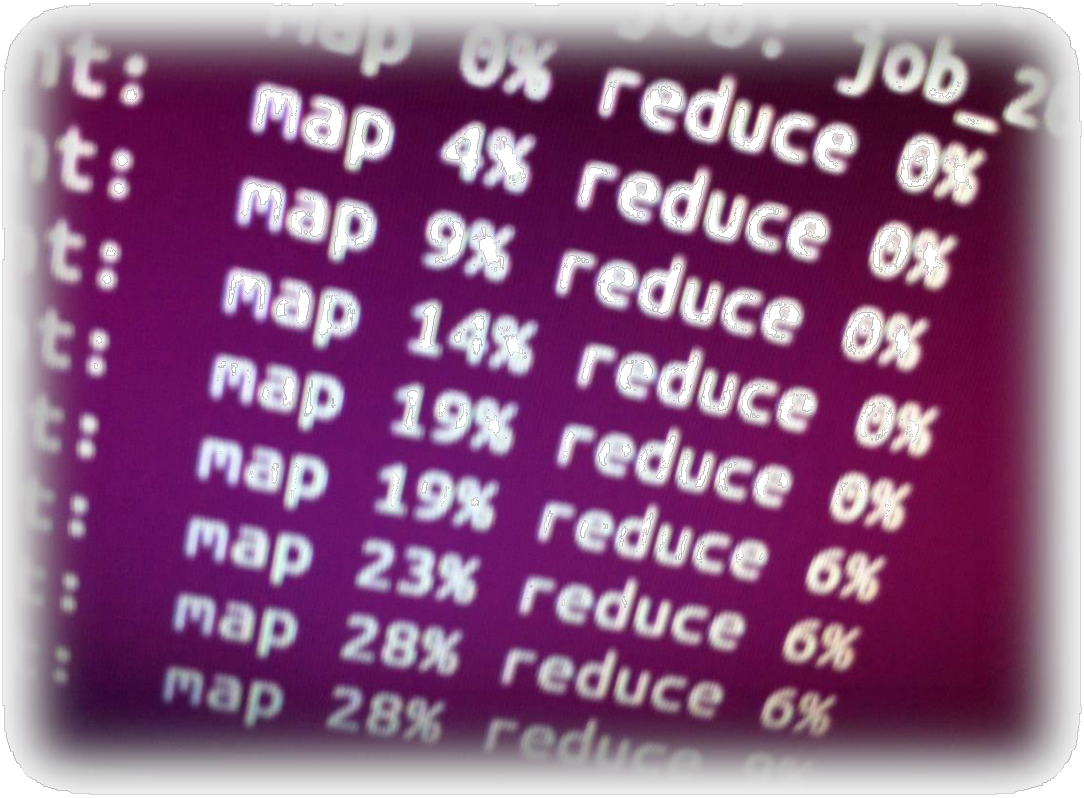


Big Data and Geospatial Data Mining

Hadoop Laboratory Work

August, 2024

Ángel Martín Furones

This document was written using Hadoop version 3.3.6, Although the current version as of August 2024 is 3.4.0, the commands and results must be the same.

HDFS and MapReduce from Hadoop V.3

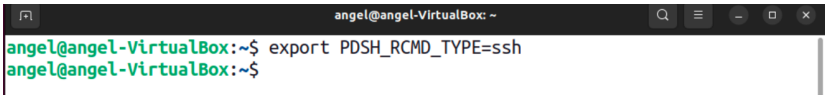
After installing Hadoop in a pseudo-distributed way, a first MapReduce job has been executed, the calculation of the pi number, for this we have not needed extra data or the use of HDFS.

Let execute another work, the classic word counter of a text (wordcount), for which we will need a data file and distribute it within the cluster through HDFS.

