

大数据处理综合实验课程 Lab4

HBase和Hive的安装与使用

姓名	学号	邮箱
陈昕元	161220018	161220018@smail.nju.edu.cn
陈翔	161220017	161220017@smail.nju.edu.cn
陈亚栋	161220019	161220019@smail.nju.edu.cn

1. 实验环境

本次实验使用的软件及其版本如下：

软件	版本
HBase	1.4.9
Hive	2.3.4
Hadoop	2.9.2
Java	1.8

2. 实验内容与要求

1. 在自己本地电脑上正确安装和运行HBase(推荐1.4.9)和Hive(推荐2.3.4)。
2. 在HBase中创建一张表Wuxia，用于保存下一步的输出结果。
3. 修改第3次实验的MapReduce程序，在Reduce阶段将倒排索引的信息通过文件输出，而每个词语及其对应的“平均出现次数”信息写入到HBase的表“Wuxia”中。
4. 编写Java程序，遍历上一步中保存在HBase中的表，并把表格的内容(词语以及平均出现次数)保存到本地文件中。
5. Hive安装完成后，在Hive Shell命令行操作创建表(表名:Wuxia(word STRING, count DOUBLE))、导入平均出现次数的数据、查询(出现次数大于300的词语)和前100个出现次数最多的词。

3. HBase和Hive的安装与使用

(1) HBase的安装

1) 下载HBase安装包

从Apache网站上(hbase.apache.org)下载HBase稳定发布包: <https://mirrors.cnnic.cn/apache/hbase/1.4.9/>，我们下载目录下的hbase-1.4.9-bin.tar.gz文件至/usr/hbase目录，使用命令

```
tar -zxf hbase-1.4.9-bin.tar.gz
```

解压安装包，这一操作将在/usr/hbase目录下生成hbase-1.4.9文件夹，这即是我们Hbase的安装目录。

2) 设置系统环境变量

修改系统文件/etc/profile，添加如下内容

```
export HBASE_HOME="/usr/hbase/hbase-1.4.9"
export PATH=$PATH:$HBASE_HOME/bin
```

设置完毕后，使用source /etc/profile命令或重新登录该用户，使环境变量生效。之后，输入命令

```
hbase version
```

应可看到如下内容

```
hadoop@chris-virtual-machine:/usr/hbase$ hbase version
HBase 1.4.9
Source code repository git://apurtell-ltm4.internal.salesforce.com/Users/apurtell/src/hbase revision=d625b212e46d01cb17db9ac2e9e927fdb201afa1
Compiled by apurtell on Wed Dec 5 11:54:10 PST 2018
From source with checksum a7716fc1849b07ea6dd830a08291e754
```

3) HBase配置文件设置

此次我们需要实现的仅为单机为分布式HBase配置，因此我们对usr/hbase/hbase-1.4.9/conf中的如下文件进行配置：

修改hbase-env.sh文件中的如下内容：

```
export JAVA_HOME="/usr/java/jdk1.8.0_201"
export HBASE_CLASSPATH="/usr/hadoop/hadoop-2.9.2/etc/hadoop"
```

修改hbase-site.xml文件，在configuration中添加如下内容：

```
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://localhost:9000/hbase</value>
</property>

<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
```

