Day6

Streaming Data PySpark-Kafka

Code Repository at 6_Spark/PySpark-Kafka

Demo 1

Prerequiresites

- start Hive
- start zookeeper
- start Kafka

```
pyspark --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 --master
local[2] #start pyspark shell
```

```
# pyspark shell
kafka-topics.sh --create --zookeeper 127.0.0.1:2181 --replication-factor 1 --
partitions 1 --topic voters

kafkaVoterDF =
spark.readStream.format("kafka").option("kafka.bootstrap.servers",
"127.0.0.1:9092").option("subscribe","voters").option("startingOffsets",
"earliest").load()

rawVoterQuery = kafkaVoterDF.writeStream.trigger(processingTime='10
seconds').outputMode("append").format("console").start()

# Stop pyspark cell first
stringVoterQuery.stop()
```

```
# Terminal 2
./pushKafkaData.sh #Terminates Ctrl + C
```

Demo2

```
kafkaVoterDF =
spark.readStream.format("kafka").option("kafka.bootstrap.servers",
```

```
"127.0.0.1:9092").option("subscribe", "voters").option("startingOffsets",
"earliest").load()

voterStringDF = kafkaVoterDF.selectExpr("CAST(value AS STRING)")

stringVoterQuery = voterStringDF.writeStream.trigger(processingTime='10
seconds').outputMode("append").format("console").option("truncate",
"false").start()

stringVoterQuery.stop()
```

```
# Terminal 2
./pushKafkaData.sh #Terminates Ctrl + C
```

Demo3

```
kafkaVoterDF =
spark.readStream.format("kafka").option("kafka.bootstrap.servers",
"127.0.0.1:9092").option("subscribe", "voters").option("startingOffsets",
"earliest").load()

from pyspark.sql.types import StringType, LongType, StructType, StructField

voterSchema = StructType([StructField("gender",StringType(), True),
StructField("age",LongType(), True), StructField("party",StringType(), True)])

from pyspark.sql.functions import from_json

voterStatsDF =
kafkaVoterDF.select(from_json(kafkaVoterDF["value"].cast(StringType()),
voterSchema).alias("voterJSON")).groupBy("voterJSON.gender",
"voterJSON.party").count()

voterStatsQuery = voterStatsDF.writeStream.trigger(processingTime='1
minute').outputMode("complete").format("console").start()

voterStatsQuery.stop()
```

```
# Terminal 2
./pushKafkaData.sh #Terminates Ctrl + C
```

Lab 26 - Pyspark working with MySql

```
mysql -u bigdata -p
```

```
use test;
```

```
create table salaries ( gender varchar(1), age int, salary double, zipcode
int);

load data local infile '/tmp/salaries.txt' into table salaries fields
terminated by ',';

ALTER table salaries add column `id` int(10) unsigned primary KEY
AUTO_INCREMENT;

DESCRIBE salaries;

SELECT * FROM salaries LIMIT 5;
```

```
url = "jdbc:mysql://127.0.0.1:3306/test?
useSSL=false&allowPublicKeyRetrieval=true"
driver = "com.mysql.jdbc.Driver"
user = "bigdata"
password = "Bigdata@123"

df =  spark.read\
    .format("jdbc")\
    .option("driver", driver)\
    .option("url", url)\
    .option("user", user)\
    .option("password", password)\
    .option("dbtable", "salaries")\
    .load()

# Avg Salary per gender
df.groupBy("gender").agg({"salary": "avg"}).show()
```

PySpark MLlib

3 C of ML in PySpark MLlib
Collaborative filtering
Classification
Clustering

Collaborative filtering

Collaborative filtering is a technique for recommender systems wherein users' ratings and interactions with various products are used to recommend new ones

- User-User Collaborative Filtering
- Item-Item Collaborative Filtering

Lab 18