

# Importation et exportation de données dans HDFS

## Introduction

Dans une installation type, Hadoop se trouve au cœur d'un flux de données complexe. Ces données proviennent souvent de systèmes disparates et sont ensuite importées dans le système de fichiers distribué d'Hadoop (HDFS, *Hadoop Distributed File System*). Puis un traitement leur est appliqué à l'aide de MapReduce ou de l'un des nombreux langages construits au-dessus de MapReduce (Hive, Pig, Cascading, etc.). Enfin, les résultats filtrés, transformés et agrégés sont exportés vers un ou plusieurs systèmes externes.

Prenons comme exemple concret celui d'un grand site web pour lequel nous voulons produire des données analytiques de base quant à son audience. Les données de journalisation fournies par plusieurs serveurs sont collectées et stockées dans HDFS. Un job MapReduce est démarré, avec pour entrée les journaux web. Les données sont analysées, résumées et associées à des informations de géolocalisation d'adresse IP. La sortie produite montre pour chaque cookie l'URL, les pages consultées et la localisation. Ce rapport est exporté dans une base de données relationnelles. Des requêtes peuvent alors être lancées sur ces données. Les analystes sont capables de produire rapidement des rapports sur le nombre total de cookies uniques, les pages les plus fréquentées, la répartition des visiteurs par région ou toute autre conclusion sur ces données.

Les solutions décrites dans ce chapitre se focalisent sur l'importation et l'exportation de données depuis et vers HDFS. Les sources et les destinations comprennent notamment le système de fichiers local, des bases de données relationnelles, NoSQL ou distribuées, et d'autres clusters Hadoop.

## Importer et exporter des données à l'aide de commandes du shell Hadoop

La plupart des fonctionnalités de HDFS sont disponibles au travers de commandes du shell, qui se fondent sur l'API du système de fichiers. Hadoop dispose d'un script shell qui sert d'interface pour toutes les interactions depuis la ligne de commande. Ce script se nomme `hadoop` et se trouve généralement dans le répertoire indiqué par `$HADOOP_BIN`, c'est-à-dire le dossier des binaires d'Hadoop. Pour des raisons de commodité, il est préférable d'ajouter `$HADOOP_BIN` à la variable d'environnement `$PATH`. Toutes les commandes du shell pour le système de fichiers d'Hadoop ont le format suivant :

```
hadoop fs -COMMANDE
```

Pour obtenir la liste complète des commandes, il suffit d'exécuter le script `hadoop` en lui passant l'option `fs`, sans mention d'une commande (voir Figure 1.1) :

```
hadoop fs
```

```
[cloudera@localhost Desktop]$ hadoop fs
Usage: hadoop fs [generic options]
    [-cat [-ignoreCrc] <src> ...]
    [-chgrp [-R] GROUP PATH...]
    [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
    [-chown [-R] [OWNER][:[GROUP]] PATH...]
    [-copyFromLocal <localsrc> ... <dst>]
    [-copyToLocal [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-count [-q] <path> ...]
    [-cp <src> ... <dst>]
    [-df [-h] [<path> ...]]
    [-du [-s] [-h] <path> ...]
    [-expunge]
    [-get [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-getmerge [-nl] <src> <localdst>]
    [-help [cmd ...]]
    [-ls [-d] [-h] [-R] [<path> ...]]
    [-mkdir [-p] <path> ...]
    [-moveFromLocal <localsrc> ... <dst>]
    [-moveToLocal <src> <localdst>]
    [-mv <src> ... <dst>]
    [-put <localsrc> ... <dst>]
    [-rm [-f] [-r|-R] [-skipTrash] <src> ...]
    [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
    [-setrep [-R] [-w] <rep> <path/file> ...]
    [-stat [format] <path> ...]
    [-tail [-f] <file>]
    [-test [-ezd] <path>]
    [-text [-ignoreCrc] <src> ...]
    [-touchz <path> ...]
    [-usage [cmd ...]]
```

**Figure 1.1**

Liste des commandes reconnues par le script.

Les noms des commandes et leurs fonctionnalités ressemblent énormément à celles du shell Unix. L'option `help` fournit des informations complémentaires sur une commande précise (voir Figure 1.2) :

```
hadoop fs -help ls
```

```
[cloudera@localhost Desktop]$ hadoop fs -help ls
-ls [-d] [-h] [-R] [<path> ...]:      List the contents that match the specified file pattern. If
path is not specified, the contents of /user/<currentUser>
will be listed. Directory entries are of the form
    dirName (full path) <dir>
and file entries are of the form
    fileName(full path) <r n> size
where n is the number of replicas specified for the file
and size is the size of the file, in bytes.
-d Directories are listed as plain files.
-h Formats the sizes of files in a human-readable fashion
  rather than a number of bytes.
-R Recursively list the contents of directories.
[cloudera@localhost Desktop]$
```

**Figure 1.2**

*Aide sur une commande.*

La documentation officielle, consultable à l'adresse <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html>, recense les commandes du shell et en donne de courtes descriptions.

Dans cette section, nous allons utiliser les commandes du shell Hadoop de façon à importer des données dans HDFS et à exporter des données à partir de HDFS. Ces commandes sont souvent employées pour charger des données appropriées, à télécharger des données traitées, à maintenir le système de fichiers et à consulter le contenu des dossiers. Il est donc indispensable de les connaître pour exploiter efficacement HDFS.

## Préparation

Récupérez le jeu de données `weblog_entries.txt` fourni avec les exemples de cet ouvrage, disponibles sur le site web Pearson (<http://www.pearson.fr>).

