

# 3 HBase

## 3.1 实验介绍

### 3.1.1 关于本实验

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。

### 3.1.2 实验目的

掌握 HBase 的安装部署、常用操作、Region 操作及 Filter 的使用

### 3.1.3 内容描述

- 子实验 1 安装部署 BigDataPro 解决方案的 HBase 组件
- 子实验 2 掌握 HBase 的常用操作、Filter 的使用及 Region 操作等

## 3.2 实验任务 1

### 3.2.1 启动 Hadoop 集群, 验证集群是否可用

步骤 1 在 node1 节点执行以下命令:

```
> start-dfs.sh ; start-yarn.sh
```

返回信息中有以下内容时, 表示 Hadoop 集群启动成功:

```
Starting namenodes on [node1]
Starting secondary namenodes [node1]
starting yarn daemons
```

## 步骤 2 查看 Java 进程

使用 jps 命令在 node1-4 中查看 Java 进程, node1 中有 NameNode、SecondaryNameNode 和 ResourceManager 进程:

```
[root@node1 ~]# jps
1538 WrapperSimpleApp
5732 SecondaryNameNode
5508 NameNode
6205 Jps
5918 ResourceManager
```

node2-4 中有 NodeManager 和 Datanode 进程:

```
[root@node2 ~]# jps
3026 Jps
2740 DataNode
1515 WrapperSimpleApp
2862 NodeManager
```

步骤 3 访问 <http://node1:50070>, 可以登录 Namenode 的 Web 界面

Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities

## Overview 'node1:8020' (active)

Started:	Mon Apr 13 14:08:02 +0800 2020
Version:	2.8.3, rb3fe56402d908019d99af1f1f4fc65cb1d1436a2
Compiled:	Tue Dec 05 11:43:00 +0800 2017 by jdu from branch-2.8.3
Cluster ID:	CID-b9ee4afe-bb8f-4076-b892-73c070471d31
Block Pool ID:	BP-1222873527-127.0.0.1-1586413786055

## Summary

Security is off.

Safemode is off.

15 files and directories, 3 blocks = 18 total filesystem object(s).

图 3-1 HDFS 的 WebUI 界面

## 3.2.2 安装部署 HBase 组件

### 3.2.2.1 准备 HBase 组件

在 node1 上准备 HBase 组件，然后拷贝到其它从节点上。

#### 步骤 1 解压 HBase 软件包

在 node1 节点解压 HBase 软件包至/home/modules 路径下

```
> tar -zxvf /home/extend_tools/hbase-2.1.0-bin.tar.gz -C /home/modules/  
> ls /home/modules/ | grep hbase  
hbase-2.1.0
```

#### 步骤 2 修改 hbase-env.sh 文件

将 hbase-env.sh 文件的 JAVA\_HOME 变量修改为当前操作系统正确变量：

```
> vim /home/modules/hbase-2.1.0/conf/hbase-env.sh  
  
# 在最后一行补充以下内容：  
export JAVA_HOME=/usr/lib/jvm/java
```

#### 步骤 3 修改 hbase-site.xml 配置文件（请记得修改桶名称）

```
> vim /home/modules/hbase-2.1.0/conf/hbase-site.xml  
  
# 将<configuration>至</configuration>中的内容替换为如下：  
<property>  
  <name>hbase.rootdir</name>  
  <value>obs://bucket_name/hbasetest001</value>  
</property>  
<property>  
  <name>zookeeper.session.timeout</name>  
  <value>120000</value>  
</property>  
<property>  
  <name>hbase.zookeeper.property.tickTime</name>  
  <value>6000</value>  
</property>  
<property>  
  <name>hbase.zookeeper.property.dataDir</name>  
  <value>/home/modules/hbase-2.1.0/data/zookeeper</value>  
</property>  
<property>  
  <name>hbase.cluster.distributed</name>  
  <value>true</value>  
</property>  
<property>  
  <name>hbase.zookeeper.quorum</name>
```

```
<value>node1,node2,node3</value>
</property>
<property>
  <name>hbase.tmp.dir</name>
  <value>/home/modules/hbase-2.1.0/tmp</value>
</property>
<property>
  <name>hbase.wal.provider</name>
  <value>org.apache.hadoop.hbase.wal.FSHLogProvider</value>
</property>
<property>
  <name>hbase.wal.dir</name>
  <value>hdfs://node1:8020/hbase</value>
</property>
<property>
  <name>hbase.client.write.buffer</name>
  <value>5242880</value>
</property>
<property>
  <name>hbase.regionserver.handler.count</name>
  <value>200</value>
</property>
<property>
  <name>hbase.hstore.compaction.min</name>
  <value>6</value>
</property>
<property>
  <name>hbase.hregion.memstore.block.multiplier</name>
  <value>16</value>
</property>
<property>
  <name>hfile.block.cache.size</name>
  <value>0.2</value>
</property>
```

