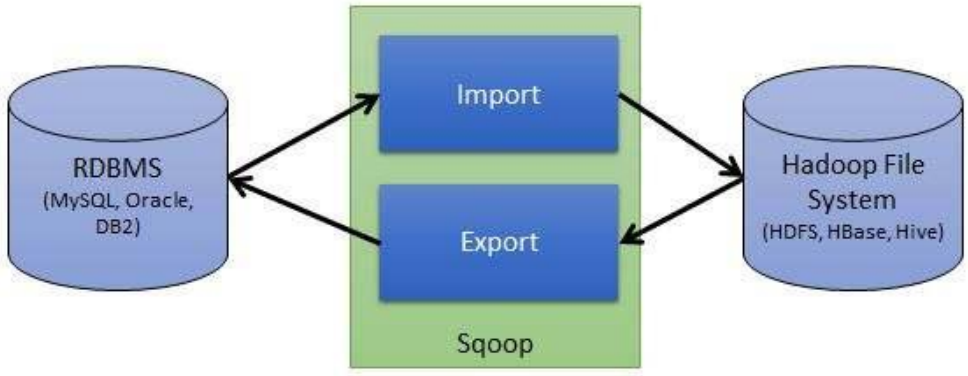
Sqoop

刘磊

2020 年 11 月

目录

Sqoop 是一个用于在 Hadoop 和关系数据库之间传输数据的工具。它用于从关系数据库（如 MySQL，Oracle）导入数据到 Hadoop HDFS，并从 HDFS 导出到关系数据库。Sqoop 工作的机制是将导入或导出命令翻译成 MapReduce 程序来实现在翻译出的 MapReduce 中主要是对 InputFormat 和 OutputFormat 进行定制。



Sqoop 的优点:

1) 可以高效、可控的利用资源，可以通过调整任务数来控制任务的并发度。

2) 可以自动的完成数据映射和转换。由于导入数据库是有类型的，它可以自动根据数据库中的类型转换到 Hadoop 中，当然用户也可以自定义它们之间的映射关系

3) .支持多种数据库，如 mysql，orcale 等数据库。

安装 Sqoop

1. 将 Sqoop 的安装包复制到/apps 目录下，并解压。

```
cp ~/big_data_tools/sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz /apps/
cd /apps
tar zxvf sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz
```

将解压的 sqoop-1.4.7.bin__hadoop-2.6.0 重命名为 sqoop。

```
mv /apps/sqoop-1.4.7.bin__hadoop-2.6.0/ /apps/sqoop
```

删除压缩包

```
rm sqoop-1.4.7.bin__hadoop-2.6.0.tar.gz
```

2. 修改环境变量，使用 vim 打开~/.bashrc 文件。

```
vim ~/.bashrc
```

添加以下内容到~/.bashrc 文件中。

```
# Sqoop
export SQOOP_HOME=/apps/sqoop
```

```
export PATH=$SQOOP_HOME/bin:$PATH
```

目前已经安装了 Java，Hadoop，Hbase, Hive, Sqoop，添加的内容如下图所示。

```
# Java
export JAVA_HOME=/apps/java
export PATH=$JAVA_HOME/bin:$PATH

# Hadoop
export HADOOP_HOME=/apps/hadoop
export PATH=$HADOOP_HOME/bin:$PATH

# Hbase
export HBASE_HOME=/apps/hbase
export PATH=$HBASE_HOME/bin:$PATH

# Hive
export HIVE_HOME=/apps/hive
export PATH=$HIVE_HOME/bin:$PATH

# Sqoop
export SQOOP_HOME=/apps/sqoop
export PATH=$SQOOP_HOME/bin:$PATH
```

使用 source 命令，使用户环境变量生效。

```
source ~/.bashrc
```

由于在导数据过程中，可能会涉及到连接 MySQL，所以需要把 MySQL 的 jdbc 连接包

mysql-connector-java-5.1.46-bin.jar，拷贝到 Sqoop 的 lib 目录下。

```
cp ~/big_data_tools/mysql-connector-java-5.1.46-bin.jar
/apps/sqoop/lib/
```

3. 配置 Sqoop。

切换到/apps/sqoop/conf 目录下，将 sqoop-env-template.sh 重命名为 sqoop-env.sh。

```
cd /apps/sqoop/conf/
```

```
mv sqoop-env-template.sh sqoop-env.sh
```

