

**大数据处理实验报告**

实验三：MapReduce的基本操作

姓 名：刘日星

学 号：X2020I1007

院 系：计算机科学与技术

专 业：计科与金融

年 级：CS1804-交换生

指导教师：石宣化

2021年 12 月 15 日

一：实验目的

1、了解MapReduce的用途

2、掌握MapReduce的基本命令

二：实验要求

1. 第四节中的实验内容要附上完整的实验过程截图以及必要的文字说明，每个人的IP地址等不同，不能直接套用样例的截图。
2. 请同学们在完成报告后，将报告的pdf版本命名为：大数据实验三+姓名+学号.pdf，并在这周五前,发到邮箱:

[aaaaltaaaa@126.com(石老师班)](mailto:aaaaltaaaa@126.com(石老师班)，)

798792873@qq.com(郑老师班)

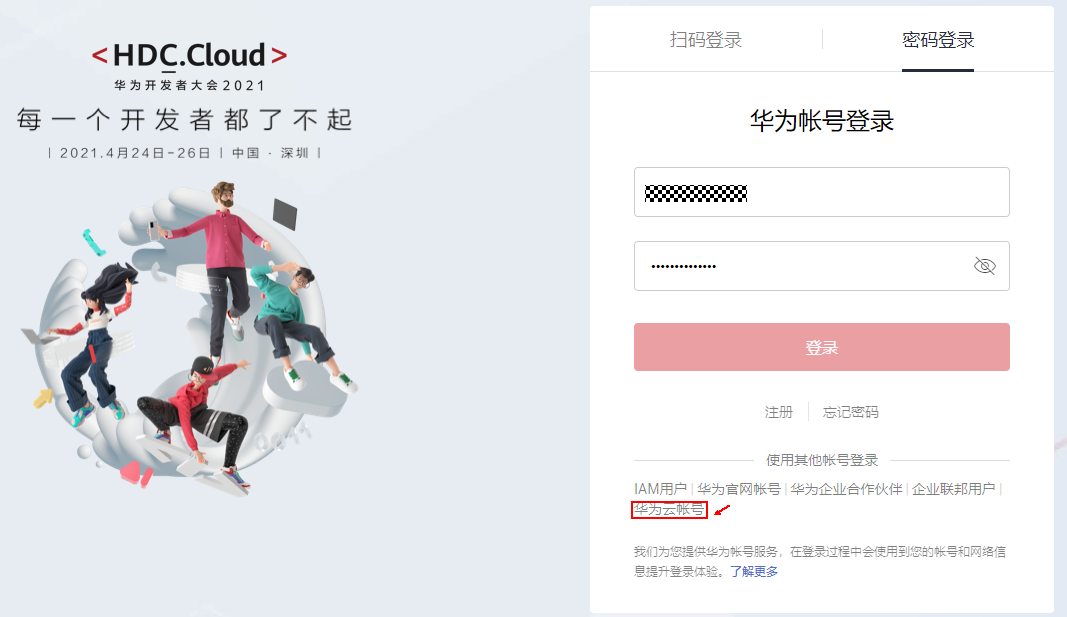
三：实验环境配置

登录华为云网站

<https://www.huaweicloud.com>



点击右上角登录，输入账号和密码



注意：华为云已统一登录入口，若仍不能登录则点击下方华为云账号进行登录。

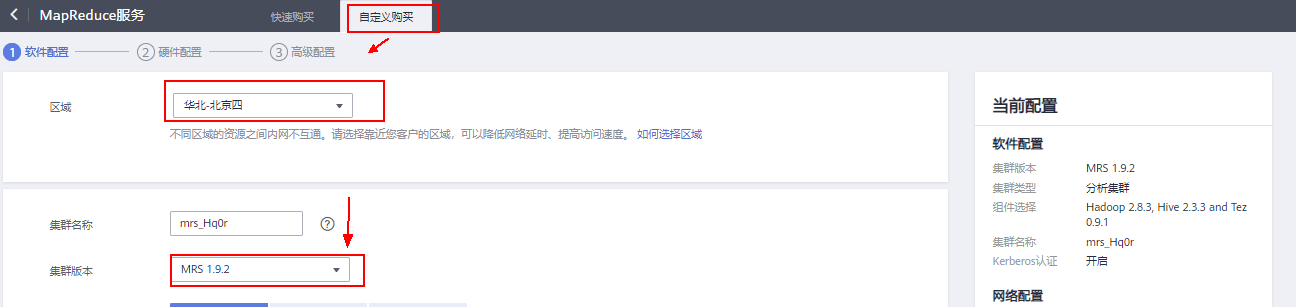
点击“EI企业智能”选择“MapReduce服务”



点击“立即购买”



选择“自定义购买”(这里还要选上Spark)