For that we are going to use the book by Frank Kane “Building an Internet Business”, used previously in the MapReduce laboratory work with the file name “Book.txt”

From the Ubuntu terminal, to prevent pdsh and ssh from conflicting when launching the different Hadoop processes (daemons), run the command:

$ export PDSH_RCMD_TYPE=ssh



Go to the *bin* directory of hadoop and format the namenode:

$ hdfs namenode –format

Go to the *sbin* directory of hadoop and start all the processes (daemons) executing the orders:

$./start-dfs.sh
$./start-yarn.sh

From this moment the namenode can be monitored from localhost 9870:

http://localhost:9870

and the ResourceManager from localhost 8088:

http://localhost:8088

Return to the *bin* directory and create the input directory within HDFS:

$ hdfs dfs –mkdir /input

Copy Book.txt file to the input directory of hdfs:

$ hdfs dfs –put /home/angel/Documents/BigData/mapreduce/WordFrequency/Book.txt /input

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$ hdfs dfs -put /home/angel/Documents/BigData/mapreduce/WordFrequenc
y/Book.txt /input
```

To see all the directories in HDFS you can use the order:

$ hdfs dfs –ls /

and to see what's in a specific directory:

$ hdfs dfs –ls /input

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - angel supergroup          0 2023-08-08 16:18 /input
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$ hdfs dfs -ls /input
Found 1 items
-rw-r--r--   1 angel supergroup     264875 2023-08-08 16:18 /input/Book.txt
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$
```

Number of replicas configured in the hdfs-site.xml file

Size of the file

To remove a directory in a recursive way use:

$ hdfs dfs -rmr /name of the directory in HDFS

Or:

$ hdfs dfs -rm -r /name of the directory in HDFS

If you want to move several files to HDFS, for example rename the file Book.txt as Bookb.txt and make a copy like Bookc.txt, both inside the local directory /WordFrequency, it can be done, in this case, by executing the order:

$ hdfs dfs –put /home/angel/Documents/BigData/mapreduce/WordFrequency/*.txt /input

If you want to see only the size of the files, execute

$ hdfs dfs –du /input

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$ hdfs dfs -du /input
264875  264875  /input/Book.txt
264875  264875  /input/Bookb.txt
264875  264875  /input/Bookc.txt
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$
```

If you want to see the size of a single file:

$ hdfs dfs –du /input/Book.txt

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$ hdfs dfs -du /input/Book.txt
264875  264875  /input/Book.txt
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$
```

And if you only want to see the total size of a directory:

$ hdfs dfs –du –s /input

It is also possible to execute Linux commands through pipelines, for example, to see only the first 20 lines of a file:

$ hdfs dfs –text /input/4363.txt | head –n 20

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$ hdfs dfs -text /input/Book.txt | head -n 20
Self-Employment: Building an Internet Business of One
Achieving Financial and Personal Freedom through a Lifestyle Technology Business
By Frank Kane


Copyright © 2015 Frank Kane.
All rights reserved worldwide.


CONTENTS
Disclaimer
Preface
Part I: Making the Big Decision
Overcoming Inertia
Fear of Failure
Career Indoctrination
The Carrot on a Stick
Ego Protection
Your Employer as a Security Blanket
text: Unable to write to output stream.
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$
```

The command:

$ hdfs dfs –help

Show all available options of the hdfs dfs command:

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/sbin$ hdfs dfs -help
Usage: hadoop fs [generic options]
        [-appendToFile [-n] <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum [-v] <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-concat <target path> <src path> <src path> ...]
        [-copyFromLocal [-f] [-p] [-l] [-d] [-t <thread count>] [-q <thread pool queue size>] <localsrc> ... <dst>]
        [-copyToLocal [-f] [-p] [-crc] [-ignoreCrc] [-t <thread count>] [-q <thread pool queue size>] <src> ... <localdst>]
        [-count [-q] [-h] [-v] [-t [<storage type>]] [-u] [-x] [-e] [-s] <path> ...]
        [-cp [-f] [-p | -p[topax]] [-d] [-t <thread count>] [-q <thread pool queue size>] <src> ... <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] [-v] [-x] <path> ...]
        [-expunge [-immediate] [-fs <path>]]
        [-find <path> ... <expression> ...]
        [-get [-f] [-p] [-crc] [-ignoreCrc] [-t <thread count>] [-q <thread pool queue size>] <src> ... <localdst>]
        [-getfacl [-R] <path>]
        [-getfattr [-R] {-n name | -d} [-e en] <path>]
        [-getmerge [-nl] [-skip-empty-file] <src> <localdst>]
        [-head <file>]
        [-help [cmd ...]]
        [-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [-e] [<path> ...]]
```

