# 实验六 实现MapReduce程序完成行统计

**·实验要求：**

实现一个统计航所属的mapReduce程序。

统计文件中的行数，写出程序代码，并将程序在Hadoop集群运行，显示输出结果。

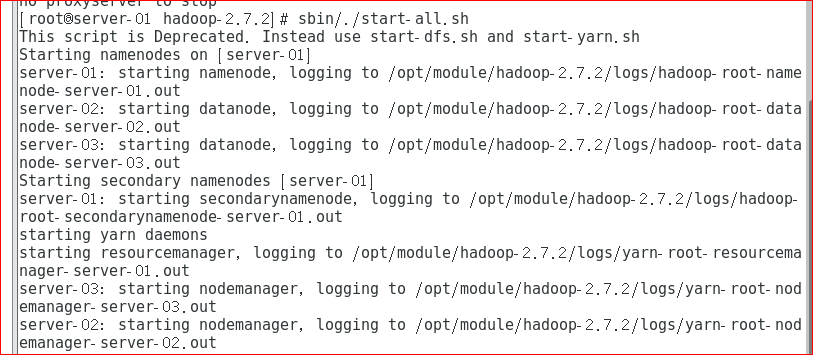
**·实验环境：**

Hadoop集群正常启动，eclipse工具。

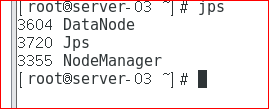
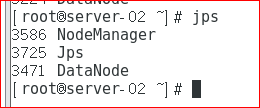
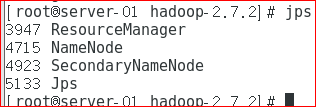
**·实验步骤：**

