# 实验二——MapReduce分布式数据处理

姓名：陈朴炎 学号：2021211138

目录

[实验二——MapReduce分布式数据处理 1](#_Toc164716755)

[1 实验流程 2](#_Toc164716756)

[2 实验结果与分析 18](#_Toc164716757)

[2.1 jar包生成图 18](#_Toc164716758)

[2.2 执行命令截图 18](#_Toc164716759)

[2.3 步骤8执行命令结果图 19](#_Toc164716760)

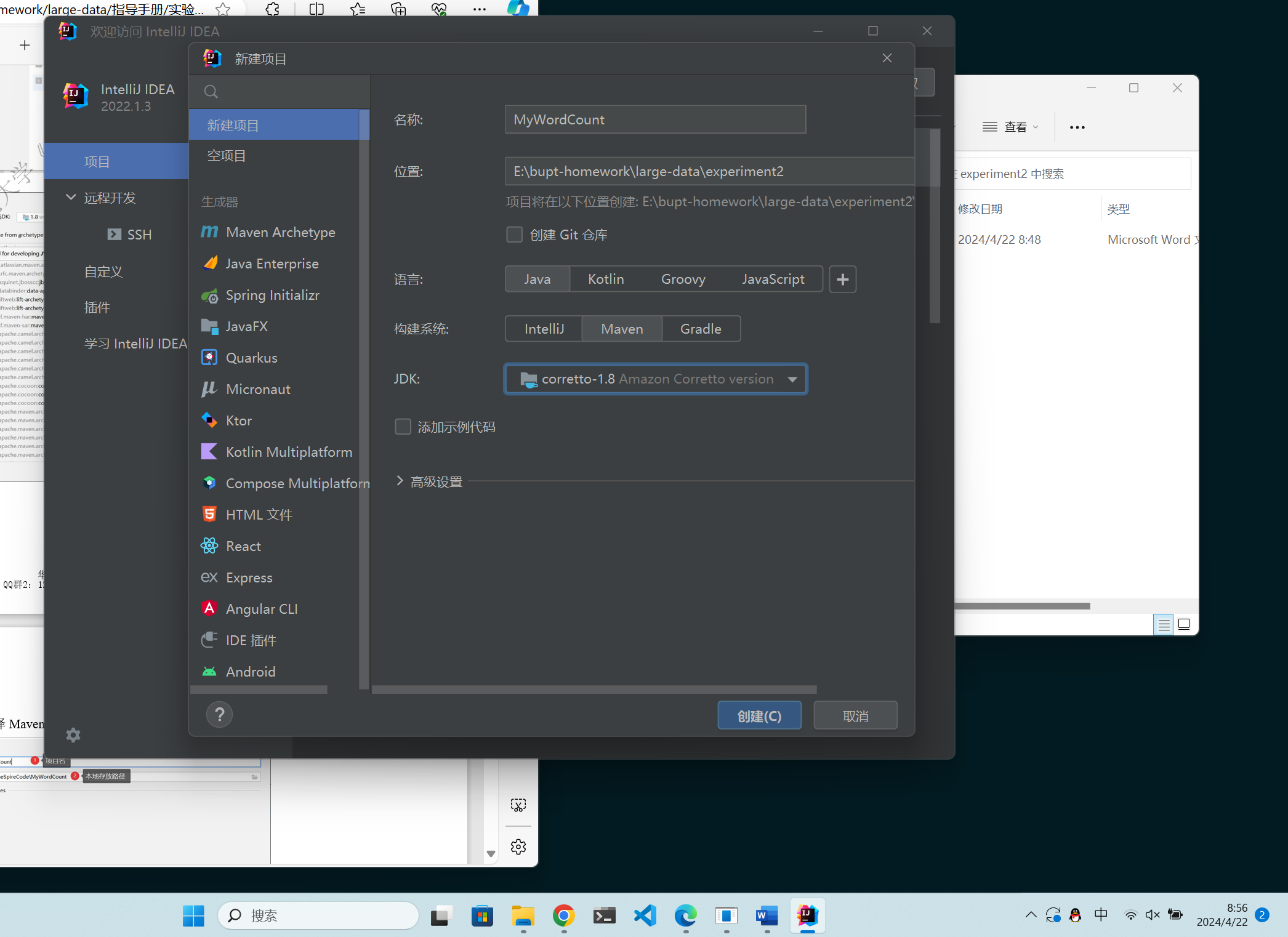
[2.4 步骤9 执行命令结果图 20](#_Toc164716761)

[2.5 提交jar包（另附件） 21](#_Toc164716762)

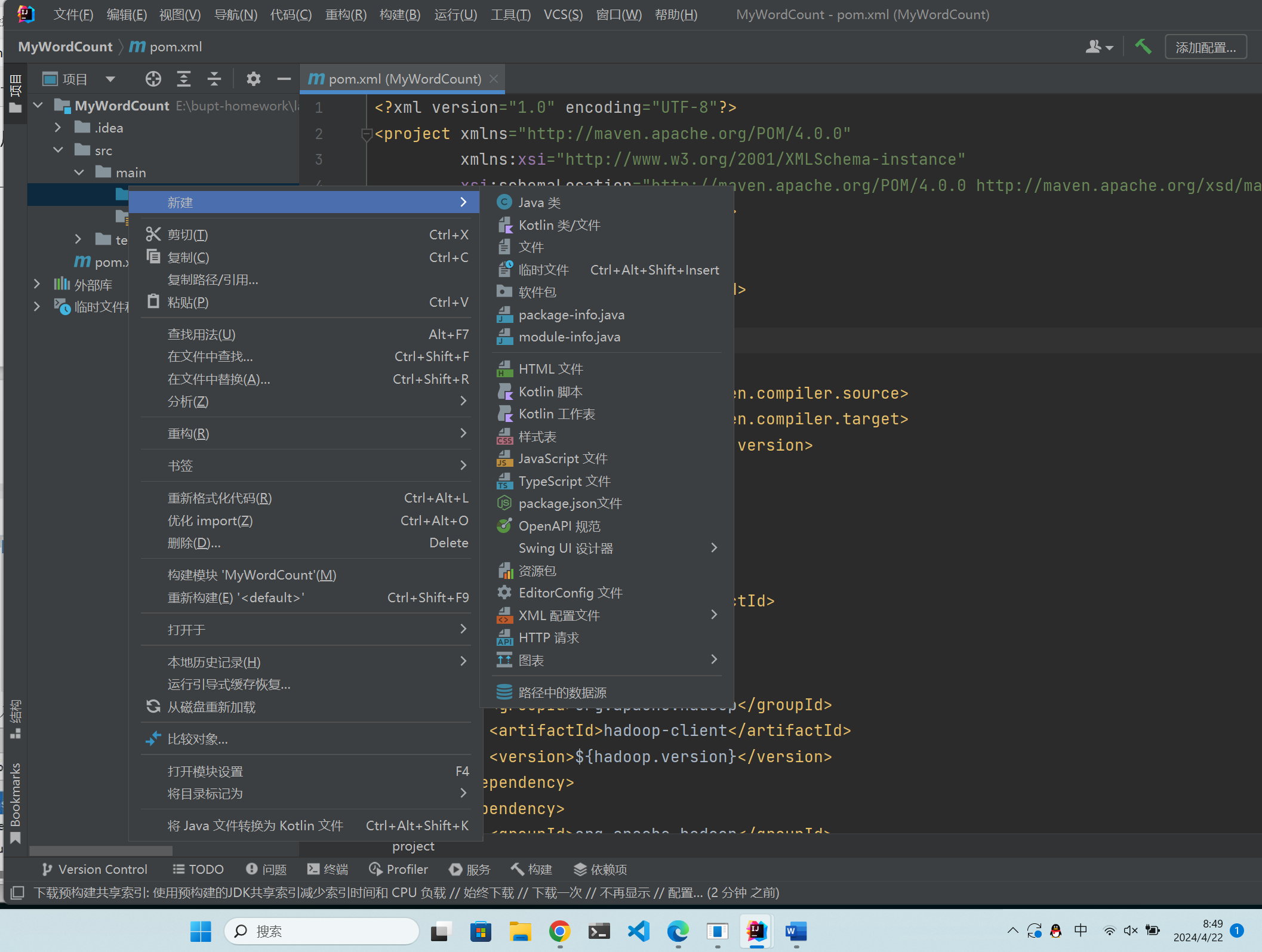
[2.6 解释WordCount程序代码 21](#_Toc164716763)

## 1 实验流程

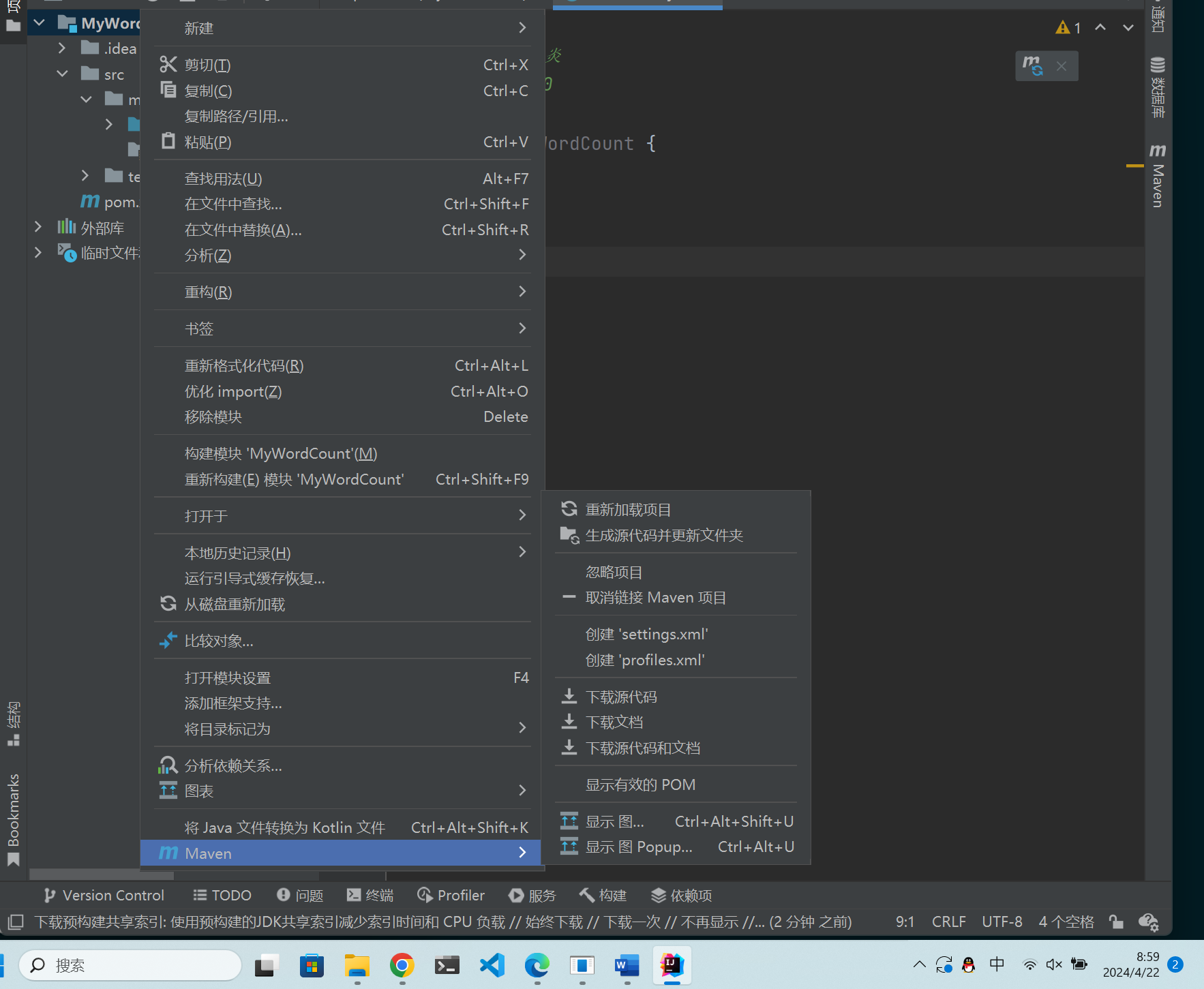
使用IDEA新建一个java项目，jdk选择1.8



新建一个MyWordCount项目，修改里面的pom.xml文件，将实验一的配置信息复制到这里面。并新建一个java类名为WordCount



修改完pom.xml配置文件后，重新构建项目



WordCount类中添加如下代码

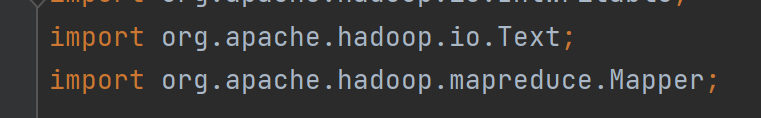
这个类的作用是将输入的文本数据切分成单词，并为每个单词输出一个键值对，其中键是单词本身，值是1，表示该单词出现了一次。

1. **public** **class** WordCount {
2. **public** **static** **class** TokenizerMapper **extends** Mapper<Object, Text, Text, IntWritable>{
3. **private** **final** **static** IntWritable one = **new** IntWritable(1);
4. **private** Text word = **new** Text();
5. **public** **void** map(Object key, Text value, Context context)
6. **throws** IOException, InterruptedException {
7. StringTokenizer itr = **new** StringTokenizer(value.toString());
8. **while** (itr.hasMoreTokens()) {
9. word.set(itr.nextToken());
10. context.write(word, one);
11. }
12. }
13. }
15. }

