电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2021090921019\_\_ \_

姓 名 杨径骁\_\_\_\_ \_\_\_

（实验） 课程名称 大数据分析与智能计算\_ \_

理论教师 汤宇、林迪\_\_\_\_\_ \_

实验教师 杨珊\_\_\_\_\_\_\_\_ \_

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名： 杨径骁 学号： 2021090921019 指导教师：杨珊**

**实验地点： 二教110 实验时间：2023.10.10**

1. **实验名称：**Hadoop和Spark安装配置
2. **实验学时：**2学时
3. **实验目的：**

1. 掌握Hadoop的环境搭建

2. 掌握Hadoop、Linux的基本命令。

1. **实验原理**

首先需要创建用户，然后授予用户权利，在linux系统上，如果用户没有读写运行的权利将会有许多事务收到限制。然后安装ssh，安装ssh的目的在能在后期做多设备hadoop实验时候能够登录到别人的设备上，并且无需密码。运行的wordCount代码在hadoop的share包里，能够检索文本里面单词出现的次数，但这次实验只在单机上运行，所以无论输入和输出都在单机上进行。

1. **实验内容**

首先安装SSH，SSH 为 Secure Shell 的缩写，由 IETF 的网络小组（Network Working Group）所制定；SSH 为建立在应用层基础上的安全协议。安装SSH的目的在于为了方便配置集群,能在一台机器上登陆到集群中其他机器上。

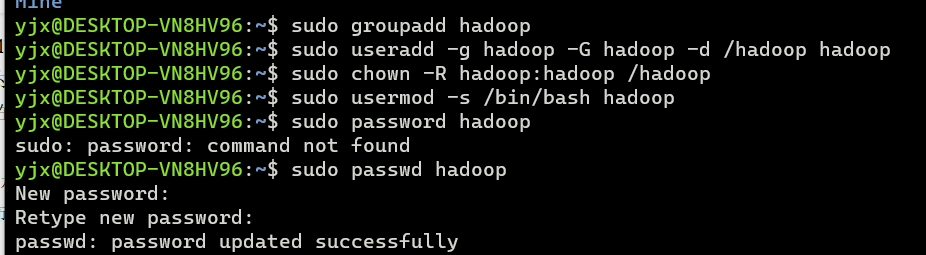
然后在ubuntu上安装hadoop，Hadoop是一个由Apache基金会所开发的分布式系统基础架构。用户可以在不了解分布式底层细节的情况下，开发分布式程序。

最后安装Spark.

