TP 1 : Gestion et analyse des Big Data

I/ Suivre les étapes suivantes pour exécuter le programme Wordcount en langage Java

1/ Démarrer l'hadoop en suivant les commandes

```
sudo rm -r hdfs/datanode/current

sudo rm -r hdfs/namenode/current

hdfs namenode -format

start-dfs.sh

start-yarn.sh
```

Soyez sûr que les six services sont démarrés

Remarque :

Le fichier /usr/local/hadoop/etc/hadoop/mapred-site.xml doit contenir la propriété :

```
<property>
    <name>mapreduce.application.classpath</name>

<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*</value>
  </property>
```

Et le fichier /usr/local/hadoop/etc/hadoop/yarn-site.xml doit aussi contenir la propriété

```
<property>
    <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_HOME,PATH,LANG,TZ,HADOOP_MAPRED_HOME</value>
  </property>
```

2/ préparer un fichier texte avec une bonne taille ; mettre le dans /home/Desktop/exemple.txt

3/ Mettre ce ficher dans le hdfs

```
hadoop fs -put /home/Desktop/exemple.txt /imput_dir
```

Verifier dans le navigateur la présence du dossier imput_dir

4/ exécuter la classe wordcount pour compter le nombre d'occurrence de mots

hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.1.jar wordcount /imput_dir /output_dir

verifier votre hdfs

5/ afficher le contenu du fichier part-r-00000

hdfs dfs -cat /output_dir/part-r-00000

II/ Suivre les étatpes suivantes pour exécuter wordcount écrit en Python

1/ créer les fichiers mapper.py

cd Documents

touch mapper.py

nano mapper.py

saisir ce code python qui la fonction map

```
#!/usr/bin/python3


# import sys because we need to read and write data to STDIN and STDOUT
import sys


# reading entire line from STDIN (standard input)
for line in sys.stdin:
        # to remove leading and trailing whitespace
        line = line.strip()
        # split the line into words
        words = line.split()


        # we are looping over the words array and printing the word
        # with the count of 1 to the STDOUT
        for word in words:
                # write the results to STDOUT (standard output);
```

```
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        print ('%s\t%s' % (word, 1))
```

1/ créer les fichiers reducer.py

```
cd Documents
touch reducer.py
nano reducer.py
```

saisir ce code python qui la fonction reduce

```
#!/usr/bin/python3

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# read the entire line from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # splitting the data on the basis of tab we have provided in mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
```

```
            # count was not a number, so silently
            # ignore/discard this line
            continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
            current_count += count
    else:
            if current_word:
                    # write result to STDOUT
                    print ('%s\t%s' % (current_word, current_count))
            current_count = count
            current_word = word

# do not forget to output the last word if needed!
if current_word == word:
        print ('%s\t%s' % (current_word, current_count))
```

3/ vérifier si vous avez Python 3 ; sinon install le

```
sudo apt update
sudo apt install python3
sudo add-apt-repository ppa:deadsnakes/ppa
```

4/ Préparer un fichier texte et mettre le dans Documents/word_count_data.txt

5/ donner les droits d'accès aux deux programmes

```
chmod 777 /home/Documents/mapper.py
chmod 777 /Documents/reducer.py
```

6/ vérifier le fonctionnement des deux programmes

cat word_count_data.txt |python3 mapper.py

cat word_count_data.txt |python3 mapper.py | sort -k1,1 | python3 reducer.py

7/ sortir du repertoire Documents

8/ Démarrer Hadoop

hdfs dfs -mkdir /word_count_in_python

hdfs dfs -copyFromLocal /home/votre-user/Documents/word_count_data.txt word_count_in_python

votre-user c'est le nom de votre compte

maintenant mettre le jar hadoop-streaming.3.3.1.jar dans le dossier Documents

hadoop jar /home/votre-user/Documents/hadoop-streaming-3.3.1.jar -input word_count_in_python/word_count_data.txt -output word_count_in_python/output -mapper /home/votre-user/Documents/mapper.py -reducer /home/votre-user/Documents/reducer.py

9/ vérifier le résultat et afficher le contenu du fichier part-00000

Travail à rendre dans la séance

Ecrire en pseudo-code python les fonctions Map et Reduce qui comptent le nombre de voyelles et celui des consonnes dans un texte en entrée.

Note : le travail sera calculé dans la note TP