

一、实验过程

1. HBase 单机模式运行

我选择 Docker Desktop 配合先前 Hadoop MapReduce 使用的容器来进行 HBase 实验。

首先在容器 h01 下载并解压 HBase 2.3.3，编辑/etc/profile 文件添加环境变量。

```
root@h01: /
export HBASE_HOME=/usr/local/hbase
export PATH=$PATH:$HBASE_HOME/bin
```

在容器 h01 的 hbase 目录下编辑 conf/hbase-env.sh，添加配置条目。

```
root@h01: /usr/local/hbase
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HBASE_MANAGES_ZK=true
```

对于单机模式，将 conf/hbase-site.xml 中的 hbase.rootdir 设置为本地目录。

```
root@h01: /usr/local/hbase
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>file:///usr/local/hbase/data</value>
  </property>
</configuration>
```

运行 bin/start-hbase.sh，再运行 bin/hbase shell。

```
root@h01: /usr/local/hbase
root@h01:/usr/local/hbase# bin/start-hbase.sh
running master, logging to /usr/local/hbase/bin/../logs/hbase--master-h01.out
root@h01:/usr/local/hbase# bin/hbase shell
2020-11-30 14:56:29,306 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.3.3, r3e4bf4bee3a08b25591b9c22fea0518686a7e834, Wed Oct 28 06:36:25 UTC 2020
Took 0.0005 seconds
hbase(main):001:0>
```

可见 shell 正常运行，HBase 单机模式配置正确。

2. HBase 伪分布模式运行

在上述基础上，对于伪分布模式，将 conf/hbase-site.xml 中的 hbase.rootdir 设置为 hdfs 目录，并且将 hbase.cluster.distributed 设为 true。

```
root@h01: /usr/local/hbase
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://h01:9000/hbase</value>
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
  </property>
</configuration>
```

运行 bin/start-hbase.sh, 再运行 bin/hbase shell。

```
root@h01: /usr/local/hbase
root@h01:/usr/local/hbase# bin/start-hbase.sh
127.0.0.1: running zookeeper, logging to /usr/local/hbase/bin/../logs/hbase-root-zookeeper-h01.o
ut
running master, logging to /usr/local/hbase/bin/../logs/hbase--master-h01.out
: running regionserver, logging to /usr/local/hbase/bin/../logs/hbase--regionserver-h01.out
root@h01:/usr/local/hbase# bin/hbase shell
2020-11-30 15:48:29,020 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.3.3, r3e4bf4bee3a08b25591b9c22fea0518686a7e834, Wed Oct 28 06:36:25 UTC 2020
Took 0.0006 seconds
hbase(main):001:0>
```

可见 shell 正常运行, HBase 伪分布模式配置正确。

3. HBase 集群模式运行

对于集群模式, 将 conf/hbase-site.xml 中的 hbase.rootdir 设置为 hdfs 目录, 同时
将其他参数如下设置。

```
root@h01: /usr/local/hbase
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://h01:9000/hbase</value>
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
  </property>
  <property>
    <name>hbase.master</name>
    <value>h01:60000</value>
  </property>
  <property>
    <name>hbase.zookeeper.quorum</name>
    <value>h01, h02, h03, h04, h05</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/home/hadoop/zoodata</value>
  </property>
</configuration>
64, 1 Bot
```

编辑 conf/regionservers 文件, 修改为如下内容。

```
root@h01: /usr/local/hbase
h01
h02
h03
h04
h05
5, 1 All
```