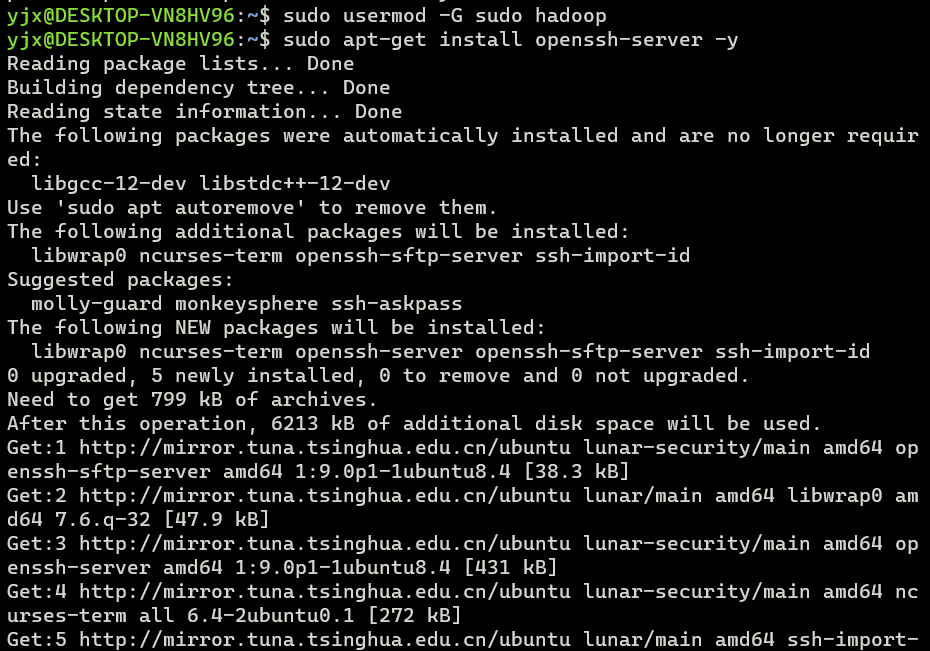
1. **实验设备及环境**

VMware Workstation Pro，Ubuntu-20.04.6

1. **实验步骤**
2. **Hadoop安装配置**
3. 添加用户及用户组

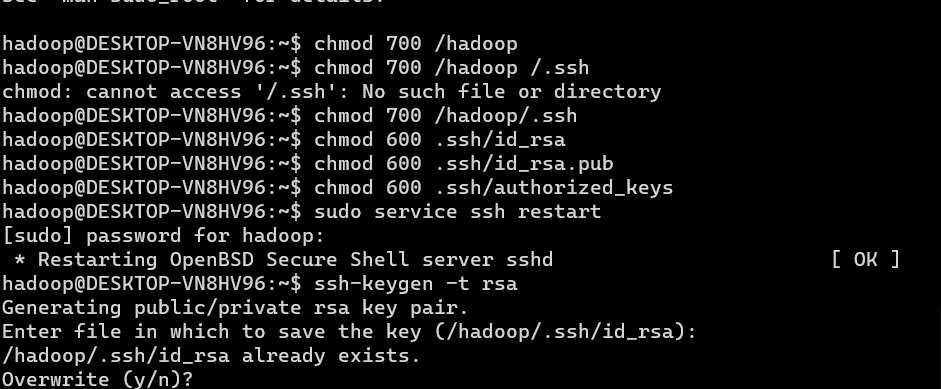


1. 添加sudo权限及安装及配置依赖的软件包

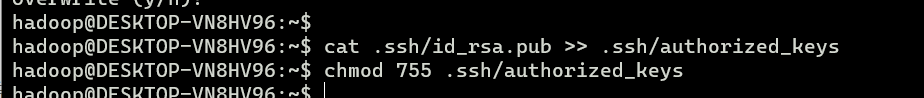


1. 配置ssh免密登录

3.1更改权限

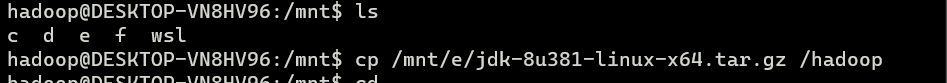


3.2设置免密



1. 下载安装Java和Hadoop

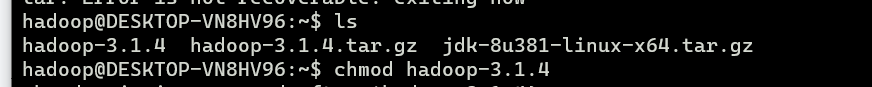
4.1下载jdk1.8并拷贝到/hadoop下（注：这里我使用的是WSL）

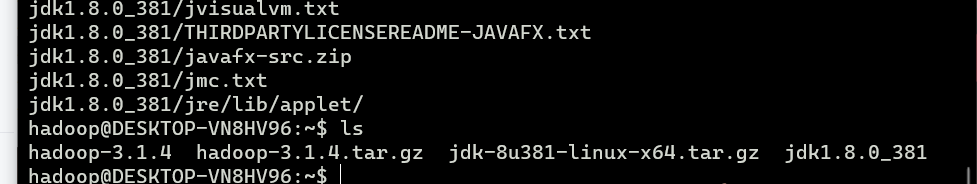


4.2下载Hadoop并拷贝到/hadoop下



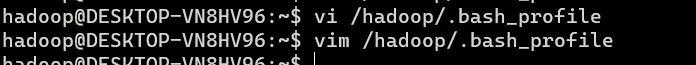
4.3解压并安装,修改权限



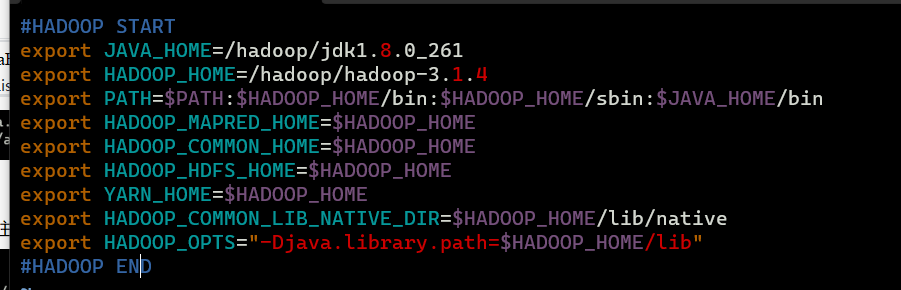


1. 配置Hadoop环境变量

5.1修改配置文件



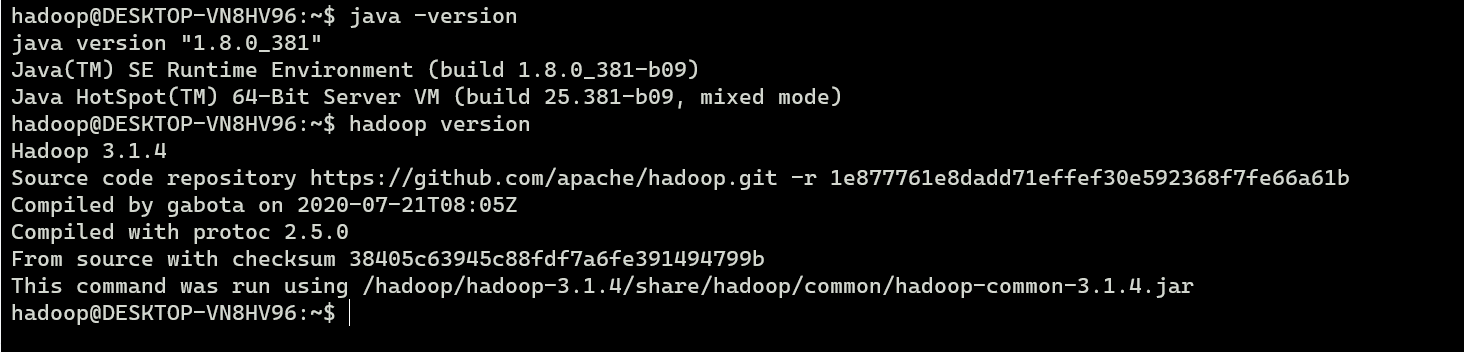
注意：指导教程缺少第三行，需要自行加上



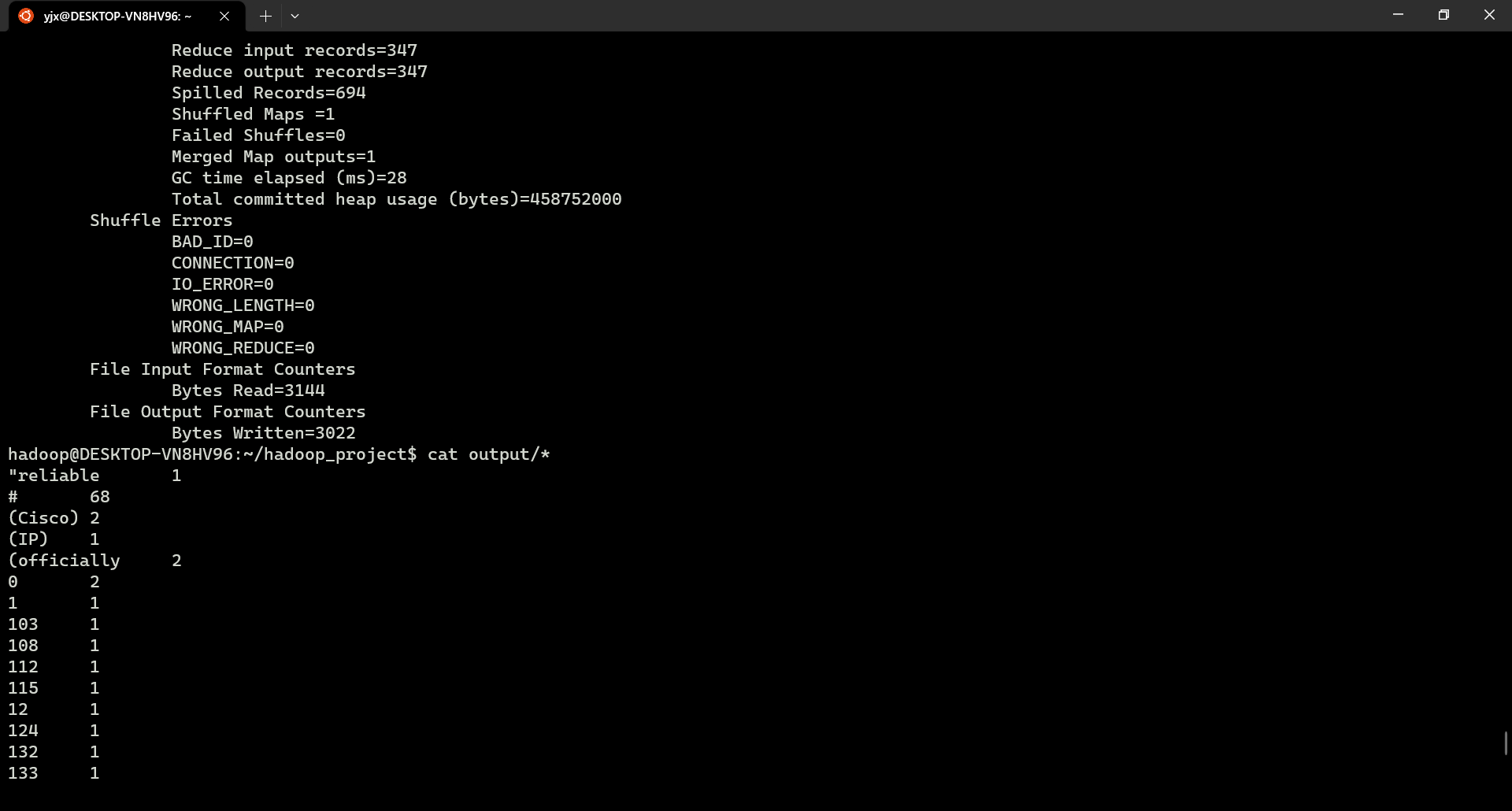
5.2 激活新增环境变量



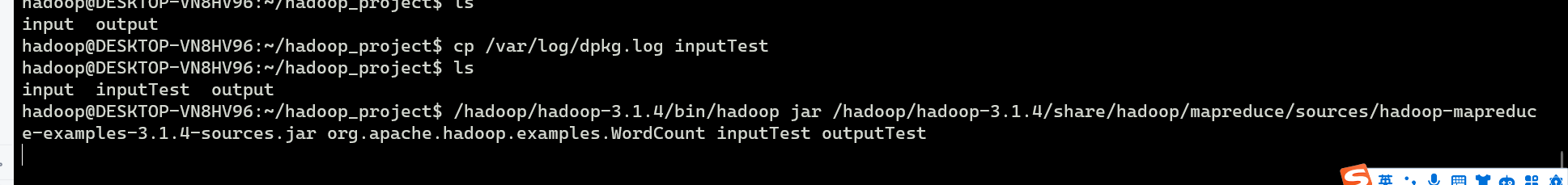
5.3查看Java和Hadoop的版本信息

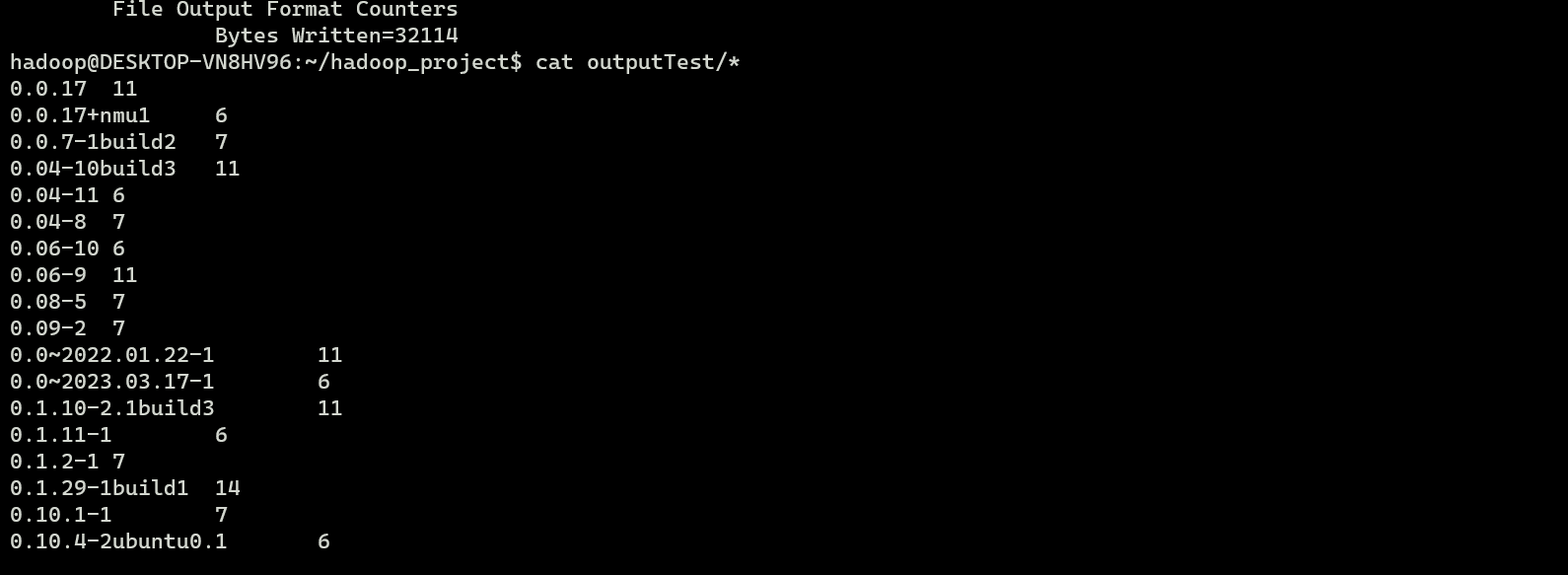


1. 测试，词频统计



1. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**





|  |
| --- |
| /hadoop/hadoop-3.1.4/bin/hadoop jar /hadoop/hadoop-3.1.4/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-3.1.4-sources.jar org.apache.hadoop.examples.WordCount inputTest outputTest |

该代码展示了如何使用Hadoop MapReduce框架来运行一个简单的WordCount示例。对日志文件/var/log/dpkg.log进行词频统计。

/hadoop/hadoop-3.1.4/bin/hadoop:这是Hadoop的命令行工具，用于执行Hadoop任务和操作。

Jar：表示运行的任务是一个Java jar文件

/hadoop/hadoop-3.1.4/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-3.1.4-sources.jar：这是jar文件的路径，包含了示例带Hadoop MapReduce示例代码。

org.apache.hadoop.examples.WordCount ：要运行的Java类的全限定名。WordCount 是示例中的一个类，它实现了WordCount任务，用于计算文本中各个单词出现的次数。

