实验报告

报告标题：MR-Hbase-Hive
学号：21190630
姓名：黄艺杰
日期：2022 年 12 月 11 日

一、实验环境

1. 操作系统：Windows 10、Linux

2. 相关软件（含版本号）：VMWare15pro、FinalShell3.9、Intellij IDEA2019Professional

3. 其它工具：JDK1.8

二、实验内容及其完成情况

（针对上述实验内容逐一详述实验过程）

1. MapReduce 编程题

要求：基于天气数据文件（weather.txt），计算月平均气温。要求有上传气象数据的代码，MR 程序代码，运行过程，下载并查看结果的代码验证方法：在虚拟机中查看运行过程与结果。

先在 IDEA 上使用 put 函数将 weather.txt 文件上传到自定义文件夹 homework 中

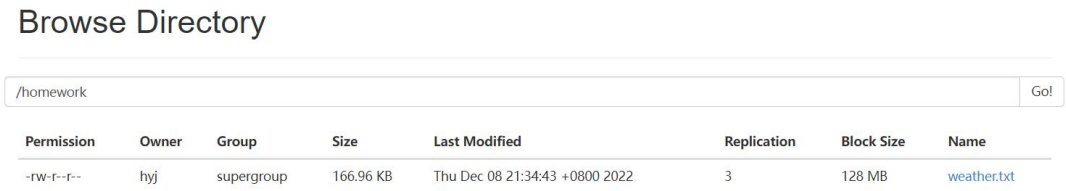

图 1.1HDFS 中 homework 文件夹

上传代码如下：

```
put("/D:/qiyeshixun/wenjian/weather.txt","/homework/weather.txt");
```

```
private static void put(String src,String dest) throws IOException {
        fileSystem.copyFromLocalFile(new Path(src),new Path(dest));
        System.out.println("put ok");
}
```

MR 代码如下：

WeatherMapper：

```
public class WeatherMapper extends Mapper<LongWritable, Text, Text, LongWritable> {
    @Override
    protected void map(
            LongWritable key,
            Text value,
            Mapper<LongWritable, Text, Text, LongWritable>.Context context
    ) throws IOException, InterruptedException {
        String[] ss = value.toString().split(",");
        if (ss[0].equals("DATE")) {
            return;
        }

        String yymm = ss[0].substring(0, 6);
        try {
            context.write(
                    new Text(yymm),
                    new LongWritable(Long.parseLong(ss[5]))
            );
        } catch (Exception ex) {
        }
    }
}
```

WeatherReducer：

```
public class WeatherReducer extends Reducer<Text, LongWritable, Text, DoubleWritable> {
    @Override
    protected void reduce(
            Text key,
            Iterable<LongWritable> values,
            Reducer<Text, LongWritable, Text, DoubleWritable>.Context context
    ) throws IOException, InterruptedException {
        long sum = 0L;
        long cnt = 0L;
        for (LongWritable lw : values) {
            cnt++;
            sum += lw.get();
        }
        context.write(
                key,
                new DoubleWritable(sum * 1.0 / cnt)
        );
    }
}
```

WeatherApp：

```
1.  public class WeatherApp {
2.      public static void main(String[] args) throws IOException {
3.          Configuration conf = new Configuration();
4.          Job job = Job.getInstance(conf);
5.          job.setJarByClass(WeatherApp.class);
6.          job.setMapperClass(WeatherMapper.class);
7.          job.setReducerClass(WeatherReducer.class);
8.          job.setMapOutputKeyClass(Text.class);
9.          job.setMapOutputValueClass(LongWritable.class);
10.         job.setOutputKeyClass(Text.class);
11.         job.setOutputValueClass(DoubleWritable.class);
12.         FileInputFormat.setInputPaths(job, new Path(args[0]));
13.         FileOutputFormat.setOutputPath(job, new Path(args[1]));
14.         try {
15.             job.waitForCompletion(true);
16.         } catch (Exception ex) {
17.             System.out.println(ex);
18.         }
19.     }
20. }
```

