电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2021090921020

姓 名 谭宇乔

（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：谭宇乔 学号：2021090921020 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。

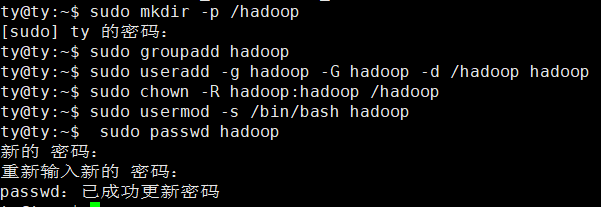
最后安装Spark。

1. **实验设备及环境**

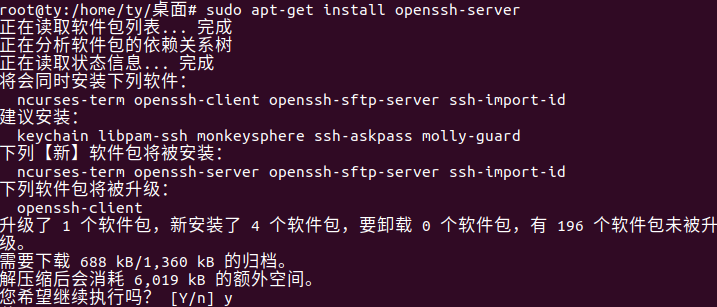
VMware Workstation Pro，Ubuntu-20.04.6

1. **实验步骤**
2. **Hadoop安装**

1.创建用户



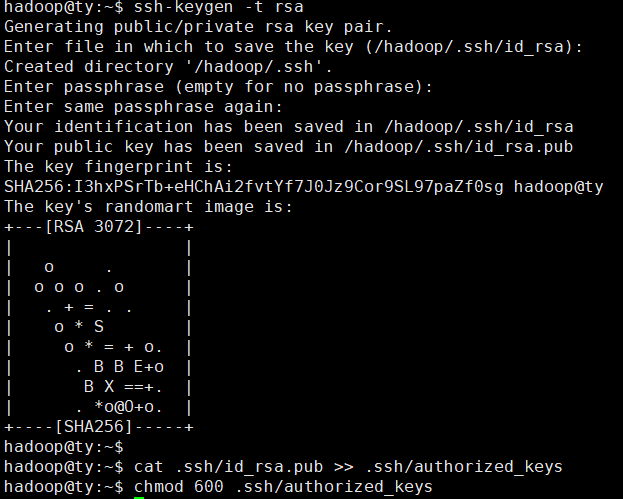
2.安装SSH



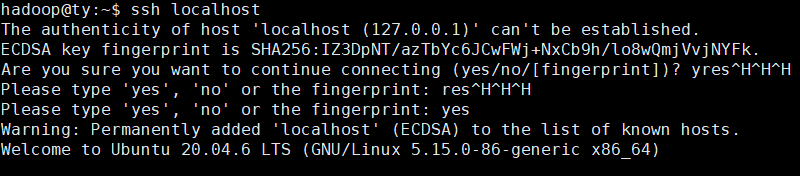
3.验证



4.配置免密码登录



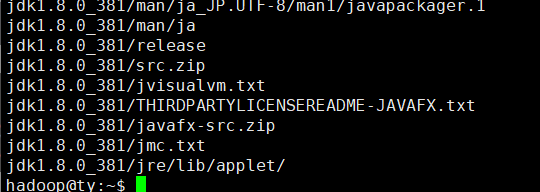
5.免密码登录

6.将共享文件拷贝到Linux环境



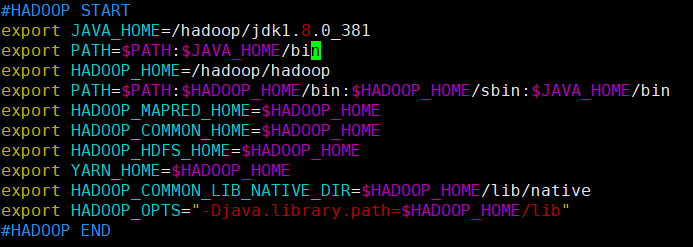


7.解压jdk和hadoop

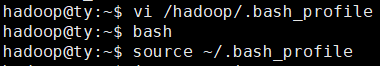


8.配置并激活环境变量

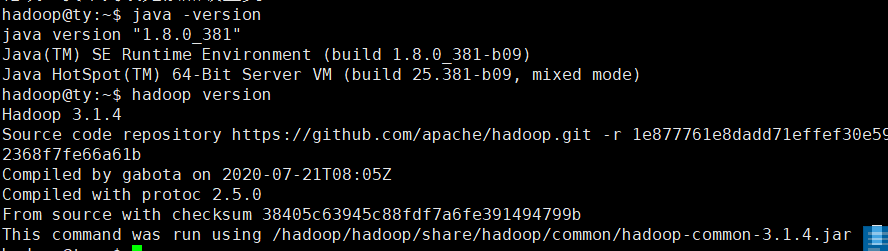
原文档中需要添加export PATH=$PATH:$JAVA\_HOME/bin，否则会报错找不到主类。