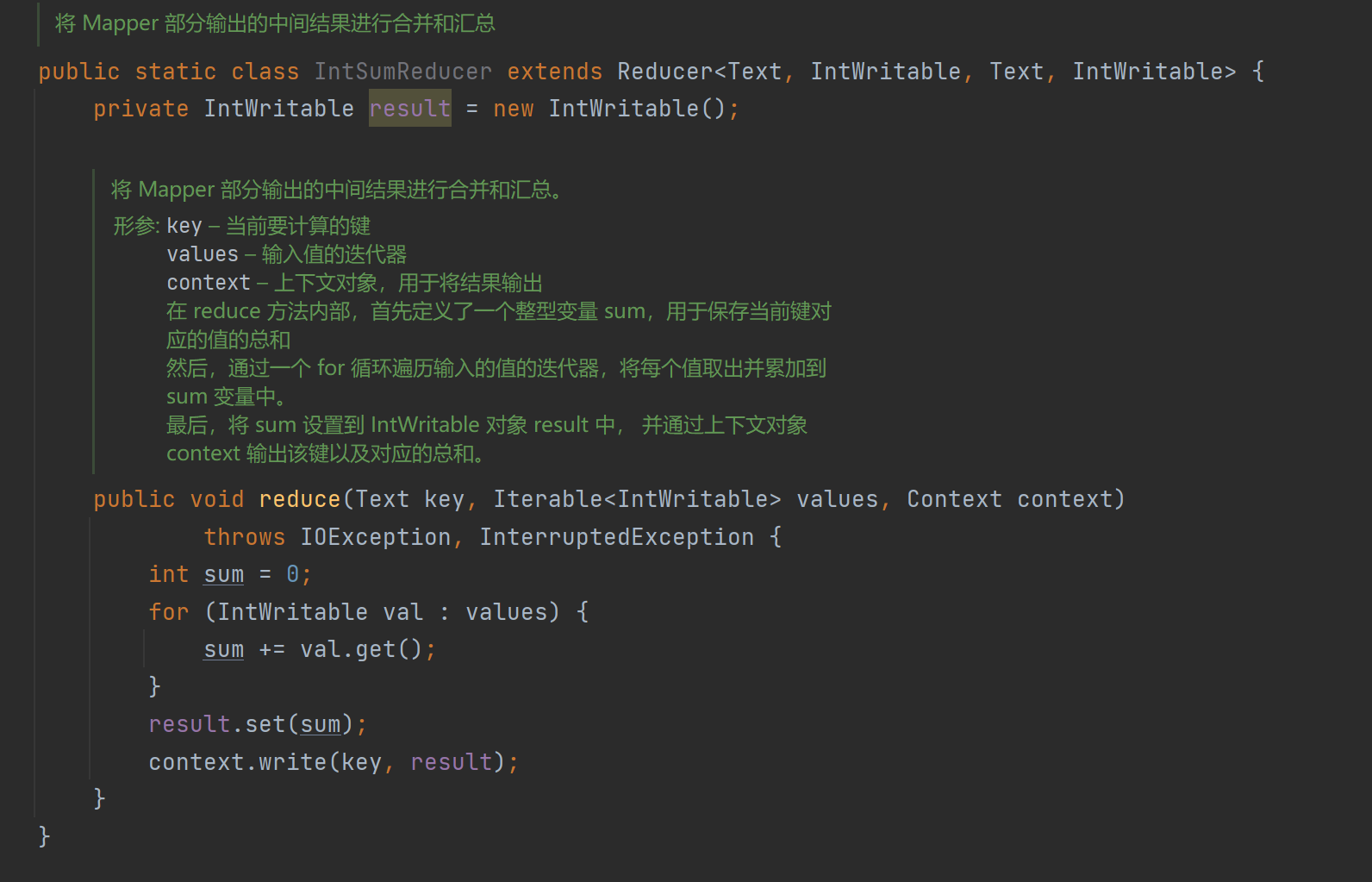
但是很可惜，会有错误，原因是Text类在很多包里面都有，项目没办法自己选择所需要的包，需要我们手动添加。



导入以下两个包后红色报错消失



之后再创建一个Reducer对象，继承hadoop的Reducer，用来将Mapper的输出结构进行合并和总结，如下图所示。



类和函数说明：

在reduce函数作为Reducer类的主要函数，它的功能是将 Mapper 部分输出的中间结果进行合并和汇总。在reduce函数中，首先定义了一个整型变量 sum，用于保存当前键对应的值的总和。然后，通过一个 for 循环遍历输入的值的迭代器，将每个值取出并累加到 sum 变量中。最后，将 sum 设置到 IntWritable 对象 result 中， 并通过上下文对象 context 输出该键以及对应的总和。

上面两个类编写完成之后，可以开始写WordCount的主函数了。

主函数的运行逻辑为：

1. 判断输入的参数格式是否正确

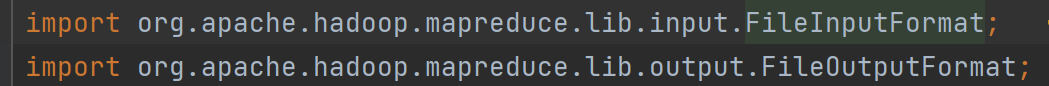
2. 创建job对象，并为job对象初始化

3. 设置输出的Key和Value

4. 设置输入输出的路径

在主函数中，需要先new一个Configuration对象。需要注意的是，当前这个Configuration对象应该选择apache.hadoop.conf.Configuration类。

在第四步的时候，要注意FileInputFormat和FileOutputFormat应该选择导入的包是以下这两个，不然会报错的。

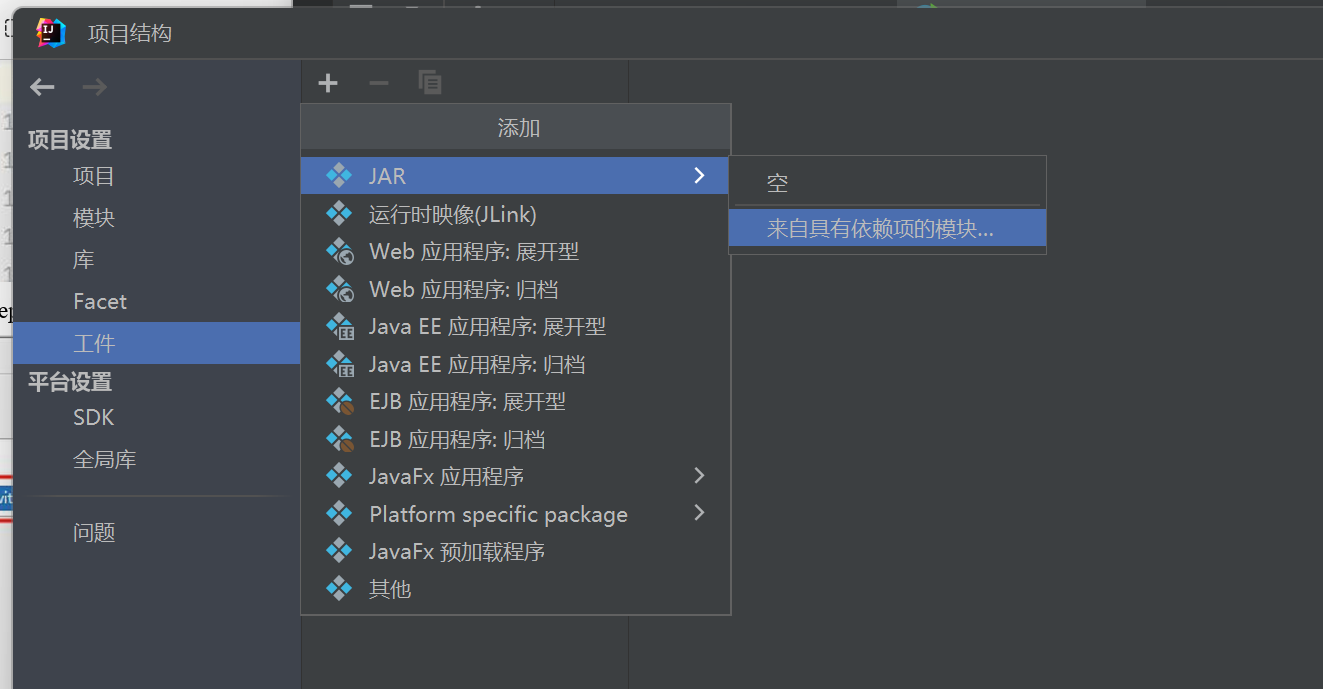


main函数如下所示：

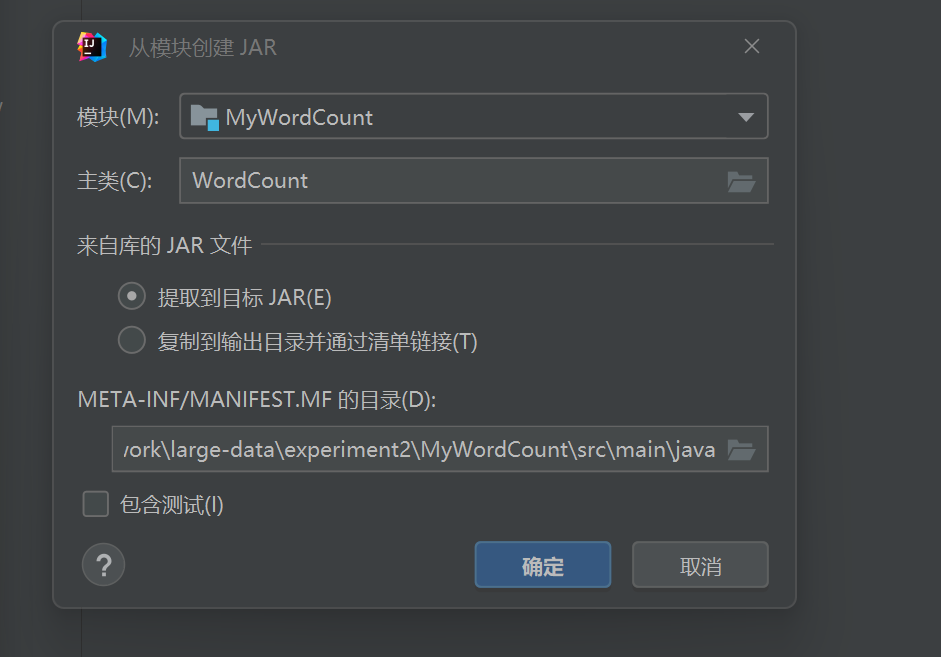
1. **public** **static** **void** main (String [] args) **throws** Exception{
3. Configuration conf = **new** Configuration();
4. // 限定输出参数必须为2个
5. String []otherArgs = **new** GenericOptionsParser(conf, args).getRemainingArgs();
7. **if**(otherArgs.length!=2){
8. System.err.println("Please usage: word-count <in> <out>");
9. System.exit(2);
10. }
12. // 创建job对象
13. Job job = **new** Job(conf, "word count");
15. // 初始化job对象
16. job.setJarByClass(WordCount.**class**);
17. job.setMapperClass(TokenizerMapper.**class**);
18. job.setCombinerClass(IntSumReducer.**class**);
19. job.setReducerClass(IntSumReducer.**class**);
21. // 设置输出格式
22. job.setOutputKeyClass(Text.**class**);
23. job.setOutputValueClass(IntWritable.**class**);
25. // 设置输入输出
26. FileInputFormat.addInputPath(job,**new** Path(otherArgs[0]));
27. FileOutputFormat.setOutputPath(job, **new** Path(otherArgs[1]));
28. System.exit(job.waitForCompletion(**true**)?0:1);
29. }