1. 启动hadoop服务：
   1. 启动hadoop集群：

|  |
| --- |
| cd /opt/module/hadoop-2.7.2  sbin/./start-all.sh |

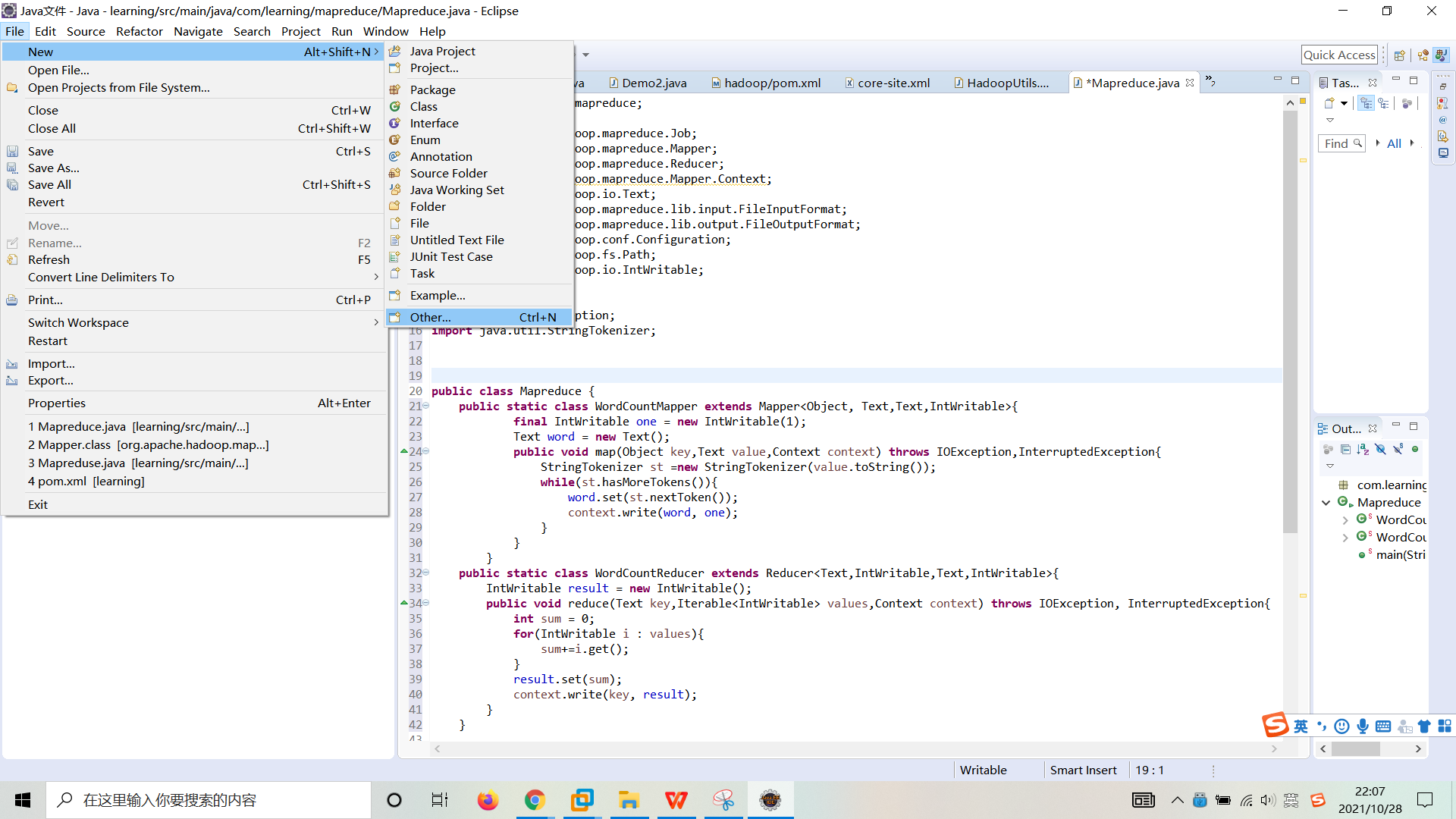
6

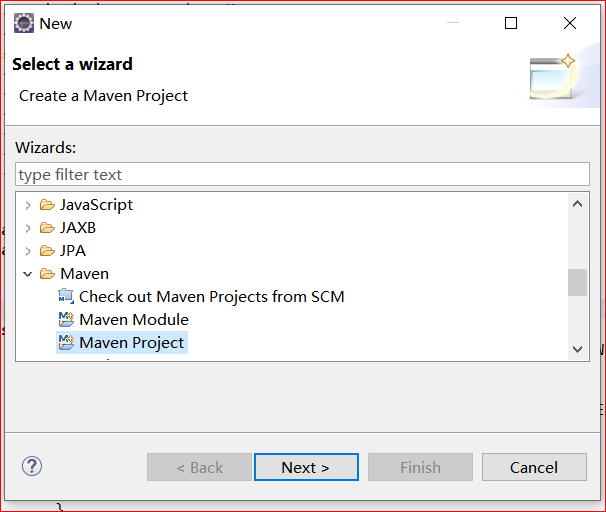
使用jps命令查看已启动的服务：

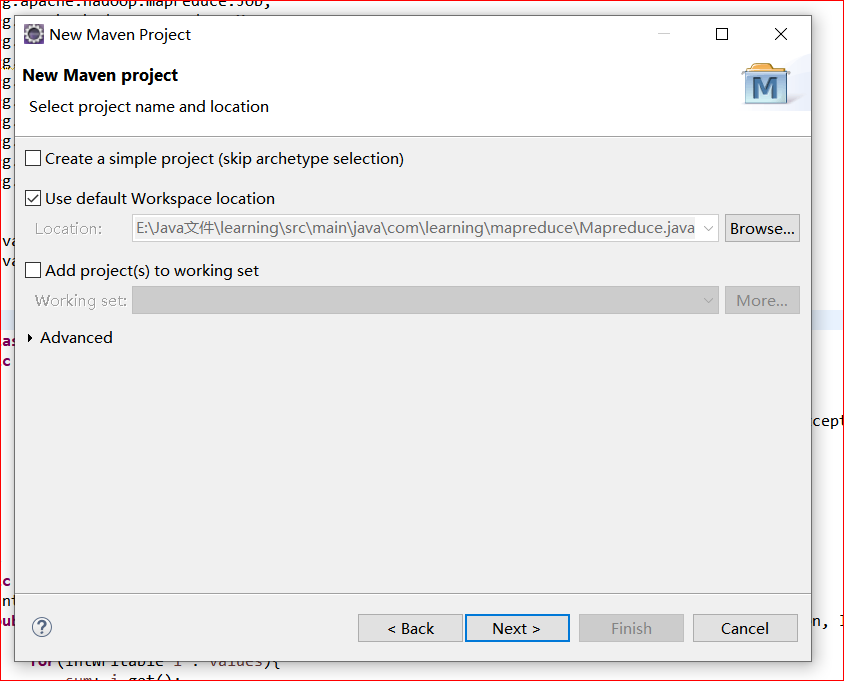


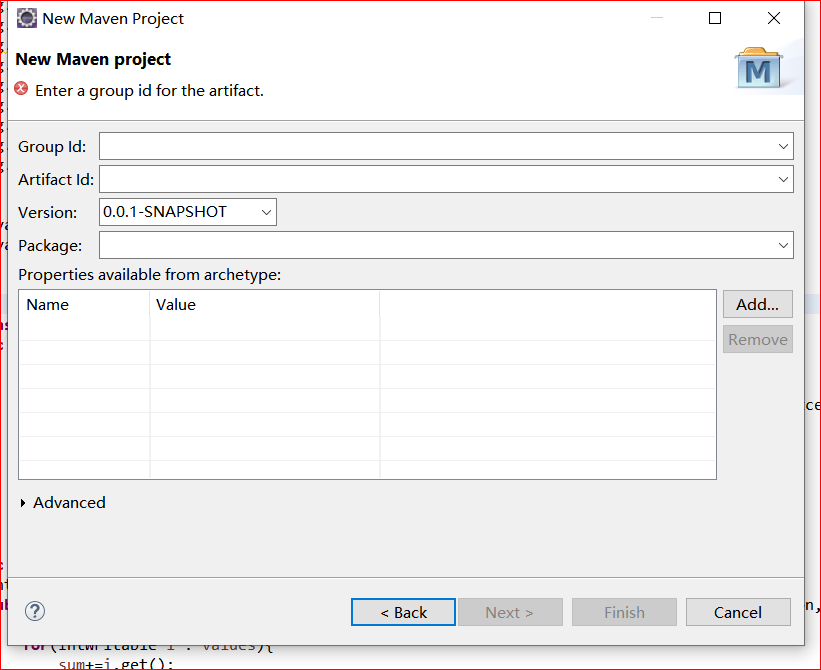
1. 开发代码：

2.1、打开eclipse工具并新建Maven项目DataCount，并配置buildpath（设置jdk版本为1.8）







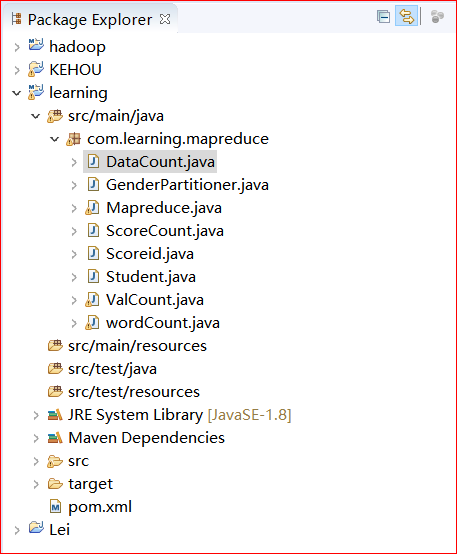


写入信息

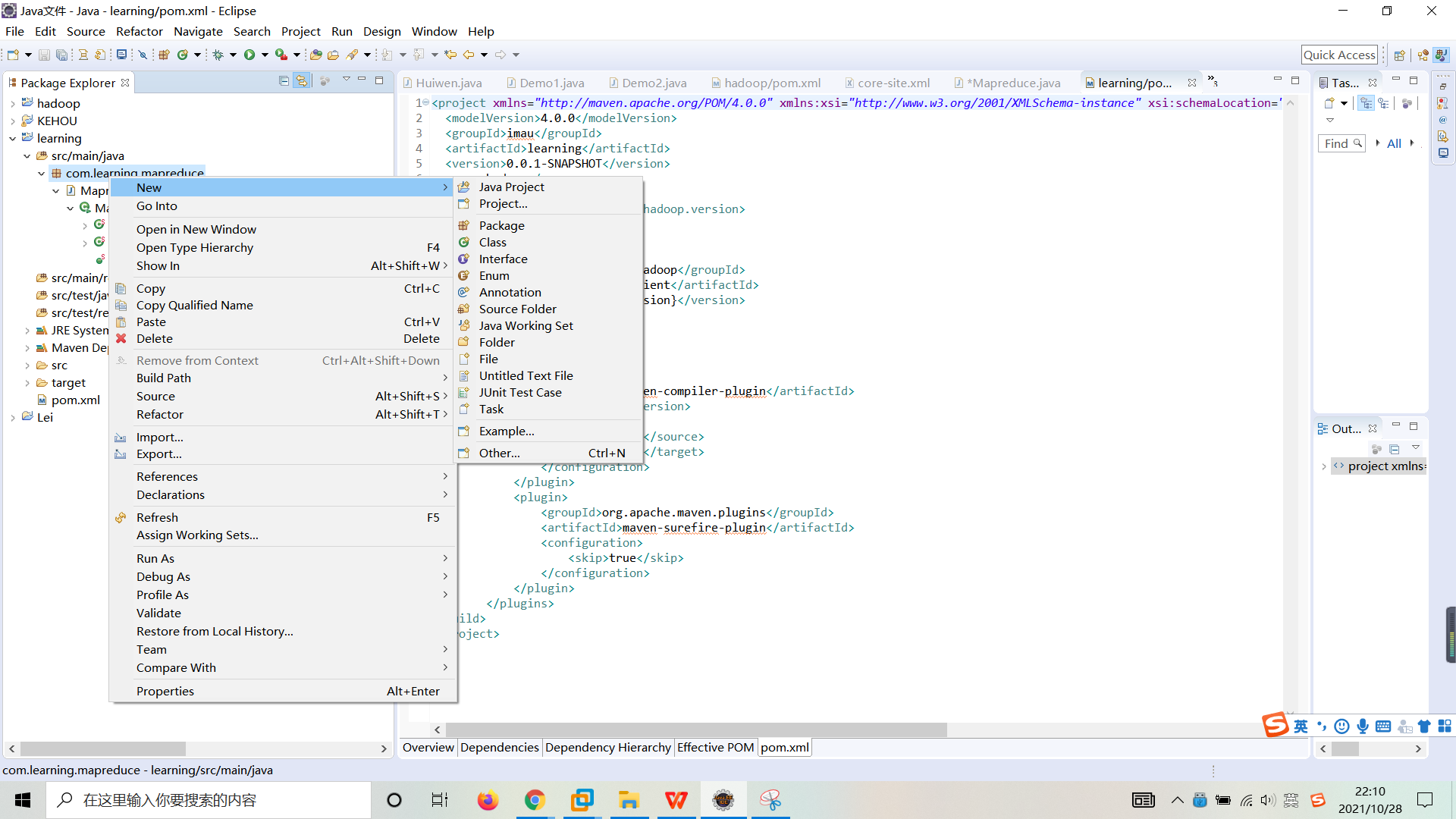
添加pom.xml文件内容，并下载相关maven包：

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>imau</groupId>  <artifactId>learning</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>hadoop</name>  <properties>  <hadoop.version>2.7.2</hadoop.version>  </properties>  <dependencies>  <dependency>  <groupId>org.apache.hadoop</groupId>  <artifactId>hadoop-client</artifactId>  <version>${hadoop.version}</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.0</version>  <configuration>  <source>1.8</source>  <target>1.8</target>  </configuration>  </plugin>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-surefire-plugin</artifactId>  <configuration>  <skip>true</skip>  </configuration>  </plugin>  </plugins>  </build>  </project> |