将以上文件打包后，上传到虚拟机中，并在虚拟机中运行结果，将结果保存到 HDFS 的 out1 文件夹中

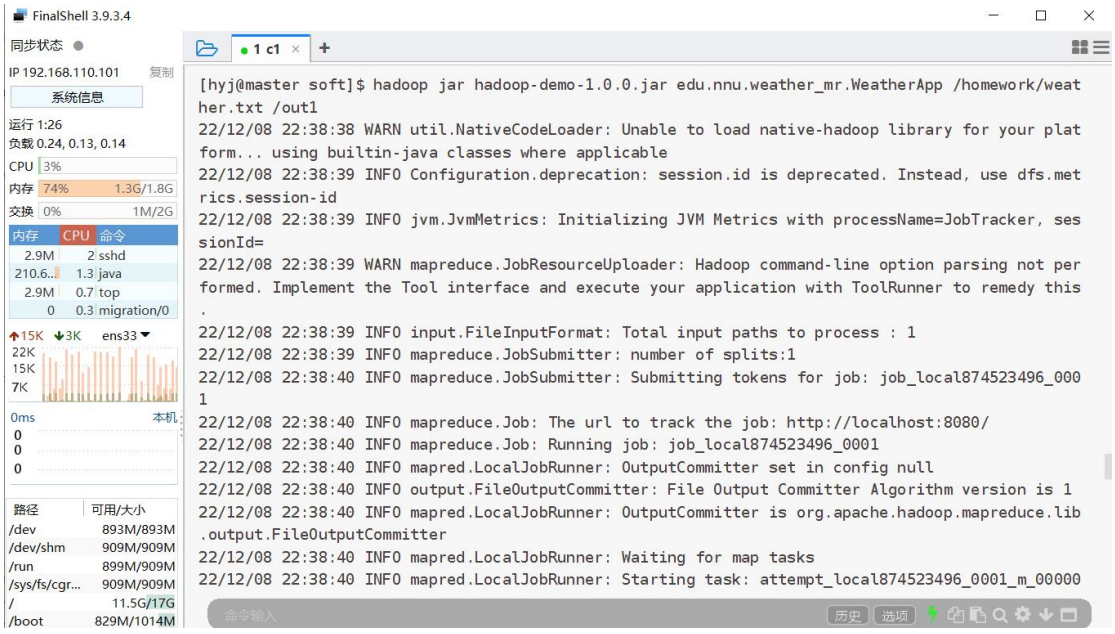

图 1.2 在虚拟机上运行 WeatherApp

可在虚拟机上远程查看结果内容

```
[hyj@master soft]$ hadoop fs -cat /out1/part-r-00000
22/12/08 22:39:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your plat
form... using builtin-java classes where applicable
201601  -3.808199121522694
201602  1.680161943319838
201603  9.741197183098592
201604  16.772661870503597
201605  21.534170153417016
201606  25.697014925373136
[hyj@master soft]$
```

图 1.3 虚拟机上远程查看结果

将实验结果下载到本地 result 文件中的 weather.txt，并在本地查看，下载代码如下：

```
1.  get("/out1/part-r-00000","/D:/qiyeshixun/result/weather.txt");
```

```
1.  private static void get(String src, String dest) throws IOException {
2.          fileSystem.copyToLocalFile(false,new Path(src),new Path(dest),true);
3.          System.out.println("get ok!");
4.      }
```

在本地打开下载的 weather.txt 文件查看结果：

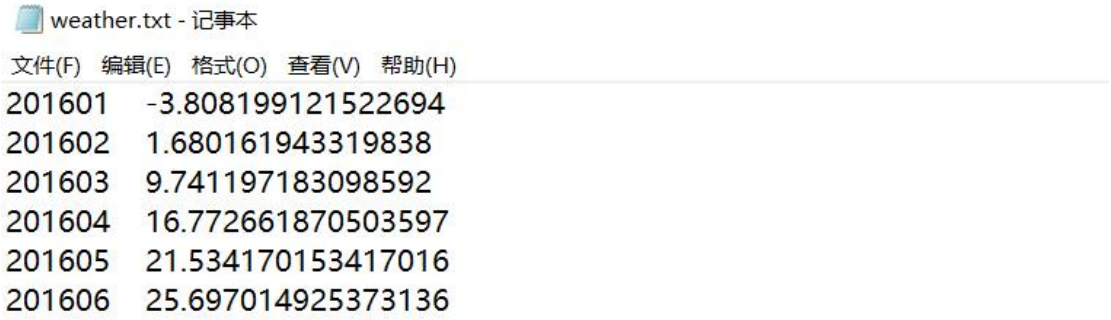

图 1.4 在本地查看下载的实验结果

2. MapReduce 编程题

要求：统计在四大名著中出现总次数最多的 20 个词（2 个字或以上的词）。要求有文件上传的代码、MR 程序、运行过程，下载结果的代码，排序数据并打印最终结果（输出：词和出现次数）的代码。验证方法：在虚拟机中查看运行过程，在 IDEA 中查看最终结果。

