

华为认证 Big Data 系列教程

HCIA - Big Data V2.0

大数据工程师

实验指导手册

版本:2.0



华为技术有限公司

版权所有 © 华为技术有限公司 2017。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编： 518129

网址： <http://e.huawei.com>

前言

简介

本实验手册所有实验均以华为 FusionInsight HD 大数据平台作为实验环境，指导学生完成 HCIA-Big Data 课程所规定的实验任务，旨在使学员掌握 FusionInsight HD 平台大数据重要组件的使用方法。

内容描述

本实验指导书共包含 8 个实验小节：FusionInsight 客户端安装、HBase 数据库实战、HDFS 文件系统实战、Loader 数据导入导出实战、Flume 数据采集实战、Kafka 消息订阅实战、Hive 数据仓库实战、集群综合实验等。

实验须知

学员在实验过程中，不得随意删除文件。

学员在对目录、topic、文件进行命名时，均须包含学员账号 stuxx 或 userxx 字样，如目录 stu06_data，表 user01_socket。

学员登录环境所需用户名和密码由讲师统一管理和分配，如有不清楚的，请咨询讲师。

参考文档

《FusionInsight HD 产品文档》。

实验环境说明

实验硬件及软件

服务器	最低配置	推荐配置
CPU	Intel 4 核 *2	Intel 8 核 *2
Bit-mode	64 位	64 位
内存	48GB	64GB
网卡	2 张千兆网卡	2 张千兆网卡
硬盘数量	Disk * 7	Disk *7

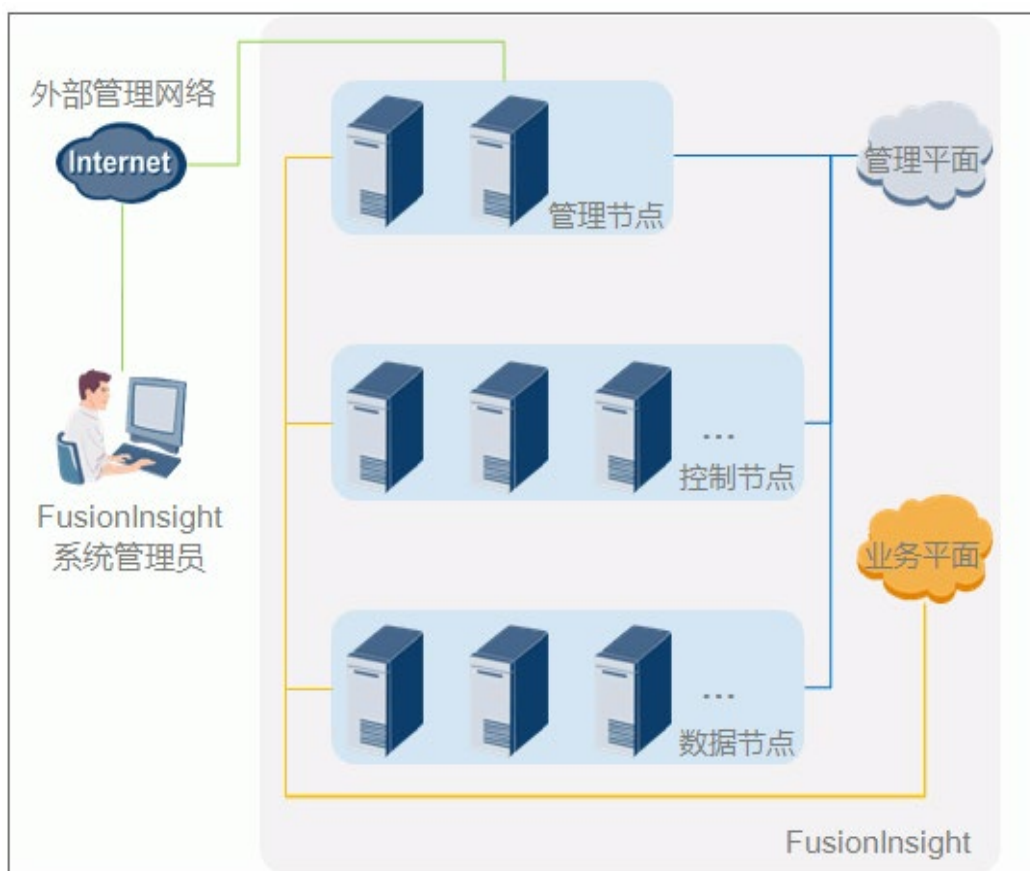
硬盘大小	Disk ≥ 610G	Disk ≥ 610G
操作系统版本	推荐 SUSE Linux Enterprise Server 11 SP3 (SuSE11.3)	

其他硬件：

交换机	最低配置全 1Gb 口以太网交换机，推荐配置全 10Gb 口以太网交换机
-----	--------------------------------------

实验拓扑

本实验采用 3 个服务器节点，每台服务器配置 2 张网卡，分别用于管理平面和业务平面。
服务器系统盘做 RAID1，数据盘做 RAID5。



学员账号及软件说明

每个学员分配 2 个账号：stu 开头的是 FusionInsight HD 集群账号，可用于登录 FusionInsight Manager 管理界面和组件之间通信进行认证，以及在访问大数据组件时的认证；user 开头的账号是集群节点 OS 账号，用于登录群节点操作系统，进行组件实验操作。

为方便学员，实验过程中所用到的集群客户端软件和文件放在每个集群节点的 /FusionInsight_Client 目录下，学员在使用的时候可以该目录下获取。

实验过程中所用到的 SSH 工具和文件上传工具放在 <ftp://10.175.199.8/> 下 07 other tool 目录下，ftp 用户名/密码为 admin1/admin1，学员可自行获取。

1 FusionInsight HD 客户端安装

1.1 实验背景

FusionInsight HD 客户端是用户与集群交互的接口，也是后续实验的基础。安装客户端后，在集群安全模式部署的情况下，需进行安全认证才可与集群进行交互。

1.2 实验目的

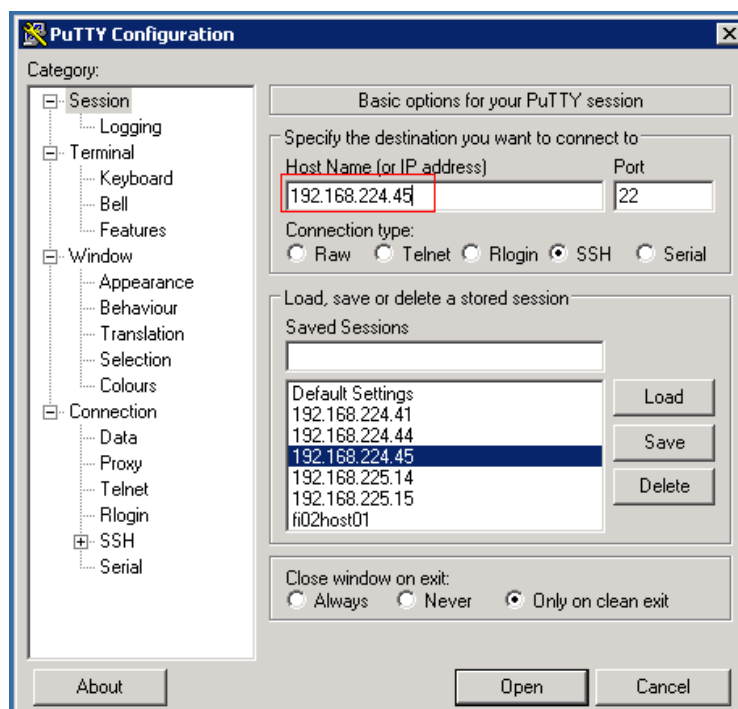
- 掌握客户端的下载、安装方法。

1.3 实验任务

1.3.1 安装客户端

步骤 1 登录集群节点。

使用 putty 工具，学员账号登录集群节点，例如 192.168.224.45，stu01(具体节点 IP 地址须由讲师分配)。



复制 FusionInsight HD 客户端到 user01 的 home 目录下，客户端文件在每台集群节点的 /FusionInsight_Client 目录下。

```
> cd /FusionInsight_Client
> cp FusionInsight_V100R002C60SPC200_Services_ClientConfig.tar /home/user01
```

步骤 2 解压客户端软件。

```
> cd /home/user01
> tar -xvf FusionInsight_V100R002C60SPC200_Services_ClientConfig.tar
```

步骤 3 安装客户端。

进入 FusionInsight_V100R002C60SPC200_Services_ClientConfig 目录，并执行安装命令，将软件安装在当前用户的 home 目录下 “/home/user01/1001_hadoopclient”。

```
> cd /home/user01/FusionInsight_V100R002C60SPC200_Services_ClientConfig/
> ./install.sh /home/user01/hadoopclient
```

系统提示 Components client installation is complete 表示安装完成。

步骤 4 执行环境变量文件并完成认证。

进入/home/user01/hadoopclient，执行：source bigdata_env 完成环境变量的设置。

进行安全认证：kinit stu01 输入密码 Huawei@123，完成认证。

执行如下命令进行环境变量设置

```
> source hadoopClient/bigdata_env
> kinit stu01
```

Password for admin@HADOOP.COM:

注：初始密码 Huawei@123（或咨询授课老师），第一次认证若提示修改，密码统一修改成 Huawei12#\$。

步骤 5 测试客户端。

使用 hdfs 命令测试客户端：

```
> hdfs dfs -ls /
drwxr-x---+ - flume   hadoop           0 2017-07-15 00:39 /flume
drwx-----+ - hbase   supergroup      0 2018-03-31 10:28 /hbase
drwxrwxr-x+ - admin   supergroup      0 2018-01-28 15:52 /mapreduceInput
drwxrwxrwx+ - mapred  hadoop           0 2017-07-15 00:39 /mr-history
```

测试成功，标明客户端安装成功！

--结束！

1.4 实验小结

本实验主要讲述 FusionInsight HD 客户端的安装，在安装过程中，客户端软件要解压两次，需要注意的是指定安装客户端的目录下不能有文件或文件夹，否则安装失败。

2 HDFS 文件系统实战

2.1 实验背景

HDFS 是 Hadoop 大数据平台中的分布式文件系统，为上层应用或其他大数据组件提供数据存储，如 Hive，Mapreduce，Spark，HBase 等。在 HDFS shell 客户端我们可以实现对分布式文件系统的操作和管理等。掌握 HDFS 的使用对我们更好的理解和掌握大数据大有裨益。

2.2 实验目的

- 掌握 HDFS 常用操作。
- 掌握 HDFS 文件系统管理操作。

2.3 实验任务

2.3.1 HDFS 常用操作

2.3.1.1 常用命令操作

步骤 1 `-help` 功能：查看命令使用说明。

```
> hdfs dfs -help
Usage: hadoop fs [generic options]
[-appendToFile <localsrc> ... <dst>]
[-cat [-ignoreCrc] <src> ...]
[-checksum <src> ...]
[-chgrp [-R] GROUP PATH...]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown 艘日 [-R] [OWNER] [:[GROUP]] PATH...]
[-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
[-count [-q] [-h] [-v] [-t [<storage type>]] <path> ...]
[-cp [-f] [-p | -p[topax]] <src> ... <dst>]
[-createSnapshot <snapshotDir> [<snapshotName>]]
[-deleteSnapshot <snapshotDir> <snapshotName>]
[-df [-h] [<path> ...]]
[-du [-s] [-h] <path> ...]
```

步骤 2 -ls 功能：显示目录信息。

```
~> hdfs dfs -ls /
-rw-r--r--+ 3 wkj    supergroup      13 2018-04-02 16:42 /HDFS
drwxrwxr-x+ - hive    supergroup      0 2017-07-15 00:43 /apps
drwxr-xr-x+ - admin   supergroup      0 2018-03-13 19:44 /bigdata
drwxr-x---+ - flume    hadoop          0 2017-07-15 00:39 /flume
drwx-----+ - hbase    supergroup      0 2018-03-31 10:28 /hbase
drwxrwxr-x+ - admin   supergroup      0 2018-01-28 15:52 /mapreduceInput
drwxrwxrwx+ - mapred   Hadoop          0 2017-07-15 00:39 /mr-history
```

步骤 3 -mkdir 功能：在 HDFS 文件系统上创建目录。

```
> hdfs dfs -mkdir /app_stu01
> hdfs dfs -ls /
drwxr-xr-x+ - wkj    supergroup      0 2018-04-02 17:20 /0402
drwxr-xr-x+ - wkj    supergroup      0 2018-04-02 16:57 /0810
-rw-r--r--+ 3 wkj    supergroup      13 2018-04-02 16:42 /HDFS
drwxr-xr-x+ - stu01   supergroup      0 2018-04-04 15:04 /app_stu01
```

步骤 4 -put 功能：上传本地文件到 HDFS 指定目录。

```
> hdfs dfs -put Service_Client.tar /app_stu01
> hdfs dfs -ls -h /app_stu01
-rw-r--r--+ 3 stu01 supergroup 2.7 G 2018-04-04 14:50
/app_stu01/Service_Client.tar
```

步骤 5 -get 功能：等同于 copyToLocal，就是从 hdfs 下载文件到本地。

拷贝/app_stu01/test01.txt 到本地

```
> hdfs dfs -get /app_stu01/test01.txt ./
> ll
total 2881728
drwxr-xr-x 15 stu01 hadoop      4096 Apr  4 10:58 1001_hadoopclient
-rw-r--r--  1 stu01 hadoop        63 Apr  4 16:30 appendtext.txt
drwxr-xr-x  2 stu01 hadoop      4096 Apr  4 10:03 bin
-rw-r--r--  1 stu01 hadoop         0 Apr  4 15:28 hdfs
-rwxr-xr-x  1 stu01 hadoop 2947983360 Apr  4 10:05 Service_Client.tar
-rw-r--r--  1 stu01 hadoop        38 Apr  4 16:27 stu01.txt
-rw-r--r--  1 stu01 hadoop        38 Apr  4 17:54 test01.txt
```

步骤 6 -moveFromLocal 功能：从本地剪切粘贴到 HDFS。

在 stu01 的 home 目录下面有 abcd 文件。

```
> ll
total 2881716
drwxr-xr-x 15 stu01 hadoop      4096 Apr  4 10:58 1001_hadoopclient
drwxr-xr-x  2 stu01 hadoop      4096 Apr  4 10:03 bin
-rw-r--r--  1 stu01 hadoop         0 Apr  4 15:28 abcd
-rwxr-xr-x  1 stu01 hadoop 2947983360 Apr  4 10:05 Service_Client.tar
```


使用 moveFromLocal 将 abcd 文件移动到 HDFS 文件系统的/app_stu01 目录下：

```
> hdfs dfs -moveFromLocal hdfs /app_stu01
```

执行结束后查看 stu01 的 home 本地目录，hdfs 文件已经没有了。

```
> ll
```

```
total 2881716
```

```
drwxr-xr-x 15 stu01 hadoop      4096 Apr  4 10:58 1001_hadoopclient
```

```
drwxr-xr-x  2 stu01 hadoop      4096 Apr  4 10:03 bin
```

```
-rwxr-xr-x  1 stu01 hadoop 2947983360 Apr  4 10:05 Service_Client.tar
```

文件已经被移动到 HDFS 文件系统中：

```
> hdfs dfs -ls -h /app_stu01
```

```
-rw-r--r--+  3 stu01 supergroup          0 2018-04-04 15:04 /app_stu01/hdfs
```

步骤 7 -cat 功能：显示文件内容。

```
> hdfs dfs -cat /app_stu01/stu01.txt
```

```
01,HDFS
```

```
02,Zookeeper
```

```
03,HBase
```

```
04,Hive
```

步骤 8 -appendToFile 功能：在文件末尾追加数据。

在本地有文件 appendtext.txt，其内容为：

```
> cat appendtext.txt
```

```
10,Spark
```

```
11,Storm
```

```
12,Kafka
```

```
13,Flink
```

```
14,ELK
```

```
15,FusionInsight HD
```

将 appendtext.txt 中的内容追加到 stu01.txt 末尾：

```
> hdfs dfs -appendToFile ./appendtext.txt /app_stu01/stu01.txt
```

查看追加结果：

```
> hdfs dfs -cat /app_stu01/stu01.txt
```

```
01,HDFS
```

```
02,Zookeeper
```

```
03,HBase
```

```
04,Hive
```

```
10,Spark
```

```
11,Storm
```

```
12,Kafka
```

```
13,Flink
```

```
14,ELK
```

```
15,FusionInsight HD
```

步骤 9 -chmod 功能：更改文件所属权限。

```
> hdfs dfs -ls /app_stu01
```

```
-rw-r--r--+ 3 stu01 supergroup 2.7G 2018-04-04 14:50 /app_stu01/Service_Client.tar
-rw-r--r--+ 3 stu01 supergroup 0 2018-04-04 15:04 /app_stu01/hdfs
-rw-r--r--+ 3 stu01 supergroup 101 2018-04-04 16:32 /app_stu01/stu01.txt
```

将/app_stu01/stu01.txt 文件权限属性改为 755:

```
>hdfs dfs -chmod 755 /app_stu01/stu01.txt
> hdfs dfs -ls /app_stu01/stu01.txt
-rwxr-xr-x+ 3 stu01 supergroup 101 2018-04-04 16:32 /app_stu01/stu01.txt
```

说明: chown 的使用需要 superuser 权限。

步骤 10 -cp 功能: 实现文件的拷贝。

将/app_stu01/stu01.txt 拷贝到/tmp 下:

```
> hdfs dfs -cp /app_stu01/stu01.txt /tmp/
> hdfs dfs -ls /tmp
drwxrwxr-x+ - admin supergroup 0 2018-01-21 20:58 /tmp/checkpoint
-rw-r--r--+ 3 stu01 supergroup 4651 2018-03-19 19:19 /tmp/conf.py
-rw-r--r--+ 3 stu01 hadoop 101 2018-04-04 17:12 /tmp/stu01.txt
```

步骤 11 -mv 功能: 移动文件。

将/app_stu01/stu01.txt 移动到/user 目录下

```
> hdfs dfs -mv /app_stu01/stu01.txt /user/
> hdfs dfs -ls /user
-rwxr-xr-x+ 3 stu01 supergroup 101 2018-04-04 16:32 /user/stu01.txt
```

步骤 12 -getmerge 功能: 合并下载多个文件。

在/app_stu01 目录下有 2 个文件 file01, test01.txt

```
> hdfs dfs -ls /app_stu01/
-rw-r--r--+ 3 stu01 supergroup 120 2018-04-08 09:03 /app_stu01/file01
-rw-r--r--+ 3 stu01 supergroup 38 2018-04-04 17:46 /app_stu01/test01.txt
```

并且两个文件的内容如下:

```
> hdfs dfs -cat /app_stu01/file01
001 FusionInsight HD
002 FusionInsight Miner
003 FusionInsight LibrA
004 FusionInsight Farmer
005 FusionInsight Manager
fi01host01:~> hdfs dfs -cat /app_stu01/test01.txt
01,HDFS
02,Zookeeper
03,HBase
04,Hive
```

将文件进行合并后, 并拷贝到本地目录:

```
> hdfs dfs -getmerge /app_stu01/ Merge_file
> cat Merge_file
001 FusionInsight HD
```

```
002 FusionInsight Miner
003 FusionInsight LibrA
004 FusionInsight Farmer
005 FusionInsight Manager
01,HDFS
02,Zookeeper
03,HBase
04,Hive
```

步骤 13 -rm 功能：删除文件或文件夹。

删除/app_stu01/file01 文件

```
> hdfs dfs -rm -f /app_stu01/file01
INFO fs.Trash: Moved: 'hdfs://hacluster/app_stu01/file01' to trash at:
hdfs://hacluster/user/stu01/.Trash/Current
```

步骤 14 -df 功能：统计文件系统的可用空间信息。

```
> hdfs dfs -df -h /
Filesystem      Size      Used Available Use%和
hdfs://hacluster 1.7 T  11.9 G      1.7 T    1%
```

步骤 15 -du 功能：统计文件夹的大小信息。

```
> hdfs dfs -du -h /user
213.1 M      /user/admin
0            /user/hdfs
75           /user/hdfs-examples
213.1 M      /user/hive
4.3 K        /user/loader
493          /user/mapred
```

步骤 16 -count 功能：统计一个指定目录下的文件数量。

```
> hdfs dfs -count -h /user/
344          494          3.2 G /user
```

第一列 344 表示/user/下文件夹的数量，第二列 494 表示/user/下文件的个数。3.2G 表示/user/目录下所有文件占用的磁盘容量（不计算副本个数）。

2.3.1.2 回收站使用

日常工作中，有时会误删文件。此时我们可以在 hdfs 的回收站中找回被误删的文件，回收站默认将被删除文件保存 7 天。例如在上述实验中，我们使用 -rm 参数删除了文件 file01，删除后，系统会提示被删除的文件被存放于 fs.Trash: Moved: 'hdfs://hacluster/app_stu01/file01' to trash at: hdfs://hacluster/user/stu01/.Trash/Current，但是 HDFS 系统会对被删除的文件进行归档，目录会有不同：

```
hdfs dfs -ls /user/stu01/.Trash/
> hdfs dfs -ls /user/stu01/.Trash/
..... 2018-04-08 09:10 /user/stu01/.Trash/180408100000
```

深入查看/user/stu01/.Trash/180408100000 目录

```
> hdfs dfs -ls -h /user/stu01/.Trash/180408100000/app_stu01
```

..... 2018-04-08 09:03 /user/stu01/.Trash/180408100000/app_stu01/**file01**

然后使用-mv 参数，将文件移动到指定目录即可，具体使用方法可参考文中-mv 部分。

2.3.2 HDFS 文件系统管理操作

2.3.2.1 HDFS 配额管理

在有多个租户共同使用 HDFS 文件系统时，往往需要限定租户对 HDFS 空间大小的使用，此时需要用到 HDFS 的配额管理。

2.3.2.1.1 创建配额配置

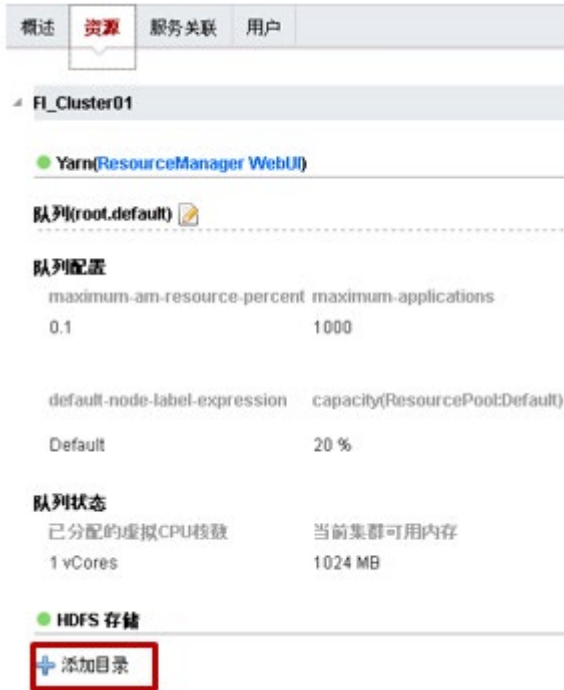
步骤 1 在 FusionInsight Manager 界面，单击“租户管理”。

在左侧租户列表，单击需要修改 HDFS 存储目录的租户 queueA。



步骤 2 单击“资源”页签。

步骤 3 在“HDFS 存储”表格，单击“添加目录”。



步骤 4 添加目录

添加目录

*

路径:

E.g. /PathName,hdfs://Cluster/PathName

文件目录数上限:

*

存储空间配额:

MB

确定

取消

“路径”：填写分配给租户使用的目录路径。如果目录不存在，系统将会自动创建。

“文件\目录数上限”：存储的文件数与目录数总和的上限值。

“存储空间配额”：创建目录的存储空间配额大小。

编辑目录

*

路径:

/stu01

文件目录数上限:

2

*

存储空间配额:

1000

MB

确定

取消

注：学员填写配置时，路径不可相同。

单击“确定”完成租户目录添加。

步骤 5 检查添加目录结果。

执行 HDFS 上传文件命令：

```
> hdfs dfs -put test.txt /stu01
```

查看文件上传结果，执行命令：

```
> hdfs dfs -ls /stu01
Found 1 items
-rw-r--r--+ 3 stu01 supergroup 62 2018-04-08 17:37 /stu01/test.txt
```

出现上述结果，表示目录/stu01 创建成功，且当前用户具备上传文件的权限！

步骤 6 测试“存储空间配额”。

磁盘空间预申请值=文件对应的 Block 数* blockSize*3，blockSize 默认大小为 128M，故磁盘预申请最小值（1 个数据块）为 128M*3=384M。步骤 4 中设置的目录存储空间配额为 1000M，因此最大文件大小为 2*128M=256M，当文件大于 256M 时，需要预申请至少 3 个数据块 3*128*3>1000M，配额空间无法满足需求，文件将会上传失败。（文件对应的 Block 数=文件大小/128，若无法除尽则 Block 数向上取整）。如下是在存储空间为 1000M 时，上传一个大于 256M 的文件，进行存储空间测试的例子。

执行如下命令拷贝 FusionInsight-Flume-1.6.0.tar.gz 到当前目录：

```
> cp /FusionInsight_Client/FusionInsight-Flume-1.6.0.tar.gz ./
```

查看 FusionInsight-Flume-1.6.0.tar.gz 文件大小信息：

```
> ll -h
-rw----- 1 stu01 wheel 296M Mar 19 19:00 FusionInsight-Flume-1.6.0.tar.gz
```

执行如下命令上传文件到 HDFS：

```
> hdfs dfs -put FusionInsight-Flume-1.6.0.tar.gz /stu01
put: The DiskSpace quota of /stu01 is exceeded:quota = 1048576000 B=1000 MB
but disk space consumed = 1207964949 B =1.13 GB
```

由此可见，在预设磁盘配额空间为 1000M 时，当文件大于 256M 时，文件无法上传成功。

步骤 7 测试“文件\目录上限”参数。

步骤 4 中所配置的文件夹与文件总数上限，在上传文件数目大于 2 个时，会出现上传文件失败。执行如下命令进行测试：

```
>hdfs dfs -put switchuser.py /stu01
>hdfs dfs -put install.bat /stu01
put: The NameSpace quota (directories and files) of directory /stu01 is
exceeded:quota = 2 file count=3
```

执行文件查看命令，查询 HDFS 指定路径下的文件列表：

```
>hdfs dfs -ls /stu01
Found 1 items
-rw-r--r--+ 3 stu01 supergroup 1799 2018-04-08 15:56 /stu01/switchuser.py
```

如上结果显示没有看到 install.bat 文件，表示该文件上传失败。

2.3.2.1.2 修改配额配置

步骤 1 修改“文件\目录上限”为 3，修改“存储空间配额”为 1500M。



步骤 2 重新上传数据进行测试，然后查看目录的文件列表，执行如下命令：

```
>hdfs dfs -put FusionInsight-Flume-1.6.0.tar.gz /stu01
>hdfs dfs -ls /stu01
Found 2 items
..... 2018-04-09 15:29 /stu01/FusionInsight-Flume-1.6.0.tar.gz
..... 2018-04-09 10:59 /stu01/c.txt
```

上述命令执行的结果表明，通过修改配置，完成了大文件(296M)的上传操作，同时也实现了更多文件(2 个)的存储。

2.3.2.1.3 删除配额

步骤 1 登录 FusionInsight Manager 管理界面，依次点击“租户管理”，“queueA”，“资源”页签到租户管理界面的资源页。



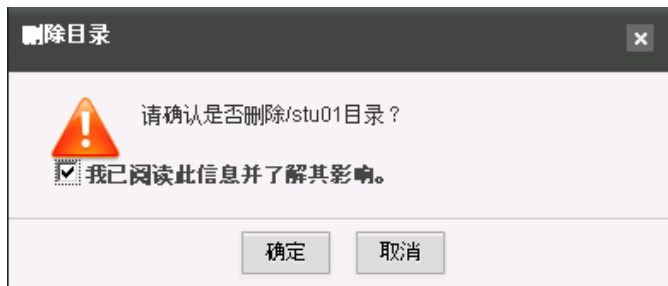
步骤 2 点击资源页下的“HDFS 存储”部分指定目录“操作”下的删除操作，执行删除存储资源操作。

HDFS 存储

+ 添加目录

路径	文件目录数上限	存储空间配额	操作
/tenant/queueA		1000 MB	 
/stu01	3	1500 MB	 

步骤 3 在弹出“删除目录”对话框，勾选复选框，点击确定，实现存储目录删除。



2.3.2.2 HDFS 元数据备份和恢复

为了确保 HDFS 元数据安全性，或者系统管理员需要对 HDFS 集群进行重大操作（如升级或迁移等），需要对 HDFS 元数据进行备份，以保证系统出现问题时，能够及时进行 HDFS 元数据恢复，保证 HDFS 集群数据的安全可靠。

2.3.2.2.1 数据备份

数据备份的具体操作如下：

步骤 1 选择“系统设置 > 备份管理”。



步骤 2 单击“创建备份任务”。



步骤 3 选择“NameNode”复选框，进行 NameNode 元数据备份任务参数配置，包括“任务名称”，“路径类型”，“最大备份数”，“实例名称”等参数的配置，之后点击确定。

系统设置 > 备份管理 > 创建备份任务

创建备份任务

* 任务名称: NameNodeBackup ✓

备份类型: ☐ 周期备份 ☒ 手动备份

* 备份配置:

元数据

☐ DBService

☒ NameNode

路径配置

* 路径类型: LocalDir ✓

* 最大备份数: 3 ✓

* 实例名称: hacluster

☐ LdapServer

☐ OMS

步骤 4 点击操作中的“开始”，执行元数据备份任务。

当次任务状态	当次任务进度	操作
准备	0%	      
失败	100%	      
失败	100%	      
记录 1 到 3 总记录数: 3 10 条/页		

步骤 5 当任务进度为 100%时，表示任务执行完成，HDFS 元数据备份成功。

当次任务状态	当次任务进度	操作
成功	100%	      
失败	100%	      
失败	100%	      
记录 1 到 3 总记录数: 3 10 条/页		

2.3.2.2.2 数据恢复

数据恢复主要依据数据备份的结果进行恢复，数据恢复的具体操作如下：

步骤 1 点击“系统设置”，在“任务”下拉列表中点击“备份管理”。



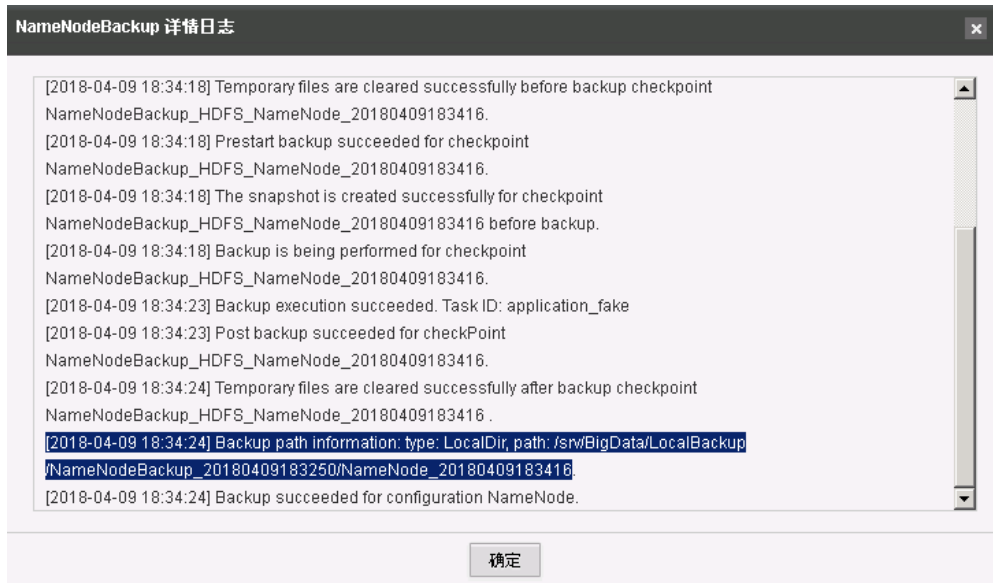
步骤 2 点击“NameNodeBackup”任务中的“查看历史”操作按钮。



步骤 3 查看数据备份日志，并点击操作中的“查看”。



步骤 4 从日志文件中获取备份数据存储文件的具体路径，如下图所示。



步骤 5 复制框中路径，点击“系统设置”中“任务列表”下的“恢复管理”，点击创建恢复任务。



步骤 6 进入如下“恢复管理”界面，点击“创建恢复任务”。



步骤 7 进行恢复任务配置，包括“任务名称”，“路径类型”，“源端路径”，“实例名称”等参数的设置，其中源端路径表示的是步骤 4 中获取的文件路径。参数配置完成之后，点击确定。

创建恢复任务

* 任务名称:

sdfsdas

✓

* 恢复配置:

元数据

☐ DBService
 ☒ NameNode

路径配置

* 路径类型:

LocalDir

✓

* 源端路径:

/srv/BigData/LocalBackup/NameNodeBackup_20

✓

* 实例名称:

hacluster

✓

☐ LdapServer
 ☐ OMS

步骤 8 点击对应任务后面的“开始”操作，进行数据恢复。



上图运行结果显示，NameNode 数据恢复成功。

2.4 实验小结

本实验主要讲述 HDFS 的常见操作以及 HDFS 的管理，通过本实验，学员可掌握常见的 HDFS 使用和管理操作。

3 HBase 数据库实战

3.1 实验背景

HBase 是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，是行业中最常用的 NoSQL 数据库。掌握 HBase 的使用，可加深学员对 HBase 的理解，为综合应用大数据打下坚实的基础。

3.2 实验目的

- 掌握 HBase 的常用操作、region 操作及 Filter 的使用。

3.3 实验任务

3.3.1 HBase 常用操作

3.3.1.1 进入 HBase shell 客户端

步骤 1 进入 HBase shell 客户端。

```
> cd /home/user01/hadoopClient
> source bigdata_env
> kinit stu01
Password for stu01@HADOOP.COM:
> hbase shell
.....
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.0.2, rUnknown, Thu May 12 17:02:55 CST 2016
hbase(main):001:0>
```

上述结果表明，已经成功进入 HBase shell 客户端。

3.3.1.2 创建普通表

步骤 1 创建普通表的语法为：create '表的名称'，'列族的名称'。

输入命令：

```
> create 'cga_info','info'
0 row(s) in 0.3620 seconds
```

```
=> Hbase::Table - cga_info
```

创建表“cga_info”成功。

步骤 2 `-list` 功能：查看系统中共有多少个普通表。

```
> list
TABLE
cga_info
Socket
t1
3 row(s) in 0.2300 seconds
=> ["cga_info", "socket", "t1"]
```

由此看出系统中已经存在了 3 个普通表。

3.3.1.3 创建 namespace

创建 namespace 的语法为：create_namespace ‘名称’。

```
> create_namespace 'nn'
0 row(s) in 0.1280 seconds
```

3.3.1.4 在指定 namespace 下创建表

在指定 namespace 下创建表：create ‘namespace 的名称：表名’，‘列族’。

```
> create 'nn:student', 'info'
0 row(s) in 0.2680 seconds
=> Hbase::Table - nn:student
```

3.3.1.5 查看指定 namespace 下的表

查看指定 namespace 下的表：list_namespace_tables ‘namespace 的名称’。

```
> list_namespace_tables 'nn'
TABLE
student
1 row(s) in 0.0220 seconds
```

3.3.1.6 增加数据

增加数据：put ‘表的名称’，‘RowKey’，‘列的名称’，‘具体的赋值’。

将一个名字为 Kobe，居住在洛杉矶的 40 岁男人的信息输入到表“cga_info”中：

```
> put 'cga_info', '123001', 'info:name', 'Kobe'
0 row(s) in 0.1580 seconds
> put 'cga_info', '123001', 'info:gender', 'male'
0 row(s) in 0.0390 seconds
> put 'cga_info', '123001', 'info:age', '40'
0 row(s) in 0.0250 seconds
> put 'cga_info', '123001', 'info:address', 'Los Angeles'
0 row(s) in 0.0170 seconds
```

3.3.1.7 get 方式查询数据

步骤 1 **get** 功能：精确查询。

精确查询某一个 RowKey 中存储的内容：get '表的名称', 'RowKey'

```
> get 'cga_info','123001'
COLUMN          CELL
info:address     timestamp=1523350574004, value=Los Angeles
info:age         timestamp=1523350540131, value=40
info:gender      timestamp=1523350499780, value=male
info:name        timestamp=1523350443121, value=Kobe
4 row(s) in 0.0540 seconds
```

步骤 2 精确查询某一个 RowKey 中的一个单元格中存储的内容。

语法：get '表的名称', 'RowKey', '列名'

```
> get 'cga_info','123001','info:name'
COLUMN          CELL
info:name       timestamp=1523350443121, value=Kobe
1 row(s) in 0.0310 seconds
```

3.3.1.8 scan 方式查询数据

步骤 1 按照 4.3.1.6 的方法，在表中多输入多条不同的数据。

步骤 2 **scan** 功能：在某个范围内查询。

查询表中某个列族下所有列的信息：scan '表的名称', {Columns=>'列'}

```
> scan 'cga_info',{COLUMNS=>'info'}
ROW          COLUMN+CELL
123001      column=info:address, timestamp=1523350574004, value=Los Angeles
123001      column=info:age, timestamp=1523350540131, value=40
123001      column=info:gender, timestamp=1523350499780, value=male
123001      column=info:name, timestamp=1523350443121, value=Kobe
123002      column=info:address, timestamp=1523351932415, value=London
123002      column=info:age, timestamp=1523351887009, value=40
123002      column=info:gender, timestamp=1523351993106, value=female
123002      column=info:name, timestamp=1523351965188, value=Victoria
123003      column=info:address, timestamp=1523352194766, value=Redding
123003      column=info:age, timestamp=1523352108282, value=30
123003      column=info:gender, timestamp=1523352060912, value=female
123003      column=info:name, timestamp=1523352091677, value=Taylor
123004      column=info:address, timestamp=1523352217267, value=Cleveland
123004      column=info:age, timestamp=1523352229436, value=33
123004      column=info:gender, timestamp=1523352267416, value=male
123004      column=info:name, timestamp=1523352251926, value=LeBron
4 row(s) in 0.0480 seconds
```

步骤 3 查询表中具体的一个列中存储的信息。

语法：scan '表的名称', {Columns=>'列的具体名称'}

```
> scan 'cga_info',{COLUMNS=>'info:name'}
ROW                                COLUMN+CELL
123001                            column=info:name, timestamp=1523350443121, value=Kobe
123002                            column=info:name, timestamp=1523351965188, value=Victoria
123003                            column=info:name, timestamp=1523352091677, value=Taylor
123004                            column=info:name, timestamp=1523352251926, value=LeBron
4 row(s) in 0.0300 seconds
```

3.3.1.9 指定条件查询数据

步骤 1 查询 RowKey 为“123002”和“123003”中的数据。

```
> scan 'cga_info',{STARTROW=>'123002','LIMIT'=>2}
ROW                                COLUMN+CELL
123002                            column=info:address, timestamp=1523351932415, value=London
123002                            column=info:age, timestamp=1523351887009, value=40
123002                            column=info:gender, timestamp=1523351993106, value=female
123002                            column=info:name, timestamp=1523351965188, value=Victoria
123003                            column=info:address, timestamp=1523352194766, value=Redding
123003                            column=info:age, timestamp=1523352108282, value=30
123003                            column=info:gender, timestamp=1523352060912, value=female
123003                            column=info:name, timestamp=1523352091677, value=Taylor
2 row(s) in 0.0170 seconds
```

步骤 2 查询 Rowkey 为“123001”和“123002”中列名称为 name 的单元格中存储的信息。

```
> scan 'cga_info',{STARTROW=>'123001','LIMIT'=>2,COLUMNS=>'info:name'}
ROW                                COLUMN+CELL
123001                            column=info:name, timestamp=1523350443121, value=Kobe
123002                            column=info:name, timestamp=1523351965188, value=Victoria
2 row(s) in 0.0500 seconds
```

注：除了列（COLUMNS）修饰词外，HBase 还支持 Limit（限制查询结果行数），STARTROW（ROWKEY 起始行。会先根据这个 key 定位到 region，再向后扫描）、STOPROW（结束行）、TIMERANGE（限定时间戳范围）、VERSIONS（版本数）、和 FILTER（按条件过滤行）等参数。

3.3.1.10 更新数据

步骤 1 首先查询表中 Rowkey 为 123001 的年龄信息。

```
> get 'cga_info','123001','info:age'
COLUMN                            CELL
info:age                          timestamp=1523350540131, value=40
1 row(s) in 0.0260 seconds
```

步骤 2 更改表中 Rowkey 为 123001 的年龄信息。

```
> put 'cga_info','123001','info:age','18'
0 row(s) in 0.0340 seconds
```

步骤 3 再次查询表中 Rowkey 为 123001 的年龄信息。


```
> get 'cga_info','123001','info:age'
COLUMN          CELL
info:age         timestamp=1523353910053, value=18
1 row(s) in 0.0040 seconds
```

由步骤 2 和步骤 3 的结果比较可得，年龄信息已经被更新。

3.3.1.11 删除数据

3.3.1.11.1 使用 delete 删除某一列数据

步骤 1 首先查询表中 Rowkey 为 123001 的信息。

```
> get 'cga_info','123001'
COLUMN          CELL
info:address     timestamp=1523350574004, value=Los Angeles
info:age         timestamp=1523353910053, value=18
info:gender      timestamp=1523350499780, value=male
info:name        timestamp=1523350443121, value=Kobe
4 row(s) in 0.0380 seconds
```

步骤 2 使用 delete 删除 123001 中 age 列所存储的数据。

```
> delete 'cga_info','123001','info:age'
0 row(s) in 0.0300 seconds
```

步骤 3 再次查询表中 Rowkey 为 123001 的信息。

```
> get 'cga_info','123001'
COLUMN          CELL
info:address     timestamp=1523350574004, value=Los Angeles
info:gender      timestamp=1523350499780, value=male
info:name        timestamp=1523350443121, value=Kobe
3 row(s) in 0.0220 seconds
```

由步骤 1 和步骤 3 的结果比较可得，年龄信息已经被删除了。

3.3.1.11.2 使用 deleteall 删除整行数据

步骤 1 使用 deleteall 删除表 cga_info 中 123001 的整行数据。

```
> deleteall 'cga_info','123001'
0 row(s) in 0.0320 seconds
```

步骤 2 再次查询表中 Rowkey 为 123001 的信息。

```
> get 'cga_info','123001'
COLUMN          CELL
0 row(s) in 0.0190 seconds
```

此时表中已经没有 RowKey 为 123001 的信息，说明行数据删除成功。

3.3.1.11.3 使用 drop 删除数据表

步骤 1 创建表名为 cga_info1 的新表。

```
> create 'cga_info1','info'
0 row(s) in 0.3920 seconds
=> Hbase::Table - cga_info1
```

步骤 2 首先 disable '表的名称'，然后再使用 drop '表的名称'删除数据表。

```
> disable 'cga_info1'
0 row(s) in 1.2270 seconds
> drop 'cga_info1'
2018-04-10 18:12:23,566 INFO [main] client.HBaseAdmin: Deleted cga_info1
0 row(s) in 0.3940 seconds
```

步骤 3 查询当前命名空间下的表。

```
> list
TABLE
cga_info
Socket
t1
3 row(s) in 0.2300 seconds
=> ["cga_info", "socket", "t1"]
```

结果显示表 cga_info1 已经被删除了。

3.3.2 Filter 过滤器使用

Filter 允许在 Scan 过程中，设置一定的过滤条件，符合条件的用户数据才返回，所有的过滤器都在服务端生效，以保证被过滤掉的数据不会传送到客户端。

示例 1: 查询年龄为 40 的人。

```
> scan 'cga_info',{FILTER=>"ValueFilter(=,'binary:40')"}
ROW COLUMN+CELL
123002 column=info:age, timestamp=1523351887009, value=40
1 row(s) in 0.1230 seconds
```

示例 2: 查询名叫 LeBron 的人。

```
> scan 'cga_info',{FILTER=>"ValueFilter(=,'binary:LeBron')"}
ROW COLUMN+CELL
123004 column=info:name, timestamp=1523352251926, value=LeBron
1 row(s) in 0.2240 seconds
```

示例 3: 查询表中所有人的性别信息。

```
> scan 'cga_info',FILTER=>"ColumnPrefixFilter('gender') "
ROW COLUMN+CELL
123002 column=info:gender, timestamp=1523351993106, value=female
123003 column=info:gender, timestamp=1523352060912, value=female
123004 column=info:gender, timestamp=1523352267416, value=male
3 row(s) in 0.0570 seconds
```

示例 4: 查询表中所有人的地址信息并且找出住在伦敦的人。

```
> scan 'cga_info',{FILTER=>"ColumnPrefixFilter('address') AND ValueFilter(=,  
'binary:London')"}  
ROW                                COLUMN+CELL  
123002                             column=info:address, timestamp=1523351932415, value=London  
1 row(s) in 0.0100 seconds
```

Filter 可以根据列族，列，版本等更多的条件来对数据进行过滤，这里只演示了 4 种过滤方式，带有过滤条件的 RPC 查询请求会把过滤器分发到各个 RegionServer，这样可以降低网络传输的压力。

3.3.3 创建预分 region 表

3.3.3.1 以 rowkey 切分，随机分为 4 个 region

步骤 1 创建一个新的表 “cga_info2” 并且划分成 4 个 region。

```
create '表的名称','列族的名称',{NUMREGIONS=>4,SPLITALGO=>'UniformSplit'}  
> create 'cga_info2','info',{NUMREGIONS=>4,SPLITALGO=>'UniformSplit'}  
0 row(s) in 0.3720 seconds  
=> Hbase::Table - cga_info2
```

步骤 2 进入 FusionInsight Manager 界面，点击服务管理，然后点击 “HBase”。

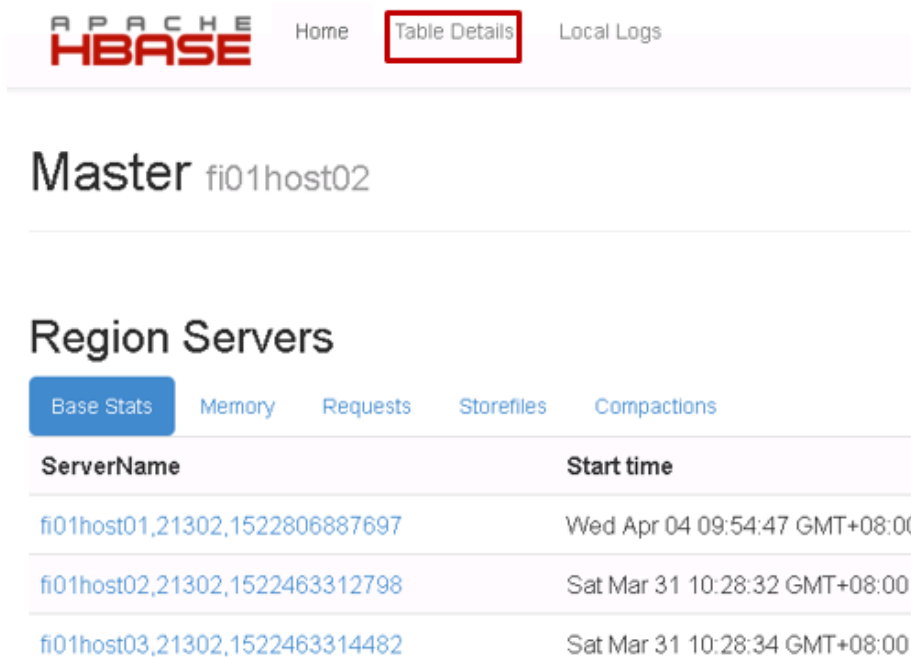


步骤 3 点击 “HMaster(主)”。



The screenshot shows the FusionInsight Manager interface. The top navigation bar includes '系统概览', '服务管理' (highlighted), '主机管理', '告警管理', and '审计管理'. Below this, the 'HBase 服务状态' (HBase Service Status) page is displayed. It features tabs for '服务状态' (Service Status), '实例' (Instances), '服务配置' (Service Configuration), and '资源贡献排名' (Resource Contribution Ranking). The '服务状态' tab is active, showing a summary of HBase services. The '健康状态' (Health Status) is '良好' (Good), and the '配置状态' (Configuration Status) is '已同步' (Synchronized). The version is '1.0.2', and there are '0' requests and '0' flush operation queue sizes. A red box highlights the 'HMaster(主)' link, which is the primary HMaster WebUI.

