**目录**

[实验1： 3](#_Toc163668107)

[实验2： 4](#_Toc163668108)

[实验3： 5](#_Toc163668109)

[实验总结 6](#_Toc163668110)

# 实验1： MapReduce实验部分

**1.1 实验概述**

介绍本次实验的目的意义、目标、要求及安排等

通过本实验，学习如何在Linux环境下配置Hadoop运行环境，在本地和伪分布式下安装和配置Hadoop，以及如何运行本地/伪分布式Hadoop实例。为后续的大数据处理和分析工作做好准备。

**1.2 实验内容**

介绍本次实验的总体主要内容

**1.2.1 阶段1 Hadoop安装与配置实验**

1.任务描述：通过本实验，学习如何在Linux环境下配置Hadoop运行环境，在本地和伪分布式下安装和配置Hadoop，以及如何运行本地/伪分布式Hadoop实例。为后续的大数据处理和分析工作做好准备

2.实验设计：无

3.实验过程：按照任务书的

4.实验结果：配置成功，正确显示hadoop版本号，如下图所示。

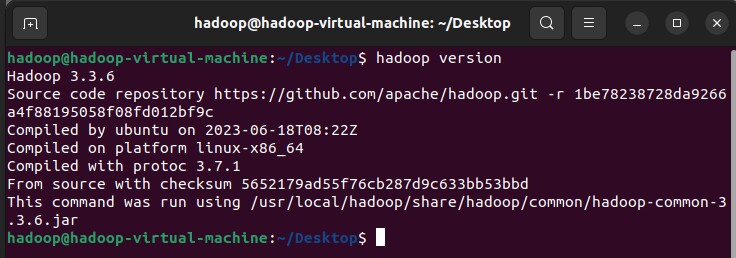


图1.1 hadoop版本为3.3.6

**1.2.2 阶段2 HDFS的配置、启动和使用**

1.任务描述：本实验旨在帮助学生深入理解HDFS在Hadoop体系结构中的角色，以及如何使用Hadoop提供的Shell命令和Java API来操作HDFS。通过此实验，学生将能够熟练地使用HDFS管理大规模数据，包括文件的存储、检索和操作。

2.实验设计：按照任务书中给出的提示shell命令，完成实验部分。

3.实验过程：由于这一部分要求多，这里给出部分过程：

首先我们配置好hdfs，并且通过start-dfs.sh启动hdfs服务，启动之后可以使用jps观察是否成功启动，如下图所示：

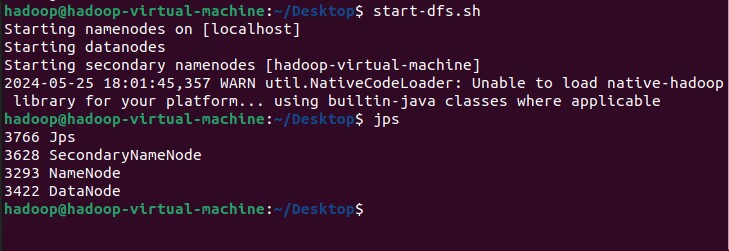


图1.2启动hdfs服务，并且查看进程

给定HDFS中某一个目录，输出该目录下的所有文件的读写权限、大小、创建时间、路径等信息，如果该文件是目录，则递归输出该目录下所有文件相关信息；

使用hadoop fs -ls -R -h/

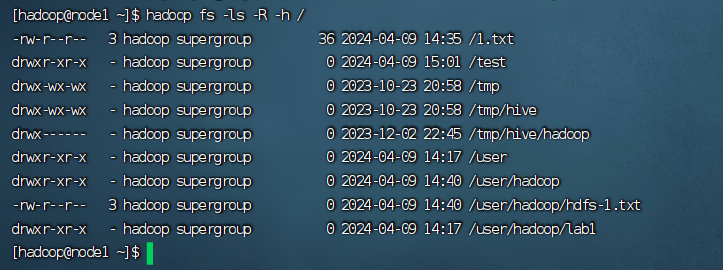


图1.3 hdfs操作

向HDFS中指定的文件追加内容，由用户指定内容追加到原有文件的开头或结尾；

首先在本地创建一个append.txt文件用于将内容追加到文件结尾。

追加到文件开头：

hdfs dfs -get text.txt

cat text.txt >> local.txt

hdfs dfs -copyFromLocal -f text.txt text.txt

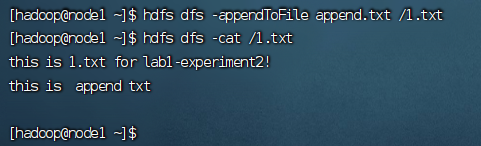


图1.4hdfs操作

4.实验结果：熟练掌握hdfs相关操作为下面的内容做准备

**1.2.3 阶段3 WordCount实验**

1.任务描述：本实验旨在通过编写和执行基于MapReduce编程模型的WordCount程序，帮助学生深入理解MapReduce的工作原理，并学会使用Hadoop框架进行大规模数据处理。通过此实验，学生将能够掌握MapReduce编程的基本概念、编写简单的MapReduce程序以及运行它们在分布式环境中。