将四大名著上传到 HDFS 的 demo 文件夹中并合并成一个文件 Four_masterpieces.txt

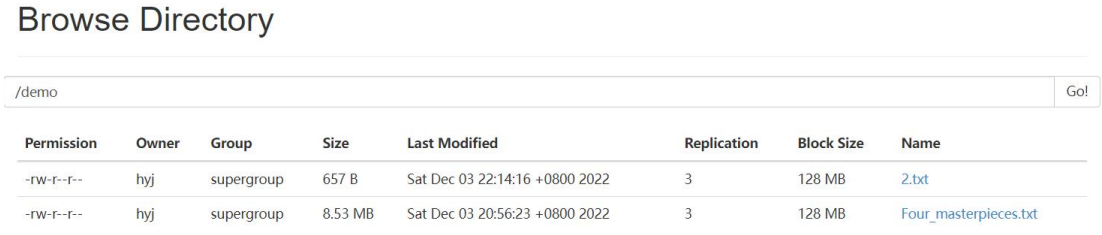

图 2.1 上传四大名著并合并成一个文件

上传代码如下：

```
writeAllFilesOfDir("/D:/qiyeshixun/book","/demo/Four_masterpieces.txt");
```

```
private static void writeAllFilesOfDir(String dir,String dest) {
        File path = new File(dir);
        if(path.exists() && path.isDirectory()){
            File[] files=path.listFiles();
            byte[] bts=new byte[4096];
            int size = 0;
            FSDataOutputStream outputStream=null;
            try {
                outputStream = fileSystem.create(new Path(dest));
                for (int i = 0; i < files.length; i++) {
                    FileInputStream fis = null;
                    if (files[i].isFile()) {
                        try {
                            fis = new FileInputStream(files[i]);
                            while ((size = fis.read(bts)) > 0) {
                                outputStream.write(bts,0,size);
                            }
                        } catch (Exception ex) {
                            System.out.println(ex);
                        } finally {
                            try {
                                fis.close();
                            } catch (Exception e) {
                            }
                        }
                    }
                }
            }
            catch(Exception ex){
                System.out.println(ex);
            }
            finally{
            }
        }
        else{
            System.out.println("not found");
        }
    }
```

MR 代码如下：

WordCountMapper 代码：

```
public class WordCountMapper extends Mapper<LongWritable, Text,Text,LongWritable> {
    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        JiebaSegmenter seg=new JiebaSegmenter();
        List<SegToken> tokenList=seg.process(value.toString(),JiebaSegmenter.SegMode.SEARCH);
       LongWritable one=new LongWritable(1);
       for(SegToken segToken : tokenList){
           if(segToken.word.matches("[\u4e00-\u9fa5]{2,}")){
           context.write(new Text(segToken.word),one);
           }
       }
    }
}
```

WordCountReducer 代码：

```
public class WordCountReducer extends Reducer<Text, LongWritable,Text,LongWritable> {
    @Override
    protected void reduce(Text key, Iterable<LongWritable> values, Context context) throws IOException, InterruptedException {
        //super.reduce(key, values, context);
        long sum=0;
        for (LongWritable value : values) {
            sum+=value.get();
        }
        context.write(key,new LongWritable(sum));
    }
}
```

WordCountApp 代码：

```
public class WordCountApp {
    public static void main(String[] args) throws Exception {
        Configuration config=new Configuration();
        Job job=Job.getInstance(config);

        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);
        job.setJarByClass(WordCountApp.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(LongWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);

```

```
        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job,new Path(args[1]));

        job.waitForCompletion(true);
    }
}
```

将以上代码打包在虚拟机上运行，将结果保存在 HDFS 的 out2 文件夹，运行过程如下：

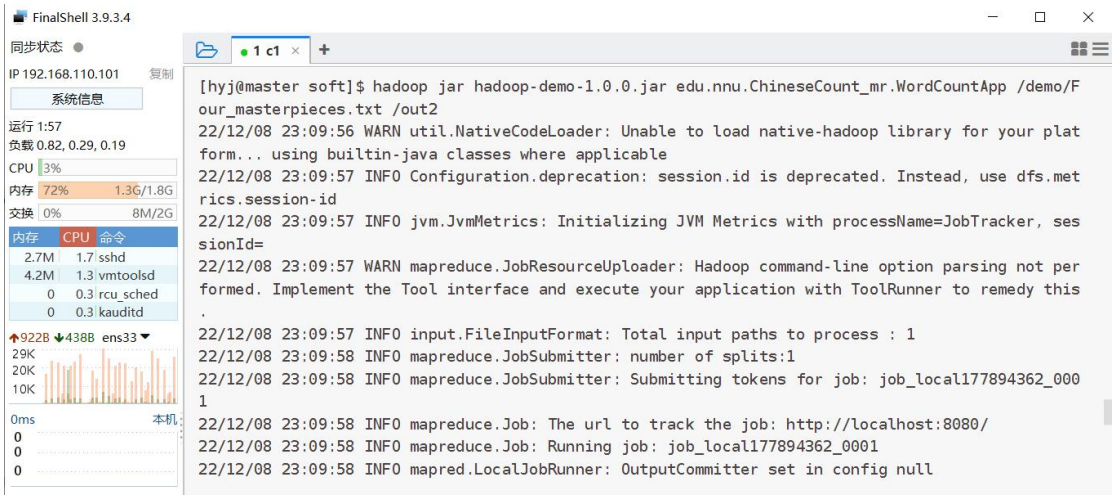

图 2.2WordCountApp 在虚拟机上运行过程

下载结果的代码如下：

```
get("/out2/part-r-00000","/D:/qiyeshixun/result/Four_masterpieces.txt");
```

```
1.private static void get(String src, String dest) throws IOException {
2.        fileSystem.copyToLocalFile(false,new Path(src),new Path(dest),true);
3.        System.out.println("get ok!");
4.    }
```

最后在 IDEA 排序数据并打印最终的结果，其代码如下：

```
public class WordCountResultApp {
    static class Word{
        public String key;
        public Integer count;
    }
    public static void main(String[] args){
        File file =new File("D:/qiyeshixun/result/Four_masterpieces.txt");
        if(!file.exists()){
            System.out.println("file not found");
            return;
        }
        InputStream is=null;
```

```
        Reader reader=null;
        BufferedReader br=null;
        List<Word> words=new ArrayList<Word>();
        try{
            is=new FileInputStream(file);
            reader =new InputStreamReader(is);
            br=new BufferedReader(reader);
            String line=null;
            while((line=br.readLine())!=null){
                String []tmp=line.split("\t");
                Word word=new Word();
                word.key=tmp[0];
                word.count=Integer.parseInt(tmp[1]);
                words.add(word);
            }
        }catch(Exception ef){
            System.out.println(ef);
        }finally{
            try{
                br.close();
            }catch (Exception e){}
            try{
                reader.close();
            }catch(Exception e){}
            try{
                is.close();
            }catch(Exception e){}
        }
        words.sort((a,b)->b.count-a.count);
        for(int i=0;i<20;i++){
            System.out.println(words.get(i).key+":"+words.get(i).count);
        }
    }
}
```