步骤 4 点击 “Table Details”。



The screenshot shows the Apache HBase 'Table Details' page for the 'Master' table. The page has a navigation bar with 'Home', 'Table Details' (highlighted), and 'Local Logs'. Below the navigation bar, the table name 'Master' and its identifier 'fi01host02' are displayed. The 'Region Servers' section is visible, showing a table with columns 'ServerName' and 'Start time'. The table lists three region servers: 'fi01host01,21302,1522806887697' (Wed Apr 04 09:54:47 GMT+08:00), 'fi01host02,21302,1522463312798' (Sat Mar 31 10:28:32 GMT+08:00), and 'fi01host03,21302,1522463314482' (Sat Mar 31 10:28:34 GMT+08:00).

步骤 5 找到所创建的新表“cga_info2”。

APACHE HBASE	
Home Table Details Local Logs	
boy_info	'boy_info', {NAME => 'info', BLOOMFILTER => 'ROW', VERSION => '1', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
cga:student	'cga:student', {NAME => 'info', BLOOMFILTER => 'ROW', VERSION => '1', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
cga_info	'cga_info', {NAME => 'info', BLOOMFILTER => 'ROW', VERSION => '1', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
cga_info2	'cga_info2', {NAME => 'info', BLOOMFILTER => 'ROW', VERSION => '1', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}

步骤 6 查询 region 的切分结果，表“cga_info2”确实被分成了 4 个 region，Name 当中依次包含的是表的名称，StartKey（第一个 region 没有 StartKey），时间戳以及 region 的 ID。

APACHE
HBASE

Home

Table Details

Local Logs

Table Regions

Name	Region Server	Start Key
cga_info2,,1523504763115.574fdf7e2ac6d95040cc87b0e0fe74c6.	fi01host01,21302,1522806887697	
cga_info2,@\x00\x00\x00\x00\x00\x00,1523504763115.767c512a9cae211931d80ca4ee64decc.	fi01host01,21302,1522806887697	@\x00\x00\x00\x00\x00\x00
cga_info2,\x80\x00\x00\x00\x00\x00,1523504763115.b310249ea9b87e8fe0a76277f9ff962.	fi01host02,21302,1522463312798	\x80\x00\x00\x00\x00\x00
cga_info2,\xC0\x00\x00\x00\x00\x00,1523504763115.bc00f56b055aaa14b12afe2a5206f7d6.	fi01host03,21302,1522463314482	\xC0\x00\x00\x00\x00\x00

3.3.3.2 指定 region 的 startKey 和 endKey

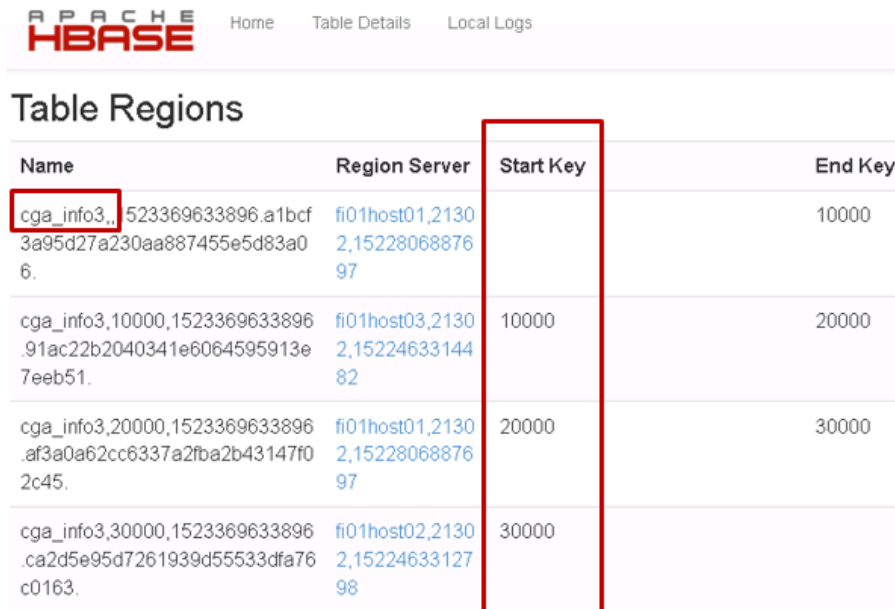
步骤 1 创建表时指定 region 的 StartKey 和 EndKey。

```
create '表的名称', ", SPLITS => ['第一个 StartKey', '第二个 StartKey', '第三个 StartKey']
```

示例：创建表名为'cga_info3'，三个 StartKey 分别为 10000,20000,30000

```
> create 'cga_info3','info',SPLITS => ['10000', '20000', '30000']
0 row(s) in 0.6820 seconds
=> Hbase::Table - cga_info3
```

步骤 2 按照 4.3.3.1 的步骤来到 Table Regions 界面。



Name	Region Server	Start Key	End Key
cga_info3,523369633896.a1bcf3a95d27a230aa887455e5d83a06,	fi01host01,21302,1522806887697		10000
cga_info3,10000,1523369633896.91ac22b2040341e6064595913e7eeb51,	fi01host03,21302,1522463314482	10000	20000
cga_info3,20000,1523369633896.af3a0a62cc6337a2fba2b43147f02c45,	fi01host01,21302,1522806887697	20000	30000
cga_info3,30000,1523369633896.ca2d5e95d7261939d55533dfa76c0163,	fi01host02,21302,1522463312798	30000	

实验结果表明表“cga_info3”确实按照 StartKey 10000,20000,30000 被分为了 4 个 region。

3.3.3.3 通过文件创建预分 region 表

步骤 1 按 Ctrl+C 退出 shell。

```
stu01@fi01host01:~>
```

步骤 2 在/tmp/stu01/目录下创建 splitFile。

```
> touch /tmp/stu01/splitFile.dat
```

步骤 3 进入/tmp/stu01/文件夹。

```
> cd /tmp/stu01
```

步骤 4 在 splitFile.dat 中输入内容[10000,20000,30000]。

```
> vim splitFile.dat
```

进入编辑界面按 i，然后依次输入 10000enter 20000enter 30000enter。

步骤 5 输入完成后按 esc : wq 结束编辑。

步骤 6 再次进入 HBase shell。

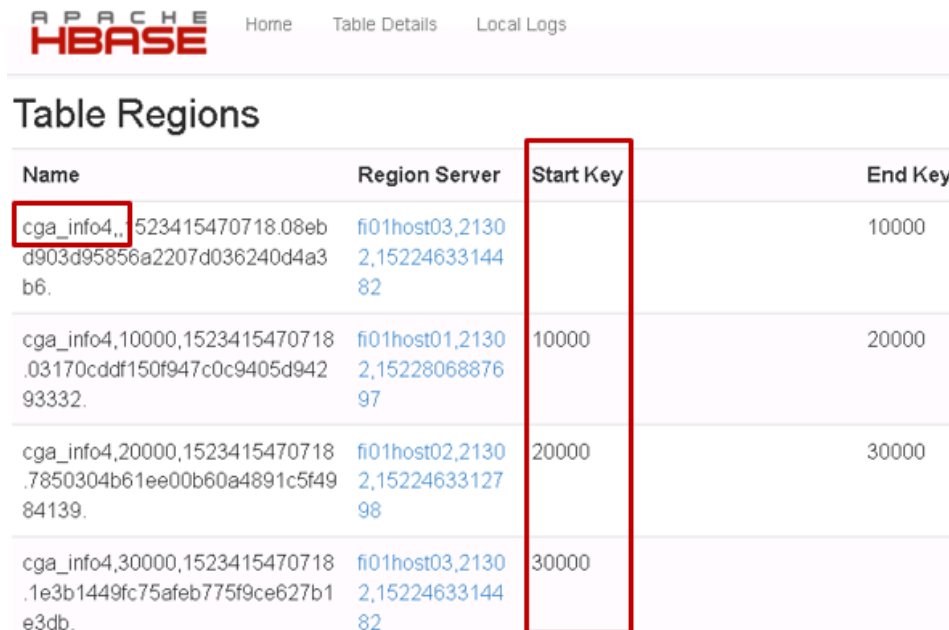
```
> cd /tmp/stu01/hadoopClient
> source bigdata_env
> kinit stu01
```

```
Password for stu01@HADOOP.COM:
> hbase shell
```

步骤 7 创建一个名为“cga_info4”的表，并且用预先创建好的 splitFile 对其进行预分。

```
> create 'cga_info4','info',SPLITS_FILE =>'/tmp/stu01/splitFile'
0 row(s) in 0.4650 seconds
=> Hbase::Table - cga_info4
```

步骤 8 按照 4.3.3.1 的步骤来到 Table Regions 界面。



Name	Region Server	Start Key	End Key
cga_info4,1523415470718.08ebd903d95856a2207d036240d4a3b6.	fi01host03,2130 2,15224633144 82		10000
cga_info4,10000,1523415470718.03170cddf150f947c0c9405d94293332.	fi01host01,2130 2,15228068876 97	10000	20000
cga_info4,20000,1523415470718.7850304b61ee00b60a4891c5f4984139.	fi01host02,2130 2,15224633127 98	20000	30000
cga_info4,30000,1523415470718.1e3b1449fc75afeb775f9ce627b1e3db.	fi01host03,2130 2,15224633144 82	30000	

实验结果表明表“cga_info4”按照 splitFile.dat 中指定的 StartKey“10000,20000,30000”被分为了 4 个 region。

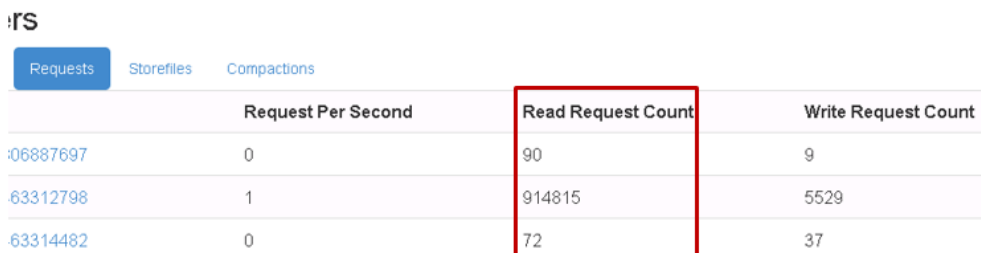
注：指定 start_key 和 end_key 的创建预分 region 表，region 范围是左闭右开。即 region row_key 区间为[start_key,end_key)。

3.3.4 HBase 负载均衡实验

3.3.4.1 查看 HBase Web UI

步骤 1 通过 4.3.3.1 的前 4 个步骤来到 Hbase 的 Region Servers 界面。

步骤 2 点击“Requests”。



	Request Per Second	Read Request Count	Write Request Count
06887697	0	90	9
63312798	1	914815	5529
63314482	0	72	37

步骤 3 点击 “Base Stats”。

Region Servers

Base Stats Memory Requests Storefiles Compactions

ServerName	Start time	Requests Per Second	Num. Regions
fi01host01,21302,1522806887697	Wed Apr 04 09:54:47 GMT+08:00 2018	0	24
fi01host02,21302,1522463312798	Sat Mar 31 10:28:32 GMT+08:00 2018	1	38
fi01host03,21302,1522463314482	Sat Mar 31 10:28:34 GMT+08:00 2018	0	26
Total 3		1	88

从图中我们可以看到，出现了严重的负载均衡问题。fi01host02 主机负担过重，此时我们可以手动将热点 region 移动到 fi01host01 主机。

3.3.4.2 移动 region

步骤 1 点击 “fi01host02”。

HBASE Home Table Details Local Logs

Master fi01host02

Region Servers

Base Stats Memory Requests Storefiles Compactions

ServerName	Start time	Requests Per Second	Num. Regions
fi01host01,21302,1522806887697	Wed Apr 04 09:54:47 GMT+08:00 2018	0	24
fi01host02,21302,1522463312798	Sat Mar 31 10:28:32 GMT+08:00 2018	0	38
fi01host03,21302,1522463314482	Sat Mar 31 10:28:34 GMT+08:00 2018	0	26
Total 3		0	88

步骤 2 查看 fi01host02 负责哪些 region。

HBASE Home Local Logs

Regions

Base Info Request metrics Storefile Metrics Memstore Metrics Compaction Metrics

Region Name	Read Request Count	Write Request Count
socket1_1522486953418fa5cfe24d9aaf6ae6565cca3d495b61	5722	5228
body1_info_152267253463567aee3318a626ec0b1265e26d46c151	9	7
qbo:t1_1522742221778cd371bdfa1722ae748328633417b1827	0	0
qbo:t1,2000,1522742221778.650b17bf5e65c9405868515da6f8cdea	0	0
qbo:t3,1000,1522742867214.4049e7f649e3da1369f73b11ad5f0b7	0	0
qbo:t3,2000,1522742867214.0f97d1815929e9a641094e23b60adbdf	0	0
qbo:t4_1522763066604ca3d4c6f2101984fb36db0f3d40e50	9	3
socket_1518101538544.46680382b10bce88547b796b85b03bf	0	0
hbase:meta_1.1588230740	914885	216
hbase:acl_1500050545984d278ea8ba57e8fc98b1a7d1f2baa04	57	48

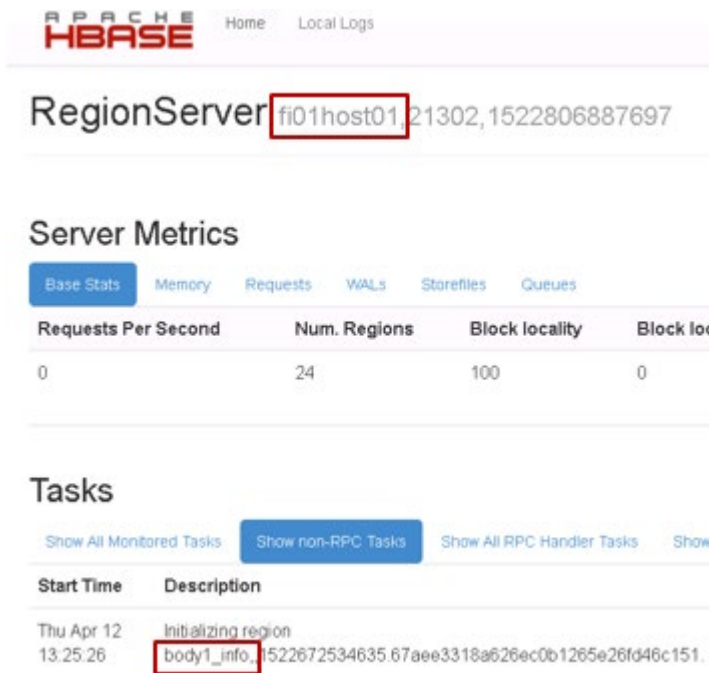
从图中可看到，是 meta 表造成的负载不均衡。但不建议移动 meta 表，本次实验我们移动 body1_info 表。

步骤 3 退出 hbase shell。

步骤 4 将文件'67aee3318a626ec0b1265e26fd46c151'转移到 RegionServer'fio1host01,21302,1522806777697'当中。

```
> echo "move
'67aee3318a626ec0b1265e26fd46c151','fio1host01,21302,1522806777697'" | hbase
shell
move '67aee3318a626ec0b1265e26fd46c151','fio1host01,21302,1522806777697'
0 row(s) in 0.4200 seconds
```

步骤 5 查看 HBase Web UI:



The screenshot shows the Apache HBase Web UI. At the top, there's a navigation bar with 'Home' and 'Local Logs'. Below it, the title 'RegionServer fio1host01,21302,1522806887697' is displayed. The 'Server Metrics' section is active, showing a table with the following data:

Requests Per Second	Num. Regions	Block locality	Block loc
0	24	100	0

Below the metrics, the 'Tasks' section is shown. It has four tabs: 'Show All Monitored Tasks', 'Show non-RPC Tasks' (which is selected), 'Show All RPC Handler Tasks', and 'Show'. The tasks table shows a single task:

Start Time	Description
Thu Apr 12 13:25:26	Initializing region body1_info_1522672534635.67aee3318a626ec0b1265e26fd46c151.

从 Web UI 中看出 该 region 已成功移到 fio1host01。

3.4 实验小结

本实验主要讲述了 HBase 表的创建与删除、数据的增删改查等操作，同时介绍了 HBase 预分 Region 的使用方法，最后还讲述了如何手动实现负载均衡。通过该实验，学员可掌握 HBase 的多种使用方法，加深对 HBase 的理解。

4 Hive 数据仓库实战

4.1 实验背景

Hive 是重要的数据仓库工具，在数据挖掘、数据汇总、统计分析等领域有重要作用。特别的在电信业务中，Hive 扮演相当重要的角色，可以利用 Hive 统计用户的流量、话费、资费等信息，也可挖掘出用户的消费模型以帮助运营商更好的规划套餐内容。

4.2 实验目的

- 掌握 Hive 的常用操作。
- 学会在 Hue 上使用和运行 HQL。

4.3 实验任务

4.3.1 Hive 常用函数

步骤 1 首先执行环境变量。

```
> source /home/user01/hadoopClient/bigdata_env
```

步骤 2 进入 Hive 客户端 beeline。

```
> /home/user01/hadoopClient/Hive/Beeline/bin/beeline
...
Connected to: Apache Hive (version 1.3.0)
Driver: Hive JDBC (version 1.3.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 1.3.0 by Apache Hive
0: jdbc:hive2://192.168.225.11:21066/>
```

步骤 3 字符串函数 length(string A)。

字符串长度函数：length

语法: length(string A)

返回值: int

说明：返回字符串 A 的长度

```
hive> select length('abcdefg');
```

7

步骤 4 字符串反转函数 reverse。

语法: reverse(string A)

返回值: string

说明: 返回字符串 A 的反转结果

```
hive> select reverse('abcdefg');  
gfdecba
```

步骤 5 字符串连接函数 concat。

语法: concat(string A, string B...)

返回值: string

说明: 返回输入字符串连接后的结果, 支持任意个输入字符串

```
hive> select concat('abc','def','gh');  
abcdefgh
```

步骤 6 带分隔符字符串连接函数 concat_ws。

语法: concat_ws(string SEP, string A, string B...)

返回值: string

说明: 返回输入字符串连接后的结果, SEP 表示各个字符串间的分隔符

```
hive> select concat_ws('-', 'abc', 'def', 'gh');  
abc-def-gh
```

步骤 7 字符串截取函数 substr, substring。

语法: substr(string A, int start, int len), substring(string A, int start, int len)

返回值: string

说明: 返回字符串 A 从 start 位置开始, 长度为 len 的字符串

```
hive> select substr('abcde', 3, 2);  
cd  
hive> select substr('abcde', -2, 2);  
de
```

步骤 8 字符串转大写函数 upper, ucase。

语法: upper(string A) ucase(string A)

返回值: string

说明: 返回字符串 A 的大写格式

```
hive> select upper('abC');  
ABC  
hive> select ucase('abC');  
ABC
```

步骤 9 字符串转小写函数 lower,lcase。

语法: lower(string A) lcase(string A)

返回值: string

说明: 返回字符串 A 的小写格式

```
hive> select lower('abC');
abc
hive> select lcase('abC');
abc
```

步骤 10 去空格函数 trim。

语法: trim(string A)

返回值: string

说明: 去除字符串两边的空格

```
hive> select trim(' abc ');
abc
```

步骤 11 分割字符串函数 split。

语法: split(string str, string pat)

返回值: array

说明: 按照 pat 字符串分割 str, 会返回分割后的字符串数组

```
hive> select split('abtcdef','t');
["ab","cd","ef"]
```

步骤 12 日期函数。

获取当前 UNIX 时间戳函数: unix_timestamp

语法: unix_timestamp()

返回值: bigint

说明: 获得当前时区的 UNIX 时间戳。

```
hive> select unix_timestamp() from dual;
1521511607
```

UNIX 时间戳转日期函数: from_unixtime

语法: from_unixtime(bigint unixtime[, string format])

返回值: string

说明: 转化 UNIX 时间戳 (从 1970-01-01 00:00:00 UTC 到指定时间的秒数) 到当前时区的时间格式。

```
hive> select from_unixtime(1521511607,'yyyyMMdd');
20180320
```

4.3.2 Hive 创建表

4.3.2.1 建表语句

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
[(col_name data_type [COMMENT col_comment], ...)]
[COMMENT table_comment]
[PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
[CLUSTERED BY (col_name, col_name, ...)
[SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS]
[ROW FORMAT row_format]
[STORED AS file_format]
[LOCATION hdfs_path]
```

4.3.2.2 创建内部表

创建内部表 cga_info1, 包含 name, gender 和 time 三列。

```
> create table cga_info1(name string,gender string,time int) row format
delimited fields terminated by ',' stored as textfile;
No rows affected (0.293 seconds)
```

其中, row format delimited fields terminated by ',' 表示行分隔符设置为 ',' , 如果不设置则使用默认分隔符。Hive 的 HQL 语句最后是以 ';' 结束的。

查看表 "cga_info1":

```
> show tables like 'cga_info1';
```

```
+-----+
| tab_name |
+-----+
| cga_info1 |
+-----+
```

```
1 row selected (0.07 seconds)
```

4.3.2.3 创建外部表

创建外部表时要注明 external table。

```
> create external table cga_info2 (name string,gender string,time int) row format delimited fields
terminated by ',' stored as textfile;
No rows affected (0.343 seconds)
```

查看创建表 "cga_info2"。

```
> show tables like 'cga_info2';
```

```
+-----+
| tab_name |
+-----+
| cga_info2 |
+-----+
```

```
1 row selected (0.078 seconds)
```

4.3.2.4 载入本地数据

步骤 1 预先在本地建立一个文件。

```
> touch 'cga111.dat'
```

步骤 2 使用 vim 命令编辑文件 cga111.dat，按照 name,gender,time 的先后顺序输入几行数据，行分隔符为','，换行按 enter，输入完毕后按 ESC，:wq，保存并退出到 linux 界面。

