RAPPORT PROJET BIG-DATA

Etude des performances d'Apache Hadoop et de twitter storm pour une application : Cas de WordCount

5GI, ENSP

20 novembre 2015

Créé par : Dokmegang Joel

Table des matières

1 Introduction

Hadoop est un framework java, open-source, développé par apache dans le but de traiter de larges quantités de données réparties dans un environnement distribué. Hadoop possède naivement plusieurs options de configuration. Dépendamment de ces configurations, les organisations et particuliers peuvent expérimenter des performances exceptionnelles ou de piètres rendements. Il est donc important de savoir configurer hadoop dans le but d'obtenir des performances optimales. Dans ce document, nous essayons, au travers des configurations, d'obtenir les performances maximales d'une implantation avec hadoop d'une application de comptage de mots. Pour ce faire, nous allons tour à tour parler de l'installation et des configurations d'hadoop, de la préparation de nos tests, ainsi que des résultats des tests et les interprétations que nous leur donnons.

2 Installation et configurations

2.1 Prérequis

2.1.1 Plateformes

- Les plateformes **GNU/Linux** sont supportées comme plateformes de développement et de production. Windows est également supportée comme plateforme de développement.
  - Nos expérimentations ne se basent que sur Linux, notamment **Ubuntu 14.04**

2.1.2 Logiciels

- **Java** doit être installée. Les versions recommandées de java sont décrites à l'adresse http://wiki.apache.org/hadoop/HadoopJavaVersions
  - Nous utilisons la version **1.8.0_45 de la jdk**
- **ssh** doit être installé et le service **ssh** démarré afin d'utiliser les scripts hadoop qui agissent sur les démons hadoop distants
  - Nous utilisons la version **openSSH_6.6.1p1**

2.1.3 Installation des logiciels requis

Si vous n'avez pas les logiciels requis, vous devez les installer. Sur Ubuntu, utilisez les commandes suivantes :

```
$ sudo apt-get install openjdk
$ sudo apt-get install ssh
$ sudo apt-get install rsync
```

2.2 Téléchargement et installation

2.2.1 Téléchargement

La dernière version stable de hadoop est téléchargeable à l'adresse http://www.apache.org/dyn/closer.cgi/hadoop/common/.

Télécharger la et décompressez-la à l'emplacement du disque où vous désirez que hadoop soit installé.

Pour nos expérimentations, nous avons utilisé la version **2.7.1 de hadoop**

2.2.2 Configurations globales du système

A ce niveau, il s'agit majoritairement de la configuration de certaines variables et de leur ajout dans la variable d'environnement. Les commandes ci-dessous feront le travail.

```
# Exporter la variable JAVA_HOME
$ export JAVA_HOME=chemin_vers_dossier_installation_java

# Ajouter JAVA_HOME à la variable d'environnement
$ export PATH=$JAVA_HOME/bin:$PATH

# Exporter la variable HADOOP_HOME
$ export HADOOP_HOME=
chemin_vers_dossier_installation_hadoop

# Ajouter le dossier bin de hadoop à la variable
# d'environnement
$ export PATH=$HADOOP_HOME/bin:$PATH

# Ajouter le dossier sbin de hadoop à la variable
d'environnement
$ export PATH=$HADOOP_HOME/sbin:$PATH

# Ajouter l'archive tools.jar de java à la variable
d'environnement
$ export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
```

Pour simplifier la tâche, vous pouvez utiliser le script « **sys-conf.sh** » livré avec ce document pour effectuer toutes ces tâches. Vous l'utiliserez de la manière suivante :

```
$ sh sys-conf.sh chemin_vers_dossier_installation_java
chemin_vers_dossier_installation_hadoop
```

Ayant terminé l'installation et les configurations globales du système, nous pouvons à présent démarrer hadoop. Il existe 03 modes de démarrage d'hadoop :

- ✓ Démarrage en mode « standalone »
- ✓ Démarrage en mode « pseudo-distributed »
- ✓ Démarrage en mode « fully-distributed »

2.3 Démarrage

2.3.1 Démarrage en mode « standalone »

Hadoop est configuré par défaut pour s'exécuter dans ce mode. La commande suivante permet de démarrer les daemons de hadoop:

```
# Démarrer les deamons du namenode et du datanode
$ start-dfs.sh
```

2.3.2 Démarrage en mode « pseudo-distributed »

Hadoop peut aussi s'exécuter sur une seule machine dans un mode pseudo-distribué, dans lequel chaque deamon hadoop s'exécute dans un processus distinct.

