Lab 9

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.

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

Note: If you execute the application locally, use the following command:
spark-submit --class it.polito.bigdata.spark.SparkDriver --deploy-mode client --master local[*] YouAppliction.jar input_folder prefix_outputfolder 2>log.txt

The option --master local[*] is used to specify that the application can use all the cores of your machine. You need at least 2 cores to run a Spark streaming application.
The last part of the command (i.e., "2>log.txt") is used to redirect errors and log messages on the file log.txt (it is used only for improving the readability of the output printed on the standard output by your application).

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 in order to print on the standard output, and store in the output folder, only the hashtags that occurred at least 100 times in the last 30 seconds. 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).