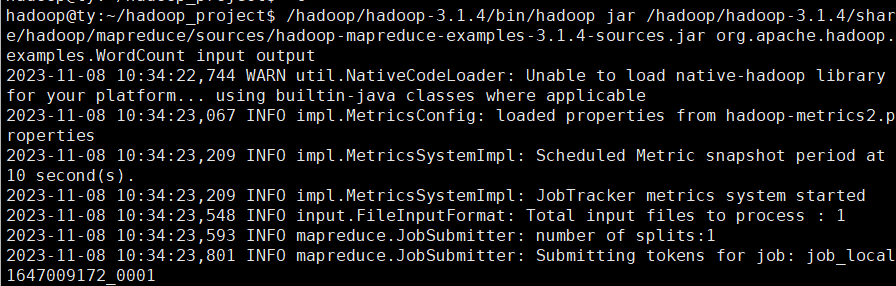
配置完成后激活新的环境



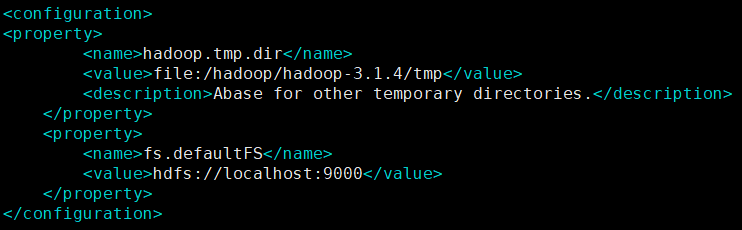
9.查看Java和Hadoop版本



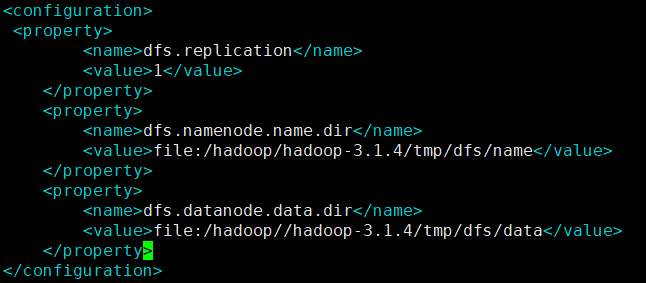
10.词频统计



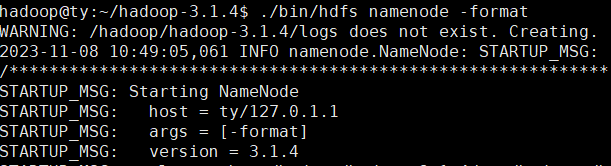
1. **Spark安装**
   1. **Hadoop伪分布式设置**
2. 修改core-site.xml



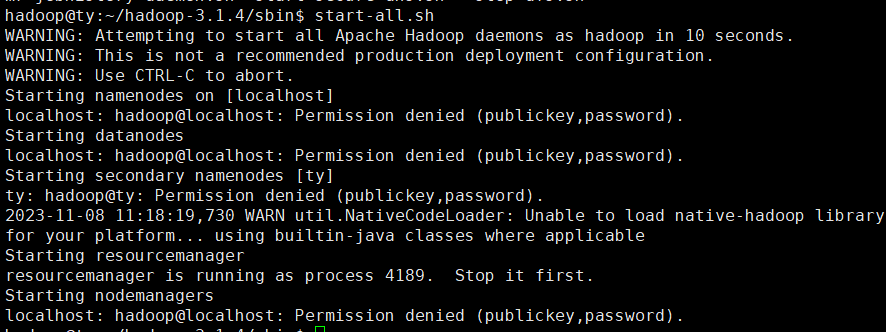
1. 修改hdfs-site.xml



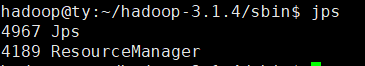
1. ./bin/hdfs namenode -format  实现namenode的格式化