点击下一步，进入硬件配置

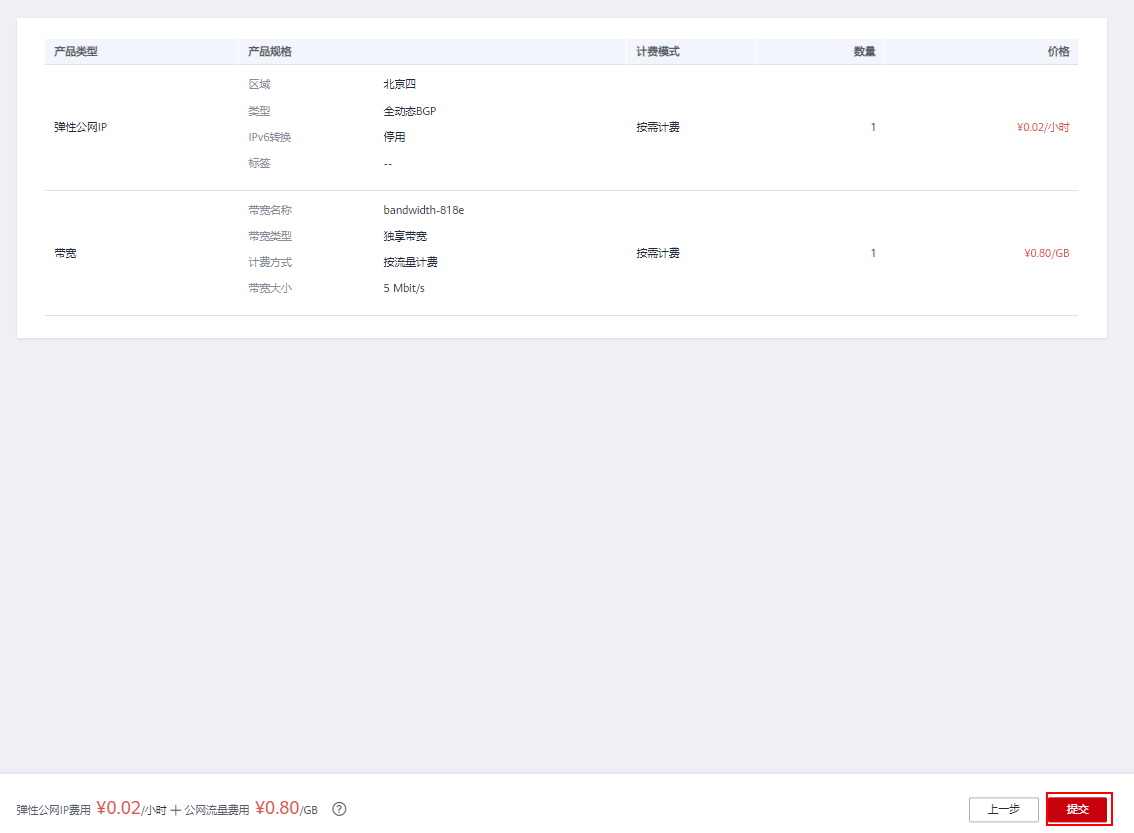
选择“按需计费”，“可用区2”，点击“弹性公网IP”，如下图：



点击“购买弹性公网IP”，选择“按需计费”，“按流量计费”，“5M”,点击“立即购买”，如下图：



点击“提交”，如下图：



购买成功，如下图：



返回MapReduce服务自定义购买界面绑定EIP



选择“鲲鹏计算”，关闭高可用，调整core节点数为1,如下图：



点击“下一步”

高级配置项参考如下：



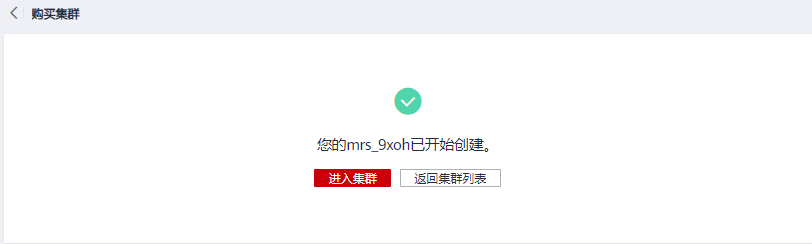


点击“确认授权”

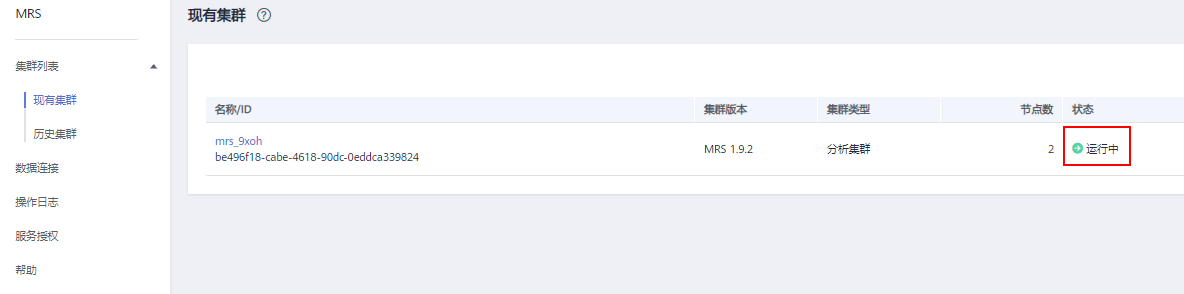


点击“立即购买”