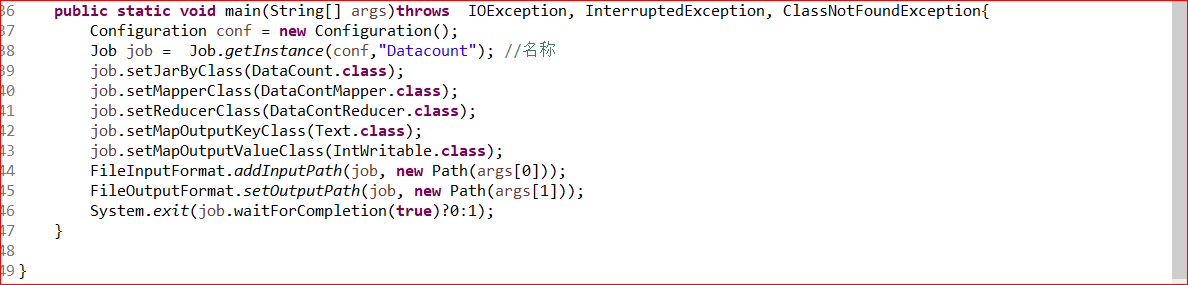
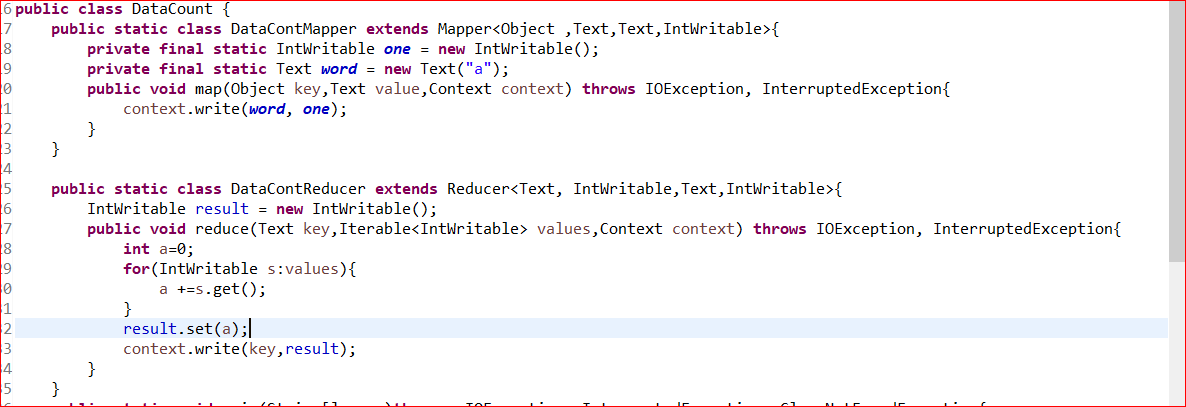
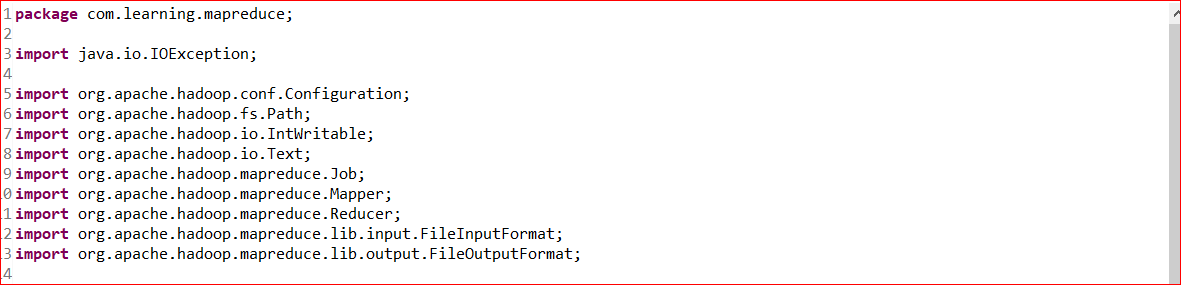




在src/main/java/下新建 com.learning.mapreduce 包，并在com.learning.mapreduce下新建 DataCount class文件：