使用 scp 命令将配置好的 hbase 目录从容器 h01 复制到 h02 至 h04 的相同位置。
然后运行 bin/start-hbase.sh, 再运行 bin/hbase shell。

```
root@h01: /usr/local/hbase
root@h01:/usr/local/hbase# bin/start-hbase.sh
h02: running zookeeper, logging to /usr/local/hbase/bin/../logs/hbase-root-zookeeper-h02.out
h03: running zookeeper, logging to /usr/local/hbase/bin/../logs/hbase-root-zookeeper-h03.out
h05: running zookeeper, logging to /usr/local/hbase/bin/../logs/hbase-root-zookeeper-h05.out
h01: running zookeeper, logging to /usr/local/hbase/bin/../logs/hbase-root-zookeeper-h01.out
h04: running zookeeper, logging to /usr/local/hbase/bin/../logs/hbase-root-zookeeper-h04.out
running master, logging to /usr/local/hbase/bin/../logs/hbase--master-h01.out
h01: running regionserver, logging to /usr/local/hbase/bin/../logs/hbase-root-regionserver-h01.o
ut
h02: running regionserver, logging to /usr/local/hbase/bin/../logs/hbase-root-regionserver-h02.o
ut
h03: running regionserver, logging to /usr/local/hbase/bin/../logs/hbase-root-regionserver-h03.o
ut
h04: running regionserver, logging to /usr/local/hbase/bin/../logs/hbase-root-regionserver-h04.o
ut
h05: running regionserver, logging to /usr/local/hbase/bin/../logs/hbase-root-regionserver-h05.o
ut
root@h01:/usr/local/hbase# bin/hbase shell
2020-11-30 15:35:30,135 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library
for your platform.. using builtin-java classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.3.3. r3e4bf4bee3a08b25591b9c22fea0518686a7e834, Wed Oct 28 06:36:25 UTC 2020
Took 0.0007 seconds
hbase(main):001:0>
```

可见 shell 正常运行，HBase 集群模式配置正确。

4. Java 程序编写

由于任务较为简单，所有程序逻辑都在 Main 类的 main 函数中实现。

首先初始化 configuration、connection 和 admin，连接上 HBase。

```
1. configuration = HBaseConfiguration.create();
2. configuration.set("hbase.rootdir", "hdfs://h01:9000/hbase");
3. connection = ConnectionFactory.createConnection(configuration);
4. admin = connection.getAdmin();
```

通过构造 TableDescriptorBuilder 对象，并且调用 admin.createTable()来创建具有 4 个列族的表 Students。

```
1. TableDescriptorBuilder tableDescriptorBuilder = TableDescriptorBuilder.newBu
ilder(TableName.valueOf("Students"));
2. Collection<ColumnFamilyDescriptor> columnFamilies = new ArrayList();
3. columnFamilies.add(ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("I
D")).build());
4. columnFamilies.add(ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("D
escription")).build());
5. columnFamilies.add(ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("C
ourses")).build());
6. columnFamilies.add(ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("H
ome")).build());
7. tableDescriptorBuilder.setColumnFamilies(columnFamilies);
8. admin.createTable(tableDescriptorBuilder.build());
```

通过 connection.getTable()来获取 Table 对象。通过构造 Put 对象，并且调用

table.put()来新增表中条目。

```
1. Table table = connection.getTable(TableName.valueOf("Students"));
2.
3. Put put = new Put("001".getBytes());
4. put.addColumn("Description".getBytes(), "Name".getBytes(), "Li Lei".getBytes());
5. put.addColumn("Description".getBytes(), "Height".getBytes(), "176".getBytes());
6. put.addColumn("Courses".getBytes(), "Chinese".getBytes(), "80".getBytes());
7. put.addColumn("Courses".getBytes(), "Math".getBytes(), "90".getBytes());
8. put.addColumn("Courses".getBytes(), "Physics".getBytes(), "95".getBytes());
9. put.addColumn("Home".getBytes(), "Province".getBytes(), "Zhejiang".getBytes());
10. table.put(put);
11.
12. put = new Put("002".getBytes());
13. put.addColumn("Description".getBytes(), "Name".getBytes(), "Han Meimei".getBytes());
14. put.addColumn("Description".getBytes(), "Height".getBytes(), "183".getBytes());
15. put.addColumn("Courses".getBytes(), "Chinese".getBytes(), "88".getBytes());
16. put.addColumn("Courses".getBytes(), "Math".getBytes(), "77".getBytes());
17. put.addColumn("Courses".getBytes(), "Physics".getBytes(), "66".getBytes());
18. put.addColumn("Home".getBytes(), "Province".getBytes(), "Beijing".getBytes());
19. table.put(put);
20.
21. put = new Put("003".getBytes());
22. put.addColumn("Description".getBytes(), "Name".getBytes(), "Xiao Ming".getBytes());
23. put.addColumn("Description".getBytes(), "Height".getBytes(), "162".getBytes());
24. put.addColumn("Courses".getBytes(), "Chinese".getBytes(), "90".getBytes());
25. put.addColumn("Courses".getBytes(), "Math".getBytes(), "90".getBytes());
26. put.addColumn("Courses".getBytes(), "Physics".getBytes(), "90".getBytes());
27. put.addColumn("Home".getBytes(), "Province".getBytes(), "Shanghai".getBytes());
28. table.put(put);
```

