|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名： | | 学号： | 专业年级：  软工2020级 | 班级：14955017 |
| 分组：第1组 | | 实验室：计算机大楼402机房 | 指导教师： | 实验日期： |
| **实验的准备阶段** | **课程名称** | **大数据基础** | | |
| **实验名称** | 单词统计 | | |
| **实验目的** | 1. 理解mapreduce执行原理 2. 理解map,reduce阶段 3. 熟悉统计的原理 | | |
| **实验内容** | 需求：  对下面原始数据进行处理，把文档中所有的英文单词进行统计相同单词的个数。  原始数据：  The ASF provides an established framework for intellectual property and financial  contributions that simultaneously limits potential legal exposure for our project committers  The ASF provides an established framework for intellectual property and financial  contributions that simultaneously limits potential legal exposure for our project committers  输出结果：  ASF 2  The 2  an 2  and 2  committers 2  contributions 2  established 2  exposure 2  financial 2 | | |
| **实验类型**  （打☑） | □验证性 □演示性☑设计性□综合性 | | |
| **实验的重点、难点** | 重点：MapReduce实现统计  难点：编程实现 | | |
| **实验环境** | centos6.5 + jdk1.7.0\_79 + hadoop2.4.1 | | |
| **实验的实施阶段** | **实验步骤及实验结果** | **【实验步骤】**  **一、项目准备阶段**          1.1 在linux系统的命令终端上切换到/simple目录，执行命令：touch source.txt创建一个文件。如图1所示    图1          1.2 在simple目录下，执行命令：vi /simple/source.txt编辑该文件（source.txt文件信息根据课程目录在/home/txtcode/下进行查找），并把数据的信息内容拷贝到该文件中，然后在simple目录可以查看到source.txt文件。如图2所示    图2          1.3 本案例如果在集群中需要用到hadoop的存储和计算，所以在编写程序之前需要先启动Hadoop服务，可以在命令终端执行命令：start-all.sh 把hdfs和yarn服务启动。如图3所示    图3  **二、程序编写**          2.1 在eclipse中的项目列表中，右键点击，选择“new“—>”Java Project…”新建一个项目“EnglishWordsCount” 。 如图4所示    图4          2.2 在项目src目录下，右键点击，选择“新建”创建一个类文件名称为“WordCountMapper”并指定包名”com.wcount” 。如图5所示    图5          2.3 在编写“WordCountMapper”类之前需要把hadoop相关的jar包导入，首先右击项目选择“New”—“Folder”创建一个lib文件夹并把指定位置中(桌面lib文件夹)的包放入该文件中 。如图6所示    图6          2.4 把lib下所有的jar包导入到环境变量，首先全选lib文件夹下的jar包文件，右键点击，选择“build path”-->“add to build path”,添加后，发现在项目下很多奶瓶图表的jar包。如图7所示    图7          2.5 让类“WordCountMapper”继承类Mapper同时指定需要的参数类型，根据业务逻辑修改map类的内容如下。   |  | | --- | | package  com.wcount;  import  java.io.IOException;  import  org.apache.hadoop.io.LongWritable;  import  org.apache.hadoop.io.Text;  import  org.apache.hadoop.mapreduce.Mapper;  public  class WordCountMapper extends Mapper<LongWritable, Text, Text,  LongWritable>{         @Override         protected  void map(LongWritable key, Text value,                       Mapper<LongWritable,  Text, Text, LongWritable>.Context context)                       throws  IOException, InterruptedException {                //get  values string                String  valueString = value.toString();                //spile  string                String  wArr[] = valueString.split(" ");                //for  iterator                for(int  i=0;i<wArr.length;i++){                       //map  out key/value                       context.write(new  Text(wArr[i]), new LongWritable(1));                }         }  } |           2.6 在项目src目录下指定的包名” com.wcount”下右键点击，新建一个类名为“WordCountReducer “并继承Reducer类，然后添加该类中的代码内容如下所示。   |  | | --- | | package  com.wcount;  import  java.io.IOException;  import  java.util.Iterator;  import  org.apache.hadoop.io.LongWritable;  import  org.apache.hadoop.io.Text;  import  org.apache.hadoop.mapreduce.Reducer;  public  class WordCountReducer extends Reducer<Text, LongWritable, Text,  LongWritable> {         @Override         protected  void reduce(Text key, Iterable<LongWritable> v2s,                       Reducer<Text,  LongWritable, Text, LongWritable>.Context context)throws  IOException, InterruptedException {                Iterator<LongWritable>  it = v2s.iterator();                //define  var sum                long  sum = 0;                //  iterator count arr                while(it.hasNext()){                       sum  += it.next().get();                }                context.write(key,  new LongWritable(sum));         }  } |           2.7 在项目src目录下指定的包名” com.wcount”下右键点击，新建一个测试主类名为” TestMapReducer”并指定main主方法。如图8所示    图8          2.8 测试代码如下所示。   |  | | --- | | package  com.wcount;  import  org.apache.hadoop.conf.Configuration;  import  org.apache.hadoop.fs.Path;  import  org.apache.hadoop.io.LongWritable;  import  org.apache.hadoop.io.Text;  import  org.apache.hadoop.mapreduce.Job;  import  org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import  org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  public  class TestMapReducer {         public  static void main(String[] args) throws Exception{                Configuration  conf = new Configuration();                //conf.set("fs.default.name",  "hdfs://192.168.0.202:9000");                //  step1 : get a job                Job  job = Job.getInstance(conf);                //step2:  set jar main class                job.setJarByClass(TestMapReducer.class);                //step3:  set map class and redcer class                job.setMapperClass(WordCountMapper.class);                job.setReducerClass(WordCountReducer.class);                //step4:  set map reduce output type                job.setMapOutputKeyClass(Text.class);                job.setMapOutputValueClass(LongWritable.class);                job.setOutputKeyClass(Text.class);                job.setOutputValueClass(LongWritable.class);                //step5:  set key/value output file format and input/output path                FileInputFormat.setInputPaths(job,  new Path("file:///simple/source.txt"));                FileOutputFormat.setOutputPath(job,  new Path("file:///simple/wcout4"));                //step6:  commit job                job.waitForCompletion(true);         }  } |           2.9 按照以上的步骤，把mapper和reducer阶段以及测试代码编写完毕之后，选中测试类” TestMapReducer “，右键点击选择”Run as”--->”Java Application”,查看控制台显示内容查看是否正确执行。如图9所示    图9          2.10 程序执行完毕之后，可以到输出信息目录/simple/output下，执行查看命令:cat part-r-00000查看对数据处理后产生的结果。如图10所示    图10 | | |
| **实验结果的处理阶段** | **实验结果的分析与总结** |  | | |