将 sqoop-env.sh 中的配置，修改为如下形式。

去掉 23 行的注释，修改为

```
export HADOOP_COMMON_HOME=/apps/hadoop
```

去掉 26 行的注释，修改为

```
export HADOOP_MAPRED_HOME=/apps/hadoop
```

去掉 29 行的注释，修改为

```
export HBASE_HOME=/apps/hbase
```

去掉 32 行的注释，修改为

```
export HIVE_HOME=/apps/hive
```

这里的配置项是告诉 Sqoop 框架，Hadoop、HBase、Hive 等的相关路径。

4. 下面再切换到/apps/sqoop/bin 目录下，修改 configure-sqoop 里面的部分脚本

```
cd /apps/sqoop/bin/
```

在 configure-sqoop 文件中，查找下面内容，128 行到 147 行，在前面加上“#”号，将脚本注释掉。用 vim 打开文件，在命令行模式下（在一般模式下输入冒号:进入命令行模式）执行如下命令，将 128 行到 147 行注释掉。

```
128,147s/^/#/g
```

```
128 ### Moved to be a runtime check in sqoop.
129 #if [ ! -d "${HBASE_HOME}" ]; then
130 #  echo "Warning: $HBASE_HOME does not exist! HBase imports will fail."
131 #  echo 'Please set $HBASE_HOME to the root of your HBase installation.'
132 #fi
133 #
134 ### Moved to be a runtime check in sqoop.
135 #if [ ! -d "${HCAT_HOME}" ]; then
136 #  echo "Warning: $HCAT_HOME does not exist! HCatalog jobs will fail."
137 #  echo 'Please set $HCAT_HOME to the root of your HCatalog installation.'
138 #fi
139 #
140 #if [ ! -d "${ACCUMULO_HOME}" ]; then
141 #  echo "Warning: $ACCUMULO_HOME does not exist! Accumulo imports will fail."
142 #  echo 'Please set $ACCUMULO_HOME to the root of your Accumulo installation.'
143 #fi
144 #if [ ! -d "${ZOOKEEPER_HOME}" ]; then
145 #  echo "Warning: $ZOOKEEPER_HOME does not exist! Accumulo imports will fail."
146 #  echo 'Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.'
147 #fi
```

5. 测试一下，当我们输入 sqoop version 时，显示版本信息如下

```
sqoop version
```

```
lei@ubuntu:~$ sqoop version
/apps/hadoop/libexec/hadoop-functions.sh: line 2326: HADOOP_ORG.APACHE.SQOOP.SQO
OP_USER: bad substitution
/apps/hadoop/libexec/hadoop-functions.sh: line 2421: HADOOP_ORG.APACHE.SQOOP.SQO
OP_OPTS: bad substitution
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/apps/hadoop/share/hadoop/common/lib/slf4j-log
4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apps/hbase/lib/slf4j-log4j12-1.7.10.jar!/org/
slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2020-09-01 22:44:53,015 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
Sqoop 1.4.7
git commit id 2328971411f57f0cb683dfb79d19d4d19d185dd8
Compiled by maugli on Thu Dec 21 15:59:58 STD 2017
```

使用 sqoop help 查看支持的命令

```
sqoop help
```

```
Available commands:
  codegen            Generate code to interact with database records
  create-hive-table  Import a table definition into Hive
  eval               Evaluate a SQL statement and display the results
  export             Export an HDFS directory to a database table
  help               List available commands
  import             Import a table from a database to HDFS
  import-all-tables  Import tables from a database to HDFS
  import-mainframe   Import datasets from a mainframe server to HDFS
  job                Work with saved jobs
  list-databases     List available databases on a server
  list-tables        List available tables in a database
  merge              Merge results of incremental imports
  metastore          Run a standalone Sqoop metastore
  version            Display version information
```