2.实验设计：按照任务书中的博客内容来完成即可。

3.实验过程：

本次实验使用命令行和eclipse来编译打包hadoop mapreduce程序：

1. 使用eclipse来编译打包：

首先需要按照博客的教程，完成eclipse的安装以及eclipse插件的安装，在两者安装好之后在eclipse文件夹中使用命令./eclipse来运行eclipse软件：

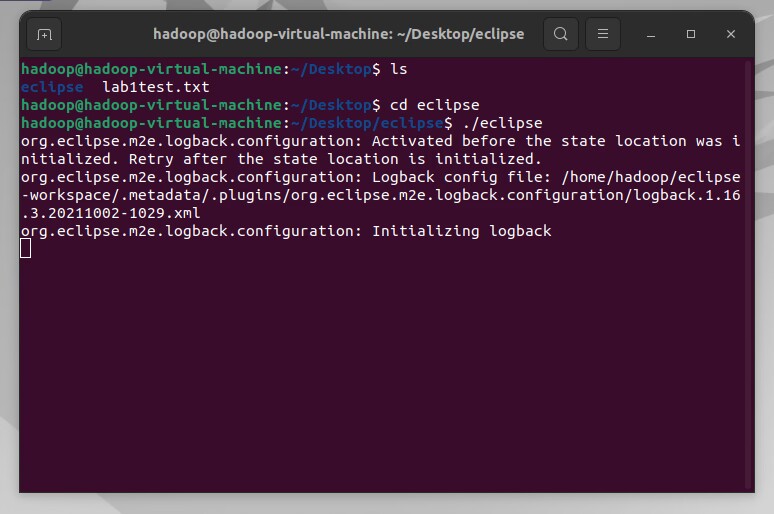


图1.5 成功运行eclipse

运行成功后会打开eclipse软件的界面，接下来按照博客里的教程在eclipse中编写wordcount的程序，并且在ubuntu终端中使用start-dfs.sh来开启hdfs服务，本次实验均需要使用到。

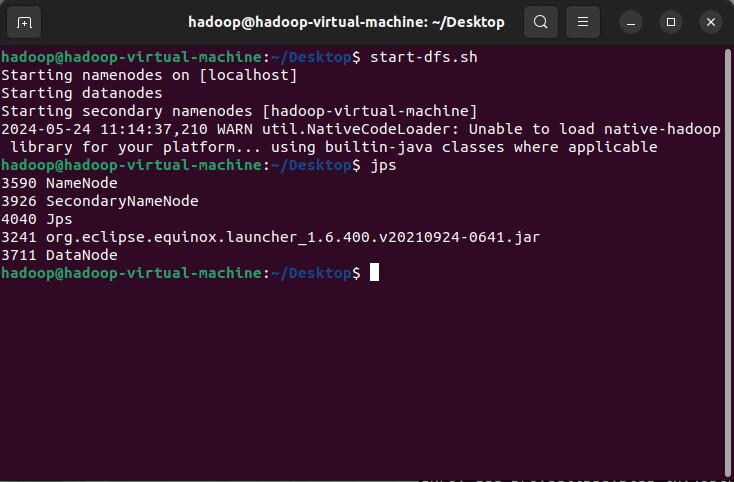


图1.6 打开hdfs为实验做准备

运行之后可以使用jps来查看是否开启了结点。在wordcount文件中由于我们指定了读取hdfs系统中的input文件夹里的文件，我们需要提前在hdfs系统上传上我们想要统计的文件。使用hdfs dfs -mkdir input来创建文件夹，使用hdfs dfs -put lab1test.txt ./input 将文件上传到input文件夹中。随后使用hdfs dfs -ls和hdfs dfs -ls input指令我们可以查看相关文件是否已经上传到hdfs上，我们亦可以通过在eclipse中查看hdfs上的文件直接图形化观察，如下图所示。

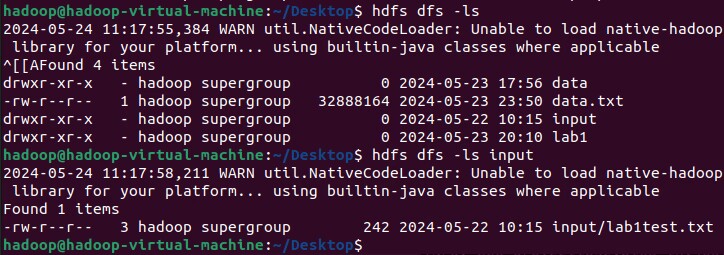


图1.7 上传数据文件到hdfs系统中

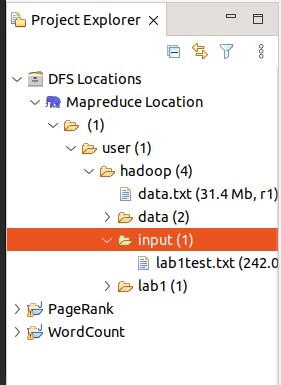


图1.8在eclipse中可以观察到dfs系统中文件存在

4.实验结果：在运行wordcount项目之前需要删除掉已经有的output文件夹，否则会报错，配置完成之后，直接运行run on Hadoop即可，运行完毕后左侧文件栏右键选择刷新，即可在hdfs系统中看到output文件夹，选择里面的part-r-00000文件，就可以看到wordcount运行的结果，如下图所示：

