Lab 5

The objective of this laboratory is to start playing around with Apache Spark. We provide you a template in Java, from which you can write your first application. As a first task, you will repeat the exercise you had in Lab 2, which is reported below for your convenience. In Lab 2 you used MapReduce. Now you must use Spark and RDDs.

The workflow for building is the same as the one we used with Hadoop: once unzipped the template, import and open the project in your IDE (Eclipse), modify it, and then export a jar. Then, submit this job either locally (i.e., on the PC of LABINF):

```
spark-submit  --class it.polito.bigdata.spark.example.SparkDriver
--deploy-mode client --master local SparkProject.jar arguments
```

or on the cluster (you must execute spark-submit on bigdatalab.polito.it to submit your application on the cluster):

```
spark2-submit  --class it.polito.bigdata.spark.example.SparkDriver
--deploy-mode cluster --master yarn SparkProject.jar arguments
```

Notice how in the template we did not, even if possible, configure the Spark master and other settings explicitly in the main class of the driver. In this way, you can always submit the same jar on any configuration, either Spark standalone locally or a Yarn cluster. The main advantage of this method is for testing your application: first you can try your application on your machine, on a sample file, and then safely submit the same application for the real work.

1. Filter a file

If you completed Lab 1, you should now have (at least one) huge 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 **Spark** application to filter these results, and analyze the filtered data.
The filter you should implement is the following:

- keep only the lines containing words that start with a prefix (a string) that is passed as a command-line parameter

1. Run you application locally on the **LABINF PC** by using the following command and select only the lines containing the words starting with “ho” from the local file SampleLocalFile.csv (open a linux shell on the LABINF PC and execute the following command in the folder containing the jar file associated with your application and SampleLocalFile.csv)

```
spark-submit  --class it.polito.bigdata.spark.example.SparkDriver
--deploy-mode client --master local SparkProject.jar
SampleLocalFile.csv LocalOutputFolder "ho"
```

Analyze the content of the local output folder.

2. Run you application on the cluster by using the following command and select only the lines containing the words starting with "ho" from the content of the HDFS folder /data/students/bigdata-01QYD/Lab2/ (Use ssh to open a shell on the gateway bigdatalab.polito.it by using the usual procedure and then execute the following command)

```
spark2-submit  --class it.polito.bigdata.spark.example.SparkDriver
--deploy-mode cluster --master yarn SparkProject.jar
/data/students/bigdata-01QYD/Lab2/ HDFSOutputFolder "ho"
```

Analyze the content of the HDFS output folder.

How to access logs by using the job history web page

The logs of your applications on the cluster is available at https://ma1-bigdata.polito.it:18489/
If there is any error and you need to analyze the content of the error log files follow these steps:

1. Open a new terminal on your local machine (do **not** use the previous terminal connected to bigdatalab.polito.it).
2. On the local terminal command line, type the command
   `kinit sXXXXXX`
   (then, you will be asked for your BigDataLab password)
   This command is used to obtain a Kerberos ticket (without it you cannot access the log web page)
3. Now you should have access to the spark job history and logs at https://ma1-bigdata.polito.it:18489/
4. Select your job and click on its App ID
   You can use the search box to select only your jobs
5. Click on the executor tab and analyze the stderr associated with you application.
   Pay attention that there is one log file for the driver and one log file for each executor.
   If you application raised an exception, analyze all the stderr files to identify the reason of the error (depending on the error type, the log containing the needed information can be the one associated with the driver or that of an executor)

How to access logs without the web page and the Kerberos ticket

If you are connecting from outside Polito, and hence 1) you cannot request a Kerberos ticket and 2) you cannot access the history web page, you can proceed as follows to retrieve the log files from the command line:

1. Open a shell on the gateway by using ssh
   ssh smatricola@bigdatalab.polito.it

2. Execute the following command in the remote shell:
   yarn logs -applicationId *application_1521819176307_2195*
   The last parameter is the application/job ID. You can retrieve the job ID of your application on the HUE interface: https://bigdatalab.polito.it:8080/hue/jobbrowser#!jobs

Bonus questions

Explore the web dashboard of your jobs[1]. The history and the dashboard of the jobs can be accessed at the following URL https://ma1-bigdata.polito.it:18489/
Note that a Kerberos ticket is needed to access that web page. Use the same step reported above to obtain a Kerberos ticket and access the interface available at https://ma1-bigdata.polito.it:18489/

Check out the DAG of your application (under Jobs tab click on the saveAsTextFile at SparkDriver.java link to access the DAG) and the statistics for the application, and then answer the following questions:

- How many executors have been allocated to your job? (The information is available in the Executors tab)
- How was the input data split among them?
- Was your application well balanced among the executors, or were there tasks much slower/heavier/bigger than the others? Why?

