

        # Run the spout and bolt simulation for a fixed number of iterations

        for \_ in range(iterations):

            # Emit a sentence from the spout

            emitted\_sentence = self.spout.next\_tuple()

            # Process the emitted sentence with the bolt

            self.bolt.process(emitted\_sentence)

# Create and run the topology

topology = ManualTopology()

topology.run()

**experiment 7 Implement mechanisms for model updates in response to streaming data changes.**

import numpy as np

from sklearn.linear\_model import SGDRegressor

import matplotlib.pyplot as plt

from collections import deque

# Generate streaming data

def generate\_streaming\_data():

    while True:

        for i in range(1000):

            X = np.random.normal(loc=0.0, scale=1.0, size=(1, 1))

            y = X + np.random.normal(loc=0.0, scale=0.1, size=(1, 1))

            yield X[0, 0], y[0, 0]

        for i in range(1000):

            X = np.random.normal(loc=5.0, scale=1.0, size=(1, 1))

            y = X + np.random.normal(loc=0.0, scale=0.1, size=(1, 1))

            yield X[0, 0], y[0, 0]

# Initialize model with initial data

X\_init, y\_init = np.random.normal(0, 1, (100, 1)), np.random.normal(0, 0.1, (100, 1))

model = SGDRegressor(max\_iter=1000, tol=1e-3).fit(X\_init, y\_init.ravel())

# Process streaming data

performance\_log = deque(maxlen=100)

data\_stream = generate\_streaming\_data()

X\_batch, y\_batch = [], []

for i, (X\_new, y\_new) in enumerate(data\_stream):

    y\_pred = model.predict([[X\_new]])

    performance\_log.append(abs(y\_pred[0] - y\_new))

    X\_batch.append([X\_new])

    y\_batch.append(y\_new)

    if (i + 1) % 100 == 0:

        model.partial\_fit(np.array(X\_batch), np.array(y\_batch).ravel())

        X\_batch, y\_batch = [], []  # Clear batch after update

    if (i + 1) % 500 == 0:

        print(f"Sample {i+1}, Avg Error: {np.mean(performance\_log):.4f}")

    if i >= 5000:  # Stop after 5000 samples

        break

# Plot performance

plt.plot(performance\_log)

plt.xlabel('Sample Number')

plt.ylabel('Absolute Prediction Error')

plt.title('Model Performance Over Time')

plt.grid(True)

plt.show()

**exp 6 :Develop a real-time analytics application with a simple machine learning model.**

*import random*

*import time*

*from sklearn.linear\_model import LinearRegression*

*import joblib*

*# Simulated training data*

*X\_train = [[20], [21], [22], [23], [24], [25], [26], [27]]*

*y\_train = [20, 21, 22, 23, 24, 25, 26, 27]*

*# Train the model*

*model = LinearRegression().fit(X\_train, y\_train)*

*# Save the model*

*joblib.dump(model, 'temperature\_predictor.pkl')*

*# Temperature data spout*

*def temperature\_spout():*

*while True:*

*yield round(random.uniform(20.0, 40.0), 2)*

*time.sleep(1)*

*# Prediction function (bolt)*

*def predict\_bolt(temp):*

*prediction = model.predict([[temp]])[0]*

*print(f"Temperature: {temp}, Predicted: {prediction}")*

*return prediction*

*# Real-time processing (simulate for 5 iterations)*

*for \_ in range(5):*

*temp = next(temperature\_spout()) # Get temperature from spout*

*predict\_bolt(temp) # Predict using the bolt*

**exp- 2 producer consumer using kafka**

It seems you're running on Windows, and Kafka's scripts for Linux (.sh files) aren't directly compatible with Windows. For Kafka to work on Windows, you need to use the .bat scripts instead of the .sh files.

**Steps for Running Kafka on Windows:**

**1. Start Zookeeper**

Kafka on Windows provides a batch script for starting Zookeeper. Instead of the Linux zookeeper-server-start.sh script, you should use zookeeper-server-start.bat:

Navigate to the Kafka bin/windows directory:

bash

Copy code

**cd C:\kafka\_2.12-3.8.0\bin\windows**

Then run the Zookeeper script:

bash

Copy code

**zookeeper-server-start.bat ..\..\config\zookeeper.properties**

This should start Zookeeper and print logs indicating it’s running.

**2. Start Kafka Broker**

Once Zookeeper is running, you can start the Kafka broker. Kafka on Windows also provides a batch script for starting the broker (kafka-server-start.bat).

In the same bin/windows directory, run:

**kafka-server-start.bat ..\..\config\server.properties**

This will start the Kafka broker and print logs showing the broker is active.