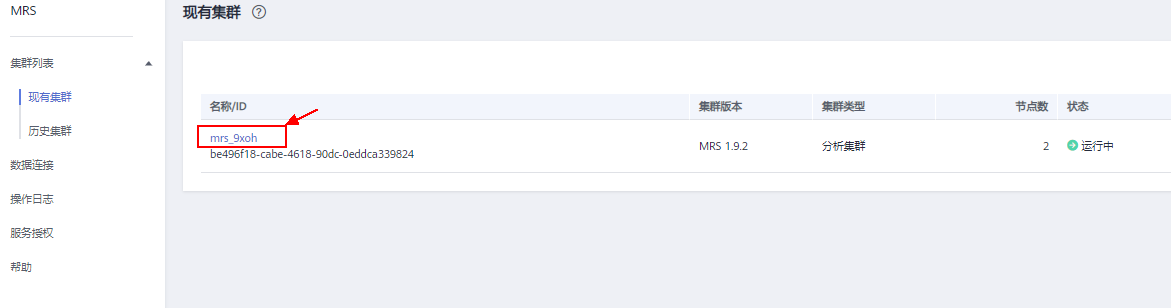
点击“返回集群列表”，如下图：



创建过程需要等待几分钟，待状态变为“运行中”集群创建完成

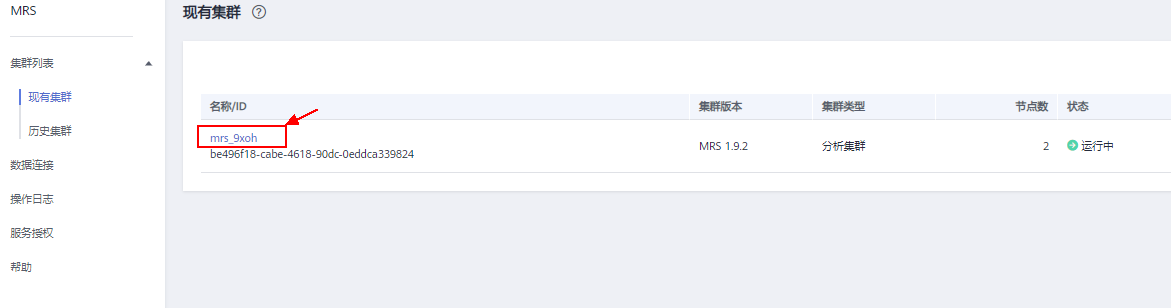


点击集群名称



配置安全组

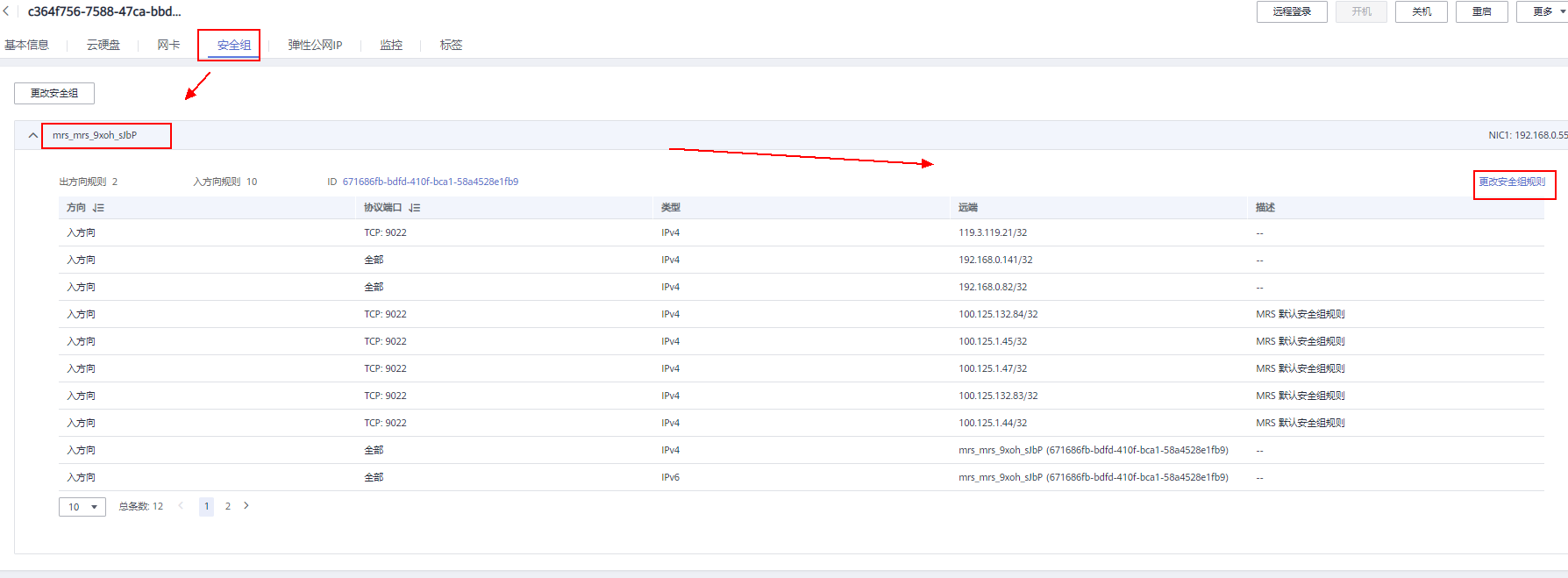
点击集群名称



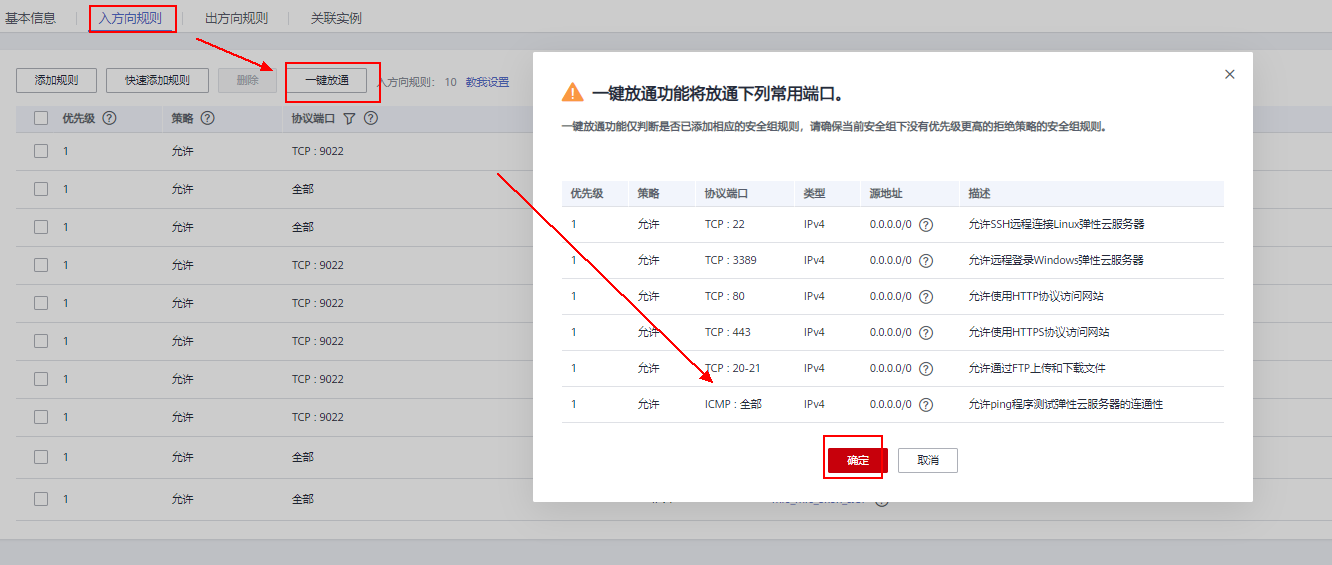
选择“节点管理”，点击含有“master1”的节点



在弹出页面中选择“安全组”，点击“更改安全组规则”，如下图所示：

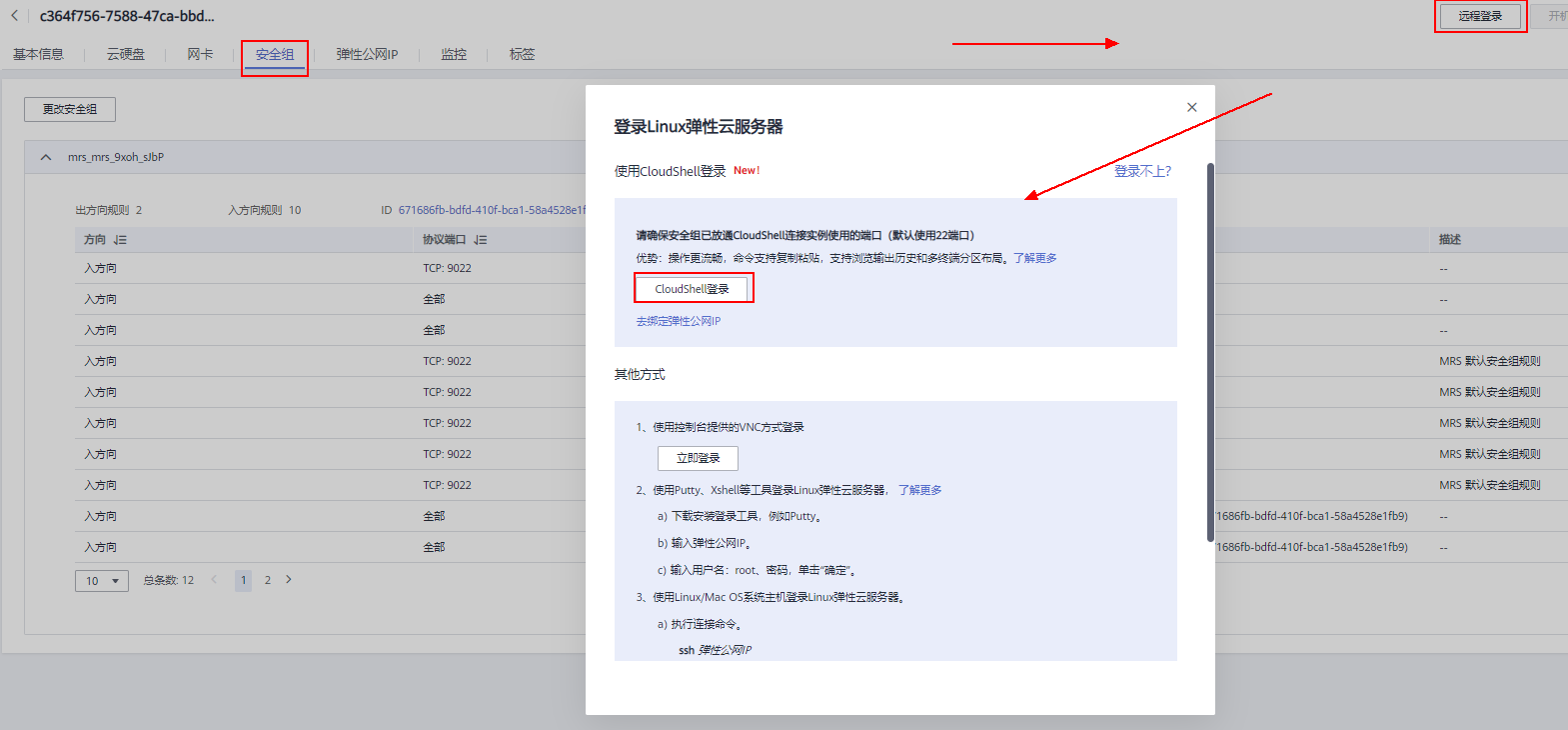


选择“入方向规则”，点击“一键放通”，确认即可。



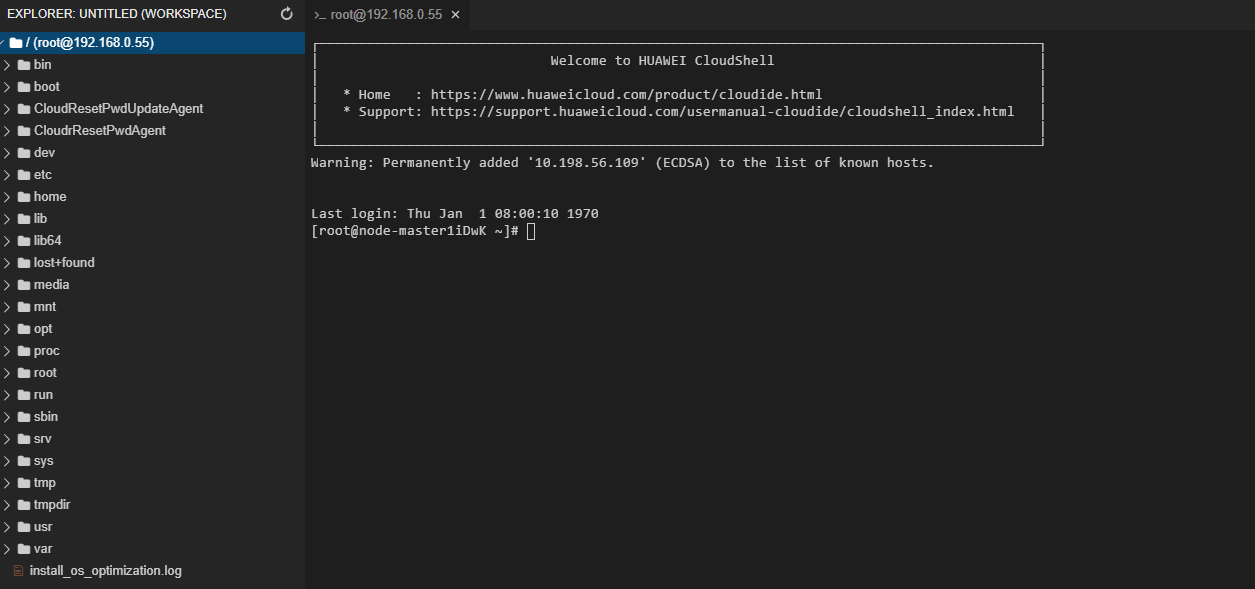
