Lab 1

This introductory lab is composed of a set of tasks. Your final objective is to run your first MapReduce application locally on your PC and then on the BigData@Polito cluster and also write your first MapReduce application. For this goal, you must learn how to import a maven project, compile the source code, produce a jar file, copy your application on a remote machine and submit your application on the BigData@Polito cluster.

Please note:

- To run your application locally, inside Eclipse, Linux or macOS are suggested (Windows users can try to set up their windows environment by following the instructions available on the course web page: ConfigureWindowsEnvironment.pdf or use a virtual machine running Linux)
- To import a maven-based project in Eclipse, write your code, compile your application, produce a jar file and submit/execute your application on the BigData@Polito cluster also Windows can be used

1. Import a maven project in Eclipse and run your application locally with Eclipse

In this task, we will import a maven-based Java project into Eclipse, compile it and run it locally on your PC to analyze the content of the local folder example_data. The imported project is the MapReduce implementation of the word count application.

1. Download from the course web page the zip file Lab1.zip, which contains the MapReduce-based implementation of the word count application and the data folder example_data
    - Use the zip file Lab1Windows.zip if you plan to run your application locally by using Windows
2. Watch the following video to learn how to import the content of Lab1 in Eclipse and run the Lab1 application locally on your PC:
    - Direct link: https://www.dropbox.com/s/niilmhv6k1130dt/01_ImportProject_LocalRun.mp4?dl=0
3. Have a look at the source files and the structure of the project. Run the application by changing the number of reducers (first parameter of the application) and analyze the number of output files with respect to the number of reducers.

2. Run your application on BigData@Polito

The objective of this task is to run your application on the cluster BigData@Polito (that is a Hadoop cluster) on the same data you used in the previous task.

First, you must produce a jar file containing your application and copy it on the local file system of the gateway (a server that is connected with the Hadoop cluster and can be used to submit MapReduce jobs on the Hadoop cluster). Then, you must upload the data you want to analyze in the distributed HDFS file system of Hadoop. Finally, you can run your application on the cluster to analyze the data you stored on the distributed file system.

1. Watch the following video to learn how to create a jar file containing your application by using Eclipse + maven, how to copy data on HDFS and how to submit your application on the cluster BigData@Polito:
    a. Direct link: https://www.dropbox.com/s/65xy3hu9qvqp2oc/02_Jar_ClusterExecution.mp4?dl=0
2. Check the number of mappers (i.e., the number of instances of the mapper class)
    a. You can retrieve the information about the number of mappers (map tasks) and other statistics in the information showed on the terminal during the execution of your application (last part of the showed information).
3. Run the application on the cluster by changing the number of reducers (first parameter of the application) and analyze the content of the output folder of the HDFS file system.

To remove a folder from HDFS, by using the hdfs command line tool, you can use the following command:

hdfs dfs -rm -r <path of the folder you want to delete>

If you need to access the log files associated with the execution of your application by using the command line, use the following commands:

1. To retrieve the log associated with the standard output
    - yarn logs -applicationId <id of your application> -log_files stdout
        - The "id of your application" is printed on the terminal at the beginning of the execution of your application
            - The format of "id of your application" is application_number_number
            - Example of "application id" application_1584304411500_0009
    - The returned result contains one stdout log section for each task (driver, map and reduce tasks)
        - One for the Driver
        - One for each Mapper
        - One for each Reducer
2. To retrieve the log associated with the standard error
    - yarn logs -applicationId <id of your application> -log_files stderr

Test on a bigger file

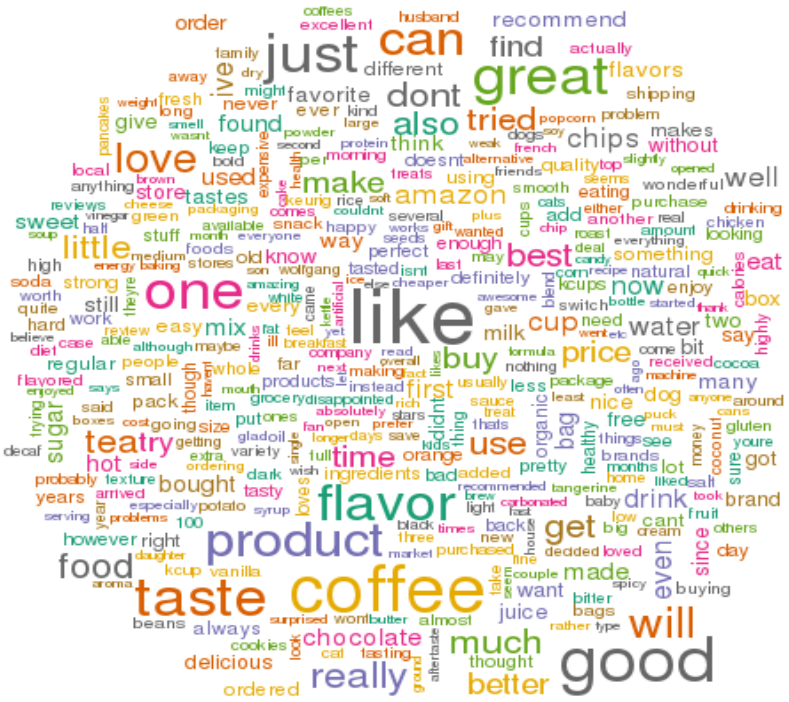
Now you will execute your application on a larger file that we already uploaded on the HDFS file system of BigData@Polito.

The absolute path of that larger file in HDFS is the following:
/data/students/bigdata-01QYD/Lab1/finefoods_text.txt

It is a large collection of Amazon reviews in the food category. Each line of finefoods_text.txt contains one review.

1. Launch your application on the Amazon review file:
    a. Set the second parameter of your application to /data/students/bigdata-01QYD/Lab1/finefoods_text.txt
2. Analyze the results.
    a. Can you understand any interesting facts from your results?
    b. Do you see any space for improvements in your analysis?
3. The following figure was done on a small sample of your data (10000 reviews).

a. Is it consistent with what you found on the complete dataset?
b. Do you think a small sample is enough to represent the whole?



3. Bonus task - Filter an input dataset

In this bonus, you will write by your own a complete Hadoop application.
Start by importing the template project available in Lab1Bonus_Skeleton.zip (Lab1WindowsBonus_Skeleton.zip if you use Windows). Once you have imported the template, modify the content of the classes to implement the application described in the following. The template contains the skeleton of a standard MapReduce application based on three classes: Driver, Mapper, and Reducer. Analyze the problem specification and **decide if you really need all classes** to solve the assigned problem.

From now on, keep in mind always the complexity of your program. Specifically try to understand, before even submitting a job, what will be the effort you will require to the cluster in terms of network

- How many pairs and bytes will be emitted by the mappers and hence how many data will be sent on the network?

If you completed the first part of this lab, you should now have (at least one) large files with the word frequencies in the amazon food reviews, in the format **word\tnumber** (a copy of that output is available in the HDFS shared folder /data/students/bigdata-01QYD/Lab1_OutputFirstPart/). You should also have realized that inspecting this result obtained running the word count application on finefoods_text.txt manually is not feasible/is not easy. Your task is to write a MapReduce application to filter the content of the output obtained running the word count application on finefoods_text.txt and analyze the filtered data.

The filter you should implement is the following:

- keep only the lines associated with words that start with the prefix “ho”.

How large is the result of this filter? Do you need to filter more?

Modify the application in order to accept the beginning string as a command-line parameter. Execute the new version of the program to select the words starting with the prefix that you prefer.