inputTest outputTest：这是我自己命名的输入输出文件。

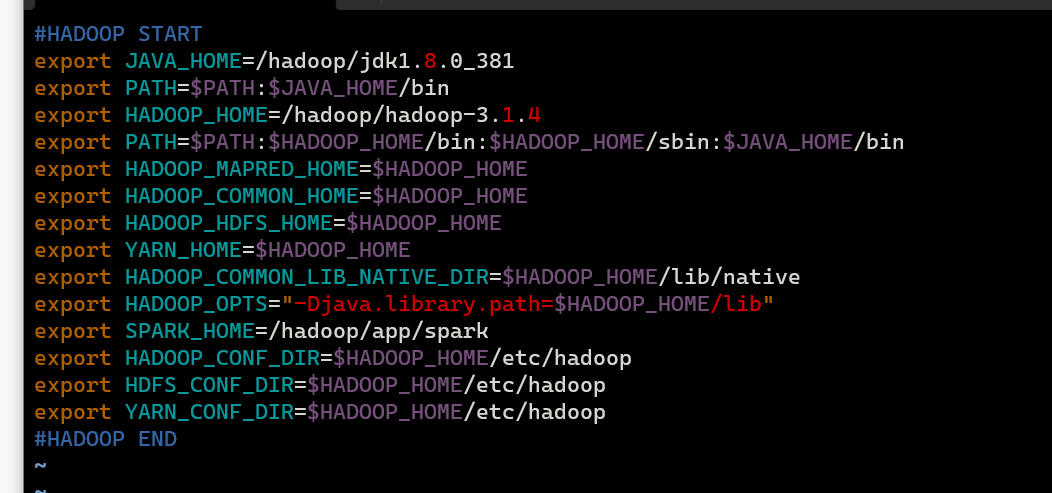
## 二、Spark安装配置

**一.hadoop伪分布式配置**

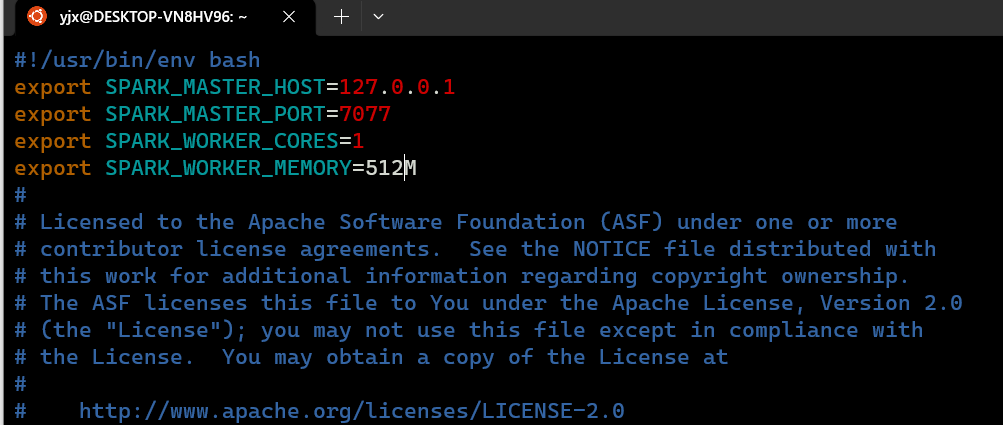
1.修改配置文件

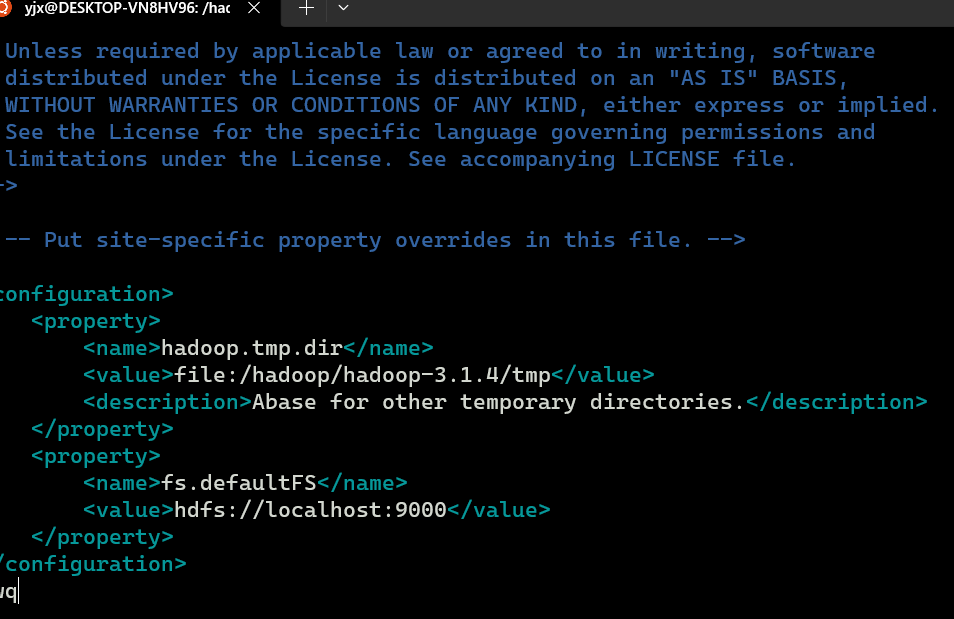
在 Yarn 上运行 Spark 需要配置 HADOOP\_CONF\_DIR、 YARN\_CONF\_DIR 和