## Réalisation

Déroulez les étapes suivantes de façon à créer un dossier dans HDFS et copiez le fichier `weblog_entries.txt` à partir du système de fichiers local vers HDFS :

1. Créez un dossier dans HDFS pour y stocker le fichier `weblog_entries.txt` :

```
hadoop fs -mkdir /data/weblogs
```

2. À partir du système de fichiers local, copiez `weblog_entries.txt` dans le dossier créé lors de l'étape précédente :

```
hadoop fs -copyFromLocal weblog_entries.txt /data/weblogs
```

3. Affichez les informations qui concernent le fichier `weblog_entries.txt` (voir Figure 1.3) :

```
hadoop fs -ls /data/weblogs/weblog_entries.txt
```

```
[cloudera@localhost Desktop]$ hadoop fs -ls /data/weblogs
Found 1 items
-rw-r--r-- 1 cloudera supergroup      254129 2012-12-31 11:06 /data/weblogs/weblog_entries.txt
[cloudera@localhost Desktop]$
```

**Figure 1.3**

*Informations à propos du fichier `weblog_entries.txt`.*

### NOTE

Les résultats d'un job exécuté dans Hadoop peuvent être exploités par un système externe ou peuvent nécessiter un traitement supplémentaire par un ancien système, mais il est également possible que les contraintes de traitement ne correspondent pas au paradigme de MapReduce. Dans tous ces cas, les données devront être exportées à partir de HDFS. Pour cela, l'une des solutions les plus simples consiste à utiliser le shell Hadoop.

4. La commande suivante copie le fichier `weblog_entries.txt` depuis HDFS vers le dossier courant du système de fichiers local (voir Figure 1.4) :

```
hadoop fs -copyToLocal /data/weblogs/weblog_entries.txt
â ./weblog_entries.txt
```

```
[cloudera@localhost data]$ hadoop fs -copyToLocal /data/weblogs/weblog_entries.txt ./w
eblog_entries.txt
[cloudera@localhost data]$ ls -ltr
total 252
-rwxr-xr-x 1 cloudera cloudera 254129 Dec 31 11:15 weblog_entries.txt
[cloudera@localhost data]$
```

**Figure 1.4**

*Copie de `weblog_entries.txt` sur le système de fichiers local à partir de HDFS.*

Lorsque vous copiez un fichier à partir de HDFS vers le système de fichiers local, n'oubliez pas de vérifier que l'espace local disponible est suffisant et tenez compte de la rapidité de la connexion réseau. Il n'est pas rare que les fichiers placés dans HDFS aient une taille de plusieurs téraoctets. Avec une connexion à 1 Gb et à condition de disposer d'un espace suffisant, il faudra presque vingt-trois heures pour recopier sur le système de fichiers local un fichier de 10 To stocké sur HDFS !

## Explications

Les commandes du shell Hadoop constituent une enveloppe autour des API du système de fichiers HDFS. En réalité, le lancement du script shell hadoop avec l'option `fs` fixe le point d'entrée de l'application Java à la classe `org.apache.hadoop.fs.FsShell`. Cette classe instancie un objet `org.apache.hadoop.fs.FileSystem` et relie les méthodes du système de fichiers aux arguments passés à `fs`. Par exemple, `hadoop fs -mkdir /data/weblogs` équivaut à l'instruction `FileSystem.mkdirs(new Path("/data/weblogs"))`. De manière comparable, `hadoop fs -copyFromLocal weblog_entries.txt /data/weblogs` est équivalent à `FileSystem.copyFromLocal(new Path("weblog_entries.txt"), new Path("/data/weblogs"))`. Les mêmes correspondances s'appliquent lors de la copie des données à partir de HDFS vers le système de fichiers local. La commande `copyToLocal` exécutée par le shell Hadoop devient `FileSystem.copyToLocal(new Path("/data/weblogs/weblog_entries.txt"), new Path("./weblog_entries.txt"))`. Pour de plus amples informations sur la classe `FileSystem` et ses méthodes, consultez la page Javadoc officielle à l'adresse <http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/fs/FileSystem.html>.

La commande `mkdir` prend la forme générale suivante :

```
hadoop fs -mkdir CHEMIN1 CHEMIN2
```

Par exemple, `hadoop fs -mkdir /data/weblogs/12012012 /data/weblogs/12022012` crée deux dossiers dans HDFS : `/data/weblogs/12012012` et `/data/weblogs/12022012`. La commande `mkdir` retourne `0` en cas de succès et `-1` en cas d'erreur (voir Figure 1.5) :

```
hadoop fs -mkdir /data/weblogs/12012012 /data/weblogs/12022012
hadoop fs -ls /data/weblogs
```

```
[cloudera@localhost data]$ hadoop fs -mkdir /data/weblogs/12012012 /data/weblogs/12022012
[cloudera@localhost data]$ hadoop fs -ls /data/weblogs
Found 3 items
drwxr-xr-x - cloudera supergroup          0 2012-12-31 11:18 /data/weblogs/12012012
drwxr-xr-x - cloudera supergroup          0 2012-12-31 11:18 /data/weblogs/12022012
-rw-r--r-- 1 cloudera supergroup    254129 2012-12-31 11:06 /data/weblogs/weblog_entries.txt
[cloudera@localhost data]$
```

**Figure 1.5**

*Création de deux dossiers dans HDFS.*

Voici le format général de la commande `copyFromLocal` :

```
hadoop fs -copyFromLocal CHEMIN_FICHIER_LOCAL URI
```

Si l'URI n'est pas indiqué explicitement, un URI par défaut est utilisé. Cette valeur par défaut peut être modifiée en fixant la propriété `fs.default.name` dans le fichier `core-site.xml`. `copyFromLocal` retourne `0` en cas de succès et `-1` en cas d'erreur.

