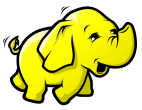
TME 1 - Hadoop, une plate-forme open-source de MapReduce. Installation et prise en main

Jonathan Lejeune, Julien Sopena

Contexte

- ✯ Le modèle *MapReduce* est aujourd'hui l'un des modèles de programmation parallèle les plus utilisés. Définissant une architecture Maître-Esclave, dans laquelle s'exécute une succession d'ensemble de tâches indépendantes, il permet le traitement parallèle de grandes masses de données.
- ✯ Ce TME se propose de mettre en pratique ce modèle au travers d'exemples simples (numériques et textuels). Il repose sur l'utilisation de la plate-forme Hadoop qui est l'implémentation open-source du MapReduce d'Apache (http ://hadoop.apache.org/mapreduce/).



Exercice(s)

Exercice 1 – Installation de la plate-forme Hadoop

Question 1

Pour éviter de taper votre mot de passe à chaque lancement des démons d'Hadoop, créez , si ce n'est déjà fait, un couple de clés ssh grâce à la commande ssh-keygen et ajoutez-la aux clés autorisées.

```
ssh-keygen -q -N '' -f ${HOME}/.ssh/id_rsa
cat ${HOME}/.ssh/id_rsa.pub >> ${HOME}/.ssh/authorized_keys
```

Question 2

La plateforme Hadoop, ainsi que des sources de ce TP se trouvent à l'URL

```
http://pagesperso-systeme.lip6.fr/Jonathan.Lejeune/documents/PSIA-TP_Hadoop.zip
```

Extrayez le contenu de hadoop-1.2.1.tar.gz dans votre home. Attention, vous aurez besoin en tout (dossier Hadoop + ressources TME) d'au moins 50 Mo, utiliser le /tmp si besoin. Pour fonctionner, Hadoop a besoin que la variable d'environnement JAVA_HOME soit définie pour indiquer le repertoire d'une jvm. Ajoutez les lignes suivantes dans votre *.bashrc* :

```
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0_07
export MY_HADOOP_HOME=<votre_repetoire_hadoop>
export PATH=$PATH:$MY_HADOOP_HOME/bin
```

Pour prendre en compte ces modifications dans votre terminal ouvert tapez :

```
source ~/.bashrc
```

Vérifiez que Hadoop est opérationnel en tapant :

```
hadoop version
```

La sortie terminale de cette commande doit être :

```
Hadoop 1.2.1
Subversion https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -r 1503
Compiled by mattf on Mon Jul 22 15:23:09 PDT 2013
From source with checksum 6923c86528809c4e7e6f493b6b413a9a
```

Question 3

La commande *hadoop* que vous venez de lancer, est en fait une commande shell générique capable d'administrer la plateforme, de configurer et/ou d'exécuter des programmes MapReduce. Pour en faciliter l'utilisation, un certain nombre de scripts sont livrés avec elle dans le repertoire *bin*. Ces scripts encapsulent les différents appels au script *hadoop* permettant de réaliser des actions complexes. En pratique, vous aurez besoin des scripts : *start-all.sh* et *stop-all.sh*.
Analysez le fonctionnement de ces deux scripts.

Question 4

Avant de lancer la plate-forme Hadoop, il faut éditer ses fichiers de configuration se trouvant dans le dossier *conf*. Dans ce TP nous commencerons par nous limiter à l'utilisation d'une seule machine (la machine locale). Vous devez donc configurer Hadoop de la façon suivante :

- les fichiers *masters* et *slaves* (utilisés pour automatiser les connections ssh) permettent d'indiquer le nom de la ou des machines utilisées, ici cela sera votre machine de TME qui est :

  ```
  localhost
  ```

- le fichier *conf/hdfs-site.xml* est le fichier de configuration du HDFS. Dans le cadre de ce TME son contenu doit être :

  ```
  <?xml version="1.0"?>
  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
  <configuration>
          <property>
                  <name>dfs.replication</name>
                  <value>1</value>
          </property>
  </configuration>
  ```

  Ceci indique au HDFS que les blocs ne doivent pas être répliqués (au lieu de 3 réplications par défaut).

- le fichier *conf/core-site.xml* est le fichier général de configuration de la plate-forme . Dans le cadre de ce TME son contenu doit être :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Ceci indique l'URL du processus serveur Namenode du HDFS. Vérifiez au préalable que le port 9000 de votre machine n'est pas déjà utilisé (commande "netstat -a | grep tcp | grep LISTEN | grep 9000").