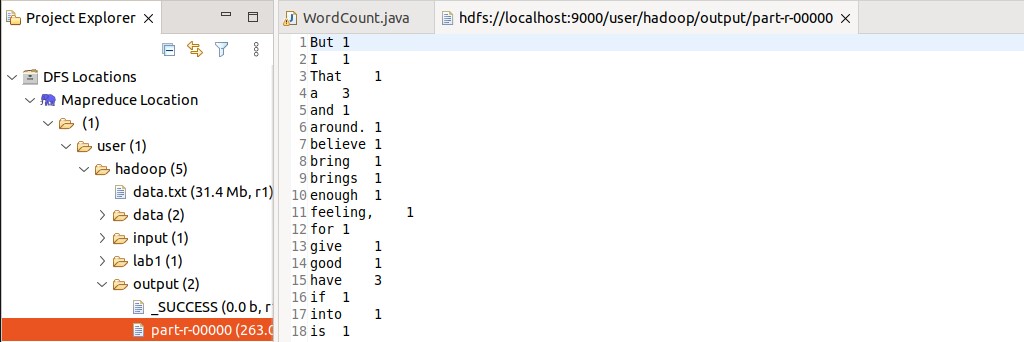


图1.9 wordcount运行结果

**1.2.4 阶段4 PageRank实验**

1.任务描述：本实验旨在通过编写和执行基于MapReduce编程模型的PageRank程序，帮助学生深入理解MapReduce的工作原理，并学会使用Hadoop框架进行大规模数据处理。通过此实验，学生将能够掌握MapReduce编程的基本概念、编写简单的MapReduce程序以及运行它们在分布式环境中。

2.实验设计：按照任务书中来完成实验

3.实验过程：

在eclipse中我们创建了wordcount项目，我们接着创建pagerank项目，并且选择mapreduce类型项目，填写项目相关信息即可。按照wordcount项目的创造过程即可，包名为org.apache.hadoop.examples，在这个包内，我们创建了EdgeCounter.java、PageRank.java和SortRank.java。其中EdgeCounter.java文件中我们需要统计data.txt文件的图中节点数、提取并统计图中的边信息以及为结点添加初始的权重。SortRank.java文件中将所有的文本输出成（权重值，id）的键值对，随后按照键值对中的权重值排序输出。PageRank.java文件负责按照pagerank算法的公式来进行迭代，如果迭代次数到达规定值或者迭代的误差值在规定值之内就可以判断是收敛的并且将最终的结果输出保存下来。

在PageRank.java文件中，我们需要将路径从main函数输入，所以我们需要将项目的输入值进行配置，在右侧项目右键选择run as选项，里面有run configurations，可见下面图片。

