

实验3

HBase安装

HBase下载

<https://mirrors.nju.edu.cn/apache/hbase/2.4.8/hbase-2.4.8-bin.tar.gz>有问题，查看<https://mirrors.nju.edu.cn/apache/hbase/>后发现现在只有如下版本，选择2.5.5版本

Directory: /apache/hbase/

File Name ↓	File Size ↓	Date ↓
Parent directory/	-	-
hbase-connectors-1.0.1/	-	2023-10-28 00:39:16
2.5.6/	-	2023-10-21 02:12:56
hbase-thirdparty-4.1.5/	-	2023-10-06 05:55:23
2.5.5/	-	2023-06-13 19:52:04
stable/	-	2023-06-13 19:52:04
3.0.0-alpha-4/	-	2023-06-07 14:15:01
2.4.17/	-	2023-04-06 20:26:51
hbase-connectors-1.0.0/	-	2022-06-17 12:27:19
hbase-filessystem-1.0.0-alpha1/	-	2022-06-17 12:27:19
hbase-operator-tools-1.2.0/	-	2022-06-17 12:26:52
HEADER.html	1067	2023-05-09 02:08:30

```
sudo wget https://mirrors.nju.edu.cn/apache/hbase/2.5.5/hbase-2.5.5-hadoop3-bin.tar.gz
```

```

[hadoop@localhost local]$ sudo wget https://mirrors.nju.edu.cn/apache/hbase/2.4.17/hbase-2.4.17-bin.tar.gz
[sudo] hadoop 的密码:
--2023-11-24 17:14:11-- https://mirrors.nju.edu.cn/apache/hbase/2.4.17/hbase-2.4.17-bin.tar.gz
正在解析主机 mirrors.nju.edu.cn (mirrors.nju.edu.cn)... 210.28.130.3, 2001:da8:1007:4011::3
正在连接 mirrors.nju.edu.cn (mirrors.nju.edu.cn)|210.28.130.3|:443... 已连接。
已发出 HTTP 请求，正在等待响应... 200 OK
长度: 284858851 (272M) [application/octet-stream]
正在保存至: hbase-2.4.17-bin.tar.gz

100%[=====>] 284,858,851 34.3MB/s 用时 7.8s

2023-11-24 17:14:19 (34.8 MB/s) - 已保存 hbase-2.4.17-bin.tar.gz [284858851/284858851]

[hadoop@localhost local]$ ls
apache-maven-3.9.5  hadoop  lib  share
bin  hbase-2.4.17-bin.tar.gz  lib64  src
etc  idea-IU-232.10072.27  libexec
games  include  sbin
[hadoop@localhost local]$

```

解压: `sudo tar xzvf hbase-2.5.5-hadoop3-bin.tar.gz`

`xzvf` 参数是解压tar.gz文件的参数

更改名称 (hbase-2.5.5->hbase)

```

[hadoop@localhost local]$ ls
apache-maven-3.9.5  bin  etc  games  hadoop  hbase  hbase-2.4.17-bin.tar.gz  hbase-2.5.5-hadoop3-bin.tar.gz  idea-IU-232.10072.27  include  lib  lib64  libexec  sbin  share  src
[hadoop@localhost local]$

```

修改 hbase-env.sh

`vim hbase-env.sh`

添加如下内容:

```

1  #本机的java路径
2  export JAVA_HOME=/usr/lib/java-1.8.0/jdk1.8.0_381
3  #HBase 将管理自己的 ZooKeeper 实例
4  export HBASE_MANAGES_ZK=true
5  #为了避免 HBase 在启动时查找 Hadoop 的类路径，这一行很重要!!
6  export HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP="true"

```

修改 hbase-site.xml

`vim hbase-site.xml`

文件配置修改如下:

```

1  <!--修改成true 分布式-->
2  <property>

```

```

3      <name>hbase.cluster.distributed</name>
4      <value>true</value>
5  </property>
6
7  <property>
8      <name>hbase.tmp.dir</name>
9      <value>/usr/local/hbase/data/tmp</value>
10 </property>
11
12 <property>
13     <name>hbase.unsafe.stream.capability.enforce</name>
14     <value>false</value>
15 </property>
16
17 <!--HBase的数据保存在HDFS对应目录-->
18 <property>
19     <name>hbase.rootdir</name>
20     <value>hdfs://localhost:9000/hbase</value>
21 </property>
22
23 <!--配置ZK的地址,有5个节点启用ZooKeeper-->
24 <property>
25     <name>hbase.zookeeper.quorum</name>
26     <value>localhost:2181</value>
27 </property>
28
29 <!--主节点和从节点允许的最大时间误差-->
30 <property>
31     <name>hbase.master.maxclockskew</name>
32     <value>180000</value>
33 </property>
34
35 <!--zookeeper数据目录-->
36 <property>
37     <name>hbase.zookeeper.property.dataDir</name>
38     <value>/usr/local/hbase/data/zookeeper</value>
39 </property>