DataCount内容：



将map传递下来的内容进行读取拼接，使用for循环，将每一行读取次数相加，由此计算文件内行数。

写相关map代码，定义one为整型，Word为text型，里面定义一个常量，并将Word和one写入下面的reduce中

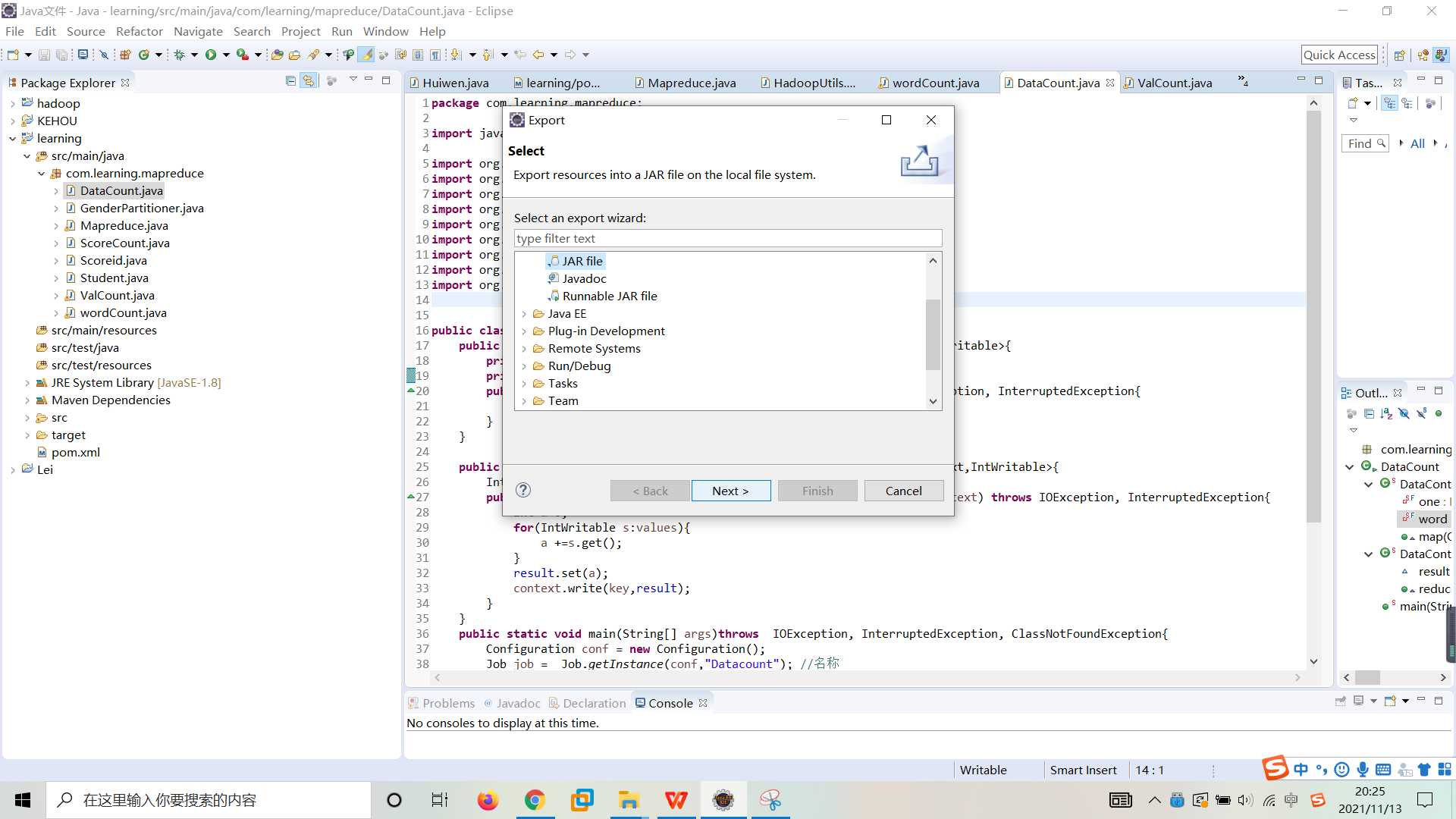
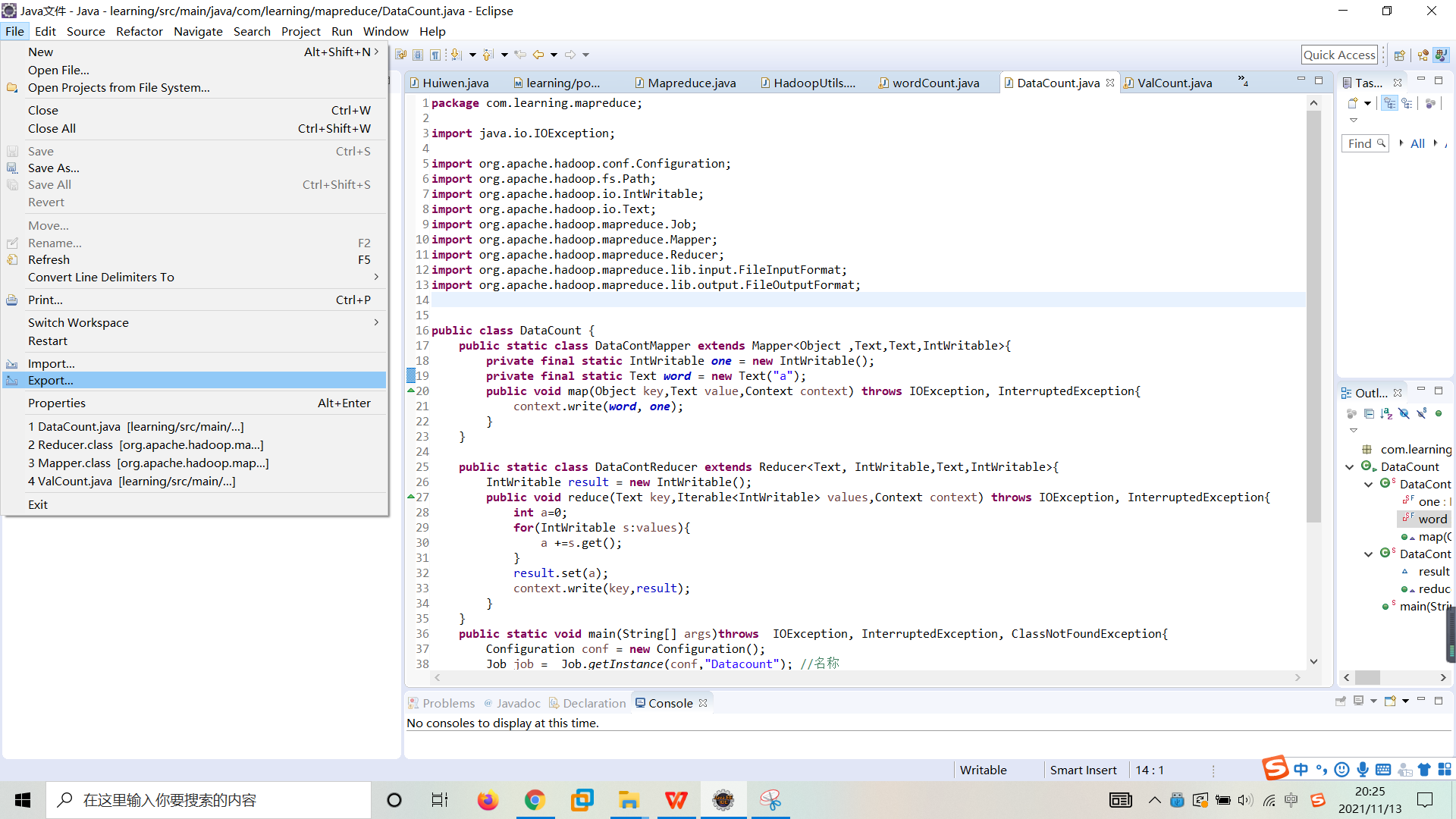
引入相关包

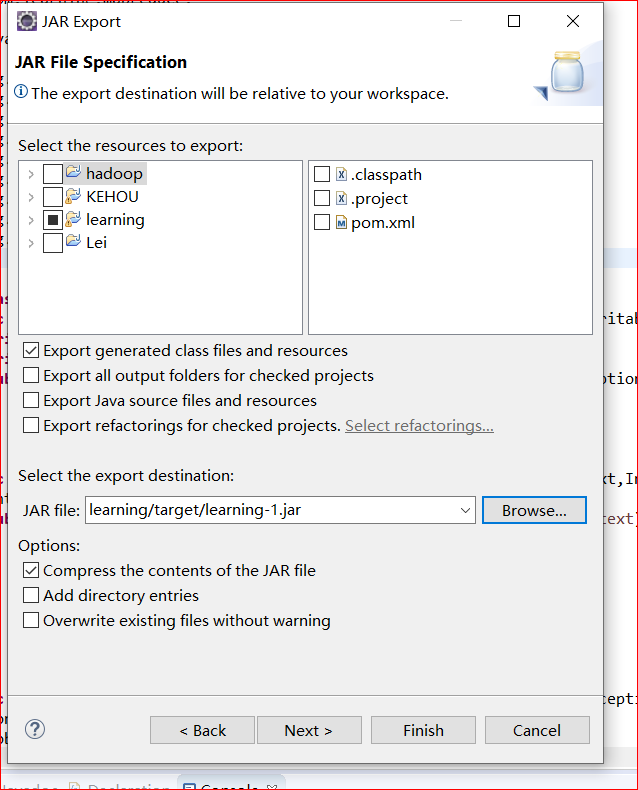
因为mapper就是对文件的内容一行一行进行读取的，所以定义的Word是一个常量，只要对mapper读取的次数计数就可以了，所以reduce中的key值为常数，value值为行数。

