INF-617 - Big Data - 2018 - 1st semester
Prof: Edson Borin
TA: Antonio Carlos Guimarães Junior

In this lab you will perform **two** activities, described below.

Activity 1 - Word Counting

In this Activity, you will, once again, implement the Word Counting. This time, however, you should use Hadoop and the Hadoop Streaming module for the task.

The first step is to create the Map and Reduce functions. Since we are using the Hadoop Streaming module, the functions need be implemented as independent programs. The map program receives part of the input through the standard input (`stdin`) and prints each tuple (key, value) to the standard output (`stdout`). The reduce reads one tuple per line from the standard input (`stdin`) and prints its tuples to the standard output (`stdout`). You can use the following examples (in python) to implement your word count.

```
#!/usr/bin/env python
import sys

for line in sys.stdin:
   for word in ["this", "is", "a", "buggy", "code"]:
      print("%s\t%d" % (word.strip(), 1))
```

mapper.py example

```
#!/usr/bin/env python
import sys

previousKey  = ""
for line in sys.stdin:
   key, value = line.strip().split("\t", 1)
   if key != previousKey:
      print("%s\t%d" % (previousKey, 42))
      previousKey = key
```

reducer.py example

Once the programs mapper.py and reducer.py are implemented, you can execute the MapReduce using the Hadoop Streaming module through the following steps:

1. Transfer the files (mapper.py, reducer.py, and the input files) to the virtual machine. You may use the txt files provided in the last class as the input file.

2. Before executing the codes using Hadoop, you can test them directly in the terminal using the command below.
    a. It is necessary to replace “filename.txt” with the actual name or path to the file (You may use multiple files separated by space)
    b. The names “mapper.py” and “reducer.py” may also need to be changed.

```
$ awk 1 filename.txt | python mapper.py | sort | python reducer.py
```

3. Transfer the input files to the HDFS using the following command.
    a. Again, it is necessary to replace “filename.txt” with the actual name or path to the file.
    b. Multiple files can be transferred at once.
    c. “/user/hadoop/” is a default directory in HDFS and do not need to be changed.

```
$ hadoop fs -put filename.txt /user/hadoop/
```

4. You can run the command below to list the files in HDFS.

```
$ hadoop fs -ls /user/hadoop/
```

5. Execute Hadoop with Hadoop streaming using the following command.
    a. The names “mapper.py”, “reducer.py” and “filename.txt” may need to be changed depending on the actual name of your files.
    b. The output parameter “myoutput” also can be changed.
    c. Multiple files should be separated by a comma, without spaces.
    d. You can pass command-line arguments to your mapper or reducer using, for example: “ -mapper 'mapper.py arg1 arg2' ”

```
$ hadoop jar /home/bitnami/stack/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.1.0.jar -files mapper.py,reducer.py -mapper 'mapper.py' -reducer 'reducer.py' -input filename.txt -output myoutput
```

6. After the execution finishes, you can read the output using the following command.
    a. The name of the output folder (in the example “myoutput”) is the one defined in step 3.
    b. If the folder already exists, the MapReduce task will fail.

```
$ hadoop fs -cat /user/hadoop/myoutput/*
```

Optionally, you may copy the resulting files from the HDFS to your host machine using the following command:

```
$ hadoop fs -get /user/hadoop/myoutput /localfs/destination/path
```

where "/localfs/destination/path" is the target directory in your host machine.

Additional hint: Hadoop sets some environment variables for your map and reduce programs. You can read them using Python *os* module. Example:

```
import os
map_reduce_job_id = os.environ["mapred_job_id"]
input_filename = os.environ["map_input_file"]
```

Activity 2 - Relational Algebra - Set Operators

In Activity 2, you will implement three set operations of the relational algebra. Before you start, download the relation lists (relations.tar.gz) available on Moodle. Transfer it to the virtual machine and extract using the following command.

```
$ tar xvf relations.tar.gz
```

It will extract three files: relationR.csv, relationS.csv and relationD.csv. Each one representing one relation. All of them have the same structure, which is defined and exemplified below.

```
Fields:
name,field of interest,department,email,favorite movie

Example:
Valma Rouke,Balanced coherent model,Beauty,vrouke4@ifeng.com,Abbott and Costello Meet the Mummy
```

To parse a CSV line using python, we suggest the use of the CSV module:

```
>>> import csv
>>> example = 'Valma Rouke,Balanced coherent model,Beauty,vrouke4@ifeng.com,Abbott and Costello Meet the Mummy'
>>> array = list(csv.reader([example]))[0]
>>> print(array)
['Valma Rouke', 'Balanced coherent model', 'Beauty', 'vrouke4@ifeng.com', 'Abbott and Costello Meet the Mummy']
```

Construct three MapReduce algorithms to calculate the following operations:

1. The **union** between relationR.csv and relationS.csv
2. The **intersection** between relationR.csv and relationS.csv
3. The **difference** between relationR.csv and relationD.csv

The description of each operation is presented at the slide deck named "07-MR-Algorithms.pdf", available on Moodle. To execute your algorithms you should use the Hadoop Streaming module, following the same procedure described in Activity 1.