#### 步骤 4 配置 Regionserver

```
> vim /home/modules/hbase-2.1.0/conf/regionservers

# 删除默认的 localhost, 然后添加内容如下为:
node1
node2
node3
node4
```

#### 步骤 5 同步 Hadoop 的配置

拷贝 Hadoop 的 core-site.xml 配置文件至 hbase/conf 目录中:

```
> cp /home/modules/hadoop-2.8.3/etc/hadoop/core-site.xml
/home/modules/hbase-2.1.0/conf/
```

### 步骤 6 替换旧版依赖包

HBase 的 2.1.0 版本默认的 Jar 包是 Hadoop 2.7.4 版本，我们需要替换成 2.8.3 版本的：

```
> rm -rf /home/modules/hbase-2.1.0/lib/*  
> cp -r /home/extend_tools/hbase_lib/* /home/modules/hbase-2.1.0/lib/
```

### 步骤 7 分发组件

在 node1 中配置好 HBase 后，分发组件文件到 node2-4 节点中：

```
> for i in {2..4};do scp -r /home/modules/hbase-2.1.0  
root@node${i}:/home/modules/;done
```

拷贝完毕后，在 node2~node4 节点，均可查看到如下目录：

```
> ls /home/modules/ | grep hbase  
hbase-2.1.0
```

### 步骤 8 配置 HBase 的环境变量，执行下面命令：

```
> vim /etc/profile
```

在文件末尾添加如下的内容：

```
export HBASE_HOME=/home/modules/hbase-2.1.0  
export PATH=$HBASE_HOME/bin:$PATH
```

node1~node4 节点均配置好后，执行如下命令确保环境变量生效：

```
> source /etc/profile
```

## 3.2.3 启动并验证 HBase

### 步骤 1 启动 HBase

在 node1 节点，执行如下命令启动 hbase

```
> start-hbase.sh
```

### 步骤 2 查看 HBase 的 Java 进程

在 node1 节点执行 jps 命令，存在 HMaster、HRegionServer、HquorumPeer 进程：

```
[root@node1 ~]# jps  
8529 Jps  
7971 HRegionServer  
5507 ResourceManager  
7732 HQuorumPeer  
5096 NameNode  
7816 HMaster  
5321 SecondaryNameNode
```

```
1503 WrapperSimpleApp
```

在 node2 节点执行 jps 命令，存在 HRegionServer、HquorumPeer 进程：

```
[root@node2 ~]# jps
3830 HRegionServer
2664 NodeManager
1513 WrapperSimpleApp
3738 HQuorumPeer
2540 DataNode
4159 Jps
```