本次实验数据从eclipse-workspace文件夹中的PageRank项目文件夹读取，我们将所需数据移入文件夹即可。

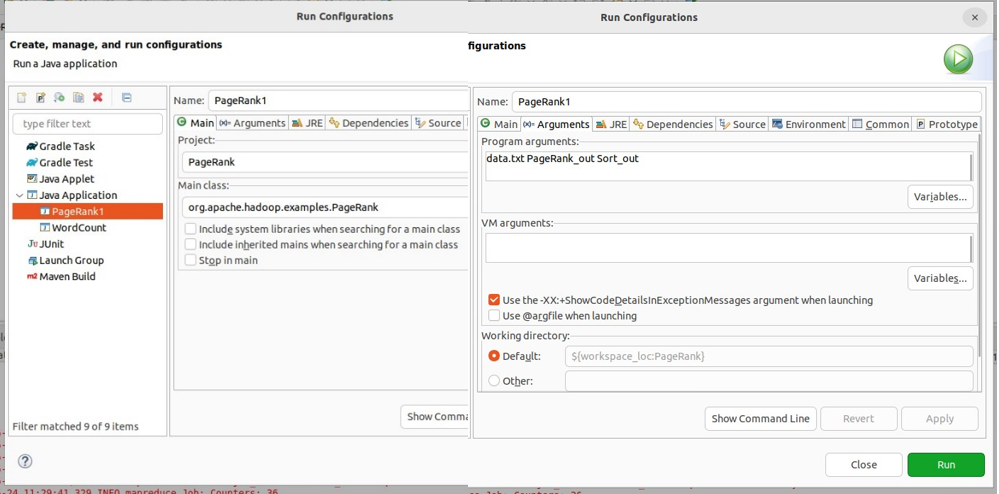


图1.10 在eclipse中调整项目的运行配置

4.实验结果：配置完成之后，直接选择run进行运行即可，由于数据量比较大时间需要比较久，完成之后即可在项目中看到输出的答案。

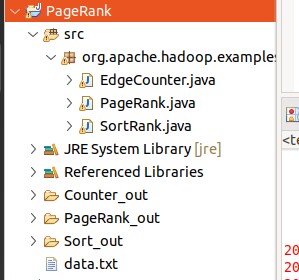


图1.11 在dfs中可以看到项目运行结果

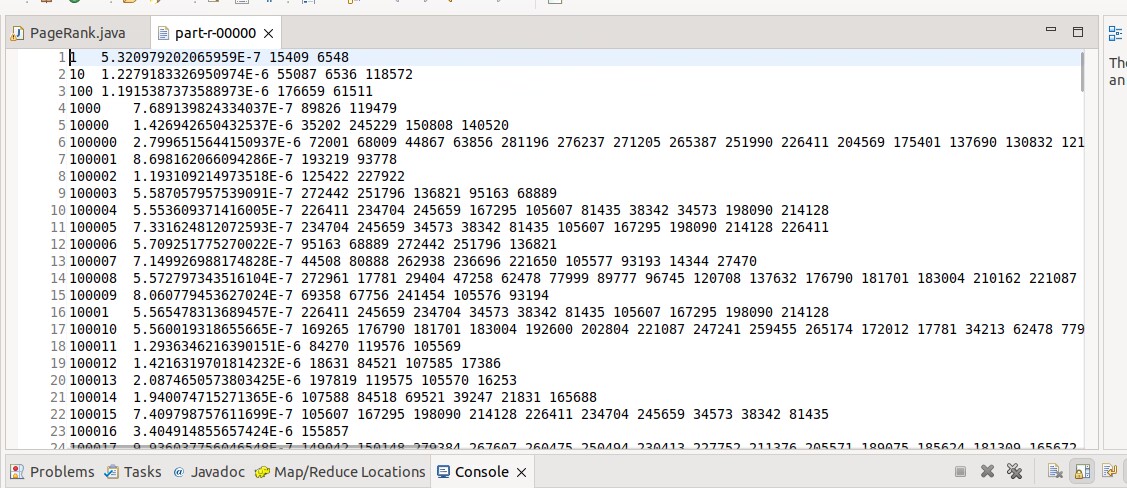


图1.12 在dfs中可以看到项目运行结果

**1.3 实验小结**

本次实验我掌握了如何在eclipse软件中编写map/reduce的项目，首先需要导入相关的包，在编写程序的时候需要导入与hadoop和mapreduce相关的库。在程序中需要通过使用相关的函数来进行map、shuffle、combine、reduce的相关的部分操作。

在实验配置的过程中，林子雨老师博客写的十分详细，并且有些小问题还需要自己额外再查找相关内容来解决。比如eclipse的软件在导入ubuntu系统后，还需要装插件，需要修改相关文件操作权限来实现，这一部分折磨了我很久。

在pagerank的部分，其实在去年已经用python模拟过这一个算法，在今年的实验中加深了我对该算法的理解。算法中需要多次迭代，提前规定好收敛的范围以及最大迭代次数。我们需要借助相关内容来处理文件，将顶点和边分别处理完成，带入到迭代公式里去计算。

# 实验2：

**2.1 实验概述**

介绍本次实验的目的意义、目标、要求及安排等

**2.2 实验内容**

介绍本次实验的总体主要内容

**2.2.1 阶段1 XXXXXXX**

1.任务描述：给出阶段1的任务描述

2.实验设计：给出解题思路分析和拟采用的技术和方法等

3.实验过程：详细描述实验的具体过程

4.实验结果：给出阶段1的实验结果和必要的结果分析

……

**2.2.2 阶段2 WordCount实验**

1.任务描述：给出阶段x的任务描述

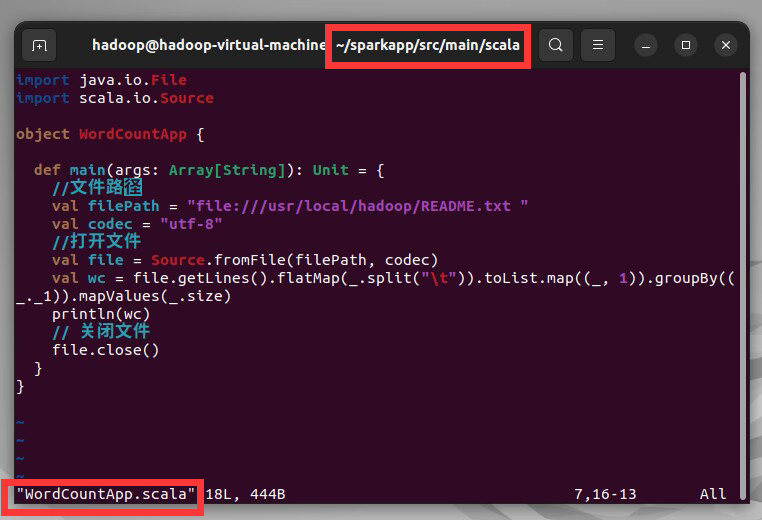
2.实验设计：给出解题思路分析和拟采用的技术和方法等

3.实验过程：详细描述实验的具体过程

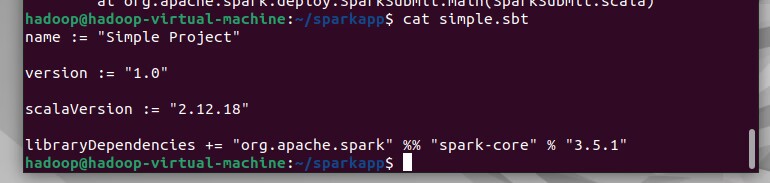
4.实验结果：给出阶段x的实验结果和必要的结果分析

用scala语言实现wordcount：

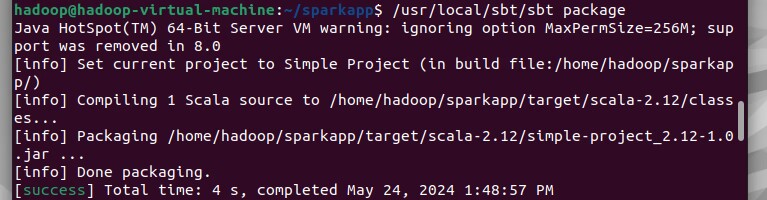
创建文件夹sparkapp，在文件夹下创建WordCountApp.scala文件，vim /src/main/scala/WordCountApp.scala如下图所示。



