Report

All the codes used in the instance handling the California-tweets part are present in the folder "CA_code".
All the codes used in the instance handling the New-york-tweets part are present in the folder "NY_code".

I have written script for executing the entire steps in the file “scriptt.sh”.
The file “accessing_published_tweets.py” has the code for taking tweets for 10 minutes. This has use of Twitter-API.

I have made 2 instances, one for taking CA tweets, and other for taking NY tweets. “scriptt.sh” of NY instance calls “accessing_published_tweets.py” by giving arguement as NY, so that, only NY tweets are received. Conversely, in CA machine, only CA tweets are received.

Note that all my files were present in cloud instance at the location- "/home/ubuntu/*".

Command to trigger this is-
/usr/bin/python /home/ubuntu/accessing_published_tweets.py CA

The tweets got are redirected in the file “tweets.txt”.

Say the CA instance is ec2-54-210-100-54.compute-1.amazonaws.com,
and the NY instance is ec2-54-180-101-45.compute-1.amazonaws.com.

Zookeeper and kafka server are started on both the instances, using their start commands. They run in background.
Commands for these are-
/home/ubuntu/kafka_2.12-1.1.1/bin/zookeeper-server-start.sh /home/ubuntu/kafka_2.12-1.1.1/config/zookeeper.properties &

/home/ubuntu/kafka_2.12-1.1.1/bin/kafka-server-start.sh /home/ubuntu/kafka_2.12-1.1.1/config/server.properties &

I am using 1 broker on California instance, that is taking in the tweets that are present in tweets.txt

Kafka producer is started on California and NY instances thus to publish tweets to Kafka-
/home/ubuntu/kafka_2.12-1.1.1/bin/kafka-console-producer.sh --broker-list "ec2-54-210-100-54.compute-1.amazonaws.com:9092" --topic "mytopic" < /home/ubuntu/tweets.txt

"mytopic" is name of the topic.

First all the CA tweets will be taken by the broker. After all of them have been fed, can NY tweets enter. So its important to start scriptt.sh of NY, a bit later than that of CA.

Next, spark-streaming is done.

timeout 60 /home/ubuntu/spark-3.0.0-preview2-bin-hadoop2.7/bin/spark-submit /home/ubuntu/stream.jar "ec2-54-210-100-54.compute-1.amazonaws.com:9092" "mygroup" "mytopic"

This is how tweets are consumed.

stream.jar is the jar file that was prepared by the command 'sbt assembly' of the scala code for streaming. The scala code for streaming is present in stream.scala, which is present inside the folder 'project', present in CA_code, and present in src/main/scala, inside 'project'.

After streaming, all the CA_tweets are streamed in the 1st file of Text_files folder, present in CA_code.
This 1st file is renamed to 'save', thus-

```
arr=(/home/ubuntu/Text_files/*/)
mv -v ${arr[0]} /home/ubuntu/Text_files/save
```

Thus, the first file is renamed 'save'.

Now, scala FP-growth requires input file to have words that are comma-separated and sorted.

So, insert.py code is applied on /home/ubuntu/Text_files/save/part-00000, which have the tweets, to clean tweet data, and introduce commas between words in 1 tweet, and sort the words of 1 tweet.
The result of this is put in "/home/ubuntu/comma_save_file-1.txt", which is input file to FP_Growth.

By- tweet_count=$(grep -c "" /home/ubuntu/comma_save_file-1.txt),
the tweet_count/ no of tweets is got.

To get words in tweets that have appeared >=3 times, you need support of 3/(# of tweets).

By-

```
minSupport=$(div 3 $tweet_count)

/home/ubuntu/spark-3.0.0-preview2-bin-hadoop2.7/bin/spark-submit
/home/ubuntu/fp.jar /home/ubuntu/comma_save_file-1.txt --minSupport $minSupport

echo "Tweet_count: $tweet_count" >> /home/ubuntu/fp_output
```

FP Growth is called.

fp.jar is the jar file created by running 'sbt assembly' on the scala code 'FP.scala', which is present in the folder 'fp', of NY_code. Thus, it is used for running FP-growth.

FP.scala has code to put the result of >3 times words, in fp_output, in case of CA machine, and in fp_output_2 in case of NY machine.

Next, I put the key to the CA machine, namely, 'Heyy.pem', in NY instance, and used scp command, to transport the final result of NY machine, i.e, fp_output_2, to the CA machine.
Thus, the CA machine has both outputs of itself, and the NY machine, and we just need to print this.

For this, apache server has been installed in the CA instance.
In /var/www/html,'button.php', 'Function_CA.php' and 'Function_NY.php' have the appropriate codes for displaying the contents of CA and NY, when a button of 'California!' or 'New-York!' is clicked, respectively.

Interface_coud.png has the final interface I created, and after clicking the respective buttons, the FP-output is shown in ‘california_tweets.png’ and ‘new_york_tweets.png’ respectively.