1. 启动hadoop（namenode节点）

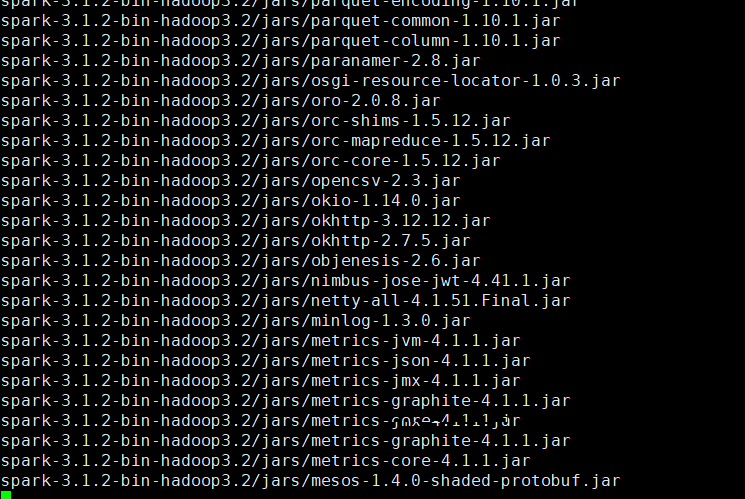


1. 检查是否运行成功

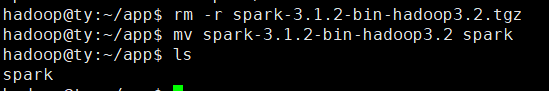


* 1. **安装与配置 Spark**

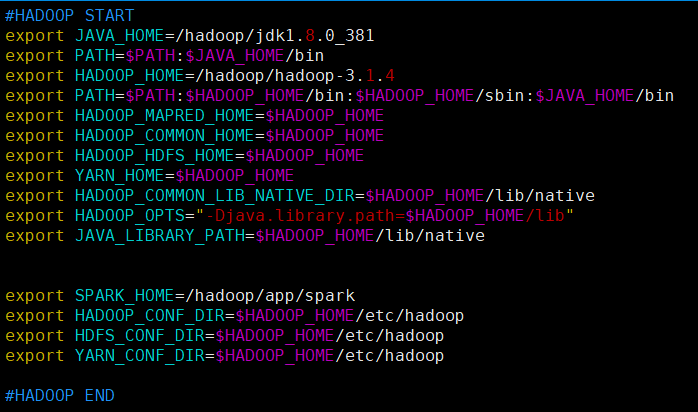
1. 解压并安装Spark



删除压缩包并重命名文件夹



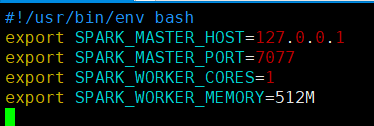
1. 配置Hadoop环境变量



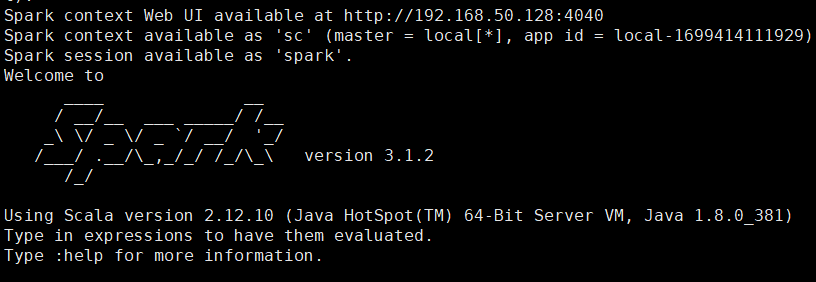
执行以下命令使得环境变量生效：



1. 修改配置文件：spark-env.sh

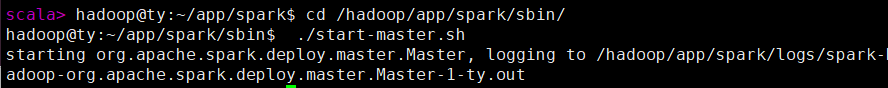


1. Spark启动
   1. 进入Spark Shell

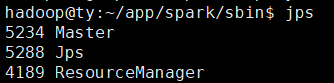


* 1. 启动Spark

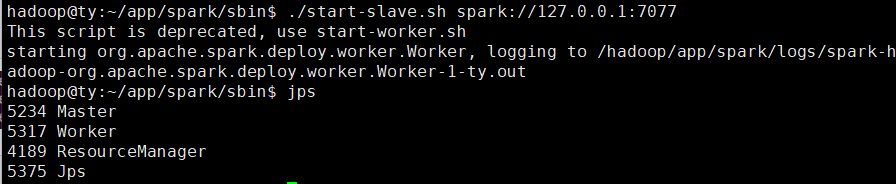
首先启动Master



查看Master进程是否启动

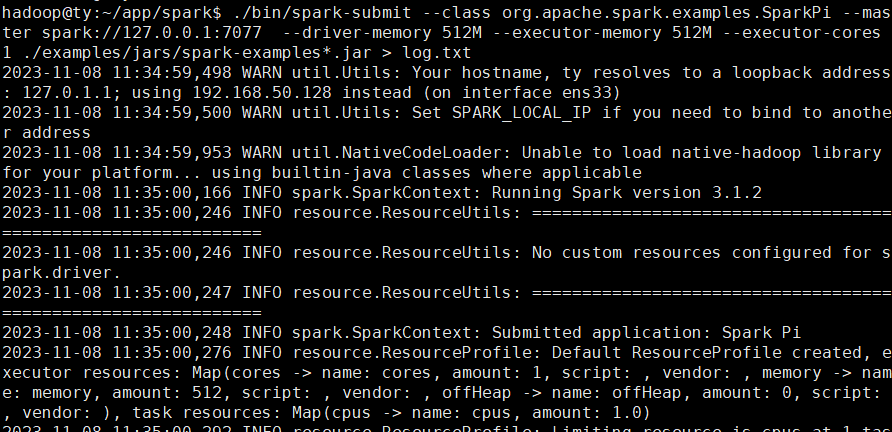


启动Slave



1. 验证Spark

运行Pi(π)的实例





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）**

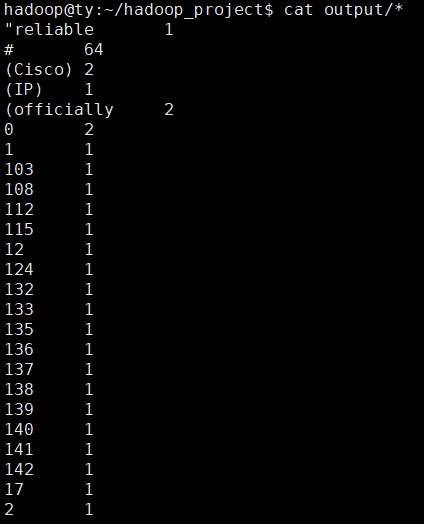
cp /etc/protocols input

/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 input output

cat output/\*

代码1 对/ect/protocols 进行词频统计代码

运行结果：



|  |
| --- |
| cp /var/log/dpkg.log input  /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 input output  cat output/\* |

代码2 对/var/log/dpkg.log进行词频统计代码

代码解释：

1. cp /pwd input：这个命令复制了/pwd文件到input目录中。在Hadoop的WordCount示例中，input目录通常包含要进行单词计数的文本文件。
2. /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 input output：这个命令运行了Hadoop MapReduce作业，使用WordCount示例，从input目录中的文本文件中计算单词频率，并将结果存储在output目录中。
3. cat output/\*：一旦WordCount作业完成，使用cat命令来查看output目录中的所有文件的内容。这将显示WordCount示例的结果，即每个单词和其出现的次数。
4. **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。

**十、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：谭宇乔 学号：2021090921020 指导教师：杨珊**

**实验地点：二教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。
5. **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**

