**大数据分析技术与应用实验报告**

**实验二:MapReduce实例实现**

专 业： 电子信息

姓 名： **李建辉**

学 号： S202075100

指导老师： 李建强、朱娜斐

2021年 04 月 27 日

信息学部软件学院

目 录

[1 实验任务 1](#_Toc477628915)

[2 实验过程 1](#_Toc477628916)

[2.1 Hadoop、Eclipse编程环境搭建 1](#_Toc477628917)

[2.2 Eclipse运行Wordcount实例 1](#_Toc477628918)

[2.3基于Hadoop的数据去重实例实现 1](#_Toc477628919)

[3 实验总结 1](#_Toc477628920)

# 1 实验任务

1. 搭建Hadoop、Eclipse编程环境；
2. 运行实验指导上提供的Wordcount程序；
3. 在Eclipse上面查看HDFS文件目录；
4. 在Eclipse上面查看Wordcount程序运行结果；
5. 熟悉Hadoop、Eclipse编程流程及思想；

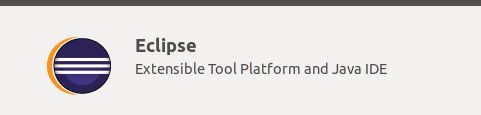
# 2 实验过程

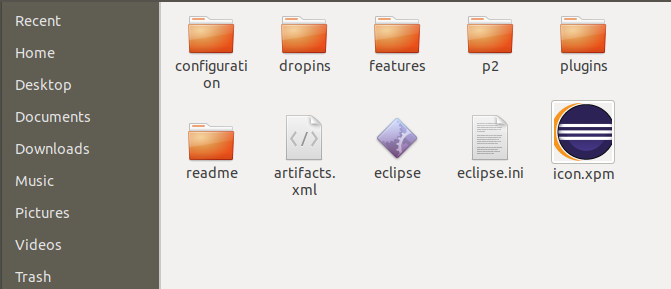
## 2.1 Hadoop、Eclipse编程环境搭建

环境：Ubuntu 18.04，jdk 1.8，Hadoop 2.10.1，eclipse

1.环境配置

在Ubuntu软件中心找到eclipse，并安装

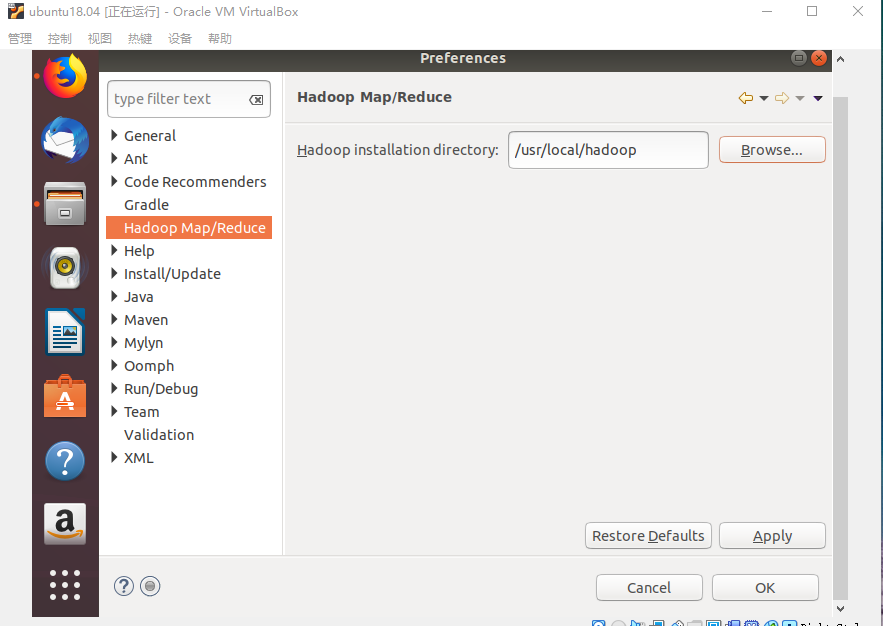


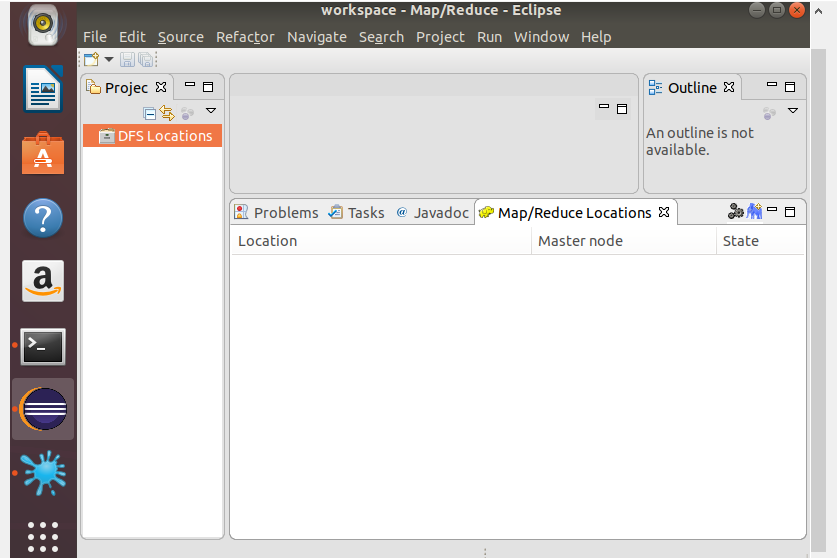
Eclipse安装路径/usr/lib/，并将对应的 将hadoop-eclipse-kepler-plugin-2.10.1.jar（我自己用ant编译成功hadoop-eclipse-kepler-plugin-2.10.1.jar）放到eclipse解压文件夹的plugins文件夹下，以便顺利启动。 