接下来应该要打包项目并运行了。

在文件->项目结构中，选择工件，添加，如下图所示

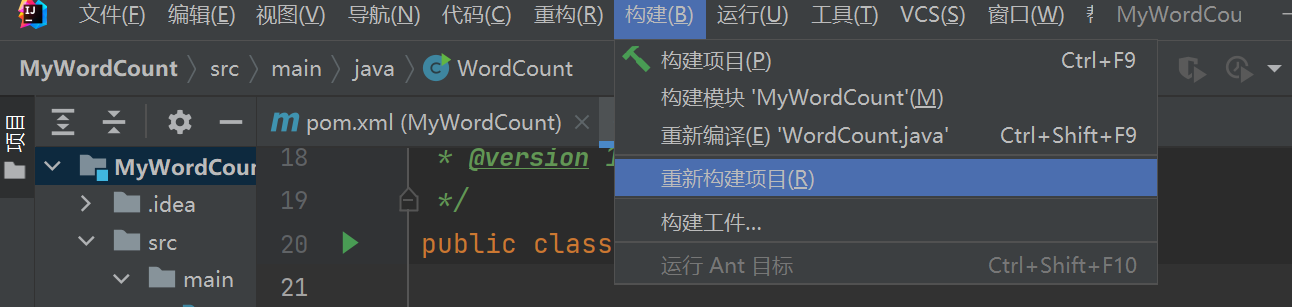


并将主类设置为WordCount，如下图



之后选择应用，确定即可。

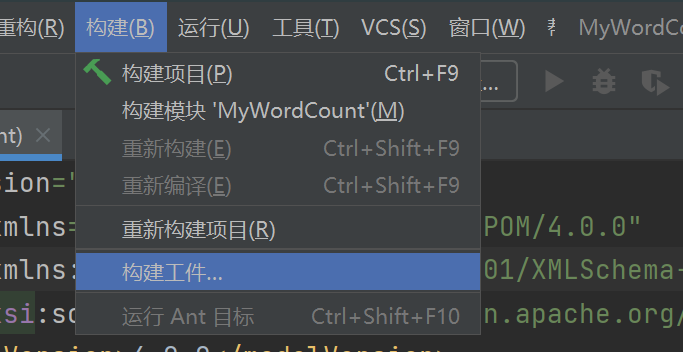
选择 构建 -> 重新构建项目



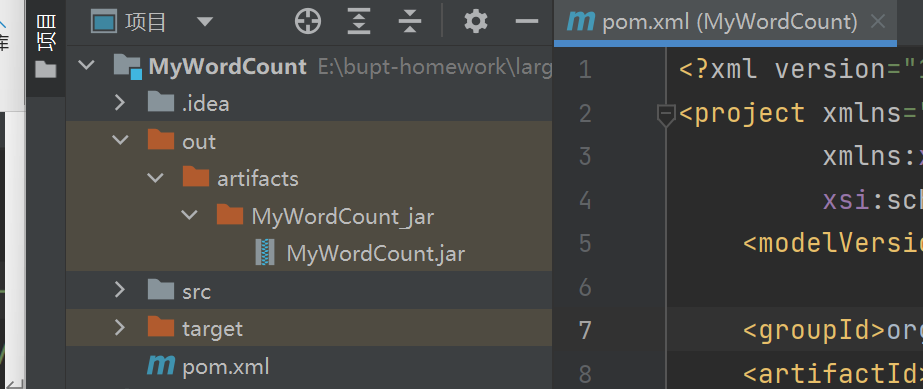
但是这里我没能成功出现out包。

原因是没有构建工件。

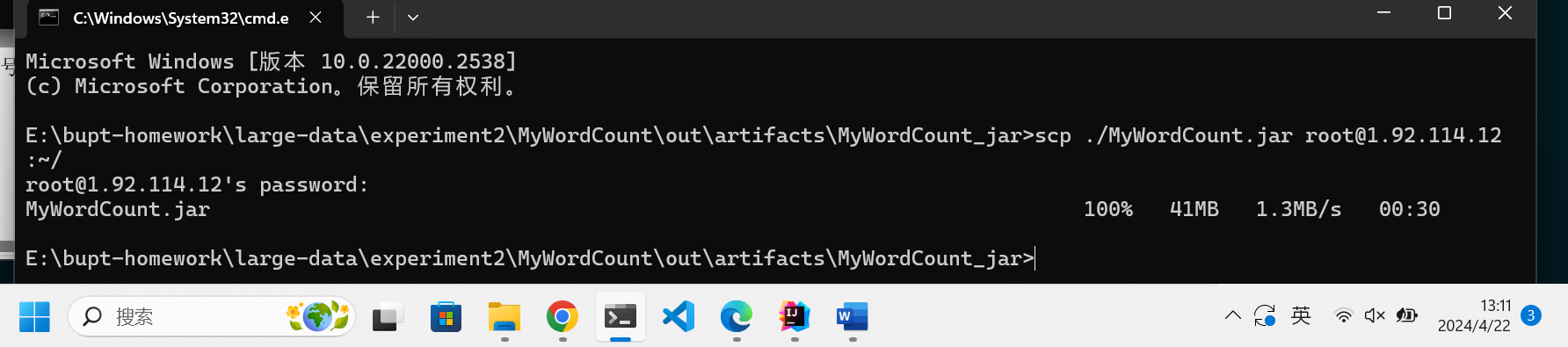
重新构建工件，如下图所示

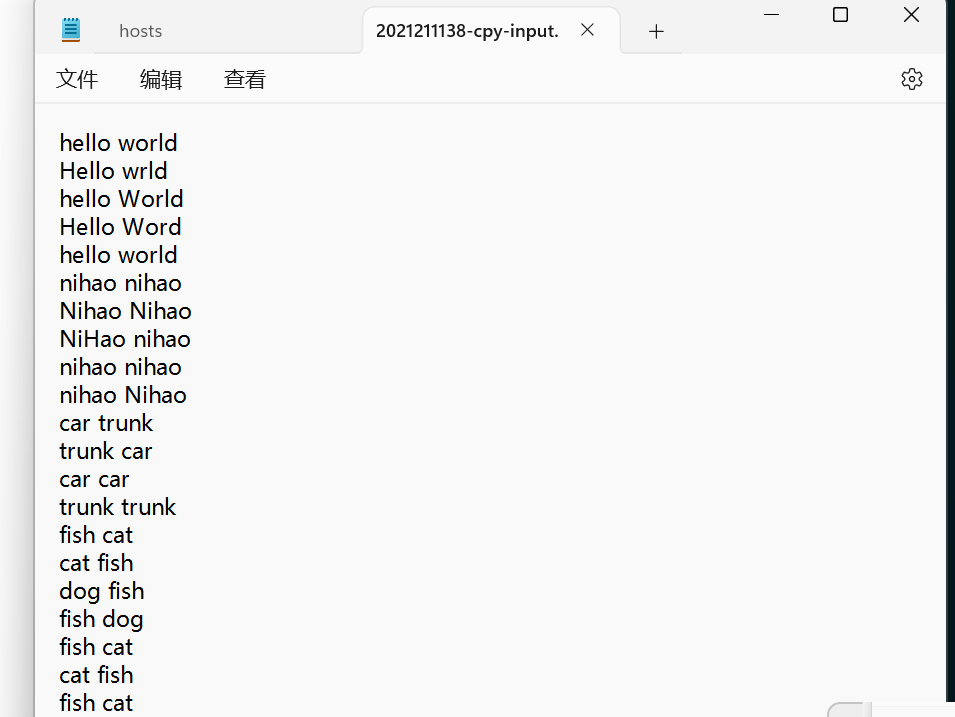


选择构建工件 -> MyWordCount -> 构建，就出现out包了



使用scp命令将这个jar文件发给服务器

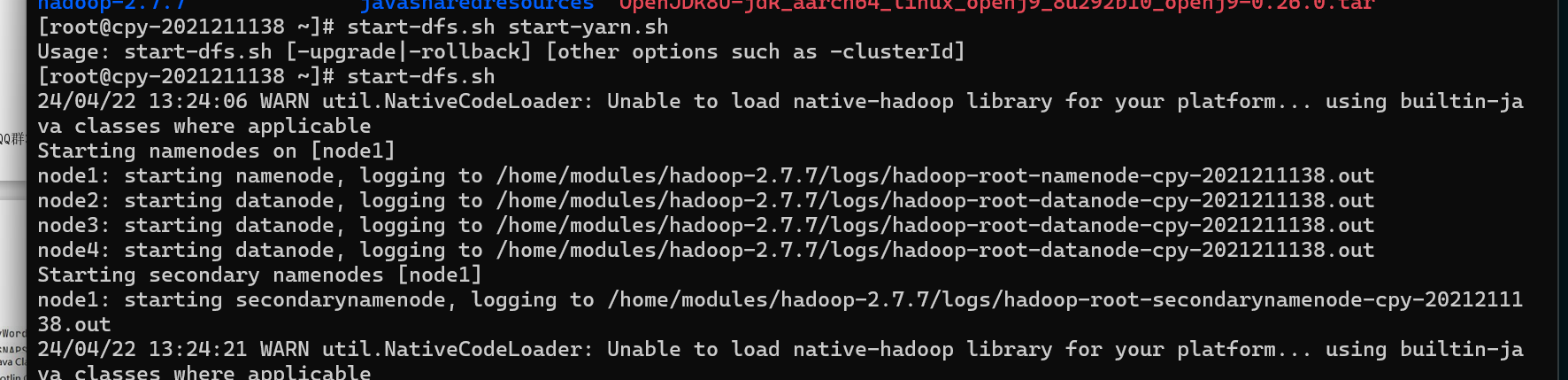


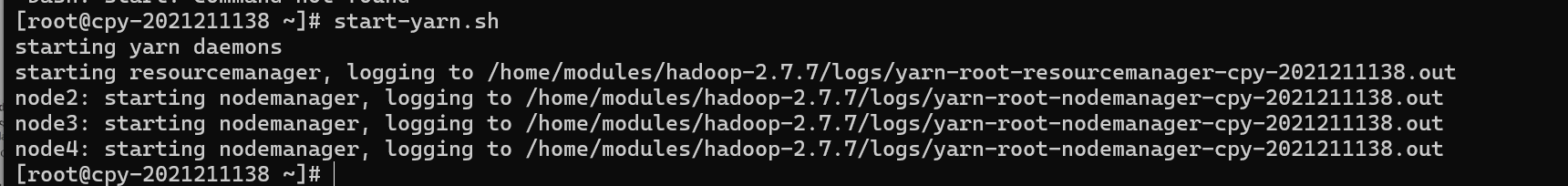


同样的，把这个输入文件传送给服务器

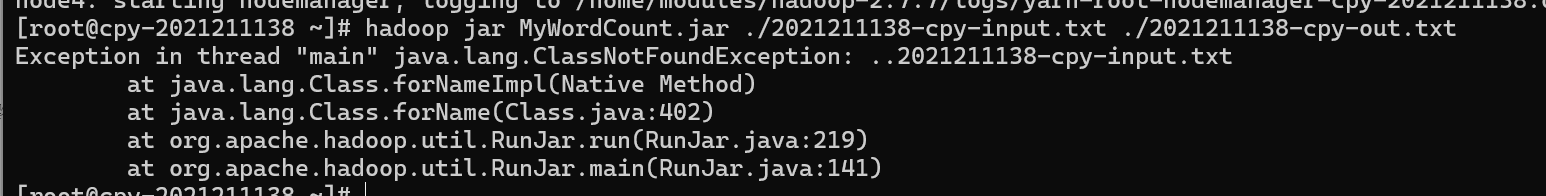


连上四台服务器，启动hadoop和yarn

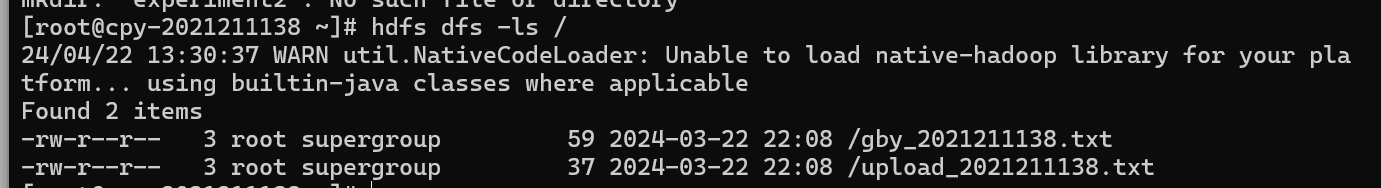




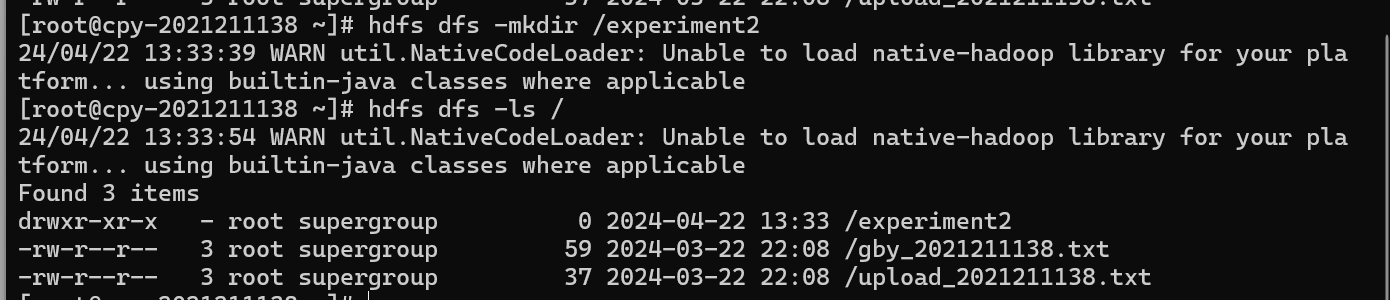
尝试运行jar包，主函数抛出异常



我觉得应该是hdfs找不到文件导致的

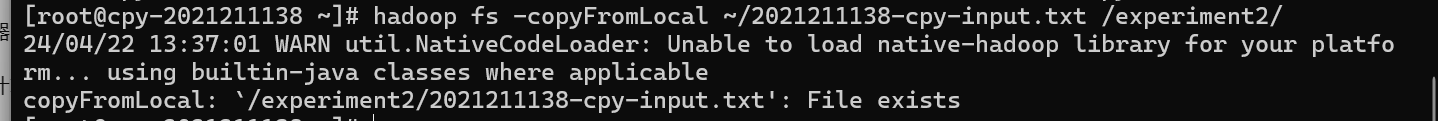


使用hdfs dfs -ls / 查看hdfs系统根目录 / 文件夹下的内容，发现并没有我们想要的输入文件。

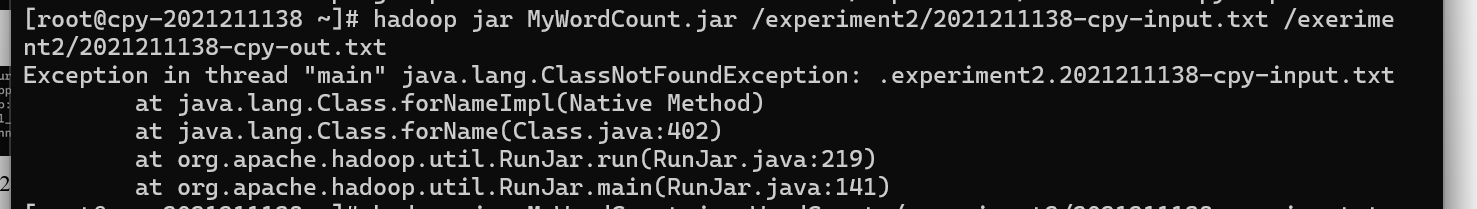


使用hdfs dfs -mkdir /experiment2，创建一个hdfs文件夹

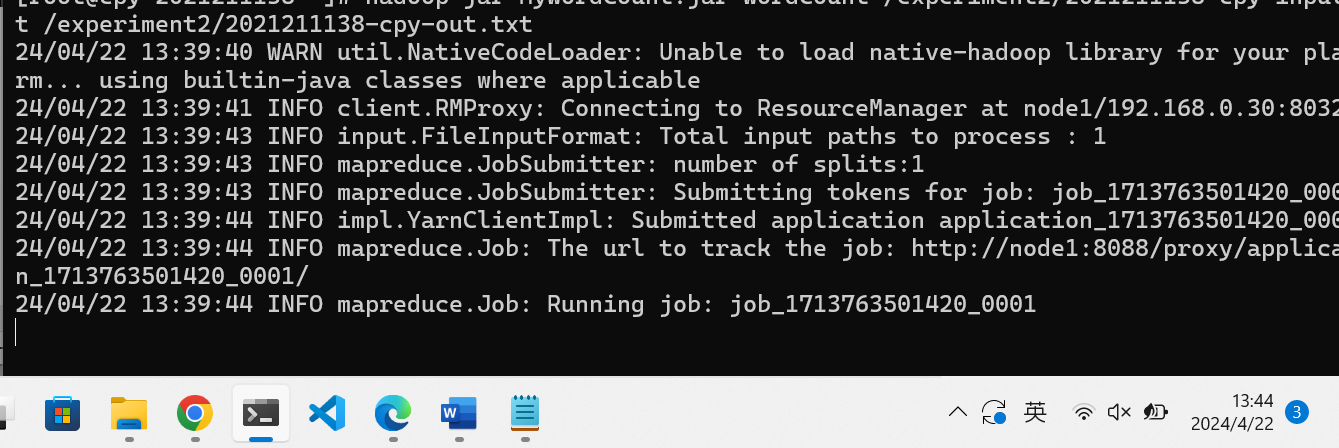
使用hadoop fs -copyFromLocal <localSrc> <hdfsDst>将本地文件拷到hdfs文件系统的路径下