```

伪分布式运行

1. 先启动 `hadoop`

2. 启动 `Hbase`

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

```

hadoop@localhost ~]$ start-hadoop
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [localhost.localdomain]
Starting resourcemanager
Starting nodemanagers
[hadoop@localhost ~]$ which hbase
/usr/local/hbase/bin/hbase
[hadoop@localhost ~]$ cd /usr/local/hbase/bin/
[hadoop@localhost bin]$ ./start-hbase.sh
localhost: running zookeeper, logging to /usr/local/hbase/bin/../logs/hbase-hadoop-zookeeper-localhost.localdomain.out
running master, logging to /usr/local/hbase/bin/../logs/hbase-hadoop-master-localhost.localdomain.out
running regionserver, logging to /usr/local/hbase/bin/../logs/hbase-hadoop-regionserver-localhost.localdomain.out
[hadoop@localhost bin]$

```

```

[hadoop@localhost hbase]$ jps
9888 HMaster
9090 NodeManager
8451 DataNode
9763 HQuorumPeer
8309 NameNode
8709 SecondaryNameNode
10022 HRegionServer
8954 ResourceManager
11082 Jps

```

Home

Guide Details

Procedures & Links

HCRC Report

Operation Details

Process Metrics

Local Logs

Log Level

Debug Dump

Metrics Dump

Profiler

HBase Configuration

Startup Progress

Master localhost

Region Servers

Reset Filters

Memory

Requests

Statistics

Comparisons

Replications

ServerName	Start time	Last contact	Version	Requests Per Second	Num. Regions
localhost:16030/1700880105333	Sat Nov 25 11:44:55 CST 2023	0 s	0.8.6-hd3000	0	3
Total				0	3

Backup Masters

ServerName	Port	Start Time
Total		

Tables

Open Tables

System Tables

Snapshots

TABLES is set (details). Click count below to see list of regions currently in 'state' designated by the column title. For 'Other' region state, browse to [HBase meta](#) and adjust filter on 'Meta' column to query on states other than those listed here. Queries may take a while if the table/meta table is large.

Namespace	Name	State	Regions								Description
			OPEN	OPENING	CLOSED	CLOSING	OFFLINE	SPLIT	Other		
default	test	FINISHED	1	0	0	0	0	0	0	1	test: TABLE FATTYFILTER => (METAINFO => {Hbase store file broken? => OFFLINE?}) [SUM => 1]

3. 关闭 HBase

```
./bin/stop-hbase.sh
```

实验内容

Task1: 设计并创建合适的表

1. 进入shell

```

[hadoop@localhost hbase]$ hbase shell
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 2.5.5- hadoop3, r7ebd4381261fef78fc2acf258a95184f4147cee, Thu Jun 1 17:59:44 PDT 2023
Took 0.0018 seconds
hbase:001:0>

```

2. 学生表的构建

```

hbase:006:0> create 'student', 'info'
Created table student
Took 2.2119 seconds
=> Hbase:Table - student
hbase:007:0> put 'student', '2120001', 'info:S_No', '2120001'
Took 0.2251 seconds
hbase:008:0> put 'student', '2120001', 'info:S_Name', 'Li Lei'
Took 0.0145 seconds
hbase:009:0> put 'student', '2120001', 'info:S_Sex', 'male'
Took 0.0139 seconds
hbase:010:0> put 'student', '2120001', 'info:S_Age', '20'
Took 0.0095 seconds
hbase:011:0> scan 'student'
ROW
2120001
2120001
2120001
2120001
row(s)

COLUMN+CELL
column=info:S_Age, timestamp=2023-11-25T12:37:11.060, value=20
column=info:S_Name, timestamp=2023-11-25T12:36:56.756, value=Li Lei
column=info:S_No, timestamp=2023-11-25T12:36:40.139, value=2120001
column=info:S_Sex, timestamp=2023-11-25T12:37:03.998, value=male

