Lab 2

In this lab, you will write by your own a complete Hadoop application. Start by importing the template project available in **Lab2_Skeleton**.zip. 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 (even if we do not explicitly ask for it) the complexity of your program. Specifically try to understand, what is the effort you will require to the cluster in terms of time, network and I/O.

- How many pairs and bytes are read from HDFS?
- How many pairs and bytes are emitted by the mappers and hence how many data are sent on the network?

1. Filter an input dataset

If you completed Lab 1, you should now have (at least one) large files with the word frequencies in the Amazon food reviews, in the format `word\tnumber`, where number is an integer (a copy of the output of Lab 1 is available in the HDFS shared folder /data/students/bigdata-01QYD/Lab2/). You should also have realized that inspecting these results manually is not feasible. Your task is to write a Hadoop application to **filter** the content of the **output of Lab 1** and analyze the filtered data.
The filter you should implement is the following:

- Keep only the lines containing words that start with “ho” (**prefix**)

Store the selected lines (`word\tnumber`) in the output folder.

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

Then, extend the application, such that:

- it accepts the beginning string (i.e., the **prefix**) as a command-line parameter
- it prints on the **standard output of the Driver** the amount of selected and discarded words

Execute the new version of the program to select the words starting with the prefix that you prefer.

2. Frequency distribution of all the words

Develop a second, independent Hadoop application to evaluate the frequency distribution of all the words in the dataset. The input of the application is the output of Lab 1, the word frequencies in the Amazon food reviews (format `word\tnumber`). Given the frequency of each word (value `number` in the input file), compute for each of the following group the number of words with an associated frequency belonging to such group:

- **Group 0**: interval [0, 100), words with an associated frequency between 0 and 99
- **Group 1**: interval [100, 200), words with an associated frequency between 100 and 199
- **Group 2**: interval [200, 300), words with an associated frequency between 200 and 299

- **Group 3**: interval [300, 400), words with an associated frequency between 300 and 399
- **Group 4**: interval [400, 500), words with an associated frequency between 400 and 499
- **Group 5**: interval [500, +inf), words with an associated frequency of 500 or more

Example

Input file:

Word	Number
Hello	1
World	99
Hadoop	501
Spark	500
BigData	342

Group	Words in the group
0	Hello, World
1	
2	
3	BigData
4	
5	Hadoop, Spark

Output file:

Group0 2
Group3 1
Group5 2

Access the logs

If you need to access the log files associated with the execution of your application, use the following commands in the terminal of jupyter.polito.it:

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
     - You can retrieve the application id also from the HUE interface
   - The returned result contains one stdout log section for each task
     - 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

⚠⚠⚠ Shut down JupyterHub container ⚠⚠⚠

As soon as you complete all the tasks and activities on JupyterHub environment, please remember to shut down the container to let all your colleagues in all the sessions connect on JupyterHub and do all the lab activities.

1. Go into File -> Hub Control Panel menu

2. A new browser tab opens with the “Stop My Server” button. Click on it and wait till it disappears.