

# Bases de données NOSQL et Big data

---

## TP2 : Initiation à Hadoop

---

Diplôme National d'Ingénieur en Informatique

*Spécialité :*

**Génie Logiciel**

*Réalisée par :*

**Oussama Ben Slama**

Année Universitaire 2024/2025

# Chapitre 1

## Introduction

### 1.1 Hadoop MapReduce

Travailler avec une quantité massive de données peut poser plusieurs problèmes en termes de traitement. C'est pourquoi Hadoop introduit Hadoop MapReduce, un framework qui permet d'écrire facilement des applications traitant d'énormes quantités de données (ensemble de données de plusieurs téraoctets) en parallèle sur de grands clusters (des milliers de nœuds) de matériel standard, de manière fiable et tolérante aux pannes.

Dans cette section, nous allons simuler le mécanisme de ce logiciel en utilisant un simple code Python. Nous avons commencé par écrire un code `mapper.py` qui prend en entrée un texte, puis extrait les mots en utilisant les différents séparateurs possibles. Ensuite, nous avons développé `reducer.py` qui prend en entrée la sortie de *mapper.py* et calcule l'occurrence de chaque mot de manière optimale.

```

mapper.py ×
mapper.py
1  import sys
2
3  for line in sys.stdin:
4      words = line.strip().split(' ')
5      for word in words :
6          word = word.lower()
7          print(word , "\t" , 1 )
8
9
10 | You, 2 hours ago • Add mapper and reducer
11
12

```

FIGURE 1.1 – Code de *mapper.py*

```

reducer.py ×
reducer.py > ...
You, 2 hours ago | 1 author (You)
1  import sys
2
3  word = ""
4  count = 0
5  for line in sys.stdin:
6      item = line.strip().split('\t')
7      if(word == "" or item[0] == word) :
8          count += 1
9          word = item[0]
10     else :
11         print(word, "\t" , count)
12         word = item[0]
13         count = 1
14
15     print(word, "\t" , count)
16
17

```

FIGURE 1.2 – Code de *reducer.py*

L'exécution de ces fichiers donne le résultat suivant.

```
PS C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\Big-Data-Travaux-Pratiques\TP2> cat file.txt | python mapper.py | sort | python reducer.py
big 2
bonjour 2
cœur 1
data 2
du 1
est 1
et 1
hadoop 2
le 1
spark 1
PS C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\Big-Data-Travaux-Pratiques\TP2>
```

FIGURE 1.3 – Résultat de l'exécution

La commande utilisée est :

```
cat file.txt | python mapper.py | sort | python reducer.py
```

FIGURE 1.4 – Commande d'exécution

### 1.1.1 Installation de Python sur les nœuds

Pour installer Python sur les nœuds *namenode*, *historyserver*, *resource-manager*, *datanode* et *nodemanager*, il faut exécuter les commandes suivantes sur chaque nœud :

```
apt install python
```

FIGURE 1.5 – Modifier sources.list

```
sed -i 's/stretch/buster/g' /etc/apt/sources.list
```