使用 sqoop help COMMAND 显示具体命令的信息，例如

```
sqoop help import
```

6. 下面我们测试一下 Sqoop 能否连接 MySQL。

首先，来查看一下 MySQL 服务是否已经启动。

```
sudo service mysql status
```

如果状态为 stopped 则需要执行启动命令。

```
sudo service mysql start
```

然后我们查询 MySQL 中都有哪些数据库，测试 Sqoop 能否连接 MySQL。

```
sqoop list-databases --connect jdbc:mysql://localhost:3306/ --
username root --password 123456
```

```
lei@ubuntu:~$ sqoop list-databases --connect jdbc:mysql://localhost:3306/ --user
name root --password 123456
/apps/hadoop/libexec/hadoop-functions.sh: line 2326: HADOOP_ORG.APACHE.SQOOP.SQO
OP_USER: bad substitution
/apps/hadoop/libexec/hadoop-functions.sh: line 2421: HADOOP_ORG.APACHE.SQOOP.SQO
OP_OPTS: bad substitution
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/apps/hadoop/share/hadoop/common/lib/slf4j-log
4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apps/hbase/lib/slf4j-log4j12-1.7.10.jar!/org/
slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2020-09-01 22:48:34,699 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
2020-09-01 22:48:34,903 WARN tool.BaseSqoopTool: Setting your password on the co
mmand-line is insecure. Consider using -P instead.
2020-09-01 22:48:35,167 INFO manager.MySQLManager: Preparing to use a MySQL stre
aming resultset.
Tue Sep 01 22:48:35 CST 2020 WARN: Establishing SSL connection without server's
identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ an
d 5.7.6+ requirements SSL connection must be established by default if explicit
option isn't set. For compliance with existing applications not using SSL the ve
rifyServerCertificate property is set to 'false'. You need either to explicitly
disable SSL by setting useSSL=false, or set useSSL=true and provide truststore f
or server certificate verification.
information_schema
hive
mysql
performance_schema
sys
```

通过上述命令列出了 MySQL 中的数据库。说明 Sqoop 安装成功！

使用 Sqoop

在并行操作里，首先要解决输入数据是以什么方式负债均衡到多个 map 的，即怎么保证每个 map 处理的数据量大致相同且数据不重复。--split-by 指定了 split column，在执行并行操作时(多个 map task)，其思想是：

1、先查出 split column 的最小值和最大值

2、然后根据 map task 数对(max-min)之间的数据进行均匀的范围切分

例如 id 作为 split column,其最小值是 0、最大值 1000，如果设置 4 个 map 数，每个

map task 执行的查询语句类似于：

SELECT * FROM sometable WHERE id >= lo AND id < hi;

每个 task 里(lo,hi)的值分别是 (0, 250), (250, 500), (500, 750), and (750, 1001)。

Sqoop 不能在多列字段上进行拆分，如果没有索引或者有组合键，必须显示设置 splitting column；默认的主键作为 split column，如果表里没有主键或者没有指定--split-by，就要设置 num-mappers 1 或者--autoreset-to-one-mapper，这样就只会启动一个 task。

进入 MySQL 创建并使用数据库 mydb。

```
create database mydb;
use mydb;
```

在 mydb 数据库中创建有五个字段的表 record

```
create table record
(
  id varchar(100),
  buyer_id varchar(100),
  dt varchar(100),
  ip varchar(100),
  opt_type varchar(100)
);
```

将本地文件/data/buyer_log 里的内容，导入的 mydb 数据库 record 表中

```
load data local infile '/data/buyer_log' into table record fields
terminated by '\t';
```

使用 Sqoop 查看 MySQL 中的数据库

```
sqoop list-databases \
--connect jdbc:mysql://localhost:3306/ \
--username root \
--password 123456
```

```
information_schema
hive
mydb
mysql
performance_schema
sys
```

用 Sqoop 查看 MySQL 中的表

```
sqoop list-tables \
--connect jdbc:mysql://localhost:3306/mydb \
--username root \
--password 123456
```

将 MySQL 中的数据导入到 HDFS

将表格 record 中的数据导入到 HDFS /mysqoop 目录下

```
sqoop import \
--connect jdbc:mysql://localhost:3306/mydb \
--username root \
--password 123456 \
--table record -m 1 \
--target-dir /mysqoop
```

查看 HDFS 上/mysqoop 目录下的文件内容

```
hadoop fs -ls /mysqoop
```

```
lei@ubuntu:~$ hadoop fs -ls /mysqoop
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further detail
s.
Found 2 items
-rw-r--r--   1 lei supergroup          0 2020-09-03 16:33 /mysqoop/_SUCCESS
-rw-r--r--   1 lei supergroup       2948 2020-09-03 16:33 /mysqoop/part-m-00000
```

