Exercises sheet n° 3: algorithms and programming in MapReduce and Spark

Using the Python shell in Spark.

Once you are connected to the AWS cluster (see previous lab-session), just use this command

> pyspark

In order to activate the shell and interact with Spark in Python.

How to install Spark on your Linux/Unix machine (optional)

Go to this web page

https://spark.apache.org/downloads.html

and select version a recent version of Spark.

The following relies on the 2.2.0 version, for more recent versions x.y.z, just replace '2.2.0' with 'x.y.z' in what follows.

Select the 'Pre-built for Hadoop 2.7' version, then download the file

spark-2.2.0-bin-hadoop2.7.tgz

into a directory of you choice.

By using a terminal, go into that directory and use the following command to unzip the file

tar xzvf spark-2.2.0-bin-hadoop2.7.tgz

You will see that a directory

spark-2.2.0-bin-hadoop2.7

will appear.

Now you just need the following command to launch pyspark

spark-2.2.0-bin-hadoop2.7/bin/pyspark

How to install Spark on Windows

You can refer to this useful guide

http://www.ics.uci.edu/~shantas/Install_Spark_on_Windows10.pdf

Exercise 1. *(to finish if not finished last lab-session)* Encoding SQL queries in MapReduce. Consider the following simple relational schema containing informations about clients and orders they made.

Customer(cid, startDate, name)

Order(#cid, total)

Note that, for the sake of simplicity, we do not have any primary key for Order. Also assume that all the fields are mandatory.

Provide the Pyhton MapReduce encoding of the following SQL queries.

- SELECT name FROM Customer WHERE month(startDate)=7
- SELECT DISTINCT name FROM Customer WHERE month(startDate)=7
- SELECT O.cid, SUM(total), COUNT(DISTINCT total) FROM Order O GROUP BY O.cid
- SELECT C.cid, O.total FROM Customer C, Order O WHERE C.cid=O.ci

For testing the jobs use the following files (use wget *file-url* for downloading them)

https://www.dropbox.com/s/tmt6u80mkrwfjkv/Customer.txt

https://www.dropbox.com/s/8n5cbmufqhzs4r3/Order.txt

Exercice 2. Consider a text file of you choice (for instance the shake.txt file of previous lab sessions), and design and implement in Python a Spark algorithm for the word-count problem.

In order to create the initial RDD, save you text file on HDFS and then use the following command do read the file and transform it into an RDD which we name 'document'.

Use pyspark for this lab session, we will see next how to launch/submit Spark programs.

```
>>> document = sc.textFile("hdfs://…path to …your .txt document »)
```

Observe that sc is a predefined object (sc stands for Spark Context) automatically created by the pyspark shell. sc is used as a *gateway* to the Spark world.

Exercice 3. Consider the RDDs pets seen in the course, and design a Spark script for compute the average quantity of each pet.
Still use pyspark.

Exercice 3 Design MapReduce jobs to randomly generate two text files F1.txt and F2.txt including points/vectors in R^5. For instance a point in the files is represented as '3.45, 5.0, 0, 0, 2.0'.
Consider that F1.txt should contain at least 5 thousands points, while F2.txt must contain exactly 10 points.
The output of the jobs can be concatenated in order to build Fi.txt. To this end use the HDFS command -cat

Questions:

1. Do you really need MapReduce to generate F2.txt ?
2. To generate F1.txt the ideal way to proceed is to have parallelism. How could you ensure that 10 parallel tasks generate the content to put in the file ?
3. Also, how you could ensure that duplicated points are never generated by the 10 parallel tasks ?

Exercice 4. Consider files F1.txt and F2.txt are on HDFS, inside a directory that you name InputForCrossProduct, under /user/hadoop.

Design a MapReduce job that takes has input the two files and builds the cross-product of F1.txt and F2.txt.
To illustrate if F1.txt contains '3.45, 5.0, 0, 0, 2.0' and F2.txt contains F1.txt contains '2.45, 1.0, 2.0, 2.0', then the output of the job must contain their concatenation

'3.45, 5.0, 0, 0, 2.0 2.45, 1.0, 2.0, 2.0'

where '\t' is used to separate the two vectors.

If needed, do not hesitate to pre-process F2.txt, even on the local file system. Also assume that 10, the cardinality of F2.txt, is initially known by the job you design.

Exercice 5. Solve exercices 3 and 4 with **Spark**. You are not allowed to use the Spark cross product transformation. You can use the zipWithIndex transformation to add a unique index to each element of an RDD. For instance

```
>>> rdd1 = sc.parallelize(['a','d','x'])
>>> rddzipped = rdd1.zipWithIndex()
>>> rddzipped.collect()
[('a', 0), ('d', 1), ('x', 2)]
>>>
```