**实 验 报 告**

**《MR操作及编程实践》实验 三**

**专业 信息与计算科学 班级 信息151 实验日期\_ 2018 \_年\_03月\_ 26 \_日**

**学号\_ 2015210202019 \_ 姓名 赵华洋 同组学生姓名\_\_无\_\_\_**

|  |  |
| --- | --- |
| 课程名称 | 大数据技术原理与应用 |
| 实验 二 名称 | MR操作及编程实践 |
| 实验地点  与实验环境 | X-711  Hadoop |
| 实验内容、  主要步骤及结果 | **实验内容：**   1. 在eclipse中编译运行输出如下。右击工程Export为Runable JAR file，选择好运行主类和JAR名字后点击Finish。在Terminal中输入命令hadoop jar mywordcount.jar input output即可运行mapreduce任务。  1. 先附上代码：   Job类,name:Statistic  **import** java.io.IOException;  **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.lib.input.FileInputFormat;  **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  **public** **class** Statistic {  **public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException, InterruptedException {  Configuration conf = **new** Configuration();  Job job = Job.*getInstance*(conf, "Statistic");  job.setJarByClass(Statistic.**class**);  job.setMapperClass(MyMapper.**class**);  job.setReducerClass(MyReducer.**class**);  job.setOutputKeyClass(Text.**class**);  job.setOutputValueClass(IntWritable.**class**);  FileInputFormat.*addInputPath*(job, **new** Path("/stat/input"));  FileOutputFormat.*setOutputPath*(job, **new** Path("/stat/output"));  System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);  }  }  Mapper类，name:MyMapper  import java.io.IOException;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  public class MyMapper extends Mapper<Object, Text, Text, IntWritable>{  private static final IntWritable one = new IntWritable(1);  private Text word = new Text();    @Override  public void map(Object key,Text value,Context context) throws IOException, InterruptedException{  String str = value.toString();  char[] chars = str.toLowerCase().toCharArray();  for (int i = 0; i < chars.length; i++) {  char c = chars[i];  if (c>=97 && c<=122) {  this.word.set(String.valueOf(c));  context.write(this.word, one);  }  }  }  }  Reducer类，name:MyReducer  **import** java.io.IOException;  **import** java.util.Iterator;  **import** org.apache.hadoop.io.IntWritable;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.mapreduce.Reducer;  **public** **class** MyReducer **extends** Reducer<Text, IntWritable, Text, IntWritable>{    **private** Text newkey = **new** Text();  **private** IntWritable result = **new** IntWritable();  @Override  **public** **void** reduce(Text key, Iterable<IntWritable> values,Context context) **throws** IOException, InterruptedException{  **int** sum = 0;    IntWritable val;  **for** (Iterator<IntWritable> i$ = values.iterator(); i$.hasNext();) {  val = (IntWritable) i$.next();  sum += val.get();  }  **this**.result.set(sum);  **this**.newkey.set(key.toString()+"("+key.toString().toUpperCase()+")");  context.write(newkey, result);  }  }  文件：1.txt,2.txt    运行结果：   1. 数据去重   程序:  **import** java.io.IOException;  **import** org.apache.hadoop.io.IntWritable;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.mapreduce.Reducer;  **import** org.apache.hadoop.mapreduce.Mapper;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.Path;  **import** org.apache.hadoop.mapreduce.Job;  **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  **public** **class** rmduplicate {  **public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException, InterruptedException {  Configuration conf = **new** Configuration();  Job job = Job.*getInstance*(conf, "rmduplicate");  job.setJarByClass(rmduplicate.**class**);  job.setMapperClass(MyMapper.**class**);  job.setReducerClass(MyReducer.**class**);  job.setOutputKeyClass(Text.**class**);  job.setOutputValueClass(Text.**class**);  FileInputFormat.*addInputPath*(job, **new** Path("/week3/input3"));  FileOutputFormat.*setOutputPath*(job, **new** Path("/week3/output3"));  System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);  }    **public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text>{    @Override  **protected** **void** map(Object key,Text value,Context context) **throws** IOException, InterruptedException{  context.write(value, **new** Text());  }  }    **public** **static** **class** MyReducer **extends** Reducer<Text, IntWritable, Text, Text>{    @Override  **protected** **void** reduce(Text key,Iterable<IntWritable> values,Context context) **throws** IOException, InterruptedException{  context.write(key, **new** Text());  }  }  }  结果:   1. 数据排序   程序:  **import** java.io.IOException;  **import** org.apache.hadoop.io.IntWritable;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.mapreduce.Reducer;  **import** org.apache.hadoop.mapreduce.Mapper;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.Path;  **import** org.apache.hadoop.mapreduce.Job;  **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  **public** **class** MySort {  **public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException, InterruptedException {  Configuration conf = **new** Configuration();  Job job = Job.*getInstance*(conf, "MySort");  job.setJarByClass(MySort.