VMware Workstation Pro，Ubuntu-20.04.6

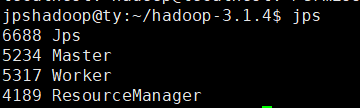
1. **实验步骤**
2. **实验前准备**

启动ssh,口令输入：hadoop



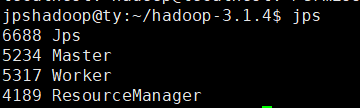
1. **启动Hadoop**

启动命令为：$ start-all.sh ，检查是否运行成功。



1. **启动Spark**

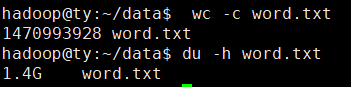
查看Master和Worker进程是否启动



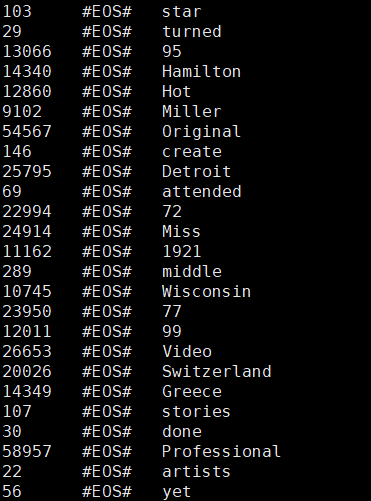
1. **将实验数据文件上传至HDFS**

查看数据集大小和数据集字符串数

数据来自：<https://nlp.cs.nyu.edu/wikipedia-data/>

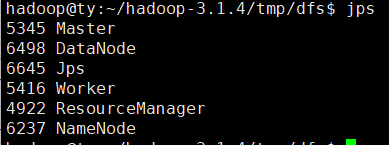


查看字符集内容

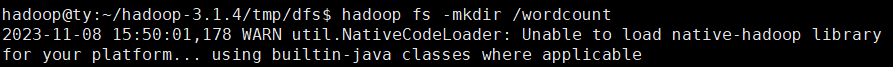


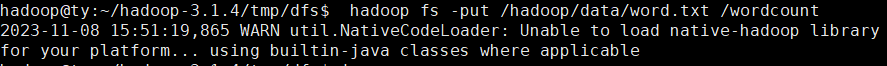
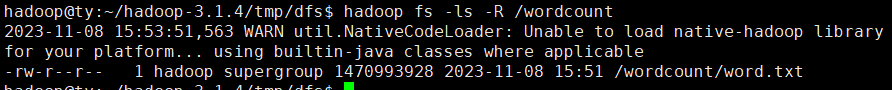
**问题**：文档方式无法启动NameNode，需要执行：hdfs --daemon start namenode