又报出如下错误

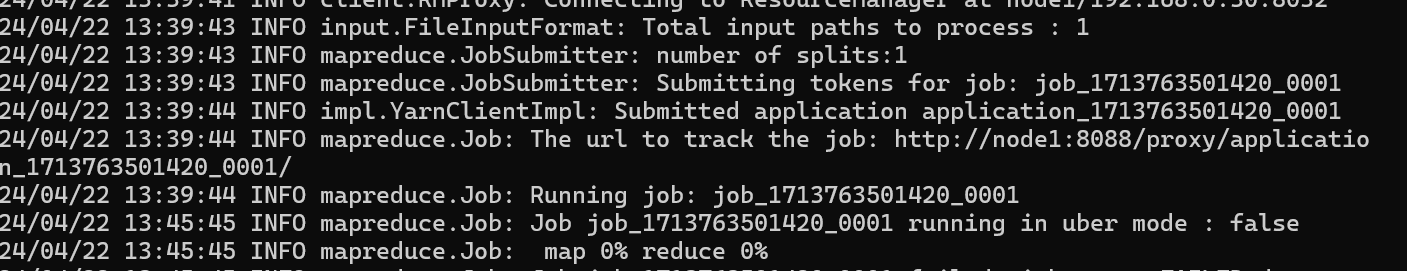


原因是hadoop jar 命令的格式应该是 hadoop jar <JAR 文件路径> <主类> [参数]，而我的命令中缺少了主类的指定。所以应该将命令改为如下所示。



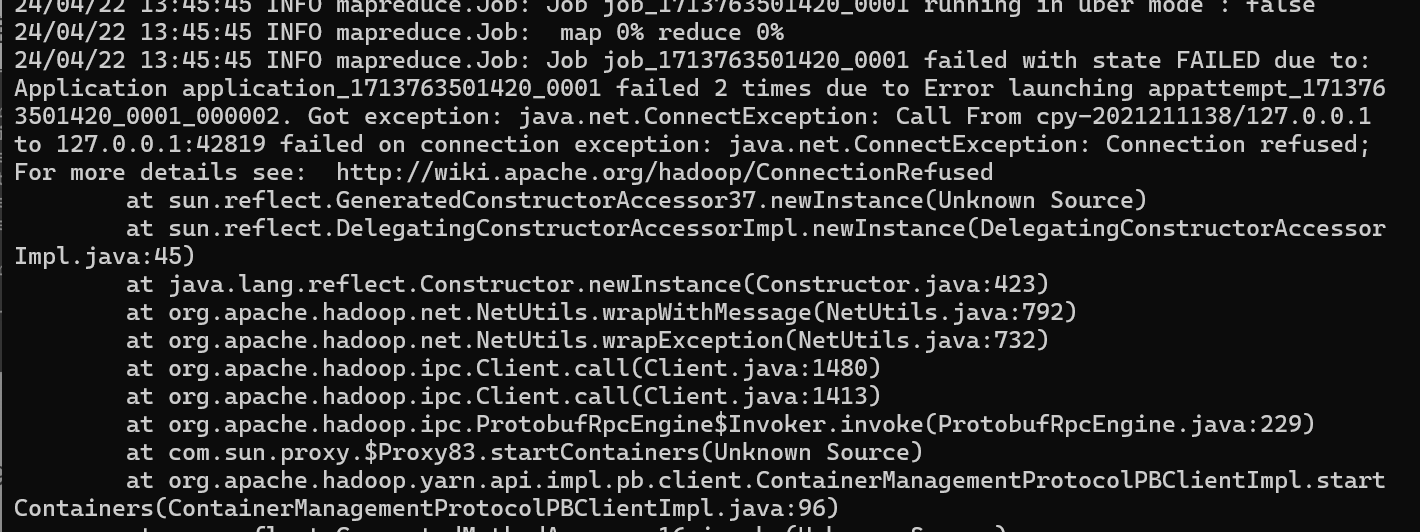
但是可以从时间戳上看到卡住了从39分到44分，很久都没有动静。

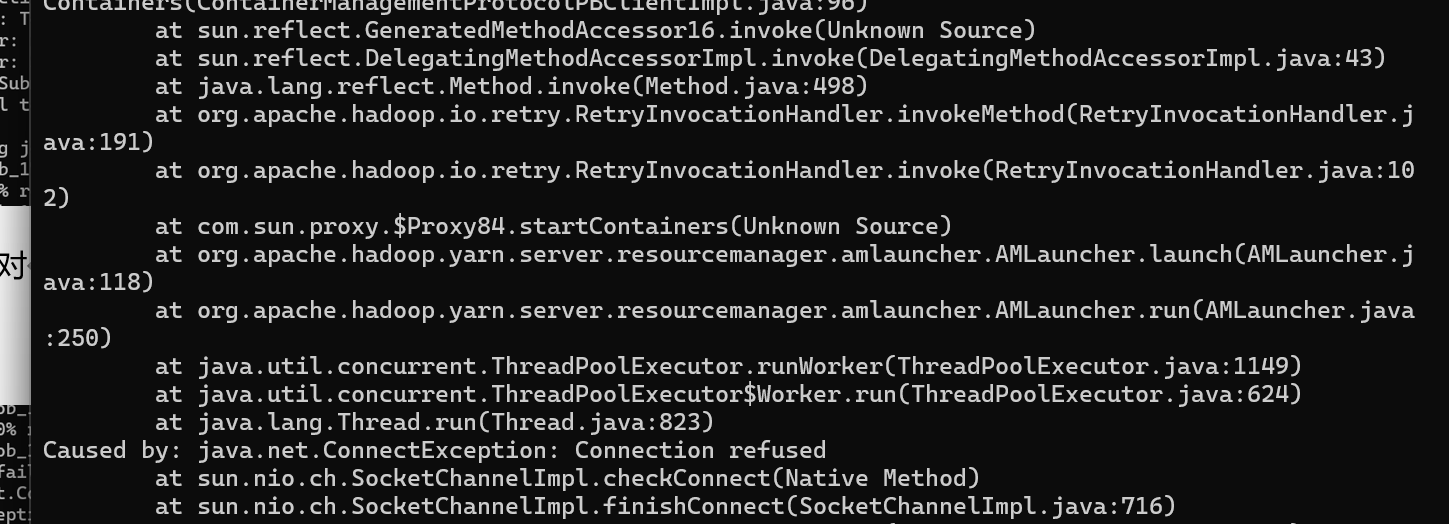
一直到45分，又有动静了

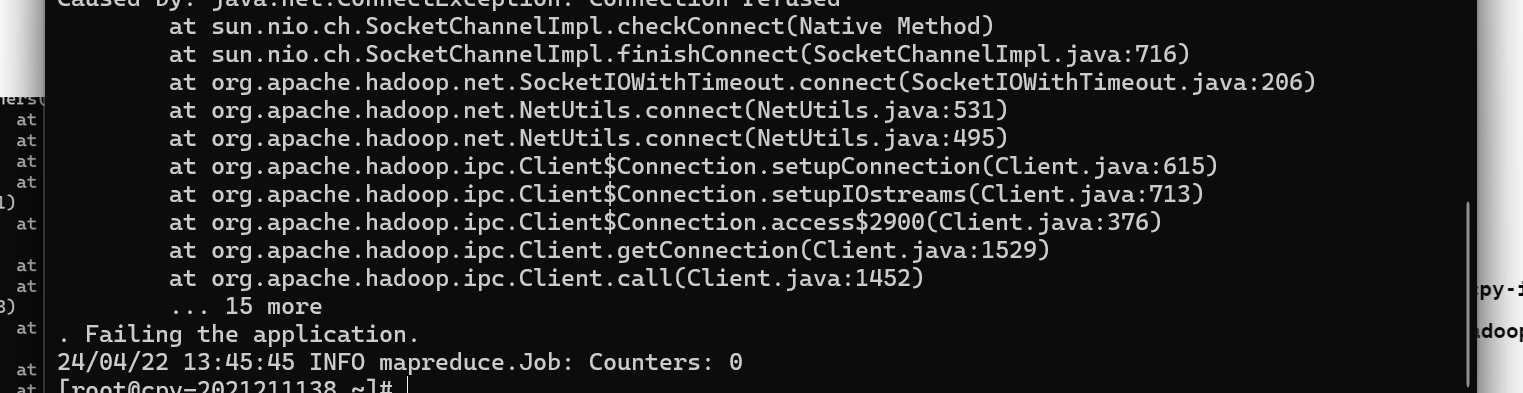


但是map 0% reduce 0%，显然不对

剩下的信息如下图所示



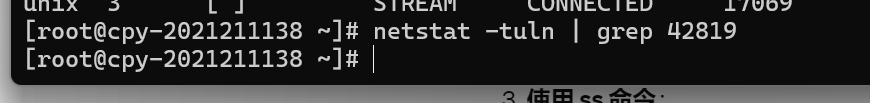




从下面这句话中，能得出，连接到42819端口失败了。

24/04/22 13:45:45 INFO mapreduce.Job: Job job\_1713763501420\_0001 failed with state FAILED due to: Application application\_1713763501420\_0001 failed 2 times due to Error launching appattempt\_1713763501420\_0001\_000002. Got exception: java.net.ConnectException: Call From cpy-2021211138/127.0.0.1 to 127.0.0.1:42819 failed on connection exception: java.net.ConnectException: Connection refused;