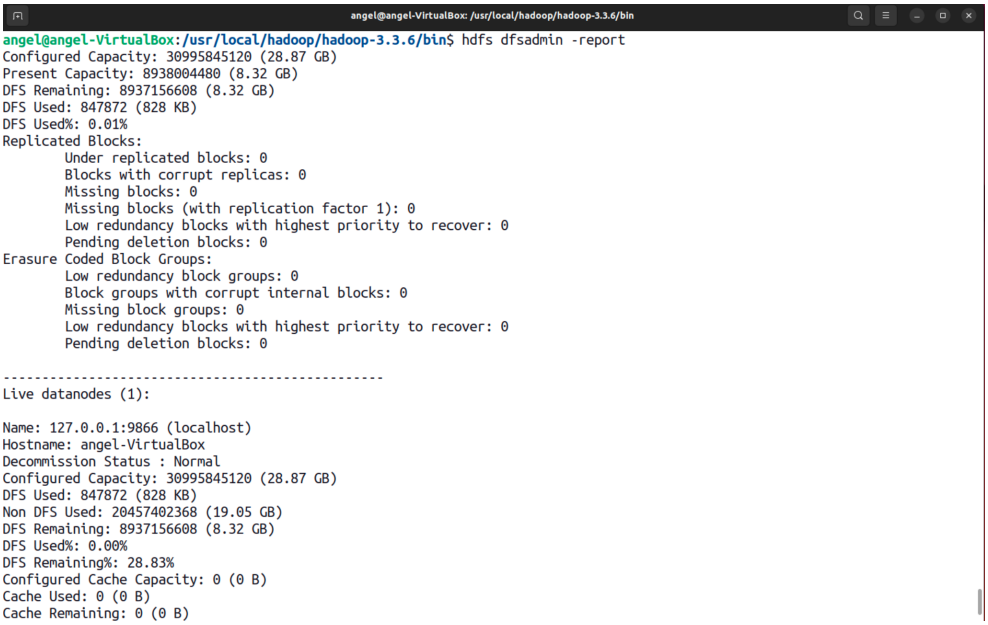
If we want to know a specific command (*du* for example):

$ hdfs dfs –help du

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$ hdfs dfs -help du
-du [-s] [-h] [-v] [-x] <path> ... :
  Show the amount of space, in bytes, used by the files that match the specified
  file pattern. The following flags are optional:

  -s  Rather than showing the size of each individual file that matches the
      pattern, shows the total (summary) size.
  -h  Formats the sizes of files in a human-readable fashion rather than a number
      of bytes.
  -v  option displays a header line.
  -x  Excludes snapshots from being counted.

  Note that, even without the -s option, this only shows size summaries one level
  deep into a directory.