运行后输出四大名著出现总次数最多的 20 个词：

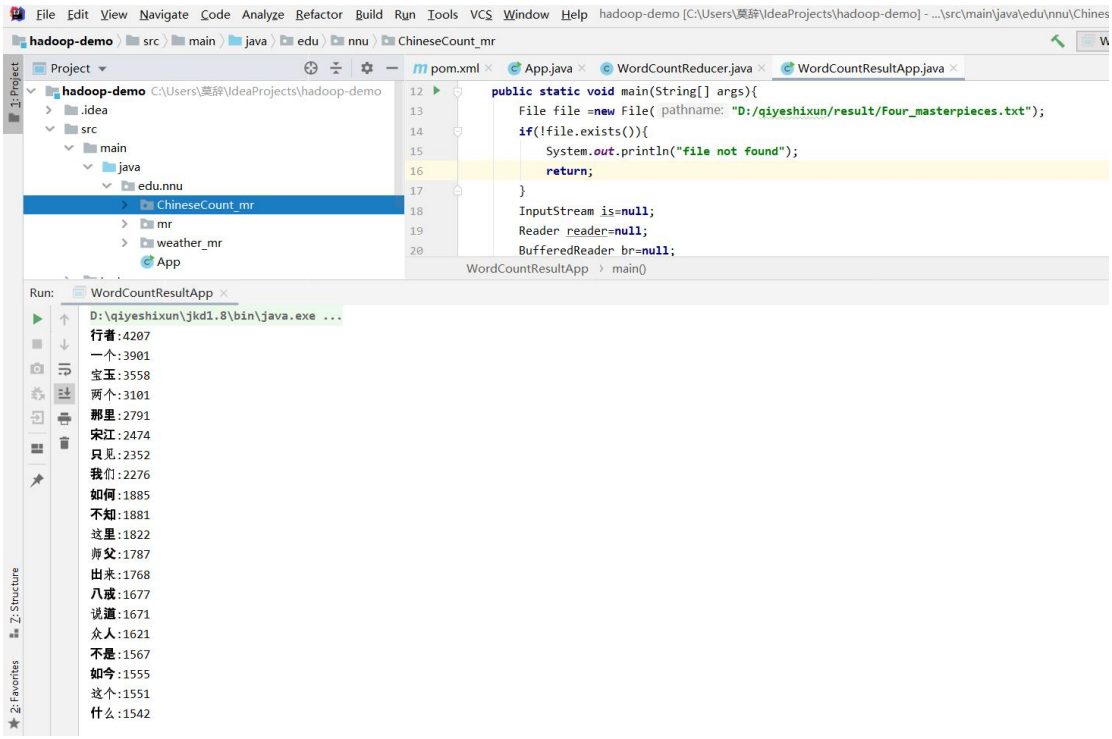

图 2.3 统计四大名著热词

3. Hbase 集群的安装配置（包括 Zookeeper 的分布式配置）：

验证方法：在宿主机中通过浏览器查看 CentOS 7 的 60010 端口

先在虚拟机上将 zookeeper 的压缩包解压到根目录并改名为 zookeeper

```
[hyj@master ~]$ cd soft
[hyj@master soft]$ ll
总用量 27408
-rw-rw-r--. 1 hyj hyj 28065455 12月  9 12:43 zookeeper-3.4.5-cdh5.7.0.tar.gz
[hyj@master soft]$ tar xvf zookeeper-3.4.5-cdh5.7.0.tar.gz -C ~/
```

