Sqoop课程笔记

一、概述

1、什么是sqoop？

Hadoop的优势在于对数据的存储和处理，相比以前传统的数据库，在处理较较多的数据时，传统数据行业通过提升单机性能以提高处理性能，而且性价比随着性能提高越来越低，在场景下派生出的大数据行业。

同样的数据处理，hadoop无论是处理的性能和成本都远低于传统通过单机处理，但是从传统的数据处理切换到新生的hadoop平台，避免不了有数据迁移的过程，需要将传统数据按照hadoop的规则进行转换，中间需要一个转换的工具，由此派生出sqoop这样一个优秀的工具，不仅可以将关系数据库（oracle、mysql、postgresql等）数据转换到hadoop中进行处理，同样可以将hadoop数据处理的结果导入到关系数据库中。

2、sqoop的应用场景

不用说，既然是一个转换工具，在关系数据库和hadoop之间数据的相互转换就少不了。但是不是所有的数据都适合于hadoop处理，比如安全性高的银行业以及结构化很强的数据就不是很适合使用hadoop进行处理、存储。

3、sqoop的版本问题

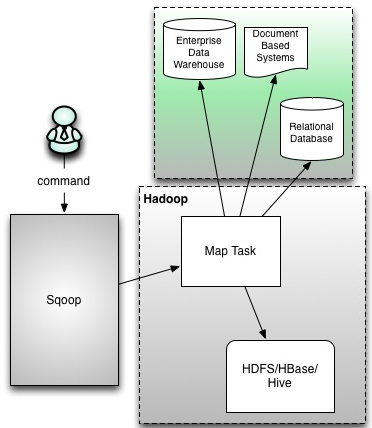
sqoop有两个版本，完全不兼容，可以从版本号进行区分，1.4.x为sqoop1，1.99.x为sqoop2。

4、sqoop的地位

唯我独尊！

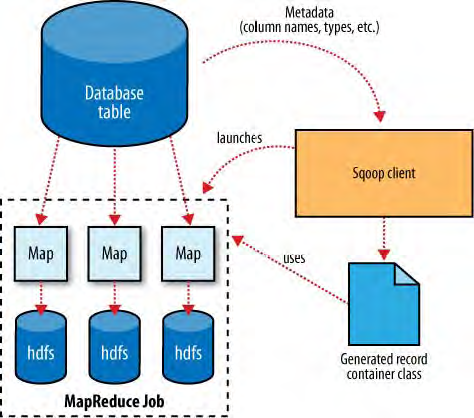
二、架构

1、sqoop整体架构

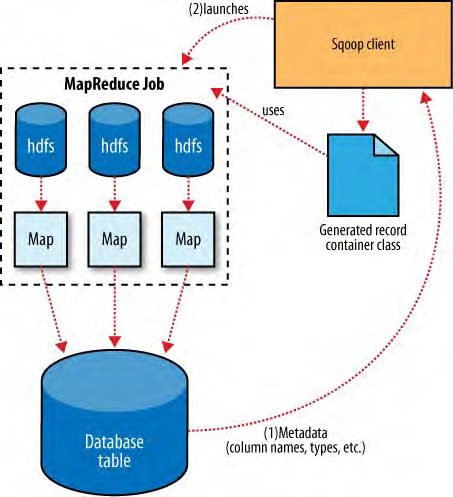


sqoop架构非常简单，是hadoop生态系统的架构最简单的框架，sqoop1由client端直接接入hadoop，任务通过解析生成对应的maprecue执行。

2、导入



3、导出



3、执行流程

1）读取要导入数据的表结构，生成运行类，默认是QueryResult，打成jar包，然后提交给Hadoop

2）设置好job，主要也就是设置好的各个参数

3）这里就由Hadoop来执行MapReduce来执行Import命令了，

      a.首先要对数据进行切分，也就是DataSplit

          DataDrivenDBInputFormat.getSplits(JobContext job)

      b.切分好范围后，写入范围，以便读取

       DataDrivenDBInputFormat.write(DataOutput output) 这里是lowerBoundQuery and  upperBoundQuery

      c.读取以上2）写入的范围

          DataDrivenDBInputFormat.readFields(DataInput input)

d.然后创建RecordReader从数据库中读取数据

DataDrivenDBInputFormat.createRecordReader(InputSplit split,TaskAttemptContext context)

e.创建Map

TextImportMapper.setup(Context context)

f.RecordReader一行一行从关系型数据库中读取数据，设置好Map的Key和Value，交给Map

DBRecordReader.nextKeyValue()

g.运行map

TextImportMapper.map(LongWritable key, SqoopRecord val, Context context)