这里的文件路径可以看到是[file:///usr/local/hadoop/README.txt](file:///C:\\usr\\local\\hadoop\\README.txt)，保证这里的文件存在，可以查看是否存在。编写完WordCountApp之后，我们需要退回到/sparkapp文件夹中，先使用vim simple.sbt编写sbt文件为后续的打包进行准备，文件内容如下所示：



使用/usr/local/sbt/sbt package命令来对我们编写好的内容进行打包：

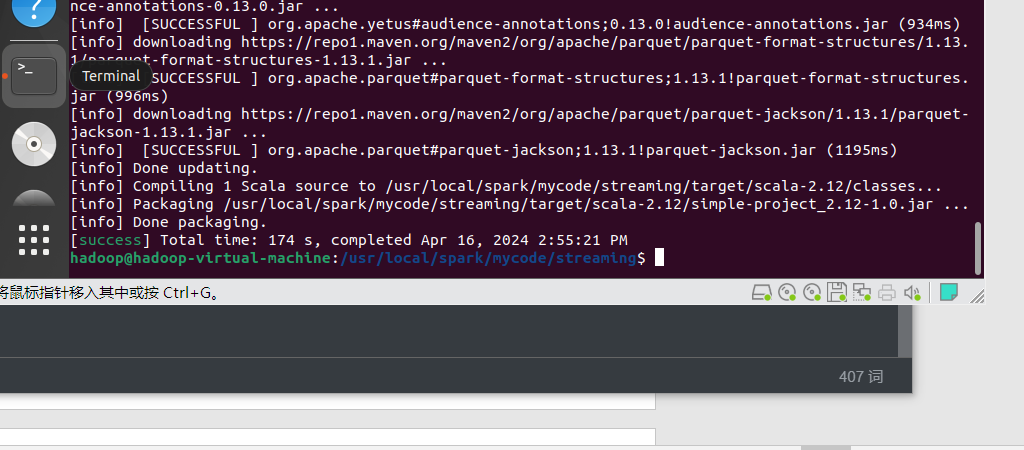


使用/usr/local/spark/bin/spark-submit --class "WordCountApp" ~/sparkapp/target/scala-2.12/simple-project\_2.12-1.0.jar就可以将打包好的jar包直接开始运行。

这里的问题在于，我们需要提前确认自己scala的版本以及spark的详细版本，并且编写的时候需要进行替换即可。

**2.2.3 阶段3 Spark Streaming实验**

我们按照博客https://dblab.xmu.edu.cn/blog/1387/中的指导，在/usr/local/spark文件夹中创建mycode文件夹，然后在mycode中创建了streaming文件夹来完成本次实验。在文件中同样有：/src/main/scala的文件夹，然后在scala下创建NetworkWordCount.scala和StreamingExamples.scala文件，这里内容按照博客里的来完成就可以。我们同样需要在streaming文件夹内配置simple.sbt文件来为打包做准备，编写好之后就可以使用/usr/local/sbt/sbt package来直接打包了，本次打包时间较久，如下所示