通过 scan 对象和 table.getScanner()获得扫描结果，遍历扫描结果并输出。

```
1. Scan scan = new Scan();
2. ResultScanner results = table.getScanner(scan);
3.
4. for (Result row : results) {
5.     for (Cell cell : row.listCells()){
6.         System.out.println("RowKey:" + Bytes.toString(row.getRow())
7.             + " Family:" + Bytes.toString(CellUtil.cloneFamily(cell))
8.             + " Qualifier:" + Bytes.toString(CellUtil.cloneQualifier(cell))
9.             + " Value:" + Bytes.toString(CellUtil.cloneValue(cell))
10.        );
11.    }
12. }
```

通过构造 Get 对象，并且调用 table.get()获得查询结果，并输出。

```
1. Get get = new Get("001".getBytes());
2. get.addColumn("Home".getBytes(), "Province".getBytes());
3. Result result = table.get(get);
4. System.out.println("001's home is in "+new String(result.getValue("Home".getBytes(), "Province".getBytes()))+" Province.");
```

再次通过构造 Put 对象，并且调用 table.put()来新增表中条目，同时也新增了列。

```
1. put = new Put("001".getBytes());
2. put.addColumn("Courses".getBytes(), "English".getBytes(), "99".getBytes());
3. table.put(put);
```

为了新增列族，需要首先调用 disableTable()，然后通过 addColumnFamilyAsync()新增列族，最后调用 enableTableAsync()。

```
1. admin.disableTable(TableName.valueOf("Students"));
2. admin.addColumnFamilyAsync(TableName.valueOf("Students"), ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("Contact")).build());
3. admin.enableTableAsync(TableName.valueOf("Students"));
```

再次通过构造 Put 对象，并且调用 table.put()来新增表中条目。再扫描一次表查看修改结果。

```
1. put = new Put("001".getBytes());
2. put.addColumn("Contact".getBytes(), "Email".getBytes(), "001@hbase.com".getBytes());
```

```

3. table.put(put);
4.
5. scan = new Scan();
6. results = table.getScanner(scan);
7.
8. for (Result row : results) {
9.     for (Cell cell : row.listCells()){
10.         System.out.println("RowKey:" + Bytes.toString(row.getRow())
11.                             + " Family:" + Bytes.toString(CellUtil.cloneFamily(cell))
12.                             + " Qualifier:" + Bytes.toString(CellUtil.cloneQualifier(cell))
13.                             + " Value:" + Bytes.toString(CellUtil.cloneValue(cell))
14.                             );
15.     }
16. }

```

最后在 disableTable()后调用 deleteTable()来删除表，并且关闭相关连接。

```

1. table.close();
2.
3. admin.disableTable(TableName.valueOf("Students"));
4. admin.deleteTable(TableName.valueOf("Students"));
5.
6. admin.close();
7. connection.close();