|  |
| --- |
| package com.learning.mapreduce;  import java.io.IOException;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Job;  import org.apache.hadoop.mapreduce.Mapper;  import org.apache.hadoop.mapreduce.Reducer;  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  public class DataCount {  public static class DataContMapper extends Mapper<Object ,Text,Text,IntWritable>{  private final static IntWritable one = new IntWritable();  private final static Text word = new Text("a");  public void map(Object key,Text value,Context context) throws IOException, InterruptedException{  context.write(word, one);  }  }    public static class DataContReducer extends Reducer<Text, IntWritable,Text,IntWritable>{  IntWritable result = new IntWritable();  public void reduce(Text key,Iterable<IntWritable> values,Context context) throws IOException, InterruptedException{  int a=0;  for(IntWritable s:values){  a +=s.get();  }  result.set(a);  context.write(key,result);  }  }  public static void main(String[] args)throws IOException, InterruptedException, ClassNotFoundException{  Configuration conf = new Configuration();  Job job = Job.getInstance(conf,"Datacount"); //名称  job.setJarByClass(DataCount.class);  job.setMapperClass(DataContMapper.class);  job.setReducerClass(DataContReducer.class);  job.setMapOutputKeyClass(Text.class);  job.setMapOutputValueClass(IntWritable.class);  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  System.exit(job.waitForCompletion(true)?0:1);  }  } |

1. 导出jar包：

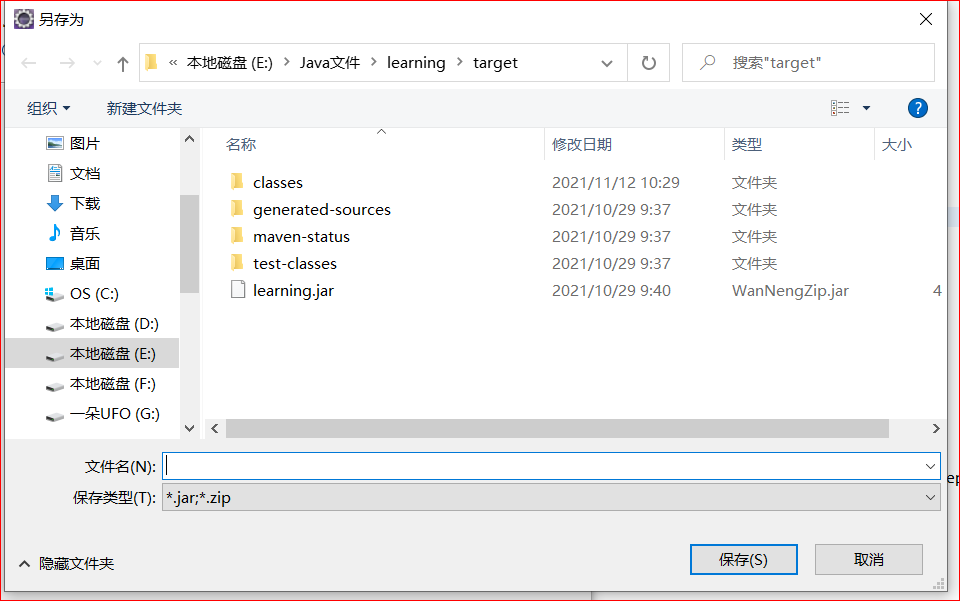
在MapReduce项目上单击右键，点击“Export”：





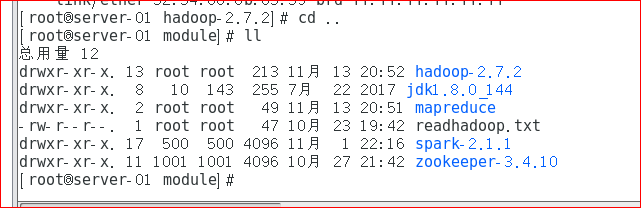
q

找到保存位置，并命名为learning-02.jar:

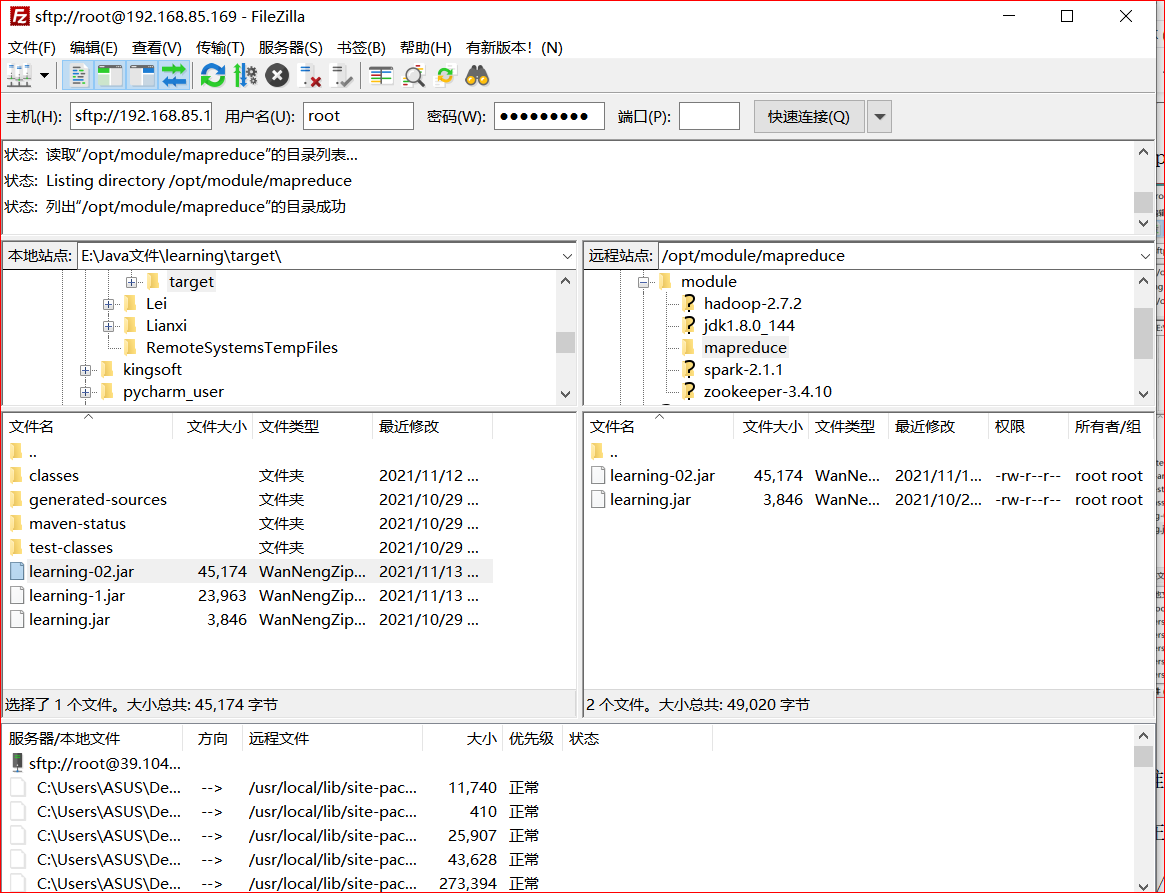


1. 将jar包文件上传到master服务器上

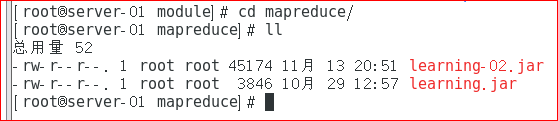
查看/opt/module/下mapreduce文件是否存在：



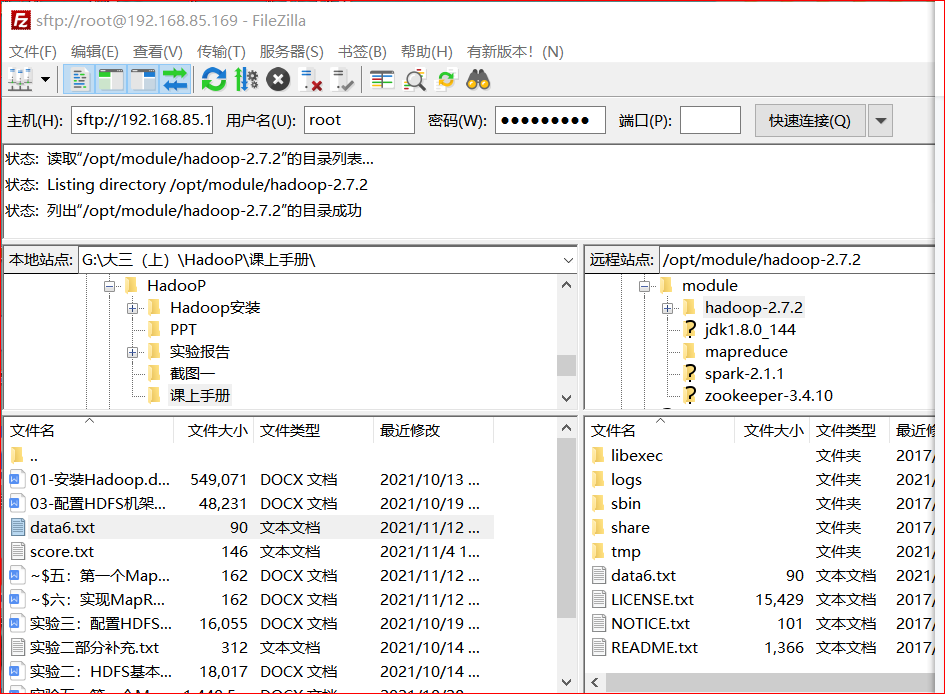
将jar传入server-01已有的/opt/module/mapreduce目录下：



