CoDel - 5I451

TP 01 – Prise en main d'Hadoop.

Julien Sopena

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 TP 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 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 grace 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 dans le repertoire */Vrac/Sopena/PSIA-TP_Hadoop*. Sans copier cette archive, extrayez le contenu de hadoop-0.20.203.0rc1.tar.gz dans votre *home*. Attention, vous aurez besoin d'au moins $80Mo$, utiliser le */tmp* si besoin.

Pour fonctionner *Hadoop* a besoin qu'une variable d'environnement `JAVA_HOME` indique le repertoire d'une jvm. Ajoutez les lignes suivantes dans votre `bashrc` :

```
export JAVA_HOME=/usr/lib/jvm/java-6-sun
export HADOOP_HOME=<votre_repetoire_hadoop>
export PATH=$PATH:$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 0.20.203.0
Subversion http://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.20-security-203 -r 1099333
Compiled by oom on Wed May  4 07:57:50 PDT 2011
```

Question 3

La commande *Hadoop* que vous venez de lancer, est en fait une commande shell générique capable d'administrer, de configurer et/ou d'exécuter des programmes *MapReduce*. Pour en facilité l'utilisation un certain nombre de scripts sont livrés avec elle dans le repertoire *bin*. Ces scripts encapsulent les différents appels aux scripts *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'utilisiation d'une seule machine. Vous devez donc configurer *Hadoop* de la façon suivante :

— dans les fichiers **masters** et **slaves** (utilisé pour automatiser les connections ssh) indiquez le nom de la ou des machines utilisées :
`ari-31-201-xx`

— dans **conf/hdfs-site.xml** ajoutez :

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

— dans **conf/core-site.xml** ajoutez (en remplaçant `xx` par le numéro de votre machine) :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs ://ari-31-201-xx :9000</value>
  </property>
</configuration>
```

— dans **conf/mapred-site.xml** ajoutez (en remplaçant `xx` par le numéro de votre machine) :

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>ari-31-201-xx :9001</value>
  </property>
</configuration>
```

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 de la façon 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* 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. 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 checkStart.sh qui vous a été fourni dans les ressources du TME.

```
./check_start.sh
```

Question 6

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 7

Créez à partir des fichiers compilés une archive java `WordCount.jar` :

```
jar cvf <fichier jar à créer> -C <dossier à archiver> <dossier destination>}
```

Question 8

Nous allons maintenant essayer le *WordCount* sur le fichier `loremIpsum-100` qui vous a été fourni avec les sources du TME.

Dans un premier temps, il faut copier ce fichier de données sur le HDFS. Pour cela, on peut utiliser l'une des deux commandes suivantes :

```
hadoop fs -copyFromLocal <nomFichierLocal> hdfs://ari-31-201-xx:9000/
hadoop fs -copyFromLocal <nomFichierLocal> /
```

ATTENTION : Dans la deuxième commande, la racine correspond à la racine du *HDFS* et non pas la racine du système de votre machine de TME.

Question 9

Maintenant que les données se trouve sur le HDFS, on va pouvoir lancer le *Wordcount*. Lancez la commande suivante utilisant le fichier jar produit précédemment.

```
hadoop jar <Fichier jar> <nom de la classe main> <dossier d'entrée> <dossier de sortie>
```

Pendant que le job s'exécute, recontrôlez 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'intérêt de ce processus.

Question 10

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 11

Identifiez le nombre de *maps* et le nombre de *reduces* que vous avez lancés.

Question 12

Relancez un job avec le fichier `loremIpsum-75` et comparez le nombre de *maps* avec le job précédent.

Question 13

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

Question 14

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

Question 15

Modifiez le fichier *WordCount.java* pour mesurer le temps de calcul d'un job, recompilez-le et réarchivez-le. Lancez alors plusieurs jobs en faisant varier la taille du split. Pour modifier ce paramètre, il faut changer la `property mapred.max.split.size` en ajoutant dans la méthode main de la classe `WordCount` la ligne suivante. Notez que l'unité de cette option est l'octet, la taille par défaut est de 64Mo soient 67108864 octets :

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

Observez l'impact sur le temps de calcul. Qu'en concluez-vous ?

by Julien SOPENA *(http://julien.sopena.fr)*