  The output is in the form
        size    disk space consumed     name(full path)
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$
```

As already stated, the status of the NameNode can also be viewed from localhost 9870. Also, to see the status of the cluster and get details about the nodes that are supporting the HDFS file system from the terminal, you can execute the order:

$ hdfs dfsadmin –report



Where you can see the amount of free space left in each node, for example.

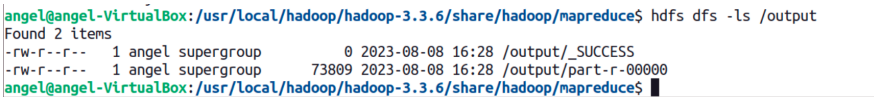
Once you have the files in HDFS, proceed to execute the MapReduce wordcount work, for this Hadoop comes with a series of sample routines ready to be executed (such as the calculation of the pi number already executed when hadoop was installed), for this go to directory /share/hadoop/mapreduce inside hadoop and run:

$ yarn jar hadoop-mapreduce-examples-3.3.6.jar wordcount /input /output

If you execute it this way, the wordcount process will be carried out on all the files that are inside the HDFS directory /input (three in this case), the results will be saved in the /output directory within HDFS (you can put the name you want but the directory must not be created in HDFS, it is a Hadoop security mechanism to ensure that results from previous jobs are not deleted, if you put the name of a file that already exists in HDFS the execution of the MapReduce job will fail).

Once the program has been executed, you can see the files generated by the process within the /output directory, for this return to the bin directory and execute from the terminal:

$ hdfs dfs –ls /output



An empty file called _SUCCESS has been generated that only informs that the process has been executed successfully and the file with the solution: part-r-00000.

To pass the HDFS files to a local directory run:

$ hdfs dfs –get /output/* /home/angel/Documents/BigData/hadoop

The process can also be monitored from localhost 8088.

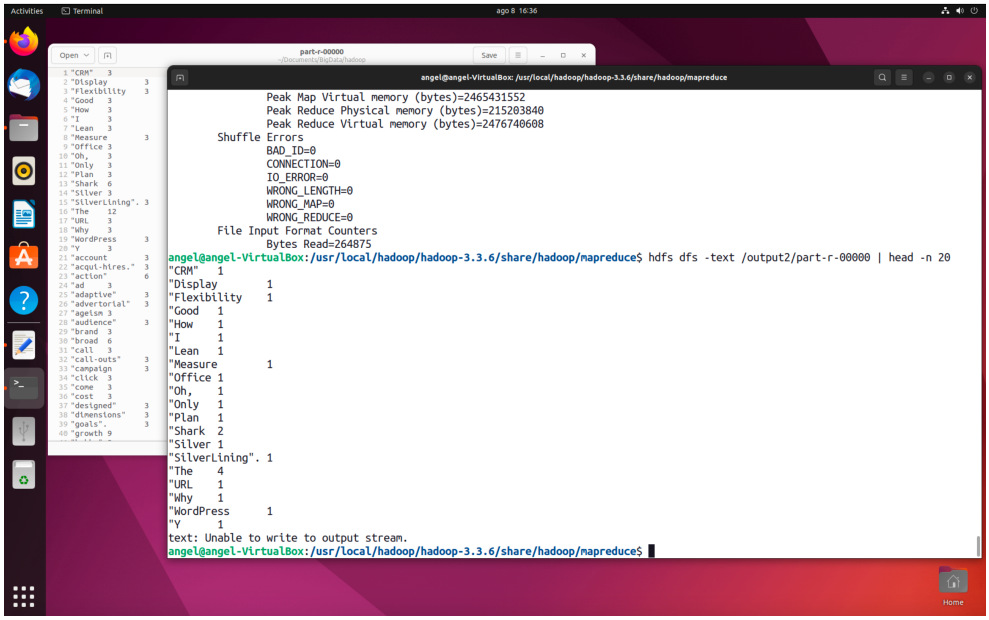
To verify that the three files have been used in the MapReduce process we can delete the two copies in HDFS with the order:

$ dhfs dfs –rm /input/Bookb.txt
$ hdfs dfs –rm /input/Bookc.txt

If you rerun the wordcount process as we did before (putting as output /output2 directory for example), you can open the resulting file with the command (from bin directory):

$ hdfs dfs –text /output2/part-r-00000 | head –n 20

and compare it with the one we had previously generated and saved locally, if everything went well, the one we have stored locally must have triple the number of words that the file resulting from the process we just executed:



The MapReduce processes that Hadoop has installed as an example for its execution are:

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/share/hadoop/mapreduce$ yarn jar hadoop-mapreduce-examples-3.3.6.jar
An example program must be given as the first argument.
Valid program names are:
  aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.
  aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in the input files.
  bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
  dbcount: An example job that count the pageview counts from a database.
  distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.
  grep: A map/reduce program that counts the matches of a regex in the input.
  join: A job that effects a join over sorted, equally partitioned datasets
  multifilewc: A job that counts words from several files.
  pentomino: A map/reduce tile laying program to find solutions to pentomino problems.
  pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.
  randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.
  randomwriter: A map/reduce program that writes 10GB of random data per node.
  secondarysort: An example defining a secondary sort to the reduce.
  sort: A map/reduce program that sorts the data written by the random writer.
  sudoku: A sudoku solver.
  teragen: Generate data for the terasort
  terasort: Run the terasort
  teravalidate: Checking results of terasort
  wordcount: A map/reduce program that counts the words in the input files.
  wordmean: A map/reduce program that counts the average length of the words in the input files.
  wordmedian: A map/reduce program that counts the median length of the words in the input files.
  wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/share/hadoop/mapreduce$
```

A very useful option for saving small files within a large one in HDFS (for example thousands of execution logs and user interaction with a web in a single file within Hadoop) is the *–appendToFile* option; for example we can execute the order (from the bin directory of Hadoop):

$ hdfs dfs –appendToFile /home/angel/Documents/BigData/mapreduce/WordFrequency/Bookb.txt /input/Book.txt

and the content of the local file will be added to the content of the file in HDFS (if the name of the file that we put in HDFS does not exist, Hadoop will create the file), to verify that it has worked we can see that the size of the file in HDFS has increased its size at about twice the original:

$ hdfs dfs –ls /input

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$ hdfs dfs -ls /input
Found 1 items
-rw-r--r--   1 angel supergroup     529750 2023-08-08 16:39 /input/Book.txt
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/bin$
```

Finally, it must be taken into account that the output of a MapReduce process can generate a large file, in which case Hadoop will divide the output into several files of the same size:

Nombre	Fecha de modificación	Tamaño	Clase
_SUCCESS	13 jun 2016 2:38	0 bytes	Docu
part-00000	13 jun 2016 2:38	3 KB	Docu
part-00001	13 jun 2016 2:38	3 KB	Docu
part-00002	13 jun 2016 2:38	3 KB	Docu
part-00003	13 jun 2016 2:38	3 KB	Docu
part-00004	13 jun 2016 2:38	3 KB	Docu
part-00005	13 jun 2016 2:38	3 KB	Docu
part-00006	13 jun 2016 2:38	3 KB	Docu

To generate a single output file locally, you can run hdfs dfs with the *–getmerge* option:

```
-getmerge [-nl] <src> <localdst> :
  Get all the files in the directories that match the source file pattern and
  merge and sort them to only one file on local fs. <src> is kept.

  -nl  Add a newline character at the end of each file.
```

Supose in the output directory are several part-* files with the output data, the order to join them all in one file would be:

$ hdfs dfs –getmerge –nl /output/part* /DIRECTORIO_LOCAL/salidaTodo.txt

To shut down the running processes, go to the sbin directory of hadoop and run:

$ ./stop-yarn.sh
$ ./stop-dfs.sh

If we turn them on again, the directory structure created in HDFS will remain, that is, we will continue to have the directories /input, /output and /output2, since the namenode has not lost its address links with respect the localhost, but if turn off the machine you will lose those links so you have to format the namenode again, with which will lose the entire directory structure generated in previous works.

MapReduce Job execution with hadoop-streaming

The hadoop-streaming code translation tool is located in the hadoop directory: share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar

For this practice we will test with the file *purchases.txt* and the *mapper.py* and *reducer.py* used previously in another practice, the process is very simple:

If we start from scratch, that is, with the virtual machine turned off, we will first format the hadoop namenode and start all java processes (daemons), then we will save the purchases.txt file in HDFS, for example in the directory /input (which we must create with mkdir).

Then, from the terminal located in the directory where we have recorded the file hadoop-streaming-3.3.6.jar execute the order:

$ hadoop jar hadoop-streaming-3.3.6.jar
-mapper “python mapper.py” → Order to execute the mapper
-reducer “python reducer.py” → Order to execute the reducer
-input /input/purchases.txt → Input file
-output /output3 → Output directory that should not exist
-file */path to mapper.py* → Complete path to mapper file
-file */path to reducer.py* → Complete path to reducer file

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/share/hadoop/tools/lib$ hadoop jar hadoop-streaming-3.3.6.jar -mapper "python
3 mapper.py" -reducer "python3 reducer.py" -input /input/purchases.txt -output /output3 -file /home/angel/Documents/BigData/mapredu
ce/purchases/mapper.py -file /home/angel/Documents/BigData/mapreduce/purchases/reducer.py
```

The solution will be found in the output file within HDFS:

```
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/share/hadoop/tools/lib$ hdfs dfs -ls /output3
Found 2 items
-rw-r--r--   1 angel supergroup          0 2023-08-08 16:49 /output3/_SUCCESS
-rw-r--r--   1 angel supergroup       3007 2023-08-08 16:49 /output3/part-00000
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/share/hadoop/tools/lib$ hdfs dfs -text /output3/part-00000 | head -n 20
Albuquerque 10052311.419999998
Anaheim 10076416.35999995
Anchorage 9933500.39999997
Arlington 10072207.969999978
Atlanta 9997146.699999945
Aurora 9992970.920000078
Austin 10057158.899999965
Bakersfield 10031208.919999931
Baltimore 10096521.449999949
Baton Rouge 10131273.229999999
Birmingham 10076606.519999951
Boise 10039166.740000017
Boston 10039473.28000001
Buffalo 10001941.190000022
Chandler 9919559.860000027
Charlotte 10112531.339999989
Chesapeake 10038504.920000048
Chicago 10062522.070000123
Chula Vista 9974951.34000001
Cincinnati 10139505.740000024
angel@angel-VirtualBox:/usr/local/hadoop/hadoop-3.3.6/share/hadoop/tools/lib$
```

Hadoop HDFS and MapReduce with the class MRJob and Hadoop-Streaming

It is possible to execute the MRJob class inside Hadoop, it contain python code so Hadoop-Streaming should be used to do it. For this you must have activated all the java processes of Hadoop.