FIGURE 1.6 – Mise à jour de apt

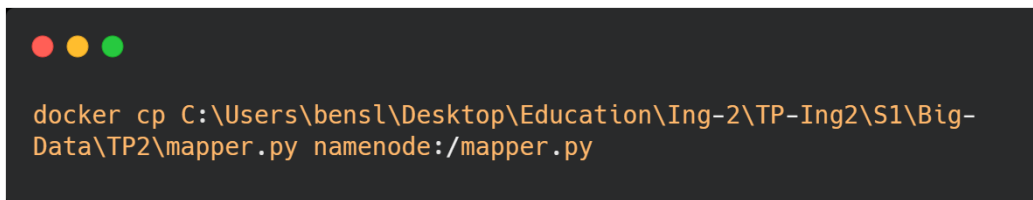


```
apt update
```

FIGURE 1.7 – Installation de Python

### 1.1.2 Copier les fichiers `mapper.py`, `reducer.py` et `file.txt` vers l'un des conteneurs

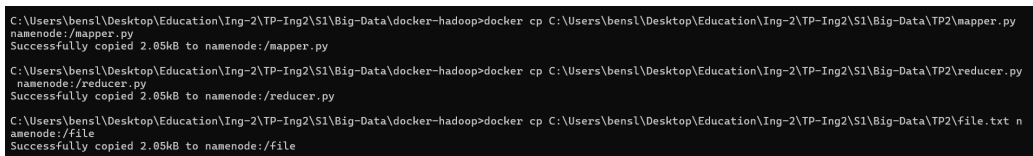
Dans cette étape, nous allons copier les fichiers `mapper.py`, `reducer.py` et `file.txt` vers le conteneur namenode en utilisant la commande **docker cp**.



```
docker cp C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\TP2\mapper.py namenode:/mapper.py
```

FIGURE 1.8 – Commande `docker cp`

Voici l'exécution :



```
C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\docker-hadoop>docker cp C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\TP2\mapper.py namenode:/mapper.py
Successfully copied 2.05kB to namenode:/mapper.py

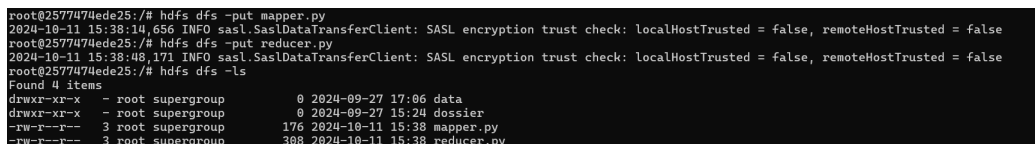
C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\docker-hadoop>docker cp C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\TP2\reducer.py namenode:/reducer.py
Successfully copied 2.05kB to namenode:/reducer.py

C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\docker-hadoop>docker cp C:\Users\bensl\Desktop\Education\Ing-2\TP-Ing2\S1\Big-Data\TP2\file.txt namenode:/file
Successfully copied 2.05kB to namenode:/file
```

FIGURE 1.9 – Résultat de la copie

### 1.1.3 Copier les fichiers vers le système de fichiers Hadoop

Pour copier des fichiers vers le système de fichiers Hadoop, on utilise la commande **hdfs dfs -put**.



```
root@2577474ede25:/# hdfs dfs -put mapper.py
2024-10-11 15:38:14,656 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
root@2577474ede25:/# hdfs dfs -put reducer.py
2024-10-11 15:38:48,171 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localhostTrusted = false, remoteHostTrusted = false
root@2577474ede25:/# hdfs dfs -ls
Found 4 items
drwxr-xr-x - root supergroup 0 2024-09-27 17:06 data
drwxr-xr-x - root supergroup 0 2024-09-27 15:24 dossier
-rw-r--r-- 3 root supergroup 176 2024-10-11 15:38 mapper.py
-rw-r--r-- 3 root supergroup 388 2024-10-11 15:38 reducer.py
```

FIGURE 1.10 – Copie vers le système de fichiers Hadoop

## 1.2 Hadoop Streaming

Nous allons chercher le fichier nommé *hadoop-streaming\*.jar* avec la commande **find**.

```
root@2577474ede25:/# find / -name 'hadoop-streaming*.jar'
/opt/hadoop-3.2.1/share/hadoop/tools/lib/hadoop-streaming-3.2.1.jar
/opt/hadoop-3.2.1/share/hadoop/tools/sources/hadoop-streaming-3.2.1-test-sources.jar
/opt/hadoop-3.2.1/share/hadoop/tools/sources/hadoop-streaming-3.2.1-sources.jar
```

FIGURE 1.11 – Recherche du fichier hadoop-streaming

Ensuite, les scripts doivent être rendus exécutables :

```
root@2577474ede25:/# chmod u+x mapper.py
root@2577474ede25:/# chmod u+x reducer.py
```

FIGURE 1.12 – Rendre les scripts exécutables

### 1.2.1 Exécuter Hadoop Streaming

```
hadoop jar /opt/hadoop-3.2.1/share/hadoop/tools/lib/hadoop-
streaming-3.2.1.jar \
> -files mapper.py,reducer.py \
> -input input \
> -output output1 \
> -mapper mapper.py \
> -reducer reducer.py
```

FIGURE 1.13 – Exécution de Hadoop Streaming