ID2221 – Data Intensive Computing

Lab3 - Spark Streaming, Kafka and Cassandra

Course Coordinator:
Amir H. Payberah

Óttar Guðmundsson

Xin Ren

2018-10-3

In this lab assignment, we created a simple Spark Streaming application, while reading streaming data from Kafka and storing the result in Cassandra datastore. The streaming data are (key, value) pairs in the form of "String,int", and we calculate the average value of each key and continuously update it, while new pairs arrive. We store the result in Cassandra continuously. The results are in the form of (key, average value) pairs.

How to run

1. Start Kafka and create a topic, named avg

zookeeper-server-start.sh $KAFKA_HOME/config/zookeeper.properties

kafka-server-start.sh $KAFKA_HOME/config/server.properties

kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic avg

2. Start Cassandra

cassandra -f

3. Compile and run KafkaSpark.scala to read streaming data from Kafka, process them and store the result in Cassandra datastore.

4. Compile and run Producer.scala to generate a streaming input (pairs of "String,int") and feed them to Kafka

5. Start the cqlsh prompt

$CASSANDRA_HOME/bin/cqlsh

6. Check the result:

use avg_space;

*select * from avg;*

KafkaSpark.scala

In the main function of object KafkaSpark, we first create a new SparkConf and then a new StreamingContext with the SparkConf. We define the StreamingContext so that the stream is divided to 1s time interval batches. Then we set the checkpoint path for the StreamingContext to the current directory.

We then connect to Cassandra with contact point "127.0.0.1" which is the localhost as Cassandra is running on the same machine. This gives us a Cassandra "session". We create keyspace avg_space and a table called avg in the keyspace with session.execute() commands. The table has a key in text and a value in float.

We then connect to Kafka with command: val messages = KafkaUtils.createDirectStream[String, String, StringDecoder, StringDecoder](ssc, kafkaConf, Set("avg")). Since records produced by Producer.scala have key and value both in String, we define key and value classes here as String and decoders as StringDecoder. "ssc" is the StreamingContext we created previously. "kafkaConf" is the configuration map we use to connect to Kafka that defines metadata broker and zookeeper addresses, consumer group id and zookeeper connection timeout. Set("avg") is the topics set.

We then process the “messages” from Kafka and generate key value “pairs”. We generate a stateful Dstream from pairs with following command:

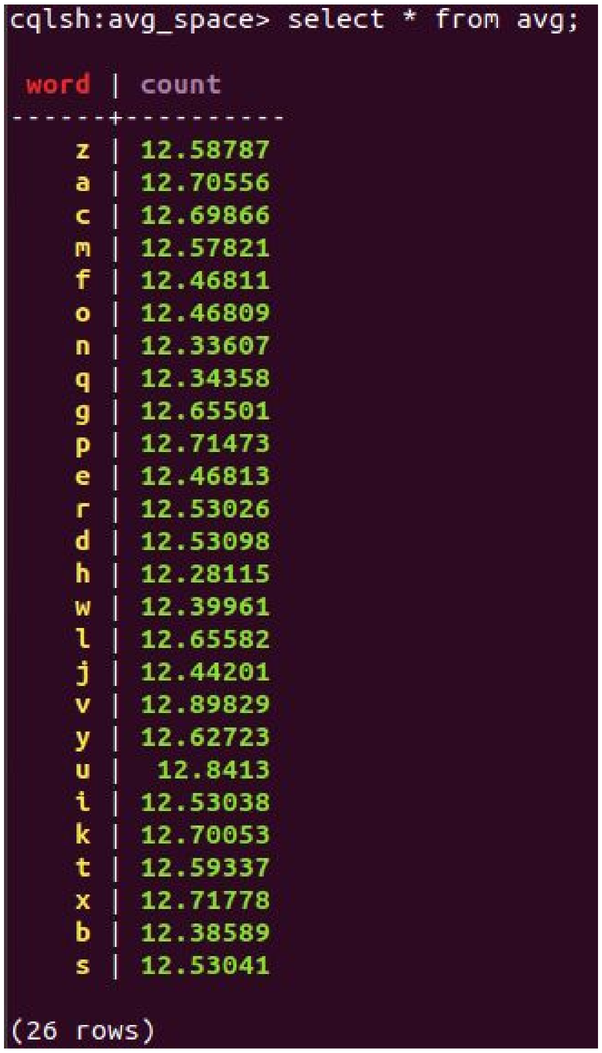
val stateDstream = pairs.mapWithState(StateSpec.function(mappingFunc _)).

mappingFunc is defined in the way that for every key value in pairs, the state records for each key how many times the key has shown up and the sum of the values of the key, and then return the key and average value pair.

In the end, we store the Dstream to table avg in avg_space on Cassandra with function saveToCassandra. We start the StreamingContext and then wait for termination.

Results

The result is changing all the time with the input coming all the time. This is the result for one check.



```
cqlsh:avg_space> select * from avg;

 word | count
------+----------
    z | 12.58787
    a | 12.70556
    c | 12.69866
    m | 12.57821
    f | 12.46811
    o | 12.46809
    n | 12.33607
    q | 12.34358
    g | 12.65501
    p | 12.71473
    e | 12.46813
    r | 12.53026
    d | 12.53098
    h | 12.28115
    w | 12.39961
    l | 12.65582
    j | 12.44201
    v | 12.89829
    y | 12.62723
    u |  12.8413
    i | 12.53038
    k | 12.70053
    t | 12.59337
    x | 12.71778
    b | 12.38589
    s | 12.53041

(26 rows)
```