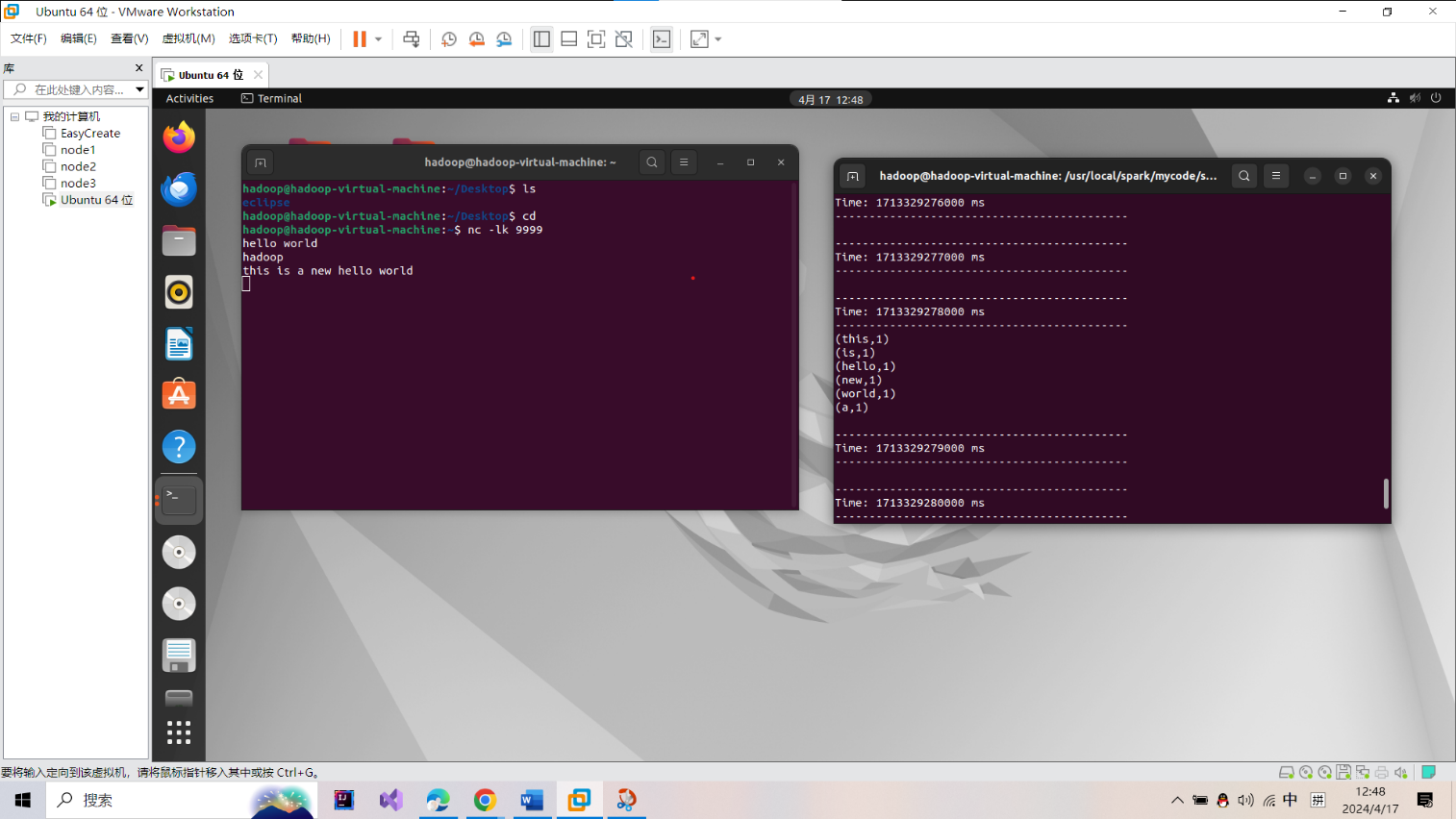
打包完成之后在streaming文件夹中使用下面的命令来进行submit即可。

cd /usr/local/spark/mycode/streaming

“/usr/local/spark/bin/spark-submit --class "org.apache.spark.examples.streaming.NetworkWordCount" /usr/local/spark/mycode/streaming/target/scala-2.12/simple-project\_2.12-1.0.jar localhost 9999”

在运行上述jar包后，当前的终端窗口会报告time以及connection refused的错误，这是由于我们还没有启动nc窗口，再打开一个终端窗口，输入以下代码即可：nc -lk 9999。

之后我们就可以在这个nc窗口中输入一些单词就可以在右边运行了jar包的窗口中看到统计单词的结果，可见下面图片。

****

**2.3 实验小结**

对本次实验使用的理论、技术、方法和结果进行总结。描述一下通过实验你有哪些收获。

# 实验3：

**3.1 实验概述**

介绍本次实验的目的意义、目标、要求及安排等

**3.2 实验内容**

介绍本次实验的总体主要内容

**3.2.1 阶段1 XXXXXXX**

1.任务描述：给出阶段1的任务描述

2.实验设计：给出解题思路分析和拟采用的技术和方法等

3.实验过程：详细描述实验的具体过程

4.实验结果：给出阶段1的实验结果和必要的结果分析

……

**3.2.2 阶段2 DFS实验和SCC实验**

1.任务描述：给出阶段x的任务描述

2.实验设计：给出解题思路分析和拟采用的技术和方法等

3.实验过程：详细描述实验的具体过程

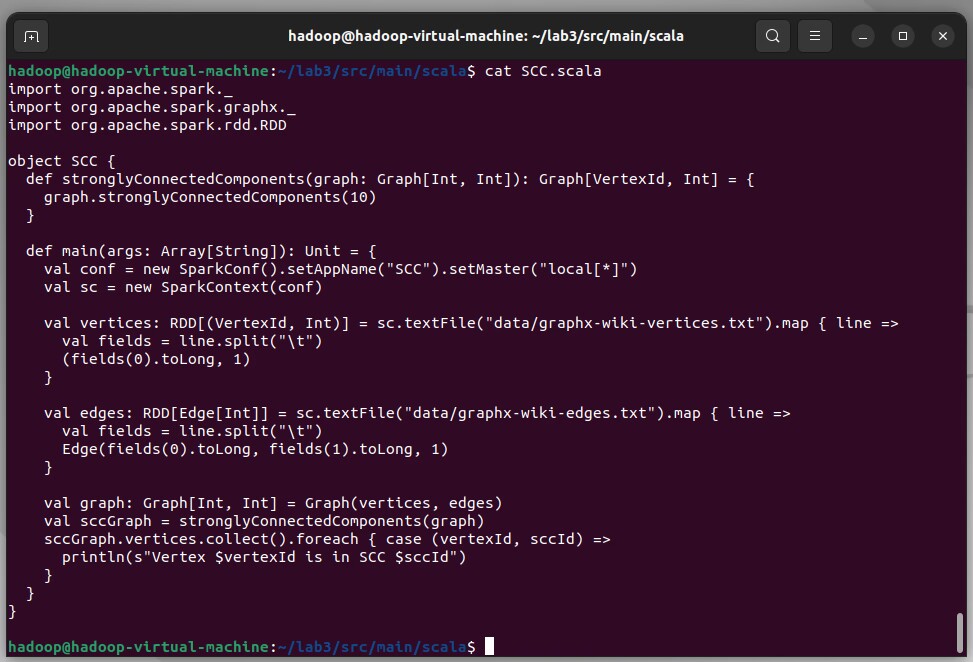
4.实验结果：给出阶段x的实验结果和必要的结果分析

由于电脑的gpu性能不够，所以我们采用spark中的graphx库来进行图计算实验。我们使用scala语言和实验2中类似，先编写scala程序，然后编写sbt文件，接着打包scala程序为jar包，最后直接submit提交就可以了。

