Initiation au paradigme MapReduce

Guide préparé par Mahran Farhat (farhatmahran@gmail.com)

Objectif :

Cette séance de TP constitue un premier pas vers l'utilisation d'outils Big Data. Nous avons fait le choix de l'environnement Apache Hadoop, car c'est un framework open-source de stockage et de traitement de données volumineuses sur un cluster de machines distribuées. Il est utilisé par un grand nombre de contributeurs et utilisateurs. Il a une licence Apache 2.0.

Vous manipulerez des fichiers dans le système de fichiers distribué HDFS et vous lancerez des traitements MapReduce.

Au terme de ce TP, vous serez capables :

- de développer une chaîne de traitement Hadoop/MapReduce (un Mapper et un Reducer) en Python,
- de tester en local sur votre propre machine si la chaîne de traitement est conforme à vos attentes,
- de déployer et exécuter votre chaîne de traitement MapReduce sur une machine virtuelle simulant un cluster Hadoop,
- de récupérer vos résultats d'analyse en local sur votre machine

1. Map Reduce

Map Reduce est un patron d'architecture de développement permettant de traiter les données volumineuses de manière parallèle et distribuée. Il se compose principalement de deux types de programmes : Les Mappers et les Reducers. Les Mappers permettent d'extraire les données nécessaires sous forme de clef/valeur, pour pouvoir ensuite les trier selon la clef. Les Reducers prennent un ensemble de données triées selon leur clef, et effectuent le traitement nécessaire sur ces données (somme, moyenne, total...).

Pour notre TP, nous utilisons le langage Python pour développer les Mappers et les Reducers. Les traitements intermédiaires (comme le tri par exemple) sont effectués automatiquement par Hadoop.

1.1 Mapper

Soit un fichier de texte comportant plusieurs lignes. Pour ce premier exercice, notre but est de déterminer le total d'occurrence de chaque mot dans notre fichier texte. Pour cela, le mapper doit :

- Diviser le texte en des lignes

- Diviser chaque ligne en un ensemble de mots
- Retourner les différents mots du texte

Pour calculer le nombre d'occurrence des mots, le code du Mapper est le suivant :

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

import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```

1.2 Reducer

Le Reducer permet de faire le traitement désiré sur des entrées sous forme de clef/valeur, préalablement triées par Hadoop (on n'a pas à s'occuper du tri manuellement). Dans l'exemple précédent, une fois que le Mapper extrait les couples (store,cost), le Reducer aura comme tâche de faire la somme de tous les coûts pour un même magasin. Le code du Reducer est le suivant :

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

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        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)
```

Travail à faire :

Aller dans le dossier **udacity_training,** ensuite :

1 : créer un fichier texte nommé isikef.txt dont le contenu est el suivant :

Hello World Bye World
Hello Hadoop Goodbye Hadoop

2 : charger ce fichier sur hadoop

3 : créer un dossier wordcount

4 : créer dans le dossier wordcount les programmes python mapper.py et reducer.py dont le code déjà donné.

5 : exécuter votre Job Map reduce

6 : récupérer et consulter votre résultat à partir de cluster hadoop

Pour cela, je vous invite à taper les commandes suivantes :

```
cd udacity_training
```

Créer le fichier isikef.txt et modifier son contenu

```
gedit isikef.txt
```

Consulter le contenu de hadoop

```
hadoop fs -ls
```

Créer un dossier sur hadoop

```
hadoop fs -mkdir myinput
```

Charger le fichier sur hadoop

```
hadoop fs -put isikef.txt myinput
```

Afficher le contenu du dossier myinput

```
hadoop fs -ls -h myinput
```

Cérer le dossier wordcount

```
mkdir wordcount
```

Aller dans le dossier wordcount

```
cd wordcount
```

Créer votre mapper et spécifier son contenu :

```
gedit mapper.py
```

Créer votre reducer et spécifier son contenu :

```
gedit reducer.py
```

Maintenant, vous êtes capables d'exécuter votre Job map reduce. Pour cela, y a deux méthodes à suivre :

Méthode 1 :

exécution du job à travers l'utilisation de l'alias hs :

```
hs wordcount/mapper.py wordcount/reducer.py myinput outisikef
```

Méthode 2 :

exécution du job à travers une commande complète et en utilisant les fichiers jar :

```
hadoop jar /usr/lib/hadoop-0.20-mapreduce/contrib/streaming hadoop-streaming-2.0.0-mr1
cdh4.1.1.jar -mapper code/mapper.py -reducer wordcount/reducer.py -file code/mapper.py -file
wordcount/reducer.py -input myinput -output outisikef
```

Pour afficher le résultat de votre Job

```
hadoop fs -cat outisikef/part-00000
```

Pour copier le résultat à partir de cluster hadoop sur votre machine :

```
hadoop fs -get outisikef/part-00000 results.txt
```

Pour supprimer les deux dossier myinput et outisikef à partir du cluster hadoop :

```
hadoop fs -rm -r -f myinput
hadoop fs -rm -r -f outisikef
```

Bon travail