



# Pratique de Hadoop n°02

## Map Reduce (en Python)

### Table des matières

I.	Objectif du TP .....	2
II.	Map Reduce.....	2
A.	Présentation.....	2
B.	Wordcount.....	2
1.	Création du projet en Python : .....	3
2.	Utilisation de notre Map-Reduce dans Hadoop .....	5
3.	Vérification par les outils Web d'Hadoop .....	8
III.	Exercice suite au TP .....	8

## I. Objectif du TP

- Après le TP02\_Hadoop\_HDFS
- Initiation au Framework Hadoop et au patron MapReduce en Python

## II. Map Reduce

### A. Présentation

Un Job Map-Reduce se compose principalement de deux types de programmes:

- Mappers : permettent d'extraire les données nécessaires sous forme de clef/valeur, pour pouvoir ensuite les trier selon la clef
- Reducers : prennent un ensemble de données triées selon leur clef, et effectuent le traitement nécessaire sur ces données (somme, moyenne, total...)

### B. Wordcount

Nous allons tester un programme MapReduce grâce à un exemple très simple, le WordCount, l'équivalent du HelloWorld pour les applications de traitement de données. Le Wordcount permet de calculer le nombre de mots dans un fichier donné, en décomposant le calcul en deux étapes:

- L'étape de Mapping, qui permet de découper le texte en mots et de délivrer en sortie un flux textuel, où chaque ligne contient le mot trouvé, suivi de la valeur 1 (pour dire que le mot a été trouvé une fois)
- L'étape de Reducing, qui permet de faire la somme des 1 pour chaque mot, pour trouver le nombre total d'occurrences de ce mot dans le texte.



## 1. Création du projet en Python :

Sur le master créer deux fichier python mapper et reducer comme suite :

```
docker exec -it hadoop-master bash
```

**Note:** si votre conteneur n'est pas démarré lancer la commande suivante :

```
docker restart hadoop-master
```

### *a) Création du fichier mapper.py*

Vi mapper.py

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

```
import sys  
import logging  
  
logging.basicConfig(filename='debug.log',level=logging.DEBUG)  
logging.debug("Entering mapper.py")  
for line in sys.stdin:  
    logging.debug("Inside for loop " + line)  
    line = line.strip()  
    for word in line.split(" "):  
        if( len(word) >0 ):  
            print ("%s\t%i" %(word.lower(),1))
```

### *b) Création du fichier reducer.py*

Vi reducer.py

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

```
import sys  
import logging  
  
logging.basicConfig(filename='debug.log', level=logging.DEBUG)  
  
current_word = None  
current_count = 0  
word = None  
  
print "Entering reducer.py"  
for line in sys.stdin:  
    line = line.strip()  
    parts = line.split('\t', 1)  
    if( len(parts) < 2):  
        continue;  
    word = parts[0]  
    count = parts[1]  
    try:  
        count = int(count)  
    except ValueError:  
        continue  
  
    if current_word == word:  
        current_count = current_count + count  
    else:  
        if(current_word):  
            print("%s\t%s" %(current_word, current_count))  
  
        current_count = count  
        current_word = word  
  
if current_word == word:  
    print("%s\t%s" %(current_word, current_count))
```

### c) *Tester notre code en local*

1. Mapper uniquement en lançant la commande suivante :

```
echo "bonjour bonsoir bonjour bonsoir" | python mapper.py
```

Résultat :

```
root@hadoop-master:~# echo "bonjour bonsoir bonjour bonsoir"|python3 mapper.py
bonjour 1
bonsoir 1
bonjour 1
bonsoir 1
```

1. Mapper et Reducer :

*echo "bonjour bonsoir bonjour bonsoir"|python mapper.py |python reducer.py*

Résultat :

```
root@hadoop-master:~# echo "bonjour bonsoir bonjour bonsoir"|python3 mapper.py
python3 reducer.py
bonjour 1
bonsoir 1
bonjour 1
bonsoir 1
```

**Note:** N'oubliez pas que entre les deux fonctionnalités map et reduce, on le sort, lancer la commande suivante :

*echo "bonjour bonsoir bonjour bonsoir"|python mapper.py|sort -k1,1 |python reducer.py*

Résultat :

```
root@hadoop-master:~# echo "bonjour bonsoir bonjour bonsoir"|python3 mapper.py|s
ort -k1,1 |python3 reducer.py
bonjour 2
bonsoir 2
```

## 2. Utilisation de notre Map-Reduce dans Hadoop

1. Lancer les services Hadoop : **./start-hadoop.sh**
2. Suppression du dossier /output (à faire au cas où) : **hdfs dfs -rmkdir /output**
3. Vérifier si le fichier hadoop-streaming-2.7.2.jar existe bien dans le dossier :  
*ls \$HADOOP\_HOME/share/hadoop/tools/lib*
4. S'il n'existe pas, télécharger le jar depuis internet et copier le dans le chemin :  
*\$HADOOP\_HOME/share/hadoop/tools/lib*