图 3.1 解压 zookeeper 的压缩包

接下来在.bash_profile 中配置环境，配置 zoo.cfg 文件

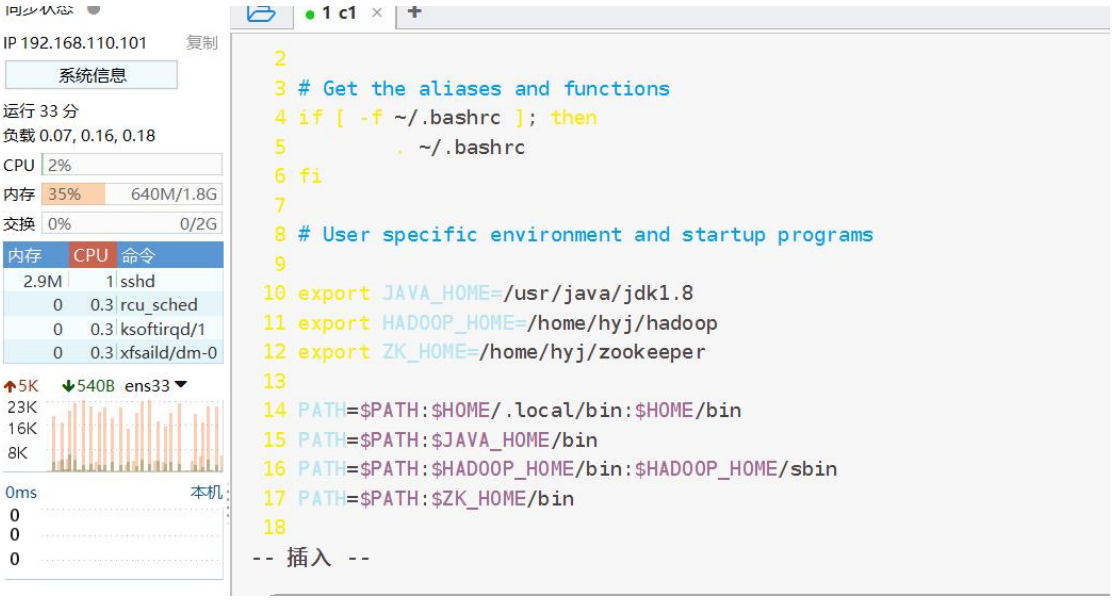

图 3.2 编辑.bash_profile 环境配置文件

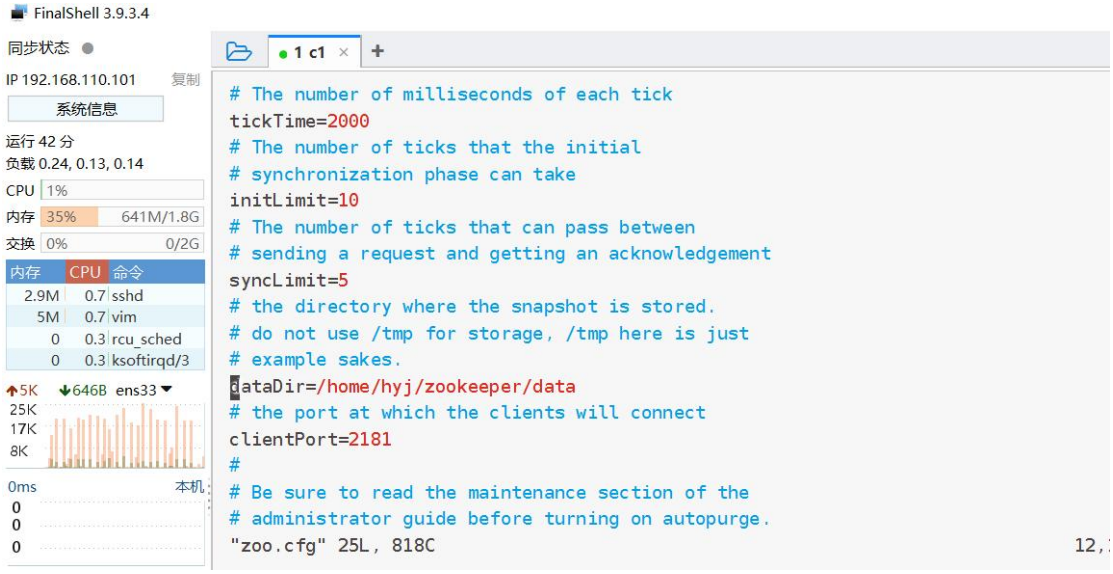

图 3.3 编辑 zoo.cfg 文件

接下来上传 Hbase 的文件，与上述过程基本类似，先上传文件并解压缩，改名，添加环境变量，修改 hbase-env.sh 文件。

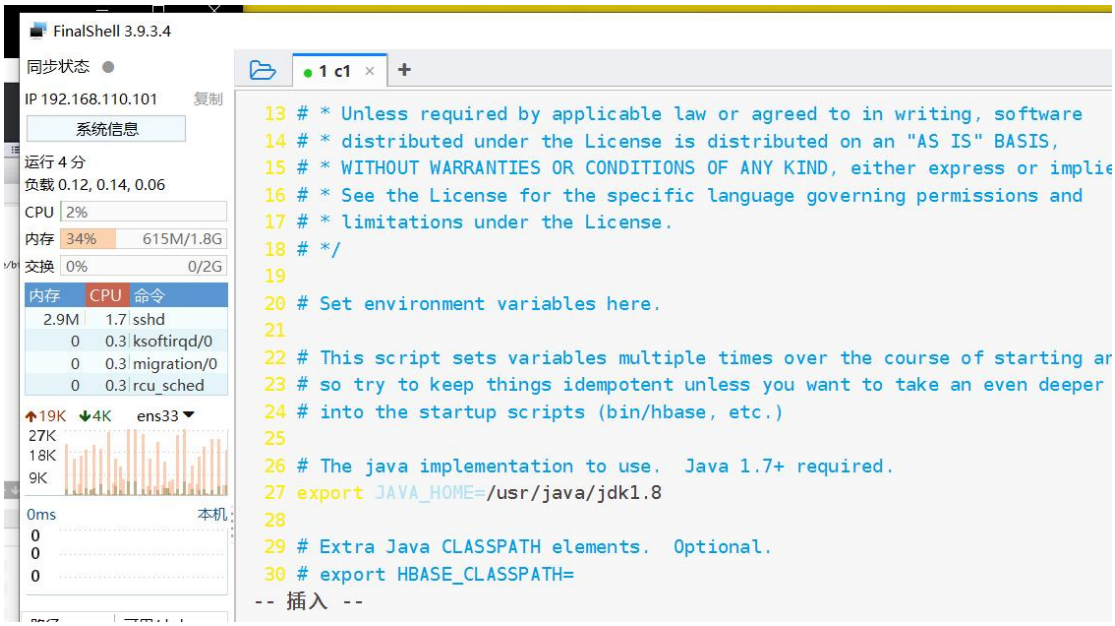

图 3.4 配置 hbase-env.sh 文件

配置 hbase-site.xml

图 3.5 配置 hbase-site.xml 文件

打开 hbase.sh，在 jps 中看到 HMaster

```
[hyj@master conf]$ start-hbase.sh
starting master, logging to /home/hyj/hbase/logs/hbase-hyj-master-master.out
[hyj@master conf]$ jps
37633 HMaster
38409 Jps
```

图 3.6 开启 hbase.sh

最后进入浏览器中 60010 端口

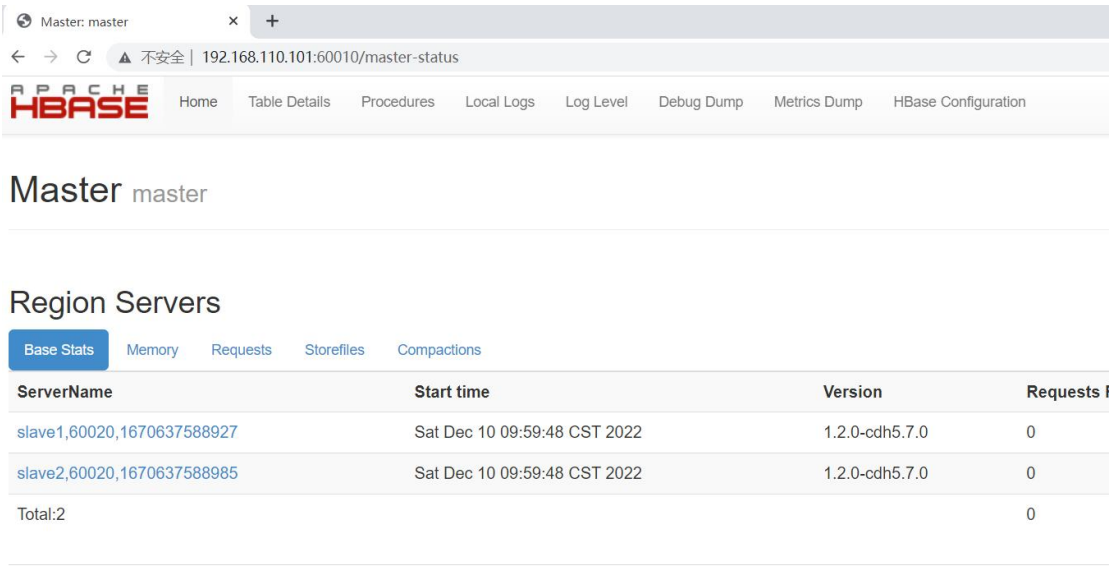

图 3.7 浏览器查看 60010 端口

4. Hbase 编程题

基于课堂案例，定义一个方法，参数为课程名称，功能为完成根据该课程名称查询出该课程所有学生的成绩总分和均分，在 main 方法中测试该方法（课程名自行指定，如果是不存在的课程名，则输出提示信息后结束）。要求有完整的代码。

验证方法：在 IDEA 中输出运行结果

该方法代码如下：

```
1.    private static void CourseScores(String coursename) throws Exception {
2.          String single="";
3.          switch(coursename){
4.              case "Java":single="C001";
5.                  break;
6.              case "大数据":single="C002";
7.                  break;
8.              case "MySQL":single="C003";
9.                  break;
10.             case "数学":single="C004";
11.                 break;
12.         }
13.         if(single.equals("")){System.out.println("无此课程");return;}
14.         Table table=connection.getTable(TableName.valueOf("student"));
15.         Scan scan=new Scan();
16.         Filter filter=new ColumnPrefixFilter(single.getBytes());
17.         scan.setFilter(filter);
18.         ResultScanner result= table.getScanner(scan);
19.         Result rs=null;
```

```
        int sum=0;
        int cnt=0;
        while((rs=result.next())!=null) {
            Cell[] cells = rs.rawCells();
            for (int i = 0; i < cells.length; i++) {
                Cell cell = cells[i];
                String val = new String(CellUtil.cloneValue(cell));
                sum+=Integer.parseInt(val);
                cnt++;
            }
        }
        System.out.println(coursename+"课程学生的总分为:"+sum+"\t 平均分为:"+(sum*1.0/cnt));
        table.close();
    }
```