之后我使用netstat命令，并没有看到42819端口打开的信息，我觉得应该是端口没打开导致的。



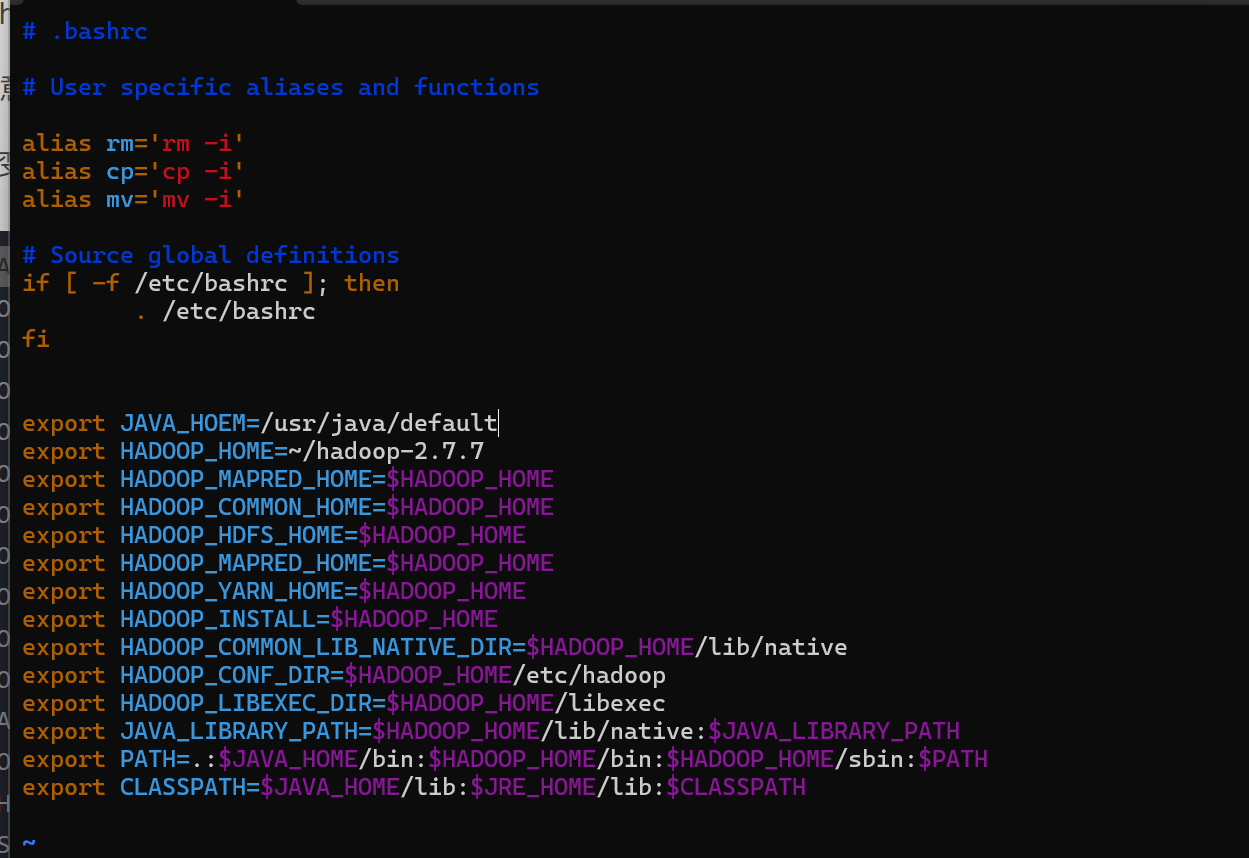
但是重新尝试了很多次每次的端口都不一样，应该不是端口的问题

我查阅了相关资料，有用的如下：

<https://blog.csdn.net/u014646662/article/details/82890443>

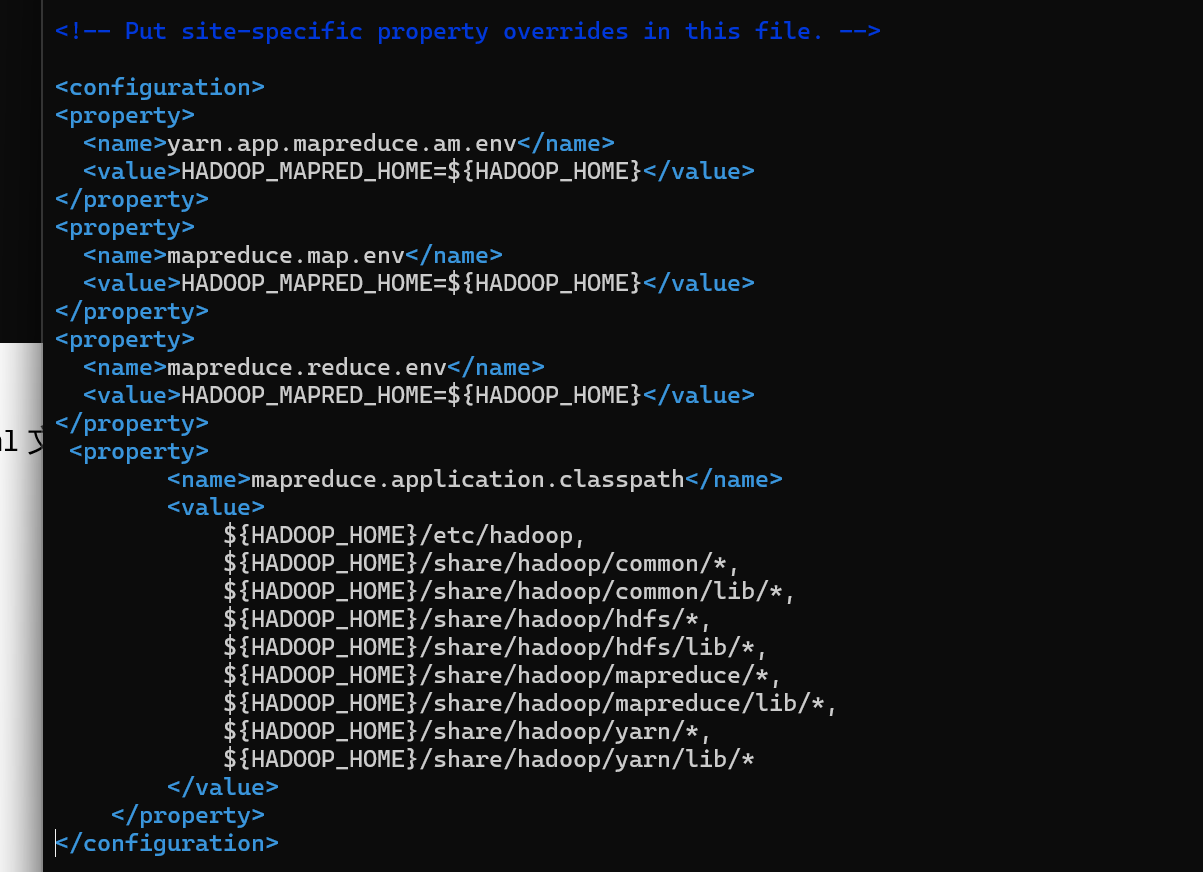
对于我这个问题，应该先配置环境变量

首先输入 vim ~/.bashrc，在最后面添加几行，如下



然后在hadoop-2.7.7/etc/hadoop 中添加一个mapred-site.xml文件

文件内容如下：



这个文件是为了配置hadoop框架的相关参数的

yarn.app.mapreduce.am.env: 这个属性用于配置 MapReduce 应用程序的 ApplicationMaster 环境变量。在这里，通过设置 HADOOP\_MAPRED\_HOME 为 ${HADOOP\_HOME}，将 MapReduce 应用程序的环境变量指向 Hadoop 安装目录。

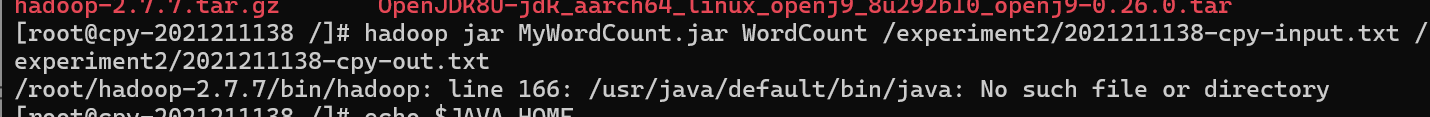
mapreduce.map.env: 这个属性用于配置 Map Task 执行器的环境变量。同样地，通过设置 HADOOP\_MAPRED\_HOME 为 ${HADOOP\_HOME}，将 Map Task 执行器的环境变量指向 Hadoop 安装目录。

mapreduce.reduce.env: 这个属性用于配置 Reduce Task 执行器的环境变量，同样地，通过设置 HADOOP\_MAPRED\_HOME 为 ${HADOOP\_HOME}，将 Reduce Task 执行器的环境变量指向 Hadoop 安装目录。

mapreduce.application.classpath: 这个属性用于配置 MapReduce 应用程序的类路径，以便能够加载所需的类和库文件。在这里，指定了各种 Hadoop 相关的目录和库文件的路径，以确保 MapReduce 任务能够正常运行。

