Illustration of Lab 2

A small modification was made to Producer.scala since in a discussion in Canvas, the professor said it was possible, the change made in the producer sends the letters as keys and their random number as values as shown in Figure 1, instead of sending null and letter with its random number as value (e.g.(key=null, value=(a,2)).

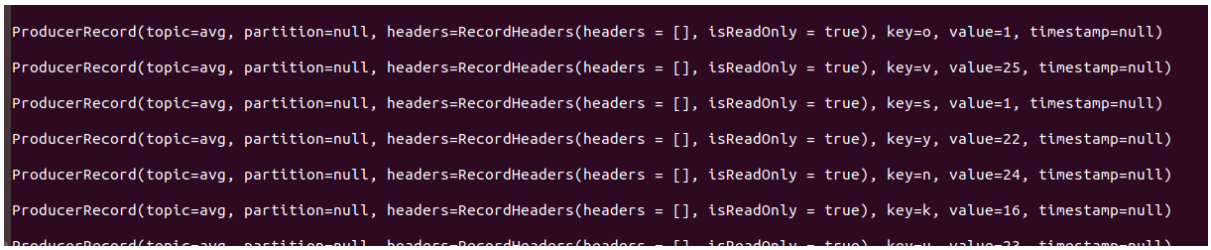


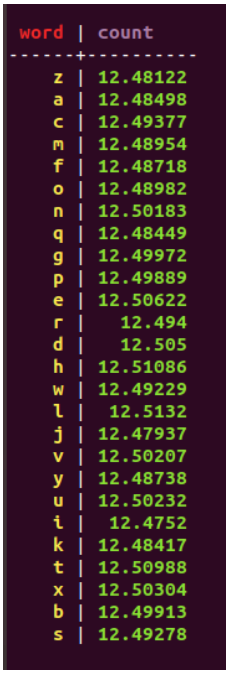
Figure 1. Output of Producer.scala

In KafkaSpark.scala, first we connected Spark to Cassandra, created a keyspace (avg_space) with a replication factor of 1 and a table inside that keyspace (avg) with columns word and count to store the results of the program.

Then, we connected Spark to Kafka, after that, we read the key, value pairs from the direct stream, casting value to double since it was sent as String by the producer. Then we used mapWithState using the function mappingFunc, where we have two states, one to sum all the values of each key (newSum) and the second state will count the number of times each key was sent (newCnt) to obtain the average value for each key in a stateful manner.

Finally we store the results in Cassandra and when checking the table using select * from avg; inside avg_space, the table shown in Figure 2 is the stored table.

Since, the random numbers sent in the producer were numbers between 0 and 25, having an output close to 12.5 for each key is expected.



word	count
z	12.48122
a	12.48498
c	12.49377
m	12.48954
f	12.48718
o	12.48982
n	12.50183
q	12.48449
g	12.49972
p	12.49889
e	12.50622
r	12.494
d	12.505
h	12.51086
w	12.49229
l	12.5132
j	12.47937
v	12.50207
y	12.48738
u	12.50232
i	12.4752
k	12.48417
t	12.50988
x	12.50304
b	12.49913
s	12.49278

Figure 2. Cassandra table avg, in avg_space keyspace of the average value of each key

To run the program just type sbt run for Producer.scala inside the zip file we sent and sbt run to KafkaSark.scala, we also modified some parts of built.sbt of the KafkaSpark file to match it with the versions we installed.