```
> vim 'cga111.dat'
```

步骤 3 重新进入 Hive。

```
> beeline
```

步骤 4 建立一个名为“cga_info3”的表。

```
> create table cga_info3(name string,gender string,time int) row format
delimited fields terminated by ',' stored as textfile;
No rows affected (0.408 seconds)
```

步骤 5 将本地数据“cga111.dat”载入表“cga_info3”中。

```
> load data local inpath '/tmp/cga111.dat' into table cga_info3;
INFO : Loading data to table default.cga_info3 from file:/tmp/cga111.dat
No rows affected (0.516 seconds)
```

步骤 6 查询表“cga_info3”中的内容。

```
> select * from cga_info3;
+-----+-----+-----+-----+
| cga_info3.name | cga_info3.gender | cga_info3.time |
+-----+-----+-----+-----+
| xiaozhao       | female           | 20              |
| xiaoqian       | male             | 21              |
| xiaosun        | male             | 25              |
| xiaoli         | female           | 40              |
| xiaozhou       | male             | 33              |
+-----+-----+-----+-----+
5 rows selected (0.287 seconds)
```

由此可见，已经成功将本地文件“cga111.dat”中的内容加载到了 Hive 表“cga_info3”中。

4.3.2.5 载入 HDFS 数据

步骤 1 首先在 HDFS 上创建文件夹“/cga/cg”。

```
> hdfs dfs -mkdir /cga
18/04/12 19:43:54 INFO hdfs.PeerCache: SocketCache disabled.
> hdfs dfs -mkdir /cga/cg
18/04/12 19:44:24 INFO hdfs.PeerCache: SocketCache disabled.
```

步骤 2 将 tmp 文件夹中的本地文件“cga111.dat”上传到 HDFS 的文件夹“/cga/cg”。

```
> hdfs dfs -put cga111.dat /cga/cg
18/04/12 14:19:39 INFO hdfs.PeerCache: SocketCache disabled.
```

步骤 3 重新进入 Hive。

```
> beeline
```

步骤 4 建立一个名为“cga_info4”的表。

```
> create table cga_info4(name string,gender string,time int) row format
delimited fields terminated by ',' stored as textfile;
No rows affected (0.404 seconds)
```

步骤 5 将 HDFS 文件“cga111.dat”载入表“cga_info4”中。

```
> load data inpath '/cga/cg/cga111.dat' into table cga_info4;
INFO : Loading data to table default.cga_info4 from
hdfs://hacluster/app_stu01/cga111.dat
No rows affected (0.341 seconds)
```

注：加载本地数据和 HDFS 数据使用的命令略有不同：

加载本地文件：load data **local** inpath '*local_inpath*' into table *hive_table*;

加载 HDFS 文件：load data inpath '*HDFS_inpath*' into table *hive_table*。

步骤 6 查询表“cga_info4”中的内容。

```
> select * from cga_info4;
+-----+-----+-----+---+
| cga_info4.name | cga_info4.gender | cga_info4.time |
+-----+-----+-----+---+
| xiaozhao      | female          | 20              |
| xiaoqian      | male            | 21              |
| xiaosun       | male            | 25              |
| xiaoli        | female          | 40              |
| xiaozhou      | male            | 33              |
+-----+-----+-----+---+
5 rows selected (0.303 seconds)
```

由此可见，已经将 HDFS 文件“cga111.dat”的内容加载到了 Hive 表“cga_info3”中。

4.3.2.6 创建表时载入数据

创建表“cga_info5”同时载入 HDFS 上 cga111.dat 的数据。

```
> create external table cga_info5 (name string,gender string,time int) row
format delimited fields terminated by ',' stored as textfile location
'/cga/cg';
No rows affected (0.317 seconds)
```

查询表“cga_info5”中的内容。

```
> select * from cga_info5;
+-----+-----+-----+---+
| cga_info5.name | cga_info5.gender | cga_info5.time |
+-----+-----+-----+---+
| xiaozhao      | female          | 20              |
```

```

| xiaoqian      | male      | 21      |
| xiaosun       | male      | 25      |
| xiaoli        | female    | 40      |
| xiaozhou      | male      | 33      |
+-----+-----+-----+
5 rows selected (0.268 seconds)

```

由此可见，我们成功的在创建表“cga_info5”的同时载入了 HDFS 上 cga111.dat 的数据。

4.3.2.7 复制一个空表

步骤 1 建立一个新表“cga_info6”，将表“cga_info1”复制到表“cga_info6”中。

```

> create table cga_info6 like cga_info1;
No rows affected (0.244 seconds)

```

步骤 2 查询表“cga_info6”中的内容。

```

> select *from cga_info6;
+-----+-----+-----+
| cga_info6.name | cga_info6.gender | cga_info6.time |
+-----+-----+-----+
+-----+-----+-----+
No rows selected (0.243 seconds)

```

由结果可知，复制空表成功。

4.3.3 查询

4.3.3.1 模糊查询表

查询以“cga 开头的表”。

```

> show tables like '*cga*';
+-----+
| tab_name |
+-----+
| cga_hive_hbase |
| cga_info1      |
| cga_info2      |
| cga_info3      |
| cga_info4      |
+-----+
5 rows selected (0.072 seconds)

```

4.3.3.2 条件查询

示例 1：使用 limit 查询表“cga_info3”中前两行的数据。

```

> select * from cga_info3 limit 2;
+-----+-----+-----+
| cga_info3.name | cga_info3.gender | cga_info3.time |
+-----+-----+-----+
| xiaozhao       | female           | 20              |
| xiaoqian       | male             | 21              |

```



```
+-----+-----+-----+---+
2 rows selected (0.295 seconds)
```

示例 2：使用 where 查询表“cga_info3”中所有女性的信息。

```
> select * from cga_info3 where gender='female';
+-----+-----+-----+---+
| cga_info3.name | cga_info3.gender | cga_info3.time |
+-----+-----+-----+---+
| xiaozhao       | female           | 20              |
| xiaoli         | female           | 40              |
+-----+-----+-----+---+
2 rows selected (0.286 seconds)
```

示例 3：使用 order 按时间递减顺数查询“cga_info3”中所有女性的信息。

```
> select * from cga_info3 where gender='female' order by time desc ;
+-----+-----+-----+---+
| cga_info3.name | cga_info3.gender | cga_info3.time |
+-----+-----+-----+---+
| xiaoli         | female           | 40              |
| xiaozhao       | female           | 20              |
+-----+-----+-----+---+
2 rows selected (24.129 seconds)
```

由结果可以看出本来 xiaozhao 的信息是先输入的，查询结果按照 time 的递减排序，所以 xiaozhao 的信息排在了第二个。

4.3.3.3 多条件查询

示例 1：对表“cga_info3”进行查询，按姓名进行分组，找出 time 值大于 30 的人。

```
> select name,sum(time) all_time from cga_info3 group by name having
all_time >=30 ;
+-----+-----+---+
| name   | all_time |
+-----+-----+---+
| xiaoli  | 40       |
| xiaozhou | 33       |
+-----+-----+---+
2 rows selected (24.683 seconds)
```

示例 2：对表“cga_info3”进行查询，按性别分组，找出 time 值最大的人。

```
> select gender,max(time) from cga_info3 group by gender;
+-----+-----+---+
| gender | _c1    |
+-----+-----+---+
| female | 40     |
| male   | 33     |
+-----+-----+---+
2 rows selected (24.35 seconds)
```

示例 3：统计表“cga_info3”中，女性和男性的总数各是多少。

```
> select gender,count(1) num from cga_info3 group by gender;
```

```
+-----+-----+---+
| gender | num |
+-----+-----+---+
| female | 2   |
| male   | 3   |
+-----+-----+---+
2 rows selected (23.828 seconds)
```

示例 4：将表“cga_info7”中女性的信息插入表“cga_info3”。

步骤 1 首先建立内部表“cga_info7”。

```
> create table cga_info7(name string,gender string,time int) row format
delimited fields terminated by ',' stored as textfile;
No rows affected (0.282 seconds)
```

步骤 2 创建/tmp 中创建本地文件“cga222.dat”并输入内容。

```
> touch cga222.dat
> vim cga222.dat
```

步骤 3 将本地数据载入表“cga_info7”中。

```
> load data local inpath '/tmp/cga222.dat' into table cga_info7;
INFO : Loading data to table default.cga_info7 from file:/tmp/cga222.dat
No rows affected (0.423 seconds)
```

步骤 4 将表“cga_info7”中女性的信息加载到表“cga_info3”中。

```
> insert into cga_info3 select * from cga_info7 where gender='female';
No rows affected (20.232 seconds)
```

步骤 5 查询表“cga_info3”中的内容。

```
> select * from cga_info3;
+-----+-----+-----+---+
| cga_info3.name | cga_info3.gender | cga_info3.time |
+-----+-----+-----+---+
| xiaozhao       | female           | 20              |
| xiaochen       | female           | 28              |
| xiaozhao       | female           | 20              |
| xiaoqian       | male             | 21              |
| xiaosun        | male             | 25              |
| xiaoli         | female           | 40              |
| xiaozhou       | male             | 33              |
+-----+-----+-----+---+
7 rows selected (0.224 seconds)
```

结果表明表“cga_info3”中增加了两条表“cga_info7”中女性信息。

示例 5：按姓名和性别分组，查询表“cga_info3”中每个人 time 值的总和。

```
> select name,gender,sum(time) time from cga_info3 group by name,gender;
+-----+-----+-----+---+
| name   | gender | time |
+-----+-----+-----+---+
```

```
+-----+-----+-----+---+
| xiaochen | female | 28   |
| xiaoli   | female | 40   |
| xiaoqian | male   | 21   |
| xiaosun  | male   | 25   |
| xiaozhao | female | 40   |
| xiaozhou | male   | 33   |
+-----+-----+-----+---+
```

6 rows selected (23.554 seconds)

从结果中可以看出，在表“cga_info3”中有两条 xiaozhao 的信息，现在已经被合并。

示例 6:首先统计表“cga_info3”中每个人的 time 总和信息，然后按照不同的性别，以 time 值降序排序。

```
> select *,row_number() over(partition by gender order by time desc) rank from
(select name,gender,sum(time) time from cga_info3 group by name,gender) b;
```

```
+-----+-----+-----+-----+---+
| b.name  | b.gender | b.time | rank |
+-----+-----+-----+-----+---+
| xiaozhao | female   | 40     | 1    |
| xiaoli   | female   | 40     | 2    |
| xiaochen | female   | 28     | 3    |
| xiaozhou | male     | 33     | 1    |
| xiaosun  | male     | 25     | 2    |
| xiaoqian | male     | 21     | 3    |
+-----+-----+-----+-----+---+
```

6 rows selected (52.762 seconds)

4.3.4 Hive Join 操作

按照 5.3.2.4 的方法创建两张表“cga_info8”和 “cga_info9”。

查询表“cga_info8”的内容。

```
> select * from cga_info8;
+-----+-----+-----+---+
| cga_info8.name | cga_info8.age |
+-----+-----+-----+---+
| GuoYijun       | 5             |
| YuanJing       | 10            |
| Liyuan         | 20            |
+-----+-----+-----+---+
```

3 rows selected (0.212 seconds)

查询表“cga_info9”的内容：

```
> select * from cga_info9;
+-----+-----+-----+---+
| cga_info9.name | cga_info9.gender |
+-----+-----+-----+---+
| YuanJing       | male              |
| Liyuan         | male              |
+-----+-----+-----+---+
```

```
| LiuYang      | female      |
| Lilei        | male        |
+-----+-----+
4 rows selected (0.227 seconds)
```

4.3.4.1 join/inner join

join 和 innerjoin 操作是内关联，它将两个表的信息相关联，然后只返回两个表共有的结果。

下面的语句使用 join 将表“cga_info8”和 “cga_info9” 中同一个人的信息相关联。

```
> select * from cga_info9 a join cga_info8 b on a.name=b.name;
+-----+-----+-----+-----+
| a.name | a.age | b.name | b.gender |
+-----+-----+-----+-----+
| YuanJing | 10    | YuanJing | male    |
| Liyuan   | 20    | Liyuan   | male    |
+-----+-----+-----+-----+
2 rows selected (24.954 seconds)
```

使用 inner join 将表“cga_info8”和 “cga_info9” 中同一个人的信息相关联。

```
> select * from cga_info9 a inner join cga_info8 b on a.name=b.name;
+-----+-----+-----+-----+
| a.name | a.age | b.name | b.gender |
+-----+-----+-----+-----+
| YuanJing | 10    | YuanJing | male    |
| Liyuan   | 20    | Liyuan   | male    |
+-----+-----+-----+-----+
2 rows selected (25.07 seconds)
```

4.3.4.2 left join

left join: 左外关联，以 left join 关键字前面的表作为主表，和其他表进行关联，返回记录和主表的记录数一致，关联不上的字段置为 NULL。

使用 left join 将表“cga_info8”和 “cga_info9” 中同一个人的信息相关联

```
select * from cga_info9 a left join cga_info8 b on a.name=b.name;
+-----+-----+-----+-----+
| a.name | a.gender | b.name | b.age |
+-----+-----+-----+-----+
| YuanJing | male    | YuanJing | 10    |
| Liyuan   | male    | Liyuan   | 20    |
| LiuYang  | female  | NULL    | NULL  |
| Lilei    | male    | NULL    | NULL  |
+-----+-----+-----+-----+
4 rows selected (24.324 seconds)
```

4.3.4.3 right join

right join: 右外关联，以 right join 关键词后面的表作为主表，和前面的表做关联，返回记录数和主表一致，关联不上的字段为 NULL。

使用 right join 将表“cga_info8”和 “cga_info9” 中同一个人的信息相关联。

```
> select * from cga_info9 a right join cga_info8 b on a.name=b.name;
+-----+-----+-----+-----+
| a.name | a.gender | b.name | b.age |
+-----+-----+-----+-----+
| NULL   | NULL     | GuoYijun | 5      |
| YuanJing | male     | YuanJing | 10     |
| Liyuan  | male     | Liyuan   | 20     |
+-----+-----+-----+-----+
3 rows selected (23.225 seconds)
```

4.3.4.4 full join

full join: 全外关联，是以两个表的记录为基准，返回两个表的记录去重之和，关联不上字段为 NULL。

使用 full join 将表“cga_info8”和 “cga_info9” 中同一个人的信息相关联。

```
> select * from cga_info9 a full join cga_info8 b on a.name=b.name;
+-----+-----+-----+-----+
| a.name | a.gender | b.name | b.age |
+-----+-----+-----+-----+
| NULL   | NULL     | GuoYijun | 5      |
| Lilei   | male     | NULL     | NULL   |
| LiuYang | female   | NULL     | NULL   |
| Liyuan  | male     | Liyuan   | 20     |
| YuanJing | male     | YuanJing | 10     |
+-----+-----+-----+-----+
5 rows selected (26.763 seconds)
```

4.3.4.5 left semi join

left semi join: 以 left semi join 关键字前面的表为主表，返回主表的 KEY 也在副表中的记录。

使用 left semi join 将表“cga_info8”和 “cga_info9” 中同一个人的信息相关联。

```
> select * from cga_info9 a left semi join cga_info8 b on a.name=b.name;
+-----+-----+
| a.name | a.gender |
+-----+-----+
| YuanJing | male     |
| Liyuan   | male     |
+-----+-----+
2 rows selected (24.96 seconds)
```

4.3.4.6 map join

mapjoin 是 Hive 的一种优化操作，其适用于小表 join 大表的场景，由于表的 join 操作是在 Map 端且在内存进行的，所以其并不需要启动 Reduce 任务也就不需要经过 shuffle 阶段，从而在一定程度上节省资源提高 join 效率。

使用 map join 将表“cga_info8”和 “cga_info9” 中同一个人的信息相关联。

```
> select /*+ mapjoin(age)*/ from cga_info9 a join cga_info8 b on
a.name=b.name;
```

```
+-----+-----+-----+-----+
| a.name | a.gender | b.name | b.age |
+-----+-----+-----+-----+
| YuanJing | male | YuanJing | 10 |
| Liyuan | male | Liyuan | 20 |
+-----+-----+-----+-----+
2 rows selected (25.129 seconds)
```

4.3.5 Hive on spark 操作

在 beeline 客户端下，将计算引擎设置成 Spark。

```
> set hive.execution.engine=spark;
No rows affected (0.004 seconds)
```

按姓名和性别分组，查询表“cga_info3”中每个人 time 值的总和。

```
> select name,gender,sum(time) time from cga_info3 group by name,gender;
+-----+-----+-----+
| name | gender | time |
+-----+-----+-----+
| xiaochen | female | 28 |
| xiaoli | female | 40 |
| xiaoqian | male | 21 |
| xiaosun | male | 25 |
| xiaozhao | female | 40 |
| xiaozhou | male | 33 |
+-----+-----+-----+
6 rows selected (1.213 seconds)
```

和 5.3.3 示例 5 的结果相比，Hive onSpark 的查询速度只要 1 秒钟，远远快于 Hive on MapReduce 的查询速率。

4.3.6 Hive 操作 HBase

步骤 1 进入 HBase shell。

```
> hbase shell
```

步骤 2 建立 HBase 表“student”。

```
> create 'student','info'
0 row(s) in 0.4750 seconds
=> Hbase::Table - student
```

步骤 3 向 HBase 表‘student’中输入信息。

```
> put 'student','001','info:name','lilei'
0 row(s) in 0.1310 seconds
> put 'student','002','info:name','tom'
0 row(s) in 0.0210 seconds
```

步骤 4 查看表“student”中的信息。

```
> scan 'student'
```

```
ROW                                COLUMN+CELL
 001                                column=info:name, timestamp=1523544015712, value=lilei
 002                                column=info:name, timestamp=1523544040443, value=tom
2 row(s) in 0.0370 seconds
```

步骤 5 创建 Hive 外部表“cga_hbase_hive”关联该“student”表。

```
> create external table cga_hbase_hive (key int,gid map<string,string>) stored
by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' with SERDEPROPERTIES
("hbase.columns.mapping" ="info:") TBLPROPERTIES ("hbase.table.name"
="student");
No rows affected (0.433 seconds)
```

步骤 6 查询表“cga_hbase_hive”中的内容。

```
> select * from cga_hbase_hive;
+-----+-----+-----+
| cga_hbase_hive.key | cga_hbase_hive.gid |
+-----+-----+-----+
| 1                   | {"name":"lilei"}    |
| 2                   | {"name":"tom"}      |
+-----+-----+-----+
2 rows selected (0.477 seconds)
```

步骤 7 查询表“cga_hbase_hive”中和姓名相关的内容。

```
> select gid['name'] from cga_hbase_hive;
+-----+
| _c0   |
+-----+
| lilei |
| tom   |
+-----+
2 rows selected (0.733 seconds)
```

由实验结果可以看出完成了 Hive 表与 HBase 表的关联操作。

4.3.7 Hive 小文件合并

步骤 1 查看 HDFS 的文件夹/user/hive/warehouse/cga_info1 中的内容。

```
> hdfs dfs -ls -h /user/hive/warehouse/cga_info1
Found 2 items
..... stu01 hive   17 2018-04-13 15:32 /user/hive/warehouse/cga_info1/hive1.log
..... stu01 hive   15 2018-04-13 15:32 /user/hive/warehouse/cga_info1/hive2.log
文件夹/cga/cg 中有 2 个文件。
```

步骤 2 在 Hive 客户端修改参数，将是否合并 Reduce 输出文件设定为 “true”。

```
> set hive.merge.mapredfiles= true;
No rows affected (0.037 seconds)
```

步骤 3 建立新表“cga_info10”将表“cga_info1”的内容载入其中。

```
> create table cga_info10 as select * from cga_info1;
No rows affected (20.93 seconds)
```

步骤 4 查看表 cga_info10 的数据。

```
> hdfs dfs -ls -h /user/hive/warehouse/cga_info10
18/04/13 15:38:23 INFO hdfs.PeerCache: SocketCache disabled.
Found 1 items
-rw-----+ 3 stu01 hive          110 2018-04-13 15:34
/user/hive/warehouse/cga_info10/000000_0
```

结果显示，由于步骤 2 的修改参数设定，本来 Reduce 阶段应该输出的 2 个小文件已经被合并成了一个。

4.3.8 Hive 列加密

Hive 列加密目前常用 AES 算法进行加密。

需在建表时指定，AES 对应加密类名称为：org.apache.hadoop.hive.serde2.AESRewriter。

步骤 1 创建表 info11，并对 name 列进行加密。

```
> create table cga_info11(name string,gender string,time int) row format serde
'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe' with serdeproperties
('column.encode.columns'='name','column.encode.classname'='org.apache.hadoop.hiv
e.serde2.AESRewriter') stored as textfile;
No rows affected (1.097 seconds)
```

步骤 2 将表 cga_info3 的数据载入表 cga_info11 中。

```
> insert into cga_info11 select * from cga_info3;
No rows affected (21.994 seconds)
```

步骤 3 查询表“cga_info11”中的内容。

```
> select * from cga_info11;
+-----+-----+-----+-----+
| cga_info11.name | cga_info11.gender | cga_info11.time |
+-----+-----+-----+-----+
| xiaozhao        | female            | 20               |
| xiaochen        | female            | 28               |
| xiaozhao        | female            | 20               |
| xiaoqian        | male              | 21               |
| xiaosun         | male              | 25               |
| xiaoli          | female            | 40               |
| xiaozhou        | male              | 33               |
+-----+-----+-----+-----+
7 rows selected (0.346 seconds)
```

步骤 4 查看加密效果。