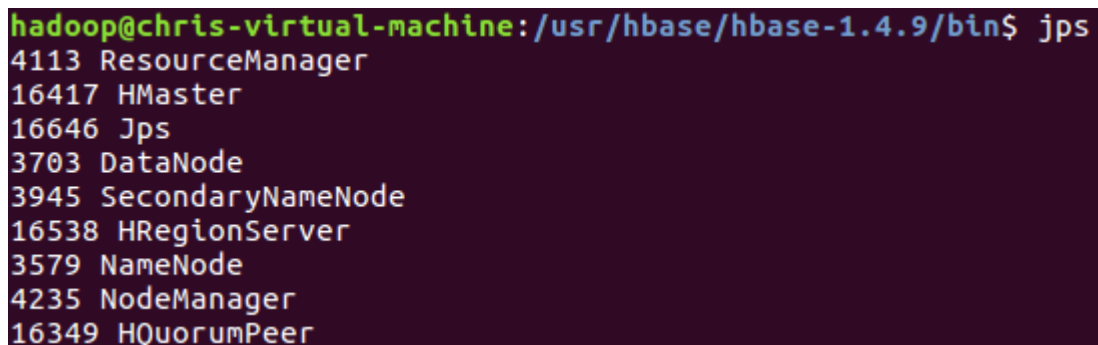
因为我们采用的是伪分布模式，这里需要将HBase的数据存储到之前的Hadoop的HDFS上，hbase.rootdir的值便是HDFS上HBase数据存储的位置。值中的主机名和端口号要和之前Hadoop的 core-site.xml中的fs.default.name的值相同。

4) 启动HBase

在HBase安装目录下，使用命令

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

即可启动HBase。启动后我们执行jps指令，可以看到额外启动了HMaster/ HRegionServer / HQuorumPeer进程



```
hadoop@chris-virtual-machine:/usr/hbase/hbase-1.4.9/bin$ jps
4113 ResourceManager
16417 HMaster
16646 Jps
3703 DataNode
3945 SecondaryNameNode
16538 HRegionServer
3579 NameNode
4235 NodeManager
16349 HQuorumPeer
```

(2) MySQL的安装

在安装Hive之前，我们需要配置好MySQL环境。默认情况下，最新版本的 MySQL 包含在 APT 软件包存储库中。使用如下命令安装MySQL服务器

```
sudo apt-get update
sudo apt-get install mysql-server
```

使用如下命令配置MySQL服务器，配置过程不表

```
sudo mysql_secure_installation
```

使用如下命令可检查MySQL服务状态

```
systemctl status mysql.service
```

使用如下命令安装MySQL客户端

```
sudo apt-get install mysql-client
sudo apt-get install libmysqlclient-dev
```

使用如下命令登录MySQL

```
mysql -u root -p
```

(3) Hive的安装

1) 下载Hive安装包

从Apache网站上下载Hive稳定发布包: <https://mirrors.cnnic.cn/apache/hive/hive-2.3.4/>。

我们下载其中的apache-hive-2.3.4-bin.tar.gz文件至/usr/hive目录，使用命令

```
tar -zxf apache-hive-2.3.4-bin.tar.gz
```

解压安装包为文件夹hive-2.3.4。因此hive的安装路径为/usr/hive/hive-2.3.4

2) 设置系统环境变量

修改系统文件/etc/profile，添加如下内容

```
export HIVE_HOME="/usr/hive/hbase-2.3.4"
export PATH=$PATH:$HIVE_HOME/bin
```

设置完毕后，使用source /etc/profile命令或重新登录该用户，使环境变量生效。

3) 设置配置文件

进入/usr/hive/hive-2.3.4/conf目录，使用命令

```
cp hive-env.sh.template hive-env.sh
```

生成一个hive-env.sh配置文件，并设置其中的HADOOP_HOME，如

```
HADOOP_HOME="/usr/hadoop/hadoop-2.9.2"
```

创建一个新文件hive-site.xml，并在文件中加入以下内容（其中，Password一项的值应与你的常用密码设置为相同，避免混淆）

```
<configuration>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://localhost:3306/sparksql?
createDatabaseIfNotExist=true&useSSL=false</value>
  </property>

  <property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.cj.jdbc.Driver</value>
  </property>

  <property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>root</value>
  </property>

  <property>
    <name>javax.jdo.option.ConnectionPassword</name>
    <value>123456</value>
  </property>
</configuration>
```

4) 拷贝MySQL驱动至hive的lib目录

使用（3）中操作顺利安装MySQL后，系统中应有MySQL相关驱动，使用如下指令寻找MySQL驱动文件所在目录

```
sudo find / -name mysql-connector*
```

