


<b>Matière :</b> Framework et architectures et Big Data <b>Enseignante responsable :</b> Leila Baccour <b>Enseignantes de TP :</b> Hana Mallek, Maysam Chaari	<b>Section :</b> T-LSI MM	AU : 2023-2024 
---	------------------------------	---

## TP3 : Exécution de jobs MapReduce sur un seul nœud

### 1. Objectifs

L'objectif de ce TP est de comprendre le fonctionnement de Hadoop Distributed File System (HDFS) et d'exécuter des programmes avec le modèle MapReduce.

### 2. Mise en place de HDFS

2.1. Commencer en lançant les démons de Hadoop en exécutant les commandes suivantes :

**start-dfs.sh, start-yarn.sh**

Puis **jps** pour vérifier l'exécution de tous les démons de Hadoop

2.2. Créer deux répertoires nommés monTravail et data dans hdfs à travers la commande suivante :

**hdfs dfs -mkdir /monTravail**

**hdfs dfs -mkdir /data**

2.3. Si ça ne vous donne pas l'accès, il faut quitter le mode safe avec la commande : **hdfs dfsadmin -safemode leave**

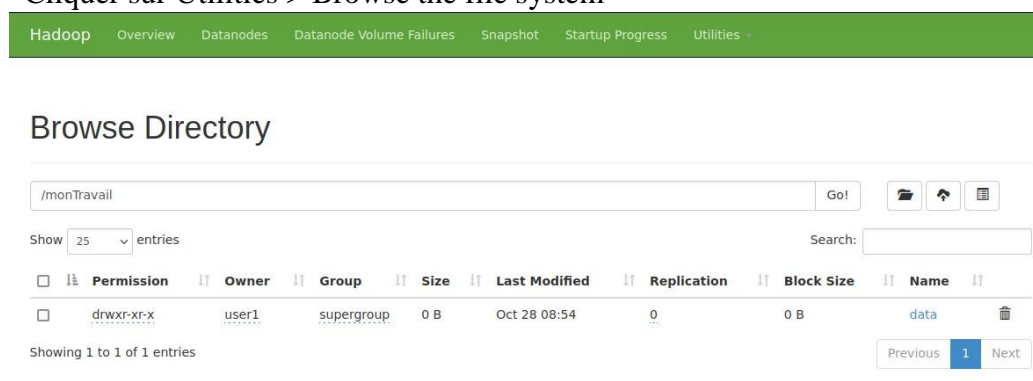
2.4. Déplacer le dossier data dans le dossier monTravail à travers la commande :

**hdfs dfs -mv /data /monTravail**

2.5. Vérifier l'existence des dossiers créés à travers la commande : **hdfs dfs -ls /monTravail**

Ou bien accéder à l'interface web du monTravail <http://localhost:50070>

Cliquer sur Utilities > Browse the file system



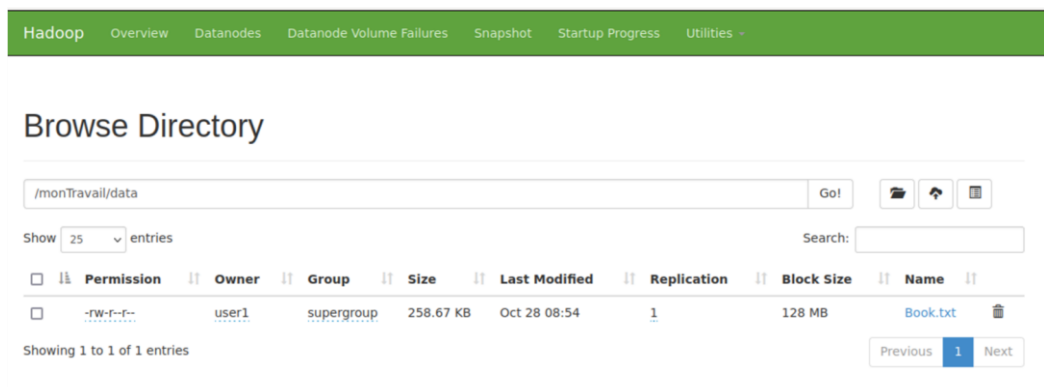
2.6. Créer dans la machine locale un dossier nommé « inputFiles » dans votre dossier personnel /home/user1

- 2.7. Téléchargez le fichier Book.txt depuis le Classroom et placez-le dans le répertoire inputFiles déjà crée.
- 2.8. Copier le fichier Book.txt du système de fichier local dans /Hadoop/data à travers la commande :

**hdfs dfs -put /home/user1/inputFiles/Book.txt /monTravail/data**

ou bien

**hdfs dfs -copyFromLocal /home/user1/inputFiles/Book.txt /monTravail/data**



- 2.9. Afficher les dernières lignes du fichier à travers la commande suivante **hdfs dfs -tail /monTravail/data/Book.txt**

### 3. Installation Python

- 3.1. Installer Python à travers la commande suivante : **sudo apt-get install python**
- 3.2. Vérifier l'installation par : **python --version**

### 4. Exécuter un programme MapReduce

#### 4.1. Exécution locale

Avant de lancer un programme MapReduce avec Hadoop, on le teste toujours localement. Dans ce cas, le programme MapReduce exécuté ne lit/écrit pas sur HDFS, mais sur le système de fichiers local.