Keep in mind these questions for the next labs as well.

[1] For local applications, the web dashboard is killed as soon as your job finishes. On the cluster, it is moved on a history server, so you can access it also after the job has completed.

Lab 6

In this lab, we continue our work on the Amazon dataset using Apache Spark. Your task is similar to that of Lab 3: given the original Amazon food dataset (that you can find in the HDFS file system at /data/students/bigdata-01QYD/Lab3/Reviews.csv), find all the pairs of items frequently reviewed together.
In the following Ex. 1 you find the steps that you are required to perform on the dataset.

Ex. 1

Write a single Spark application that:

- Transposes the original Amazon food dataset, obtaining a PairRDD of the type:

  <user_id> → <list of the product_ids reviewed by user_id>
- Counts the frequencies of all the pairs of products reviewed together;
- Writes on the output folder all the pairs of products that appear more than once and their frequencies. The pairs of products must be *sorted by decreasing frequency*.

The input Amazon food dataset (available in the HFDS shared folder of the BigData@Polito cluster: /data/students/bigdata-01QYD/Lab3/Reviews.csv) lists all the reviews per-row (one review per line), and is comma-separated. In each line, two of the columns represent the user id and product id. The schema of Reviews.csv is the following:
Id,ProductId,UserId,ProfileName,HelpfulnessNumerator,HelpfulnessDenominator,Score,Time,Summary,Text

Inspect the output of your application to search for interesting facts, and analyze the job execution as usual (performances, number of executors, etc…).

Pay attention that the line starting with “Id,” is the header of the file and must not be considered.

On the web site you can download the file ReviewsSample.csv. It contains a sample of Reviews.csv. You can use it to perform some initial tests locally.

Bonus task

Extend the implemented application in order to write on the standard output the top 10, most frequent, pairs and their frequencies.

Note that Spark 1.6, or above, provides the following actions that can be applied on an RDD of type JavaRDD<T>:
1) List<T> top(int n, java.util.Comparator<T> comp)
2) List<T> takeOrdered (int n, java.util.Comparator<T> comp)

top returns the **n largest elements** of the RDD based on the specified Comparator

takeOrdered returns **the n smallest elements** of the RDD based on the specified Comparator

Note that the standard output of the driver is stored in the log files of your application. Use the following steps top access the log files.

How to access logs by using the job history web page

The logs of your applications on the cluster is available at https://ma1-bigdata.polito.it:18489/

If there is any error and you need to analyze the content of the error log files follow these steps:

1. Open a new terminal on your local machine (do **not** use the previous terminal connected to bigdatalab.polito.it).
2. On the local terminal command line, type the command
   `kinit sXXXXXX`
   (then, you will be asked for your BigDataLab password)
   This command is used to obtain a Kerberos ticket (without it you cannot access the log web page)
3. Now you should have access to the spark job history and logs at https://ma1-bigdata.polito.it:18489/
4. Select your job and click on its App ID
   You can use the search box to select only your jobs
5. Click on the executor tab and analyze the stderr associated with you application.
   Pay attention that there is one log file for the driver and one log file for each executor.
   If you application raised an exception, analyze all the stderr files to identify the reason of the error (depending on the error type, the log containing the needed information can be the one associated with the driver or that of an executor)

How to access logs without the web page and the Kerberos ticket

If you are connecting from outside Polito, and hence 1) you cannot request a Kerberos ticket and 2) you cannot access the history web page, you can proceed as follows to retrieve the log files from the command line:

1. Open a shell on the gateway by using ssh
   ssh smatricola@bigdatalab.polito.it
2. Execute the following command in the remote shell:
   yarn logs -applicationId *application_1521819176307_2195*
   The last parameter is the application/job ID. You can retrieve the job ID of your application on the HUE interface: https://bigdatalab.polito.it:8080/hue/jobbrowser#!jobs