在本机中，MySQL驱动被查找为安装在/usr/share/java/目录下，我们将其拷贝到hive的lib目录下

```
cp /usr/share/java/mysql-connector-java-8.0.16.jar /usr/hive/hive-2.3.4/lib
```

5) 启动hive

在首次启动 hive 时:需要初始化模式(init initSchema),我们使用的是mysql数据库,所以初始化的命令是:

```
bin/schematool -initSchema -dbType mysql
```

当有如下显示时,说明初始化完成

```
Metastore connection URL:      jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true&useSSL=false
Metastore Connection Driver :   com.mysql.cj.jdbc.Driver
Metastore connection User:     hive
Starting metastore schema initialization to 2.3.0
Initialization script hive-schema-2.3.0.mysql.sql
Initialization script completed
schemaTool completed
```

然后,在hive安装目录下，使用命令

```
bin/hive
```

即可启动hive程序。需要注意的是，在启动hive之前，请确保hadoop、hbase、MySQL相关进程已启动。

使用quit指令可退出hive。

4. 实验内容

(1) 在HBase中创建表Wuxia

HBase提供了一个基于Ruby语法的命令行Shell，可以通过运行命令hbase shell来进入 Shell:

```
./bin/hbase shell
```

进入Shell模式后，通过以下命令建立表Wuxia

```
create 'wuxia', 'Info'
```

其中'Info'是列族名，今后我们可在其中添加列'Word'和'Avefrequency'表示单词和平均出现次数。

(2) 通过MapReduce程序将每个词语及其对应的“平均出现次数”信息写入到HBase的表“Wuxia”中

我们只需要修改Reduce过程，使其符合HBase的写入方法即可。这其中涉及Reduce类、main函数的改写。

1) Reduce类的改写

在Reduce类的改写任务中，最大的差别在于，我们继承的父类是TableReducer而不是Reducer

Replace:

```
public static class Reduce extends Reducer<Text, IntWritable, Text, Text>
```

To:

```
public static class Reduce extends TableReducer<Text, IntWritable, ImmutableBytesWritable>
```

从上面的代码中我们可以注意到，Reduce阶段输出的内容发生了改变。由原先的Text修改为ImmutableBytesWritable，因此我们在写入输出内容时也需要做相应的改动

Replace:

```
context.write(key_word, textValue);
```

To:

```
Put put = new Put(Bytes.toBytes(curWord));

put.add(Bytes.toBytes("Info"), Bytes.toBytes("Word"), Bytes.toBytes(curWord));
put.add(Bytes.toBytes("Info"), Bytes.toBytes("Avefrequency"),
Bytes.toBytes(String.valueOf(average)));

context.write(new ImmutableBytesWritable(Bytes.toBytes(curWord)), put);
```

其中Put类用于写入HBase的列族信息，两条put.add代码表示在列族"Info"下添加列"Word"和"Avefrequency"，列的内容分别为单词和平均出现次数。在写入Reduce过程的输出context.write时，我们写入put的相应列族信息，并将单词本身作为行索引。

2) main函数的改写

首先，我们需要改变job的配置，由原先的默认配置修改为与HBase有关的配置：

Replace:

```
Configuration conf = new Configuration();
```

To:

```
Configuration conf = HBaseConfiguration.create();
```

其次，我们改变job的Reduce方式，使用TableMapReduceUtil类中的方法，为job设定更新后的Reduce类

Replace:

```
job.setReducerClass(mReduce.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
```

To:

```
TableMapReduceUtil.initTableReducerJob("wuxia", hbaseReduce.class, job);
```

其中第一个参数"Wuxia"即是HBase中的表名。

3) 配置文件pom.xml修改

在mvn的配置文件pom.xml中需要添加与HBase有关的配置如下

```
<dependency>
  <groupId>org.apache.hbase</groupId>
  <artifactId>hbase-client</artifactId>
  <version>1.4.9</version>
</dependency>
<dependency>
  <groupId>org.apache.hbase</groupId>
  <artifactId>hbase-server</artifactId>
  <version>1.4.9</version>
</dependency>
```