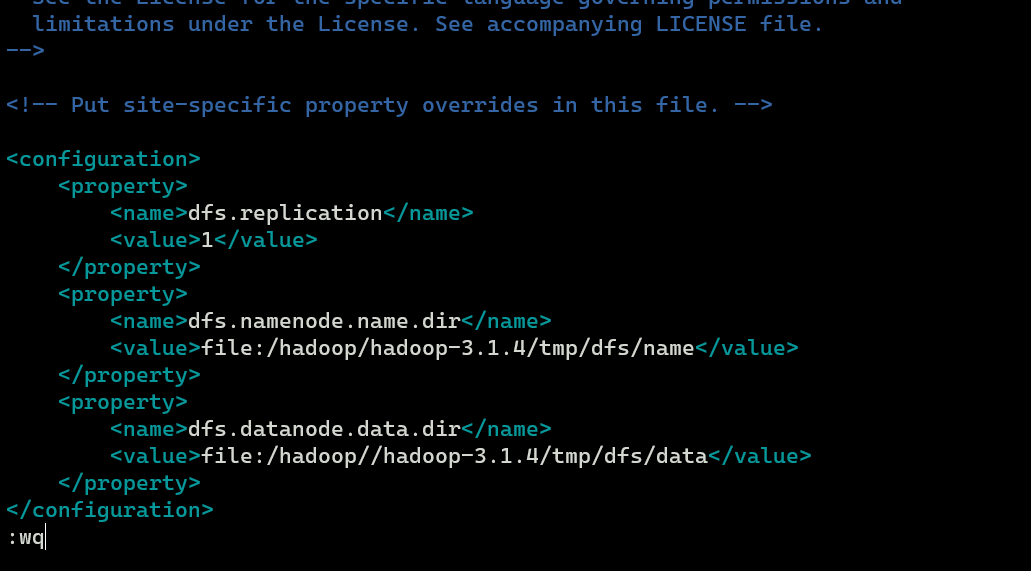
HDFS\_CONF\_DIR 环境变量命令



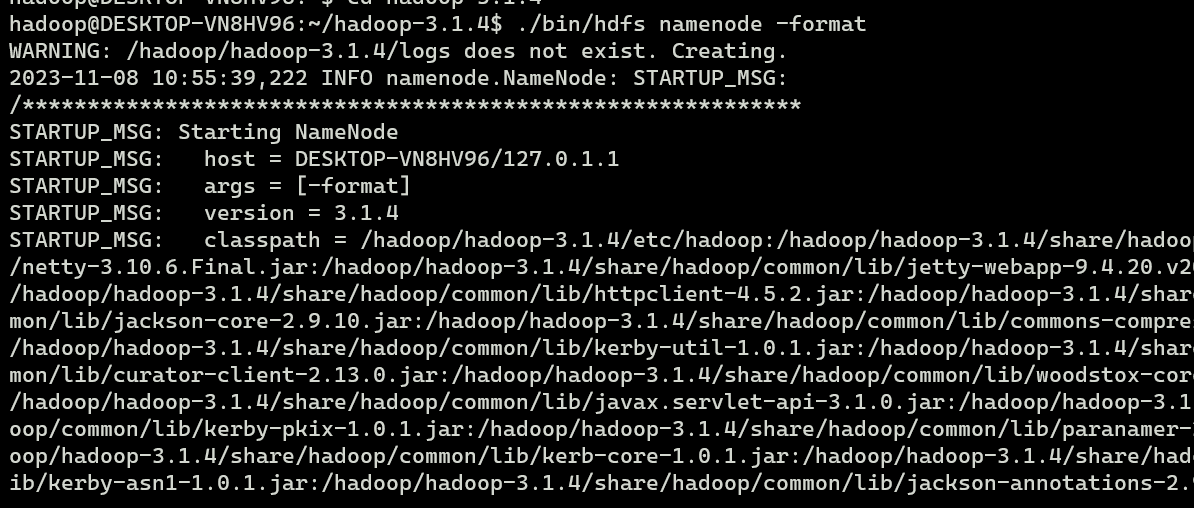
1.1修改配置文件

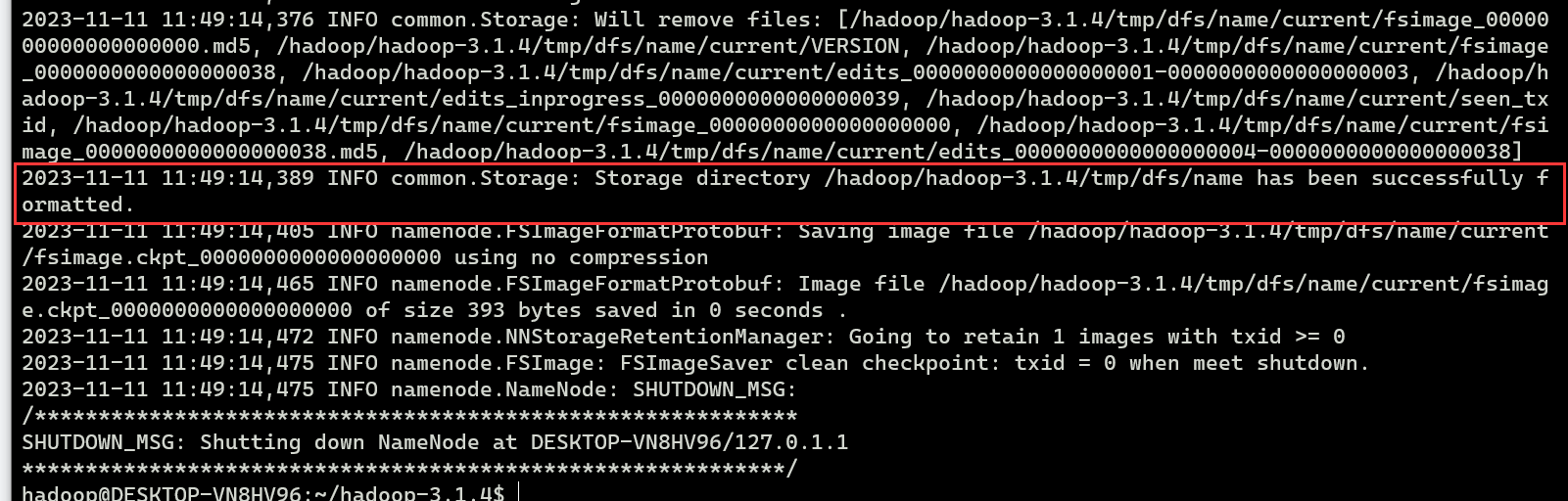




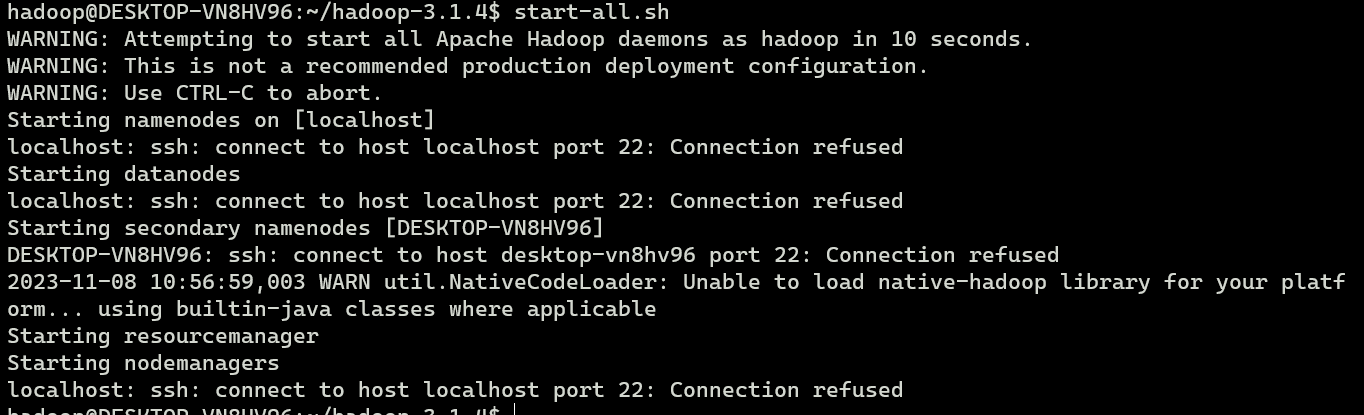


1.2namenode的格式化



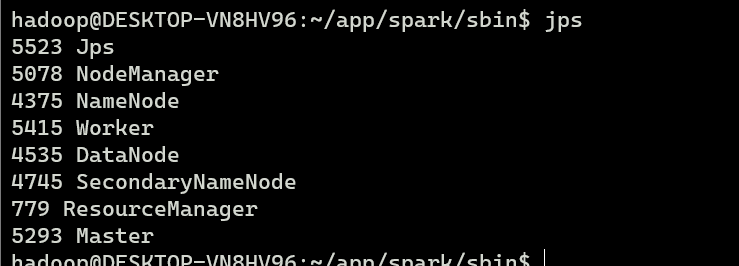


1.3.启动hadoop（namenode节点）（start-all.sh在sbin里面）



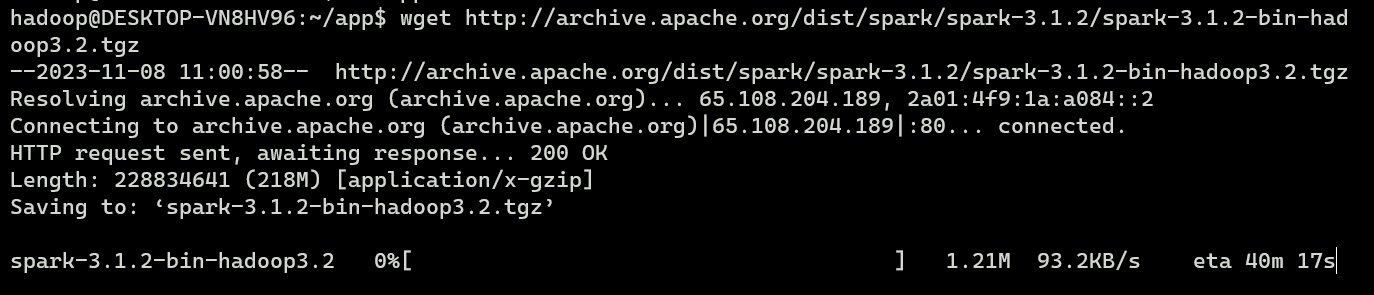
(后续已解决)

1.4 检查是否运行成功

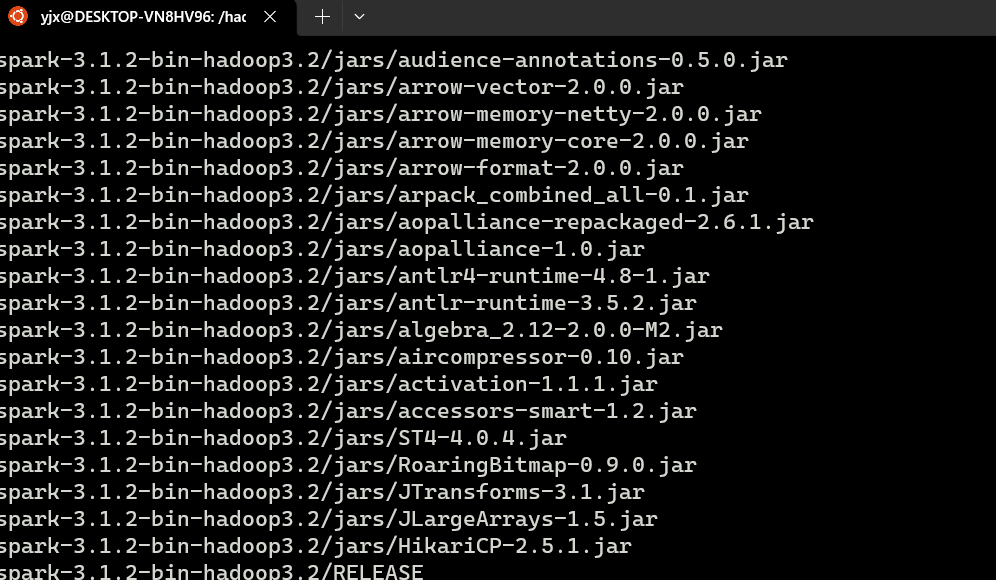


2.安装与配置 Spark

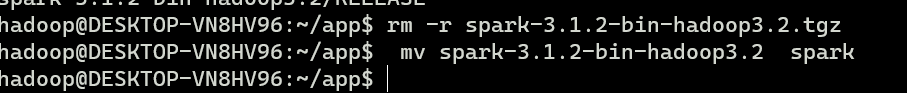
2.1解压并安装Spark



解压

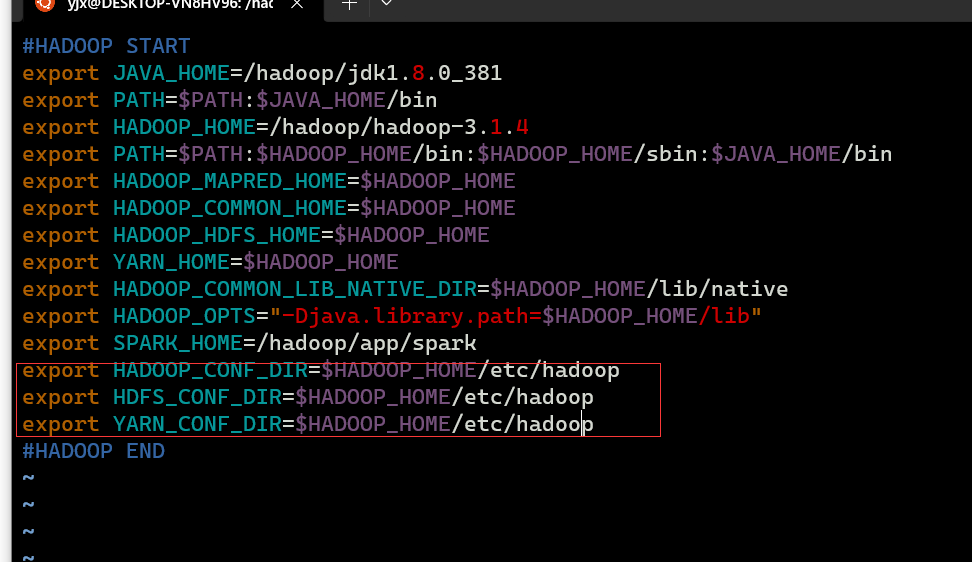


删除安装文件、修改文件名称



2.2配置 Hadoop 环境变量

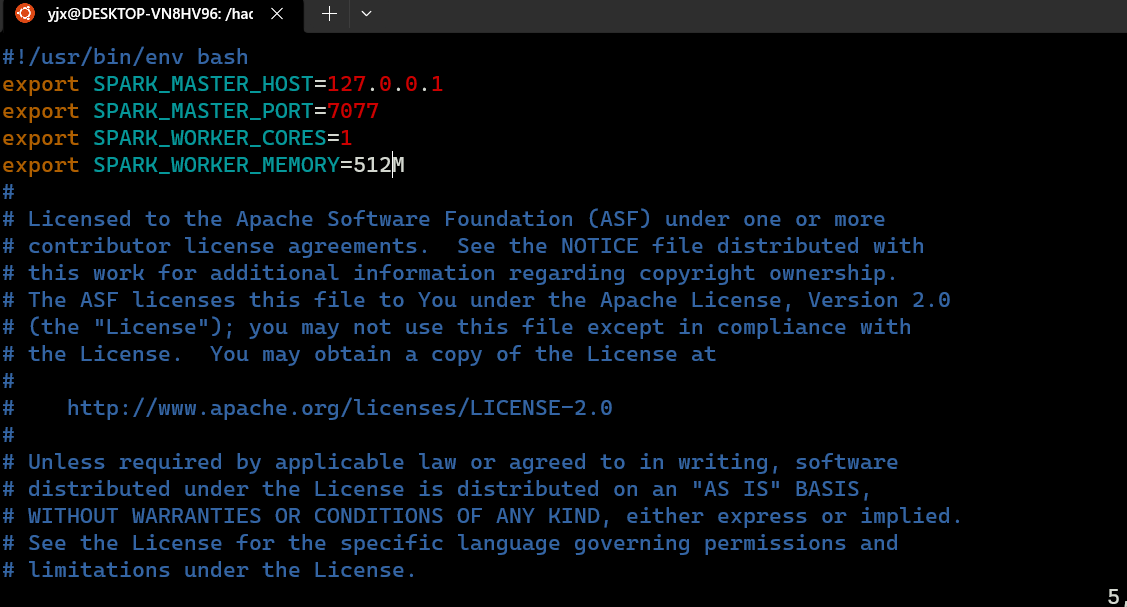
在 Yarn 上运行 Spark 需要配置 HADOOP\_CONF\_DIR、 YARN\_CONF\_DIR 和 HDFS\_CONF\_DIR 环境变量