```

将 Java 程序使用 Gradle 打包后在 Hadoop 运行。首先出现 ClassNotDefError，在 hadoop/etc/hadoop/hadoop-env.sh 文件中的 HADOOP_CLASSPATH 中增加 hbase/lib/* 即可。在程序运行前，需要同时打开 Hadoop 和 HBase，即分别执行 hadoop/sbin/start-all.sh 和 hbase/bin/start-hbase.sh，否则在程序执行过程中会提示 Connection refused，即相关进程连接不上。

在执行 hbase/bin/start-hbase.sh 时，出现了 HMaster 进程已存在的问题。经过几次尝试，发现即便重启容器 h01，HMaster 进程仍会自动运行，因此先使用 kill -9 指令杀掉 HMaster 进程，而后再执行 hbase/bin/start-hbase.sh，不再有错误提示。

最后在 Hadoop 中执行打包好的程序，输出如下。

```
root@h01: /usr/local/hadoop
2020-12-01 07:44:52,290 INFO client.HBaseAdmin: Operation: CREATE, Table Name: default:Students, procId: 9 completed
RowKey:001 Family:Courses Qualifier:Chinese Value:80
RowKey:001 Family:Courses Qualifier:Math Value:90
RowKey:001 Family:Courses Qualifier:Physics Value:95
RowKey:001 Family:Description Qualifier:Height Value:176
RowKey:001 Family:Description Qualifier:Name Value:Li Lei
RowKey:001 Family:Home Qualifier:Province Value:Zhejiang
RowKey:002 Family:Courses Qualifier:Chinese Value:88
RowKey:002 Family:Courses Qualifier:Math Value:77
RowKey:002 Family:Courses Qualifier:Physics Value:66
RowKey:002 Family:Description Qualifier:Height Value:183
RowKey:002 Family:Description Qualifier:Name Value:Han Meimei
RowKey:002 Family:Home Qualifier:Province Value:Beijing
RowKey:003 Family:Courses Qualifier:Chinese Value:90
RowKey:003 Family:Courses Qualifier:Math Value:90
RowKey:003 Family:Courses Qualifier:Physics Value:90
RowKey:003 Family:Description Qualifier:Height Value:162
RowKey:003 Family:Description Qualifier:Name Value:Xiao Ming
RowKey:003 Family:Home Qualifier:Province Value:Shanghai
001's home is in Zhejiang Province.
2020-12-01 07:44:52,525 INFO client.HBaseAdmin: Started disable of Students
2020-12-01 07:44:56,670 INFO client.HBaseAdmin: Operation: DISABLE, Table Name: default:Students, procId: 12 completed
2020-12-01 07:44:57,243 INFO client.HBaseAdmin: Started enable of Students
RowKey:001 Family:Contact Qualifier:Email Value:001@hbase.com
RowKey:001 Family:Courses Qualifier:Chinese Value:80
RowKey:001 Family:Courses Qualifier:English Value:99
RowKey:001 Family:Courses Qualifier:Math Value:90
RowKey:001 Family:Courses Qualifier:Physics Value:95
RowKey:001 Family:Description Qualifier:Height Value:176
RowKey:001 Family:Description Qualifier:Name Value:Li Lei
RowKey:001 Family:Home Qualifier:Province Value:Zhejiang
RowKey:002 Family:Courses Qualifier:Chinese Value:88
RowKey:002 Family:Courses Qualifier:Math Value:77
RowKey:002 Family:Courses Qualifier:Physics Value:66
RowKey:002 Family:Description Qualifier:Height Value:183
RowKey:002 Family:Description Qualifier:Name Value:Han Meimei
RowKey:002 Family:Home Qualifier:Province Value:Beijing
RowKey:003 Family:Courses Qualifier:Chinese Value:90
RowKey:003 Family:Courses Qualifier:Math Value:90
RowKey:003 Family:Courses Qualifier:Physics Value:90
RowKey:003 Family:Description Qualifier:Height Value:162
RowKey:003 Family:Description Qualifier:Name Value:Xiao Ming
RowKey:003 Family:Home Qualifier:Province Value:Shanghai
2020-12-01 07:44:58,826 INFO client.HBaseAdmin: Started disable of Students
2020-12-01 07:45:00,958 INFO client.HBaseAdmin: Operation: DISABLE, Table Name: default:Students, procId: 19 completed
2020-12-01 07:45:03,141 INFO client.HBaseAdmin: Operation: DELETE, Table Name: default:Students, procId: 22 completed
2020-12-01 07:45:03,142 INFO client.ConnectionImplementation: Closing master protocol: MasterService
root@h01: /usr/local/hadoop#
```