#### Exercice 1 : Word Count avec MapReduce

Écrire un programme MapReduce pour compter le nombre d'occurrences de chaque mot ("WordCount" vu en cours) dans un fichier texte nommé book.txt.

a) Dans un éditeur de texte, copiez le code suivant et enregistrer-le sous le nom de mapper.py

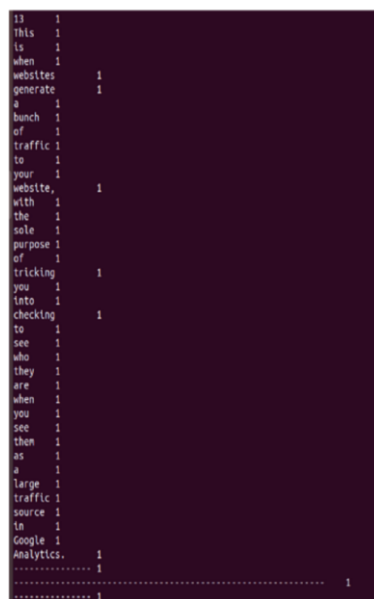
```
#!/usr/bin/env python
import sys
# Lecture de la ligne entière depuis STDIN (entrée standard)
for line in sys.stdin:
    # Pour supprimer les espaces en début et en fin de ligne
    line = line.strip()
    # Divisez la ligne dans words
    words = line.split()

    # Nous parcourons le tableau de mots et imprimons le word
    # avec un compteur de 1 vers STDOUT
    for word in words:
        # Écrivez les résultats vers STDOUT (sortie standard);
        # ce que nous produisons ici sera l'entrée pour l'étape de
        # réduction, c'est-à-dire l'entrée pour reducer.py
        print '%s\t%s' % (word, 1)
```

Tester notre fichier mapper.py en local pour vérifier s'il fonctionne correctement ou non.

**cat** <Emplacementfichier\_Book> | **python** <emplacement\_de\_code\_mapper>

La sortie du code mapper.py est affichée ci-dessous :



```
13 1
this 1
is 1
when 1
websites 1
generate 1
a 1
bunch 1
of 1
traffic 1
to 1
your 1
website, 1
with 1
the 1
sole 1
purpose 1
of 1
tricking 1
you 1
into 1
checking 1
to 1
see 1
who 1
they 1
are 1
when 1
you 1
see 1
then 1
as 1
a 1
large 1
traffic 1
source 1
in 1
Google 1
Analytics, 1
..... 1
..... 1
```

Lancer l'équivalent du shuffle par la commande suivante afin de trier la sortie de mapper en fonction des mots :

**cat** <Emplacementfichier\_Book>| **python** mapper.py | **sort -k1,1**

**sort** : Il s'agit de la commande de tri en

Unix/Linux.

**-k1,1**: Cela spécifie que nous voulons trier en utilisant la première colonne (clé) comme critère de tri. Le format -kX,Y signifie que nous utilisons la colonne X à la colonne Y pour trier. Dans ce cas, "1,1" signifie que nous utilisons uniquement la première colonne.

- b) Dans un autre éditeur de texte, copiez le code suivant et enregistrer-le sous le nom de reducer.py

```
#!/usr/bin/env python
import sys
# Initialisation de variables
current_word = None
current_count = 0
word = None
# Lire la ligne entière depuis STDIN (entrée standard)
for line in sys.stdin:
    # Supprimer les espaces en début et en fin de ligne
    line = line.strip()
    # Diviser les données en utilisant la tabulation comme séparateur tel que
    # spécifié dans mapper.py
    word, count = line.split('\t', 1)
    # Convertir count (actuellement une chaîne de caractères) en entier
    try:
        count = int(count)
    except ValueError:
        # Si count n'était pas un nombre, alors ignorons/supprimons cette ligne
        continue
    # Cette condition IF fonctionne parce que Hadoop trie la sortie du mappage
    # par clé (ici : le mot) avant de la transmettre au réducteur
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # Écrire le résultat vers STDOUT
            print '%s\t%s' % (current_word, current_count)
            current_count = count
            current_word = word
# N'oubliez pas de produire la dernière occurrence si nécessaire !
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

Ce code permet de lire la sortie de mapper.py depuis l'entrée standard (STDIN) et fait l'agrégation du nombre d'occurrences de chaque mot, puis il écrit la sortie finale sur la sortie standard (STDOUT).

Vérifier maintenant que notre code réducteur reducer.py fonctionne correctement avec mapper.py en utilisant la commande suivante :

**cat <Emplacementfichier\_Book> | python mapper.py | sort -k1,1 | python reducer.py**