最后生成的Key是行数据，由QueryResult生成，Value是NullWritable.get()

三、安装

1、下载地址

sqoop1:

<http://archive.cloudera.com/cdh5/cdh/5/sqoop-1.4.5-cdh5.4.4.tar.gz>

sqoop2:

<http://archive.cloudera.com/cdh5/cdh/5/sqoop2-1.99.5-cdh5.4.4.tar.gz>

2、基于CDH的安装

too simple！

CDH在安装了sqoop2之后就1和2 都有了。

3、原生安装

a.解压对应压缩包

b.重命名sqoop-1.4.5-cdh5.4.4/sqoop-env-template.sh 为sqoop-env.sh

配置以下参数即可：



c.配置环境变量  
 vi /etc/profile

export PATH=$PATH: /opt/sqoop-1.4.5-cdh5.4.4/bin

source /etc/profile

1. 注意事项
2. 数据库驱动：在执行sqoop命令的client的sqoop lib目录下面放入驱动jar包即可（CDH默认在/opt/cloudera/parcels/CDH/lib/sqoop/lib/目录），驱动版本使用mysql-connector-java-5.1.32-bin.jar以上。
3. client端JDK版本1.6~1.7，并且配置环境变量

四、使用

1、****工具****选项

Sqoop的包含很多工具，使用之前需要先指定具体的工具，在使用该工具的命令选项：

|  |  |
| --- | --- |
| **选项** | **含义说明** |
| **--connect <jdbc-uri>** | 指定JDBC连接字符串 |
| **--connection-manager <class-name>** | 指定要使用的连接管理器类 |
| **--driver <class-name>** | 指定要使用的JDBC驱动类 |
| **--hadoop-mapred-home <dir>** | 指定$HADOOP\_MAPRED\_HOME路径 |
| **--help** | 万能帮助 |
| **--password-file** | 设置用于存放认证的密码信息文件的路径 |
| **-P** | 从控制台读取输入的密码 |
| **--password <password>** | 设置认证密码 |
| **--username <username>** | 设置认证用户名 |
| **--verbose** | 打印详细的运行信息 |
| **--connection-param-file <filename>** | 可选，指定存储数据库连接参数的属性文件 |

2、数据导入工具import

所谓import工具，是将关系数据库数据导入到Hadoop平台，基本命令选项如下：

|  |  |
| --- | --- |
| **选项** | **含义说明** |
| **--append** | 将数据追加到HDFS上一个已存在的数据集上 |
| **--as-avrodatafile** | 将数据导入到Avro数据文件 |
| **--as-sequencefile** | 将数据导入到SequenceFile |
| **--as-textfile** | 将数据导入到普通文本文件（默认） |
| **--boundary-query <statement>** | 边界查询，用于创建分片（InputSplit） |
| **--columns <col,col,col…>** | 从表中导出指定的一组列的数据 |
| **--delete-target-dir** | 如果指定目录存在，则先删除掉 |
| **--direct** | 使用直接导入模式（优化导入速度） |
| **--direct-split-size <n>** | 分割输入stream的字节大小（在直接导入模式下） |
| **--fetch-size <n>** | 从数据库中批量读取记录数 |
| **--inline-lob-limit <n>** | 设置内联的LOB对象的大小 |
| **-m,--num-mappers <n>** | 使用n个map任务并行导入数据 |
| **-e,--query <statement>** | 导入的查询语句 |
| **--split-by <column-name>** | 指定按照哪个列去分割数据 |
| **--table <table-name>** | 导入的源表表名 |
| **--target-dir <dir>** | 导入HDFS的目标路径 |
| **--warehouse-dir <dir>** | HDFS存放表的根路径 |
| **--where <where clause>** | 指定导出时所使用的查询条件 |
| **-z,--compress** | 启用压缩 |
| **--compression-codec <c>** | 指定Hadoop的codec方式（默认gzip） |
| **--null-string <null-string>** | 如果指定列为字符串类型，使用指定字符串替换值为null的该类列的值 |
| **--null-non-string <null-string>** | 如果指定列为非字符串类型，使用指定字符串替换值为null的该类列的值 |
| **--import-all-tables** | 导入所有表 |