2.3.2.1 Configuration

1. Configurer le fichier **dossier_installation_hadoop/etc/hadoop/core-site.xml** comme suit

```
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://localhost:9000</value>
        </property>
</configuration>
```

2. Configurer le fichier **dossier_installation_hadoop/etc/hadoop/hdfs-site.xml** comme suit

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>
</configuration>
```

2.3.2.2 Configurer ssh pour un accès sans mot de passe

Vérifier que vous pouvez accéder au localhost par ssh sans mot de passe. Taper la commande suivante pour le faire

```
$ ssh localhost
```

Si un mot de passe vous est demandé, alors taper les commandes suivante pour configurer l'accès sans mot de passe

```
$ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
$ export HADOOP\_PREFIX=dossier_installation_hadoop
```

2.3.2.3 Démarrer les deamons hadoop

```
# Formatter le système de fichiers
$ hdfs namenode -format
# Démarrer les deamons du namenode et du datanode
$ start-dfs.sh
```

3 Préparation des tests

Nos test ont été réalisés sur un cluster hadoop en mode pseudo-distribué.

3.1 Matériel disponible

Matériel	Caractéristique
Ordinateur	Toshiba satellite C55-A5300
Processeur	Intel® Celeron® CPU 1037U @ 1.80GHz 1.80GHz
RAM	4Go
Plateforme	Ubuntu 14.04
Version de Hadoop	Hadoop 2.7.1
Taille du fichier	996,1 MB (996 105 216 Octets)

3.2 Préparation du hdfs (hadoop file system) et démarrage d'hadoop

3.2.1 Formater le namenode

```
# Formater le système de fichiers hadoop
$ hdfs namenode -format
```

3.2.2 Création des répertoires nécessaires

```
# Créer le répertoire /user/{votre username} dans le hdfs
$ hdfs dfs –mkdir /user
$ hdfs dfs –mkdir /user/$USER
```

3.2.3 Démarrage de hadoop

```
# Démarrer hadoop
$ start-dfs.sh
```

3.3 Préparation de l'application wordcount.java

3.3.1 Code

Pour nos expérimentations, nous avons dû modifier quelque peu le code du programme WordCount fourni afin de pouvoir agir sur certains paramètres et récupérer le temps d'exécution.

La fonction **main** se présente ainsi comme suit :

```
public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();

  Job job = Job.getInstance(conf, "word count");
  job.setJarByClass(WordCount.class);
  job.setMapperClass(TokenizerMapper.class);
  job.setCombinerClass(IntSumReducer.class);
  job.setReducerClass(IntSumReducer.class);
  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  FileInputFormat.addInputPath(job, new Path(args[0]));
  FileOutputFormat.setOutputPath(job, new Path(args[1]));

  long startTime = System.currentTimeMillis();
  Boolean finished = job.waitForCompletion(false);
  long endTime   = System.currentTimeMillis();
  long execTime = endTime - startTime;

  writeExecTime(execTime, "exec_time");

  System.exit(finished ? 0 : 1);
}
```

3.3.2 Compilation

Utiliser les commandes suivantes pour compiler le code ci-dessus dans le but d'avoir un exécutable .jar