在 node3 节点执行 jps 命令，存在 HRegionServer、HquorumPeer 进程：

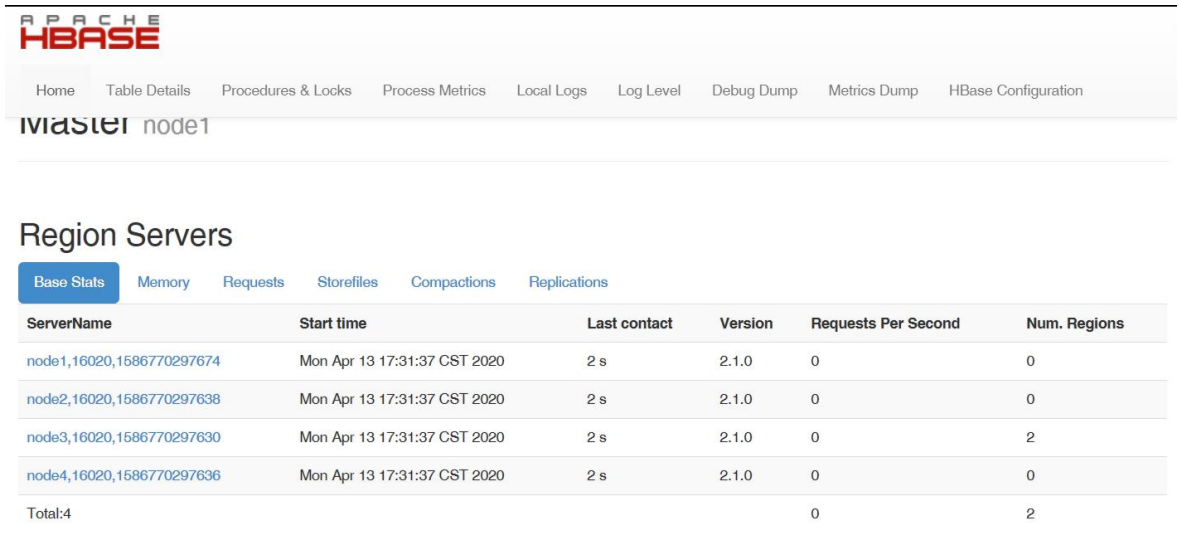
```
[root@node3 ~]# jps
3730 HQuorumPeer
2531 DataNode
3828 HRegionServer
1573 WrapperSimpleApp
4198 Jps
2655 NodeManager
```

在 node4 节点执行 jps 命令，存在 HRegionServer 进程：

```
[root@node4 ~]# jps
3684 Jps
2471 NodeManager
2347 DataNode
1516 WrapperSimpleApp
3356 HRegionServer
```

### 步骤 3 访问 HBaseWebUI 界面

访问路径为：[http://node1 的弹性公网 ip:16010](http://node1的弹性公网ip:16010)



ServerName	Start time	Last contact	Version	Requests Per Second	Num. Regions
node1,16020,1586770297674	Mon Apr 13 17:31:37 CST 2020	2 s	2.1.0	0	0
node2,16020,1586770297638	Mon Apr 13 17:31:37 CST 2020	2 s	2.1.0	0	0
node3,16020,1586770297630	Mon Apr 13 17:31:37 CST 2020	2 s	2.1.0	0	2
node4,16020,1586770297636	Mon Apr 13 17:31:37 CST 2020	2 s	2.1.0	0	0
Total:4				0	2

图 3-2 HBase 首页网址

## 3.3 实验任务 2

### 3.3.1 HBase 常用操作

#### 3.3.1.1 进入 HBase shell 客户端

步骤 1 在 node1 节点执行 HBase shell 进入客户端。

```
> hbase shell
.....
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 2.1.0, re1673bb0bbfea21d6e5dba73e013b09b8b49b89b, Tue Jul 10 17:26:48 CST 2018
Took 0.0030 seconds
hbase(main):001:0>
```

上述结果表明，已经成功进入 HBase shell 客户端。注意：以下命令为在 HBase shell 客户端输入命令。

#### 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'
Created table cga_info
Took 8.7301 seconds
=> Hbase::Table - cga_info
```

创建表 “cga\_info” 成功。

步骤 2 查看系统中共有多少个普通表：list。

```
> list
TABLE
cga_info
1 row(s)
Took 0.0277 seconds
=> ["cga_info"]
```

由此看出系统中已经存在了 1 个普通表。

### 3.3.1.3 创建 namespace

namespace 是一组表的逻辑划分，类似于数据库中的 DataBase 概念。

创建 namespace 的语法为：create\_namespace ‘名称’。

```
> create_namespace 'nn'
0 row(s)
Took 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)
Took 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)
Took 0.3681 seconds
=> ["student"]
```

### 3.3.1.6 查看表结构

