TP4 Grid & Cloud Computing

Traitements Map/Reduce avec Hadoop

Préambule : installation standalone d'Hadoop

Nous allons utiliser Hadoop en mode non distribué sur `etud` (ou tout autre machine de l'ISIMA). Cela vous permettra de développer vos codes Java avec votre environnement de développement préféré, et de tester simplement vos algorithmes Map/Reduce.

1. Vérifier votre quota : vous avez besoin d'un minimum de 700Mo d'espace disque
2. Vérifiez que java et la variable d'environnement `JAVA_HOME` sont correctement configurés.
3. Téléchargez le tarball `hadoop-2.8.5.tar.gz` directement depuis le site d'Hadoop ou depuis la plateforme de cours en ligne.
4. Décompressez l'archive dans un répertoire dédié (noté `$TP4DIR` par la suite).
5. Définissez les variables d'environnement `HADOOP_PREFIX` et `HADOOP_CLASSPATH` :

   ```
   $ export HADOOP_PREFIX=$TP4DIR/hadoop-2.8.5
   $ export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
   ```

6. Ajoutez le répertoire `$HADOOP_PREFIX/bin` à votre variable `PATH` :

   ```
   $ export PATH="$HADOOP_PREFIX/bin:$PATH"
   ```

7. Ajoutez les bibliothèques Hadoop à la variable d'environnement CLASSPATH :

   ```
   $ export CLASSPATH="$(yarn classpath):$CLASSPATH"
   ```

8. Vérifiez que votre environnement est correctement configuré (depuis le répertoire `$TP4DIR`) :

   ```
   $ hadoop version
   ```

9. Créez un répertoire `hdfs` dans votre répertoire `$TP4DIR`. Il constituera votre système de fichier HDFS local.

TP : traitement sur deux oeuvres de la littérature française

Récupérez depuis l plateforme en ligne, les 2 jeux de données issus du projet Gutenberg [1] :

- *Les misérables* de Victor Hugo, en 5 fichiers au format texte (vhugo.tar.gz),
- *A la recherche du temps perdu* de Marcel Proust (incomplet), en 9 fichiers au format texte (mproust.tar.gz).

1. Préparation des données Décompressez les archives et déposez les fichiers dans 2 répertoires HDFS différents :

```
$ hadoop fs -mkdir hdfs/vhugo
$ hadoop fs -put lm*.txt hdfs/vhugo
```

2. Word Count Récupérez le code de l'exemple vu en cours (WordCount.java) disponible sur la plateforme de cours en ligne. Compilez le code tel quel et créez un JAR :

```
$ javac WordCount.java
$ jar cf wc.jar WordCount*.class
```

Lancer le traitement MapReduce sur les 2 jeux de données en spécifiant en arguments le répertoire HDFS d'entrée et le répertoire HDFS de sortie (qui ne doit pas exister) :

```
$ hadoop jar wc.jar WordCount  hdfs/vhugo hdfs/output
```

Vous pouvez ensuite visualiser votre sortie avec la commande suivante :

```
$ hadoop fs -ls hdfs/output
$ hadoop fs -cat hdfs/output/part-00000 | head
```

1. `https://www.gutenberg.org/`

3. Richesse du vocabulaire Améliorez le code Map/Reduce précédent pour travailler uniquement avec des mots en minuscules de plus de 3 lettres en minuscule et en supprimant les signes de ponctuations qui biaisent les résultats.
Quel auteur semble avoir le vocabulaire le plus riche ?

4. Recherche d'anagrammes Ecrivez un programme Map/Reduce qui recherche les anagrammes de plus de 4 lettres et testez-le sur les 2 corpus de textes.

5. Style de phrase Ecrivez un programme Map/Reduce qui analyse sommairement le style des phrases en calculant la longueur maximale et la longueur moyenne des phrases (en nombre de mots).
La différence de style de phrase entre les deux auteurs est-elle significative ?