```
$ hadoop com.sun.tools.javac.Main WordCount.java
$ jar cf wc.jar WordCount*.class
```

3.3.3 Exécution

L'exécution est la phase la plus délicate de notre travail, car c'est à ce niveau qu'il faut décider de quels paramètres faire varier, ou encore dans quelle plage de valeurs les faire varier, afin d'obtenir des résultats significatifs. Nous avons au départ désiré de faire varier les paramètres suivants :

- ✓ La taille des blocks
- ✓ Degré de replication du fichier
- ✓ Le nombre de mappers
- ✓ Le nombre de reducers

Cependant, au cours de nos expérimentations, nous n'avons pu faire varier que la **taille des blocs**. Ce fait est dû à des raisons diverses.

Nous utilisons le code suivant pour l'exécution d'un test avec une valeur précise de ce paramètre

```
# Mettre le fichier duquel il faut compter les mots dans
# le système de fichiers hadoop
$ hdfs dfs -put nom_du_fichier input

# Changer la taille des blocks du fichier
$ hadoop fs -
Ddfs.block.size=$((taille_des_blocs*1024*1024)) -cp -f
input input

# Supprimer le repertoire "output" contenant éventuellement
# les résultats des précédentes exécutions de WordCount
hdfs dfs -rm -r hdfs:///user/joel/output

# Lancer le programme WordCount le fichier destination
# copié plus haut
hadoop jar wc.jar WordCount input output
```

L'ensemble de ces commandes est contenu dans le fichier de script **countwords.sh** fourni avec ce rapport. Il prend les paramètres suivants :

- ✓ Le nom du fichier
- ✓ La taille (en Mega octet) des blocs que l'on veut

Pour exécuter tous les tests, nous bouclons sur l'ensemble choisi des valeurs possibles de ces paramètres. Le code est le suivant :