```

- 创建表 `create '<table_name>', '<列族名>'`
- 添加一行 `put <table>, <rowkey>, <family:column>, <value>`
- 查看表: `scan '<table_name>'`

```

hbase:023:0> scan 'student'
ROW
2120001
2120001
2120001
2120001
2120002
2120002
2120002
2120002
2120003
2120003
2120003
2120003
2120004
2120004
2120004
2120004
row(s)

COLUMN+CELL
column=info:S_Age, timestamp=2023-11-25T12:37:11.060, value=20
column=info:S_Name, timestamp=2023-11-25T12:36:56.756, value=Li Lei
column=info:S_No, timestamp=2023-11-25T12:36:40.139, value=2120001
column=info:S_Sex, timestamp=2023-11-25T12:37:03.998, value=male
column=info:S_Age, timestamp=2023-11-25T12:44:36.485, value=21
column=info:S_Name, timestamp=2023-11-25T12:44:36.467, value=Han Meimei
column=info:S_No, timestamp=2023-11-25T12:44:36.445, value=2120002
column=info:S_Sex, timestamp=2023-11-25T12:44:36.476, value=female
column=info:S_Age, timestamp=2023-11-25T12:45:00.630, value=20
column=info:S_Name, timestamp=2023-11-25T12:45:00.595, value=Zhang Li
column=info:S_No, timestamp=2023-11-25T12:45:00.572, value=2120003
column=info:S_Sex, timestamp=2023-11-25T12:45:00.617, value=female
column=info:S_Age, timestamp=2023-11-25T12:45:38.830, value=9
column=info:S_Name, timestamp=2023-11-25T12:45:38.815, value=Li Ming
column=info:S_No, timestamp=2023-11-25T12:45:38.796, value=2120004
column=info:S_Sex, timestamp=2023-11-25T12:45:38.822, value=male

```

3. 创建课程表

与创建学生表类似

```

hbase:023:0> scan 'course'
ROW
220001
220001
220001
220002
220002
220002
220003
220003
220003
220003
row(s)

COLUMN+CELL
column=course_info:C_Credit, timestamp=2023-11-25T13:32:19.382, value=4.0
column=course_info:C_Name, timestamp=2023-11-25T13:32:19.367, value=Math
column=course_info:C_No, timestamp=2023-11-25T13:32:19.330, value=220001
column=course_info:C_Credit, timestamp=2023-11-25T13:32:19.431, value=2.0
column=course_info:C_Name, timestamp=2023-11-25T13:32:19.417, value=English
column=course_info:C_No, timestamp=2023-11-25T13:32:19.400, value=220002
column=course_info:C_Credit, timestamp=2023-11-25T13:32:20.744, value=4.0
column=course_info:C_Name, timestamp=2023-11-25T13:32:19.473, value=BigData
column=course_info:C_No, timestamp=2023-11-25T13:32:19.459, value=220003