```
hbase(main):025:0> describe 'cga_info'
```



```
Table cga_info is ENABLED
cga_info
COLUMN FAMILIES DESCRIPTION
{NAME => 'info', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false',
NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'false',
LS => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL
=> 'FOREVER', MIN_VERSIONS => '0',
REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false',
IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false',
N_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE',
BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
1 row(s)
Took 0.0735 seconds
```

### 3.3.1.7 查看 OBS 存储

步骤 1 登录华为公有云，选择 OBS 对象存储服务

步骤 2 在 OBS 控制台中选择要对接 Hadoop 的 OBS 桶

步骤 3 在右侧列表栏中选择“对象”，再打开“hbasetest001”文件夹对象

对象存储服务 / hbasetest001

对象 | 已删除对象 | 碎片

对象是数据存储在 OBS 中的基本单位，在 OBS 中文件和文件夹都是对象。您可以上传任何类型（文本、图片、视频等）的文件，并在桶中对这些文件进行管理。 [了解更多](#)

上传对象 | 新建文件夹 | 恢复 | 删除 | 修改存储类别

输入对象名前缀搜索 Q C

<input type="checkbox"/>	名称	存储类别	大小	加密状态	恢复状态	最后修改时间	操作
<input type="checkbox"/>	hbasetest001	-	-	-	-	-	分享   复制路径   更多
<input type="checkbox"/>	OBS_Test	-	-	-	-	-	分享   复制路径   更多

对象存储服务 / hbasetest001 / hbasetest001

对象 | 已删除对象 | 碎片

对象是数据存储在 OBS 中的基本单位，在 OBS 中文件和文件夹都是对象。您可以上传任何类型（文本、图片、视频等）的文件，并在桶中对这些文件进行管理。 [了解更多](#)

上传对象 | 新建文件夹 | 恢复 | 删除 | 修改存储类别

输入对象名前缀搜索 Q C

返回上一级

<input type="checkbox"/>	名称	存储类别	大小	加密状态	恢复状态	最后修改时间	操作
<input type="checkbox"/>	staging	-	-	-	-	-	分享   复制路径   更多
<input type="checkbox"/>	mobdir	-	-	-	-	-	分享   复制路径   更多
<input type="checkbox"/>	data	-	-	-	-	-	分享   复制路径   更多
<input type="checkbox"/>	archive	-	-	-	-	-	分享   复制路径   更多
<input type="checkbox"/>	tmp	-	-	-	-	-	分享   复制路径   更多
<input type="checkbox"/>	hbck	-	-	-	-	-	分享   复制路径   更多
<input type="checkbox"/>	hbase.id	标准存储	42 byte	未加密	-	2020/04/13 17:31:43 GM...	下载   分享   更多
<input type="checkbox"/>	hbase.version	标准存储	7 byte	未加密	-	2020/04/13 17:31:42 GM...	下载   分享   更多

**图 3-3 Hbase 底层数据文件**

可以查看到 HBase 的数据文件是存储在华为公有云 OBS 对象存储服务中的, 由 OBS 服务实现数据快速访问、安全管理等功能。

### 3.3.1.8 增加数据

增加数据: put '表的名称', 'RowKey', '列族的名称列的名称', '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)
Took 0.1580 seconds

> put 'cga_info','123001','info:gender','male'
0 row(s)
Took 0.0390 seconds

> put 'cga_info','123001','info:age','40'
0 row(s)
Took 0.0250 seconds

> put 'cga_info','123001','info:address','Los Angeles'
0 row(s)
Took 0.0170 seconds
```

### 3.3.1.9 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
1 row(s)
Took 0.0390 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=1586826576549, value=Kobe
1 row(s)
```

Took 0.0069 seconds

### 3.3.1.10 scan 方式查询数据

步骤 1 按照 3.3.1.6 的方法，在表中多输入多条不同的数据，要插入数据的内容参考步骤 2 查询结果：

```
> put 'cga_info','123002','info:name','Victoria'
> put 'cga_info','123002','info:gender','female'
> put 'cga_info','123002','info:age','40'
> put 'cga_info','123002','info:address','London'

> put 'cga_info','123003','info:name','Taylor'
> put 'cga_info','123003','info:gender','female'
> put 'cga_info','123003','info:age','30'
> put 'cga_info','123003','info:address','Redding'