**3. Create a Kafka Topic**

You can create a topic with a batch script as well. In the same bin/windows directory, run the following command to create the topic:

bash

Copy code

**kafka-topics.bat --create --topic my\_topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1**

**Run Producer and Consumer**:

bash

Copy code

**python producer.py**

**python consumer.py**

from kafka import KafkaProducer

import json

import time

*# Function to serialize data to JSON*

def json\_serializer(data):

    return json.dumps(data).encode("utf-8")

*# Initialize Kafka producer*

producer = KafkaProducer(

    bootstrap\_servers=['localhost:9092'],

    value\_serializer=json\_serializer

)

*# Send messages to Kafka*

for i in range(10):

    message = {"number": i, "text": f"Message {i}"}

    producer.send('my\_topic', message)

    print(f"Sent: {message}")

    time.sleep(1)  *# simulate delay*

*# Finalize producer and send any remaining messages*

producer.flush()

producer.close()

from kafka import KafkaConsumer

import json

*# Initialize Kafka consumer*

consumer = KafkaConsumer(

    'my\_topic',

    bootstrap\_servers=['localhost:9092'],

    auto\_offset\_reset='earliest',  *# start from the beginning of the topic*

    enable\_auto\_commit=True,

    group\_id='my-group',

    value\_deserializer=lambda x: json.loads(x.decode('utf-8'))

)

*# Continuously read messages from Kafka*

print("Consumer started. Reading messages from Kafka:")

for message in consumer:

    print(f"Received: {message.value}")

1. Install the given java file by double clicking on it

2. Assuming that the kafka file is extracted and stored in downloads folder as below

C:\Users\USER-PC\Downloads\kafka\bin\windows

3. Copy the above address and press enter. Now you will be in above folder then select above address and type as cmd

4. Now a command prompt will be opened and paste the following command then press enter

zookeeper-server-start.bat ..\..\config\zookeeper.properties

5. Again open fresh file explorer as C:\Users\USER-PC\Downloads\kafka\bin\windows and paste the following command then press enter

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

6.Again open fresh file explorer as C:\Users\USER-PC\Downloads\kafka\bin\windows and paste the following command then press enter

kafka-topics.bat --create --topic my-topic --bootstrap-server localhost:9092 --replication-factor 1 --partitions 3

7.Again open fresh file explorer as C:\Users\USER-PC\Downloads\kafka\bin\windows and paste the following command then press enter

kafka-console-producer.bat --broker-list localhost:9092 --topic my-topic

8. Again open fresh file explorer as C:\Users\USER-PC\Downloads\kafka\bin\windows and paste the following command then press enter

kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic my-topic --from-beginning

9. Now type any statement in the command prompt which was opened in step7 now the message will be passed to the command prompt in opened in step8

ex2 producer code

from kafka import KafkaProducer

import json

import time

# Function to serialize data to JSON

def json\_serializer(data):

return json.dumps(data).encode("utf-8")

# Initialize Kafka producer

producer = KafkaProducer(

bootstrap\_servers=['localhost:9092'],

value\_serializer=json\_serializer

)

print("Enter messages to send to Kafka. Type 'exit' to stop.")

# Send custom messages to Kafka

while True:

text = input("Enter message text: ")

if text.lower() == 'exit':

break

message = {"text": text}

producer.send('my\_topic', message)

print(f"Sent: {message}")

time.sleep(1) # simulate delay if needed

# Finalize producer and send any remaining messages

producer.flush()

producer.close()

**Exp – 1 Implement program using apache flink**

In c:

**mvn archetype:generate -DgroupId=com.example.WordCount -DartifactId=WordCount -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false**

Go to the C:/WordCount/pom.xml and Right click and select edit

**Paste the pom.xml**

Step 26: Open the java file and to code it , Delete everything in the file and Copy all and

**Rename file as WordCount.java**

**mvn clean package**

WordCount.java

package com.example.WordCount;

import org.apache.flink.api.common.functions.FlatMapFunction;

import org.apache.flink.api.java.tuple.Tuple2;

import org.apache.flink.streaming.api.datastream.DataStream;

import

org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;

import org.apache.flink.util.Collector;

public class SocketWordCount {

  public static void main(String[] args) throws Exception {

    final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

*// Connect to the socket server on localhost:9999*

    DataStream < String > text = env.socketTextStream("localhost", 9999);

*// Parse the data, group by word, and aggregate the counts*

    DataStream < Tuple2 < String,

    Integer >> counts = text.flatMap(new Tokenizer()).keyBy(value - >value.f0).sum(1);

*// Print the result to the console*

    counts.print();

*// Execute the program*

    env.execute("WordCount Example");

  }

*// User-defined function to split lines into words*

  public static final class Tokenizer implements FlatMapFunction < String,

  Tuple2 < String,

  Integer >> {@Override

    public void flatMap(String value, Collector < Tuple2 < String, Integer >> out) {

*// Normalize and split the line into words*

      String[] tokens = value.toLowerCase().split("\\W+");

*// Emit the words with a count of 1*

      for (String token: tokens) {

        if (token.length() > 0) {

          out.collect(new Tuple2 < >(token, 1));

        }

      }

    }

  }

}

Pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example.WordCount</groupId>