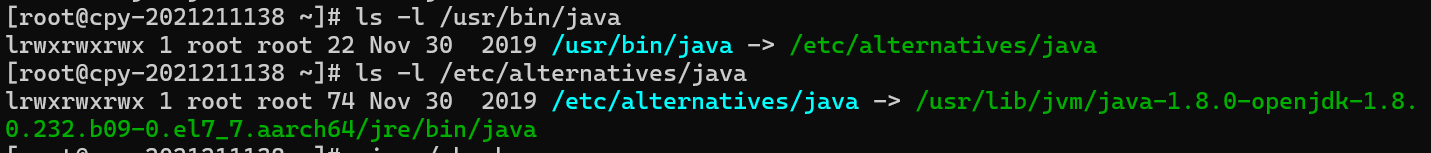
保存后，执行source ~/.bashrc 激活新的环境变量

但是这个时候又报错了，如下：

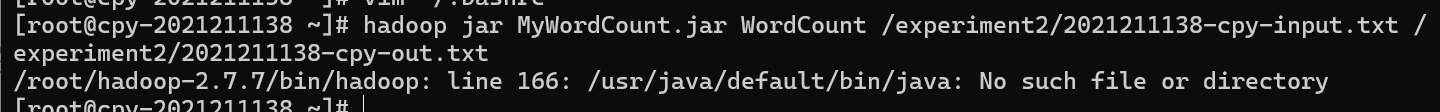


找不到/usr/java/deault/ 路径

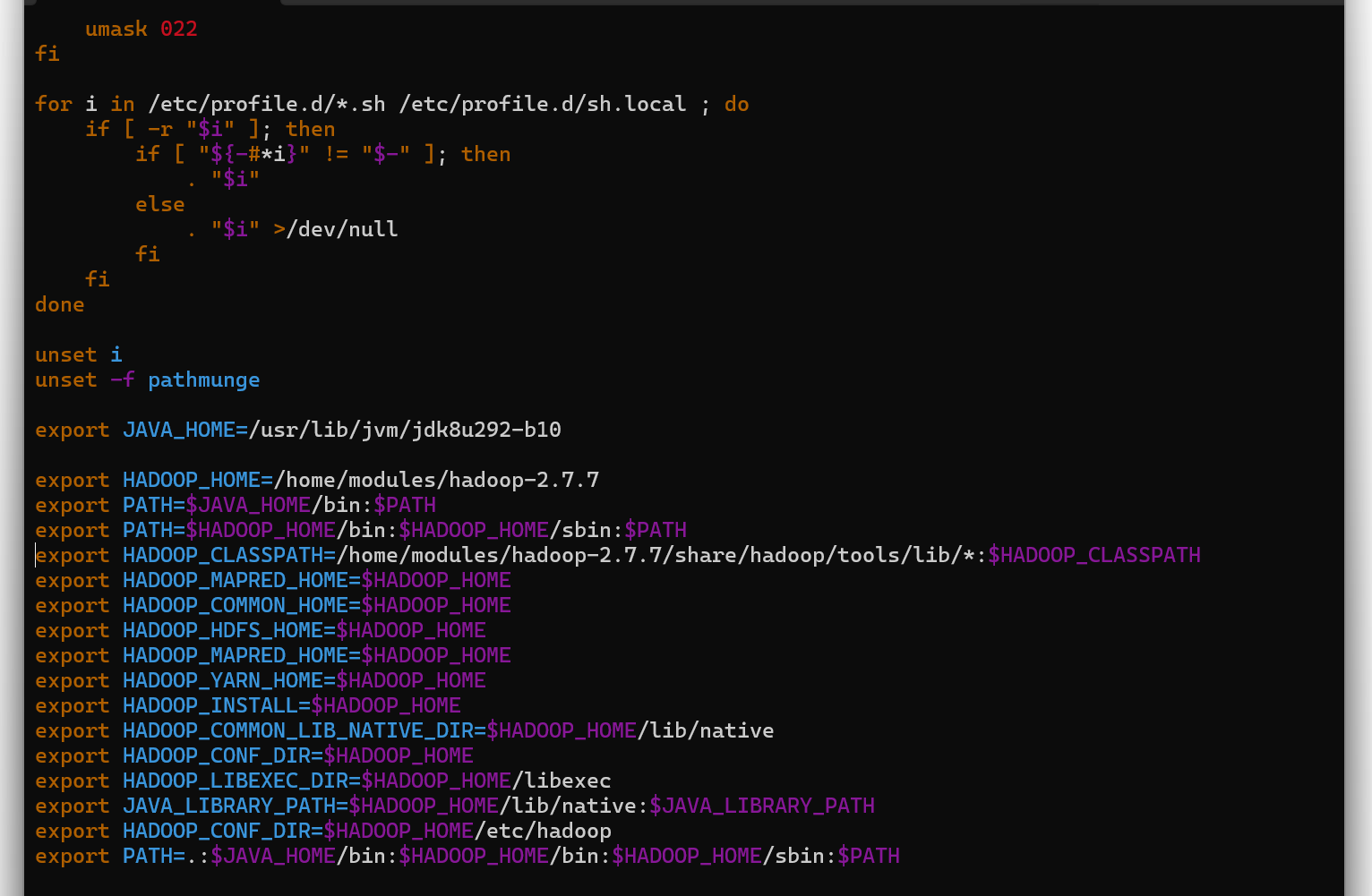
我通过如下命令找到了java的安装路径



修改了~/.bashrc的JAVA\_HOME后，还是不行



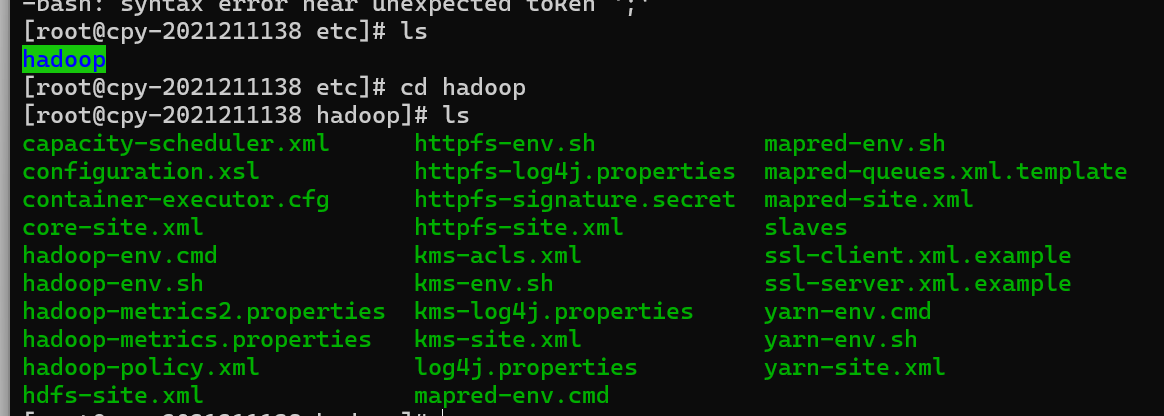
尝试修改 /etc/profile



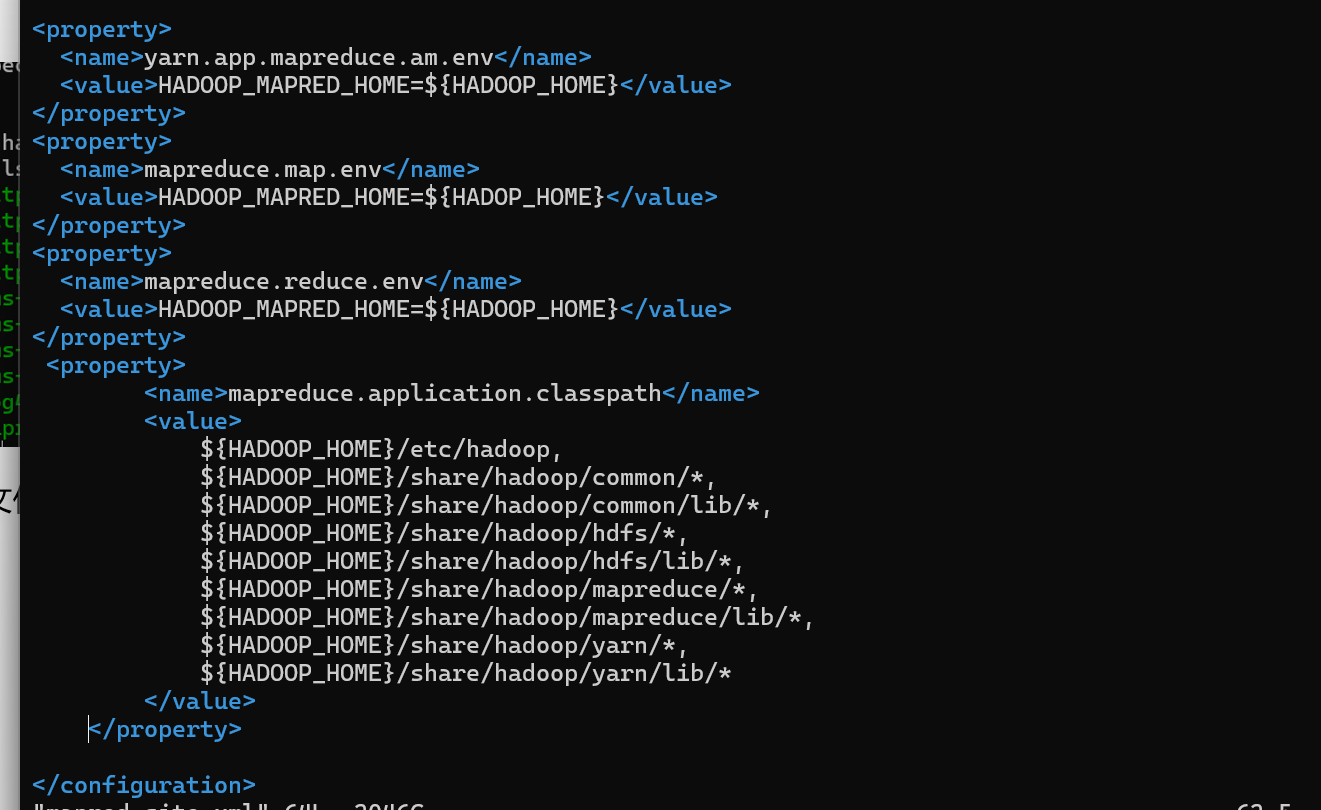
使用source /etc/profile重新激活

现在进入到HADOOP\_HOME目录中，如上图我的是/home/modules/hadoop-2.7.7

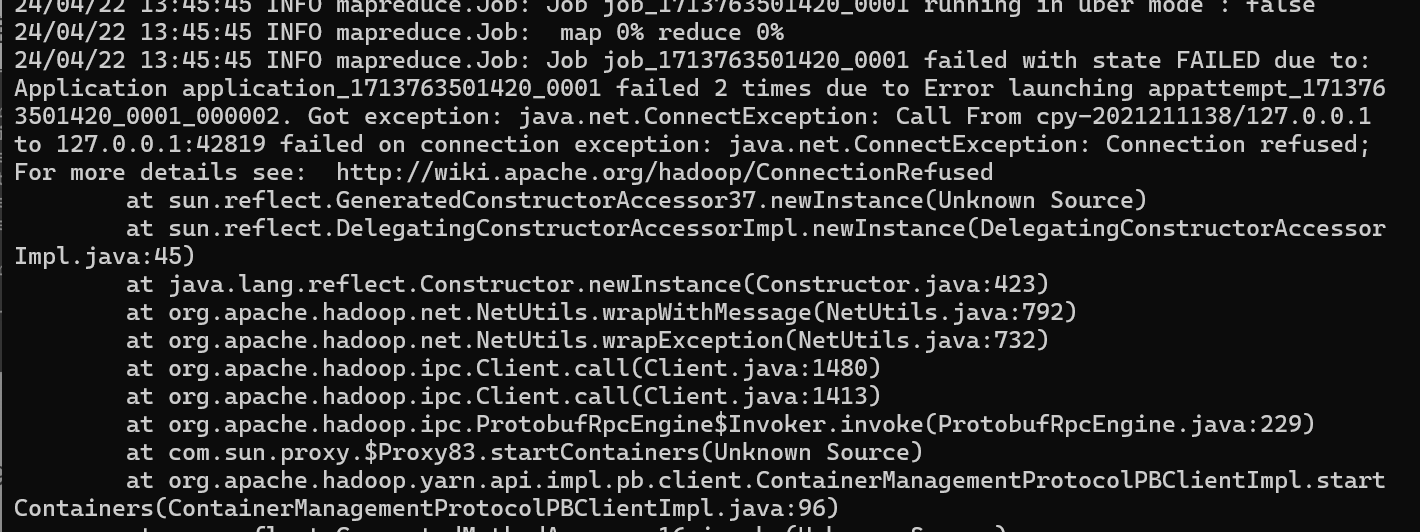
进入到etc/hadoop中



修改mapred-site.xml文件，添加如下信息

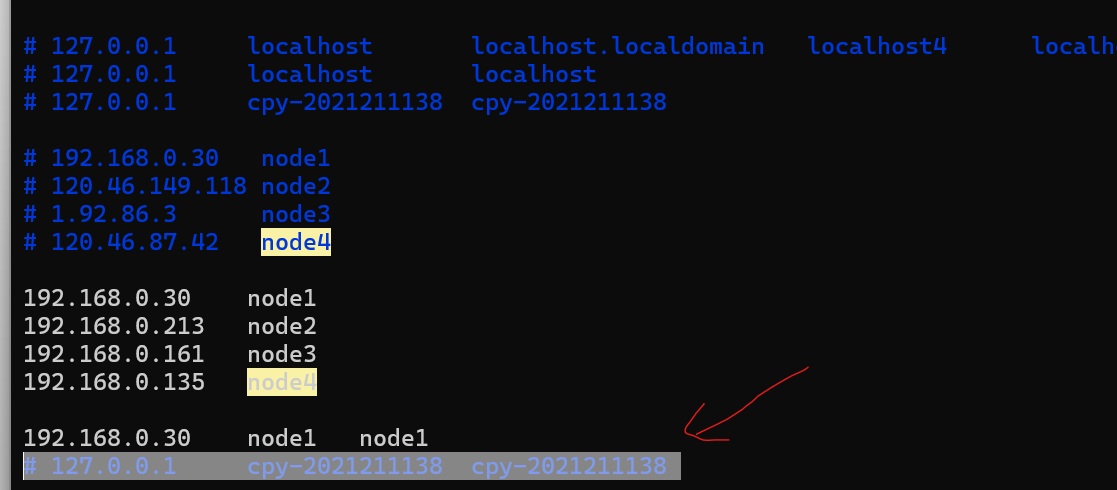


重新执行jar指令，但还是不行。这时候我认真看了一下报错信息

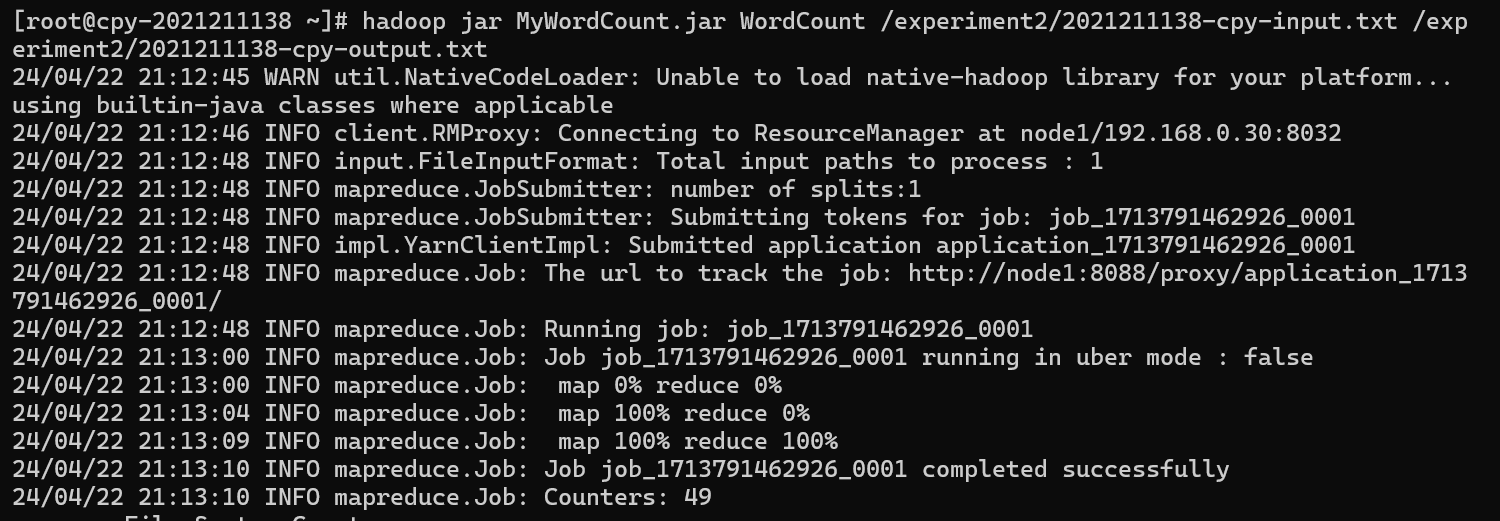


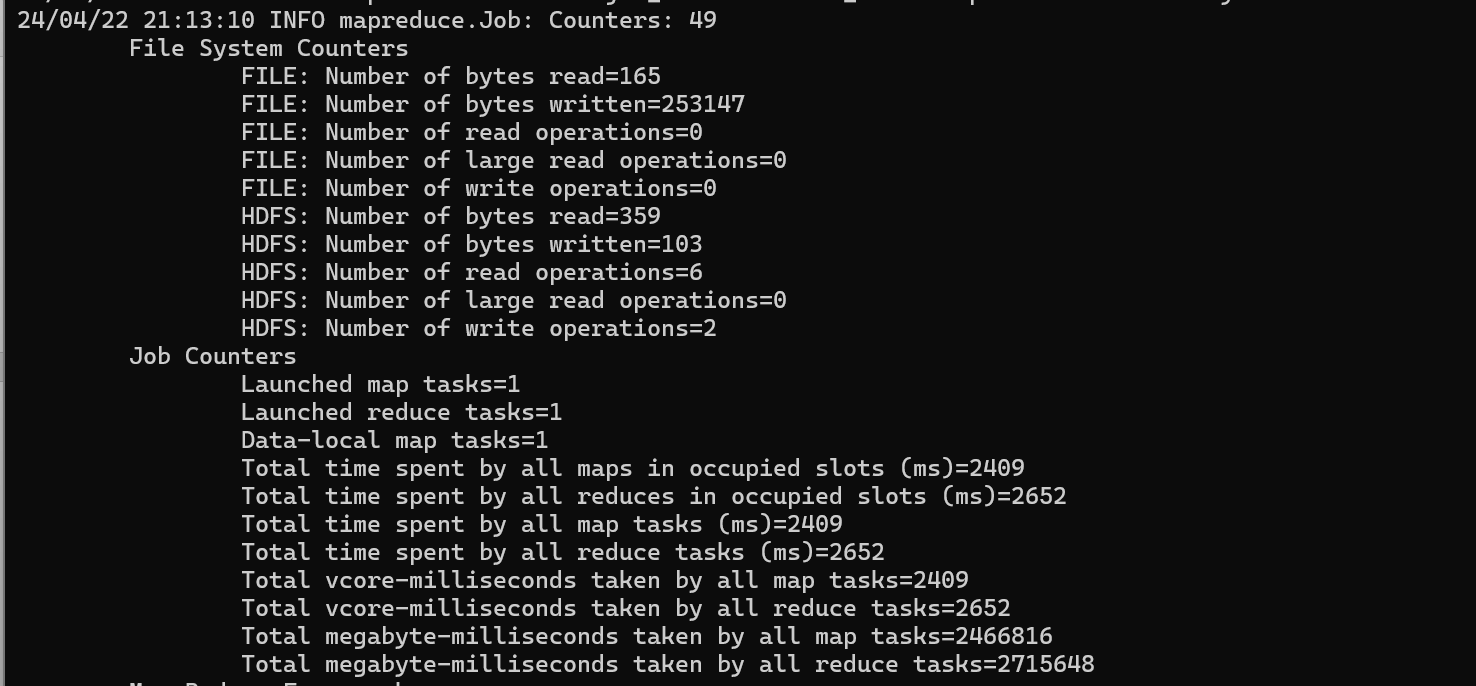
它其实是java连接错误，在连接到127.0.0.1的时候出错了。为什么它会连到42819端口呢？我们的hadoop服务的端口号并不是这个。我重新回顾了一下这个实验和上一个实验，定位了问题所在：hosts文件有问题。

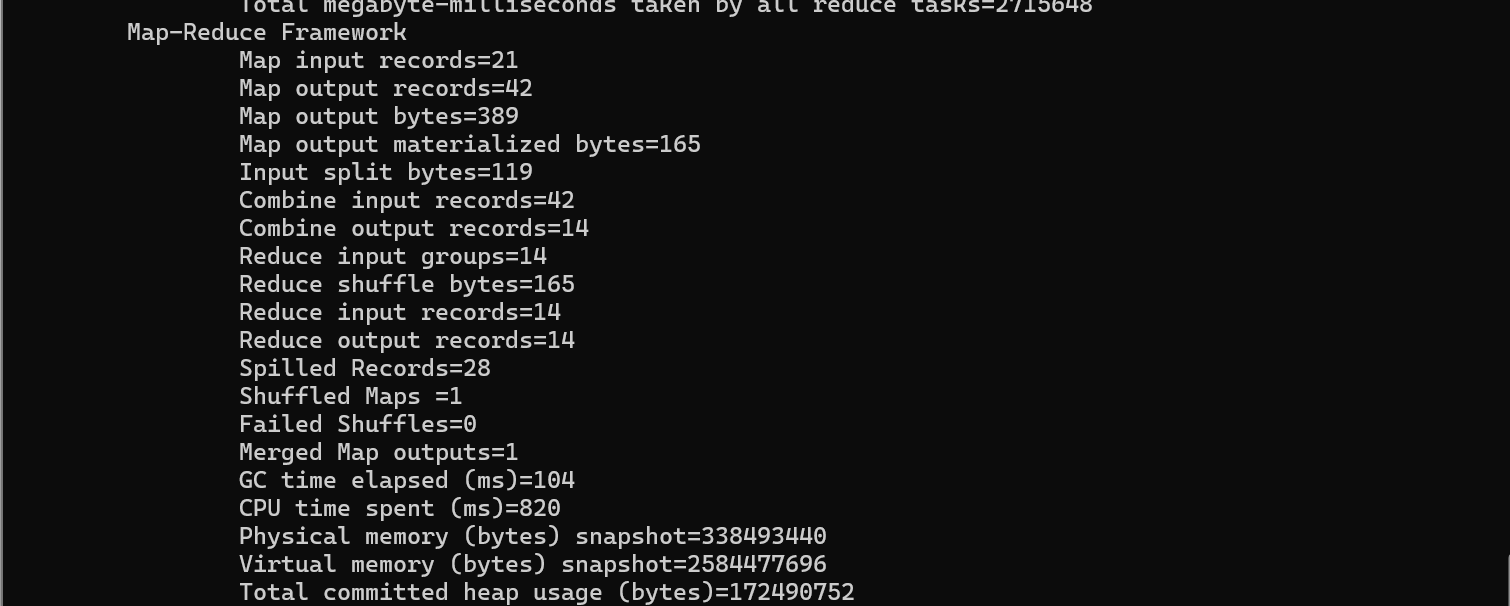
打开/etc/hosts文件



因为服务器每次启动都会自己添加一个127.0.0.1的配置，所以它这里自动多了一行127.0.0.1 的配置信息。这也是为什么我始终运行不成功的原因了。把这一行删掉，或者注释掉之后，重新执行jar指令。执行过程及结果如下图所示：