```
> hdfs dfs -cat /user/hive/warehouse/cga_info11/000000_0
```



```
18/04/13 15:21:52 INFO hdfs.PeerCache: SocketCache disabled.  
jR091mQ/LIKY0XBCJi8dsw==female20  
BRaQqw7046X/L1YH1ujKEA==female28  
jR091mQ/LIKY0XBCJi8dsw==female20  
t84/+Zo8Pxiidltw8rAyTA==male21  
J3y40cz4TMGs2uKJfHHaEA==male25  
pz64eOp896fiocKrV0IpoA==female40  
g/sTgzi4MYs9Uotztgg+BQ==male33
```

由结果可以看出表中所有人的姓名信息都被加密了。

4.3.9 使用 Hue 执行 HQL

步骤 1 首先点击服务管理，然后点击“Hue”。



步骤 2 点击“Hue（主）”。



步骤 3 把鼠标移动到 Query Editors 处，然后点击下滑菜单中的 Hive。



步骤 4 在空白处编写 HQL 程序。



步骤 5 编写 HQL 程序结束后，先选择计算引擎，再点击执行。

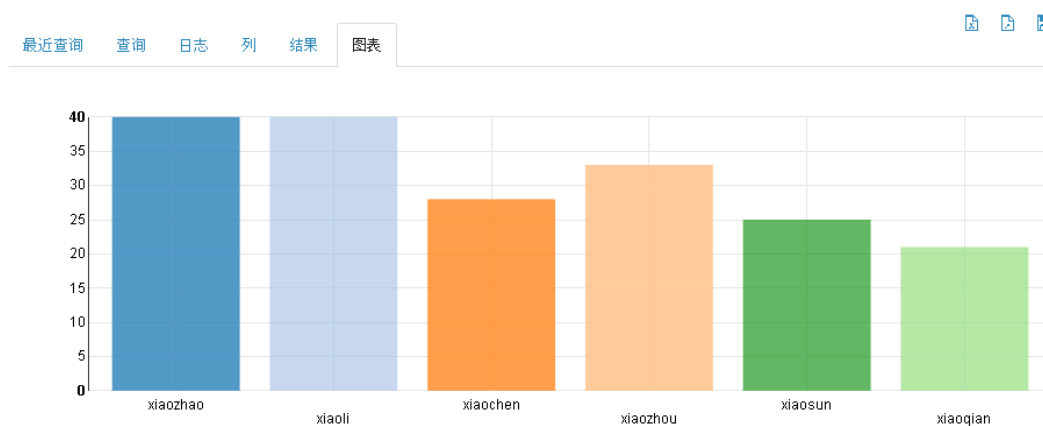


步骤 6 查看结果。

最近查询 查询 日志 列 结果 图表

	b.name	b.gender	b.time	rank
0	xiaozhao	female	40	1
1	xiaoli	female	40	2
2	xiaochen	female	28	3
3	xiaozhou	male	33	1
4	xiaosun	male	25	2
5	xiaoqian	male	21	3

还能以图表形式展示结果：



4.4 实验小结

本实验讲述了 Hive 数据仓库的增删改查操作，还介绍了 Hive On Spark、Hive 操作 HBase 的方法。在 Hive join 操作中，介绍了多种 join 方法，使学员对 join 的类型和区别有更直观的认识。通过本章节实验，学员将增强对 Hive 的理解和使用。需要注意的是 load data 时，必须在创建表时指定 stored as textfile，否则数据将无法导入。

5 Loader 数据导入导出实战

5.1 实验背景

大数据业务中经常涉及数据迁移操作，尤其是关系型数据库与大数据组件间的数据迁移操作，比如，实现 MySQL 与 HDFS/HBase 间的数据迁移。Loader 的图形化操作使得数据迁移更加方便易行。

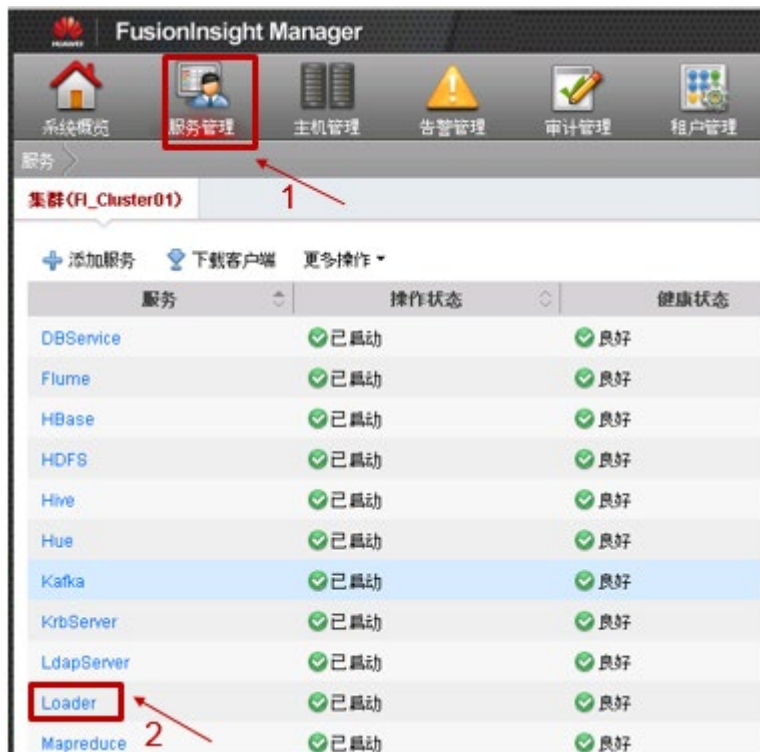
5.2 实验目的

- 掌握 Loader 的使用，能够在业务场景中实现数据迁移操作。

5.3 实验任务

5.3.1 HBase 数据导入 HDFS

步骤 1 首先点击服务管理，然后点击“Loader”。



步骤 2 点击“LoaderServer(主)”。



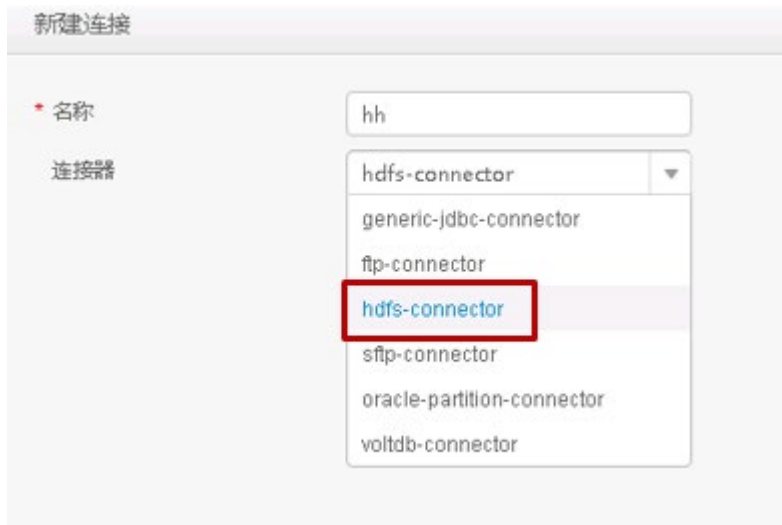
步骤 3 点击新建作业。



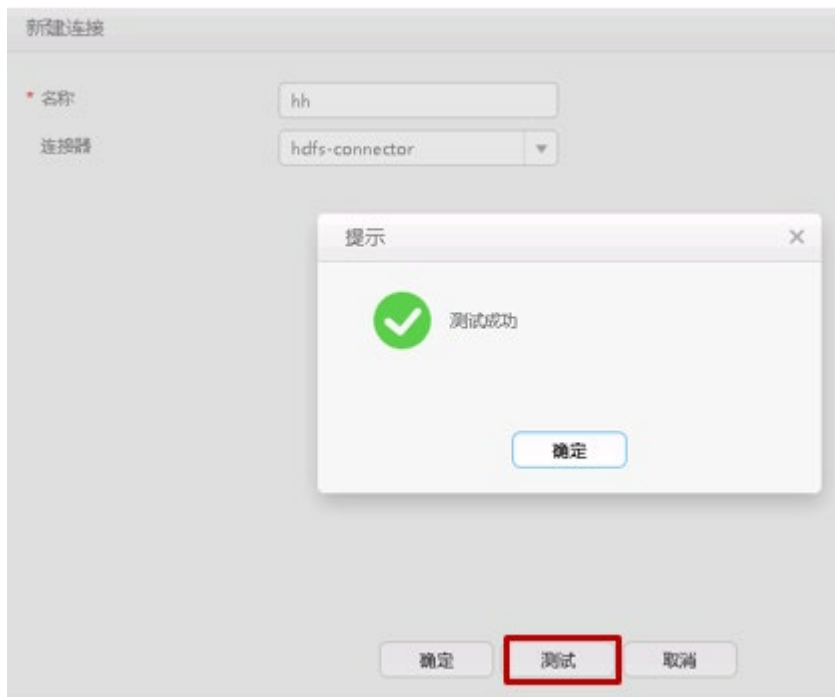
步骤 4 配置 Loader，设置任务名称，选择类型（数据从 HBase 导出到 HDFS，选择导出）。



步骤 5 点击上图中的“添加”。选择 hdfs-connector，名称自定义（不可重复）。



步骤 6 点击测试，显示“测试成功”表明系统可用。



步骤 7 基本信息配置如下，点击“下一步”。

1.基本信息

2.输入设置

* 名称: cg_hbase2hdfs

* 类型: 导出

* 连接: hh

组: 请选择...

* 队列: default

优先级: NORMAL

下一步 取消

步骤 8 配置输入信息，源文件类型选择 HBase，个数为 Map 的任务数，此处填 1 即可，然后点击下一步。

作业 > 新建作业

1.基本信息

2.输入设置

* 源文件类型: HBASE

* HBase实例: HBase

* 个数: 1

返回 下一步 取消

步骤 9 配置转换步骤，点击左侧的输入，然后选择“HBase 输入”，将“HBase 输入”按钮拖动到右侧区域。

1.基本信息

2.输入设置

3.转换

4.输出设置

选择...

输入

HBase输入

输出 +

转换 +

步骤 10 接着点击左侧的“输出”，然后选择“文件输出”，将“文件输出”按钮拖动到右侧区域。



步骤 11 配置输入输出，查询表 `cga_info` 中的内容。

```
> scan 'cga_info'
ROW                                COLUMN+CELL
123002                             column=info:address, timestamp=1523351932415, value=London
123002                             column=info:age, timestamp=1523351887009, value=40
123002                             column=info:gender, timestamp=1523351993106, value=female
123002                             column=info:name, timestamp=1523351965188, value=Victoria
123003                             column=info:address, timestamp=1523352194766, value=Redding
123003                             column=info:age, timestamp=1523352108282, value=30
123003                             column=info:gender, timestamp=1523352060912, value=female
123003                             column=info:name, timestamp=1523352091677, value=Taylor
123004                             column=info:address, timestamp=1523352217267, value=Cleveland
123004                             column=info:age, timestamp=1523352229436, value=33
123004                             column=info:gender, timestamp=1523352267416, value=male
123004                             column=info:name, timestamp=1523352251926, value=LeBron
3 row(s) in 0.0560 seconds
```

步骤 12 配置 HBase 输入，双击 Web UI 中 HBase 输入的按钮。首先输入表的名称“`cga_info`”，然后点击添加，依次输入列族名，列名，字段名，类型，第三步要勾选“主键”，最后点击确定。

HBase Input-HBase输入

HBase表类型: normal

HBase表名: cga_info

HBase输入字段

导入 导出

列族名	列名	字段名	类型	长度	主键
info	rowkey	rowkey	VARCHAR		<input checked="" type="checkbox"/>
info	name	name	VARCHAR		<input type="checkbox"/>
info	gender	gender	VARCHAR		<input type="checkbox"/>
info	age	age	VARCHAR		<input type="checkbox"/>
info	address	address	VARCHAR		<input type="checkbox"/>

添加

确定 取消

步骤 13 双击 Web UI 中文件输出按钮，配置 HDFS 输出。首先指定输出分隔符“.”,第二步点击关联，第三步在位置信息上依次输入编号，最后点击确定。

File Output-文件输出

输出分隔符: .

换行符:

输出字段

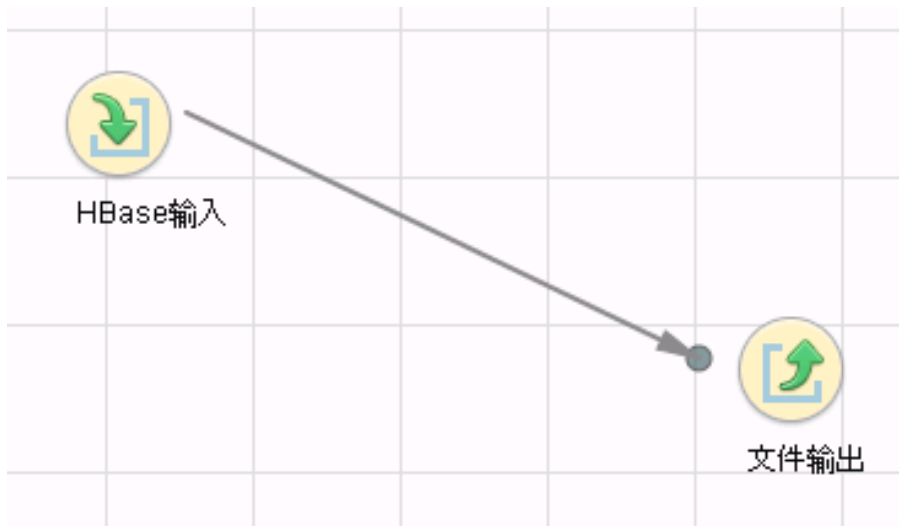
关联 导入 导出

位置	字段名	类型
1	rowkey	VARCHAR
2	name	VARCHAR
3	gender	VARCHAR
4	age	VARCHAR
5	address	VARCHAR

添加

确定 取消

步骤 14 连接 “HBase 输入” 和 “文件输出”。



步骤 15 点击下一步进入输出设置。



步骤 16 填写输出路径，选择文件格式，最后点击“保存并运行”。



步骤 17 查看运行结果。



查看 HDFS 输出结果：

```
> hdfs dfs -cat /tmp/cg_hbasetohdfs/export_part_1522461215526_0104_0000000  
123002.Victoria.female.40.London  
123003.Taylor.female.30.Redding  
123004.LeBron.male.33.Cleveland
```

由程序结果可以看出表“cga_info”的内容已经成功转移到/tmp/cg_hbasetohdfs 目录下的 export_part_1522461215526_0104_0000000 文件上。

5.3.2 HDFS 数据导入 HBase

步骤 1 创建名为'cg_hdfstohbase'的表。

```
> create 'cg_hdfstohbase','info'  
0 row(s) in 0.4350 seconds  
=> Hbase::Table - cg_hdfstohbase
```

步骤 2 按照 6.3.1 的前三个步骤来到配置 Loader 的界面，设置基本信息。

作业 > 新建作业

1.基本信息

2.输入设置

3.转换

* 名称	cg_hdfstohbase		
* 类型	导入		
* 连接	hh	+ 添加	编辑 删除
组	请选择...	+ 添加	
* 队列	default		
优先级	NORMAL		

下一步

取消

步骤 3 配置输入设置。

“输入路径”为：/tmp/cg_hbasetohdfs/export_part_1522461215526_0104_0000000， “文件过滤器”为‘*’， “编码类型”选择“UFT_8”，设置完毕后点击下一步。

作业 > 新建作业



1. 基本信息 2. 输入设置

* 输入路径 /tmp/cg_hbase2ohdfs/export_p.

路径过滤器

* 文件过滤器 +

* 编码类型 UTF_8

后缀名

返回 下一步 取消

步骤 4 配置转换步骤，点击左侧的“输入”，然后选择“CSV 文件输入”，将 CSV 文件输入按钮拖动到右侧区域。



作业 > 新建作业

1. 基本信息 2. 输入设置 3. 转换

过滤...

输入

- CSV 文件输入
- HTML 输入
- 固定宽度文件输入

输出 +

转换 +

CSV 文件输入

步骤 5 接着点击左侧的“输出”，然后选择“文件输出”，将文件输出按钮拖动到右侧区域。



步骤 6 配置 CSV 文件输入，双击 Web UI 中 CSV 文件输入的按钮。首先输入表的分隔符“.”,然后点击添加，依次输入位置编号，字段名，类型，最后点击确定。

位置	字段名	类型
1	rowkey	VARCHAR
2	name	VARCHAR
3	gender	VARCHAR
4	age	VARCHAR
5	address	VARCHAR

步骤 7 配置 HBase 输出。双击 Web UI 中 HBase 输出的按钮，点击关联。

HBase表类型

NULL值处理方式 ☐

HBase输出字段

关联

字段名	表名	列族名	列名
<input type="button" value="添加"/>			

步骤 8 勾选名称旁边的方框，然后点击确定。

<input checked="" type="checkbox"/>	名称	类型
<input checked="" type="checkbox"/>	rowkey	VARCHAR
<input checked="" type="checkbox"/>	name	VARCHAR
<input checked="" type="checkbox"/>	gender	VARCHAR
<input checked="" type="checkbox"/>	age	VARCHAR
<input checked="" type="checkbox"/>	address	VARCHAR

总条数: 5 < 1/1 >

步骤 9 首先输入表名，然后勾选“rowkey”为主键，最后点击确定。

HBase Output-HBase输出

HBase表类型: normal

NULL值处理方式: ☐

HBase输出字段

关联 导入 导出

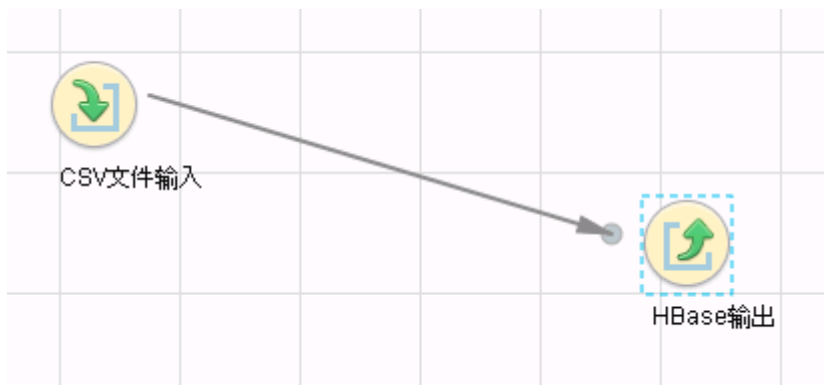
字段名	表名	列族名	列名	类型	长度	主键
rowkey	cg_hdfstohbas	info	rowkey	VARCHAR		<input checked="" type="checkbox"/>
name	cg_hdfstohbas	info	name	VARCHAR		<input type="checkbox"/>
gender	cg_hdfstohbas	info	gender	VARCHAR		<input type="checkbox"/>
age	cg_hdfstohbas	info	age	VARCHAR		<input type="checkbox"/>
address	cg_hdfstohbas	info	address	VARCHAR		<input type="checkbox"/>

添加

确定 取消

1 2 3

步骤 10 连接“CSV 文件输入”和“HBase 输出”。



步骤 11 点击下一步进入输出设置，“存储类型”设置为 HBASE_PUTLIST，“个数”设置为 1，最后点击“保存并运行”。

1. 基本信息 2. 输入设置

* 存储类型 HBASE_PUTLIST

* HBase实例 HBase

☒ Map数 ☐ Map数据块大小

* 个数 1

返回 保存 保存并运行 取消

步骤 12 查看结果。

作业ID	名称	描述	开始时间	执行者	进度	状态
180	cg_hdfstohbase	从 HDFS 导入 到 H...	2018-04-13 05:27...	stu01	100%	成功

步骤 13 在 HBase 中查询表'cg_hdfstohbase'中的内容。

```
> scan 'cg_hdfstohbase'
ROW                                COLUMN+CELL
123002    column=info:address, timestamp=1523623659052, value=London
123002    column=info:age, timestamp=1523623659052, value=40
123002    column=info:gender, timestamp=1523623659052, value=female
123002    column=info:name, timestamp=1523623659052, value=Victoria
123003    column=info:address, timestamp=1523623659052, value=Redding
123003    column=info:age, timestamp=1523623659052, value=30
123003    column=info:gender, timestamp=1523623659052, value=female
123003    column=info:name, timestamp=1523623659052, value=Taylor
123004    column=info:address, timestamp=1523623659052, value=Cleveland
123004    column=info:age, timestamp=1523623659052, value=33
123004    column=info:gender, timestamp=1523623659052, value=male
123004    column=info:name, timestamp=1523623659052, value=LeBron
3 row(s) in 0.0480 seconds
```

由结果可以看到，已经将 HDFS 文件

/tmp/cg_hbasetohdfs/export_part_1522461215526_0104_0000000 的内容加载到表 'cg_hdfstohbase'中。

5.3.3 HDFS 数据导入 MySql

步骤 1 准备 HDFS 测试文件，首先在/tmp 文件夹下创建一个名为 test_mysql 的文件，并且输入内容。

```
> touch test_mysql
> vim test_mysql
```


步骤 2 将本地文件 test_mysql 上传文件到 hdfs 的/loader_test 目录下。

```
> hdfs dfs -put test_mysql /loader_test
> hdfs dfs -ls /loader_test
Found 1 items
-rw-r--r--+ 3 stu01 supergroup      47 2018-04-15 13:09/loader_test/test_mysql
```

步骤 3 查看表 test_mysql 中的内容。

```
> hdfs dfs -cat /loader_test/test_mysql
1,tom,male,8
2,lily,female,24
3,lucy,female,50
```

步骤 4 在 Linux 节点下进入 MySQL。

```
> mysql
```

步骤 5 创建数据库 test_database。

```
mysql> create database test_database;
Query OK, 1 row affected (0.00 sec)
mysql> set names gbk;
Query OK, 0 rows affected (0.00 sec)
mysql> use test_database;
Database changed
```

步骤 6 创建 cga_mysql 表。

```
mysql> create table cga_mysql(id int(4) not null primary key auto_increment,
-> name varchar(255) not null,
-> gender varchar(255) not null,
-> time int(4));
```

注：创建 mysql 表必须有主键。

步骤 7 查询表 cga_mysql 中的内容。

```
mysql> desc cga_mysql;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id    | int(4)        | NO   | PRI | NULL    | auto_increment |
| name  | varchar(255)  | NO   |     | NULL    |                |
| gender | varchar(255)  | NO   |     | NULL    |                |
| time  | int(4)        | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

步骤 8 复制 mysql 链接 jar 包到主备 Loader 指定目录下。

```
>cp /opt/huawei/Bigdata/FusionInsight_V100R002C60SPC200/FusionInsight-Hive-
1.3.0/hive-1.3.0/lib/mysql-connector-java-5.1.21.jar
```

```
/opt/huawei/Bigdata/FusionInsight_V100R002C60SPC200/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib/
```

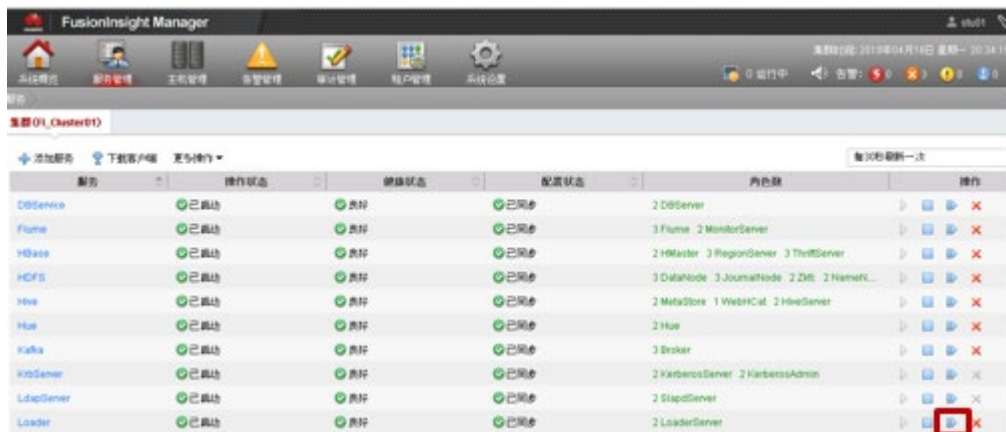
步骤 9 在主备节点上分别查看

/opt/huawei/Bigdata/FusionInsight_V100R002C60SPC200/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib/中的内容。

```
>ll /opt/huawei/Bigdata/FusionInsight_V100R002C60SPC200/FusionInsight-Sqoop-1.99.3/FusionInsight-Sqoop-1.99.3/server/webapps/loader/WEB-INF/ext-lib>
total 940
-rwxr-xr-x 1 root root 118057 Jan 23 11:36 hive-jdbc-1.3.0.jar
-rwxr-xr-x 1 omm wheel 827942 Feb 8 10:36 mysql-connector-java-5.1.21.jar
-rwxr-xr-x 1 omm wheel 18 Nov 23 2015 readme.properties
```

已经将 mysql 链接 jar 包复制到主备 loader 指定目录下。

步骤 10 重启 Loader。



注：若主备 Loader 该目录下已有可用的 mysql-connector-java-x.x.x.jar，此时无需重启 Loader。Loader 主备节点都需要复制该 jar 文件。