应用场景

1. **导入HDFS**

默认情况下，使用导入语句，直接导入到HDFS当前用户的目录下面，生成相应的表明，文件夹名称默认为表名。

默认（在行命令的机器上添加驱动）：

sqoop import --connect jdbc:mysql://crxy172:3306/testsqoop --username 'root' --password '123456' --table info

指定到入目录：

sqoop import --connect jdbc:mysql://crxy172:3306/testsqoop --username 'root' --password '123456' --table info –target-dir info\_dir

如果目录以及存在，使用--delete-target-dir：

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info --delete-target-dir

1. **指定map个数**

默认启动4个进程导入（map数量），可以设置 1表示不并行

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info -m 1

1. **增量导入**

|  |  |
| --- | --- |
| 参数 | 说明 |
| –check-column (col) | 用来作为判断的列名，如id |
| –incremental (mode) | append：追加，比如对大于last-value指定的值之后的记录进行追加导入。lastmodified：最后的修改时间，追加last-value指定的日期之后的记录 |
| –last-value (value) | 指定自从上次导入后列的最大值（大于该指定的值），也可以自己设定某一值 |

同一个目录进行数据追加

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info -m 1 --append

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info -m 1 --append --check-column 'id' --incremental append --last-value 23

1. **指定条件**

指定条件时需要注意两点：第一、不能含中文 ；第二、"job='CTO'"

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info -m 1 --append --where "job='CTO'" （可以使用模糊批量匹配）

1. **启用压缩**

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info -m 1 --append --where "job like 'CTO'" -z （默认Gzip压缩）

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root -password 123456 --table info -m 1 --append --where "job like 'CTO'" -z --compression-codec org.apache.hadoop.io.compress.SnappyCodec （直接指定压缩编码）

1. **导入空值处理**

字符串类型：

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info -m 1 --append --null-string "--" （不可以，不能是关键字符）

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info -m 1 --append --null-string "\*"

非字符串类型：

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info -m 1 --append --null-string "\*" --null-non-string "="

1. **Sql导入**

sqoop import --connect jdbc:mysql://crxy172:3306/test --username 'root' --password '123456' --query "select name,job from info where \$CONDITIONS" -m 1 --target-dir queryinfo --fields-terminated-by "," --split-by ","

1. **导入hive**

| **参数** | **说明** |
| --- | --- |
| --hive-home <dir> | 直接指定hive安装目录 |
| --hive-import | 使用默认分隔符导入hive |
| --hive-overwrite | 覆盖掉在hive表中已经存在的数据 |
| --create-hive-table | 生成与关系数据库表的表结构对应的HIVE表。如果表不存在，则创建，如果存在，报错。 |
| --hive-table <table-name> | 导入到hive指定的表，可以创建新表 |
| --hive-drop-import-delims | 导入数据到hive时，删除字符串字段中的 \n, \r, and \01 |
| --hive-delims-replacement | 用自定义的字符串替换掉数据中的\n, \r, and \01等字符 |
| --hive-partition-key | 创建分区，后面直接跟分区名即可，创建完毕后，通过describe 表名可以看到分区名，默认为string型 |
| --hive-partition-value <v> | 该值是在导入数据到hive中时，与–hive-partition-key设定的key对应的value值。 |
| --map-column-hive <map> | 生成hive表时，可以更改生成字段的数据类型，格式如：–map-column-hive LAST\_ACCESS\_TIME=string |
| --fields-terminated-by | 指定分隔符(hive默认的分隔符是/u0001) |

**追加数据**

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info --hive-import -m 1

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info --hive-import -m 1 --fields-terminated-by "|"