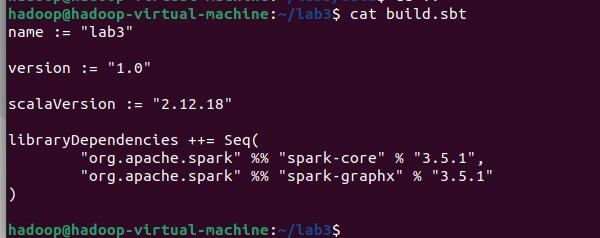
我们在根文件夹底下创建lab3/src/main/scala文件夹，在scala文件夹创建DFS.scala和SCC.scala即可。

DFS算法编写，我们需要先创建两个集合visited和result分别保存已经访问过的结点和存储dfs结果路径。接下来，我们需要定义一个内部递归函数，从一个顶点开始如果未访问过则将这个添加到visited集合中，并且将当前的顶点id添加到result集合中。然后获取它的邻居结点，对于每个结点再去调用dfs函数，继续探索该邻居结点的分支。

对于SCC算法，我们可以直接使用graphx库中的stronglyConnectedComponents函数直接来计算强连通分量，详细的可见下方图片。



两个文件编写完成后，在lab3文件中，先编写sbt文件，由于本次scala程序中需要使用graphx库，所以我们在sbt文件需要添加graphx库作为依赖，如下所示。



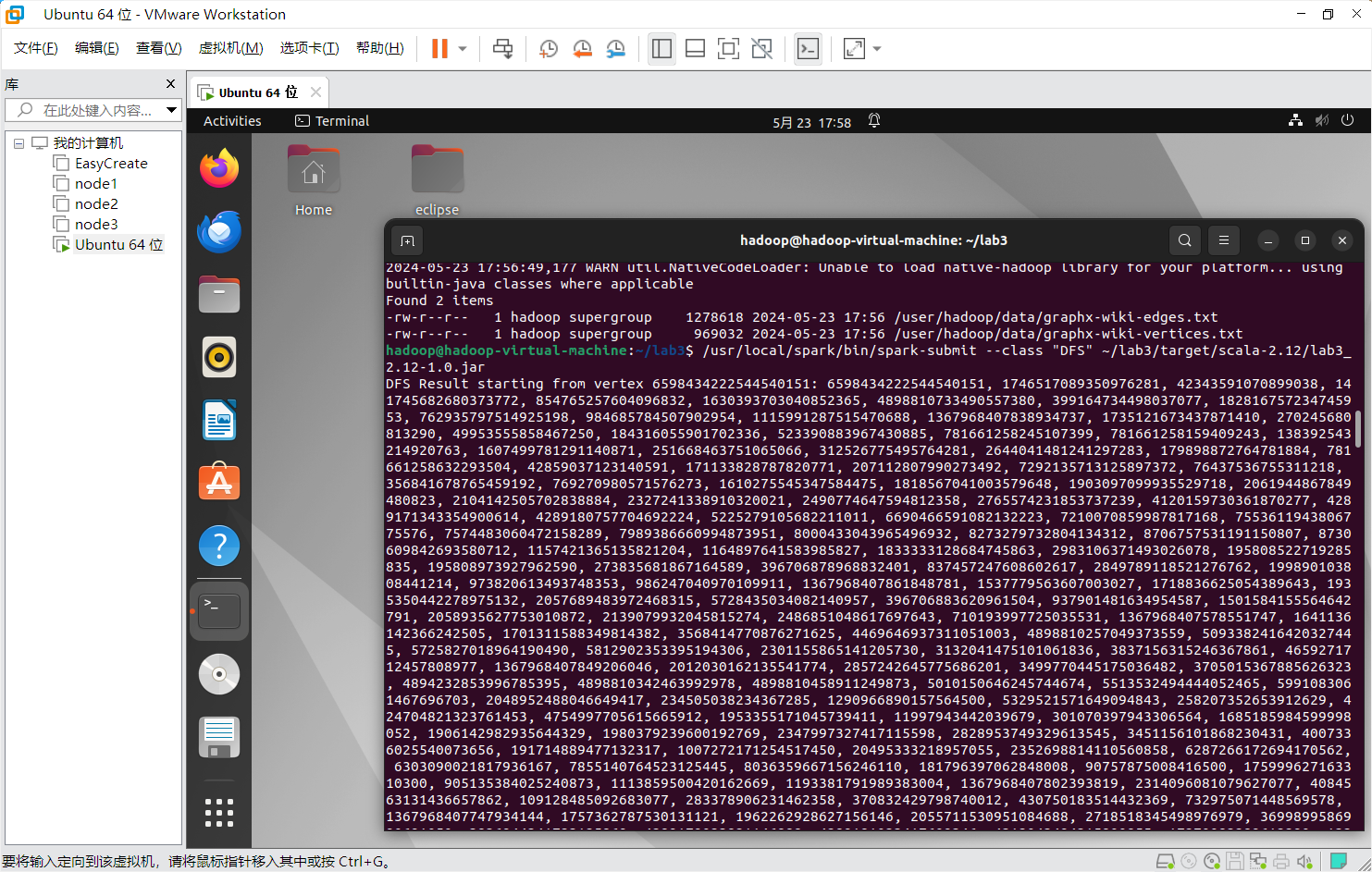
使用/usr/local/sbt/sbt package来进行打包。在程序中我们指定了读取数据路径就在lab3中所以我们将data数据移动到lab3中即可。打包完成后我们分别使用:

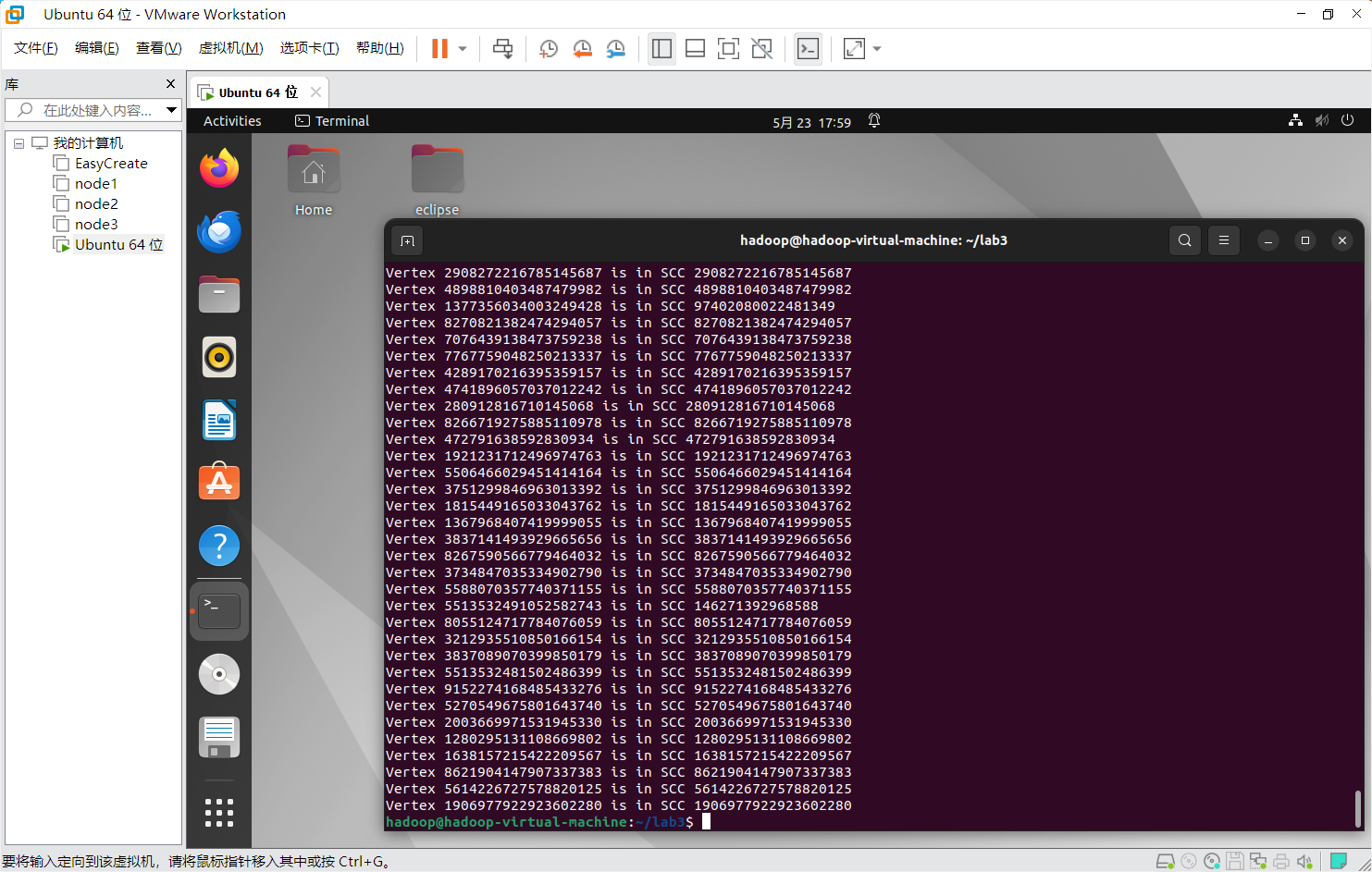
/usr/local/spark/bin/spark-submit --class "DFS" ~/lab3/target/scala-2.12/lab3\_2.12-1.0.jar

和

/usr/local/spark/bin/spark-submit --class "SCC" ~/lab3/target/scala-2.12/lab3\_2.12-1.0.jar

就可以来运行这两个jar包，并且得到结果，如下图所示





**3.3实验小结**

对本次实验使用的理论、技术、方法和结果进行总结。描述一下通过实验你有哪些收获。

# 实验总结

全面总结实验成果，描述通过实验得到收获（400字以上）