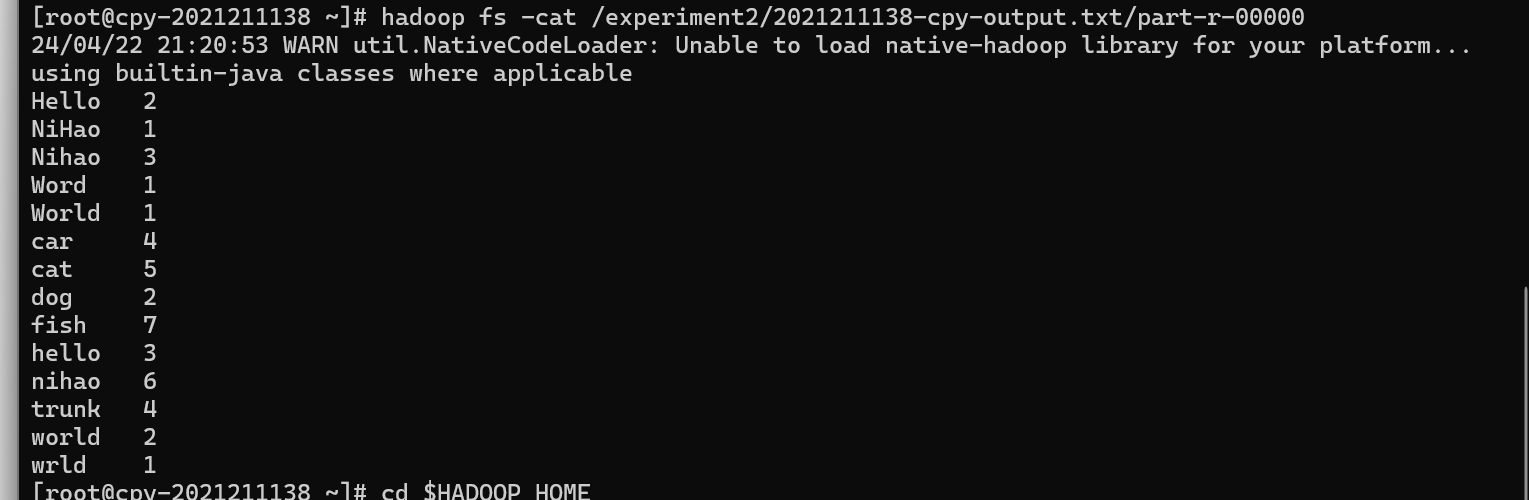






最后执行命令：

hadoop fs -cat /experiment2/2021211138-cpy-output.txt/part-r-00000，结果如下：



实验就顺利完成了。

## 2 实验结果与分析

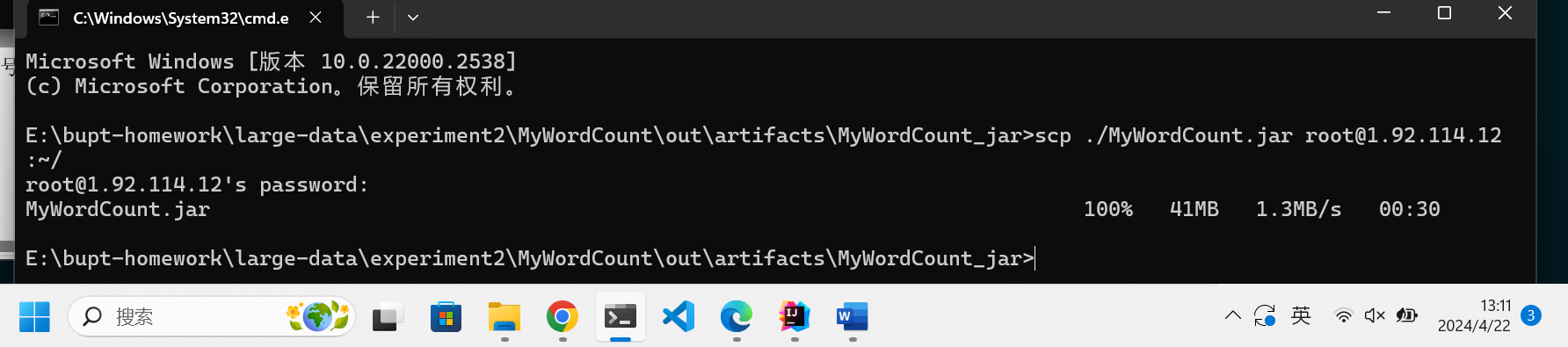
其实实验结果与分析已经在[1实验流程](#_1_实验流程)里都有了。这里我再次放一下。