激活环境变量



2.3修改配置文件

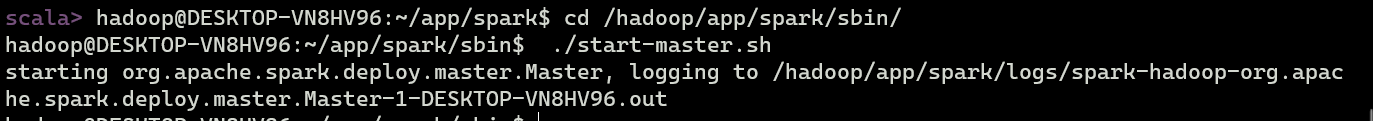


#### 3. Spark的 启动

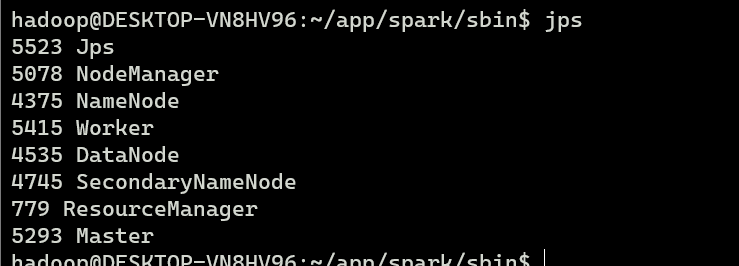
3.1进入spark-shell



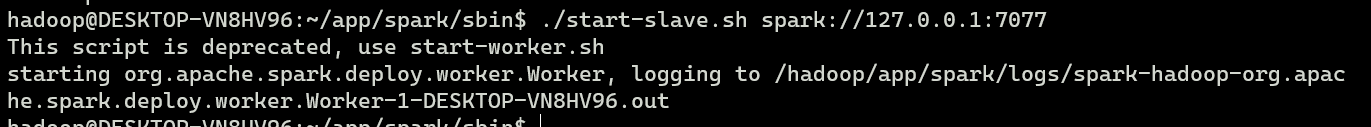
3.2启动spark



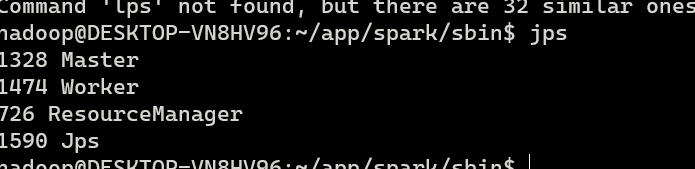
(1)查看master是否启动



1. 启动slave

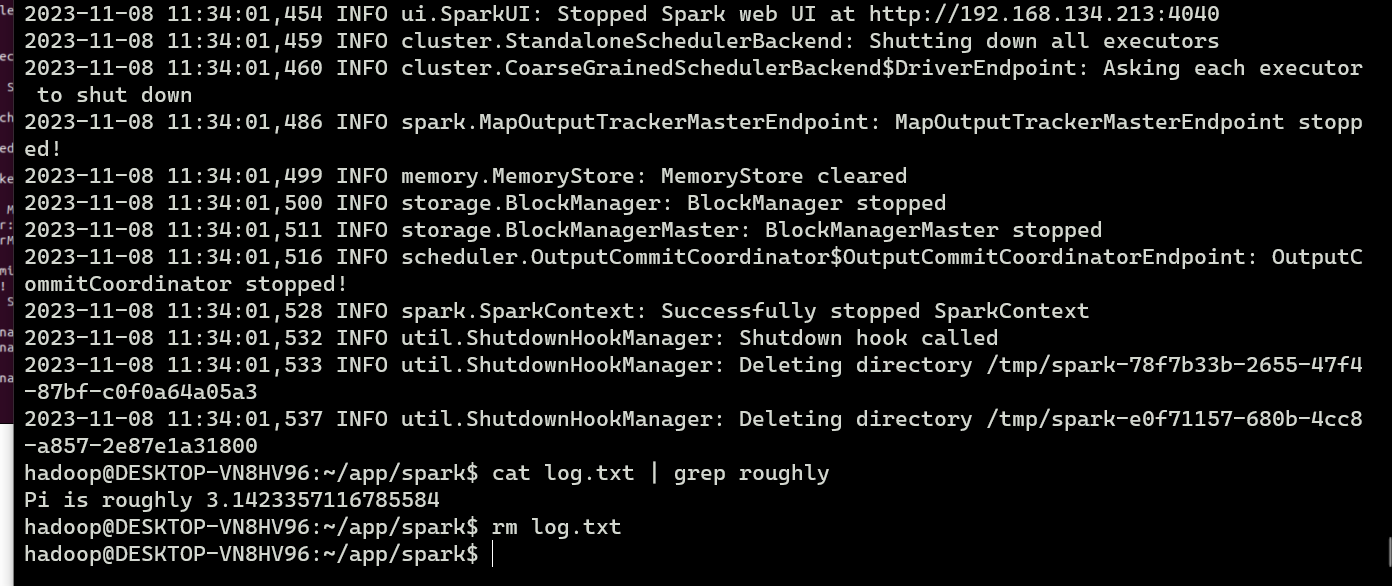


1. 查看woker是否启动

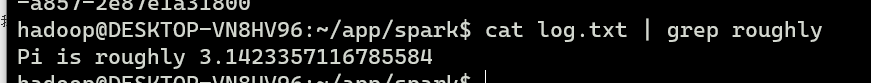


3.3验证Spark

1. 运行pi（π）的实例



（2）将结果输入到log.txt中输出



1. **总结及心得体会：**

学会了如何配置Hadoop的单节点伪分布式环境，包括创建用户、设置权限等。掌握了Hadoop基本命令，可以进行文件操作、查看日志、启动和停止服务等。学会安装和配置SSH，方便后续搭建集群，能够实现在多台机器之间无密码登录。也实现了Hadoop和Spark安装，后续进行了wordcount实验 ，有助于理解Hadoop的基本工作原理和MapReduce编程模型。

1. **对本实验过程及方法、手段的改进建议：**

可以在实验中引入更多的实际应用场景，例如基于Hadoop和Spark的大数据分析任务，并且流程之间的衔接性不强

**报告评分：**

**指导教师签字：**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名 杨径骁 学号：2021090921019 指导教师： 杨珊**

**实验地点：第二教学楼110 实验时间：2023.11.8**

1. **实验名称：**MapReduce与Spark内存计算的性能对比
2. **实验学时：**2学时
3. **实验目的：**
   1. 比较MapReduce和Spark框架的性能： 通过对同一WordCount分析程序在Hadoop MapReduce和Spark框架下的运行，对比两者的性能表现，包括运行时间、资源利用率等方面的指标。
   2. 实践MapReduce和Spark编程： 通过实际操作，学习和熟悉MapReduce和Spark编程模型，了解它们在大数据处理中的应用。
   3. 应用于中文文本的词频统计： 选作部分旨在拓展实验，尝试对中文文本进行词频统计，以了解MapReduce和Spark在处理非英文文本时的性能和适用性。
4. **实验原理**
   1. MapReduce框架： MapReduce是一种分布式计算框架，它将大规模数据处理任务分为Map和Reduce两个阶段。Map阶段负责数据切分和初步处理，Reduce阶段进行最终的结果合并。Hadoop是一个实现了MapReduce的开源框架。
   2. Spark框架： Spark是一个快速、通用、可扩展的大数据处理引擎，它支持丰富的数据处理任务。Spark中的基本抽象是弹性分布式数据集（RDD），它允许并行计算，并提供高层次的API（如Spark SQL、Spark Streaming等）。
   3. WordCount程序： WordCount是一个经典的大数据处理示例，它统计文本中每个单词出现的次数。该程序在MapReduce中通过Map和Reduce任务实现，而在Spark中可以使用RDD的转换和操作完成。

4. 性能比较： 通过准备相同大小的文本文件，在两个框架下运行相同的 WordCount程序，记录运行时间、资源利用情况等指标。比较两者的性 能，包括执行速度、扩展性、容错性等方面。

1. **实验内容**

使用Hadoop MapReduce、Spark框架分别运行wordcount分析程序，来对MapReduce和Spark的性能进行对比。

* 1. 准备600-700M左右的word.txt文件
  2. 依照步骤完成MapReduce代码和运行
  3. 依照步骤完成基于Spark的代码和运行
  4. 比较Hadoop MapReduce、Spark框架的运行效率
  5. 选作：尝试使用中文文本进行词频统计，并比较性能

1. **实验设备及环境**

•ubuntu-18.04及以上环境

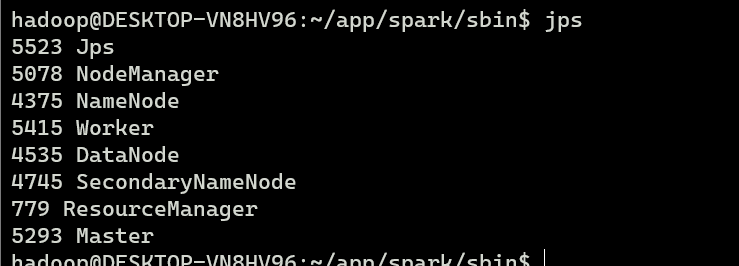
• jdk-8u261-linux-x64.tar.gz

• Hadoop 3.1.4

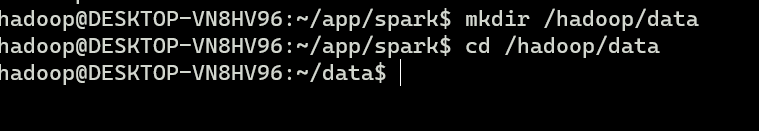
• spark-3.0.1-bin-hadoop3.2.tgz

1. **实验步骤**
2. **实验前准备**

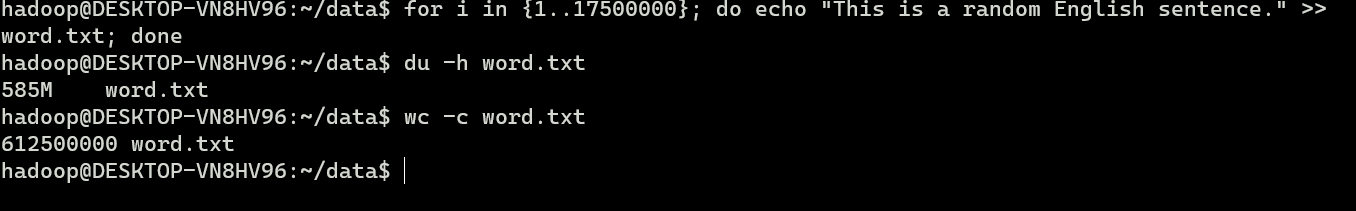
**启动ssh、启动柜Hadoop、启动spark**



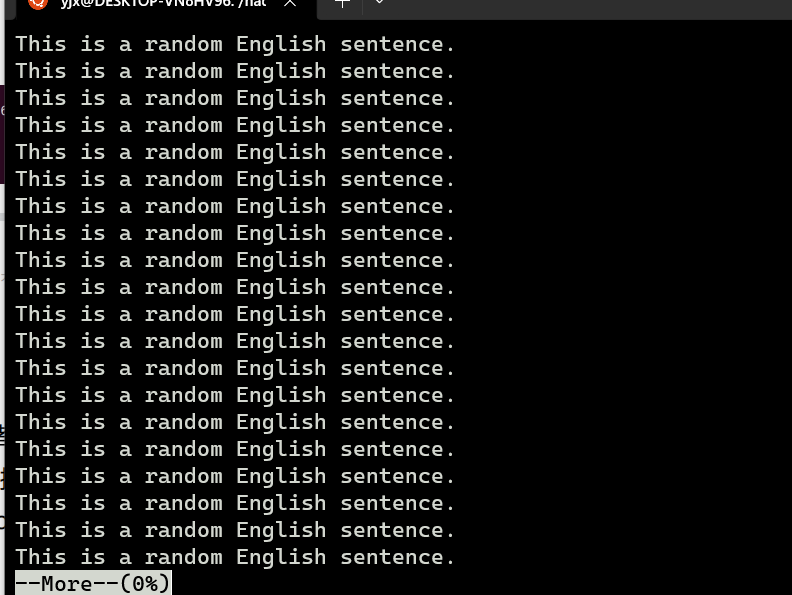
1. **将本次实验的数据文件上传到HDFS文件系统**
   1. 可以建立一个/hadoop/data目录。



