Data Intensive Computing

October 12th, 2020

Lab2 Documentation

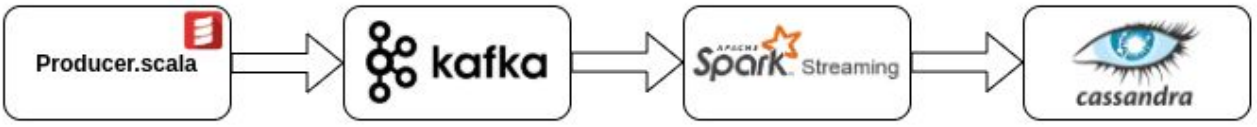
Edoardo Realini 19970701-T491
Riccardo Sommaruga 19970329-T519

1. Introduction

This lab assignment has the goal is to implement a system that does the following operations:

1. Produces key-value tuples in the format `"String, Int"` where the string is a letter of the english alphabet and the value is a random number associated to it. The random integer, generated by `Random.nextInt(alphabet.size)` randomically samples a number between 0 and `alphabet.size` following a *uniform* distribution.
2. The key-value pairs are directly passed to Kafka in the topic called "avg".
3. With SparkStreaming, the data managed by Kafka is loaded from the topic "avg" by creating a DirectStream. Working with Spark we want to extract, for each key, the average value in real time and store it to Cassandra.
4. The last operation is to continuously update the values stored in Cassandra under the keyspace avg_space. The result is a table containing for each letter, the real-time updated average of the encountered values until the observation time.

In the image reported below there is a summary of the pipeline with all the involved systems.



2. Implementation

The main tasks of the assignment were:

1. Setting up the environment by installing all the needed tools, setting up the Kafka topic and the Cassandra keyspace (explained in the instructions section).
2. Implementing the code in `SparkStreaming.kafka` file.

Regarding the *second point*:

1. *(lines 24-25)* create a connection with the already running Cassandra server.
2. *(lines 28-29)* create, if not present, the keyspace in Cassandra and name it "avg_space"; create the table with fields word (text) and count (float).
3. *(lines 32-45)* create a SparkStreaming configuration, a SparkStreaming Context (ssc) on which the checkpoint directory has to be specified (in order to use the mapWithState function). Define the configuration parameters for Kafka (kafkaConf) and the topic_set as Set containing "avg".
4. *(lines 47-49)* create the directStream and map the content into tuples of the format (key, value).
5. *(lines 55-69)* definition of the mapping function needed for executing the mapWithState operation. In this function we consider a state in which the average for a certain key and the number of times that the key has been encountered are stored. These values are used to compute the new updated average value. the function returns the couple (key, new_average).
6. *(line 71)* apply the mapWithState function to the tuples generated at point 4.
7. *(line 74)* store the results in Cassandra under the keypace "avg_space".
8. *(lines 76-77)* start the context and await for the termination.

3. Instructions

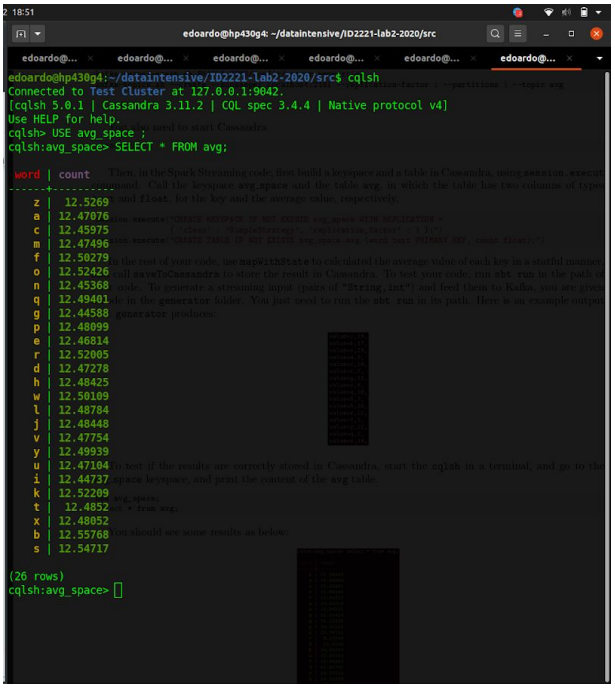
The requirements for the correct execution of the implemented pipeline is to have installed all the involved tools. In order to run the whole pipeline, the following steps shall be accomplished:

1. Run the following commands in order to start Zookeeper, Kafka and enable the creation of the topic "avg" in Kafka:
   a. `zookeeper-server-start.sh $KAFKA_HOME/config/zookeeper.properties`
   b. `kafka-server-start.sh $KAFKA_HOME/config/server.properties`
   c. `kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic avg`
2. Start the Cassandra server with: `cassandra -f`
3. Move to the folder containing the file Producer.scala and run the command: `sbt run`
4. Move to the folder containing the file SparkStreaming.scala and run the command: `sbt run`
5. To check the results run the command cqlsh that opens a CLI for Cassandra.
6. Run the following query in cqlsh to find the results:

```
use avg_space;
select * from avg;
```

4. Results

The results obtained after one minute of execution are reported in the following screenshot:



The fact that all the computed averages become more and more stable around the number 13 is **correct**. This happens since the values are generated using a uniform distribution between 0 and 26 which is the number of letters contained in the english alphabet.