4) 执行结果

在这一步的测试中，仅使用样例中的两个武侠小说文档进行测试，使用命令执行代码

```
hadoop jar hadoop-1.0-SNAPSHOT.jar com.myschool.hadoop.InvertedIndexer /exp3_sample_data
/output
```

其中hadoop-1.0-SNAPSHOT.jar是打包后的java包，com.myschool.hadoop.InvertedIndexer是执行的java模块，exp3_sample_data是hadoop集群上的输入目录，output是hadoop集群上的输出目录。

测试结果MapReduce结果如下：

```
19/05/04 17:23:59 INFO mapreduce.Job: Running job: job_1556960582674_0003
19/05/04 17:23:59 INFO mapreduce.Job: Job job_1556960582674_0003 running in uber mode : false
19/05/04 17:23:59 INFO mapreduce.Job: map 100% reduce 100%
19/05/04 17:23:59 INFO mapreduce.Job: Job job_1556960582674_0003 completed successfully
19/05/04 17:23:59 INFO mapreduce.Job: Counters: 50
  File System Counters
    FILE: Number of bytes read=1198076
    FILE: Number of bytes written=3086304
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=1752805
    HDFS: Number of bytes written=0
    HDFS: Number of read operations=4
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=0
  Job Counters
    Killed map tasks=1
    Launched map tasks=2
    Launched reduce tasks=1
    Rack-local map tasks=2
    Total time spent by all maps in occupied slots (ms)=22258
    Total time spent by all reduces in occupied slots (ms)=9124
    Total time spent by all map tasks (ms)=22258
    Total time spent by all reduce tasks (ms)=9124
    Total vcore-milliseconds taken by all map tasks=22258
    Total vcore-milliseconds taken by all reduce tasks=9124
    Total megabyte-milliseconds taken by all map tasks=22792192
    Total megabyte-milliseconds taken by all reduce tasks=9342976
  Map-Reduce Framework
    Map input records=5184
    Map output records=279945
    Map output bytes=8702354
    Map output materialized bytes=1198082
    Input split bytes=300
    Combine input records=279945
    Combine output records=34869
    Reduce input groups=34869
    Reduce shuffle bytes=1198082
    Reduce input records=34869
    Reduce output records=25968
    Spilled Records=69738
    Shuffled Maps =2
    Failed Shuffles=0
    Merged Map outputs=2
    GC time elapsed (ms)=1011
    CPU time spent (ms)=26280
    Physical memory (bytes) snapshot=842215424
    Virtual memory (bytes) snapshot=5981417472
    Total committed heap usage (bytes)=506462208
```

