Lab 10

In this lab, we analyze a streaming of tweets to analyze the appearing hashtags and compute the number of occurrences of each of them over a sliding window. Your task consists of extracting the hashtags appearing in the input tweets and computing their occurrences every 10 seconds by considering the last 30 seconds of data.
The stream of tweet data is simulated by uploading, during the execution of your application, a set of files in the input HDFS folder. Each input file contains one tweet per line. Each line of the input files has the following format:

- *userId***\t***text_of_the_tweet*
    - the two fields are separated by a tab

For example, the line

26976263 Gym time!!!! #fitness

means that user **26976263** tweeted the text "**Gym time!!!! #fitness**". The text of this tweet contains also a hashtag: **#fitness**

Ex. 1

Write a Spark streaming application that counts the number of occurrences of each hashtag appearing in the input data streaming. Specifically, every 10 seconds, your application must

- extract the hashtags appearing in the last 30 seconds of the input data stream
- count the number of occurrences of each (extracted) hashtag by considering only the last 30 seconds of data (i.e., the last 30 seconds of data of the input data stream)
- store in the output HDFS folder, sorted by num. of occurrences, the pairs (num. of occurrences , hashtag) related to the last 30 seconds of data
- print on the standard output the first 10 hashtags in terms of number of occurrences, related to the last 30 seconds of data

The application performs the analysis every 10 seconds by considering the last 30 seconds of streaming data (i.e., window length = 30 seconds and sliding interval = 10 seconds).

The input data stream is based on the content of an input HDFS folder in which, during the execution of the application, you will upload files formatted according to the format specified in the first part of this problem specification. Upload the input files one at a time to simulate a stream on data. Specifically:

1. Create the local input folder exampledata_tweets on the local file system of jupyter.polito.it and upload the files you want to use to test your application in that folder
2. Create an empty HDFS folder (the folder associated with the input of your application)
    E.g., hdfs dfs -mkdir input_hdfs_tweets

3. Open another shell in jupyter and run your application by means of spark-submit. Use the following command

   spark-submit --class it.polito.bigdata.spark.SparkDriver --deploy-mode client --master local[*] Lab10Ex1-1.0.0.jar input_HDFS_folder ex_outLab10

4. Copy, one at a time, the input files from the local input folder to the HDFS input folder of your application by using the command line **hdfs**

   E.g., hdfs dfs -put exampledata_tweets /tweets_blockaa input_hdfs_tweets /

5. Pay attention that if a file is already in the input HDFS folder and you copy another version of the same file the system will not consider the new version of the file. Do not use the HUE interface to upload the files in HDFS because otherwise the system will not consider those files.

A set of example input files are available on the web site (exampledata_tweets.zip)

The option --master local[*] is used to specify that the application can use all the cores of the jupyter container. You need at least 2 cores to run a Spark streaming application locally.

Ex. 2

We are interested in implementing a simple alert system that prints on the standard output, and write in the HDFS folder, only “relevant” hashtags. A hashtag is defined as relevant if it occurred at least 100 times in the last 30 seconds.
Extend your application to select only the hashtags that occurred at least 100 times in the last 30 seconds.
Store in the output HDFS folder, sorted by num. of occurrences, the selected pairs (num. of occurrences , hashtag) related to the last 30 seconds of data and print on the standard output the first 10 hashtags in terms of number of occurrences.
Also this application must perform the analysis every 10 seconds by considering the last 30 seconds of streaming data (i.e., window length = 30 seconds and sliding interval = 10 seconds).