**Note :** si vous avez téléchargé le fichier sur votre machine :



- a. Copier les fichiers dans un dossier de partage sous Docker (dans un dossier hadoop par exemple)
  - b. `docker cp hadoop/ hadoop-streaming-2.7.2.jar hadoop-master:/root/ hadoop-streaming-2.7.2.jar`
5. Lancer le mapper et le reducer sur hadoop avec le fichier purchases.txt :
- a. `hadoop fs -mkdir /user/root/output`
  - b. `hadoop fs -mkdir /user/root/intput`
  - c. `hadoop fs -put purchases.txt /user/root/intput`
  - d. `hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.7.2.jar -file /root/mapper.py -mapper "python3 mapper.py" -file /root/reducer.py -reducer "python3 reducer.py" -input /user/root/input/purchases.txt -output /user/root/output/wordcount`
6. Afficher le résultat stocké sur HDFS:

*`hadoop fs -tail output/ wordcount /part-00000`*



Petersburg	8430
Philadelphia	8471
Phoenix	8431
Pittsburgh	8470
Plano	8323
Portland	8367
Raleigh	8345
Reno	8334
Richmond	8388
Riverside	8338
Rochester	8440
Rouge	8396
Sacramento	8597
Saint	8494
San	42110
Santa	8416
Scottsdale	8443
Seattle	8339
Spokane	8356
Sporting	48207
Springs	8534
St.	16881
Stockton	8289
Supplies	48265
Tampa	8400
Toledo	8314
Toys	48463
Tucson	8546
Tulsa	8444
Vegas	16957
Video	48439
Virginia	8465
Visa	174018
Vista	8510
Washington	8477
Wayne	8527
Wichita	8547
Winston-Salem	8459
Women's	48252
Worth	8462
York	8529
and	48408

### 3. Vérification par les outils Web d'Hadoop

Il vous est possible de monitorer vos Jobs Map Reduce, en allant à la page: <http://localhost:8088>. Vous trouverez votre Job dans la liste des applications comme suit:

The screenshot shows the Hadoop web interface at localhost:8088. The 'All Applications' page displays a table of cluster metrics and a list of applications. The application 'application\_1517050438813\_0001' is highlighted in red, indicating it is the job of interest. The table shows the application is in a 'FINISHED' state with a 'SUCCEEDED' final status.

Cluster Metrics															
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	1	0	0 B	16 GB	0 B	0	16	0	2	0	0	0	0

  

Scheduler Metrics													
Scheduler Type		Scheduling Resource Type		Minimum Allocation		Maximum Allocation							
Capacity Scheduler		(MEMORY)		<memory:1024, vCores:1>		<memory:8192, vCores:8>							

  

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
application_1517050438813_0001	root	word count	MAPREDUCE	default	Sat Jan 27 11:58:15 +0100 2018	Sat Jan 27 11:59:36 +0100 2018	FINISHED	SUCCEEDED		History	N/A

Il est également possible de voir le comportement des noeuds esclaves, en allant à l'adresse: <http://localhost:8041> pour **slave1**, et <http://localhost:8042> pour **slave2**. Vous obtiendrez ce qui suit:

The screenshot shows the Hadoop web interface at localhost:8042/node. The 'NodeManager' page displays detailed information about the node's resources and health. The 'NodeManager Information' section shows that the node is healthy and has sufficient resources allocated for containers.

NodeManager Information	
Total Vmem allocated for Containers	16.80 GB
Vmem enforcement enabled	true
Total Pmem allocated for Container	8 GB
Pmem enforcement enabled	true
Total VCores allocated for Containers	8
NodeHealthyStatus	true
LastNodeHealthTime	Sat Jan 27 11:14:00 UTC 2018
NodeManager Version:	2.7.2 from Unknown by root source checksum c63f7cc71b8f63249e35126f0f7492d on 2016-05-27T18:16Z
Hadoop Version:	2.7.2 from Unknown by root source checksum d0fda26633fa762bfb7ec759ebe689c on 2016-05-27T18:05Z

## III. Exercice suite au TP

- Écrire un Job **Map Reduce** permettant, à partir du fichier **purchases initial**, de déterminer le total des ventes par magasin.
- Écrire un Job **Map Reduce** permettant, à partir du fichier **purchases initial**, de déterminer le total des ventes, la moyenne des ventes et nombre des ventes par magasin.

La structure du fichier **purchases** est de la forme suivante:

**date temps magasin produit cout paiement**

- Veiller à toujours tester votre code en local avant de lancer un job sur le cluster.