进入hbase的shell模式中，采用scan命令列出'Wuxia'表中的数据，结果截图如下


```

hbase(main):001:0> scan 'Wuxia'
ROW                                COLUMN+CELL
CO                                 column=Info:Avefrequency, timestamp=1556961834930, value=1.0
CO                                 column=Info:Word, timestamp=1556961834930, value=CO
COM                                column=Info:Avefrequency, timestamp=1556961834930, value=2.0
COM                                column=Info:Word, timestamp=1556961834930, value=COM
J                                  column=Info:Avefrequency, timestamp=1556961834930, value=1.0
J                                  column=Info:Word, timestamp=1556961834930, value=J
J.                                 column=Info:Avefrequency, timestamp=1556961834930, value=1.0
J.                                 column=Info:Word, timestamp=1556961834930, value=J.
0                                  column=Info:Avefrequency, timestamp=1556961834930, value=1.0
0                                  column=Info:Word, timestamp=1556961834930, value=0
WWW.azbbs.com                     column=Info:Avefrequency, timestamp=1556961834930, value=1.0
WWW.azbbs.com                     column=Info:Word, timestamp=1556961834930, value=WWW.azbbs.com
WaP.azbbs.com                     column=Info:Avefrequency, timestamp=1556961834930, value=1.0
WaP.azbbs.com                     column=Info:Word, timestamp=1556961834930, value=WaP.azbbs.com
X                                  column=Info:Avefrequency, timestamp=1556961834930, value=4.0
X                                  column=Info:Word, timestamp=1556961834930, value=X
sp                                 column=Info:Avefrequency, timestamp=1556961834930, value=1.0
sp                                 column=Info:Word, timestamp=1556961834930, value=sp
\xE3\x81\x81                     column=Info:Avefrequency, timestamp=1556961834930, value=1.0
\xE3\x81\x81                     column=Info:Word, timestamp=1556961834930, value=\xE3\x81\x81
\xE3\x81\x83                     column=Info:Avefrequency, timestamp=1556961834930, value=2.0
\xE3\x81\x83                     column=Info:Word, timestamp=1556961834930, value=\xE3\x81\x83
\xE3\x82\xA3                     column=Info:Avefrequency, timestamp=1556961834930, value=1.0
\xE3\x82\xA3                     column=Info:Word, timestamp=1556961834930, value=\xE3\x82\xA3
\xE3\x82\x81                     column=Info:Avefrequency, timestamp=1556961834930, value=1.0
\xE3\x82\x81                     column=Info:Word, timestamp=1556961834930, value=\xE3\x82\x81
\xE4\xB8\x80\xE4\xB8\x87\xE4\xB8\x80\xE5\x8D\x83\xE4\xB8\xA4 column=Info:Avefrequency, timestamp=1556961834930, value=1.0
4\xB8\xA4                        column=Info:Word, timestamp=1556961834930, value=\xE4\xB8\x80\xE4\xB8\x87\xE4\xB8\x80\xE5\x8D\x83\xE4\xB8\xA4
4\xB8\xA4                        column=Info:Avefrequency, timestamp=1556961834930, value=10.0
\xE4\xB8\x80\xE4\xB8\x87\xE4\xB8\xA4 column=Info:Word, timestamp=1556961834930, value=\xE4\xB8\x80\xE4\xB8\x87\xE4\xB8\xA4
\xE4\xB8\x80\xE4\xB8\x87\xE4\xB8\xA4 column=Info:Avefrequency, timestamp=1556961834930, value=1.0
\xE4\xB8\x80\xE4\xB8\x87\xE4\xB8\xAA column=Info:Word, timestamp=1556961834930, value=\xE4\xB8\x80\xE4\xB8\x87\xE4\xB8\xAA
\xE4\xB8\x80\xE4\xB8\x87\xE4\xB8\xAA column=Info:Avefrequency, timestamp=1556961834930, value=1.0
4\xB8\xA4                        column=Info:Word, timestamp=1556961834930, value=\xE4\xB8\x80\xE4\xB8\x87\xE4\xB8\x8C\xE5\x8D\x83\xE4\xB8\xA4
4\xB8\xA4                        column=Info:Avefrequency, timestamp=1556961834930, value=4.0
\xE4\xB8\x80\xE4\xB8\x87\xE4\xB8\x88 column=Info:Word, timestamp=1556961834930, value=\xE4\xB8\x80\xE4\xB8\x88
\xE4\xB8\x80\xE4\xB8\x88         column=Info:Avefrequency, timestamp=1556961834930, value=1.0
\xE4\xB8\x80\xE4\xB8\x8A\xE5\x8F\xB0 column=Info:Word, timestamp=1556961834930, value=\xE4\xB8\x80\xE4\xB8\x8A\xE5\x8F\xB0
\xE4\xB8\x80\xE4\xB8\x8A\xE5\x8F\xB0 column=Info:Avefrequency, timestamp=1556961834930, value=1.0
\xE4\xB8\x80\xE4\xB8\x8A\xE6\x9D\xA5 column=Info:Word, timestamp=1556961834930, value=\xE4\xB8\x80\xE4\xB8\x8A\xE6\x9D\xA5
\xE4\xB8\x80\xE4\xB8\x8B\xE5\x9C\xBA column=Info:Avefrequency, timestamp=1556961834930, value=1.0
\xE4\xB8\x80\xE4\xB8\x8B\xE5\x9C\xBA column=Info:Word, timestamp=1556961834930, value=\xE4\xB8\x80\xE4\xB8\x8B\xE5\x9C\xBA
\xE4\xB8\x80\xE4\xB8\x8B\xE5\xAD\x90 column=Info:Avefrequency, timestamp=1556961834930, value=1.0