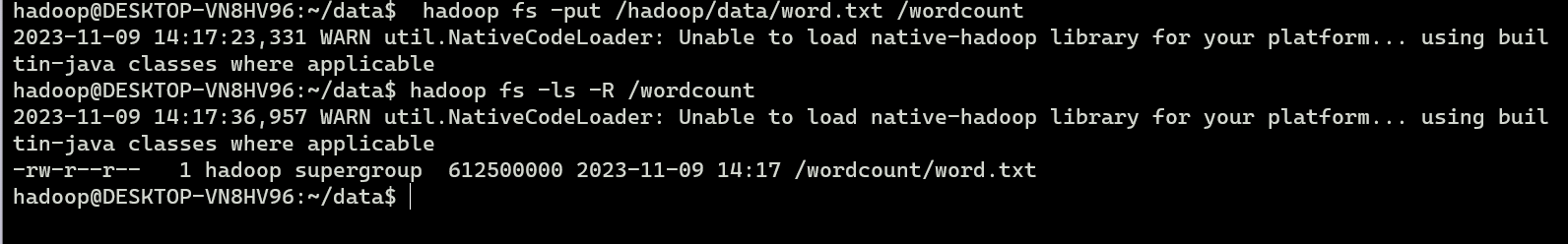
* 1. 将word.txt上传至该目录下。并查看该目录下将word.txt上传至该目录下。并查看该目录下是否有了word.txt文件是否有了word.txt文件



* 1. 查看字符集的内容



1. **将数据文件上传到HDFS文件系统**

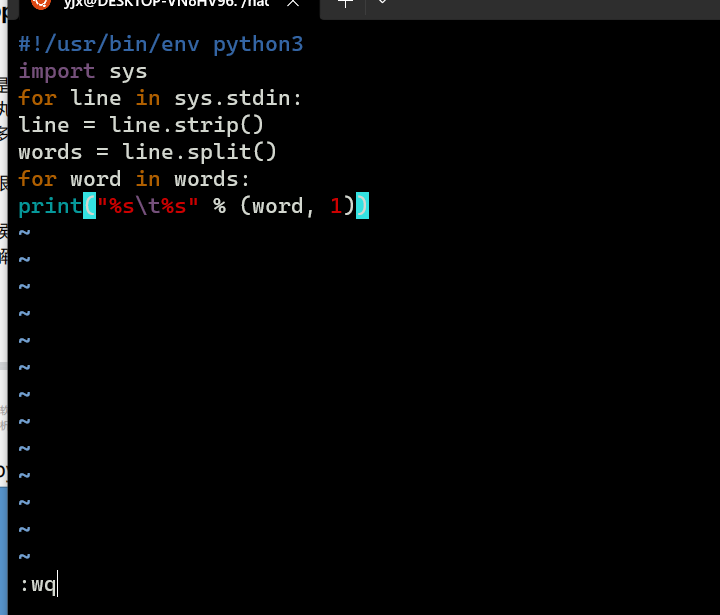
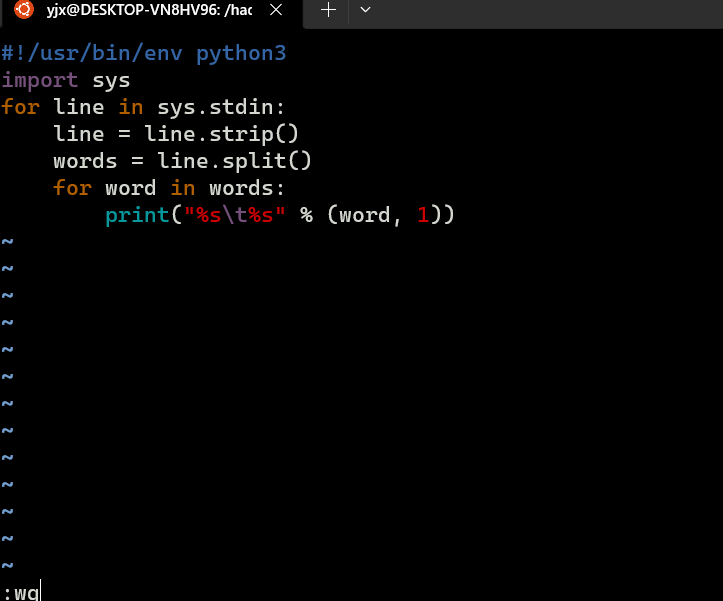


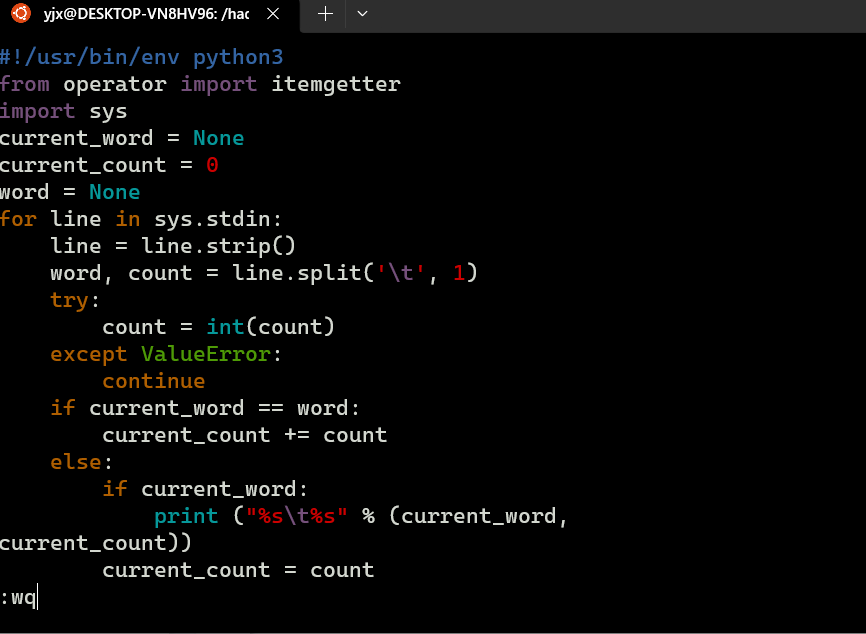
(注：警告可忽略)

1. **MapReduce实现WordCount实例（Python）**
   1. 创建/hadoop/data/mapreduce，并进入到/hadoop/data/mapreduce目录下