> put 'cga_info','123004','info:name','LeBron'
> put 'cga_info','123004','info:gender','male'
> put 'cga_info','123004','info:age','33'
> put 'cga_info','123004','info:address','Cleveland'
```

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=>‘列族’}

```
> 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)
Took 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)
Took 0.0300 seconds
```

### 3.3.1.11 指定条件查询数据

步骤 1 查询 RowKey 从 “123002” 开始, 然后读两行的数据:

```
> 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)
Took 0.0170 seconds
```

步骤 2 查询 RowKey 从 “123001” 开始, 然后读取两行, 读取列名称为 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)
Took 0.0500 seconds
```

### 3.3.1.12 统计表数据行数

```
hbase(main):024:0> count 'cga_info'
```

```
3 row(s)
Took 0.0667 seconds
=> 4
```

### 3.3.1.13 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)
Took 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'
Took 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)
Took 0.0040 seconds
```

由步骤 1 和步骤 3 的结果比较可得，年龄信息已经被更新：

### 3.3.1.14 删除数据

#### 3.3.1.14.1 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
1 row(s)
Took 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:address'
Took 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:age         timestamp=1602767730052, value=18
info:gender      timestamp=1602767146188, value=male
info:name        timestamp=1602767141431, value=Kobe
1 row(s)
Took 0.0220 seconds
```

由步骤 1 和步骤 3 的结果比较可得，地址信息已经被删除。

#### 3.3.1.14.2 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'
Took 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)
Took 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。

#### 3.3.1.14.3 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'
Took 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'
Took 1.2270 seconds
> drop 'cga_info1'
Took 0.3940 seconds
```

注意：如果直接 drop 表，会报错：ERROR: Table student is enabled. Disable it first.

步骤 3 查询当前命名空间下的表：

```
> list
TABLE
```

```
cga_info
nn:student
2 row(s)
Took 0.0106 seconds
=> ["cga_info", "nn:student"]
```

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)
Took 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)
Took 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)
Took 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)
Took 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 变更表信息

#### 步骤 1 设置 VERSIONS

设置'cga\_info'表的 info 列族中的数据存放可以 3 个版本：

```
> alter 'cga_info',{NAME=>'info',VERSIONS=>3}
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 12.0646 seconds
```

#### 步骤 2 插入'cga\_info'表的 info 列族 "name" 列三行新数据

```
> put 'cga_info','123005','info:name','Zhangsan'
> put 'cga_info','123005','info:name','Lisi'
> put 'cga_info','123005','info:name','Wangwu'
```

#### 步骤 3 查询 cga\_info 表 info 列族 name 列 123005 行值的 3 个版本数据内容

```
> get 'cga_info','123005',{COLUMN => 'info:name', VERSIONS => 3}
COLUMN                                CELL
info:name                             timestamp=1586851215306, value=Wangwu
info:name                             timestamp=1586851214780, value=Lisi
info:name                             timestamp=1586851214761, value=Zhangsan
1 row(s)
Took 0.0101 seconds
```

### 3.3.4 创建预分 Region 表

#### 3.3.4.1 以 Rowkey 切分，随机分为 4 个 Region

##### 步骤 1 创建一个新的表 "cag\_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)
Took 0.3720 seconds
=> Hbase::Table - cga_info2
```

##### 步骤 2 查看 HBase WebUI 界面



1. 使用浏览器登录，地址为 `http://node1` 弹性公网 ip:16010。
2. 选择 Table Details
3. 找到所创建的新表 “cga\_info2”：