步骤 11 由 6.3.1 的步骤 1 到 3，来到配置 Loader 基本信息的界面。



步骤 12 点击编辑进入 mysql 连接配置，mysql 的密码为 123456，填完信息后点击测试，测试完成点击确定。

JDBC 连接字符串输入：jdbc:mysql://192.168.224.41:3306/test_database。



名称: mysql

JDBC驱动程序类: com.mysql.jdbc.Driver

JDBC连接字符串: 2.168.224.41:3306/test_databases

用户名: root

密码: *****

名称: 值

添加

确定 测试

步骤 13 配置“输入设置”。

输入目录设置为: /loader_test/test_mysql。



1. 基本信息 2. 输入设置

源文件类型: HDFS

输入目录: /loader_test/test_mysql

路径过滤器: +

文件过滤器: +

文件类型: TEXT_FILE

文件分割方式: FILE

Map数 ☒ Map数据块大小 ☐

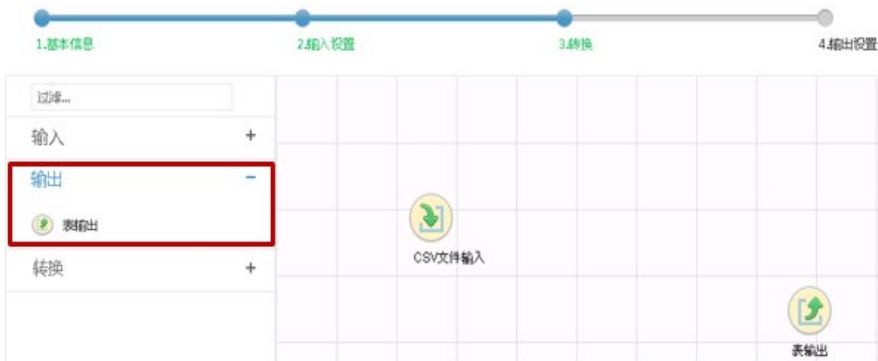
个数: 1

返回 下一步 取消

步骤 14 配置转换设置，点击左侧的输入，然后选择“CSV 文件输入”，将“CSV 文件输入”按钮拖动到右侧区域。



步骤 15 接着点击左侧的输出，然后选择“表输出”，将“表输出”按钮拖动到右侧区域。



步骤 16 配置 CSV 文件输入，双击 Web UI 中“CSV 文件输入”的按钮。首先输入分隔符“,”，然后点击添加，依次输入位置，字段名，最后点击确定。

分隔符

,

换行符

文件名是否作为字段

绝对路径

☐

验证输入字段

YES

输入字段

导入

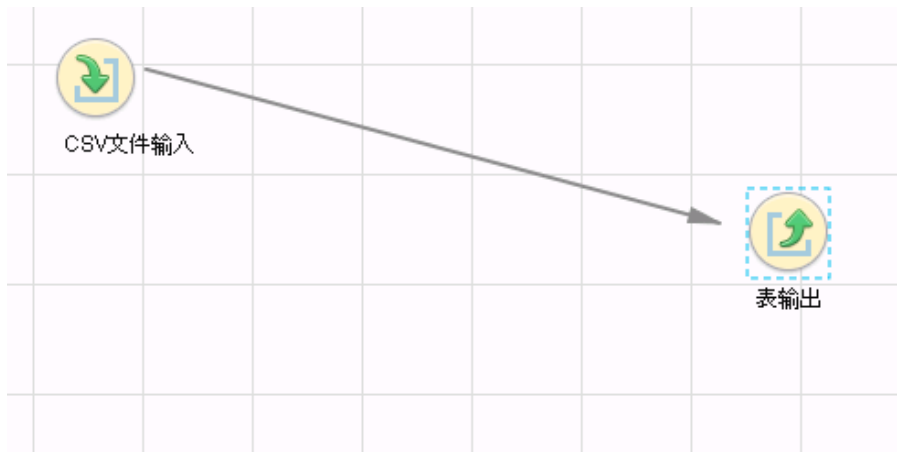
导出

位置	字段名	类型	数据格式
1	name	VARCHAR	
2	gender	VARCHAR	
3	time	INTEGER	

添加

确定

取消



步骤 20 点击下一步进入输出设置，输入表名，最后点击保存并运行。

1. 基本信息 2. 输入设置

架构名称:

* 表名:

Stage 表名:

步骤 21 运行 Loader 作业，查看结果。

作业ID	名称	描述	开始时间	执行者	进度	状态
181	cg_hdfs2mysql	从 HDFS 导出到 R...	2018-04-14 22:33...	stu01	100%	成功

步骤 22 查看 mysql 中表 cga_mysql 的内容。

```
mysql> use test_database;
Database changed
mysql> select * from cga_mysql;
+----+-----+-----+-----+
| id | name | gender | time |
+----+-----+-----+-----+
| 1  | tom  | male   | 8    |
| 2  | lily | female | 24   |
| 3  | lucy | female | 50   |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

由程序结果可见，HDFS 中文件“test_mysql”的内容已经载入到 mysql 的表“cga_mysql”中。

5.3.4 Mysql 数据导入 HDFS

步骤 1 准备 mysql 表，使用 6.3.3 中“cga_mysql”表数据。

```
mysql> select * from cga_mysql;
+----+-----+-----+-----+
| id | name | gender | time |
+----+-----+-----+-----+
| 1  | tom  | male   | 8    |
| 2  | lily | female | 24    |
| 3  | lucy | female | 50    |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

步骤 2 由 6.3.1 的步骤 1 到 3，来到配置 Loader 基本信息的界面。



步骤 3 点击下一步进入输入设置,输入表名。



注：非 default 数据库表，表名当指定为:数据库.表名。

步骤 4 点击下一步进入转换设置，点击左侧的输入，然后选择“表输入”，将“表输入”按钮拖动到右侧区域。



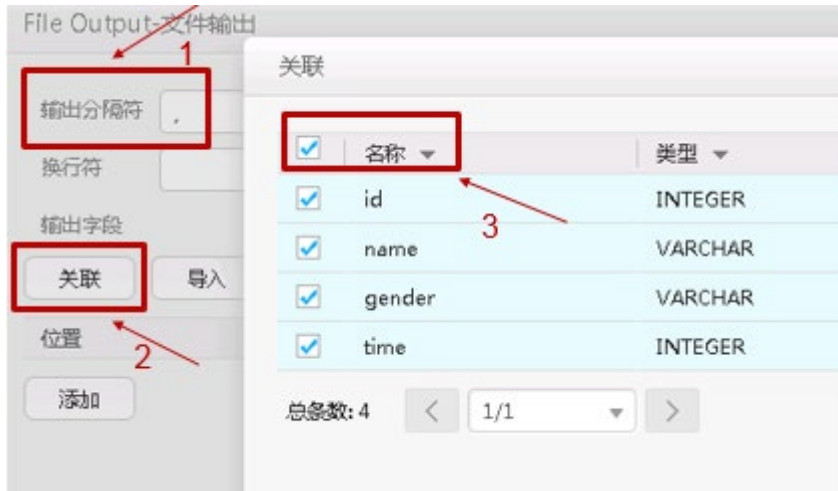
步骤 5 点击左侧的输出，然后选择“文件输出”，将“文件输出”按钮拖动到右侧区域。



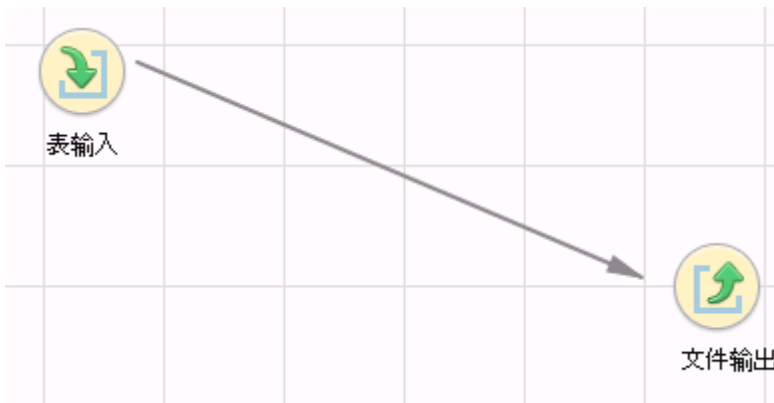
步骤 6 配置表输入，双击 Web UI 中表输入的按钮。首先点击添加，然后依次输入位置，字段名和类型，后点击确定。



步骤 7 配置文件输出，双击 Web UI 中文件输出的按钮。首先设定输出分隔符为“,”，然后点击关联，接着勾选名称左侧的方框，最后点击确定。



步骤 8 连接“表输入”和“文件输出”，点击下一步。



步骤 9 配置输出设置,输出目录填写/loader_test。



步骤 10 运行 Loader 作业，查看结果。



步骤 11 查看 HDFS 中的结果。

```
> hdfs dfs -ls /loader_test
Found 3 items
..... 2018-04-15 14:41 /loader_test/_SUCCESS
..... 2018-04-15 14:41 /loader_test/import_part_1522461215526_0114_0000000
..... 2018-04-15 13:09 /loader_test/test_mysql
> hdfs dfs -cat /loader_test/import_part_1522461215526_0114_0000000
1,tom,male,8
2,lily,female,24
3,lucy,female,50
```

由程序结果可见，已经将 mysql 表“cga_mysql”导入了 HDFS 的/loader_test 目录下。

5.3.5 MySqI 数据导入 HBase

步骤 1 准备 mysql 表，使用 6.3.3 中“cga_mysql”表数据。

```
mysql> select * from cga_mysql;
+----+-----+-----+-----+
| id | name | gender | time |
+----+-----+-----+-----+
| 1 | tom | male | 8 |
| 2 | lily | female | 24 |
| 3 | lucy | female | 50 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

步骤 2 创建 HBase 表：cg_mysqltohbase。

```
> create 'cg_mysqltohbase','info'
0 row(s) in 0.5300 seconds
```

步骤 3 由 6.3.1 的步骤 1 到 3，来到配置 Loader 基本信息的界面。

1. 基本信息 2. 输入设置

* 名称 cg_mysqltohbase

* 类型 导入

* 连接 mysql + 添加

组 请选择... + 添加

* 队列 default

优先级 NORMAL

下一步 取消

步骤 4 点击下一步进入输入设置,输入表名。

1. 基本信息 2. 输入设置

☒ 表方式 ☐ SQL方式

架构名称

* 表名 cga_mysql 处理远程数据库的数据的表名。

表列名

分区列名

分区列空值 ☐ true ☐ false

返回 下一步 取消

步骤 5 点击下一步进入转换设置, 点击左侧的输入, 然后选择“表输入”, 将“表输入”按钮拖动到右侧区域。

1. 基本信息 2. 输入设置 3. 转换

过滤...

输入 -

输出 +

转换 +

表输入

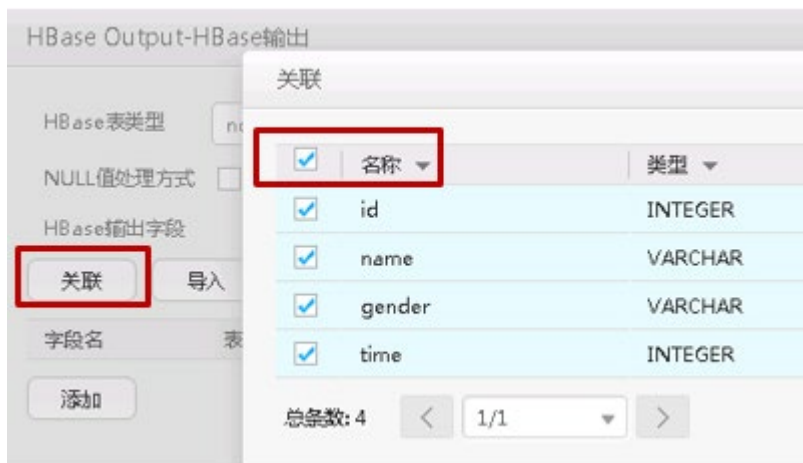
步骤 6 点击左侧的输出, 然后选择“HBase 输出”, 将“HBase 输出”按钮拖动到右侧区域。



步骤 7 配置表输入，双击 Web UI 中表输入的按钮。首先点击添加,然后依次输入位置，字段名和类型，最后点击确定。



步骤 8 配置 HBase 输出，双击 Web UI 中 HBase 输出的按钮。首先点击关联，接着勾选名称左侧的方框，最后点击确定。



步骤 9 依次输入 HBase 表名，列族名，列名和类型，选择 id 为主键，最后点击确定。

HBase Output-HBase输出

HBase表类型: normal

NULL值处理方式: ☐

HBase输出字段:

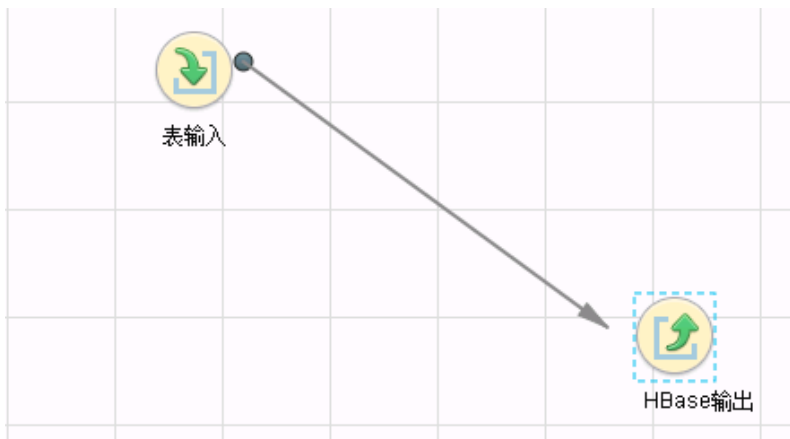
关联 导入 导出

字段名	表名	列族名	列名	类型	长度	主键
id	cg_mysqltohba	info	id	INTEGER		<input checked="" type="checkbox"/>
name	cg_mysqltohba	info	name	VARCHAR		<input type="checkbox"/>
gender	cg_mysqltohba	info	gender	VARCHAR		<input type="checkbox"/>
time	cg_mysqltohba	info	time	INTEGER		<input type="checkbox"/>

添加

确定 取消

步骤 10 连接“表输入”与“HBase 输出”，然后点击下一步。



步骤 11 配置输出信息，存储类型选择 HBASE_PUTLIST, HBase 实例选择 HBase，个数填写 1，最后点击保存并运行。

步骤 12 运行 Loader 作业，查看结果。

作业ID	名称	描述	开始时间	执行者	进度	状态
183	cg_mysqltohbase	从 RDB 导入到 H...	2018-04-15 00:17:...	stu01	100%	成功

步骤 13 查看 HBase 表数据。

```
> scan 'cg_mysqltohbase'
ROW          COLUMN+CELL
2018-04-15 15:21:33,777 INFO [hconnection-0xaa61e4e-shared--pool4-t1]
ipc.AbstractRpcClient: RPC Server Kerberos principal name for
service=ClientService is hbase/hadoop.hadoop.com@HADOOP.COM
1          column=info:gender, timestamp=1523776665511, value=male
1          column=info:name, timestamp=1523776665511, value=tom
1          column=info:time, timestamp=1523776665511, value=8
2          column=info:gender, timestamp=1523776665511, value=female
2          column=info:name, timestamp=1523776665511, value=lily
2          column=info:time, timestamp=1523776665511, value=24
3          column=info:gender, timestamp=1523776665511, value=female
3          column=info:name, timestamp=1523776665511, value=lucy
3          column=info:time, timestamp=1523776665511, value=50
3 row(s) in 0.0700 seconds
```

MySQL 表 cga_mysql 已经成功载入 HBase 表 cg_mysqltohabse。

5.3.6 HBase 数据导入 MySQL

步骤 1 使用 6.3.5 中 HBase 数据表 cg_mysqltohbase。同时在 mysql 数据库创建表 cga_hbasetomysql。

```
mysql> create table cga_hbasetomysql(id int(4) not null primary key
auto_increment, name varchar(255) not null, gender varchar(255) not null, time
int(4));
Query OK, 0 rows affected (0.09 sec)
```

步骤 2 由 6.3.1 的步骤 1 到 3，来到配置 Loader 基本信息的界面。



1. 基本信息

2. 输入设置

* 名称 cg_hbasetomysql

* 类型 导出

* 连接 mysql

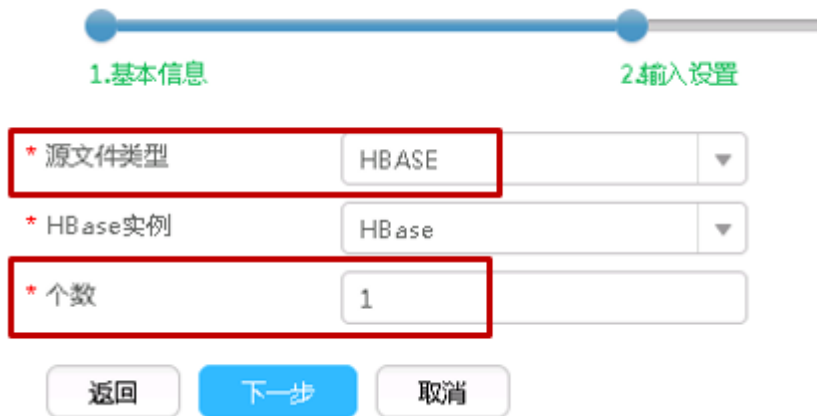
组 请选择...

* 队列 default

优先级 NORMAL

下一步 取消

步骤 3 点击下一步进入输入设置，源文件类型选择 HBASE，个数输入 1。



1. 基本信息

2. 输入设置

* 源文件类型 HBASE

* HBase实例 HBase

* 个数 1

返回 下一步 取消

步骤 4 点击下一步进入转换设置，点击左侧的输入，然后选择“HBase 输入”，将“HBase 输入”按钮拖动到右侧区域。



1. 基本信息

2. 输入设置

3. 转换

过滤...

输入 -

HBase输入

输出 +

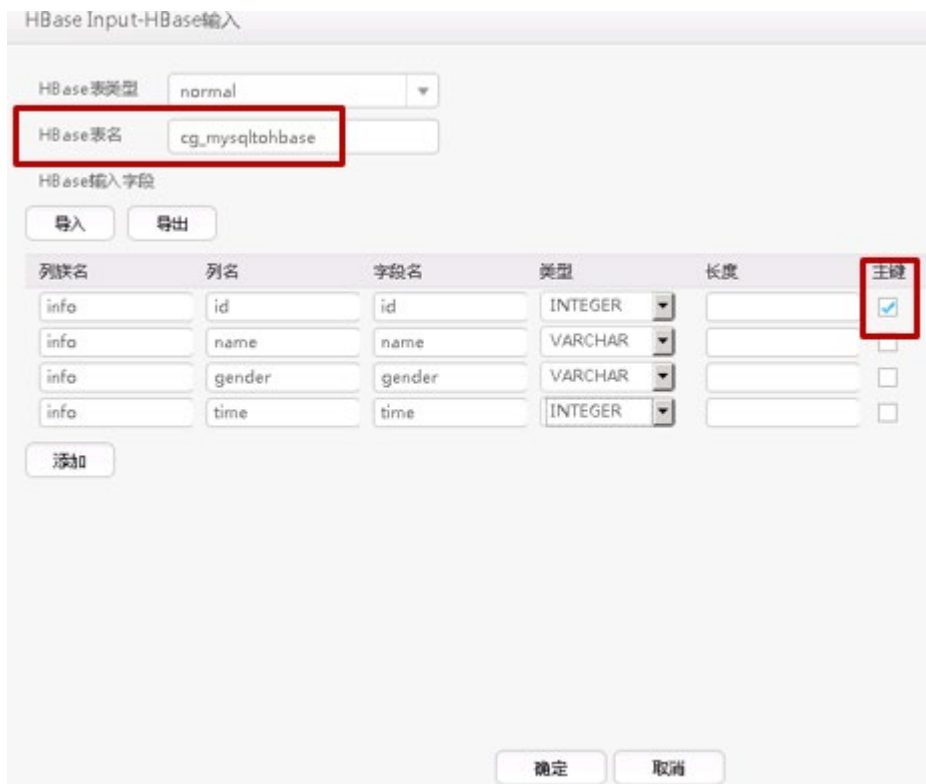
转换 +

HBase输入

步骤 5 点击左侧的输出，然后选择“表输出”，将“表输出”按钮拖动到右侧区域。



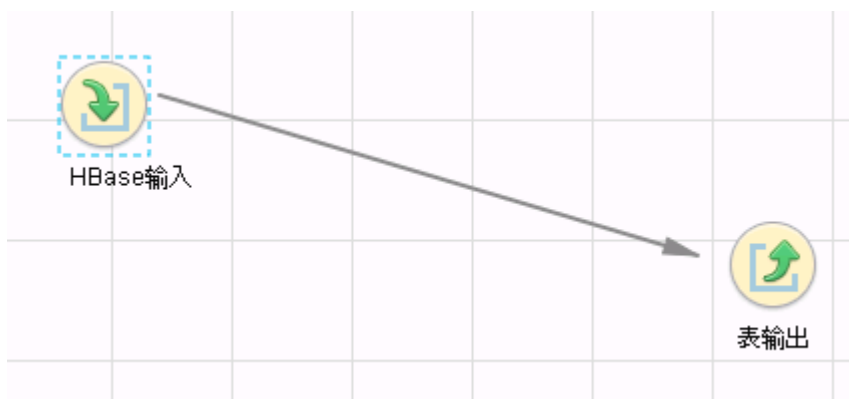
步骤 6 配置 HBase 输入，双击 Web UI 中 HBase 输入的按钮。首先输入 HBase 表名，然后点击添加，依次输入列祖名，列名，字段名和类型，接着选择 id 为主键，最后点击确定。



步骤 7 配置表输出，双击 Web UI 中表输出的按钮。首先点击关联，接着勾选名称左侧的方框，最后点击确定。



步骤 8 连接“HBase 输入”和“表输出”，点击下一步。



步骤 9 配置输出设置，填写表名 cga_hbasetomysql，点击保存并运行。



步骤 10 运行 Loader 作业，查看结果。



步骤 11 查看 mysql 表 cga_hbasetomysql 中的内容。

```
mysql> select * from cga_hbasetomysql;
+----+-----+-----+-----+
| id | name | gender | time |
+----+-----+-----+-----+
```

```
| 1 | tom | male | 8 |  
| 2 | lily | female | 24 |  
| 3 | lucy | female | 50 |  
+---+-----+-----+-----+  
3 rows in set (0.00 sec)
```

由此可见，HBase 表 cg_mysqltohive 的内容已经成功加载到 mysql 表 cga_hbasetomysql 中。

5.3.7 MySQL 数据导入 Hive

步骤 1 使用 6.3.3 中 mysql 表 cga_mysql 的数据，同时新建 Hive 表 cg_mysqltohive。

在 HDFS 上创建目录/user/hive/warehouse/cg_mysqltohive。

```
> hdfs dfs -mkdir /user/hive/warehouse/cg_mysqltohive
```

在 Hive 中创建新表 cg_mysqltohive。