```

其中存在类似"\xE3\x81\x81"这样的编码，这种编码表示中文。在HBase的shell模式下，我们不能方便地查看中文字符，但我们仍可对这样的数据进行修改和操作。随着我们接下来的操作，我们可在其他地方看到这些编码对应的中文字符。

Hadoop集群上的/output/part-r-00000文件是我们在本地保存的文件，查看文件内容如下：

```

龙争虎斗      1.0, 金庸02雪山飞狐:1
龙卷          1.0, 金庸01飞狐外传:1
龙潭虎穴      4.0, 金庸01飞狐外传:6; 金庸02雪山飞狐:2
龙翔凤舞      1.0, 金庸02雪山飞狐:1
龙虎          1.0, 金庸01飞狐外传:1
龙蛇混杂      1.0, 金庸01飞狐外传:1
龙钟          2.0, 金庸01飞狐外传:2
龙钟老态      1.0, 金庸01飞狐外传:1
龙锤          1.0, 金庸01飞狐外传:1
龙门          41.5, 金庸01飞狐外传:27; 金庸02雪山飞狐:56
龙门镇        1.0, 金庸01飞狐外传:1; 金庸02雪山飞狐:1

```

(3) 写一个java程序从上一步中读取hbase中的表"Wuxia",并把内容写入本地文件

1) 使用hbase的java api获取Wuxia内容

首先需要定义HTable 需要的配置属性:共有两项,分别是

```

conf.set("hbase.zookeeper.quorum", "localhost");
conf.set("hbase.zookeeper.property.clientPort", "2181");

```

这种使用代码的方式和将配置写为xml配置文件的作用是相同的,

其次,新建一个HTable对象,它需要上面定义的配置变量conf和要访问的表的名称,再新建Scan对象,它用来描述如何遍历表中的"行",即指定要扫描的区域,这里默认是遍历所有行,所以用它的默认构造函数即可。

然后调用HTable的getScanner方法,即可连接到表中的数据了,在遍历"行"的时,我们会得到"键值对(KeyValue)"类型的变量,它对外提供了getValue,getQualifier等方法使我们能获某一行中得具体内容,如"列名称","值"等等,

因为hbase在显示结果时,往往"一行"内容是多行显示的,比如某一个词语有属性"Avefrequency"和"Word",显示时会分两行显示,这两个属性的内容都用"getValue"获得,所以在记录时先判断"列名称",然后取值,

最终将能得到表中所有内容.

2) 将上一步获得结果写入文件

这一步是java io的过程,先用File对象来新建一个文件,

```
File outputFile = new File(outFileName);
```

然后将FileWriter绑定到outputfile上

```
Writer out = new FileWriter(outputfile);
```

最后用Writer的write方法就能将字符写入到File对象对应的文本文件中了,

3)代码使用指南

因为这是一个单独的程序,所以处在单独的项目中,它会读取运行此项目的机器的hbase,并项目根目录生成一个文本文件作为输出文件,正确运行的前提是hbase中存在'Wuxia'这个表,并且有一个列族名为'Info',

(4) 在Hive Shell命令行操作创建表、导入数据、执行查询操作

1) 进入Hive Shell并创建表

在之前的hive安装部分我们提到了如何进入hive, 进入之后在Hive Shell中使用HiveQL创建表,

```
create table wuxia(  
word STRING,  
count DOUBLE)  
row format delimited fields terminated by '\t'  
lines terminated by '\n';
```