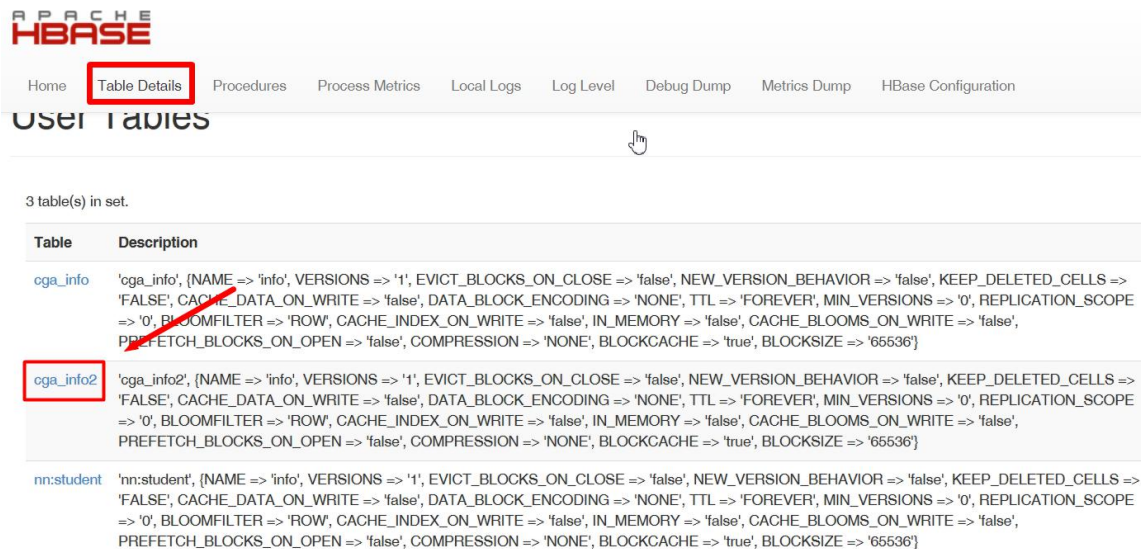


图 3-4 HBase 首页 - Table Details

步骤 3 在 Table Regions 中查询 region 的切分结果

勾选 “ShowDetailName&Start/End Key”，点击 “Reorder” 按钮

查看第一列 Name 的内容：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：

### Table Regions

Sort As RegionName Ascending ShowDetailName&Start/End Key Reorder

Name(4)	Region Server	ReadRequests (0)	WriteRequests (0)	StorefileSize (0 B)	Num. (0)
cga_info2,,1602768260715.e7979b6a05a77dba113f8e2c36ab142b.	node2:16030	0	0	0 B	0
cga_info2,@\x00\x00\x00\x00\x00\x00,1602768260715.ab78205ff799ae52b714eb4c2a61b69f.	node1:16030	0	0	0 B	0
cga_info2,\x80\x00\x00\x00\x00\x00\x00,1602768260715.39dd1cbdf29014b1fd89b448bb0f1ce.	node4:16030	0	0	0 B	0
cga_info2,\xC0\x00\x00\x00\x00\x00\x00,1602768260715.ce318bf35ca7b5c03404c93a8f2d8c84.	node3:16030	0	0	0 B	0

图 3-5 Table Regions

### 3.3.4.2 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', '列族名称', 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)
Took 0.6820 seconds
=> Hbase::Table - cga_info3
```

步骤 2 回到 HBase 首页-Table Details-Table Regions 界面。

### Table Regions

Sort As RegionName Ascending ☐ ShowDetailName&Start/End Key ☒ Reorder

Name(4)	Region Server	ReadRequests (0)	WriteRequests (0)	StorefileSize (0 B)	Num.Storefiles (0)	MemSize (0 B)	Locality	Start Key	End Key
cga_info3,,1602768604757.e8506cc293bd7ccc5f1bd9f5d9beaeec.	node3:16030	0	0	0 B	0	0 B	0.0		10000
cga_info3,10000,1602768604757.bcf4f82852327a270bf84a3a6e75e62c.	node1:16030	0	0	0 B	0	0 B	0.0	10000	20000
cga_info3,20000,1602768604757.4f41b74b331dae5778c7a23bee3d09e4.	node2:16030	0	0	0 B	0	0 B	0.0	20000	30000
cga_info3,30000,1602768604757.b68f4ad87e8fabe1ed151b9e77c9e072.	node4:16030	0	0	0 B	0	0 B	0.0	30000	

图 3-6 cga\_info3 的 Table Regions

实验结果表明表“cga\_info3”确实按照 StartKey 10000,20000,30000 被分为了 4 个 region。

## 3.4 实验小结

本实验主要讲述了鲲鹏 BigData Pro 解决方案下 HBase 组件的安装部署，表创建与删除、数据的增删改查等操作，同时介绍了 HBase 预分 Region 的使用方法。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。