```

4. 创建选课表

注意，这里第一列不是唯一的，因此不能作为rowkey，这里我用的是 `<SC_Sno>_<SC_Cno>` 保证rowkey的唯一性

```
hbase:125:0> scan 'score'
ROW
2120001_220001
2120001_220001
2120001_220001
2120001_220002
2120001_220002
2120001_220002
2120001_220003
2120001_220003
2120002_220001
2120002_220001
2120002_220001
2120002_220002
2120002_220002
2120002_220002
2120003_220001
2120003_220001
2120003_220001
2120003_220002
2120003_220002
2120003_220002
2120003_220003
2120003_220003
2120003_220003
2120004_220001
2120004_220001
2120004_220001
2120004_220003
2120004_220003
2120004_220003

COLUMN+CELL
column=score_info:SC_Cno,timestamp=2023-11-25T13:29:12.175,value=220001
column=score_info:SC_Score,timestamp=2023-11-25T13:29:12.191,value=90
column=score_info:SC_Sno,timestamp=2023-11-25T13:29:12.158,value=2120001
column=score_info:SC_Cno,timestamp=2023-11-25T13:29:12.233,value=220002
column=score_info:SC_Score,timestamp=2023-11-25T13:29:12.252,value=85
column=score_info:SC_Sno,timestamp=2023-11-25T13:29:12.215,value=2120001
column=score_info:SC_Cno,timestamp=2023-11-25T13:29:12.288,value=220003
column=score_info:SC_Score,timestamp=2023-11-25T13:29:12.307,value=75
column=score_info:SC_Sno,timestamp=2023-11-25T13:29:12.272,value=2120001
column=score_info:SC_Cno,timestamp=2023-11-25T13:29:12.343,value=220001
column=score_info:SC_Score,timestamp=2023-11-25T13:29:12.357,value=85
column=score_info:SC_Sno,timestamp=2023-11-25T13:29:12.327,value=2120002
column=score_info:SC_Cno,timestamp=2023-11-25T13:29:12.400,value=220002
column=score_info:SC_Score,timestamp=2023-11-25T13:29:12.417,value=88
column=score_info:SC_Sno,timestamp=2023-11-25T13:29:12.382,value=2120002
column=score_info:SC_Cno,timestamp=2023-11-25T13:29:12.466,value=220001
column=score_info:SC_Score,timestamp=2023-11-25T13:29:12.485,value=95
column=score_info:SC_Sno,timestamp=2023-11-25T13:29:12.451,value=2120003
column=score_info:SC_Cno,timestamp=2023-11-25T13:29:12.537,value=220002
column=score_info:SC_Score,timestamp=2023-11-25T13:29:12.564,value=88
column=score_info:SC_Sno,timestamp=2023-11-25T13:29:12.507,value=2120003
column=score_info:SC_Cno,timestamp=2023-11-25T13:29:12.628,value=220003
column=score_info:SC_Score,timestamp=2023-11-25T13:29:12.642,value=92
column=score_info:SC_Sno,timestamp=2023-11-25T13:29:12.603,value=2120003
column=score_info:SC_Cno,timestamp=2023-11-25T13:29:12.706,value=220001
column=score_info:SC_Score,timestamp=2023-11-25T13:29:12.722,value=85
column=score_info:SC_Sno,timestamp=2023-11-25T13:29:12.677,value=2120004
column=score_info:SC_Cno,timestamp=2023-11-25T13:29:12.790,value=220003
column=score_info:SC_Score,timestamp=2023-11-25T13:29:15.458,value=90
column=score_info:SC_Sno,timestamp=2023-11-25T13:29:12.756,value=2120004
```

Task2:查询选修BigData的学生的成绩

HBase是一个面向列的分布式数据库，因此它不支持内部关联查询。所以本任务使用多次使用过滤器查询数据

查询BigData的课程号

查询课程号对应的成绩

比较器

BinaryComparator

BinaryPrefixComparator

BitComparator

NullComparator

RegexStringComparator

SubstringComparator

表达式缩写

binary:值

binaryprefix:值

bit:值

null

regexstring:正则表达式

substring:值

1. 查询BigData的课程号

这里使用值过滤器和子字符串比较器

```
1 | scan 'courses', FILTER => "SingleColumnValueFilter('course_info', 'C_Name', =, 'substring:BigData')"
```

```
hbase:160:0> scan 'courses', FILTER => "SingleColumnValueFilter('course_info', 'C_Name', =, 'substring:BigData')"
ROW
220003
220003
220003
row(s)

COLUMN+CELL
column=course_info:C_Credit,timestamp=2023-11-25T13:32:20.744,value=4.0
column=course_info:C_Name,timestamp=2023-11-25T13:32:19.473,value=BigData
column=course_info:C_No,timestamp=2023-11-25T13:32:19.459,value=220003
```

可以看到，BigData的课程号是220003

2. 查询220003的成绩

```
base:161:0> scan 'score', {FILTER => 'SingleColumnValueFilter('score_info', 'SC_Cno', =, 'binary:220003')'}
ROW
2120001_220003
2120001_220003
2120001_220003
2120003_220003
2120003_220003
2120003_220003
2120004_220003
2120004_220003
2120004_220003
row(s)

COLUMN+CELL
column=score_info:SC_Cno, timestamp=2023-11-25T13:29:12.288, value=220003
column=score_info:SC_Score, timestamp=2023-11-25T13:29:12.307, value=75
column=score_info:SC_Sno, timestamp=2023-11-25T13:29:12.272, value=2120001
column=score_info:SC_Cno, timestamp=2023-11-25T13:29:12.628, value=220003
column=score_info:SC_Score, timestamp=2023-11-25T13:29:12.642, value=92
column=score_info:SC_Sno, timestamp=2023-11-25T13:29:12.603, value=2120003
column=score_info:SC_Cno, timestamp=2023-11-25T13:29:12.790, value=220003
column=score_info:SC_Score, timestamp=2023-11-25T13:29:15.458, value=90
column=score_info:SC_Sno, timestamp=2023-11-25T13:29:12.756, value=2120004
```