– le fichier *conf/mapred-site.xml* est le fichier de configuration des fonctionnalités map-reduce de Hadoop. Dans le cadre de ce TME son contenu doit être :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:9001</value>
  </property>
</configuration>
```

Ceci indique le nom de la machine ainsi que le port réseau associé sur lequel le processus Jobtracker va s'exécuter. Vérifiez au préalable que le port 9001 de votre machine n'est pas déjà utilisé (commande "netstat -a | grep tcp | grep LISTEN | grep 9001").

Question 5
Est-il possible d'avoir plusieurs esclaves (DataNode + TaskTracker) sur une seule machine hôte ? Pourquoi ?

Exercice 2 – Wordcount, le helloworld du MapReduce

Question 1
Dans cet exercice, on va réaliser un programme Hadoop permettant de compter les mots d'un texte. Pour commencer nous allons générer trois fichiers *loremIpsum-100*, *loremIpsum-75* et *loremIpsum-25* de respectivement 100 Mo, 75 Mo et 25 Mo. Pour cela vous utiliserez trois fois le script *generator.sh* et le fichier *loremIpsum* fournis dans les ressources du TME de la faÃ§on suivante :

```
./generator.sh loremIpsum 100
./generator.sh loremIpsum 75
./generator.sh loremIpsum 25
```

Question 2
Afin d'utiliser *Eclipse* pour éditer vos programmes de MapReduce, il faut ajouter les différents packages d'Hadoop dans le *Java Build Path* de vos projets.
Pour ce faire, après avoir créé un nouveau projet dans votre workspace, vous ajouterez les différents fichiers *jar* présents à la racine du dossier Hadoop ainsi que ceux présents dans le dossier *lib* à son *Java Build Path* :

– Sélectionnez *Build Path* dans le menu contextuel (clic droit sur le dossier) du projet puis *Configure Build Path*
– Dans l'onglet *Libraries*, cliquez sur *Add External JARs...*.
– Ajoutez l'ensemble des *Jar* d'Hadoop présents à la racine du dossier de la plateforme ainsi que ceux du répertoire *lib*.

– Vérifiez que les Jar ont bien été ajoutés dans le repertoire *Referenced Libraries* dans votre projet Eclipse.

Vous êtes maintenant prêt à coder votre premier programme MapReduce :).

Question 3
Copiez dans le dossier des sources de votre projet Eclipse le fichier Wordcount.java qui vous a été fourni dans les ressources du TME. En intégrant ce fichier dans Eclipse, s'assurer que la compilation se passe bien (absence de croix rouge).

Question 4
Identifiez les différents types de clés et valeurs des entrées et sorties des maps et des reduces.

Question 5
Nous allons maintenant démarrer la plate-forme, c'est à dire lancer l'ensemble des démons :

1. Formatez le HDFS :

   ```
   hadoop namenode -format
   ```

2. Démarrez Hadoop :

   ```
   start-all.sh
   ```

3. Vérifiez le bon démarrage d'Hadoop avec le script *check_start.sh* qui vous a été fourni dans les ressources du TME.

Une fois la plate-forme lancée, utilisez la commande *ps auxf* pour comprendre quels démons ont été lancés. Identifiez alors le rôle de chacun de ces démons.

Question 6
Créez à partir des fichiers compilés, une archive java *Wordcount.jar*

```
jar cvf <fichier jar a creer> -C <dossier a archiver> <dossier destination>}
```

Question 7
Nous allons maintenant essayer le *Wordcount* sur le fichier *loremIpsum-100* qui vous avez créé dans la question 1 de cet exercice. Préalablement, il faut copier ce fichier de données sur le HDFS. Pour celà tapez la commande :

```
hadoop fs -copyFromLocal <nomFichierLocal> hdfs://localhost:9000/
```

ou bien

```
hadoop fs -copyFromLocal <nomFichierLocal> /
```

La racine de la dernière commande indique la racine du HDFS et non pas la racine de votre machine de TME.

Question 8
Lancez le Wordcount avec le fichier jar produit précédemment :

```
hadoop jar <Fich jar> <nom classe main> <fich entree> <dossier sortie>
```

Notons que les chemins d'entrée et de sortie sont des chemins HDFS et non des chemins du système de fichier local. Pendant que le job s'exécute, recontrolez les processus avec la commande *ps auxf*. Vous devez remarquer qu'il y a un processus java en plus dont le père est le tastracker. Expliquez le rôle et l'intéret de ce processus.

Question 9
Une fois que le job se termine, contrôlez la sortie dans le dossier que vous avez spécifié. Quelle est le format utilisé pour les noms des fichiers de sortie ?

Question 10
Identifiez le nombre de maps et le nombre de reduces que vous avez lancés. Vous trouverez ces informations sur l'affichage que le job produit lorsqu'il se termine.

Question 11
Relancez un job avec le fichier *loremIpsum-25* et comparez le nombre de maps avec le job précédent.

Question 12
Relancez un job avec le fichier *loremIpsum-25* puis *loremIpsum-75* que vous aurez placé dans un dossier et comparez le nombre de maps avec les jobs précédents. Qu'en déduisez vous ?

Question 13
Quelle est la relation entre la donnée d'entrée et le nombre de maps ?

Question 14
Modifiez le fichier *WordCount.java* pour mesurer le temps de calcul d'un job, recompilez-le et réarchivez-le. Relancez d'autres jobs en faisant varier la taille du split. Pour modifier ce paramètre, changer la property *mapred.max.split.size* en ajoutant dans la méthode *main* de la classe *WordCount* la ligne :

```
job.getConfiguration().setLong("mapred.max.split.size", <taille>);
```

Notez que l'unité de cette option est l'octet, la taille par défaut est de 64Mo soient 67108864 o. Observez l'impact sur le temps de calcul. Qu'en concluez-vous ?