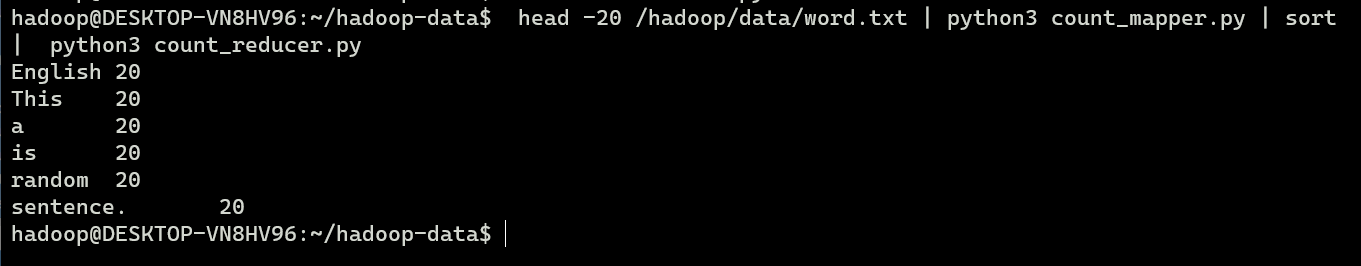


* 1. 在这个目录下首先编写MapReduce WordCount 代码

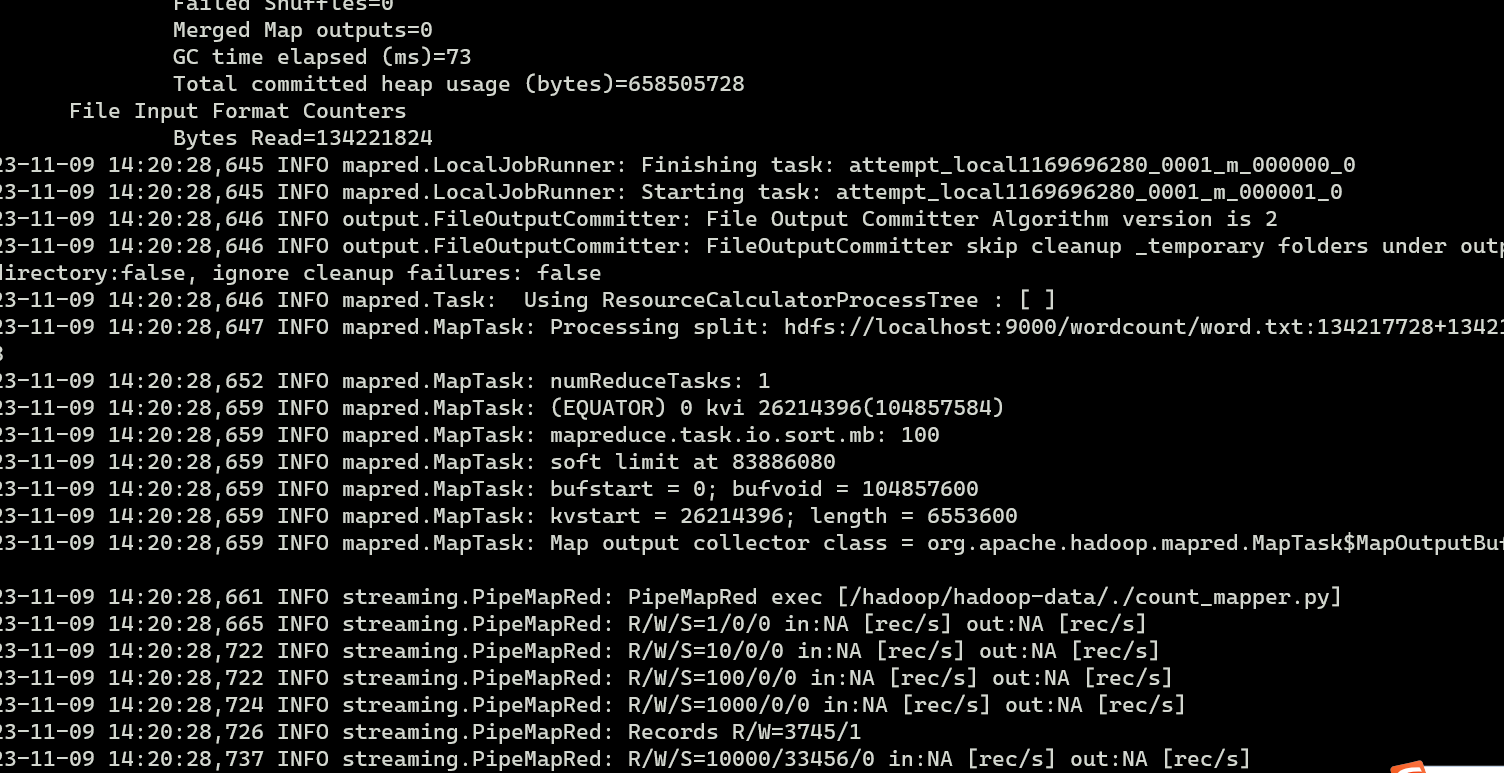


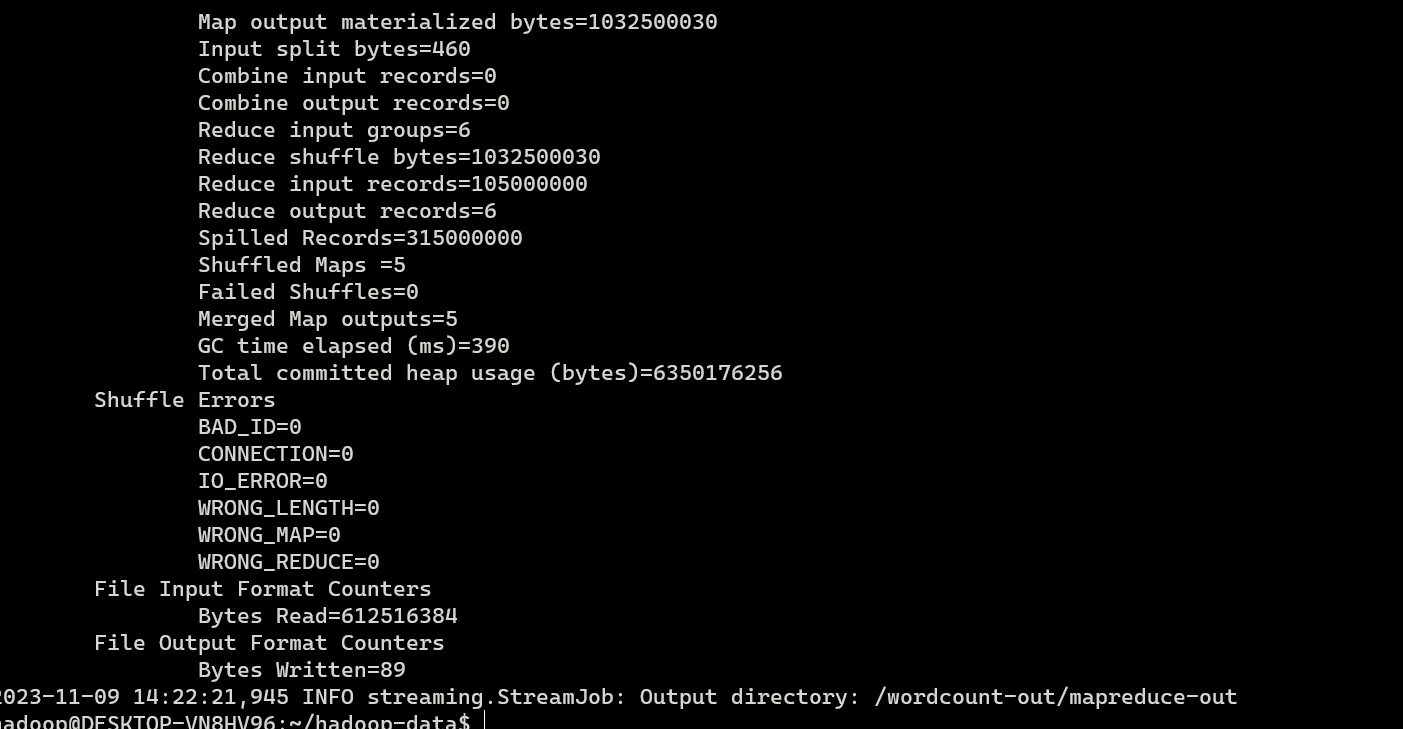


* 1. 在本地测试一下map和reduce

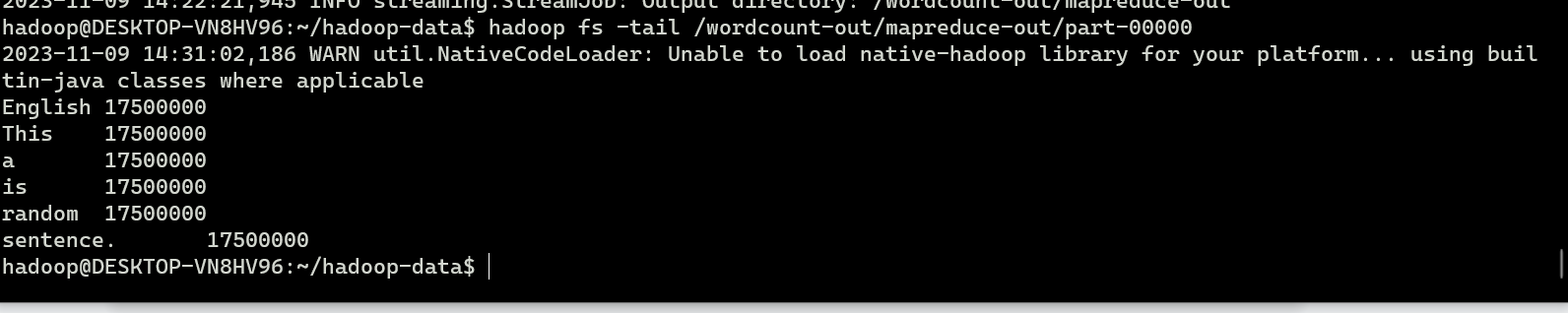


* 1. 运行该实例

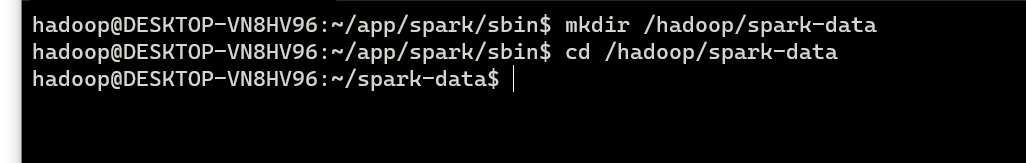




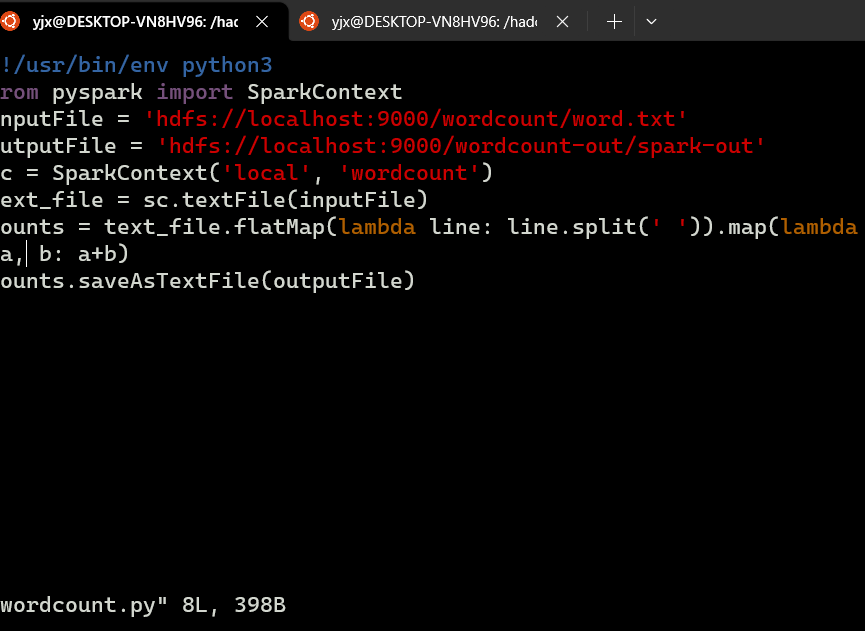
* 1. 运行该实例



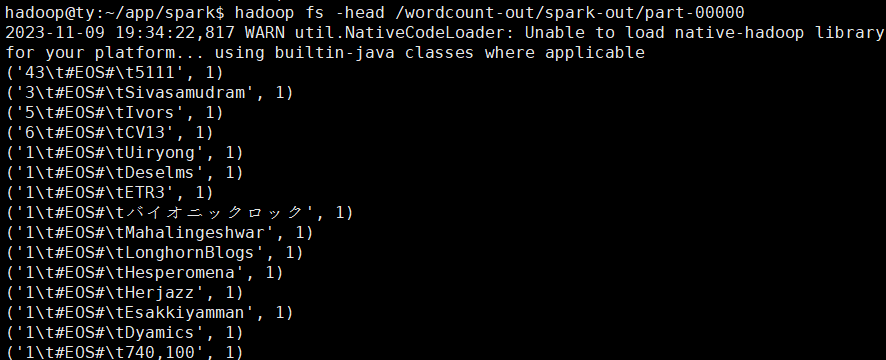
1. **Spark实现WordCount实例**
   1. 创建/hadoop/data/spark，并进入到/hadoop/data/spark目录下



* 1. 编写Spark WordCount 代码



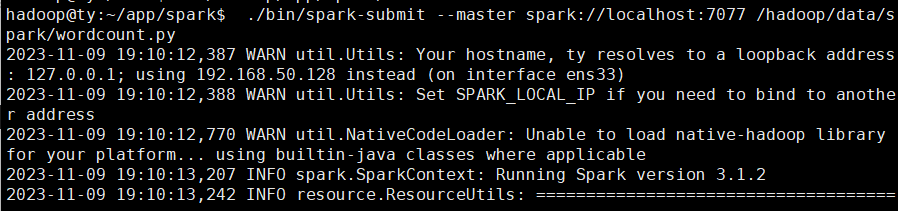
* 1. 运行该实例查看运行结果



1. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**

**1.Spark计算时间**

1.1Spark启动时间



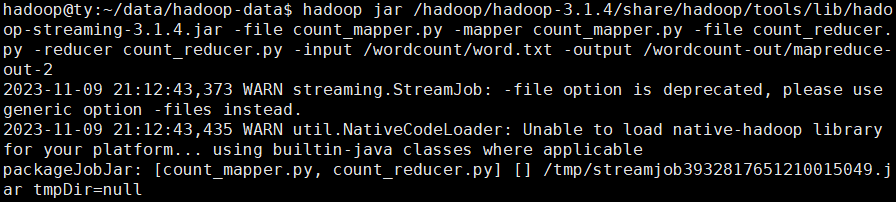
1.2 Spark结束时间



1.3 Spark总耗时：20m34s=1234s

**2. Hadoop计算时间**

2.1 Hadoop启动时间



2.2 Hadoop结束时间



2.3 Hadoop总耗时：7m55s=475s

**3.对比**

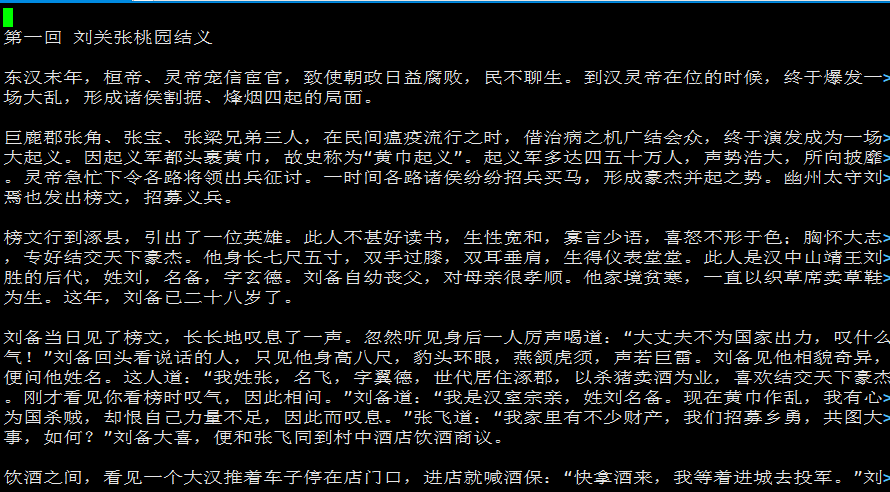
得到Hadoop总耗时475s < Spark总耗时 1234s，故Hadoop计算速度更快。