La commande `copyToLocal` prend la forme suivante :

```
hadoop fs -copyToLocal [-ignorecrc] [-crc] URI CHEMIN_FICHIER_LOCAL
```

Si l'URI n'est pas précisé, une valeur par défaut est utilisée. Celle-ci est donnée par la propriété `fs.default.name` du fichier `core-site.xml`. La commande `copyToLocal` utilise un *contrôle de redondance cyclique* (CRC) pour s'assurer que les données copiées n'ont pas été modifiées. Une copie erronée peut néanmoins être acceptée en précisant l'option facultative `-ignorecrc`. Avec l'argument facultatif `-crc`, le fichier et son CRC sont copiés.

## Pour aller plus loin

La commande `put` est comparable à `copyFromLocal`. Elle est un tantinet plus généraliste et permet de copier plusieurs fichiers sur HDFS, ainsi que lire une entrée à partir de `stdin`.

La commande `get` du shell Hadoop peut être employée à la place de `copyToLocal`. Pour le moment, elles partagent le même code d'implémentation.

Lorsque les jeux de données manipulés sont volumineux, la sortie produite par un job sera constituée d'une ou de plusieurs parties. Leur nombre est déterminé par la propriété `mapred.reduce.tasks`, qu'il est possible de modifier à l'aide de la méthode `setNumReduceTasks()` de la classe `JobConf`. Il y aura un fichier de sortie pour chaque tâche de réduction. Puisque le nombre de réducteurs employés varie d'un job à l'autre, cette propriété doit être fixée au niveau non pas du cluster mais du job. Sa valeur par défaut est 1. Autrement dit, la sortie de toutes les tâches `map` sera envoyée à un seul réducteur. À moins que la taille de la sortie cumulée de toutes les tâches `map` soit relativement réduite, moins d'un gigaoctet, la valeur par défaut ne doit pas être utilisée. Trouver le nombre optimal de tâches `reduce` tient plus de l'art que de la science. La documentation `JobConf` recommande d'employer l'une des deux formules suivantes :

- $0,95 \times \text{Nombre de nœuds} \times \text{mapred.tasktracker.reduce.tasks.maximum}$
- $1,75 \times \text{Nombre de nœuds} \times \text{mapred.tasktracker.reduce.tasks.maximum}$

Par exemple, si le cluster comprend dix nœuds qui exécutent un questionnaire de tâches et si la propriété `mapred.tasktracker.reduce.tasks.maximum` est fixée de façon à avoir au maximum cinq slots `reduce`, la première formule devient  $0,95 \times 10 \times 5 = 47,5$ . Puisque le nombre de slots `reduce` doit être un entier positif, il faut arrondir ou tronquer cette valeur.

La documentation `JobConf` explique comment choisir le multiplicateur ([http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/JobConf.html#setNumReduceTasks\(int\)](http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/JobConf.html#setNumReduceTasks(int))) :

*Avec 0,95, tous les réducteurs peuvent être lancés immédiatement et commencent à transférer les sorties dès que les tâches `map` sont terminées. Avec 1,75, les nœuds*

*les plus rapides termineront leur opération de réduction et lanceront une seconde série de réductions, améliorant ainsi la répartition de la charge.*

Il est possible de faire référence à la sortie partitionnée dans HDFS en utilisant le nom de dossier. Un job qui reçoit le nom du dossier lira chaque fichier de sortie lors du traitement. Le problème vient des commandes `get` et `copyToLocal`, qui opèrent uniquement sur des fichiers et ne peuvent donc pas copier des dossiers. Puisqu'il serait relativement pénible et inefficace de copier chaque fichier de sortie (il peut en exister des centaines, voire des milliers) et de les fusionner localement, le shell Hadoop propose la commande `getmerge` pour réunir toutes les parties de la sortie en un seul fichier et pour copier celui-ci sur le système de fichiers local.

Le script Pig suivant illustre l'utilisation de la commande `getmerge` :

```
weblogs = load '/data/weblogs/weblog_entries.txt' as
    (md5:chararray,
     url:chararray,
     date:chararray,
     time:chararray,
     ip:chararray);

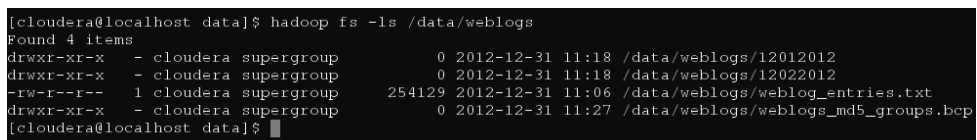
md5_grp = group weblogs by md5 parallel 4;

store md5_grp into '/data/weblogs/weblogs_md5_groups.bcp';
```

Voici comment l'exécuter depuis la ligne de commande :

```
pig -f weblogs_md5_group.pig
```