远程登录master节点

在安全组配置项，点击右上方“远程登录”，选择cloudshell登录。



输入密码，点击连接即可。





四：实验内容及步骤、实验的详细记录、实验结果分析

**请附上实验过程截图（截图需包含指令）以及必要的文字分析**

## MapReduce

## 4.1.1 进入hadoop(5’)

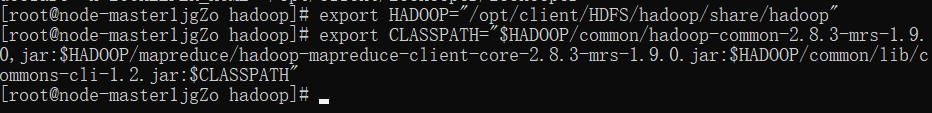
### cd /opt/client/HDFS/Hadoop



## 4.1.2 添加环境变量(10’)

export HADOOP="/opt/client/HDFS/hadoop/share/hadoop"

export CLASSPATH="$HADOOP/common/hadoop-common-2.8.3-mrs-1.9.0.jar:$HADOOP/mapreduce/hadoop-mapreduce-client-core-2.8.3-mrs-1.9.0.jar:$HADOOP/common/lib/commons-cli-1.2.jar:$CLASSPATH"



配置java环境

## 4.1.3 创建java程序WordCount.java,在里面输入以下代码(5’)

import java.io.IOException;

import java.util.StringTokenizer;

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 WordCount {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

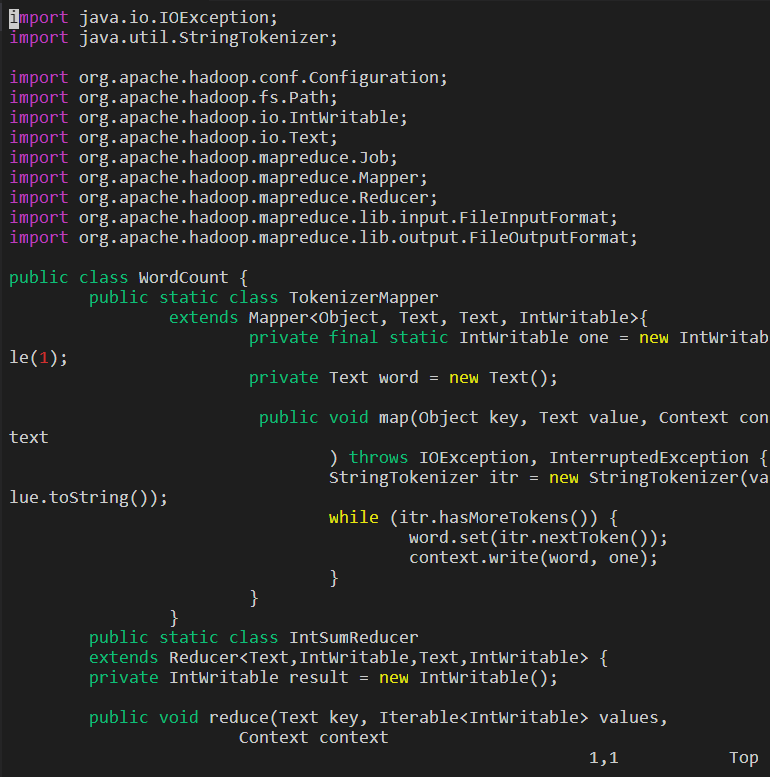
FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