这里我们必须显示指出行内分隔符和行间分割符, 否则若与默认分隔符(比如行间的默认分隔符并不是'\n')不一致的话会出现错误。

2) 将之前生成的本地文件导入表中

HiveQL极大简化了导入过程, 三行指令即可,

```
load data local inpath  
'/home/hadoop/workspace/BigData/Lab4/result/wuxia.txt'  
into table wuxia;
```

注意, 这里我们使用的是绝对地址, 具体运行时要视存储位置进行修改。

3) 使用查询语句发现感兴趣的结果

经过上一步的操作，我们已经将数据导入表中，可以使用命令

```
select * from wuxia;
```

来验证是否成功，这里我们省略。

首先查询出现次数大于300的词语，使用命令，

```
select * from wuxia where count > 300;
```

部分结果如下：

```
hive> select * from Wuxia where count > 300;
OK
一个      517.6667
一声      433.53333
丁不四    343.0
丁典      364.0
丁当      363.0
丁春秋    432.0
万震山    333.0
上        658.5333
下        331.6
不        847.4
不知      335.6
与        385.06668
东方不败      319.0
两人      315.13333
个        370.2
中        937.2
为        307.0
么        320.69232
```

然后查询出现次数最多的100个词语，使用命令，

```
select * from wuxia sort by count desc limit 100;
```

MapReduce执行情况如下：

```

hive> select * from Wuxia sort by count desc limit 100;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e.
spark, tez) or using Hive 1.X releases.
Query ID = hadoop_20190506151101_7c840552-154d-4f70-9fe3-c6552cb7be97
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2019-05-06 15:11:05,731 Stage-1 map = 100%,   reduce = 0%
2019-05-06 15:11:06,745 Stage-1 map = 100%,   reduce = 100%
Ended Job = job_local2042898714_0001
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2019-05-06 15:11:08,640 Stage-2 map = 100%,   reduce = 100%
Ended Job = job_local826313120_0002
MapReduce Jobs Launched:
Stage-Stage-1:   HDFS Read: 6079686 HDFS Write: 2026562 SUCCESS
Stage-Stage-2:   HDFS Read: 6079686 HDFS Write: 2026562 SUCCESS

```

部分运行结果如下：

```

Stage-Stage-1:   HDFS Read: 6079686 HDFS Write: 2026562 SUCCESS
Stage-Stage-2:   HDFS Read: 6079686 HDFS Write: 2026562 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
韦小宝    4914.0
张无忌    4667.0
道        3715.3333
他        3439.0
我        3370.6
段誉      3369.0
你        3173.6667
是        2073.8667
令狐冲    1905.0
了        1835.6
石破天    1792.0
黄蓉      1680.3334
虚竹      1606.0
那        1577.9333
这        1568.0667
袁承志    1518.5
也        1501.7333
胡斐      1492.5
狄云      1408.0
郭靖      1286.4
杨过      1275.25
赵敏      1250.0
谢逊      1211.0
在        1189.0
岳不群    1184.0
张翠山    1145.0

```

完整的运行结果请参见本目录下的query文件夹。

5. 实验结论

在金庸的小说中出现最频繁的单词为人名类（比如韦小宝、张无忌等）或单字类（比如道、他、我等），除此之外还有少数很常见的词组。这与我们的直观相符合。

稍显意外的是，我们印象中代表金庸的武侠精神的词语（比如江湖、武侠、仁义等），没有一个入选前100个高频词。这也说明了人的主观能够记住频率稍低却更有意义的词，而不会完全受出现次数影响。

6. 实验中遇到的问题及解决思路

(1) 权限问题

在我们的实验中，执行启动、关闭HBase等敏感操作经常会遇到权限问题，即使我们每次都使用root用户执行命令，也可能遇到难以解决的权限问题。在单机系统下，可暴力地使用如下指令赋予HBase、Hive等文件夹高权限（读、写、执行），如

```
chmod -R 777 /usr/hbase
```