**补充：加分内容：Word.txt使用中文文本，在hadoop和Spark进行计算之前进行中文分词再做计算**

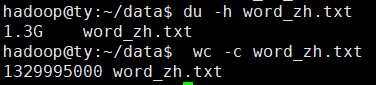
1. **word.txt使用中文文本并进行中文分词：**

1.1文本文件信息

文本文件来自《三国演义》:https://sanguo.5000yan.com/xiazai/sanguoyanyi.zip

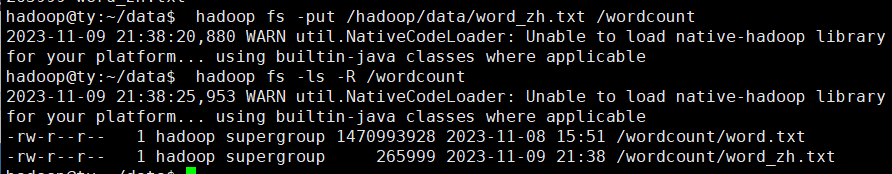


查看数据集大小和数据集字符数：



1. **上传至HDFS文件系统**

2.1上传至HDFS文件系统



2.2MapReduce实现wordcount实例

（1）编写Map阶段代码：count\_mapper\_zh.py，使用jieba分词器进行中文分词

|  |
| --- |
| #!/usr/bin/env python3  import sys  import jieba  for line in sys.stdin:  line = line.strip()  words = jieba.cut(line, cut\_all=False) # 进行中文分词  for word in words:  print("%s\t%s" % (word, 1)) |

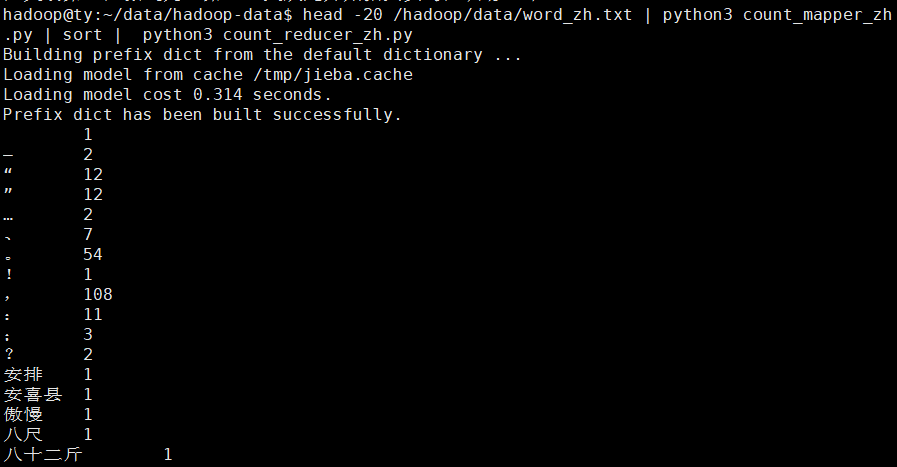
代码3 count\_mapper\_zh.py代码

（2）编写Reducer阶段代码：count\_reducer\_zh.py

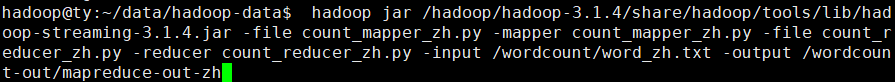
|  |
| --- |
| #!/usr/bin/env python3  from operator import itemgetter  import sys  current\_word = None  current\_count = 0  word = None  for line in sys.stdin:  line = line.strip()  if '\t' not in line:  continue  word, count = line.split('\t', 1)  try:  count = int(count)  except ValueError:  continue  if current\_word == word:  current\_count += count  else:  if current\_word:  print ("%s\t%s" % (current\_word, current\_count))  current\_count = count  current\_word = word  if current\_word == word:  print ("%s\t%s" % (current\_word, current\_count)) |

代码4 count\_reducer\_zh.py代码

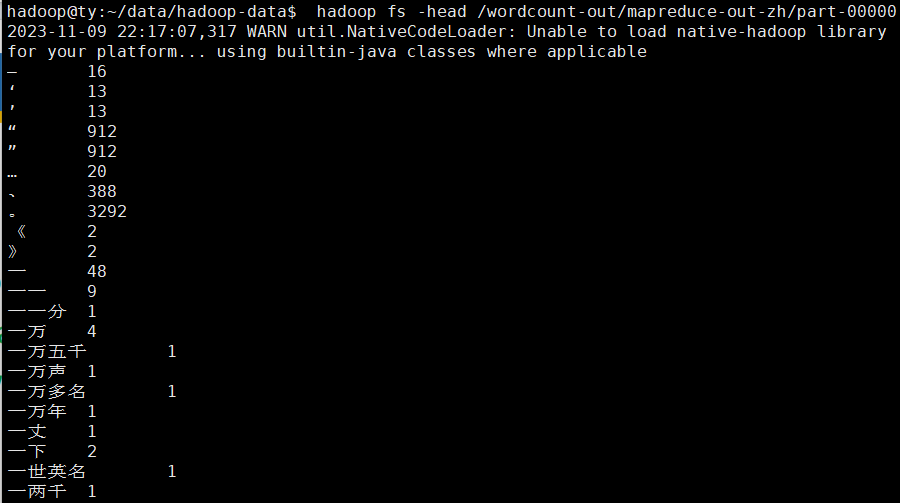
2.3本地测试Map和Reduce



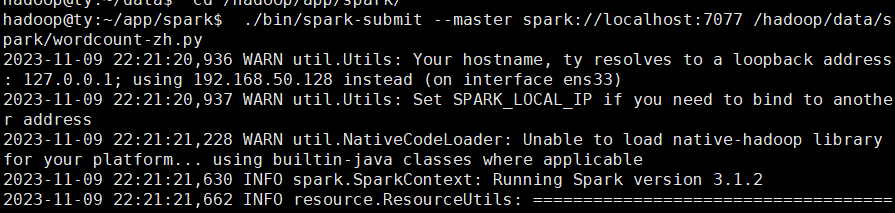
（1）运行实例



（2）查看结果



1. **Spark实现wordcount实例**

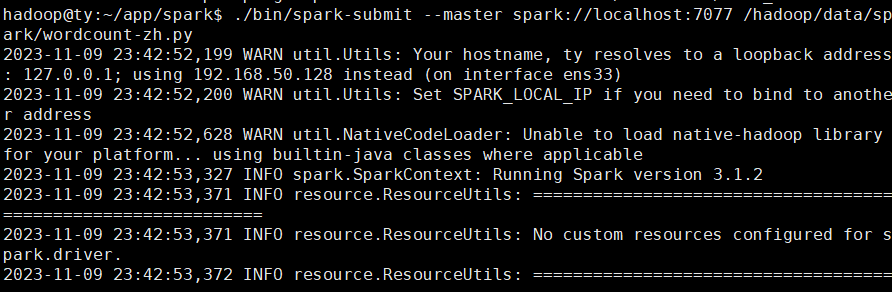


3.1编写Spark wordcount代码：wordcount-zh.py

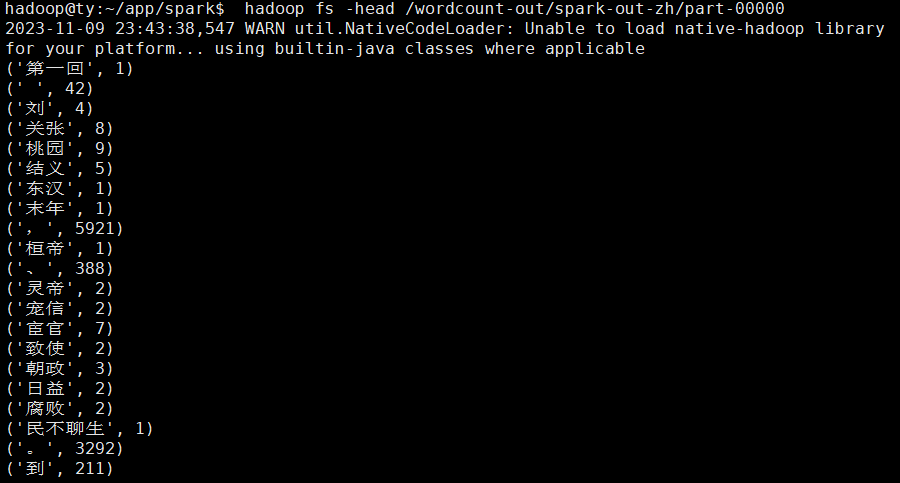
|  |
| --- |
| #!/usr/bin/env python3  from pyspark import SparkContext  import jieba  inputFile = 'hdfs://localhost:9000/wordcount/word\_zh.txt'  outputFile = 'hdfs://localhost:9000/wordcount-out/spark-out-zh'  sc = SparkContext('local', 'wordcount')  text\_file = sc.textFile(inputFile)  # 使用中文分词对每行进行分词操作  word\_counts = text\_file.flatMap(lambda line: jieba.lcut(line)).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)  word\_counts.saveAsTextFile(outputFile) |

代码5 wordcount-zh.py代码

（1）运行实例



（2）查看结果



1. **总结及心得体会：**

通过实验，深入了解了MapReduce和Spark这两个大数据处理框架。了解了它们的基本原理、编程模型以及应用场景，对分布式计算有了更加全面的认识。在实验中，通过编写WordCount程序，深入理解了MapReduce和Spark的编程模型。通过实际的代码编写和运行，对大数据处理中的分布式计算任务有了更深刻的理解。

选作部分的尝试对中文文本进行词频统计，是对框架在处理非英文文本上的一次拓展。这种尝试不仅拓宽了应用范围，还加深了对框架适用性的理解。

1. **对本实验过程及方法、手段的改进建议：**

在性能比较部分，可以加入更深入的性能分析，例如资源利用率、执行时间的详细对比等，以便更全面地了解MapReduce和Spark在不同场景下的表现

**报告评分：**

**指导教师签字：**