Le script lit chaque ligne du fichier `weblog_entries.txt`. Il regroupe ensuite les données en fonction de la valeur `md5`. `parallel 4` correspond à la manière de fixer le nombre de `mapred.reduce.tasks` dans Pig. Puisque quatre tâches de réduction seront exécutées dans ce job, nous devons nous attendre à la création de quatre fichiers de sortie. Le script place la sortie dans `/data/weblogs/weblogs_md5_groups.bcp` (voir Figure 1.6).



```
[cloudera@localhost data]$ hadoop fs -ls /data/weblogs
Found 4 items
drwxr-xr-x - cloudera supergroup          0 2012-12-31 11:18 /data/weblogs/12012012
drwxr-xr-x - cloudera supergroup          0 2012-12-31 11:18 /data/weblogs/12022012
-rw-r--r--  1 cloudera supergroup    254129 2012-12-31 11:06 /data/weblogs/weblog_entries.txt
drwxr-xr-x - cloudera supergroup          0 2012-12-31 11:27 /data/weblogs/weblogs_md5_groups.bcp
[cloudera@localhost data]$
```

**Figure 1.6**

*Regroupement des fichiers de sortie à l'aide d'un script Pig.*

Notez que `weblogs_md5_groups.bcp` est en réalité un dossier. L'affichage de son contenu est illustré à la Figure 1.7.

```
[cloudera@localhost data]$ hadoop fs -ls /data/weblogs/weblogs_md5_groups.bcp
Found 5 items
-rw-r--r-- 1 cloudera supergroup          0 2012-12-31 11:27 /data/weblogs/weblogs_md5_groups.bcp/
_SUCCESS
-rw-r--r-- 1 cloudera supergroup    85435 2012-12-31 11:27 /data/weblogs/weblogs_md5_groups.bcp/
part-r-00000
-rw-r--r-- 1 cloudera supergroup    91250 2012-12-31 11:27 /data/weblogs/weblogs_md5_groups.bcp/
part-r-00001
-rw-r--r-- 1 cloudera supergroup    87885 2012-12-31 11:27 /data/weblogs/weblogs_md5_groups.bcp/
part-r-00002
-rw-r--r-- 1 cloudera supergroup    90017 2012-12-31 11:27 /data/weblogs/weblogs_md5_groups.bcp/
part-r-00003
[cloudera@localhost data]$
```

**Figure 1.7***Contenu du dossier de regroupement.*

Le dossier `/data/weblogs/weblogs_md5_groups.bcp` contient les quatre fichiers de sortie : `part-r-00000`, `part-r-00001`, `part-r-00002` et `part-r-00003`.

La commande `getmerge` va permettre de fusionner ces quatre parties et de copier le fichier obtenu sur le système de fichiers local :

```
hadoop fs -getmerge /data/weblogs/weblogs_md5_groups.bcp
➔ weblogs_md5_groups.bcp
```

L’affichage du contenu du dossier local est illustré à la Figure 1.8.

```
[cloudera@localhost data]$ hadoop fs -getmerge /data/weblogs/weblogs_md5_groups.bcp weblogs_md5_grou
ps.bcp
[cloudera@localhost data]$ ls -ltr
total 600
-rwxr-xr-x 1 cloudera cloudera 254129 Dec 31 11:15 weblog_entries.txt
-rwxr-xr-x 1 cloudera cloudera 354587 Dec 31 15:25 weblogs_md5_groups.bcp
[cloudera@localhost data]$
```

**Figure 1.8***Contenu du dossier local.*

## Voir aussi

- La section “Lire et écrire des données dans HDFS” du Chapitre 2 explique comment utiliser directement l’API du système de fichiers.
- La page suivante recense les différentes commandes du shell qui concernent le système de fichiers :

<http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html>

- Le lien suivant mène à la documentation Java de la classe `FileSystem` :

<http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/fs/FileSystem.html>



## Déplacer efficacement des données entre des clusters avec Distributed Copy

Distributed Copy (`distcp`) est un outil d'Hadoop qui permet de copier efficacement de grands volumes de données dans ou entre des clusters. L'opération de copie se fonde sur le framework MapReduce, ce qui permet de bénéficier du parallélisme, de la gestion des erreurs, de la reprise, de la journalisation et des comptes rendus. La commande `distcp` se révélera utile pour le déplacement de données entre des environnements de clusters de développement, de recherche et de production.

### Préparation

Les clusters source et destination doivent être capables de s'atteindre l'un et l'autre.

Dans le cluster source, l'exécution spéculative doit être désactivée pour les tâches map, en fixant `mapred.map.tasks.speculative.execution` à `false` dans le fichier de configuration `mapred-site.xml`. Cela permet d'éviter les comportements indéfinis lorsqu'une tâche map échoue.

Les clusters source et destination doivent employer le même protocole RPC. En général, cela signifie qu'ils doivent être équipés de la même version d'Hadoop.

### Réalisation

Déroulez les étapes suivantes de manière à copier un dossier depuis un cluster vers un autre :

1. Copiez le dossier `weblogs` depuis le cluster A vers le cluster B :

```
hadoop distcp hdfs://namenodeA/data/weblogs hdfs://namenodeB/data/weblogs
```

2. Copiez le dossier `weblogs` depuis le cluster A vers le cluster B, en écrasant les fichiers existants :

```
hadoop distcp -overwrite hdfs://namenodeA/data/weblogs  
➡ hdfs://namenodeB/data/weblogs
```

3. Lancez la synchronisation de `weblogs` entre le cluster A et le cluster B :

```
hadoop distcp -update hdfs://namenodeA/data/weblogs  
➡ hdfs://namenodeB/data/weblogs
```

### Explications

Sur le cluster source, le contenu du dossier à copier est traité comme un grand fichier temporaire. Un job MapReduce de type `map-only` est créé afin d'effectuer la copie entre les clusters. Par défaut, chaque mappeur recevra un bloc de 256 Mo du fichier temporaire. Par exemple, si le dossier `weblogs` a une taille de 10 Go,

quarante mappeurs recevront chacun environ 256 Mo à copier. Une option de `distcp` permet de préciser le nombre de mappeurs :

```
hadoop distcp -m 10 hdfs://namenodeA/data/weblogs  
➡ hdfs://namenodeB/data/weblogs
```

Dans cet exemple, dix mappeurs vont être utilisés. Si le dossier `weblogs` fait 10 Go, chacun aura alors 1 Go à copier.

