



Pratique de Hadoop n°02

Map Reduce (en Python)

Table des matières

l.	Objectif du TP	2
	Map Reduce	
	A. Présentation	
В	3. Wordcount	
	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 démarrer lancer la commande suivante :

docker restat 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):</pre>
        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

TP02 Hadoop Map reduce





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
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 -rmdir /output
- 3. Vérifier si le fichier hadoop-streaming-2.7.2.jar existe bien dans le dossier : Is \$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









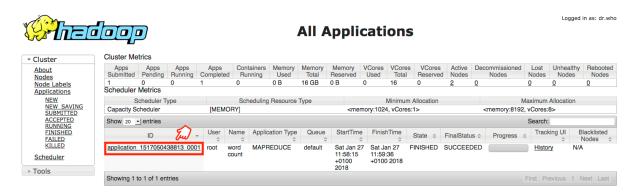
Petersb	urg	8430
Philade		8471
Phoenix		
Pittsbu	rgh	8470
Plano	8323	
Portlan	d	8367
Raleigh	8345	
Reno	8334	
Richmon	d	8388
Riversi	de	8338
Rochest	er	8440
Rouge	8396	
Sacrame	nto	8597
Saint	8494	
San	42110	
Santa	8416	
Scottsd	ale	8443
Seattle	8339	
Spokane	8356	
Sportin		48207
Springs	8534	
St.	16881	
Stockto		8289
Supplie	S	48265
Tampa	8400	
	004/	
Toledo	8314	
Toledo Toys	48463	
Toys Tucson	48463 8546	
Toys Tucson Tulsa	48463 8546 8444	
Toys Tucson Tulsa Vegas	48463 8546 8444 16957	
Toys Tucson Tulsa Vegas Video	48463 8546 8444 16957 48439	
Toys Tucson Tulsa Vegas Video Virgini	48463 8546 8444 16957 48439	8465
Toys Tucson Tulsa Vegas Video Virgini Visa	48463 8546 8444 16957 48439 a 174018	8465
Toys Tucson Tulsa Vegas Video Virgini Visa Vista	48463 8546 8444 16957 48439 a 174018 8510	
Toys Tucson Tulsa Vegas Video Virgini Visa Vista Washing	48463 8546 8444 16957 48439 a 174018 8510 ton	8465 8477
Toys Tucson Tulsa Vegas Video Virgini Visa Vista Washing Wayne	48463 8546 8444 16957 48439 a 174018 8510 ton 8527	
Toys Tucson Tulsa Vegas Video Virgini Visa Vista Washing Wayne Wichita	48463 8546 8444 16957 48439 a 174018 8510 ton 8527 8547	8477
Toys Tucson Tulsa Vegas Video Virgini Visa Vista Washing Wayne Wichita Winston	48463 8546 8444 16957 48439 a 174018 8510 ton 8527 8547 -Salem	
Toys Tucson Tulsa Vegas Video Virgini Visa Vista Washing Wayne Wichita Winston Women's	48463 8546 8444 16957 48439 a 174018 8510 ton 8527 8547 -Salem 48252	8477
Toys Tucson Tulsa Vegas Video Virgini Visa Vista Washing Wayne Wichita Winston Women's Worth	48463 8546 8444 16957 48439 a 174018 8510 ton 8527 8547 -Salem 48252 8462	8477
Toys Tucson Tulsa Vegas Video Virgini Visa Vista Washing Wayne Wichita Winston Women's	48463 8546 8444 16957 48439 a 174018 8510 ton 8527 8547 -Salem 48252	8477



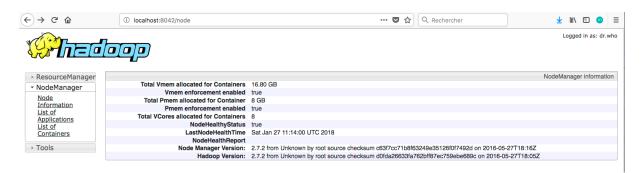


3. Vérification par les outils Web d'Hadoop

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:



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:



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.