Spark and Kafka Cluster Setup Guide

This document outlines the step-by-step process for setting up a Spark master-worker cluster integrated with Apache Kafka for real-time data streaming. It includes setup commands, test scripts, troubleshooting tips, and common configurations.

# 1. Setting Up the Spark Master Node

On the master node, start the Spark master service using the following script from the Spark 'sbin' directory:

./start-master.sh

After starting, you can access the Spark master web UI at:

http://10.5.6.240:8080/

To interact with the Spark cluster using PySpark:

pyspark --master spark://10.5.6.240:7077

# 2. Connecting a Worker Node to the Cluster

On each worker node, start a worker process and point it to the master using the following:

./start-worker.sh spark://10.5.6.240:7077

# 3. Testing the Spark Cluster with PySpark

Run the following Python code using PySpark to verify the distributed computation works:

from pyspark.sql import SparkSession  
  
spark = SparkSession.builder.appName("ClusterTest").getOrCreate()  
data = spark.sparkContext.parallelize(range(1, 1000000), numSlices=3)  
result = data.sum()  
print(f"🔥 The sum is: {result}")  
spark.stop()

# 4. Setting Up Apache Kafka

Navigate to the Kafka installation directory and run the following:

Start Zookeeper:

bin/zookeeper-server-start.sh config/zookeeper.properties

Start Kafka server:

bin/kafka-server-start.sh config/server.properties

Create a Kafka topic:

bin/kafka-topics.sh --create --topic trafi --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

To verify or interact with topics:

bin/kafka-topics.sh --list --bootstrap-server localhost:9092  
bin/kafka-console-producer.sh --topic test-topic --bootstrap-server localhost:9092  
bin/kafka-console-consumer.sh --topic test-topic --from-beginning --bootstrap-server localhost:9092

# 5. Kafka Troubleshooting Tips

If Kafka refuses to start due to a lock or ID issue, try the following:

bin/kafka-server-stop.sh  
bin/zookeeper-server-stop.sh  
sudo rm -rf /tmp/kafka-logs  
sudo bash -c "echo 1 > /var/lib/zookeeper/myid"

Ensure Zookeeper has correct ownership and directories:

sudo chown -R sem6:sem6 /var/lib/zookeeper  
sudo mkdir -p /var/lib/zookeeper/version-2

# 6. Integrating PySpark with Kafka

To enable Spark to consume Kafka data, launch PySpark with the necessary Kafka connector jars:

pyspark --master spark://10.5.6.240:7077 \  
 --jars /opt/spark/jars/spark-sql-kafka-0-10\_2.12-3.5.4.jar,/opt/spark/jars/kafka-clients-3.9.0.jar

Then, use this script to read streaming data from Kafka:

from pyspark.sql import SparkSession  
  
spark = SparkSession.builder \  
 .appName("KafkaTest") \  
 .config("spark.jars", "/opt/spark/jars/spark-sql-kafka-0-10\_2.12-3.5.4.jar,/opt/spark/jars/kafka-clients-3.5.0.jar") \  
 .config("spark.driver.extraClassPath", "/opt/spark/jars/...") \  
 .config("spark.executor.extraClassPath", "/opt/spark/jars/...") \  
 .getOrCreate()  
  
df = spark.readStream \  
 .format("kafka") \  
 .option("kafka.bootstrap.servers", "10.5.6.240:9092") \  
 .option("subscribe", "traffic-data") \  
 .load()  
  
df.printSchema()

# 7. Fixing Kafka-Spark Version Compatibility

If version mismatches occur, remove conflicting jars and download compatible ones:

rm $SPARK\_HOME/jars/spark-sql-kafka-2.13\*.jar  
wget https://repo1.maven.org/maven2/org/apache/kafka/kafka-clients/3.9.0/kafka-clients-3.9.0.jar

# 8. Submitting a Spark Streaming Job

spark-submit --master local[\*] \  
 --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.5.4 \  
 kafka\_spark\_streaming.py

# 9. Installing Confluent Kafka Python Client

sudo apt update && sudo apt install -y librdkafka-dev  
pip install --no-binary confluent-kafka confluent-kafka  
python -c "import confluent\_kafka; print('Kafka is working!')"

# 10. Project Files in Use

Final working files for the project include:

- prod2.py  
- dashpromax.py  
- Dataset - traffic\_data.csv