**问题**：文档方式无法启动DataNode，需要执行：hadoop-daemon.sh start datanode



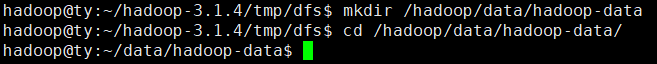
将文件上传到HDFS/wordcount



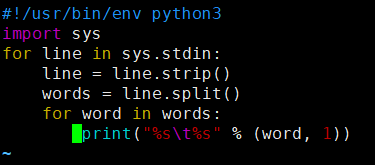
1. MapReduce实现WordCount实例

创建/hadoop/data/mapreduce，并进入到/hadoop/data/mapreduce目录下

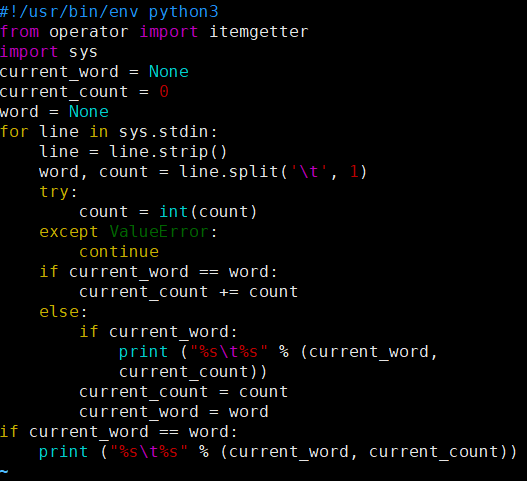
、

编写MapReduce WordCount 代码：

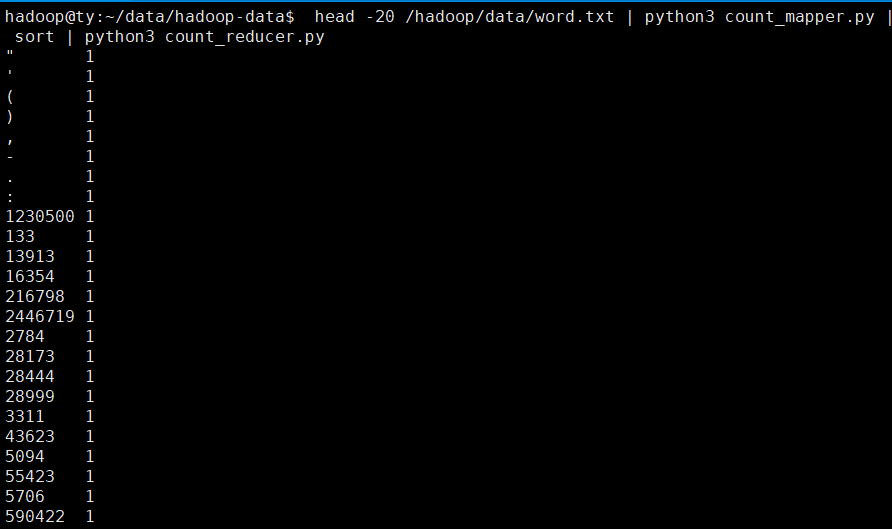
* 1. 编写map阶段代码：count\_mapper.py



* 1. 编写reduce阶段代码：count\_reducer.py



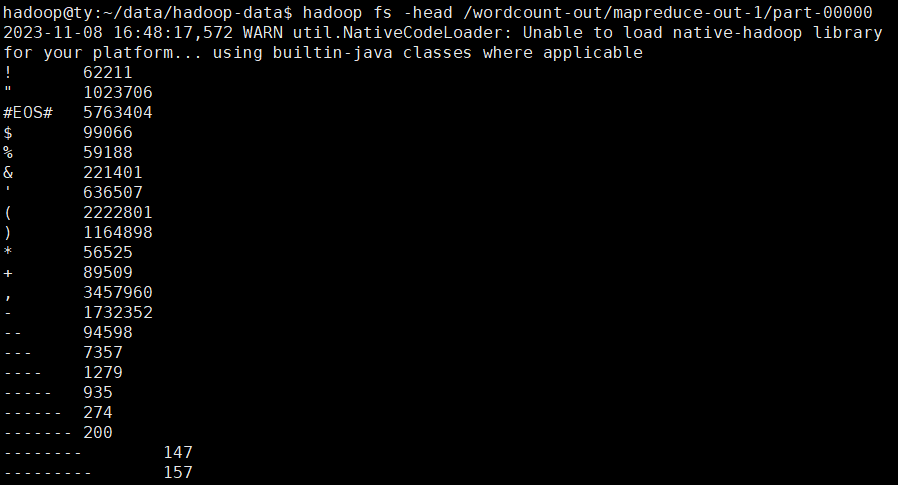
本地测试Map和Reducer