## Pour aller plus loin

Pour effectuer une copie entre deux clusters qui opèrent avec des versions différentes d'Hadoop, il est généralement conseillé d'employer `HftpFileSystem` comme source. `HftpFileSystem` est un système de fichiers en lecture seule. La commande `distcp` doit être lancée à partir du serveur destination :

```
hadoop distcp hftp://namenodeA:port/data/weblogs  
➡ hdfs://namenodeB/data/weblogs
```

Dans cette commande, la valeur de `port` est donnée par la propriété `dfs.http.address` du fichier de configuration `hdfs-site.xml`.

## Importer des données dans HDFS à partir de MySQL avec Sqoop

Sqoop est un projet Apache qui fait partie de l'écosystème Hadoop global. Par de nombreux points, il est comparable à `distcp` (voir la section “Déplacer efficacement des données entre des clusters avec Distributed Copy”). Tous deux se fondent sur MapReduce et bénéficient de son parallélisme et de sa tolérance aux pannes. Au lieu de déplacer des données entre des clusters, Sqoop les déplace entre des bases de données relationnelles, en se connectant au travers d'un pilote JDBC.

Les fonctionnalités de Sqoop sont extensibles. Cette section montre comment l'utiliser pour importer des données dans HDFS à partir d'une base de données MySQL. Les entrées d'un journal web sont prises comme exemple.

## Préparation

Notre exemple met en œuvre Sqoop v1.3.0.

Si vous utilisez CDH3, Sqoop est déjà installé. Dans le cas contraire, vous trouverez sur la page <https://ccp.cloudera.com/display/CDHDOC/Sqoop+Installation> des instructions adaptées à votre distribution.

Nous supposons que vous disposez d'une instance de MySQL opérationnelle et qu'elle peut atteindre votre cluster Hadoop. La table `mysql.user` est configurée de façon qu'un utilisateur puisse se connecter à partir de la machine sur laquelle sera

lancé Sqoop. Pour de plus amples informations sur l'installation et la configuration de MySQL, consultez le site <http://dev.mysql.com/doc/refman/5.7/en/installing.html>.

Le fichier JAR du pilote pour MySQL doit se trouver dans `$$SQOOP_HOME/libs`. Vous pouvez le télécharger à partir de <http://dev.mysql.com/downloads/connector/j/>.

## Réalisation

Déroulez les étapes suivantes pour transférer des données à partir d'une table MySQL vers un fichier HDFS :

1. Créez une base de données dans l'instance de MySQL :

```
CREATE DATABASE logs;
```

2. Créez la table weblogs et chargez des données :

```
USE logs;
CREATE TABLE weblogs(
    md5          VARCHAR(32),
    url          VARCHAR(64),
    request_date DATE,
    request_time TIME,
    ip           VARCHAR(15)
);
LOAD DATA INFILE '/path/weblog_entries.txt' INTO TABLE weblogs
FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\r\n';
```

3. Demandez le nombre de lignes présentes dans la table weblogs :

```
mysql> select count(*) from weblogs;
```

La sortie doit ressembler à la suivante :

```
+-----+
| count(*) |
+-----+
|      3000 |
+-----+
1 row in set (0.01 sec)
```

4. Importez les données de la table MySQL dans HDFS :

```
sqoop import -m 1 --connect jdbc:mysql://<HÔTE>:<PORT>/logs
➤ --username hdp_usr --password test1 --table weblogs
➤ --target-dir /data/weblogs/import
```

Vous devez obtenir une sortie comparable à la suivante :

```
INFO orm.CompilationManager: Writing jar file:
/tmp/sqoop-jon/compile/f57ad8b208643698f3d01954eedb2e4d/weblogs.jar
WARN manager.MySQLManager: It looks like you are importing from mysql.
```

```

WARN manager.MySQLManager: This transfer can be faster! Use the --direct
WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
...
INFO mapred.JobClient:      Map input records=3000
INFO mapred.JobClient:      Spilled Records=0
INFO mapred.JobClient:      Total committed heap usage (bytes)=85000192
INFO mapred.JobClient:      Map output records=3000
INFO mapred.JobClient:      SPLIT_RAW_BYTES=87
INFO mapreduce.ImportJobBase: Transferred 245.2451 KB in 13.7619 seconds
(17.8206 KB/sec)
INFO mapreduce.ImportJobBase: Retrieved 3000 records.

```

## Explications

Sqoop charge le pilote JDBC indiqué par l'option `--connect`, à partir du dossier `$$SQOOP_HOME/libs`. La valeur de `$$SQOOP_HOME` correspond au chemin complet du dossier d'installation de Sqoop. Les options `--username` et `--password` permettent d'authentifier auprès de l'instance de MySQL l'utilisateur qui soumet la commande. La table `mysql.user` doit comprendre une entrée qui correspond à l'option `--username` et à l'hôte de chaque nœud du cluster Hadoop. Dans le cas contraire, Sqoop lancera une exception afin de signaler que la connexion de l'hôte au serveur MySQL n'est pas autorisée.