运行完整代码，查询 MySQL 课程，结果如下：

图 4.1MySQL 课程学生总分和平均分

5. Hive 的安装配置（包括 MySQL 的安装配置）

验证方法：在 CentOS 7 中使用 beeline 方式连接 Hive

首先是上传并解压 hive，之后改名为 hive

```
[hyj@master soft]$ ls
hive-1.1.0-cdh5.7.0.tar.gz
[hyj@master soft]$ tar xvf hive-1.1.0-cdh5.7.0.tar.gz -C ~/
hive-1.1.0-cdh5.7.0/
```

图 5.1 上传并解压 hive 的压缩文件

然后为 hive 配置环境路径

```
[hyj@master ~]$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/hyj/.local/bin:/home/hyj/bin:/u
sr/java/jdk1.8/bin:/home/hyj/hbase/bin:/home/hyj/hadoop/bin:/home/hyj/hadoop/sbin:/home
/hyj/zookeeper/bin:/home/hyj/hive/bin
```

图 5.2 为 hive 设置环境变量

之后配置 hive-env.sh 文件和 hive-site.xml 文件，上传 mysql 的 jar 包，在虚拟机上安装相关

依赖

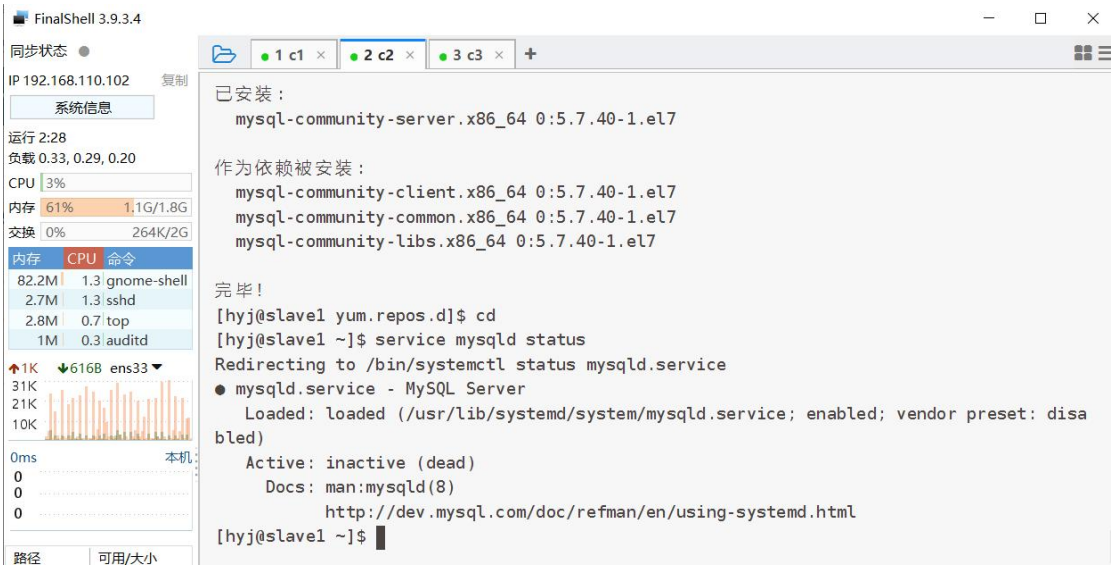

图 5.3 查看数据库状态

接着激活数据库并修改密码，修改为 Hyjnb@102

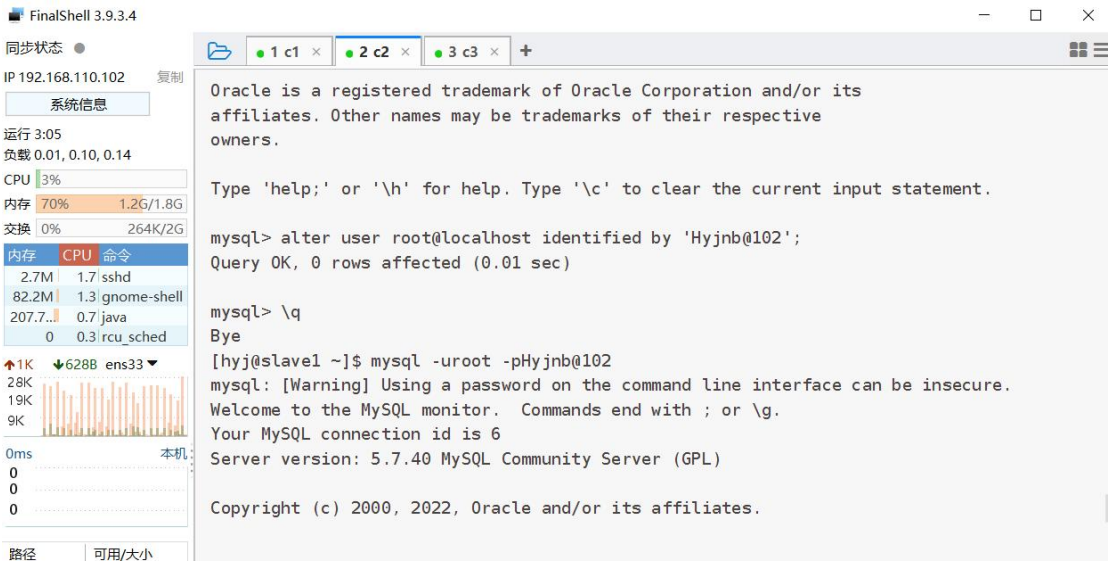

图 5.4 进入数据库并修改密码

最后用 beeline 命令进入 beeline 模式，利用！connect jdbc:hive2://master:10000 连接到 Hive

```
beeline> !connect jdbc:hive2://master:10000
scan complete in 6ms
Connecting to jdbc:hive2://master:10000
Enter username for jdbc:hive2://master:10000: hyj
Enter password for jdbc:hive2://master:10000: ******
Connected to: Apache Hive (version 1.1.0-cdh5.7.0)
Driver: Hive JDBC (version 1.1.0-cdh5.7.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://master:10000> show tables;
```

