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 an 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, based on DStreams, 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 folders, sorted by number of occurrences, the pairs (number 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 a local input folder on the local file system of jupyter.polito.it and upload the files you want to use in that folder
2. Create an empty HDFS folder (the folder associated with the input of your application)
3. 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 input_local_folder/tweets_blockaa input_HDFS_folder/

4. Pay attention that if a file is already in the input HDFS folder and you copy other version of the same file the system will not consider the new version of the file
5. Pay attention to **stop your streaming context after your tests** by invoking
   ssc.stop(stopSparkContext=False)

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

Note. If you run this application locally on your PC:

1. Create the input files in a folder and then copy them in the input folder of your application (one file at a time in order to simulate the stream of data)
2. Copy the files in the input folder of your application by using the command line **cp**
   E.g., cp tweets_blockaa input_folder/
3. Do not use the graphical interface to copy the file in the input folder of your application otherwise the output of your application will be empty
4. If you update the content of a file that is already in the input folder of your application, the updated version of the file will not be considered by the Spark streaming engine

Ex. 2

We are interested in implementing a simple alert system that prints on the standard output 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 print on the standard output, and write in the output HDFS folders, only the hashtags that occurred at least 100 times in the last 30 seconds. The returned hashtags must be sorted by 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).