运行实例：hadoop jar /hadoop/hadoop-3.1.4/share/hadoop/tools/lib/hadoop-streaming-3.1.4.jar -file count\_mapper.py -mapper count\_mapper.py -file count\_reducer.py -reducer count\_reducer.py -input /wordcount/word.txt -output /wordcount-out/mapreduce-out-1

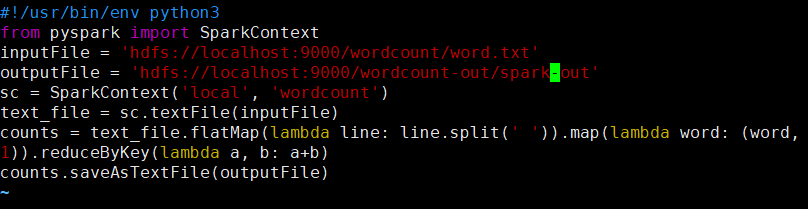


查看结果

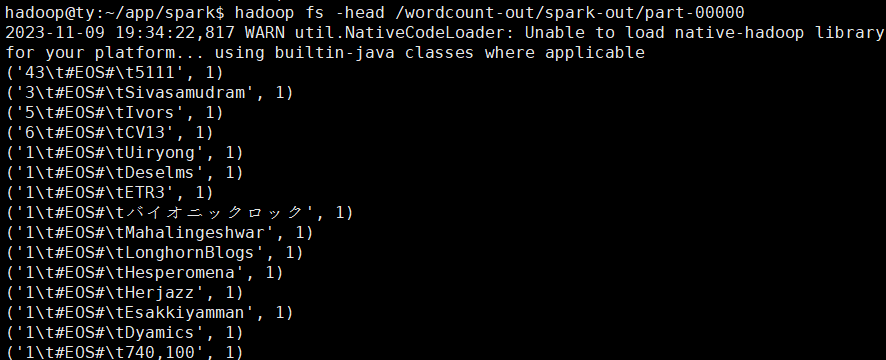


1. Spark实现WordCount实例

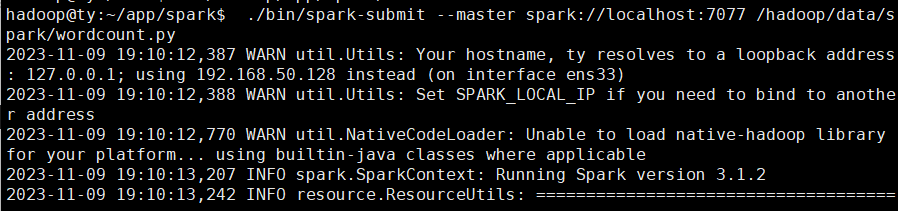
编写wordcount.py



运行该实例，并查看结果



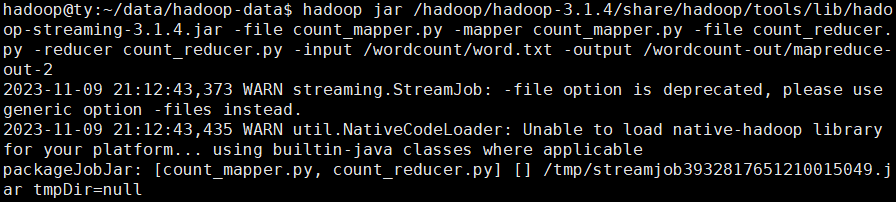
1. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**
2. **对比分析Hadoop和Spark的计算速度**
   1. **Spark计算时间**
      1. Spark启动时间