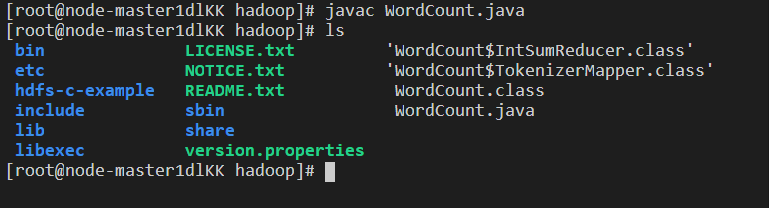
}

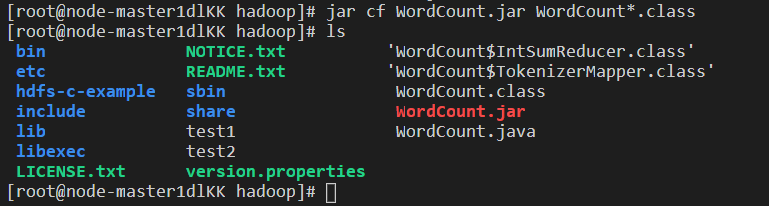




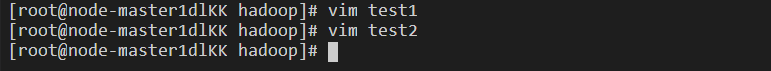
编写java文件

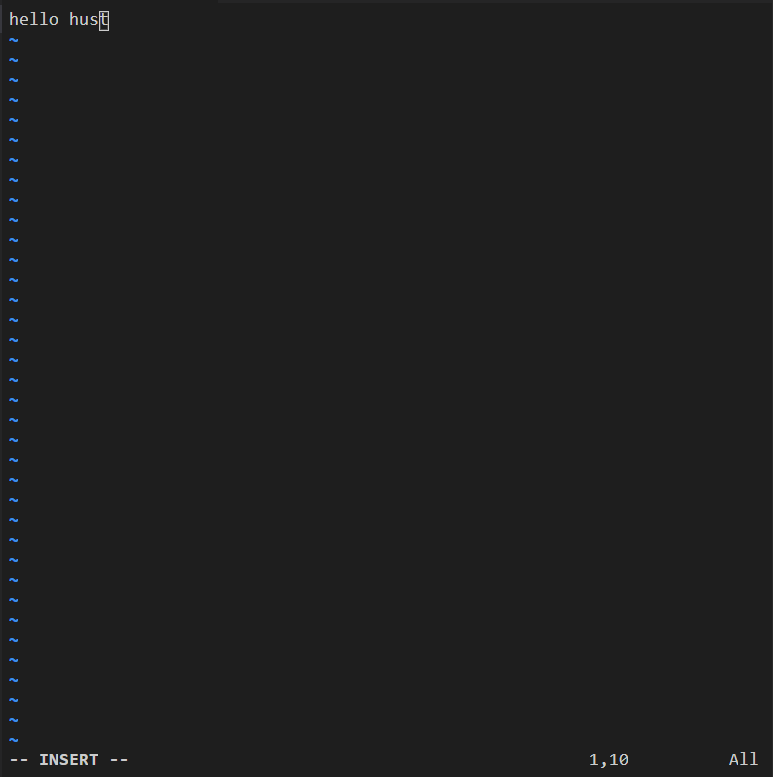
## 4.1.4编译WordCount.java(5’)



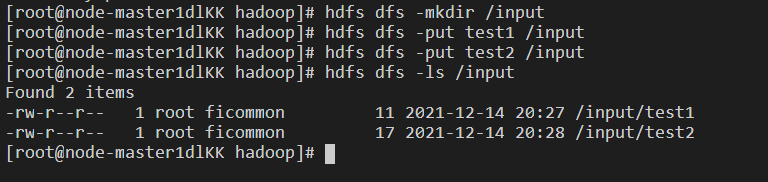


4.1.5创建文件test1，内容为hello hust，文件test2，内容为hello 学号，将他们放入hdfs的/input文件夹内。(方法见实验一)(5’)