```
> create table cg_mysqltohive(id int,name string,gender string,time int) row  
format delimited fields terminated by ',' stored as textfile location  
'/user/hive/warehouse/cg_mysqltohive';  
No rows affected (0.372 seconds)
```

步骤 2 由 6.3.1 的步骤 1 到 3，来到配置 Loader 基本信息的界面。

The screenshot shows a configuration window for a data loader. At the top, there are two tabs: '1. 基本信息' (Basic Information) and '2. 输入设置' (Input Settings). The 'Basic Information' tab is active. Below the tabs, there are several configuration fields:

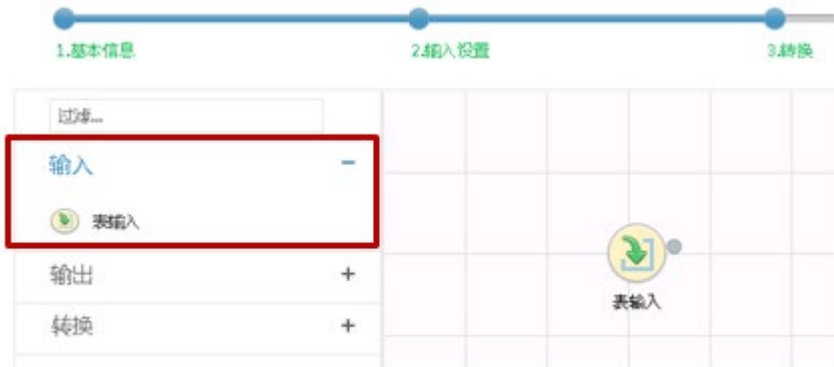
- * 名称 (Name): cg_mysqltohive
- * 类型 (Type): 导入 (Import) - This field is highlighted with a red rectangular box.
- * 连接 (Connection): mysql
- 组 (Group): 请选择... (Please select...)
- * 队列 (Queue): default
- 优先级 (Priority): NORMAL

At the bottom of the configuration area, there are two buttons: '下一步' (Next Step) and '取消' (Cancel).

步骤 3 点击下一步进入输入设置，输入表名 cga_mysql。



步骤 4 点击下一步进入转换设置，点击左侧的输入，然后选择“表输入”，将“表输入”按钮拖动到右侧区域。



步骤 5 点击左侧的输出，然后选择“Hive 输出”，将“Hive 输出”按钮拖动到右侧区域。



步骤 6 配置表输入，双击 Web UI 中表输入的按钮。首先点击添加，依次输入位置和字段名和类型，最后点击确定。

Table Input-表输入

输入字段

导入 导出

位置	字段名	类型
1	id	INTEGER
2	name	VARCHAR
3	gender	VARCHAR
4	time	INTEGER

添加

步骤 7 配置表输出，双击 Web UI 中表输出的按钮。首先点击关联，接着勾选名称左侧的方框，最后点击确定。

Hive Output-Hive输出

Hive文件存储格式

Hive文件压缩格式

输出分隔符

输出字段

关联 导入

位置

添加

关联

<input checked="" type="checkbox"/>	名称
<input checked="" type="checkbox"/>	id
<input checked="" type="checkbox"/>	name
<input checked="" type="checkbox"/>	gender
<input checked="" type="checkbox"/>	time

总条数: 4 < 1/1

步骤 8 补全位置信息，然后点击确定。

Hive Output-Hive输出

Hive文件存储格式: CSV

Hive文件压缩格式: NONE

输出分隔符: ,

输出字段:

关联 导入 导出

位置	字段名	类型	十进制概
1	id	INTEGER	
2	name	STRING	
3	gender	STRING	
4	time	INTEGER	

添加

确定

步骤 9 连接“表输入”和“Hive 输出”，然后点击下一步。



步骤 10 进入输出配置，存储类型选择 HIVE,输出目录为/user/hive/warehouse/cg_mysqltohive,个数栏填写 1,最后点击保存并运行。

1. 基本信息 2. 输入设置

* 存储类型 HIVE

* 输出目录 hive/warehouse/cg_mysqltohive

* 个数 1

返回 保存 保存并运行 取消

步骤 11 运行 Loader 作业，查看结果。

作业ID	名称	描述	开始时间	执行者	进度	状态
185	cg_mysqltohive	从 RDB 导入 到 H...	2018-04-15 01:48...	stu01	100%	成功

步骤 12 查看 Hive 表 cg_mysqltohive。

```
> select * from cg_mysqltohive;
+---+-----+-----+-----+---+
| cg.id | cg.name | cg.gender | cg.time |
+---+-----+-----+-----+---+
| 1     | tom    | male     | 8       |
| 2     | lily   | female   | 24      |
| 3     | lucy   | fema     |         |
+---+-----+-----+-----+---+
3 rows selected (0.369 seconds)
```

由程序结果可见，成功的将 mysql 表 cga_mysql 的内容加载到 Hive 表 cg_mysqltohive 中。

5.4 实验小结

本实验主要讲述 Loader 在多种业务场景中的使用方法，学员通过实验，可以很好的解决实际业务中的数据迁移问题。特别要注意的是，在 mysql、HBase、Hive 之间进行表数据迁移时，要预先建表。在涉及 mysql 数据库进行 loader 实验时，mysql 表必须要有主键。

6 Flume 数据采集实战

6.1 实验背景

Flume 是大数据组件中重要的数据采集工具，我们常利用 Flume 采集各种数据源的数据供其他组件分析使用。在日志分析业务中，我们常采集服务器日志，以分析服务器运行状态是否正常。在实时业务中，我们常将数据采集到 Kafka 中，以供实时组件 Streaming 或 Spark 等分析处理，Flume 在大数据业务中有着重要的应用。

6.2 实验目的

- 掌握 Flume 的配置和使用，能够使用 Flume 实现数据采集操作。

6.3 实验任务

6.3.1 采集 spooldir 数据到 HDFS

6.3.1.1 根据配置规划工具生成 properties.properties 文件

步骤 1 下载配置工具，配置 Flume。

工具下载地址：

<http://support.huawei.com/enterprise/docinforeader.action?contentId=DOC1000104118&idPath=7919749|7919788|19942925|21110924|21112790|21112791|21624194|21830200>

注：此处 flume 场景为监控文件目录，sink 数据到 hdfs，channel 为内存。

欢迎使用FusionInsight V100R002C60SPC200 Flume配置规划工具

工具版本号	V1.0.0
语言选择	中文
适用FusionInsight版本	V100R002C60SPC200
功能说明	1.支持生成Flume配置文件的生成
Flume名称	client
备注	1.必填配置项不能为空。 2.生成的配置文件在此工具的同级目录下。

步骤 2 配置 Source。

在配置规划工具中的“Flume 配置”表格中，点击“添加 Source”。

填写 SourceName 为 a1，spoolDir 的路径为/home/user01/spooldir(需在/home/user01 下创建 spooldir，并修改权限为 755)，channels 配置为 ch1，其他参数保持默认。

注：此处路径可根据不同账号而定，此处仅以 stu01 为例。

SourceName	Source名称，不能为空，必须唯一。	a1
type	Source类型，取值为spooldir, kafka, http, taildir, avro中的任意一个。	spooldir
spoolDir	待采集的文件所在的目录路径，此参数不能为空。该路径需存在，且对flume运行用户有读写执行权限。	/home/user01/spooldir
channels	当前source读取的数据发送到哪个channel中，此参数不能为空	ch1

步骤 3 配置 Channel 信息。

单击“添加 Channel”，ChanelName 设置为 ch1,type 选择为 memory，其他参数保持默认。

添加Channel		
Channel配置项	Channel配置描述	Channel配置内容
ChanelName	Chanel名称，不能为空，必须唯一。	ch1
type	Channel类型，取值为file, memory中的任意一个。	memory
capacity	缓冲区中缓存的事件条数，不要设置太大，建议设为10000。	10000
transactionCapacity	事务大小：即当前channel支持事务处理的事件个数，建议channel full次数，达到该次数后发送警告事件。	1000
channelFullCount	达到该次数后发送警告事件。	10
keep-alive	在缓冲区添加后者删除一个事件的超时时间。	3

步骤 4 配置 Sink。

单击“添加 Sink”，SinkName 设置为 s1，type 设置为 hdfs，hdfs.path 选择为/stuo1/flume，其他参数保持默认。

SinkName	Sink名称, 不能为空, 必须唯一。	s1
type	Sink类型, 取值为hdfs, hbase, kafka, avro, solr中的任意一个。	hdfs
hdfs.path	写入HDFS的目录, 此参数不能为空	/stu01/flume

hdfs.kerberosPrincipal 参数配置成 FusionInsight Manager 中的集群用户, 如 stu01 用户;
hdfs.kerberosKeytab 的路径为 Linux 中存放该文件的路径, 例如/home/user01/flumetest, 文件权限需配置为 755。

hdfs.kerberosPrincipal	kerberos认证时用户, 在安全版本下必须填写。安全集群需要配置此项, 非安全集群无需配置	stu01
hdfs.kerberosKeytab	kerberos认证时keytab文件路径, 在安全版本下必须填写。安全集群需要配置此项, 非安全集群无需配置	/home/user01/flumetest

配置 Channel 参数为 ch1:

channel	当前Sink读取的数据发送到哪个channel中, 此参数不能为空	ch1
---------	-----------------------------------	-----

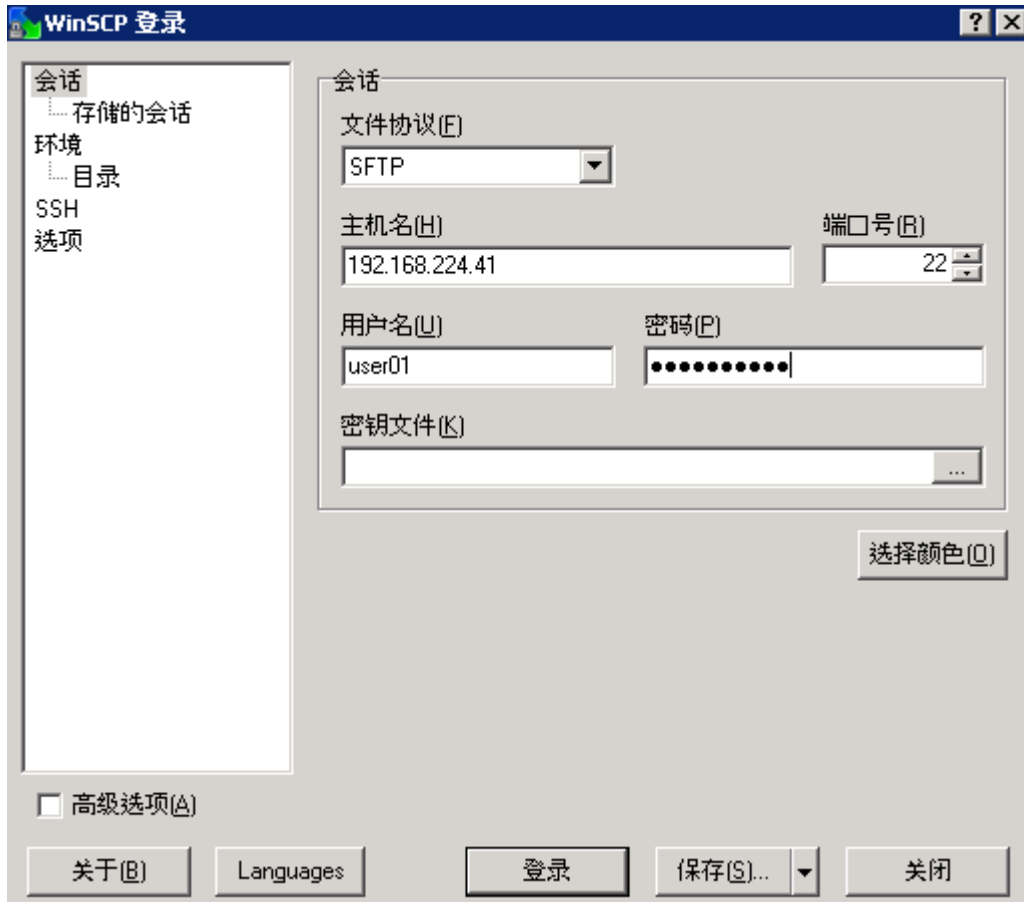
步骤 5 生成配置文件。

生成配置文件

点击生成配置文件, 自动生成一个 properties.properties 文件。

步骤 6 上传 properties.properties 文件到集群节点目录下。

打开 WinSCP 工具, 输入主机名, 用户名和密码, 例如:192.168.224.41, user01, Huawei@123, 然后点击: 登录。



将上传到文件路径/home/user01/flumetest 目录。

步骤 7 检查 Flume 数据采集结果。

```
>hdfs dfs -ls /stu01/flume
```

6.3.1.2 安装 Flume 客户端

步骤 1 解压 Flume 客户端。

```
> cp /FusionInsight_Client/FusionInsight_V100R002C60SPC200_Flume_Client.tar
/home/user01/
> tar -xvf FusionInsight_V100R002C60SPC200_Flume_Client.tar
```

解压后得到两个文件：

FusionInsight_V100R002C60SPC200_Flume_ClientConfig.tar 和
FusionInsight_V100R002C60SPC200_Flume_ClientConfig.tar.sha256

使用 tar 命令继续解压 FusionInsight_V100R002C60SPC200_Flume_ClientConfig.tar：

```
>tar -xvf FusionInsight_V100R002C60SPC200_Flume_ClientConfig.tar
```

得到目录 FusionInsight_V100R002C60SPC200_Flume_ClientConfig。

拷贝目录 FusionInsight_V100R002C60SPC200_Flume_ClientConfig 下的 Flume，到/home/user01 下，然后解压文件/home/user01/Flume/FusionInsight-Flume-1.6.0.tar.gz：

```
>tar -xvf FusionInsight-Flume-1.6.0.tar.gz
>ls
```

adapter aix batch_install flume FusionInsight-Flume-1.6.0.tar.gz install.sh
得到文件夹 adapter aix batch_install flume 和文件 install.sh。

步骤 2 获取 krb5.conf 和 user.keytab 文件。

使用 FusionInsight Manager 账户如 stu01 登录 FusionInsight Manager，然后依次点击“系统设置”，“权限配置”，“用户管理”。



然后在对应账户的操作栏，单击下载按钮进行下载，解压后得到 krb5.conf 和 user.keytab 文件。



并使用 WinScp 工具将 krb5.conf 和 user.keytab 上传到/home/user01/flumetest 目录中。

步骤 3 创建 jaas.conf 文件。

在步骤 1 中解压的/home/user01/flumetest 目录下新建一个文件，并命名为 jaas.conf。

```
>touch jaas.conf
```

编辑 jaas.conf 文件：

```
>vim jaas.conf
```

jaas.conf 文件内容如下：

```
Client {  
    com.sun.security.auth.module.Krb5LoginModule required  
    storeKey=true  
    principal="stu01" (表示 FusionInsight Manager 上面创建的用户)  
    useTicketCache=false  
    keyTab= "/home/user01/flumetest/user.keytab" (这里是修改为下载的 FusionInsight  
    Manager 用户 stu01 的认证文件 user.keytab 在 Linux 上面路径。)  
    debug=true  
    useKeyTab=true;  
};
```

步骤 4 修改 flume-env.sh 文件。

flume-env.sh 存在于 Flume 客户端解压文件的 flume/conf 目录下面，在 JAVA_OPTS 的尾部追加如下内容：

```
> vim flume-env.sh
-Djava.security.krb5.conf=/home/user01/flumetest/krb5.conf
-Djava.security.auth.login.config=/home/user01/flumetest/jaas.conf
-Dzookeeper.server.principal=zookeeper/hadoop.hadoop.com
-Dzookeeper.request.timeout=120000
```

(注：这里 Djava.security.auth.login.config, Djava.security.krb5.conf 值必须是自己实际集群中对应的路径。)

```
JAVA_OPTS="-Xms2G -Xmx4G -XX:CMSFullGCsBeforeCompaction=1 -
XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -
XX:+UseCMSCompactAtFullCollection -verbose:gc -XX:+UseGCLogFileRotation -
XX:NumberOfGCLogFiles=15 -XX:GCLogFileSize=1M -XX:+PrintGCDetails -
XX:+PrintGCDateStamps -Xloggc:${FLUME_GC_LOG_DIR}/Flume-Client-gc.log -
verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -
Djava.security.krb5.conf=/home/user01/flumetest/krb5.conf -
Djava.security.auth.login.config=/home/user01/flumetest/jaas.conf -
Dzookeeper.server.principal=zookeeper/hadoop.hadoop.com -
Dzookeeper.request.timeout=120000"
```

在已经安装好的 HDFS 客户端中，拷贝 hdfs_client/HDFS/hadoop/etc/hadoop/目录下的 hdfs-site.xml 和 core-site.xml 到/home/user01/flumetest 下；HBase 客户端 hbase_client/HBase/hbase/conf 下的 hbase-site.xml 文件到/home/user01/flumetest 下。

```
>cd /home/user01/hdfs_client/HDFS/hadoop/etc/hadoop/
> cp hdfs-site.xml /home/user01/flumetest
> cp core-site.xml /home/user01/flumetest
>cd /home/user01/hbase_client/HBase/hbase/conf
> cp hbase-site.xml /home/stu01/flumetest
```

查看/home/user01/flumetest 文件夹下的内容。

```
cd /home/user01/flumetest
> ll
total 56
-rw----- 1 stu01 users 8563 Apr 16 22:49 core-site.xml
-rw----- 1 stu01 users 9830 Apr 16 22:50 hbase-site.xml
-rw----- 1 stu01 users 15277 Apr 16 22:48 hdfs-site.xml
-rw-r--r-- 1 stu01 users 199 Apr 16 22:23 jaas.conf
-rw-r--r-- 1 stu01 users 757 Apr 15 20:24 krb5.conf
-rw-r--r-- 1 stu01 users 2119 Apr 16 21:12 properties.properties
-rw-r--r-- 1 stu01 users 126 Apr 15 20:24 user.keytab
```

步骤 5 安装客户端（在使用非 root 用户时，建议安装客户端过程中，客户端安装的目录不要过深，否则容易安装失败。）

```
> ./install.sh -d /home/user01/flume1 -f 192.168.224.41 -c
flume/conf/properties.properties -l /var/log/Bigdata/
系统提示[flume-client install]: install flume client successfully 标明客户端安装成功。
```

参数说明：

“-d”：Flume 客户端安装路径；

“-f”：两个 MonitorServer 角色的业务 IP，中间用逗号分隔，可选，若不设置则 Flume 客户端不向 MonitorServer 发送告警信息；

“-c”：配置文件，可选，安装以后可通过修改“/opt/FlumeClient/fusioninsight-flume-1.6.0/conf/properties.properties”配置 Flume 角色客户端参数；

“-l”：日志目录，可选，默认值为“/var/log/Bigdata”（“user”用户需要对此目录有写权限）；

步骤 6 查看/home/user01/spooldir，出现“.flumespool”，表示配置成功。

```
> ll /home/user01/spooldir -a
total 408
drwxrwxrwx  3 root root   4096 Feb  9 13:44 .
drwxrwxrwx 81 root root 12288 Apr 16 23:26 ..
drwxrwxrwx  2 omm wheel  4096 Jan 26 23:46 .flumespool
-rwxrwxrwx  1 root root 389592 Jan 26 23:45 zypper.log.COMPLETED
```

6.3.2 采集 avro 数据到 HDFS

Flume 对 avro 数据源的采集，就是对一种序列化数据的采集，对于这类数据的采集，会涉及到端口的配置。

6.3.2.1 根据配置规划工具生成 properties.properties 文件

步骤 1 设置 Flume 连接名称，选择 client。

欢迎使用FusionInsight V100R002C60SPC200 Flume配置规划工具

工具版本号	V1.0.0
语言选择	中文
适用FusionInsight版本	V100R002C60SPC200
功能说明	1.支持生成Flume配置文件的生成
Flume名称	client
备注	1.必填配置项不能为空。 2.生成的配置文件在此工具的同级目录下。

步骤 2 设置连接的数据源为 avro，同时设置监听的 IP 地址和端口号，设置 channels 为“ch2”，点击添加 Source。

添加Source		
Source配置项	Source配置描述	Source配置内容
SourceName	Source名称, 不能为空, 必须唯一。	a2
type	Source类型, 取值为spooldir, kafka, hbase, avro, source绑定的ip地址, 此参数不能为空。为之相连的avro source监听的端口, 此参数不能为空。为之相连的avro source处理消息的最大线程数, 类型为数字	avro
bind		192.168.225.11
port		8181
threads		5
channels	从source读取的数据发送到哪个channel中, 此参数不能为空	ch2

步骤 3 配置 Channel 参数。

点击“添加 Channel”，ChanelName 配置为：ch2，type 配置为 memory，其他参数保持默认。

ChanelName	Chanel名称, 不能为空, 必须唯一。	ch2
type	Chanel类型, 取值为file, memory中的任意一个。	memory

步骤 4 配置 Sink 参数。

点击“添加 Sink”。

SinkName 配置为 s2;

Hdfs.path 配置为/stuo1/flume_avro。

如下图所示：

SinkName	Sink名称, 不能为空, 必须唯一。	s2
type	Sink类型, 取值为hdfs, hbase, kafka, avro, solr中的任意一个。	hdfs
hdfs.path	写入HDFS的目录, 此参数不能为空	/stu01/flume_avro

设置认证信息，包括认证账户和认证文件的地址。认证账户和认证文件可与 7.3.1 中保持一致。

kerberos认证时用户，在安全版本下必须填写。安全集群需要配置此项，非安全集群无需配置	stu01
kerberos认证时keytab文件路径，在安全版本下必须填写。安全集群需要配置此项，非安全集群无需配置	/home/user01/flumetest2

在集群使用安全模式部署时，需配置 `hdfs.kerberosPrincipal` 和 `hdfs.kerberosKeytab` 参数，如果集群为非安全模式，这两个参数可以不填。

创建 `/home/user01/flumetest2` 目录，并配置权限为 755。

步骤 5 生成配置文件。

单击“生成配置文件”，并采用和之前相同的方式将 `properties.properties` 配置文件上传到集群指定目录文件夹下，如 `/home/user01/flumetest2`，会覆盖掉 7.3.1 中的 `properties.Properties` 文件。

步骤 6 获取 `krb5.conf` 和 `user.ekytab` 文件。

此步骤可对比参考 7.3.1.2 中的步骤 2 进行操作

步骤 7 创建 `jaas.conf` 文件。

此步骤可对比参考 7.3.1.2 中的步骤 3 进行操作。

6.3.2.2 新建 Flume 作业

步骤 1 重新安装 Flume 实例到 `/home/user01/flume2`。

```
> ./install.sh -d /home/user01/flume2 -f 192.168.224.41 -c
/home/user01/flumetest/properties.properties -l /var/log/Bigdata/
```

系统提示[flume-client install]: install flume client successfully.表示安装成功。

注：安装成功之后，可通过命令“`ps -ef | grep flume | grep username`”查看 Flume 服务情况。

步骤 2 往 avro 的端口提交数据。

从 `FusionInsight_Client` 目录拷贝文件 `flumeavroclient.jar` 到 `/home/user01`，`flumeavroclient.jar` 可向 8181 端口提交 100 次“Hello, World”，并保存在 `over.tmp` 文件中。

```
> cd /opt
> java -cp flumeavroclient.jar org.myorg.SSLAvroclient
```

步骤 3 查看 HDFS 中数据采集结果。

```
>hdfs dfs -ls /stu01/flume_avro
-rw-r--r--  3 stu01 supergroup 1300 2018-04-17 /stu01/flume_avro/over.tmp
```

6.4 实验小结

本实验主要讲述 Flume 中 spooldir 和 avro 两种数据源类型的数据采集操作。通过该实验，学员可掌握常见的离线数据和实时数据采集方法，使学员更好的理解和掌握 Flume。

7 集群综合实验

7.1 实验背景

大数据业务中，通常需要将多种组件构建成一个业务系统，以满足上层业务需要。

本实验将前面的组件进行有机的组合，构建一个大数据分析和实时查询平台：

首先由 Loader 定时将 Mysql 数据库数据迁移到 Hive 中，由于 Hive 数据存储于 HDFS 中，所以采用 Loader 将 HDFS 中的数据导入到 HBase 中。利用 HBase 进行实时数据查询，利用 Hive 的大数据处理能力，分析相关结果。

7.2 实验目的

- 综合运用大数据组件进行数据的转换和实时查询。

7.3 实验任务

7.3.1 Mysql+Loader+Hive+HBase 离线数据采集分析与实时查询

7.3.1.1 Mysql 数据准备

步骤 1 登录 MySQL 服务器。

MySQL 安装在 FusiInsight HD 集群节点上。

```
> mysql -uroot -p123456
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 135
Server version: 5.5.48 MySQL Community Server (GPL)
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