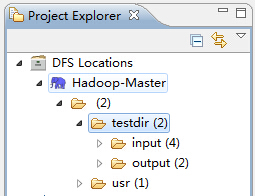
开启hadoop，并将本地input目录下文件上传到hdfs系统中：



配置环境：





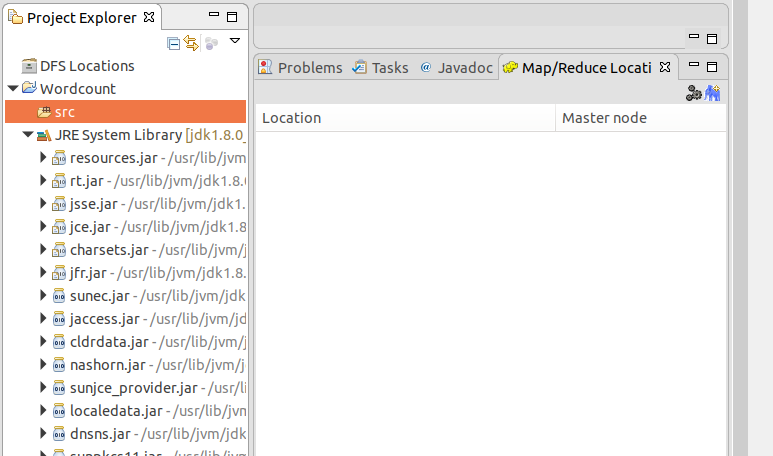


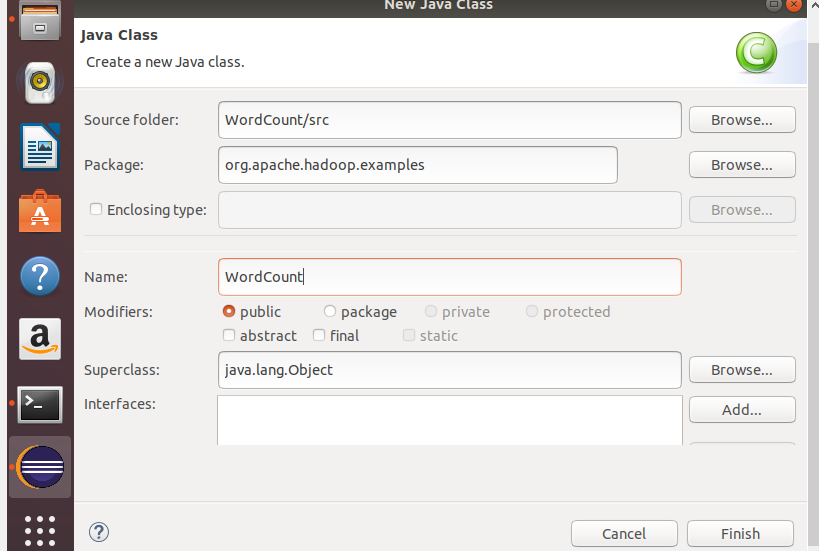
配置完成即可查看、操作文件，并且可以完成下面的编程执行工作。

## 2.2 Eclipse运行Wordcount实例

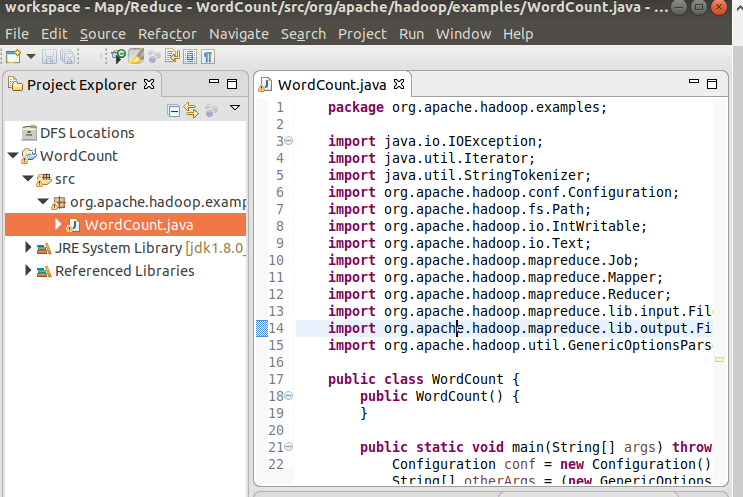
使用eclipse执行wordcount：

创建wordcount项目：

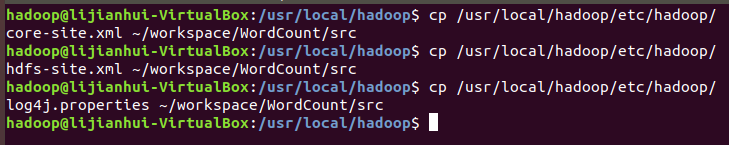




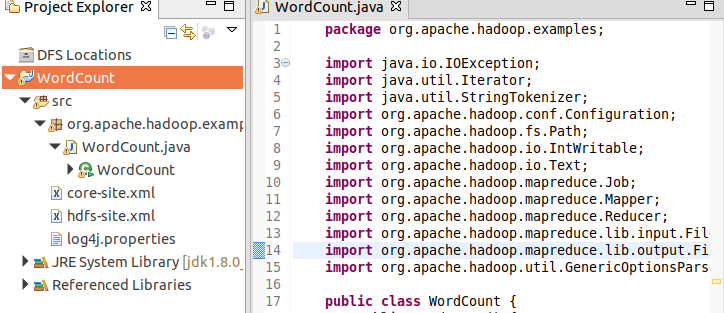
将wordcount代码复制到java文件中：



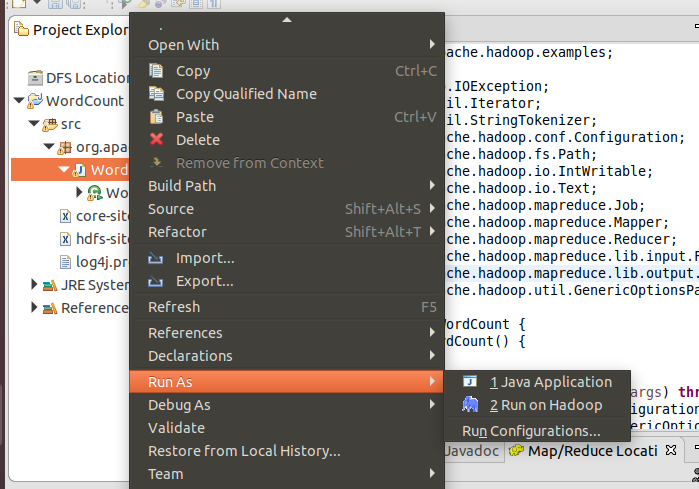
将hadoop配置文件复制到wordcount项目中：



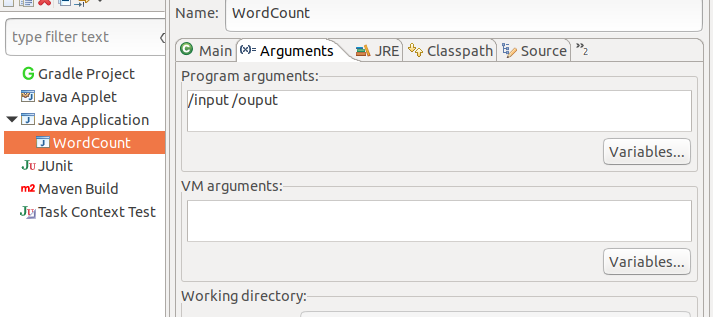