图 5.5 使用 beeline 方式连接 Hive

6. Hive 编程题

在 beeline 模式下创建数据库，命名为：hive_+自己姓名拼音缩写，将该数据文件上传到 hdfs 后，导入 music 表；编写 SQL 语句，完成以下查询：学号尾数为单数的同学统计查询每首歌的播放次数学号尾数为双数的同学统计查询每种播放端的播放次数每个步骤均要求完整截图。

先通过 beeline 连接 Hive，进入 beeline 模式

```
[hyj@master ~]$ beeline
2022-12-11 15:35:53,403 WARN  [main] mapreduce.TableMapReduceUtil: The hbase-prefix-tre
e module jar containing PrefixTreeCodec is not present.  Continuing without it.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hyj/hbase/lib/slf4j-log4j12-1.7.5.jar!/org/slf4
j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hyj/hadoop/share/hadoop/common/lib/slf4j-log4j1
2-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2022-12-11 15:35:53,800 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoo
p library for your platform... using builtin-java classes where applicable
Beeline version 1.1.0-cdh5.7.0 by Apache Hive
```

图 6.1 使用 beeline 连接

之后在 beeline 模式通过 nohup hiveserver2 > hs2.log 2>&1 &语句连接数据库

```
beeline> !connect jdbc:hive2://master:10000
scan complete in 4ms
Connecting to jdbc:hive2://master:10000
Enter username for jdbc:hive2://master:10000: hyj
Enter password for jdbc:hive2://master:10000: ******
Connected to: Apache Hive (version 1.1.0-cdh5.7.0)
Driver: Hive JDBC (version 1.1.0-cdh5.7.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
```

图 6.2 在 beeline 模式中连接数据库

进入数据库后按要求创建并查看数据库

```
0: jdbc:hive2://master:10000> create database hive_hyj;
No rows affected (1.159 seconds)
0: jdbc:hive2://master:10000> show databases;
+----------------+--+
| database_name  |
+----------------+--+
| default        |
| demo           |
| hive_hyj       |
```

图 6.3 查看新建的 hive_hyj 数据库

接着进入创建的 hive_hyj 数据库，按照给定数据类型和题目要求创建表格，将事先上传在 HDFS 上的 music.txt 导入到新创建的数据库中

```
0: jdbc:hive2://master:10000> use hive_hyj;
No rows affected (0.149 seconds)
0: jdbc:hive2://master:10000> create table music(
0: jdbc:hive2://master:10000> id int,
0: jdbc:hive2://master:10000> song string,
0: jdbc:hive2://master:10000> singer string,
0: jdbc:hive2://master:10000> sex string,
0: jdbc:hive2://master:10000> rhythm string,
0: jdbc:hive2://master:10000> player string)
0: jdbc:hive2://master:10000> row format delimited fields terminated by '\t';
No rows affected (1.582 seconds)
0: jdbc:hive2://master:10000> load data inpath 'hdfs://master:8020/homework/music.txt'
into table music;
No rows affected (1.395 seconds)
```

图 6.4 创建并导入 music 表

使用 SQL 语句查看表内容

```
0: jdbc:hive2://master:10000> select * from music;
+-----------+------------+--------------+------------+--------------+--------------
+--+
| music.id  | music.song  | music.singer  | music.sex  | music.rhythm  | music.player
|
+-----------+------------+--------------+------------+--------------+--------------
+--+
| 1         | song1       | singer1       | man        | slow          | pc
|
| 2         | song2       | singer2       | woman      | quick         | android
|
| 3         | song3       | singer3       | man        | quick         | ios
|
| 4         | song4       | singer4       | man        | slow          | pc
|
| 5         | song5       | singer5       | woman      | quick         | android
|
| 6         | song6       | singer6       | man        | quick         | ios
```

图 6.5 查看 music 表中内容

利用 SQL 语句查询每个播放器的使用次数

```
0: jdbc:hive2://master:10000> select player,count(1) from music group by player;
+----------+------+--+
|  player  | _c1  |
+----------+------+--+
| android  | 11   |
| ios      | 10   |
| pc       | 9    |
+----------+------+--+
3 rows selected (4.712 seconds)
```

图 6.6 查询不同播放器的使用次数

三、实验总结

（可以总结实验中出现的问题以及解决的思路，也可以列出没有解决的问题）

1. 在第一次用 mysql -uroot -p*****命令进入数据库时，密码中的符号前要加’\’转义，否则会报错

2. 在将 IDEA 上写的代码到虚拟机上运行时，每次都要先 clean，然后在 package，最后将打包好的文件上传