可以看到

- 学号为2120001的同学成绩为75
- 学号为2120003的同学成绩为92
- 学号为2120004的同学成绩为94

3. 根据学号查询姓名

多次使用 `get 'student', '2120001'` 获取每位同学的姓名，最后可以看到

```
base:163:0> get 'student', '2120001'
COLUMN
info: S_Age
info: S_Name
info: S_No
info: S_Sex
row(s)

CELL
timestamp=2023-11-25T12:37:11.060, value=20
timestamp=2023-11-25T12:36:56.756, value=Li Lei
timestamp=2023-11-25T12:36:40.139, value=2120001
timestamp=2023-11-25T12:37:03.998, value=male
```

- Li Lei的同学成绩为75
- Zhang Li的同学成绩为92
- Li Ming的同学成绩为94

Task3:学生表增加电子邮件列

直接多次put即可。

```
base:168:0> put 'student', '2120002', 'info:Email', 'example2@email.com'
Took 0.0040 seconds
base:169:0> put 'student', '2120003', 'info:Email', 'example2@email.com'
Took 0.0045 seconds
base:170:0> put 'student', '2120003', 'info:Email', 'example3@email.com'
Took 0.0049 seconds
base:171:0> put 'student', '2120004', 'info:Email', 'example4@email.com'
Took 0.0049 seconds
base:172:0> scan 'student'
ROW
2120001
2120001
2120001
2120001
2120002
2120002
2120002
2120002
2120003
2120003
2120003
2120003
2120004
2120004
2120004
2120004
4 row(s)
Took 0.0100 seconds

COLUMN+CELL
column=info:Email, timestamp=2023-11-25T14:04:54.093, value=example@email.com
column=info: S_Age, timestamp=2023-11-25T12:37:11.060, value=20
column=info: S_Name, timestamp=2023-11-25T12:36:56.756, value=Li Lei
column=info: S_No, timestamp=2023-11-25T12:36:40.139, value=2120001
column=info: S_Sex, timestamp=2023-11-25T12:37:03.998, value=male
column=info:Email, timestamp=2023-11-25T14:05:23.316, value=example2@email.com
column=info: S_Age, timestamp=2023-11-25T12:44:36.485, value=21
column=info: S_Name, timestamp=2023-11-25T12:44:36.462, value=Han Meimei
column=info: S_No, timestamp=2023-11-25T12:44:36.445, value=2120002
column=info: S_Sex, timestamp=2023-11-25T12:44:36.476, value=female
column=info:Email, timestamp=2023-11-25T14:05:34.549, value=example3@email.com
column=info: S_Age, timestamp=2023-11-25T12:45:00.630, value=20
column=info: S_Name, timestamp=2023-11-25T12:45:00.595, value=Zhang Li
column=info: S_No, timestamp=2023-11-25T12:45:00.572, value=2120003
column=info: S_Sex, timestamp=2023-11-25T12:45:00.617, value=female
column=info:Email, timestamp=2023-11-25T14:05:41.497, value=example4@email.com
column=info: S_Age, timestamp=2023-11-25T12:45:38.830, value=19
column=info: S_Name, timestamp=2023-11-25T12:45:38.815, value=Li Ming
column=info: S_No, timestamp=2023-11-25T12:45:38.796, value=2120004
column=info: S_Sex, timestamp=2023-11-25T12:45:38.822, value=male
```

如果要新增一个列族，那么可以先用altert添加一个列族名，然后再put也可以

```
1 | alter 'student', {NAME => 'Email'}
```

Task4:查询Li Lei的电子邮件;

该任务比较简单，就与Task2中的过滤相同使用即可，可以看到Li Lei 的电子邮箱是example@email.com

```
base:186:0 | scan 'student', FILTER => 'SingleColumnValueFilter('info', 'S_Name', => 'substring:Li Lei')'
ROW
2120001    column=info: Email, timestamp=2023-11-25T14:04:54.093, value=example@email.com
2120001    column=info: S_Age, timestamp=2023-11-25T12:37:11.060, value=0
2120001    column=info: S_Name, timestamp=2023-11-25T12:36:56.756, value=Li Lei
2120001    column=info: S_No, timestamp=2023-11-25T12:36:40.139, value=2120001
2120001    column=info: S_Sex, timestamp=2023-11-25T12:37:03.998, value=male
1 row(s)
took 0.0092 seconds
```

Task5: 删除所创建的表

这题就更简单了，正常使用语法即可

删除表操作:

- 1 | disable <table>
- 2 | drop <table>


```

hbase:187:0> list
TABLE
courses
score
student
test
4 row(s)
Took 0.0064 seconds
=> ["courses", "score", "student", "test"]
hbase:188:0> disable 'test'
Took 0.3293 seconds
hbase:189:0> drop 'test'
Took 0.1181 seconds
hbase:190:0> list
TABLE
courses
score
student
3 row(s)
Took 0.0053 seconds
=> ["courses", "score", "student"]
hbase:191:0>

```

Web UI截图

Table	Description
courses	TABLE_ATTRIBUTES => (METADATA => (HbaseTableInfo {tableName => 'courses', NAME => 'courses', INDEX_BLOCK_ENCODING => NONE, VERSIONS => 1, KEEP_DELETED_CELLS => FALSE, DATA_BLOCK_ENCODING => NONE, TTL => FOREVER, MIN_VERSIONS => 1, REPLICATION_SCOPE => 0, BLOOMFILTER => ROW, PL_MEMORY => 1024, COMPRESSION => NONE, BLOCKCACHE => true, BLOCKSIZE => 65536})
score	TABLE_ATTRIBUTES => (METADATA => (HbaseTableInfo {tableName => 'score', NAME => 'score', INDEX_BLOCK_ENCODING => NONE, VERSIONS => 1, KEEP_DELETED_CELLS => FALSE, DATA_BLOCK_ENCODING => NONE, TTL => FOREVER, MIN_VERSIONS => 1, REPLICATION_SCOPE => 0, BLOOMFILTER => ROW, PL_MEMORY => 1024, COMPRESSION => NONE, BLOCKCACHE => true, BLOCKSIZE => 65536})
student	TABLE_ATTRIBUTES => (METADATA => (HbaseTableInfo {tableName => 'student', NAME => 'student', INDEX_BLOCK_ENCODING => NONE, VERSIONS => 1, KEEP_DELETED_CELLS => FALSE, DATA_BLOCK_ENCODING => NONE, TTL => FOREVER, MIN_VERSIONS => 1, REPLICATION_SCOPE => 0, BLOOMFILTER => ROW, PL_MEMORY => 1024, COMPRESSION => NONE, BLOCKCACHE => true, BLOCKSIZE => 65536})

问题总结及解决方案

关于磁盘扩容问题

在之前我的/usr文件夹内内存不足，如果只用vmware的内存分配也不能将内存正确扩充到我的工作文件夹下，因此参考了这篇文章，成功扩容。

[Linux CentOS 7分配的磁盘空间不足，空间扩展方法，保姆级操作](#)

Tips1:创建快捷启动方式:

修改环境变量:

```
1 | sudo vim ~/.bashrc
```

添加如下别名:

```
1 | alias starthadoop='/usr/local/hadoop/sbin/start-all.sh'
```

就可以快捷启动hadoop

Tips2:快速查找文件路径

可以用which快速查询路径

```
1 | which hadoop
```

关于权限问题（很重要！）

之前配置的时候没有给hadoop用户设置管理员权限，在操作的时候都用sodo，这样在后续启动HBase的时候会出现一点问题。因此，**强烈建议**给用户设置hbase的管理员权限

```
1 | sudo chown -R hadoop:hadoop hbase
```

- `chown` 是改变文件或目录所有者的命令。
- `-R` 选项表示递归地应用于目录及其所有子目录和文件。
- `hadoop:hadoop` 是目标所有者和所属组的标识。在这里，文件/目录的所有者将设置为 `hadoop` 用户，所属组也将设置为 `hadoop` 组。
- `hbase` 是要更改所有者的目标文件或目录的名称。

关于HBase和Hadoop冲突问题

之前运行的时候一直会出现SLF4J冲突什么的。后来解决方案是在env中添加了

```
1 | export HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP="true"
```

好像就好使了

其他思考

HBase是一个基于Hadoop生态的分布式列型数据库,具有高可靠性、高并发读写和易扩展等优点,适合PB级大数据的存储。原因在于它基于HDFS文件系统实现了数据的高可靠存储,利用HBase的分布式架构可以对数据进行自动分区,实现对大规模数据的横向扩展。同时,它通过MemStore和SSTable的结构优化了对列式数据的存储 efficiency 和读取性能。这使得它可以提供每秒数十万次的读写吞吐。

但是,HBase作为一个非关系型NoSQL数据库,它也存在一定的缺陷。比如部署和使用复杂度较高,需要搭建整个Hadoop生态圈,不支持SQL,学习和使用成本较高。此外,HBase无法提供强一致性保证,在极端情况下可能会出现读取不一致数据的问题。而且HBase的读放大问题也会对某些读密集场景的性能产生一定影响。