```
eight-hour      1
either      8
either,      1
either.      5
Elance      1
electrical      1
electricians,  1
electronics    1
eliminate      3
eliminating    1
Eliminating    1
else      14
else,      5
else?      1
else.      6
Else      1
ELSE      1
else's      2
else's.      1
else's      2
email      24
email,      2
Email      5
E-mail      1
```

## 4.2 Exécution sur Hadoop

- a) Accorder la permission d'exécution aux fichiers mapper.py et reducer.py à l'aide de la commande ci-dessous :

```
cd <emplacement mapper.py et reducer.py>  
chmod 777 mapper.py reducer.py
```

- b) Vérifiez l'exécution des démons de Hadoop avec la commande jps. S'ils ne sont pas activés, lancez-les
- c) Exécuter le job MapReduce dans Hadoop à travers la commande suivante :

```
hadoop jar $HADOOP_INSTALL/share/hadoop/tools/lib/hadoop-streaming-2.10.2.jar \  
-file ../../mapper.py -mapper ../../mapper.py \  
-file ../../reducer.py -reducer ../../reducer.py \  
-input /<cheminHDFS>/Book.txt -output /<cheminHDFS>/Resultat
```

**Notez que le répertoire de sortie ne doit pas être créé avant le démarrage du job.**

Voilà le résultat de l'exécution:

```

Launched reduce tasks=1
Data-local map tasks=2
Total time spent by all maps in occupied slots (ms)=28373
Total time spent by all reduces in occupied slots (ms)=7274
Total time spent by all map tasks (ms)=28373
Total time spent by all reduce tasks (ms)=7274
Total vcore-milliseconds taken by all map tasks=28373
Total vcore-milliseconds taken by all reduce tasks=7274
Total megabyte-milliseconds taken by all map tasks=29053952
Total megabyte-milliseconds taken by all reduce tasks=7448576
Map-Reduce Framework
  Map input records=926
  Map output records=46249
  Map output bytes=356272
  Map output materialized bytes=448782
  Input split bytes=188
  Combine input records=0
  Combine output records=0
  Reduce input groups=6995
  Reduce shuffle bytes=448782
  Reduce input records=46249
  Reduce output records=6995
  Spilled Records=92498
  Shuffled Maps =2
  Failed Shuffles=0
  Merged Map outputs=2
  GC time elapsed (ms)=614
  CPU time spent (ms)=3850
  Physical memory (bytes) snapshot=586674176
  Virtual memory (bytes) snapshot=5685137408
  Total committed heap usage (bytes)=307437568
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=268971
File Output Format Counters
  Bytes Written=72450
23/10/16 10:19:48 INFO streaming.StreamJob: Output directory: /Resultat

```

- d) Accéder à l'interface web du Hadoop <http://localhost:50070> pour consulter le dossier de sortie Resultat situé à l'emplacement déclaré dans la requête /<chemin/Resultat/part00000.

Ou bien, utiliser la commande **cat** comme indiqué ci-dessous.

**hdfs dfs -cat /<chemin>/Resultat/part-00000**

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	user1	supergroup	0 B	Oct 16 10:19	1	128 MB	_SUCCESS
-rw-r--r--	user1	supergroup	70.75 KB	Oct 16 10:19	1	128 MB	part-00000

- e) Ajoutez un alias de cette commande dans le fichier .bashrc pour simplifier le démarrage du job Python en utilisant la saisie du code suivant :

```

mapreduce(){ hadoop jar $HADOOP_INSTALL/share/hadoop/tools/lib/hadoop-streaming-2.10.2.jar -file $1 -
mapper $1 -file $2 -reducer $2 -input $3 -output $4
}
alias hs= "mapreduce"

```

- f) Lancer le Job MapReduce avec cette instruction :

**hs <chemin local>/mapper.py <chemin local>/reducer.py <chemin sur hdfs>/Book.txt <chemin sur hdfs>/Resultat**

**Exercice 2 :**

Créer un programme mapReduce permettant de calculer le totale des ventes de chaque magasin du fichier transféré « magasin.txt » ;

- a. Tester localement les scripts ;
- b. Exécuter le programme sur Hadoop

**Exercice3 :**

Ouvrir le fichier 1800.csv et consulter son contenu. Ce fichier contient des informations de température dans différentes régions. Ecrire un programme mapReduce pour afficher le maximum de température de chaque région. Ceci est un extrait de fichier.

```
ITE00100554,18000101,TMAX,-75,,,E,  
ITE00100554,18000101,TMIN,-148,,,E,  
GM000010962,18000101,PRCP,0,,,E,  
EZE00100082,18000101,TMAX,-86,,,E,  
EZE00100082,18000101,TMIN,-135,,,E,  
ITE00100554,18000102,TMAX,-60,,I,E,
```

- a. Tester localement les scripts ;
- b. Exécuter le programme sur Hadoop