将 HDFS 中数据存入到 MySQL 数据库中

在 MySQL 中创建与 record 结构相同的表 recordfromhdfs

```
use mydb;
create table recordfromhdfs like record;
```

```
mysql> create table recordfromhdfs like record;
Query OK, 0 rows affected (0.02 sec)
```

将上面从 MySQL 导出到 HDFS 的数据存入新创建的表格 recordfromhdfs 中

```
sqoop export \
--connect jdbc:mysql://localhost:3306/mydb?characterEncoding=UTF-8 \
--username root \
--password 123456 \
--table recordfromhdfs -m 1 \
--export-dir hdfs://localhost:9000/mysqoop/part-m-00000
```

验证结果

```
select * from recordfromhdfs limit 5;
```

```
mysql> select * from recordfromhdfs limit 5;
+------+----------+---------------------+-----------------+----------+
| id   | buyer_id | dt                  | ip              | opt_type |
+------+----------+---------------------+-----------------+----------+
| 462  | 10262    | 2010-03-26 19:55:10 | 123.127.164.252 | 1        |
| 463  | 20001    | 2010-03-29 14:28:02 | 221.208.129.117 | 2        |
| 464  | 20001    | 2010-03-29 14:28:02 | 221.208.129.117 | 1        |
| 465  | 20002    | 2010-03-30 10:56:35 | 222.44.94.235   | 2        |
| 466  | 20002    | 2010-03-30 10:56:35 | 222.44.94.235   | 1        |
+------+----------+---------------------+-----------------+----------+
5 rows in set (0.00 sec)
```

将 MySQL 中数据导入到 HBase 中

启动 HBase

```
cd /apps/hbase/bin
./start-hbase.sh
```

在终端中输入 hbase shell，进入 HBase 命令行模式。

```
hbase shell
```

创建名为 hbaserecord，有一个列族 mycf 的表，用来保存数据

```
create 'hbaserecord', 'mycf'
```

打开一个新终端，开始导数据

```
sqoop import \
--connect jdbc:mysql://localhost:3306/mydb?characterEncoding=UTF-8 \
--username root \
--password 123456 \
--table record \
--hbase-table hbaserecord \
--column-family mycf \
--hbase-row-key dt -m 1
```

注意上述命令中，选项--hbase-row-key 的作用是选择数据在 hbase 表中的 rowkey。这里以 MySQL 表中的 dt 列的值作为 HBase 表中的 rowkey。

查看 HBase 表 hbaserecord 里的内容。

```
scan 'hbaserecord'
```

```
hbase(main):002:0> scan 'hbaserecord'
ROW                   COLUMN+CELL
 2010-03-26 19:55:10    column=mycf:buyer_id, timestamp=1599126754501, value=10262
 2010-03-26 19:55:10    column=mycf:id, timestamp=1599126754501, value=462
 2010-03-26 19:55:10    column=mycf:ip, timestamp=1599126754501, value=123.127.164.252
 2010-03-26 19:55:10    column=mycf:opt_type, timestamp=1599126754501, value=1
 2010-03-29 14:28:02    column=mycf:buyer_id, timestamp=1599126754501, value=20001
 2010-03-29 14:28:02    column=mycf:id, timestamp=1599126754501, value=464
 2010-03-29 14:28:02    column=mycf:ip, timestamp=1599126754501, value=221.208.129.117
 2010-03-29 14:28:02    column=mycf:opt_type, timestamp=1599126754501, value=1
 2010-03-30 10:56:35    column=mycf:buyer_id, timestamp=1599126754501, value=20002
 2010-03-30 10:56:35    column=mycf:id, timestamp=1599126754501, value=466
 2010-03-30 10:56:35    column=mycf:ip, timestamp=1599126754501, value=222.44.94.235
 2010-03-30 10:56:35    column=mycf:opt_type, timestamp=1599126754501, value=1
 2010-03-31 16:48:43    column=mycf:buyer_id, timestamp=1599126754501, value=10181
 2010-03-31 16:48:43    column=mycf:id, timestamp=1599126754501, value=481
 2010-03-31 16:48:43    column=mycf:ip, timestamp=1599126754501, value=123.127.164.252
```