<artifactId>WordCount</artifactId>

<packaging>jar</packaging>

<version>1.0-SNAPSHOT</version>

<name>WordCount</name>

<url>http://maven.apache.org</url>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.apache.flink</groupId>

<artifactId>flink-java</artifactId>

<version>1.18.0</version>

</dependency>

<dependency>

<groupId>org.apache.flink</groupId>

<artifactId>flink-streaming-java</artifactId>

<version>1.19.1</version>

*<!--  Update to match your*

*Flink version  -->*

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<version>3.2.0</version>

<configuration>

<archive>

<manifest>

<addClasspath>true</addClasspath>

<mainClass>com.example.WordCount.WordCount</mainClass>

</manifest>

</archive>

</configuration>

</plugin>

</plugins>

</build>

</project>

 **Navigate to the Kafka Directory**:

bash

Copy code

cd kafka\_2.12-3.8.0

 **Start Zookeeper**: Kafka requires ZooKeeper, so start it first. Open a new Command Prompt in the Kafka directory and run:

bash

Copy code

.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

This will start ZooKeeper on the default port (2181).

 **Start Kafka Broker**: In a new Command Prompt window (still in the Kafka directory), start the Kafka broker:

bash

Copy code

.\bin\windows\kafka-server-start.bat .\config\server.properties

This starts the Kafka broker on the default port (9092).

 **Create a Topic**: Once Kafka is running, you can create a topic named flink-input. In a new Command Prompt:

.\bin\windows\kafka-topics.bat --create --topic flink-input --bootstrap-server localhost:9092 --parti

C:\>mvn archetype:generate -DgroupId=com.example.FlinkKafkaApp -DartifactId=flink-kafka-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

C:\flink-kafka-app>mvn clean package

**Lab 3 – implementing flink streaming app**

**Rem…..code as same as first only execute jar file**

taskmanager.resource-id : taskmt #rem at end of confg.yml

1. Open Cywgin terminal1

cd C:

export SHELLOPTS

set -o igncr

Start apache cluster

./flink/bin/start-cluster.sh

Check if it is opened

jps

2. Open Terminal2 (don't close 1)

nc -lk 9999

3. Execute jar file from Terminal 1

Paste jar file in C: drive

./flink/bin/flink run ./WordCount-1.0-SNAPSHOT.jar

4. Go back to terminal2

Type any message

5. Open terminal3

Go to cgdrive/c/flink

tail -f ./log/flink-\*-taskexecutor\*.out

Note in terminal3 it is not ERROR , just keep writing msgs in T2 and see results in T3

localhost:8081

ex 4 **Explore and connect flink application to kafka for Real time data ingestion.**

package com.example.FlinkKafkaApp;

import org.apache.flink.api.common.functions.FlatMapFunction;

import org.apache.flink.api.java.tuple.Tuple2;

import org.apache.flink.streaming.api.datastream.DataStream;

import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;

import org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer;

import org.apache.flink.util.Collector;

import org.apache.kafka.clients.consumer.ConsumerConfig;

import org.apache.kafka.common.serialization.StringDeserializer;

import org.apache.flink.api.common.serialization.SimpleStringSchema;

import java.util.Properties;

public class KafkaWordCount {

public static void main(String[] args) throws Exception {

// Set up the execution environment

final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

// Set Kafka properties

Properties properties = new Properties();

properties.setProperty(ConsumerConfig.BOOTSTRAP\_SERVERS\_CONFIG, "localhost:9092");

properties.setProperty(ConsumerConfig.GROUP\_ID\_CONFIG, "flink-group");

properties.setProperty(ConsumerConfig.KEY\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class.getName());

properties.setProperty(ConsumerConfig.VALUE\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class.getName());

properties.setProperty(ConsumerConfig.AUTO\_OFFSET\_RESET\_CONFIG, "earliest");

// Create a Kafka consumer

FlinkKafkaConsumer<String> kafkaConsumer = new FlinkKafkaConsumer<>(

"flink-input",

new SimpleStringSchema(),

properties

);

// Add Kafka source to Flink

DataStream<String> stream = env.addSource(kafkaConsumer);

// Parse the data, group by word, and aggregate the counts

DataStream<Tuple2<String, Integer>> counts = stream

.flatMap(new Tokenizer())

.keyBy(value -> value.f0)

.sum(1);

// Print the result to the console

counts.print();

// Execute the program

env.execute("Kafka WordCount Example");

}

// User-defined function to split lines into words

public static final class Tokenizer implements FlatMapFunction<String, Tuple2<String, Integer>> {

@Override

public void flatMap(String value, Collector<Tuple2<String, Integer>> out) {

// Normalize and split the line into words

String[] tokens = value.toLowerCase().split("\\W+");

// Emit the words with a count of 1

for (String token : tokens) {

if (token.length() > 0) {

out.collect(new Tuple2<>(token, 1));

}

}

}

}

}