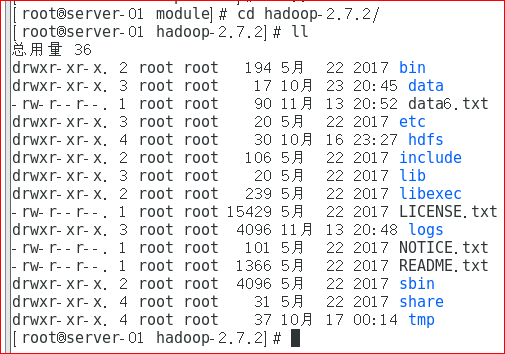
进入MapReduce文件查看是否传入成功：



将要测试的文件传入Hadoop-2.7.2下：



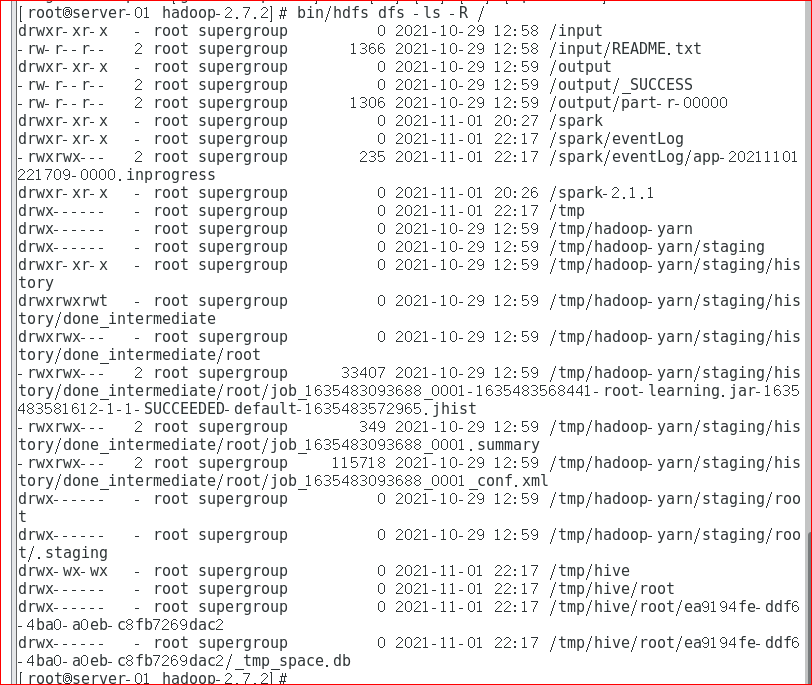
查看是否上传成功：



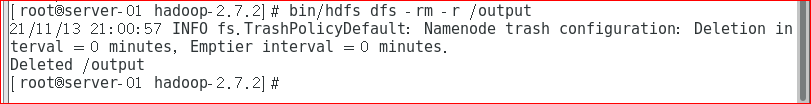
1. 准备测试数据：

先查看hdfs集群的文件，再将待测试文件上传到hdfs集群的/input下：

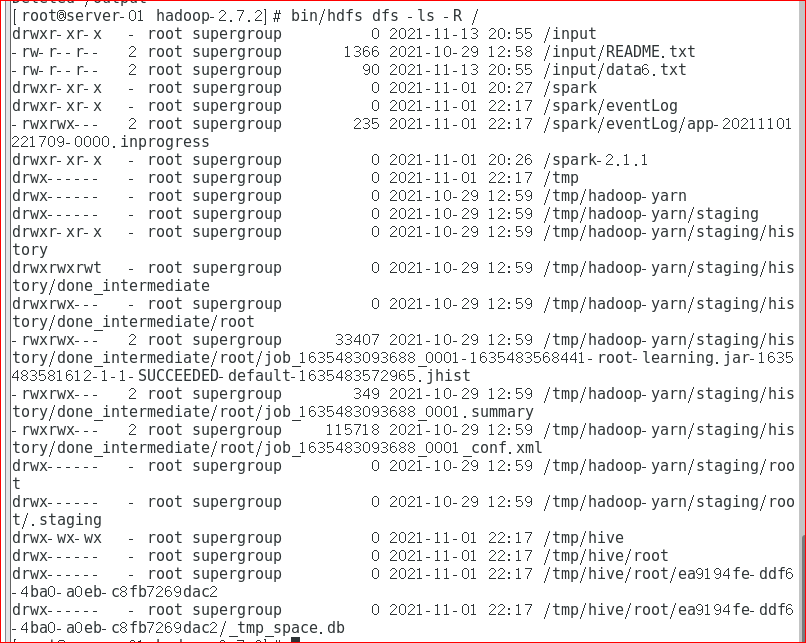
|  |
| --- |
| bin/hdfs dfs -ls -R /  bin/hdfs dfs -put /opt/module/hadoop-2.7.2/data6.txt /input/ |

17

将output文件目录删除，避免报错：



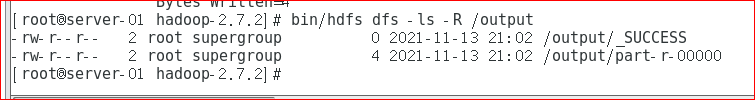
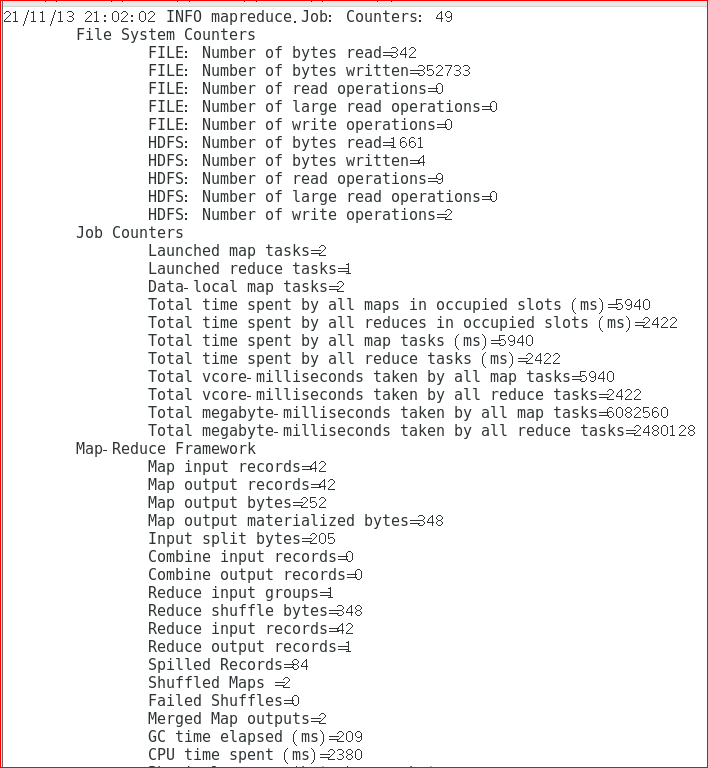
删除output文件后再测查看hdfs集群下所有文件，发现已无output的文件目录：



1. 运行DataCount代码：

使用命令：bin/hadoop jar运行DataCount程序；使用命令：bin/hdfs dfs -ls -R /查看运行成功之后生成了哪些文件和目录；使用命令：bin/hdfs dfs -cat来查看运行结果：

20

23

因为上面的代码中最后输出的Word是一个常量“a”，且并无任何计数等作用，所有这里表示的“a”是一个常量，并无计数，行数为49。

24

**·实验心得：**

在该实验中，学会了mapper中范型如何书写，知道了reduce代码中输出的值的属性定义，如何找到正确的key和value，如何生成相关文件，并进行查看。