Pour le vérifier, exécutez les requêtes suivantes :

```

mysql> USE mysql;
mysql> select host, user from user;

```

Vous devez obtenir une sortie comparable à la suivante :

```

+-----+-----+
| user   | host   |
+-----+-----+
| hdp_usr | hdp01  |
| hdp_usr | hdp02  |
| hdp_usr | hdp03  |
| hdp_usr | hdp04  |
| root    | 127.0.0.1 |
| root    | ::1     |
| root    | localhost |
+-----+-----+
7 rows in set (1.04 sec)

```

Dans cet exemple, la connexion au serveur MySQL se fait avec le nom `hdp_usr`. Le cluster est constitué de quatre machines : `hdp01`, `hdp02`, `hdp03` et `hdp04`.

L'argument de `--table` précise à Sqoop la table concernée par l'importation. Dans notre cas, nous voulons que les données de la table `weblogs` soient stockées dans HDFS. L'argument de `--target-dir` correspond au chemin du dossier HDFS dans lequel ces données seront placées.

Après l'exécution de la commande, affichez le contenu de ce dossier :

```
hadoop fs -ls /data/weblogs/import
```

Voici le résultat attendu :

```
-rw-r--r--  1  hdp_usr hdp_grp    0      2012-06-08  23:47 /data/
weblogs/import/_SUCCESS
drwxr-xr-x  -  hdp_usr  hdp_grp    0      2012-06-08  23:47 /data/
weblogs/import/_logs
-rw-r--r--  1  hdp_usr hdp_grp 251131 2012-06-08  23:47 /data/
weblogs/import/part-m-00000
```

Par défaut, les données importées seront divisées en fonction de la clé primaire. Si aucune clé primaire n'est définie sur la table concernée, l'option `-m` ou `--split-by` doit être utilisée pour préciser à Sqoop comment copier des données. Dans notre exemple, nous avons choisi l'option `-m`. Elle contrôle le nombre de mappeurs utilisés pour l'importation. Puisque nous lui avons donné la valeur 1, un seul mappeur est mis en œuvre. Chaque mappeur utilisé produira un fichier de sortie.

Cette seule ligne de commande masque une étonnante complexité. Sqoop utilise les métadonnées enregistrées dans la base de données afin de générer des classes `DBWritable` pour chaque colonne. Elles sont utilisées par `DBInputFormat`, qui correspond à un format d'entrée Hadoop ayant la possibilité de lire les résultats de requêtes quelconques effectuées sur une base de données. Dans notre exemple, un job MapReduce est démarré en utilisant la classe `DBInputFormat` pour obtenir le contenu de la table `weblogs`. L'intégralité du contenu de cette table est analysée et placée dans le dossier `/data/weblogs/import`.

## Pour aller plus loin

De nombreuses options de configuration permettent d'ajuster la procédure d'importation des données par Sqoop. Cet outil est capable de placer des données dans des fichiers au format Avro ou SequenceFile en utilisant les options `--as-avrodatafile` et `--as-sequencefile`, respectivement. Les données peuvent également être compressées en ajoutant l'option `-z` ou `--compress`. Le codec utilisé par défaut est GZIP, mais n'importe quel schéma de compression Hadoop peut être utilisé grâce à l'option `--compression-codec <CODEC>` (consultez la section "Compresser des données avec LZ0" du Chapitre 2). L'option `--direct` pourra également se révéler utile. Elle demande à Sqoop d'utiliser les outils d'importation et d'exportation natifs s'ils sont pris en charge par la base de données configurée. Dans notre exemple, si nous avons ajouté `--direct`, Sqoop aurait utilisé `mysqldump` pour une exportation rapide de la table `weblogs`. Cette option est tellement importante que son absence a déclenché l'affichage d'un message d'avertissement dans notre exemple :

```
WARN manager.MySQLManager: It looks like you are importing from mysql.
WARN manager.MySQLManager: This transfer can be faster! Use the --direct
WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
```

## Voir aussi

- La section “Exporter des données à partir de HDFS vers MySQL avec Sqoop”.

## Exporter des données à partir de HDFS vers MySQL avec Sqoop

Sqoop est un projet Apache qui fait partie de l'écosystème Hadoop global. Par de nombreux points, il est comparable à `distcp` (voir la section “Déplacer efficacement des données entre des clusters avec Distributed Copy”). Tous deux se fondent sur MapReduce et bénéficient de son parallélisme et de sa tolérance aux pannes. Au lieu de déplacer des données entre des clusters, Sqoop les déplace entre des bases de données relationnelles, en se connectant au travers d'un pilote JDBC.

Les fonctionnalités de Sqoop sont extensibles. Cette section montre comment l'utiliser pour exporter des données à partir de HDFS vers une base de données MySQL. Les entrées d'un journal web sont prises comme exemple.

## Préparation

Notre exemple met en œuvre Sqoop v1.3.0.

Si vous utilisez CDH3, Sqoop est déjà installé. Dans le cas contraire, vous trouverez sur la page <https://ccp.cloudera.com/display/CDHDOC/Sqoop+Installation> des instructions adaptées à votre distribution.