可见表 Students 创建、扫描成功，ID 为 001 的学生所在省份查询成功，观察第 2 次扫描结果，可见列族、列增加成功，最终表 Students 删除成功。

5. Shell 指令运行

使用 Shell 指令完成与 Java 程序相同的任务。

```

root@h01: /usr/local/hbase
hbase(main):001:0> create 'Students', 'ID', 'Description', 'Courses', 'Home'
Created table Students
Took 4.5997 seconds
=> Hbase::Table - Students
hbase(main):002:0> put 'Students', '001', 'Description:Name', 'Li Lei'
Took 0.1434 seconds
hbase(main):003:0> put 'Students', '001', 'Description:Height', '176'
Took 0.0440 seconds
hbase(main):004:0> put 'Students', '001', 'Courses:Chinese', '80'
Took 0.0047 seconds
hbase(main):005:0> put 'Students', '001', 'Courses:Math', '90'
Took 0.0042 seconds
hbase(main):006:0> put 'Students', '001', 'Courses:Physics', '95'
Took 0.0042 seconds
hbase(main):007:0> put 'Students', '001', 'Home:Province', 'Zhejiang'
Took 0.0042 seconds
hbase(main):008:0> put 'Students', '002', 'Description:Name', 'Han Meimei'
Took 0.0041 seconds
hbase(main):009:0> put 'Students', '002', 'Description:Height', '183'
Took 0.0052 seconds
hbase(main):010:0> put 'Students', '002', 'Courses:Chinese', '88'
Took 0.0043 seconds
hbase(main):011:0> put 'Students', '002', 'Courses:Math', '77'
Took 0.0039 seconds
hbase(main):012:0> put 'Students', '002', 'Courses:Physics', '66'
Took 0.0036 seconds
hbase(main):013:0> put 'Students', '002', 'Home:Province', 'Beijing'
Took 0.0048 seconds
hbase(main):014:0> put 'Students', '003', 'Description:Name', 'Xiao Ming'
Took 0.0043 seconds
hbase(main):015:0> put 'Students', '003', 'Description:Height', '162'
Took 0.0038 seconds
hbase(main):016:0> put 'Students', '003', 'Courses:Chinese', '90'
Took 0.0051 seconds
hbase(main):017:0> put 'Students', '003', 'Courses:Math', '90'
Took 0.0044 seconds
hbase(main):018:0> put 'Students', '003', 'Courses:Physics', '90'
Took 0.0036 seconds
hbase(main):019:0> put 'Students', '003', 'Home:Province', 'Shanghai'
Took 0.0038 seconds
hbase(main):020:0> scan 'Students'
ROW                                COLUMN+CELL
001                                column=Courses:Chinese, timestamp=2020-12-01T08:44:16.982, value=80
001                                column=Courses:Math, timestamp=2020-12-01T08:44:26.486, value=90
001                                column=Courses:Physics, timestamp=2020-12-01T08:44:38.046, value=95
001                                column=Description:Height, timestamp=2020-12-01T08:43:55.422, value=176
001                                column=Description:Name, timestamp=2020-12-01T08:43:42.824, value=Li Lei
001                                column=Home:Province, timestamp=2020-12-01T08:44:58.531, value=Zhejiang
002                                column=Courses:Chinese, timestamp=2020-12-01T08:45:36.451, value=88
002                                column=Courses:Math, timestamp=2020-12-01T08:45:46.082, value=77
002                                column=Courses:Physics, timestamp=2020-12-01T08:45:57.570, value=66
002                                column=Description:Height, timestamp=2020-12-01T08:45:27.884, value=183
002                                column=Description:Name, timestamp=2020-12-01T08:45:12.245, value=Han Meimei
002                                column=Home:Province, timestamp=2020-12-01T08:46:08.704, value=Beijing
003                                column=Courses:Chinese, timestamp=2020-12-01T08:46:45.557, value=90
003                                column=Courses:Math, timestamp=2020-12-01T08:46:53.707, value=90
003                                column=Courses:Physics, timestamp=2020-12-01T08:47:11.595, value=90
003                                column=Description:Height, timestamp=2020-12-01T08:46:34.778, value=162
003                                column=Description:Name, timestamp=2020-12-01T08:46:24.589, value=Xiao Ming
003                                column=Home:Province, timestamp=2020-12-01T08:47:26.420, value=Shanghai
3 row(s)
Took 0.0710 seconds
hbase(main):021:0> get 'Students', '001', 'Home:Province'
COLUMN                                CELL
Home:Province                        timestamp=2020-12-01T08:44:58.531, value=Zhejiang
1 row(s)
Took 0.0234 seconds