刷新后：



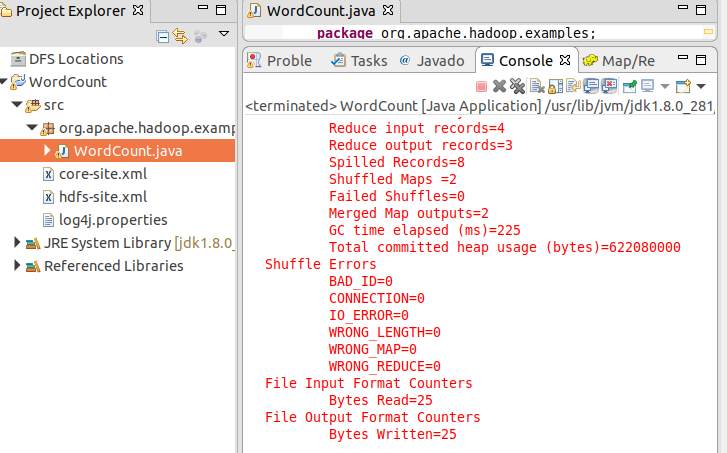
在Hadoop上运行

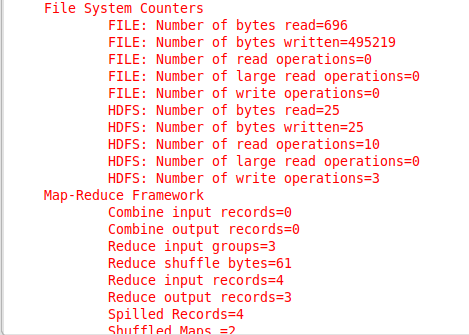


设置变量（要加斜杠，否则会报错）：

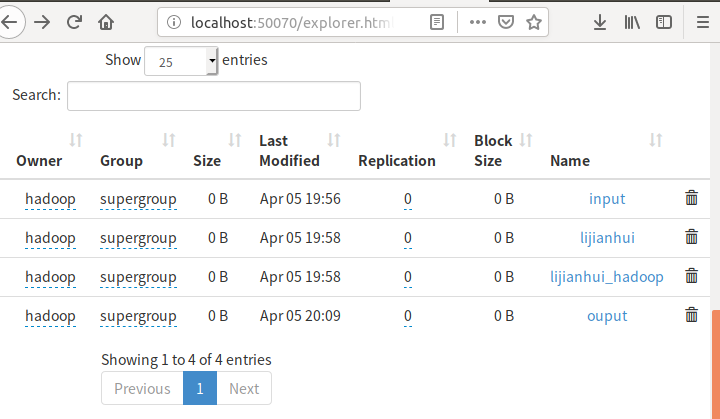


执行成功：

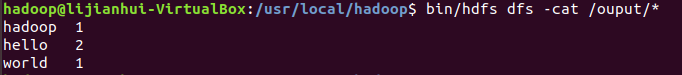




并且在web端可以看到出现了/output,成功：



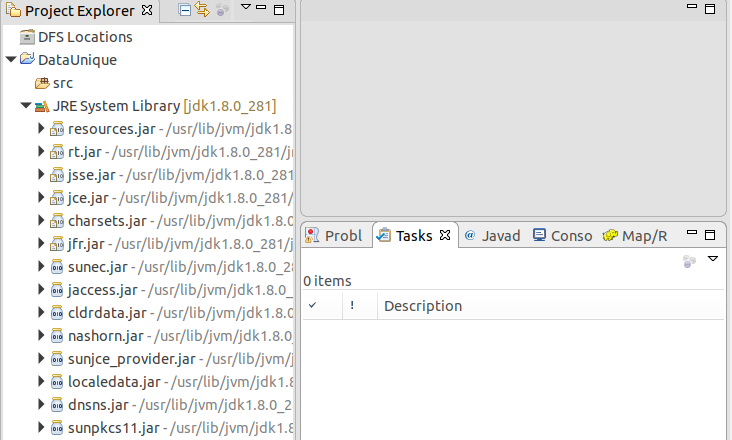
可以得到我们的词数，正确无误：



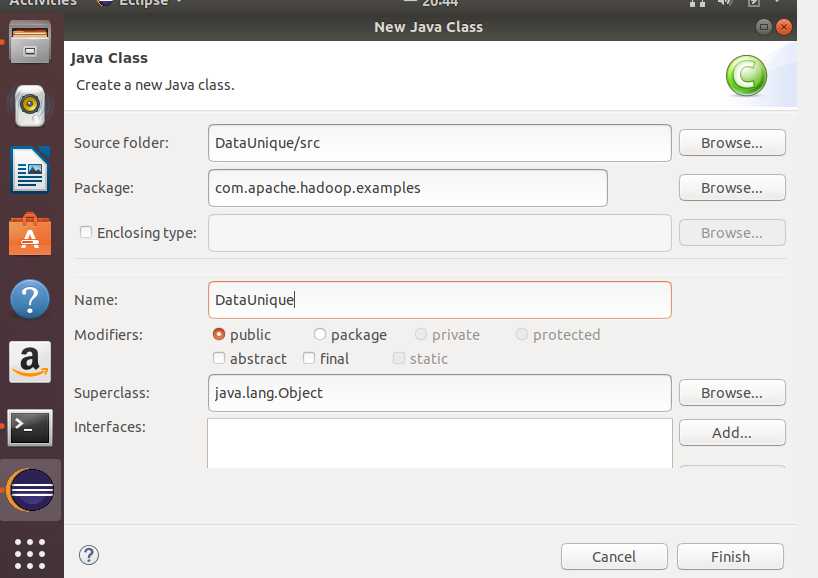
## 2.3基于Hadoop的数据去重实例实现

数据去重：

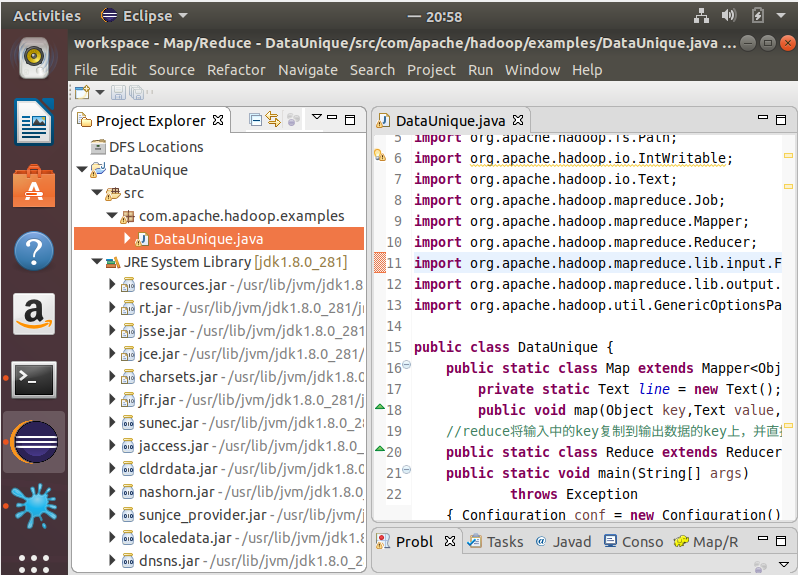
首先，建立项目：



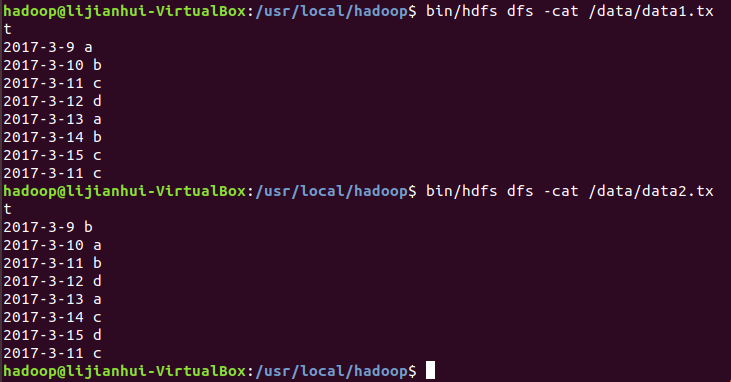
创建类：



写入程序：