将 MySQL 的数据导入到 Hive 中

将 MySQL 数据导入 Hive，需要将以下内容添加到~/.bashrc

```
export HADOOP_CLASSPATH=/apps/hadoop/lib
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:/apps/hive/lib/*
```

```
# Hadoop
export HADOOP_HOME=/apps/hadoop
export PATH=$HADOOP_HOME/bin:$PATH
export HADOOP_CLASSPATH=/apps/hadoop/lib
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:/apps/hive/lib/*
```

还有以下内容添加到~/.bashrc

```
export HIVE_CONF_DIR=$HIVE_HOME/conf
```

```
# Hive
export HIVE_HOME=/apps/hive
export PATH=$HIVE_HOME/bin:$PATH
export HIVE_CONF_DIR=$HIVE_HOME/conf
```

使配置文件生效

```
source ~/.bashrc
```

开启 Hive，直接在终端中执行

```
hive
```

在 Hive 中创建 hiverecord 表，包含（id,buyer_id,dt,ip,opt_type）五个字段，字符类型

均为 varchar(100)，分隔符为 ‘,’

```
create table hiverecord (id varchar(100),buyer_id varchar(100),dt
varchar(100), ip varchar(100), opt_type varchar(100))  row format
delimited fields terminated by ','  stored as textfile;
```

使用 Sqoop 将 MySQL 中 record 表导入 Hive 中

```
sqoop import \
--connect jdbc:mysql://localhost:3306/mydb?characterEncoding=UTF-8 \
--username root \
--password 123456 \
--table record \
--hive-import \
--hive-table hiverecord \
--fields-terminated-by ',' -m 1
```

执行成功以后，查看 hiverecord 表内容

```
select * from hiverecord;
```

```
hive> select * from hiverecord;
OK
462     10262   2010-03-26 19:55:10     123.127.164.252 1
463     20001   2010-03-29 14:28:02     221.208.129.117 2
464     20001   2010-03-29 14:28:02     221.208.129.117 1
465     20002   2010-03-30 10:56:35     222.44.94.235   2
466     20002   2010-03-30 10:56:35     222.44.94.235   1
481     10181   2010-03-31 16:48:43     123.127.164.252 1
482     10181   2010-04-01 17:35:05     123.127.164.252 1
483     10181   2010-04-02 10:34:20     123.127.164.252 1
484     20001   2010-04-04 16:38:22     221.208.129.38  1
485     10181   2010-04-04 16:54:22     222.35.127.83   1
486     20021   2010-04-06 09:22:21     222.44.94.123   2
```

将 Hive 表的数据导出到 MySQL 中

首先在 MySQL 中创建表 recordfromhive。

```
use mydb;
```

```
create table recordfromhive like record;
```

使用 Sqoop 开始导数据。

```
sqoop export \
--connect jdbc:mysql://localhost:3306/mydb?characterEncoding=UTF-8 \
--username root \
--password 123456 \
--table recordfromhive \
--export-dir /user/hive/warehouse/hiverecord/part-m-00000 \
--input-fields-terminated-by ',' -m 1
```

查看结果

```
select * from recordfromhive;
```

```
mysql> select * from recordfromhive;
+------+----------+---------------------+-----------------+----------+
| id   | buyer_id | dt                  | ip              | opt_type |
+------+----------+---------------------+-----------------+----------+
| 462  | 10262    | 2010-03-26 19:55:10 | 123.127.164.252 | 1        |
| 463  | 20001    | 2010-03-29 14:28:02 | 221.208.129.117 | 2        |
| 464  | 20001    | 2010-03-29 14:28:02 | 221.208.129.117 | 1        |
| 465  | 20002    | 2010-03-30 10:56:35 | 222.44.94.235   | 2        |
| 466  | 20002    | 2010-03-30 10:56:35 | 222.44.94.235   | 1        |
| 481  | 10181    | 2010-03-31 16:48:43 | 123.127.164.252 | 1        |
| 482  | 10181    | 2010-04-01 17:35:05 | 123.127.164.252 | 1        |
| 483  | 10181    | 2010-04-02 10:34:20 | 123.127.164.252 | 1        |
| 484  | 20001    | 2010-04-04 16:38:22 | 221.208.129.38  | 1        |
| 485  | 10181    | 2010-04-04 16:54:22 | 222.35.127.83   | 1        |
| 486  | 20021    | 2010-04-06 09:22:21 | 222.44.94.123   | 2        |
```