### 2.1 jar包生成图



### 2.2 执行命令截图

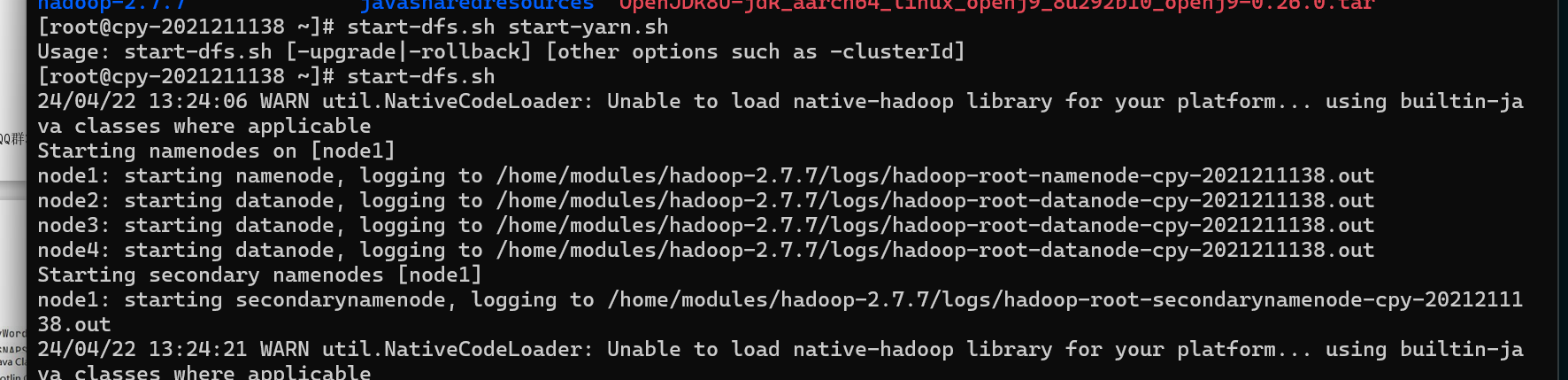
1. 发送jar包



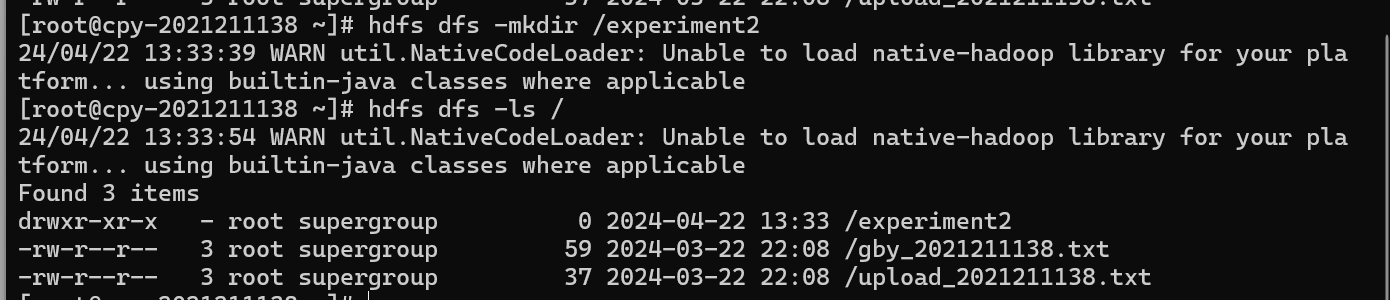
2. 发送input文件



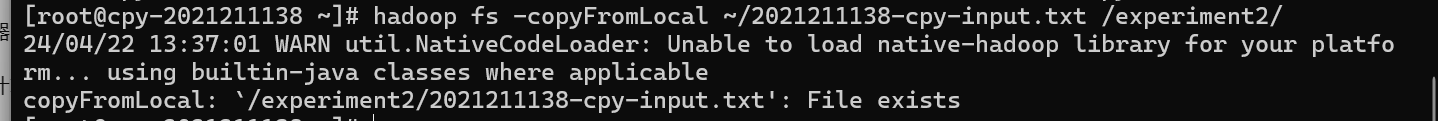
3. 启动hadoop集群



4. 查看hdfs根目录，新建目录experiment2

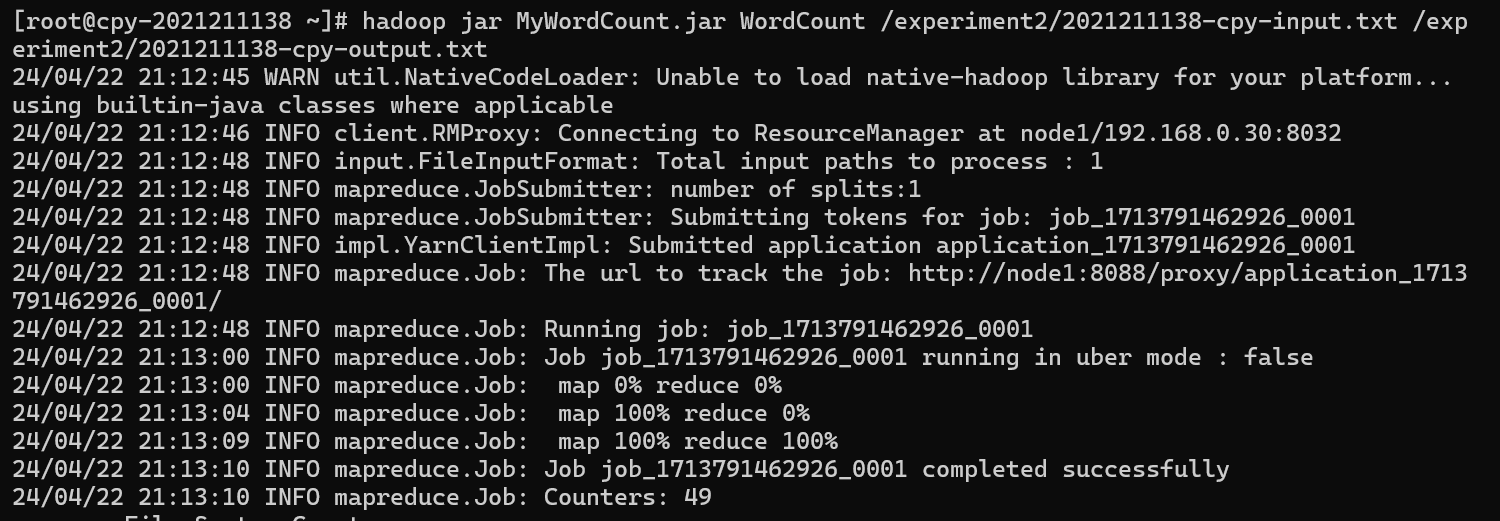


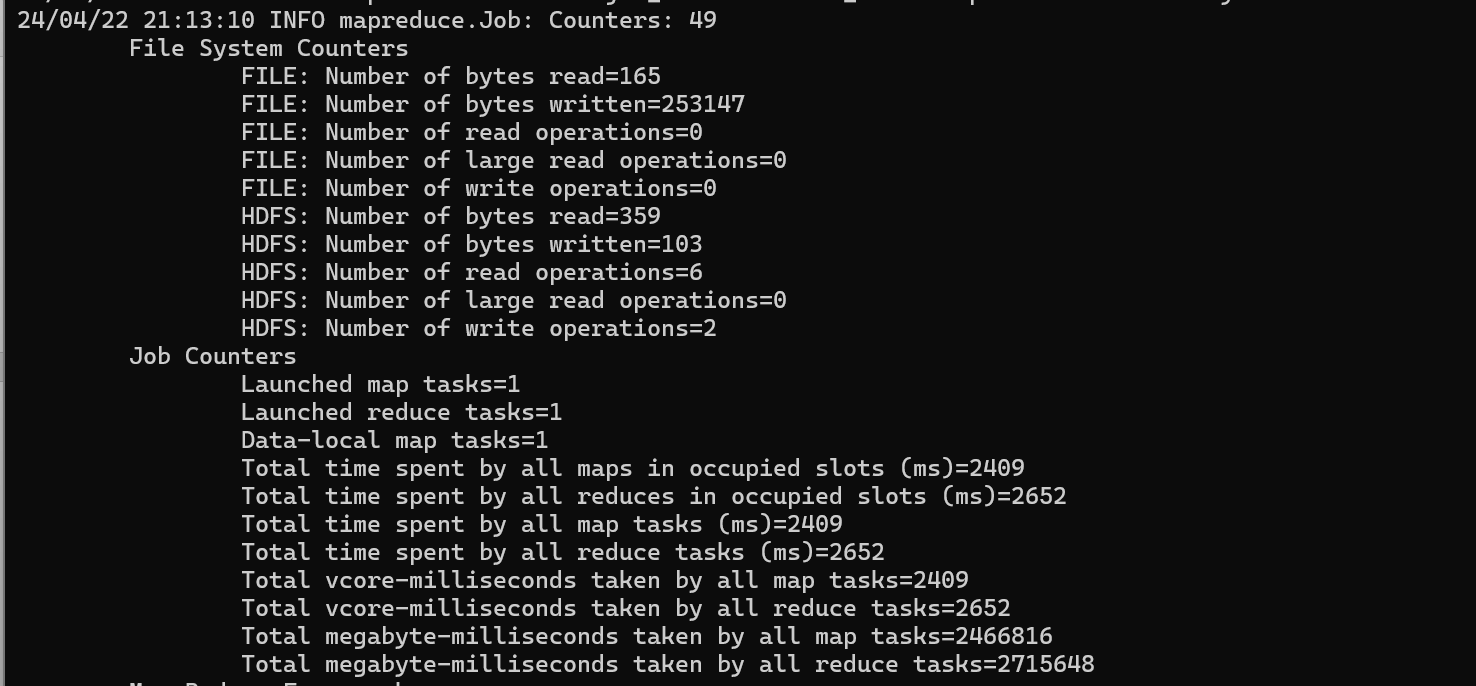
5. 将输入文件放入到新建的目录下

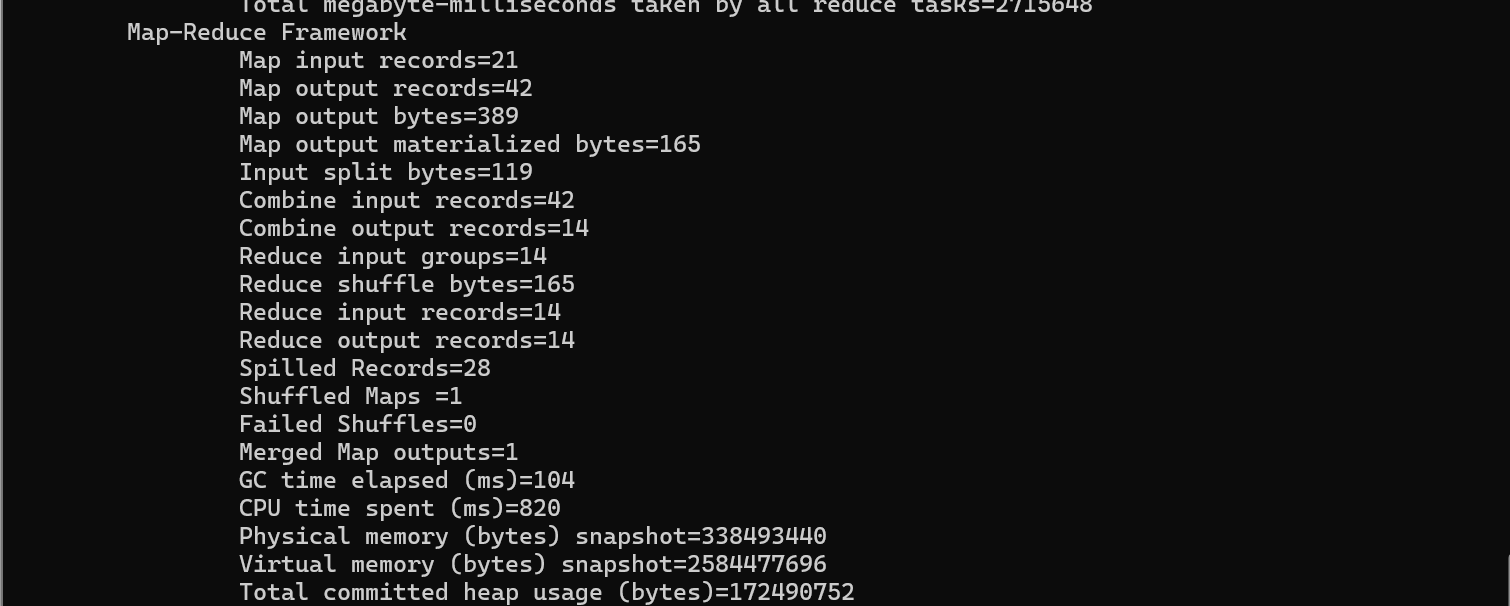


### 2.3 步骤8执行命令结果图

第8步这里实验指导手册里命令格式漏了一个好像，应该如下所示：  
 hadoop jar <jar包> <main函数的入口类> <input 文件或路径> <output路径>。

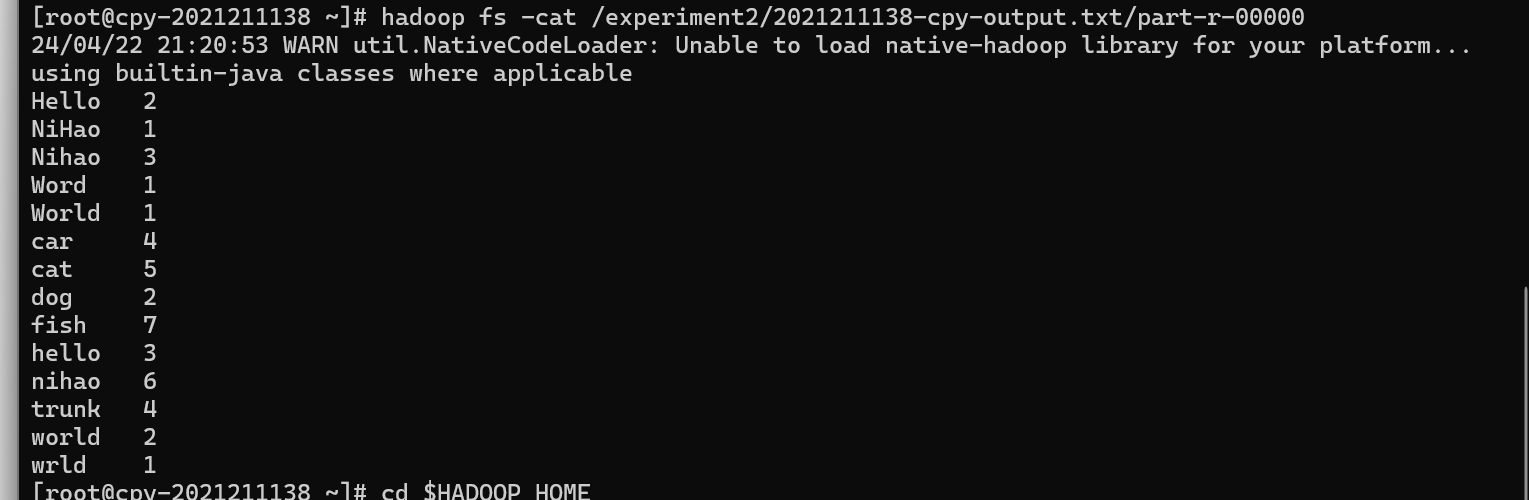








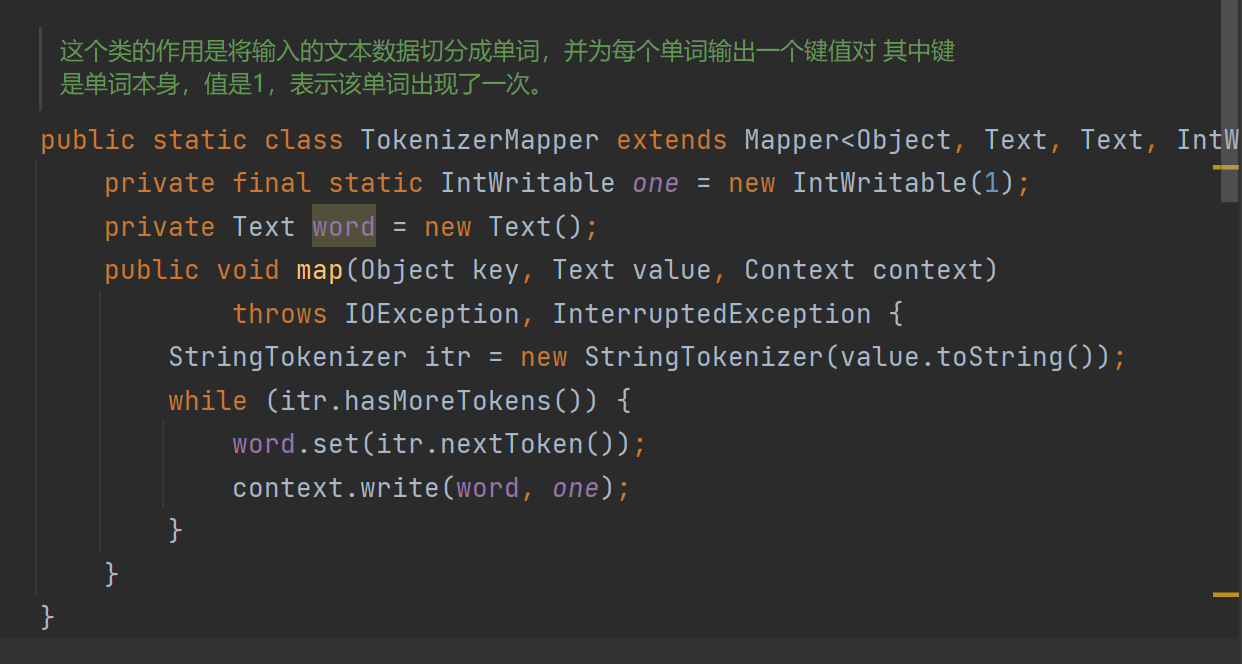
### 2.4 步骤9 执行命令结果图



### 2.5 提交jar包（另附件）

### 2.6 解释WordCount程序代码

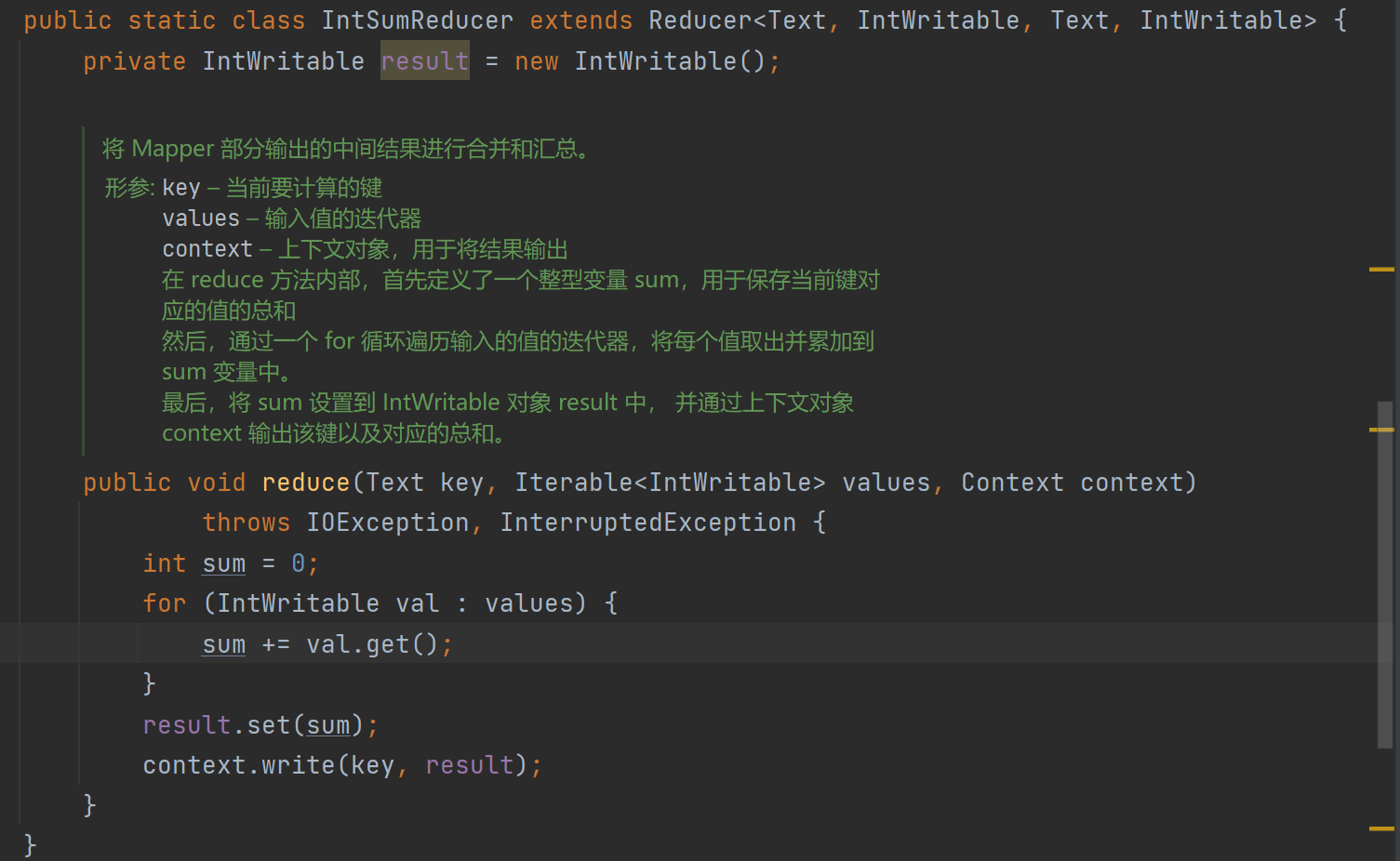
类TokenizerMapper:这个类的作用是将输入的文本数据切分成单词，并为每个单词输出一个键值对 其中键是单词本身，值是1，表示该单词出现了一次。



类IntSumReducer: 将Mapper输出的中间结果进行合并和汇总

Reduce函数：

|  |
| --- |
| 形参:  key – 当前要计算的键  values – 输入值的迭代器  context – 上下文对象，用于将结果输出  函数概述：  在 reduce 方法内部，首先定义了一个整型变量 sum，用于保存当前键对应的值的总和  然后，通过一个 for 循环遍历输入的值的迭代器，将每个值取出并累加到 sum 变量中。  最后，将 sum 设置到 IntWritable 对象 result 中， 并通过上下文对象 context 输出该键以及对应的总和。 |



主函数的运行逻辑为：

1. 判断输入的参数格式是否正确

2. 创建job对象，并为job对象初始化

3. 设置输出的Key和Value

4. 设置输入输出的路径