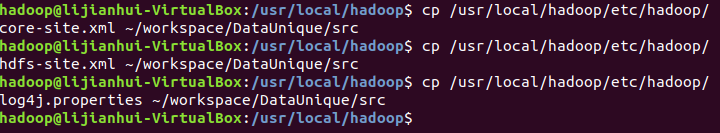
按照要求写入数据（此处查看hdfs中的data文件）：



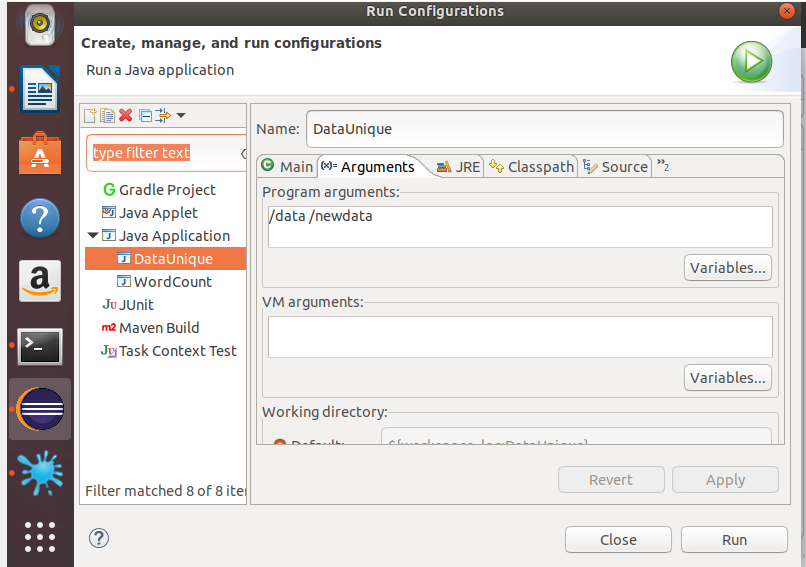
上传文件：



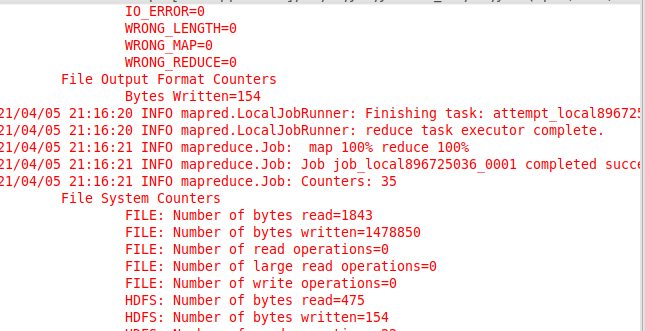
配置项目：

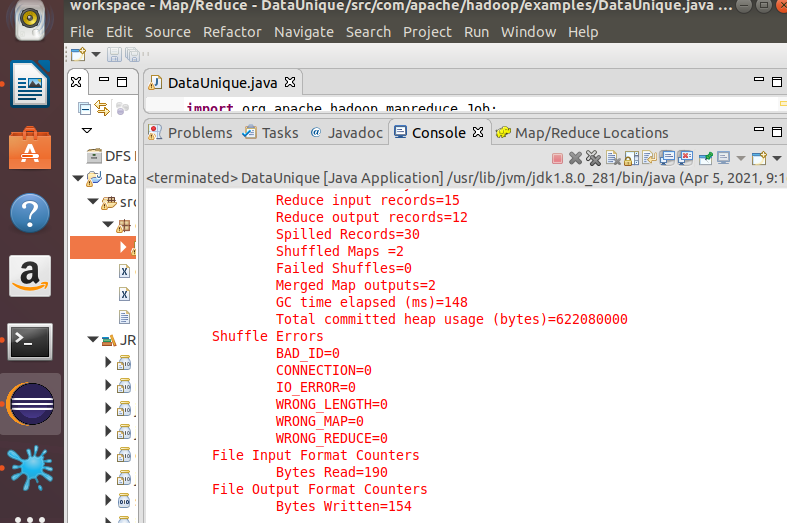


配置参数：

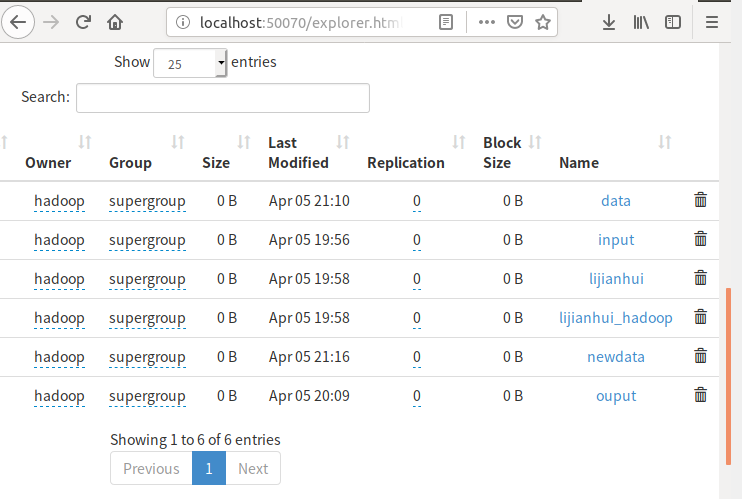


运行成功：

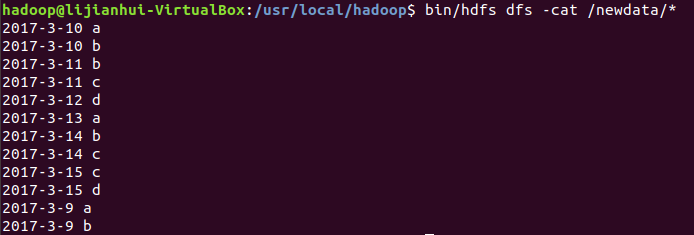




并且在后台管理系统中出现了newdata，顺利完成：



并且我们可以看到输出结果，正如预期：



源代码如下：

package com.apache.hadoop.examples;

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;

import org.apache.hadoop.util.GenericOptionsParser;