For example, from the terminal located in the directory where we have the file *purchases.txt* and the program *SolMRJob.py* run:

$ python SolMRJob.py
-r hadoop → Execution (run) on hadoop
--hadoop-streaming-jar="/COMPLETE PATH TO hadoop-streaming-3.3.6.jar"
purchases.txt > salidaHadoop.txt

```
angel@angel-VirtualBox:~/Documents/BigData/mapreduce/purchases$ python3 SolMRJob
.py -r hadoop --hadoop-streaming-jar="/usr/local/hadoop/hadoop-3.3.6/share/hadoo
p/tools/lib/hadoop-streaming-3.3.6.jar" purchases.txt > salidaHadoop.txt
```

The solution is generated temporarily in HDFS but, if not indicated otherwise, the final result is saved locally, in this case, *salidaHadoop.txt* file within the local directory where the *purchases.txt* and *SolMRJob.py* files are located.

This process uses only the MapReduce engine; it is also possible to use the HDFS storage system (a file that is already recorded in HDFS) by changing the last line of the previous order in the way:

$ python SolMRJob.py
-r hadoop
--hadoop-streaming-jar="/ COMPLETE PATH TO hadoop-streaming-3.3.6.jar"
hdfs:///input/purchases.txt > salidaHadoop2.txt

Or if we want the output file to be on HDFS (with the default Hadoop output structure):

```
$ python SolMRJob.py
      -r hadoop
      --hadoop-streaming-jar=”/ COMPLETE PATH TO hadoop-streaming-3.3.6.jar”
      --output-dir=hdfs:///output4/
      hdfs:///input/purchases.txt
```

The SolMRJob class on Hadoop will always need the input file to be provided from the terminal, as has been done in this exercise; If we enter the reading from within the *SolMRJob.py* program code with the command *sys.stdin=open("file","r")* the MapReduce process will fail.

Access to HDFS from Python

There are several Python libraries that are capable of working with HDFS, some of the most used are *pydoop, WebHDS* or *snakebyte*, all of them have some limitations and the different versions and updates of *Hadoop* can cause some of them, if not updated conveniently, it may not work properly.

To interact with HDFS, all Hadoop processes (*daemons*) must be active, so the interaction is based on typing commands directly in the terminal window. To do this, the easiest way is to use the Python library *Subprocess*.

Through the Python *subprocess* library (which usually comes already installed) it is possible to interact with the terminal and send and execute commands. This library allows you to generate new processes, connect to their input/output/error channels and obtain their return codes.

A simple programming structure where the HDFS directories can be seen in the Python terminal, where a new directory is generated in HDFS and where a file is saved from local to HDFS and vice versa can be seen below (you can download the file from PoliformaT in the Hadoop section of the practices section: ConnectHDFS.py):

File Edit Format Run Options Window Help

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Sep 6 09:40:02 2023

@author: angel
"""
import subprocess

def run_cmd(args_list):
        """
        run linux commands
        """
        proc = subprocess.Popen(args_list, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        s_output, s_err = proc.communicate()
        return s_output, s_err

##Run Hadoop -ls / command in Python
(out, err)= run_cmd(['hdfs', 'dfs', '-ls', '/'])
print(out)

##Run Hadoop -mkdir command in Python
(out, err)= run_cmd(['hdfs', 'dfs', '-mkdir', '/input2'])

##Run Hadoop -ls / command in Python
(out, err)= run_cmd(['hdfs', 'dfs', '-ls', '/'])
print(out)

##Run Hadoop -put command in Python
(out, err)= run_cmd(['hdfs', 'dfs', '-put',
                     '/home/angel/Documents/BigData/mapreduce/WordFrequency/Book.txt', '/input2'])

##Run Hadoop -get command in Python
(out, err)= run_cmd(['hdfs', 'dfs', '-get',
                     '/input2/Book.txt', '/home/angel/Documents/BigData/hadoop/'])

```

Ln: 13 Col: 11