（如果数据表已经存在，并且字段不一样，指定分隔符时会把加到第一列中，不指定时取mysql第一列id值加到hive中）

**覆盖数据**

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info --hive-import -m 1 --fields-terminated-by "|" --hive-overwrite （只覆盖数据，不覆盖表结构）

**创建hive表（自定义表名）**

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info --hive-import -m 1 --fields-terminated-by "|" --hive-table "info\_t" --hive-overwrite （如果表已经存在，不会删除，只能重新定义表名）

注意：如果导出过程失败，需要到HDFS目录当前用户目录下将临时目录删除，hdfs当前用户目录下不能存在表目录

***复制mysql的表结构到Hive***

sqoop create-hive-table --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info --hive-table users --fields-terminated-by "\0001" --lines-terminated-by "\n"

***导出所有的表到hive***

sqoop import-all-tables --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --hive-import --fields-terminated-by "\0001" --lines-terminated-by "\n"

1. **导入HBase**

|  |  |
| --- | --- |
| **选项** | **说明** |
| **--hbase-table** | 指定导入到hbase中的表 |
| **--column-family** | 创建列族 |
| **--hbase-row-key <id>** | 以id字段作为key |
| **--hbase-create-table** | 创建hbase表 |

导入hive时，默认以主键作为key，没有主键使用--split-by ，暂时处理不了联合主键情况。

sqoop import --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info --hbase-create-table --hbase-row-key id --hbase-table info --column-family xxx

3、数据导出工具export

所谓export，是将HDFS平台的数据导入到关系数据库，基本命令选项如下：

|  |  |
| --- | --- |
| **选项** | **含义说明** |
| **--validate <class-name>** | 启用数据副本验证功能，仅支持单表拷贝，可以指定验证使用的实现类 |
| **--validation-threshold <class-name>** | 指定验证门限所使用的类 |
| **--direct** | 使用直接导出模式（优化速度） |
| **--export-dir <dir>** | 导出过程中HDFS源路径 |
| **-m,--num-mappers <n>** | 使用n个map任务并行导出 |
| **--table <table-name>** | 导出的目的表名称 |
| **--call <stored-proc-name>** | 导出数据调用的指定存储过程名 |
| **--update-key <col-name>** | 更新参考的列名称，多个列名使用逗号分隔 |
| **--update-mode <mode>** | 指定更新策略，包括：updateonly（默认）、allowinsert |
| **--input-null-string <null-string>** | 使用指定字符串，替换字符串类型值为null的列 |
| **--input-null-non-string <null-string>** | 使用指定字符串，替换非字符串类型值为null的列 |
| **--staging-table <staging-table-name>** | 在数据导出到数据库之前，数据临时存放的表名称（用于事务处理） |
| **--clear-staging-table** | 清除工作区中临时存放的数据 |
| **--batch** | 使用批量模式导出 |

场景应用：

1. **导出到mysql（默认使用逗号作为分隔**）

直接导出，导出时字段需要一一对应

sqoop export --connect jdbc:mysql://crxy172:3306/test --username root --password 123456 --table info --export-dir export

**乱码问题**

sqoop export --connect "jdbc:mysql://crxy172:3306/test?useUnicode=true&characterEncoding=utf-8" --username root --password 123456 --table info --export-dir export

1. **指定map数**

sqoop export --connect "jdbc:mysql://crxy172:3306/test?useUnicode=true&characterEncoding=utf-8" --username root --password 123456 --table info --export-dir export -m 1

1. **插入和更新**

**如果存在就更新，不存在就插入**

sqoop export --connect "jdbc:mysql://crxy172:3306/test?useUnicode=true&characterEncoding=utf-8" --username root --password 123456 --table info --export-dir export -m 1 --update-key id --update-mode allowinsert