public class DataUnique {

//map将输入中的value复制到输出数据的key上，并直接输出

public static class Map extends Mapper<Object,Text,Text,Text>{

private static Text line = new Text();

public void map(Object key,Text value,Context context) throws IOException,InterruptedException{

line = value;

context.write(line, new Text(""));

}

}

//reduce将输入中的key复制到输出数据的key上，并直接输出

public static class Reduce extends Reducer<Text,Text,Text,Text>{

public void reduce(Text key,Iterable<Text> values,Context context) throws IOException,InterruptedException{

context.write(key, new Text(""));

}

}

/\*\*

\* @param args

\*/

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

// TODO Auto-generated method stub

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf,args).getRemainingArgs();

if(otherArgs.length != 2){

System.err.println("Usage WordCount <int> <out>");

System.exit(2);

}

Job job = new Job(conf,"DataUnique");

job.setJarByClass(DataUnique.class);

job.setMapperClass(Map.class);

job.setCombinerClass(Reduce.class);

job.setReducerClass(Reduce.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

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

}

}

# 3 实验总结

通过本次实验，我了解了Hadoop、eclipse的编程流程，理解了MapReduce架构，以及分布式编程思想；在实验过程中，我依然遇到了许多问题，比如找不到对应Hadoop版本的hadoop-eclipse-kepler-plugin-2.10.1.jar包，但是通过查阅网上的资料和自己的实践，完成了用ant来自己配置完成指定版本jar包的操作，完成之后十分有成就感。通过不断地遇到问题，然后解决问题，提升自己面对问题的能力。

另外，了解了在Linux系统中完成各种操作的命令，在Linux系统中，安装软件的方法以及完成各个进程的操作，实际上对我们之后的编程也有很大的帮助。

在Hadoop的帮助下，eclipse编程可以实现对于hdfs文件系统中文件中各种统计处理、数据的操作，这对于大数据来说是十分有意义的，毕竟大数据时代我们分析的就是数据。

因此，本次实验对我来讲意义匪浅，收获很大。