* + 1. Spark结束时间



* + 1. Spark总耗时：20m34s=1234s
  1. **Hadoop计算时间**
     1. Hadoop启动时间



* + 1. Hadoop结束时间

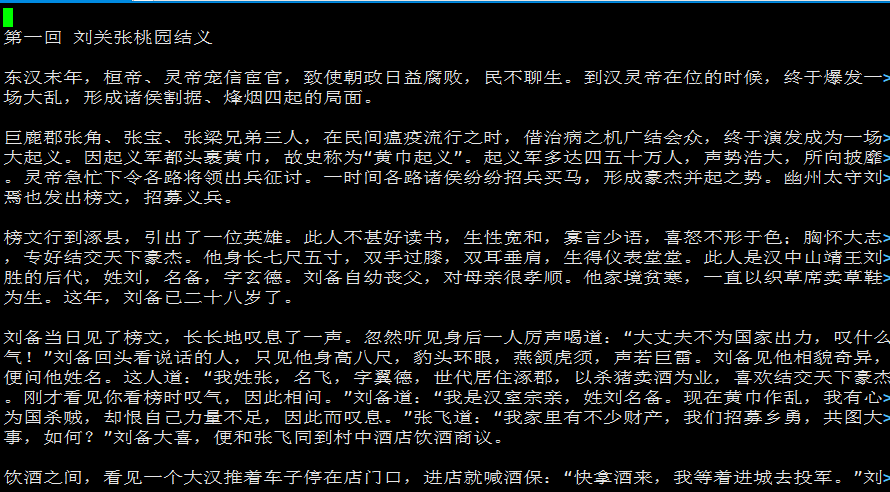


* + 1. Hadoop总耗时：7m55s=475s
  1. **对比**

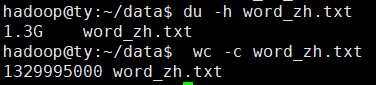
得到Hadoop总耗时475s < Spark总耗时 1234s，故Hadoop计算速度更快。

1. **word.txt使用中文文本并进行中文分词：**
   1. **文本文件信息**

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

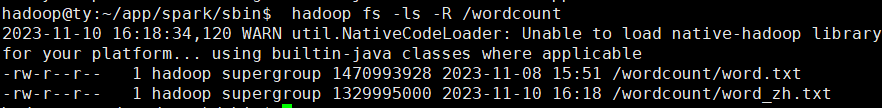


查看数据集大小和数据集字符数：



* 1. **上传至HDFS文件系统**

上传至HDFS文件系统



* 1. **MapReduce实现wordcount实例**

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)) |

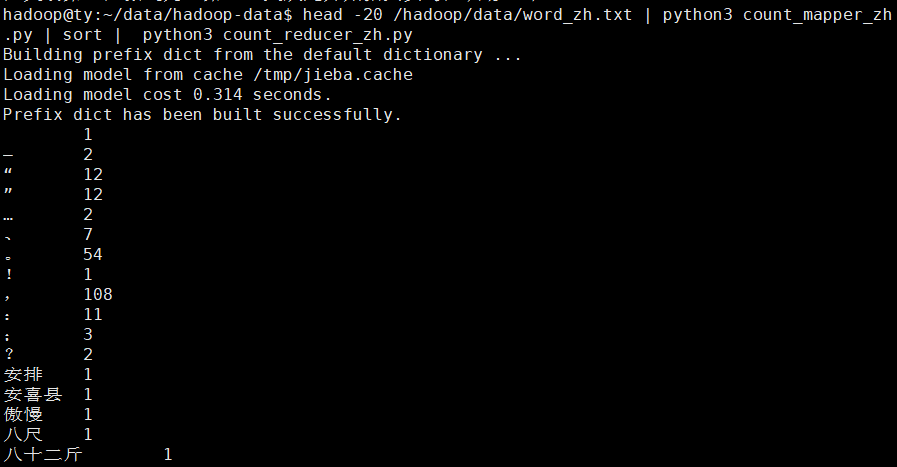
代码3 count\_mapper\_zh.py代码

编写Reducer阶段代码：count\_reducer\_zh.py

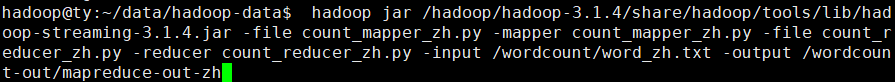
|  |
| --- |
| #!/usr/bin/env python3  from operator import itemgetter  import sys  current\_word = None  current\_count = 0  word = None  for line in sys.stdin:  line = line.strip()  if '\t' not in line:  continue  word, count = line.split('\t', 1)  try:  count = int(count)  except ValueError:  continue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