Nous supposons que vous disposez d'une instance de MySQL opérationnelle et qu'elle peut atteindre votre cluster Hadoop. La table `mysql.user` est configurée de façon qu'un utilisateur puisse se connecter à partir de la machine sur laquelle sera lancé Sqoop. Pour de plus amples informations sur l'installation et la configuration de MySQL, consultez le site <http://dev.mysql.com/doc/refman/5.7/en/installing.html>.

Le fichier JAR du pilote pour MySQL doit se trouver dans `$SQOOP_HOME/libs`. Vous pouvez le télécharger à partir de <http://dev.mysql.com/downloads/connector/j/>.

Pour charger le fichier `weblog_entires.txt` dans HDFS, suivez les indications données à la section “Importer et exporter des données à l'aide de commandes du shell Hadoop”.

## Réalisation

Déroulez les étapes suivantes pour transférer des données depuis HDFS vers une table MySQL :

1. Créez une base de données dans l'instance de MySQL :

```
CREATE DATABASE logs;
```

## 2. Créez la table `weblogs_from_hdfs` :

```
USE logs;
CREATE TABLE weblogs_from_hdfs (
  md5          VARCHAR(32),
  url          VARCHAR(64),
  request_date DATE,
  request_time TIME,
  ip           VARCHAR(15)
);
```

## 3. Exportez le fichier `weblog_entries.txt` depuis HDFS vers MySQL :

```
sqoop export -m 1 --connect jdbc:mysql://<HÔTE>:<PORT>/logs
➤ --username hdp_usr --password test1 --table weblogs_from_hdfs
➤ --export-dir /data/weblogs/05102012 --input-fields-terminated-by '\t'
➤ --mysql-delimiters
```

Vous devez obtenir une sortie comparable à la suivante :

```
INFO mapreduce.ExportJobBase: Beginning export of weblogs_from_hdfs
input.FileInputFormat: Total input paths to process : 1
input.FileInputFormat: Total input paths to process : 1
mapred.JobClient: Running job: job_201206222224_9010
INFO mapred.JobClient: Map-Reduce Framework
INFO mapred.JobClient: Map input records=3000
INFO mapred.JobClient: Spilled Records=0
INFO mapred.JobClient: Total committed heap usage (bytes)=85000192
INFO mapred.JobClient: Map output records=3000
INFO mapred.JobClient: SPLIT_RAW_BYTES=133
INFO mapreduce.ExportJobBase: Transferred 248.3086 KB in 12.2398 seconds
(20.287 KB/sec)
INFO mapreduce.ExportJobBase: Exported 3000 records.
```

## Explications

Sqoop charge le pilote JDBC indiqué par l'option `--connect`, à partir du dossier `$SQOOP_HOME/libs`. La valeur de `$SQOOP_HOME` correspond au chemin complet du dossier d'installation de Sqoop. Les options `--username` et `--password` permettent d'authentifier auprès de l'instance de MySQL l'utilisateur qui soumet la commande. La table `mysql.user` doit comprendre une entrée qui correspond à l'option `--username` et à l'hôte de chaque nœud du cluster Hadoop. Dans le cas contraire, Sqoop lancera une exception afin de signaler que la connexion de l'hôte au serveur MySQL n'est pas autorisée.

```
mysql> USE mysql;
mysql> select host, user from user;
```

```

+-----+-----+
| user      | host      |
+-----+-----+
| hdp_usr    | hdp01     |
| hdp_usr    | hdp02     |
| hdp_usr    | hdp03     |
| hdp_usr    | hdp04     |
| root       | 127.0.0.1 |
| root       | ::1       |
| root       | localhost  |
+-----+-----+
7 rows in set (1.04 sec)

```

Dans cet exemple, la connexion au serveur MySQL se fait avec le nom `hdp_usr`. Le cluster est constitué de quatre machines : `hdp01`, `hdp02`, `hdp03` et `hdp04`.

L'argument de `--table` précise à Sqoop la table qui va recevoir les données provenant de HDFS. Elle doit avoir été créée avant que la commande `export` de Sqoop soit exécutée. Sqoop utilise les métadonnées de la table, le nombre de colonnes et leur type pour valider les données qui proviennent du dossier HDFS et pour créer des instructions `INSERT`. Le job d'exportation peut être vu comme la lecture de chaque ligne du fichier `weblogs_entries.txt` stocké dans HDFS et la génération de la sortie suivante :

```

INSERT INTO weblogs_from_hdfs
VALUES('aabb15edcd0c8042a14bf216c5', '/jcwbtvnkkujo.html',
'2012-05-10', '21:25:44', '148.113.13.214');

INSERT INTO weblogs_from_hdfs
VALUES('e7d3f242f11c1b522137481d8508ab7', '/ckyhatbpxu.html',
'2012-05-10', '21:11:20', '4.175.198.160');

INSERT INTO weblogs_from_hdfs
VALUES('b8bd62a5c4ede37b9e77893e043fc1', '/rr.html',
'2012-05-10', '21:32:08', '24.146.153.181');
...

```

Par défaut, la commande `export` de Sqoop crée des instructions `INSERT`. Lorsque l'option `--update-key` est indiquée, des instructions `UPDATE` sont créées à la place. Si nous avons utilisé `--update-key md5` dans l'exemple précédent, le code généré aurait été différent :

```

UPDATE weblogs_from_hdfs SET url='/jcwbtvnkkujo.html',
request_date='2012-05-10' request_time='21:25:44' ip='148.113.13.214'
WHERE md5='aabb15edcd0c8042a14bf216c5'

UPDATE weblogs_from_hdfs SET url='/jcwbtvnkkujo.html',
request_date='2012-05-10' request_time='21:11:20' ip='4.175.198.160'
WHERE md5='e7d3f242f11c1b522137481d8508ab7'

UPDATE weblogs_from_hdfs SET url='/jcwbtvnkkujo.html',
request_date='2012-05-10' request_time='21:32:08' ip='24.146.153.181'
WHERE md5='b8bd62a5c4ede37b9e77893e043fc1'

```



Si la valeur de `--update-key` n'est pas trouvée, donner à `--update-mode` la valeur `allowinsert` déclenche l'insertion de la ligne.