```



```
root@h01: /usr/local/hbase
hbase(main):022:0> put 'Students','001','Courses:English','99'
Took 0.0043 seconds
hbase(main):023:0> alter 'Students','Contact'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 3.6722 seconds
hbase(main):024:0> put 'Students','001','Contact:Email','001@hbase.com'
Took 0.0046 seconds
hbase(main):025:0> get 'Students','001'
COLUMN                                CELL
Contact:Email                         timestamp=2020-12-01T08:50:24.371, value=001@hbase.com
Courses:Chinese                       timestamp=2020-12-01T08:44:16.982, value=80
Courses:English                       timestamp=2020-12-01T08:49:05.748, value=99
Courses:Math                          timestamp=2020-12-01T08:44:26.486, value=90
Courses:Physics                       timestamp=2020-12-01T08:44:38.046, value=95
Description:Height                    timestamp=2020-12-01T08:43:55.422, value=176
Description:Name                      timestamp=2020-12-01T08:43:42.824, value=Li Lei
Home:Province                         timestamp=2020-12-01T08:44:58.531, value=Zhejiang
1 row(s)
Took 0.0254 seconds
hbase(main):026:0> dis
disable                                disable_all                                disable_exceed_throttle_quota
disable_peer                           disable_rpc_throttle                        disable_table_replication
display
hbase(main):026:0> disable 'Students'
Took 1.1431 seconds
hbase(main):027:0> drop 'Students'
Took 2.1432 seconds
hbase(main):028:0> list
TABLE
0 row(s)
Took 0.0121 seconds
=> []
hbase(main):029:0>
```

可见运行结果均正确。

二、问题总结、解决方案及其他思考

实验过程中的所有问题及其解决方案已在实验过程中叙述，此处不再重复。由于先前实验中 Hadoop 集群环境已在本地 Docker Desktop 下配置好，本次实验中 HBase 环境的配置较为容易，没有遇到太多问题。

由于 HBase Shell 指令较为直观简洁，因此使用 Shell 指令来调用 HBase 功能也较为容易。本次实验的主要难点集中在 Java 程序中 HBase 的使用。观察 Java 代码，可见在 Java 程序中，HBase 的每个单个操作几乎都需要先新建一个对象，然后调用该对象的函数对其增加参数，然后再以该对象为参数调用表对象的函数来执行操作，非常繁琐复杂。如果 HBase Shell 指令能够像 SQL 指令一样在 Java 程序中内联，应当能有效降低程序编写和阅读的复杂度。