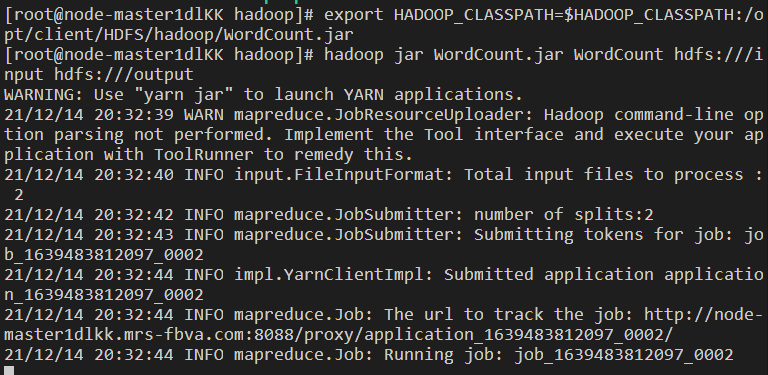


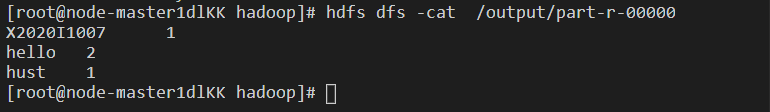
4.1.6运行WordCount.jar 将hdfs的/input作为输入，/output作为输出，并打印/output目录下的文件，显示出词频统计的结果(5’)

export HADOOP\_CLASSPATH=$HADOOP\_CLASSPATH:/opt/client/HDFS/hadoop/WordCount.jar

hadoop jar WordCount.jar WordCount hdfs:///input hdfs:///output

hdfs dfs -cat /output/part-r-00000

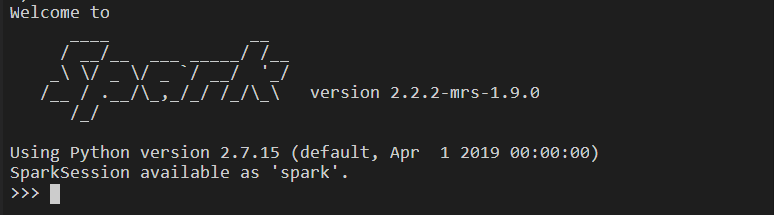




## Spark

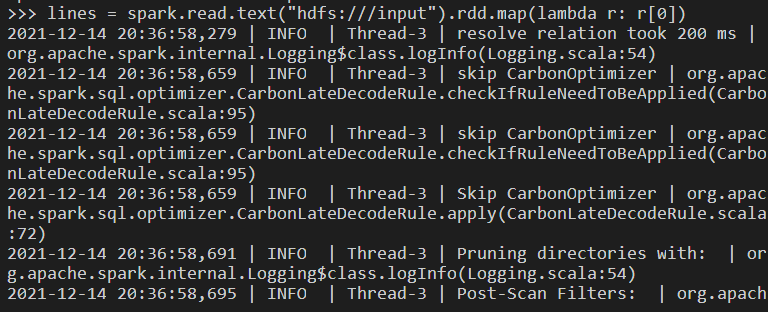
## 4.2.1 打开spark(5’)

### Pyspark



## 4.2.2 读取hdfs文件内容(5’)

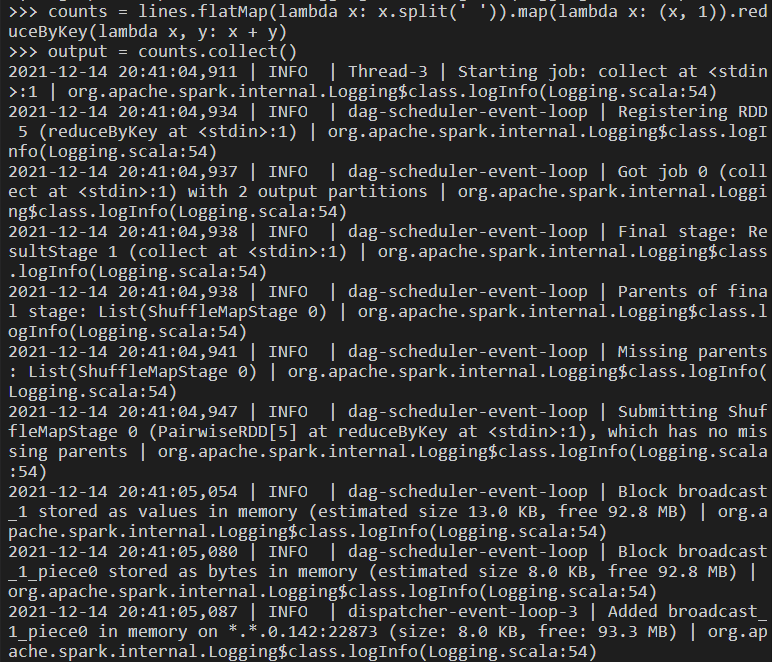
lines = spark.read.text("hdfs:///input").rdd.map(lambda r: r[0])



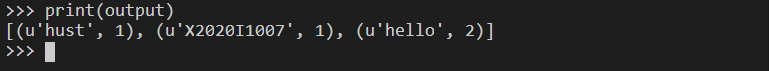
## 4.2.3 词频统计(5’)

counts = lines.flatMap(lambda x: x.split(' ')).map(lambda x: (x, 1)).reduceByKey(lambda x, y: x + y)

output = counts.collect()



## 4.2.4 输出结果(10’)



## 4.3 附加题

file1:

20210001 Math 90

20210002 Math 80

20210003 Math 70

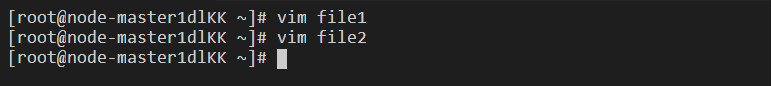
file2:

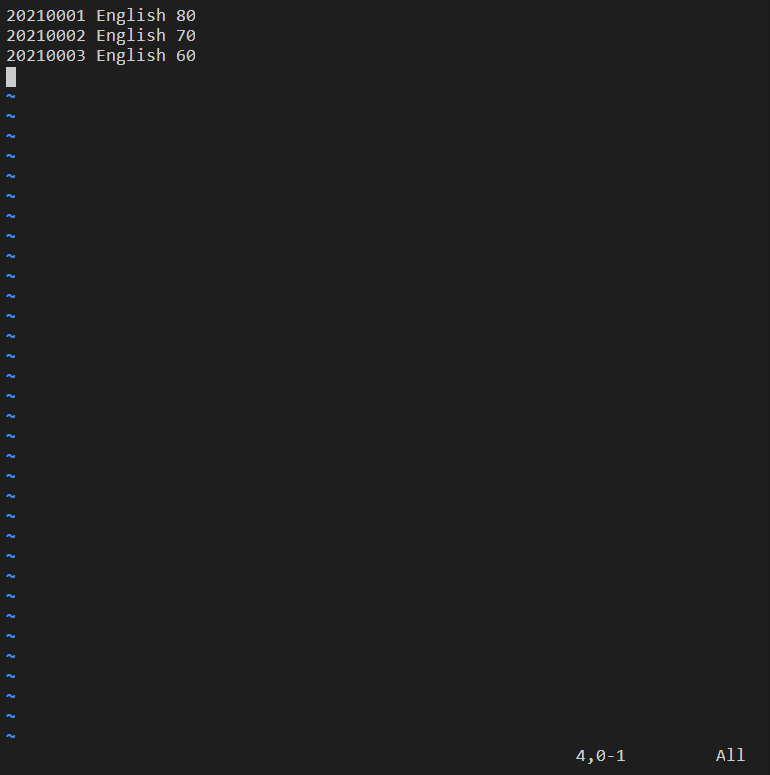
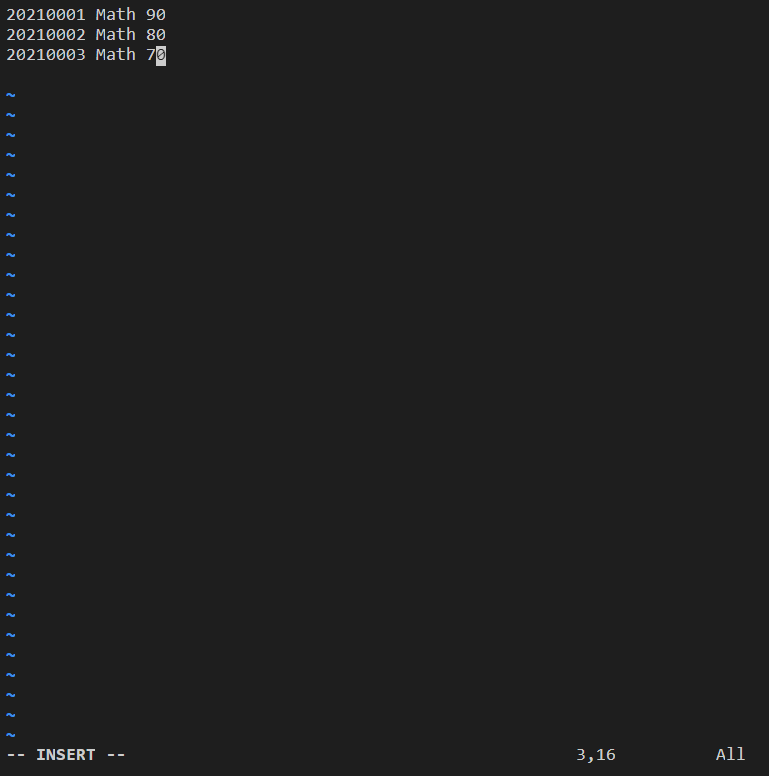
20210001 English 80

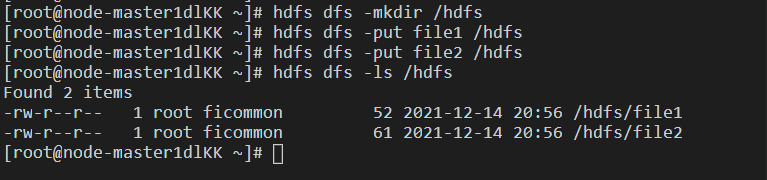
20210002 English 70

20210003 English 60

1.将以上两个文件存入hdfs(10’)







将上述内容编写入文件并存入名为hdfs文件夹内

2.编写mapreduce的程序，输出每门课的平均成绩。(10’)

3.编写mapreduce的程序，输出每位同学有多少门课成绩低于75分。(10’)

五：实验总结(10’)

通过本次实验，让我学会了在Hadoop平台上配置java环境并且编译.java文件对数据文件test1与test2进行处理以及在Spark平台上使用python代码对数据文件进行词频统计计算，从而实现WordCount功能（单词词频统计计算功能）。从中对MapReduce大数据并行处理的实现方法有了更多的认知。而本次最大的难点在于我还没学过Java这门语言，以至于在附加题最后两题上出现了卡顿，不知道如何实现那两个算法，同时还读不懂Java程序。因此我认识到了自己的短板在于缺少了Java编程语言方面的知识，在后面的实验开始前，我会好好地恶补一下这方面的知识，从而使实验完成得更加顺利。

**附录：**

1 MapReduce官方教程

<https://hadoop.apache.org/docs/r2.8.3/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

2 Spark官方教程

https://spark.apache.org/examples.html