1. **指定分隔符(和导入区别开来)**

导入导出的事务是以Mapper任务为单位。

注意： 1⃣--fields-terminated-by导入数据时（import）字段分割、行分割

2⃣ -input-fields-terminated-by解析HDFS上面的数据到数据库时使用参数

1. **从HBase到mysql**

没有直接导出方案。只能使用以下步骤，先通过mapreduce写入到HDFS在通过export工具导入到mysql。

1. **导出空值处理**

--input-null-string string类型使用参数

--input-null-non-string <null-string> 非string类型使用参数

1. 其他常用工具介绍
2. **Codegen**

|  |  |
| --- | --- |
| **参数** | **说明** |
| **–bindir <dir>** | 指定生成的java文件、编译成的class文件及将生成文件打包为JAR的JAR包文件输出路径 |
| **–class-name <name>** | 设定生成的Java文件指定的名称 |
| **–outdir <dir>** | 生成的java文件存放路径 |
| **–package-name<name>** | 包名，如cn.cnnic，则会生成cn和cnnic两级目录，生成的文件（如java文件）就存放在cnnic目录里 |
| **–input-null-non-string<null-str>** | 在生成的java文件中，可以将null字符串设为想要设定的值（比如空字符串’’） |
| **–input-null-string<null-str>** | 同上，设定时，最好与上面的属性一起设置，且设置同样的值（比如空字符串等等）。 |
| **–map-column-java<arg>** | 数据库字段在生成的java文件中会映射为各种属性，且默认的数据类型与数据库类型保持对应，比如数据库中某字段的类型为bigint，则在Java文件中 的数据类型为long型，通过这个属性，可以改变数据库字段在java中映射的数据类型，格式如：–map-column-java DB\_ID=String,id=Integer |
| **–null-non-string<null-str>** | 在生成的java文件中，比如TBL\_ID==null?”null”:””，通过这个属性设置可以将null字符串设置为其它值如ddd，TBL\_ID==null “ddd”:”” |
| **–null-string<null-str>** | 同上，使用的时候最好和上面的属性一起用，且设置为相同的值 |
| **–table <table-name>** | 对应关系数据库的表名，生成的java文件中的各属性与该表的各字段一一对应。 |

**编译jar包和record类**

sqoop codegen --connect jdbc:mysql://crxy172:3306/test --username 'root' --password '123456' --table info --bindir sqoopjar

**直接使用已经编译好的类**

sqoop import --connect jdbc:mysql://crxy172:3306/test --username 'root' --password '123456' --table info --class-name info --jar-file info.jar

导出过程生成的和Codegen生成代码区别：

1. 导出过程生成代码纯属于副产品，无法控制，默认和表名一样
2. Codegen可以指定生成代码的参数，可以用来重新生成导入过程的源代码

主要作用：a) 可以将需要导入的数据事先序列化到HDFS中

b) 检查数据表，采用最合适的数据类型

c) 如果事先已经将数据序列化到了HDFS，可以采用该方式读取出来

1. **Eval**

直接使用sql查询mysql

sqoop eval --connect jdbc:mysql://crxy172:3306/test --username 'root' --password '123456' --query "select \* from info"

插入操作：

sqoop eval --connect "jdbc:mysql://crxy172:3306/test?useUnicode=true&characterEncoding=utf-8" --username 'root' --password '123456' --query "insert into info values (24,'超人','飞人',1)"

1. **job生成**

* 创建job：--create
* 删除job：--delete
* 执行job：--exec
* 显示job：--show
* 列出job：--list

sqoop job --create myjob -- import --connect jdbc:mysql://crxy172:3306/test --username 'root' --password '123456' --table info

sqoop job --list

sqoop job --exec myjob

5、常见处理

1）事务处理

由于线程的并发性，一个导入操作可能并不是原子性的。会一次statement插入100条数据，然后每100个statement提交一次，所以一次就会提交10000条数据。如果tasks失败了（由于网络问题或者其它的问题），这些tasks会尝试从它们开始导入数据的地方重新开始，会插入重复的记录。这次写数据的时候，Sqoop不提防这种潜在的问题。Sqoop提供的一个解决办法就是使用中间表，参数为：

--staging-table <staging-table-name>

--clear-staging-table

2）字段名称问题

sqoop 从mysql导入hive的字段名称问题

hive中有些关键字限制，因此有些字段名称在mysql中可用，但是到了hive就不行。部分不能在hive中使用的字段名称

order；sort；reduce；cast；directory

3）参数处理

部分字段含有特殊字符时需要添加双引号，单双引号都有时，一般采用双引号套单引号。

6、sqoop2架构