步骤 2 新建数据库 loadertest。

```
mysql> create database loadertest;
mysql> use loadertest;
```

步骤 3 新建表 socker，并命名 time 为主键。

```
mysql> DROP TABLE IF EXISTS `socker`;
mysql> CREATE TABLE `socker` (
  `time` varchar(50) DEFAULT NULL,
  `open` float DEFAULT NULL,
  `high` float DEFAULT NULL,
  `low` float DEFAULT NULL,
  `close` float DEFAULT NULL,
  `volume` varchar(50) DEFAULT NULL,
  `endprice` float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

步骤 4 导入数据到 socker。

从/FusionInsight_Client 目录下拷贝 socker.csv 文件到本地 home 目录。

```
> cp /FusionInsight_Client/socker.csv /home/user01
```

在 mysql 客户端工具下导入 socker.csv 数据到表 socker。

```
mysql> LOAD DATA INFILE "/home/user01/socker.csv" INTO TABLE socker;
```

步骤 5 查看 socker 数据。

```
mysql> select * from socker limit 10;
+-----+-----+-----+-----+-----+-----+-----+
| time      | open  | high  | low   | close | volume  | endprice |
+-----+-----+-----+-----+-----+-----+-----+
| 1970-01-02 | 92.06 | 93.54 | 91.79 | 93    | 8050000 | 93       |
| 1970-01-05 | 93    | 94.25 | 92.53 | 93.46 | 11490000 | 93.46    |
```

1970-01-06	93.46	93.81	92.13	92.82	11460000	92.82
1970-01-07	92.82	93.38	91.93	92.63	10010000	92.63
1970-01-08	92.63	93.47	91.99	92.68	10670000	92.68
1970-01-09	92.68	93.25	91.82	92.4	9380000	92.4
1970-01-12	92.4	92.67	91.2	91.7	8900000	91.7
1970-01-13	91.7	92.61	90.99	91.92	9870000	91.92
1970-01-14	91.92	92.4	90.88	91.65	10380000	91.65
1970-01-15	91.65	92.35	90.73	91.68	11120000	91.68

+-----+-----+-----+-----+-----+-----+-----+

7.3.1.2 Mysql 数据导入 Hive

步骤 1 根据 6.3.1 的步骤 1 到步骤 3 的操作，来到配置 Loader 基本信息的界面。

点击“编辑”将 JDBC 连接字符串的数据库配置为：

`jdbc:mysql://192.168.224.41:3306/loadertest1。`

此处服务器 IP 地址需由讲师分配，如不清楚，可询问讲师。

1.基本信息 2.输入设置

* 名称 cg_mysqltohive2

* 类型 导入

* 连接 mysql +添加 编辑

组 请选择... +添加

* 队列 default

优先级 NORMAL

下一步 取消

mysql

* 名称 mysql

* JDBC驱动程序类 com.mysql.jdbc.Driver

* JDBC连接字符串 jdbc:mysql://192.168.224.41:3306/loadertest1

* 用户名 root

* 密码

名称	值
添加	

步骤 2 配置输入设置。

“表名” 设置为 socker，然后单击 “下一步”。

1. 基本信息 2. 输入设置

☒ 表方式 ☐ SQL方式

架构名称

* 表名

表列名

分区列名

分区列空值 ☐ true ☒ false

返回 下一步 取消

注：若表中没有设置主键，则分区列名要指定一个值，“1”或列名“time”。

步骤 3 配置 “输入设置”。

将“表输入”按钮拖到右侧。

1. 基本信息 2. 输入设置 3. 转换

过滤...

输入 -

表输入

输出 +

转换 +

表输入

步骤 4 双击“表输入”，输入配置中与 mysql 关联的相关属性，字段为 mysql 对应字段。

Table Input-表输入

输入字段

导入 导出

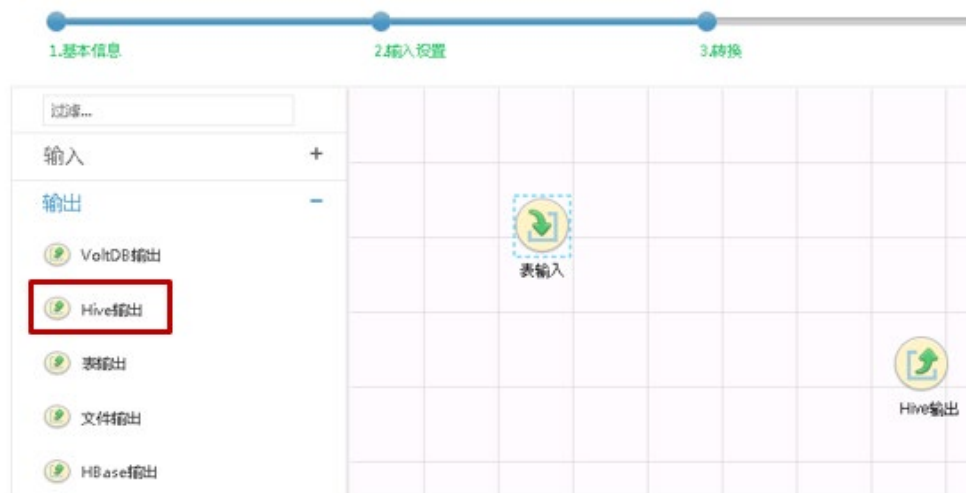
位置	字段名	类型	长度
1	time	VARCHAR	50
2	open	FLOAT	
3	high	FLOAT	
4	low	FLOAT	
5	close	FLOAT	
6	volume	VARCHAR	50
7	endprice	FLOAT	

添加

确定 取消

步骤 5 配置“Hive 输出”。

将“Hive 输出”按钮拖到右边空白处。



步骤 6 输出表参数配置。

双击“Hive 输出”按钮。根据提示填写参数，其中“输出分隔符”配置为“,”，并添加“输出字段”，如下图所示：

Hive Output-Hive输出

Hive文件存储格式 CSV

Hive文件压缩格式 NONE

输出分隔符 ,

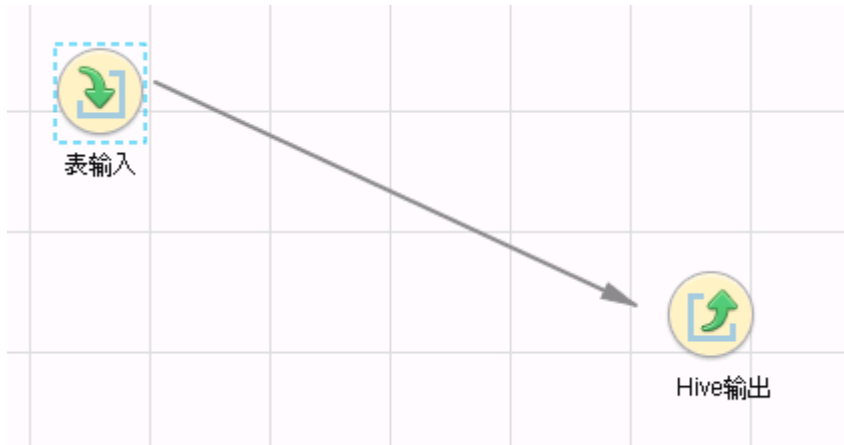
输出字段

关联 导入 导出

位置	字段名	类型
1	time	VARCHAR
2	open	FLOAT
3	high	FLOAT
4	low	FLOAT
5	close	FLOAT
6	volume	VARCHAR
7	endprice	FLOAT

添加

步骤 7 连接“表输入”与“Hive 输出”。



步骤 8 在 HDFS 中新建路径/stuo1/hive/warehouse/socker2。

```
> hdfs dfs -mkdir /stu01/hive/warehouse/socker2
```

步骤 9 在 Hive 数据仓库中新建 socker2。

```
> create table socker2(time string,open float,high float,low float,close float,volume string,endprice float)
row format delimited fields terminated by ',' stored as textfile
location '/stu01/hive/warehouse/socker2';
```

步骤 10 进行输出配置,存储类型为 HIVE, 输出目录为/stuo1/hive/warehouse/socker2,个数为 2。

1.基本信息 2.输入设置

* 存储类型

* 输出目录

* 个数

步骤 11 保存并运行，看到如下结果。

作业ID	名称	描述	开始时间	执行者	进度	状态
186	cg_mysqltohive2	从 RDB 导入 到 HIVE	2018-04-15 07:13:...	stu01	100%	成功

步骤 12 查看运行结果。

```
> select * from socker2 limit 10;
```

```
+-----+-----+-----+-----+-----+
| socker2.time | socker2.open | socker2.high | socker2.low | socker2.close |
| socker2.volume | socker2.endprice |
+-----+-----+-----+-----+-----+
| 1970-01-02 | 92.06 | 93.54 | 91.79 | 93.0 |
8050000 | 93.0 |
| 1970-01-05 | 93.0 | 94.25 | 92.53 | 93.46 |
11490000 | 93.46 |
| 1970-01-06 | 93.46 | 93.81 | 92.13 | 92.82 |
11460000 | 92.82 |
| 1970-01-07 | 92.82 | 93.38 | 91.93 | 92.63 |
10010000 | 92.63 |
| 1970-01-08 | 92.63 | 93.47 | 91.99 | 92.68 |
10670000 | 92.68 |
| 1970-01-09 | 92.68 | 93.25 | 91.82 | 92.4 |
9380000 | 92.4 |
| 1970-01-12 | 92.4 | 92.67 | 91.2 | 91.7 |
8900000 | 91.7 |
| 1970-01-13 | 91.7 | 92.61 | 90.99 | 91.92 |
9870000 | 91.92 |
| 1970-01-14 | 91.92 | 92.4 | 90.88 | 91.65 |
10380000 | 91.65 |
| 1970-01-15 | 91.65 | 92.35 | 90.73 | 91.68 |
11120000 | 91.68 |
```

```
> hdfs dfs -ls /user/hive/warehouse/socker2
```

```
18/04/15 22:23:45 INFO hdfs.PeerCache: SocketCache disabled.
Found 2 items
-rw-rw----+ 3 stu01 supergroup          0 2018-04-15 22:13
/user/hive/warehouse/socker2/_SUCCESS
-rw-rw----+ 3 stu01 supergroup    559000 2018-04-15 22:13
/user/hive/warehouse/socker2/part-m-00000
```

7.3.1.3 使用 Hive 进行分析查询

步骤 1 获取涨幅最大的股票

获取最新的涨幅数据，然后将结果保存到一张新建的表中。

```
>beeline
>select socker.time, socker.open, socker.endprice from socker where
socker.endprice> socker.open sort by socker. endprice desc;
| socker.time | socker.open | socker.endprice |
+-----+-----+-----+
| 1974-05-21 | 87.86 | 87.91 |
| 1978-03-09 | 87.84 | 87.89 |
| 1978-03-08 | 87.36 | 87.84 |
| 1975-12-04 | 87.6 | 87.84 |
| 1975-12-12 | 87.8 | 87.83 |
| 1970-02-19 | 87.44 | 87.76 |
| 1974-06-24 | 87.46 | 87.69 |
| 1978-02-23 | 87.56 | 87.64 |
| 1970-03-18 | 87.29 | 87.54 |
| 1970-12-01 | 87.2 | 87.47 |
| 1978-03-03 | 87.32 | 87.45 |
| 1970-02-18 | 86.37 | 87.44 |
| 1974-05-30 | 86.89 | 87.43 |
| 1978-03-07 | 86.9 | 87.36 |
| 1978-03-02 | 87.19 | 87.32 |
| 1975-12-09 | 87.07 | 87.3 |
.....
+-----+-----+-----+
5,228 rows selected (30.544 seconds)
```

步骤 2 获取最新的涨幅股票

```
>select socker.time, socker.open, socker.endprice from socker where
socker.endprice> socker.open sort by socker.time desc;
| socker.time | socker.open | socker.endprice |
+-----+-----+-----+
| 1970-04-09 | 88.49 | 88.53 |
| 1970-04-01 | 89.63 | 90.07 |
| 1970-03-26 | 89.77 | 89.92 |
| 1970-03-25 | 88.11 | 89.77 |
| 1970-03-24 | 86.99 | 87.98 |
| 1970-03-18 | 87.29 | 87.54 |
| 1970-03-17 | 86.91 | 87.29 |
```

```
| 1970-03-10 | 88.51 | 88.75 |
| 1970-03-03 | 89.71 | 90.23 |
| 1970-03-02 | 89.5 | 89.71 |
| 1970-02-27 | 88.9 | 89.5 |
| 1970-02-25 | 87.99 | 89.35 |
| 1970-02-20 | 87.76 | 88.03 |
| 1970-02-19 | 87.44 | 87.76 |
| 1970-02-18 | 86.37 | 87.44 |
.....
+-----+-----+-----+-----+
5,228 rows selected (26.738 seconds)
```

步骤 3 获取增长的股票总数

```
>select count(*) from socker where socker.endprice> socker.open;
+-----+---+
| _c0 |
+-----+---+
| 5228 |
+-----+---+
```

步骤 4 创建表格获取涨幅股票的数据新建一张表格，然后将表格数据在下一节导入到 HBase 中。

创建表格：

```
>create table upsocket like socker;
```

导入数据：

```
>insert into upsocket select * from socker where socker.endprice> socker.open
sort by socker.endprice desc;
```


7.3.1.4 HDFS 数据导入 HBase

步骤 1 在 HBase 中新建名叫 cg_hdfstohbase2 的数据表。

```
hbase(main):002:0> create 'cg_hdfstohbase2','info';
0 row(s) in 0.3900 seconds
=> Hbase::Table - cg_hdfstohbase2
```

步骤 2 由 6.3.1 的步骤 1 到 3，来到配置 Loader “基本信息” 的界面。

填写相关项参数，如下图所示：



1. 基本信息

2. 输入设置

* 名称	cg_hdfstohbase2		
* 类型	导入		
* 连接	hh	+ 添加	编辑
组	default	+ 添加	
* 队列	default		
优先级	NORMAL		

下一步 取消

配置好后，单击下一步。

步骤 3 配置 “输入设置”。

设置 HDFS 输入数据文件的路径，及编码类型。



1. 基本信息

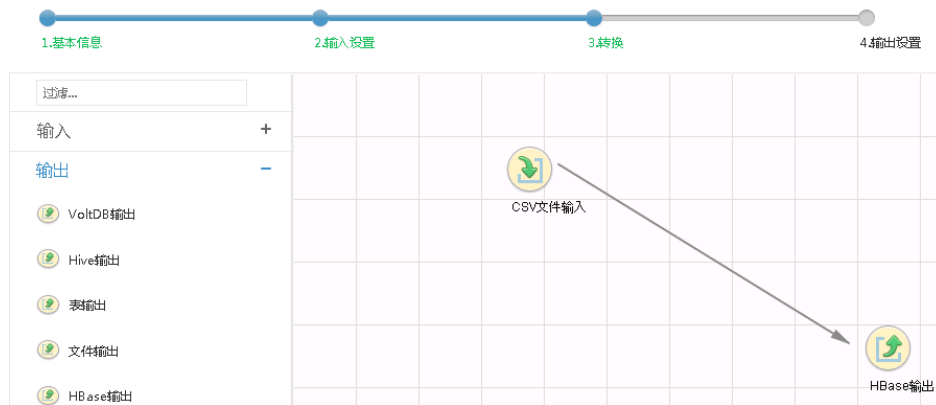
2. 输入设置

* 输入路径	rehouse/socker2/part-m-00000
路径过滤器	
* 文件过滤器	*
* 编码类型	UTF_8
后缀名	

返回 下一步 取消

步骤 4 配置 “转换”。

分别选择“CSV 文件输入”和“HBase 输出”，将它们拖到右侧空白处并连接。



步骤 5 配置“CSV 文件输入”

依据 HDFS 中存储数据的格式进行输入数据列设置。

分隔符配置为“,”，并添加输入字段，如下图所示：

CSV File Input-CSV文件输入

分隔符: ,

换行符:

文件名是否作为字段:

绝对路径: ☐

验证输入字段: YES

输入字段

位置	字段名	类型
1	time	VARCHAR
2	open	VARCHAR
3	high	VARCHAR
4	low	VARCHAR
5	close	VARCHAR
6	volume	VARCHAR
7	endprice	VARCHAR

添加

确定

步骤 6 配置“HBase 输出”按钮。

HBase 输出表的列和列簇设置。

HBase Output-HBase输出

HBase表类型: normal

NULL值处理方式: ☐

HBase输出字段

关联 导入 导出

字段名	表名	列族名	列名	类型	长度	主键
time	cg_hdfstohbase2	info	time	VARCHAR		<input checked="" type="checkbox"/>
open	cg_hdfstohbase2	info	open	VARCHAR		<input type="checkbox"/>
high	cg_hdfstohbase2	info	high	VARCHAR		<input type="checkbox"/>
low	cg_hdfstohbase2	info	low	VARCHAR		<input type="checkbox"/>
close	cg_hdfstohbase2	info	close	VARCHAR		<input type="checkbox"/>
volume	cg_hdfstohbase2	info	volume	VARCHAR		<input type="checkbox"/>
endprice	cg_hdfstohbase2	info	endprice	VARCHAR		<input type="checkbox"/>

添加

确定 取消

步骤 7 输出配置。

存储类型为 HBASE_PUTLIST, HBase 实例选择 HBase, 个数为 1。

1.基本信息 2.输入设置

* 存储类型: HBASE_PUTLIST

* HBase实例: HBase

☒ Map数 ☐ Map数据块大小

* 个数: 1

返回 保存 保存并运行 取消

步骤 8 查看运行结果。

作业ID	名称	描述	开始时间	执行者	进度	状态
187	cg_hdfstohbase2	从 HDFS 导入 到 H...	2018-04-15 07:48...	stu01	100%	成功

步骤 9 查看 HBase 表 cg_hdfstohbase2 的内容。

```
hbase(main):005:0> scan 'cg_hdfstohbase2'
...
```

```

2009-09-15      column=info:high, timestamp=1523803747562, value=1056.04
2009-09-15      column=info:low, timestamp=1523803747562, value=1043.42
2009-09-15      column=info:open, timestamp=1523803747562, value=1049.03
2009-09-15      column=info:volume, timestamp=1523803747562, value=6185620000
10022 row(s) in 8.5350 seconds

```

7.3.1.5 HBase 数据实时查询

步骤 1 在 HBase Shell 客户端下，查询表 “cg_hdfstohbase2” 中指定行 “2009-09-15” 中的信息。

```

> get 'cg_hdfstohbase2','2009-09-15'
COLUMN                                CELL
info:close                            timestamp=1523803747562, value=1052.63
info:endprice                         timestamp=1523803747562, value=1052.63
info:high                             timestamp=1523803747562, value=1056.04
info:low                              timestamp=1523803747562, value=1043.42
info:open                             timestamp=1523803747562, value=1049.03
info:volume                           timestamp=1523803747562, value=6185620000
6 row(s) in 0.0420 seconds

```

步骤 2 查询指定时间段 2009 年 8 月 15 到 2009 年 9 月 15 日内的信息。

```

> scan 'cg_hdfstohbase2',{COLUMN=>'info:endprice',STARTROW=>'2009-08-15',STOPROW=>'2009-09-15'}
ROW                                COLUMN+CELL
2009-08-17      column=info:endprice, timestamp=1523803747562, value=979.73
2009-08-18      column=info:endprice, timestamp=1523803747562, value=989.67
2009-08-19      column=info:endprice, timestamp=1523803747562, value=996.46
2009-08-20      column=info:endprice, timestamp=1523803747562, value=1007.37
.....
2009-09-09      column=info:endprice, timestamp=1523803747562, value=1033.37
2009-09-10      column=info:endprice, timestamp=1523803747562, value=1044.14
2009-09-11      column=info:endprice, timestamp=1523803747562, value=1042.73
2009-09-14      column=info:endprice, timestamp=1523803747562, value=1049.34
20 row(s) in 0.0380 seconds

```

步骤 3 查询大于某个值的所有列（系统会把数值当成字符串进行比较）。

```

> scan 'cg_hdfstohbase2',{FILTER => "ValueFilter(>,'binary:979')"}
...

2009-09-02      column=info:low, timestamp=1523803747562, value=991.97
2009-09-02      column=info:open, timestamp=1523803747562, value=996.07
2009-09-03      column=info:low, timestamp=1523803747562, value=992.25
2009-09-03      column=info:open, timestamp=1523803747562, value=996.12
661 row(s) in 0.2230 seconds

```

步骤 4 查询以 endprice 结尾的所有信息，且字符串值大于 979。

```
hbase(main):011:0> scan 'cg_hdfstohbase2',{FILTER=>"ValueFilter(>,'binary:979')
AND ColumnPrefixFilter('endprice')"}
2009-08-18      column=info:endprice, timestamp=1523803747562, value=989.67
2009-08-19      column=info:endprice, timestamp=1523803747562, value=996.46
2009-09-01      column=info:endprice, timestamp=1523803747562, value=998.04
2009-09-02      column=info:endprice, timestamp=1523803747562, value=994.75
327 row(s) in 0.1180 seconds
```

7.4 实验小结

本实验使用多种组件构建大数据分析查询平台，通过本实验，增强学员对大数据组件的理解和综合应用。

8 附录

8.1 Linux 常用命令

```
cd /home 进入 '/home' 目录'
cd .. 返回上一级目录
cd ../.. 返回上两级目录
cd 进入个人的主目录
cd ~user1 进入个人的主目录
cd - 返回上次所在的目录
pwd 显示工作路径
ls 查看目录中的文件
ls -F 查看目录中的文件
ls -l 显示文件和目录的详细资料
ls -a 显示隐藏文件
```

ls *[0-9]* 显示包含数字的文件名和目录名
tree 显示文件和目录由根目录开始的树形结构(1)
lstrree 显示文件和目录由根目录开始的树形结构(2)
mkdir dir1 创建一个叫做 'dir1' 的目录'
mkdir dir1 dir2 同时创建两个目录
mkdir -p /tmp/dir1/dir2 创建一个目录树
rm -f file1 删除一个叫做 'file1' 的文件'
rmdir dir1 删除一个叫做 'dir1' 的目录'
rm -rf dir1 删除一个叫做 'dir1' 的目录并同时删除其内容
rm -rf dir1 dir2 同时删除两个目录及它们的内容
mv dir1 new_dir 重命名/移动 一个目录
cp file1 file2 复制一个文件
cp dir/* . 复制一个目录下的所有文件到当前工作目录
cp -a /tmp/dir1 . 复制一个目录到当前工作目录
cp -a dir1 dir2 复制一个目录
ln -s file1 lnk1 创建一个指向文件或目录的软链接

8.2 HDFS 其他命令

HDFS 支持 fsck 命令用以检查各种不一致。fsck 用以报告各种文件问题，如 block 丢失或缺少 block 等。

fack 命令用法如下：

hdfs fsck <path> [-move | -delete | -openforwrite] [-files [-blocks [-locations | -racks]]]

<path>	检查的起始目录
-move	将损坏的文件移动到/lost+found 下面
-delete	删除损坏的文件
-openforwrite	打印出正在写的文件
-files	打印出所有被检查的文件
-blocks	打印出 block 报告
-locations	打印出每个 block 的位置
-racks	打印出 datanode 的网络拓扑结构

8.3 Flume 新建作业方法

Flume 作业的创建，可以采用两种方式：一种是更新 properties.properties 配置文件；另一种是重新安装客户端。

第一种针对之前已经有客户端存在的情况，第二种针对之前没有客户端的情况或是第一种方式采集数据失败的情况。

第一种处理方式如下：

重新生成 properties.properties 配置文件，然后替换之前的 properties.properties 配置文件，然后重启 Flume 服务。

第 7 章中 Flume 新建作业采用的是第二种方法。