(2) 运行MapReduce任务时出现"The auxService:mapreduce_shuffle does not exist"

出现这种情况的原因是在配置文件中没有指定aux-services。解决方案为：在\$HADOOP_HOME/etc/hadoop/yarn-site.xml中添加以下内容

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
```

(3) 运行MapReduce任务时无法关联HBase中jar包

如果没有在Hadoop配置文件中指定HBase的相关信息，则MapReduce任务无法定位到HBase中的jar包，解决方案是为HADOOP_CLASSPATH添加hbase中jar包的存放路径。在实际操作中，修改\$HADOOP_HOME/etc/hadoop/hadoop-env.sh，添加如下内容：

```
for f in /usr/hbase/hbase-1.4.9/lib/*.jar; do
  if [ "HADOOP_CLASSPATH" ]; then
    export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
  else
    export HADOOP_CLASSPATH=$f
  fi
done
```

(4) 在写hbase的java编程接口时,运行出现 Failed to detect a valid hadoop home directory 错误

因为hadoop安装目录在单独的用户(即hadoop)文件夹下,而在其它用户目录下运行hbase的api,因为HADOOP_HOME这个变量没有设置,需要在程序运行中指定 hadoop_home 的位置,代码如下:

```
System.setProperty("hadoop.home.dir", "/home/hadoop/hadoop_installs/hadoop-2.9.2/");
```

(5) 在写hbase的java编程接口时,运行出现 Hadoop “Unable to load native-hadoop library for your platform” warning

这是一个warning,意味着\$HADOOP_HOME/lib/native/libhadoop.so.1.0.0包是32位环境编译的,报出这个warning是因为我们通常在64位机器上运行,实际上这个warning 不影响hadoop的功能,如果要解决的话可以下载相应的源代码在64位环境下编译即可

(6)在上面的环境中,出现了log4j:WARN No appends could be found for logger (org.apache.hadoop.metrics2.lib.MutableMetricsFactory)

出现这个warn说明需要为log4j这个日志框架 传递配置信息,因为它可以默认读取java项目main/resources下的配置文件,log4j.xml和log4j.properties都可以,如此项目所示,我新建了log4j.properties,里面可以控制 日志的 输出格式和日志输出的最低条件等信息,

(7)HADOOP在 running_job 处卡住

经过查看日志,发现是namenode没有活动,即yarn在得到任务后并没有运行,原来是我的机器磁盘空间已经使用了90%,而yarn的磁盘使用警戒线就是90%,有两种办法,一个是清理磁盘,扩大可使用的空间,另一个是将yarn的磁盘使用上限调高,我用了后一种办法,配置文件yarn-site.xml中增加的配置项如下,将使用上限调到了98.5%:

```
<property>
  <name>yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-
percentage</name>
  <value>98.5</value>
</property>
```

(8)关于Hive

从安装hive到成功运行第一条HiveQL指令中出现了不少问题,其中最诡异的两条都与配置文件hive-site.xml相关。

第一个是每运行一条指令报大量Warning,严重影响有效信息的捕获。其内容为“Establishing SSL connection without server's identity verification is not recommended”, 搜索后发现原因为连接MySQL服务时默认useSSL参数为true,而我们并没有提供信任库。因此我们需要在配置里加上“useSSL=false”, Warning就解除了。

第二个是初始化模式失败,报错为时区冲突,这也是一个少见的问题,经搜索后在配置文件中加入“serverTimezone=UTC”即可。

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://localhost:3306/hive?
serverTimezone=UTC&createDatabaseIfNotExist=true&useSSL=false</value>
</property>
```

7. 实验分工

姓名	分工
陈昕元	编写了HBase、Hive的安装教程；完成了将实验三中倒排索引程序改写成适配向HBase输入的MapReduce程序
陈翔	完成了Hive和MySQL的相关安装，并在Hive Shell中进行了建表、装载与查询操作
陈亚栋	编写java程序遍历hbase表,并把结果写入到文件中,完成了与此相关的报告内容