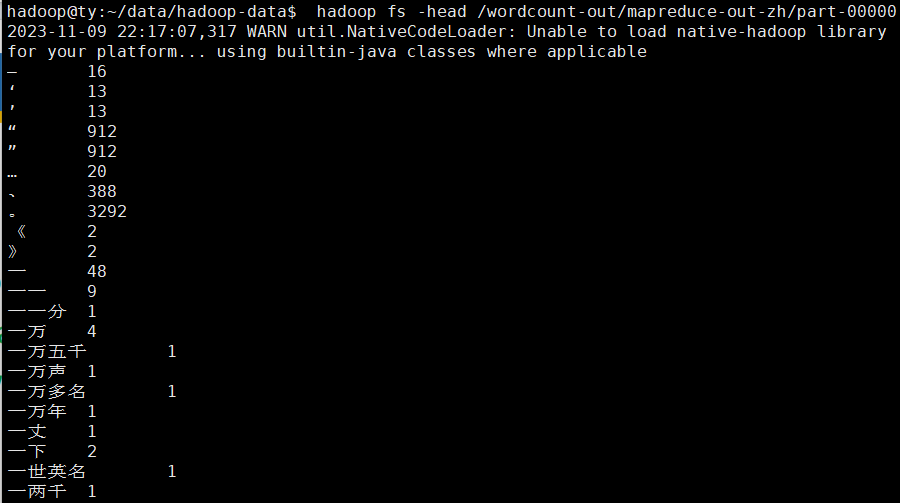
本地测试Map和Reduce



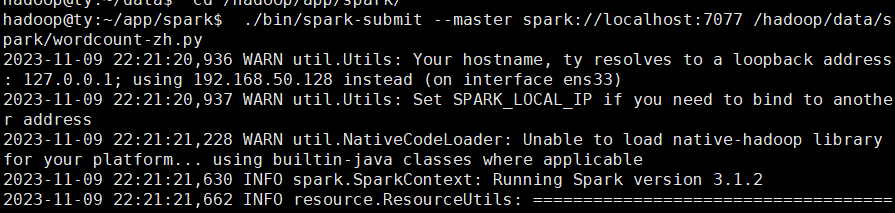
运行实例



查看结果



* 1. **Spark实现wordcount实例**

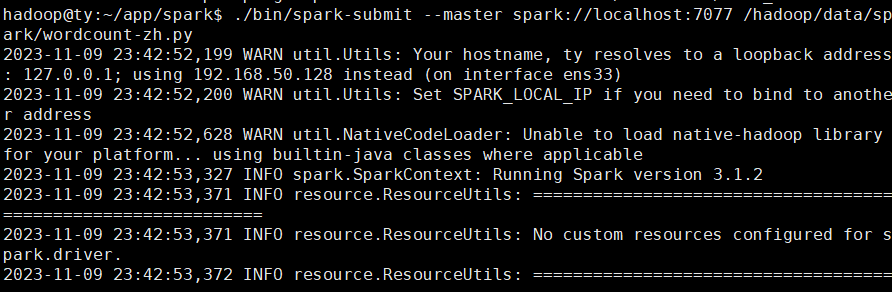


编写Spark wordcount代码：wordcount-zh.py

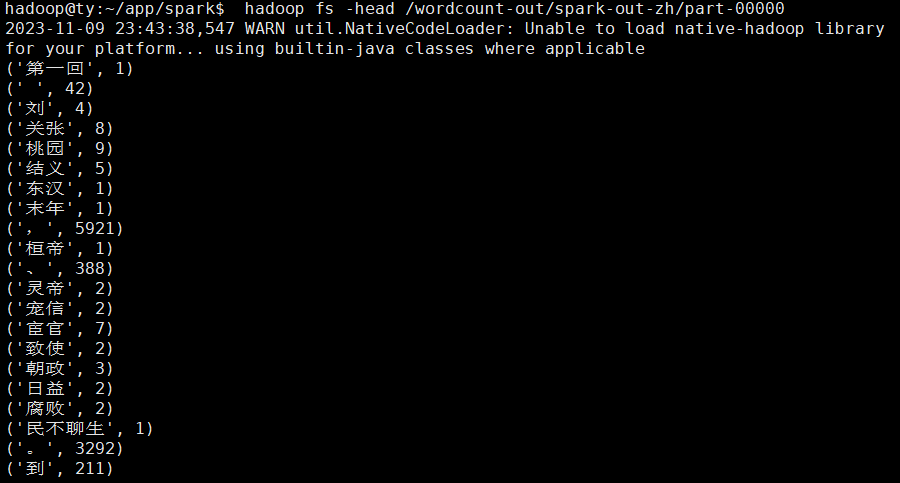
|  |
| --- |
| #!/usr/bin/env python3  from pyspark import SparkContext  import jieba  inputFile = 'hdfs://localhost:9000/wordcount/word\_zh.txt'  outputFile = 'hdfs://localhost:9000/wordcount-out/spark-out-zh'  sc = SparkContext('local', 'wordcount')  text\_file = sc.textFile(inputFile)  # 使用中文分词对每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

运行实例



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。

**十、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：**