L'option `-m` définit le nombre de jobs map qui liront les portions du fichier à partir de HDFS. Chaque mappeur aura sa propre connexion au serveur MySQL. Il insérera jusqu'à cent enregistrements par instruction. Après qu'il a terminé cent instructions `INSERT`, c'est-à-dire l'insertion de dix mille enregistrements, il valide la transaction en cours. Il est possible qu'une erreur dans la tâche map provoque une incohérence dans les données, conduisant à des collisions d'insertion ou à des doublons. Pour éviter ces problèmes, l'option `--staging-table` peut être ajoutée. Dans ce cas, le job effectue les insertions dans une table intermédiaire, puis, en une seule transaction, déplace les données de cette table vers celle indiquée dans l'option `--table`. Le format de la table indiquée par `--staging-table` doit être identique à celui de la table précisée par `--table`. La table intermédiaire doit être vide, ou l'option `--clear-staging-table` doit être ajoutée.

## Voir aussi

- La section "Importer des données dans HDFS à partir de MySQL avec Sqoop".

## Configurer Sqoop pour Microsoft SQL Server

Dans cette section, nous expliquons comment configurer Sqoop en vue de sa connexion avec des bases de données Microsoft SQL Server. Cela nous permettra de charger efficacement des données dans HDFS à partir d'une base SQL Server.

### Préparation

Notre exemple met en œuvre Sqoop v1.3.0.

Si vous utilisez CDH3, Sqoop est déjà installé. Dans le cas contraire, vous trouverez sur la page <https://ccp.cloudera.com/display/CDHDOC/Sqoop+Installation> des instructions adaptées à votre distribution.

Nous supposons que vous disposez d'une instance de SQL Server opérationnelle et qu'elle peut être connectée à votre cluster Hadoop.

### Réalisation

Déroulez les étapes suivantes pour configurer Sqoop de manière qu'il puisse se connecter à Microsoft SQL Server :

1. Téléchargez le pilote JDBC 3.0 pour Microsoft SQL Server à partir de l'adresse [http://download.microsoft.com/download/D/6/A/D6A241AC-433E-4CD2-A1CE-50177E8428F0/1033/sqljdbc\\_3.0.1301.101\\_enu.tar.gz](http://download.microsoft.com/download/D/6/A/D6A241AC-433E-4CD2-A1CE-50177E8428F0/1033/sqljdbc_3.0.1301.101_enu.tar.gz).

L'archive obtenue comprend le pilote JDBC pour SQL Server (sqljdbc4.jar) nécessaire à Sqoop car celui-ci se connecte aux bases de données relationnelles au travers de pilotes JDBC.

2. Décompressez le fichier TAR et extrayez son contenu :

```
gzip -d sqljdbc_3.0.1301.101_enu.tar.gz
tar -xvf sqljdbc_3.0.1301.101_enu.tar
```

Vous devez obtenir un nouveau dossier nommé sqljdbc\_3.0.

3. Copiez sqljdbc4.jar dans \$SQOOP\_HOME/lib :

```
cp sqljdbc_3.0/enu/sqljdbc4.jar $SQOOP_HOME/lib
```

Sqoop peut alors accéder au fichier sqljdbc4.jar et l'utiliser pour se connecter à une instance de SQL Server.

4. Téléchargez le connecteur Microsoft SQL Server pour Hadoop à partir de l'URL <http://download.microsoft.com/download/B/E/5/BE5EC4FD-9EDA-4C3F-8B36-1C8AC4CE2CEF/sqoop-sqlserver-1.0.tar.gz>.

5. Décompressez le fichier TAR et extrayez son contenu :

```
gzip -d sqoop-sqlserver-1.0.tar.gz
tar -xvf sqoop-sqlserver-1.0.tar
```

Vous devez obtenir un nouveau dossier nommé sqoop-sqlserver-1.0.

6. Fixez la valeur de la variable d'environnement MSSQL\_CONNECTOR\_HOME :

```
export MSSQL_CONNECTOR_HOME=/chemin/vers/sqoop-sqlserver-1.0
```

7. Lancez le script d'installation :

```
./install.sh
```

8. Pour l'importation et l'exportation de données, consultez les sections "Importer des données dans HDFS à partir de MySQL avec Sqoop" et "Exporter des données à partir de HDFS vers MySQL avec Sqoop". Les solutions données fonctionnent également avec SQL Server. L'argument de --connect doit être remplacé par jdbc:sqlserver://<HÔTE>:<PORT>.

## Explications

Sqoop communique avec les bases de données en utilisant JDBC. Après que le fichier sqljdbc4.jar a été placé dans le dossier \$SQOOP\_HOME/lib, Sqoop est en mesure de se connecter à des instances de SQL Server si l'option --connect jdbc:sqlserver://<HÔTE>:<PORT> est ajoutée. Pour que SQL Server soit totalement compatible avec Sqoop, des modifications doivent être apportées à la configuration. Elles sont réalisées par le script install.sh.