```
for i in `seq 1 20`;
do
        sh countwords.sh input $(($i*32))
done
```

Ce petit bout de code est contenu dans le fichier **wc_simulation.sh.**

4 Résultats des tests et interprétations

4.1 Résultats

Pour exécuter les tests, il suffit de d'exécuter le fichier **wc_simulation.sh**.

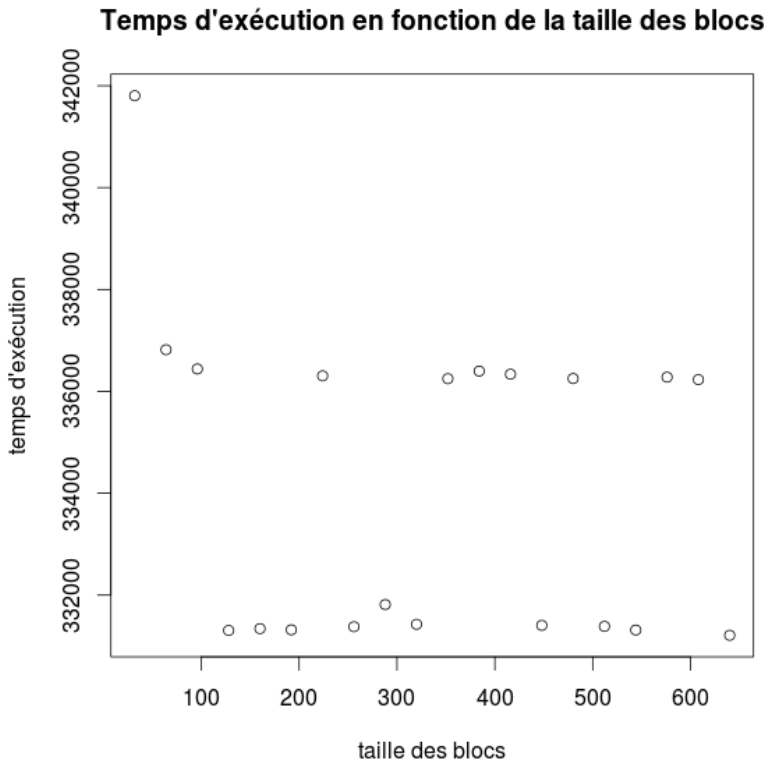
Celui se contente d'exécuter le programme de comptage des mots et de garder à chaque fois le temps mis dans un fichier. Il varie la taille des blocs de 32Mo à 640Mo avec un pas de 32Mo.

A la fin de son exécution, nous avons les résultats suivants écrits dans le fichier **results.csv**:

block_size	exec_time
32	341805
64	336819
96	336441
128	331308
160	331343
192	331320
224	336305
256	331381
288	331816
320	331428
352	336250
384	336399
416	336338
448	331407
480	336253
512	331387
544	331316
576	336280
608	336232
640	331212

4.2 Interprétation des résultats

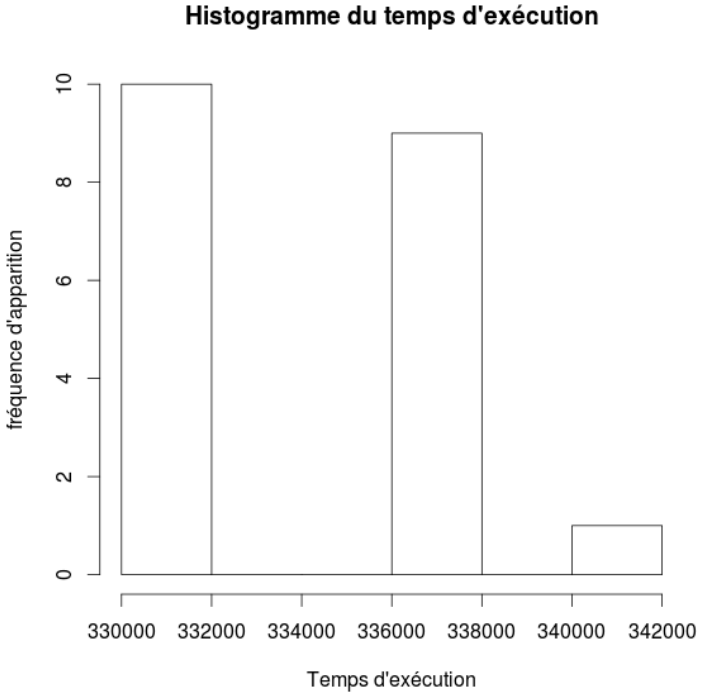
Afin de mieux interpréter les données obtenues, nous avons tracé le graphe représentant le temps d'exécution en fonction de la taille des blocs. Ce graphique est le suivant :



Ce graphique nous montre 03 tranches de temps d'exécution :

- ✓ Temps d'exécution minimal (en secondes) : [330, 332]
- ✓ Temps d'exécution moyen (en secondes) : [336, 338]
- ✓ Temps d'exécution maximal (en secondes) : [340, 342]

L'histogramme ci-dessous l'explicite encore mieux.



Ayant regroupé les données de la sorte, nous pouvons dresser ce tableau d'analyse :

	Multiple de 32	Multiple de 64	Multiple de 128	Puissance de 2
Total	20	10	5	5
TE max.	1	0	0	1
TE moy.	9	3	1	1
TE min.	10	7	4	3
Total	20	10	5	5
% TE min.	50%	70%	**80%**	60%

Nous constatons que hadoop exécute le programme WordCount en un temps minimal pour des tailles de blocs **qui sont multiples de 128 Mo**.

Remarquons que ce nombre n'est pas sans rappeler la taille par défaut des blocs sous hadoop.

5 Conclusion

L'objectif du travail présenté ici était d'obtenir les performances maximales pour une implantation d'un programme de comptage des mots avec hadoop. Pour y arriver, plusieurs étapes ont été franchies. Dans ce rapport, nous avons détaillé ces étapes depuis l'installation des logiciels nécessaires jusqu'à l'obtention et l'interprétation des résultats. Avec les moyens dont nous disposions, et de par notre procédure de test, nous sommes arrivés à la conclusion selon laquelle les performances de notre application étaient maximales pour des tailles de blocs multiples de 128 Mo.

Cependant, nous nous sommes heurtés au fait que l'application séquentielle de comptage des mots appliqué au même fichier de test s'exécutait environ 2 fois plus vite qu'une implantation hadoop de la même application. Ce fait suggère peut-être que la taille de notre fichier est trop petite pour réellement apprécier les performances de hadoop ou encore que les éléments matériels à notre disposition étaient limités.