**class**);  job.setMapperClass(MyMapper.**class**);  job.setReducerClass(MyReducer.**class**);  job.setOutputKeyClass(IntWritable.**class**);  job.setOutputValueClass(IntWritable.**class**);  FileInputFormat.*addInputPath*(job, **new** Path("/week3/input4"));  FileOutputFormat.*setOutputPath*(job, **new** Path("/week3/output4"));  System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);  }    **public** **static** **class** MyMapper **extends** Mapper<Object, Text, IntWritable, IntWritable>{    @Override  **protected** **void** map(Object key,Text value,Context context) **throws** IOException, InterruptedException{  IntWritable one = **new** IntWritable(1);  context.write(**new** IntWritable(Integer.*parseInt*(value.toString())), one);  }  }    **public** **static** **class** MyReducer **extends** Reducer<IntWritable, IntWritable, IntWritable, Text>{    @Override  **protected** **void** reduce(IntWritable key,Iterable<IntWritable> values,Context context) **throws** IOException, InterruptedException{  **for** (@SuppressWarnings("unused") IntWritable \_ : values) {  context.write(key, **new** Text());  }  }  }  }  结果：   1. 计算学生平均成绩   程序：  **import** java.io.IOException;  **import** org.apache.hadoop.io.FloatWritable;  **import** org.apache.hadoop.io.IntWritable;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.mapreduce.Reducer;  **import** org.apache.hadoop.mapreduce.Mapper;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.Path;  **import** org.apache.hadoop.mapreduce.Job;  **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  **public** **class** MyAverage {  **public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException, InterruptedException {  Configuration conf = **new** Configuration();  Job job = Job.*getInstance*(conf, "MyAverage");  job.setJarByClass(MyAverage.**class**);  job.setMapperClass(MyMapper.**class**);  job.setReducerClass(MyReducer.**class**);  job.setOutputKeyClass(Text.**class**);  job.setOutputValueClass(IntWritable.**class**);  FileInputFormat.*addInputPath*(job, **new** Path("/week3/input5"));  FileOutputFormat.*setOutputPath*(job, **new** Path("/week3/output5"));  System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);  }    **public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, IntWritable>{    @Override  **protected** **void** map(Object key,Text value,Context context) **throws** IOException, InterruptedException{  String[] str = value.toString().split(" ");  String name = str[0];  String grade = str[1];  context.write(**new** Text(name),**new** IntWritable(Integer.*parseInt*(grade)));  }  }    **public** **static** **class** MyReducer **extends** Reducer<Text, IntWritable, Text, FloatWritable>{    @Override  **protected** **void** reduce(Text key,Iterable<IntWritable> values,Context context) **throws** IOException, InterruptedException{  **int** sum = 0;  **int** times = 0;  **for** (IntWritable grade : values) {  sum += grade.get();  times ++;  }  **float** average = sum/times;  context.write(key, **new** FloatWritable(average));  }  }  }  结果：   1. 实现单表连接   **import java.io.IOException;**  **import java.util.ArrayList;**  **import java.util.Iterator;**  **import java.util.StringTokenizer;**  **import org.apache.hadoop.io.Text;**  **import org.apache.hadoop.mapreduce.Reducer;**  **import org.apache.hadoop.mapreduce.Mapper;**  **import org.apache.hadoop.conf.Configuration;**  **import org.apache.hadoop.fs.Path;**  **import org.apache.hadoop.mapreduce.Job;**  **import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;**  **import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;**  **public class SingleTable {**  **private static int time = 0;**  **public static void main(String[] args) throws IOException,**  **ClassNotFoundException, InterruptedException {**  **Configuration conf = new Configuration();**  **Job job = Job.getInstance(conf, "SingleTable");**  **job.setJarByClass(SingleTable.class);**  **job.setMapperClass(MyMapper.class);**  **job.setReducerClass(MyReducer.class);**  **job.setOutputKeyClass(Text.class);**  **job.setOutputValueClass(Text.class);**  **FileInputFormat.addInputPath(job, new Path("/week3/input6"));**  **FileOutputFormat.setOutputPath(job, new Path("/week3/output6"));**  **System.exit(job.waitForCompletion(true) ? 0 : 1);**  **}**  **public static class MyMapper extends Mapper<Object, Text, Text, Text> {**  **@Override**  **protected void map(Object key, Text value, Context context)**  **throws IOException, InterruptedException {**  **StringTokenizer itr = new StringTokenizer(value.toString());**  **String child = itr.nextToken();**  **String parent = itr.nextToken();**  **if (child != "child") {**  **context.write(new Text(child), new Text("parent:" + parent));**  **context.write(new Text(parent), new Text("child:" + child));**  **}**  **}**  **}**  **public static class MyReducer extends Reducer<Text, Text, Text, Text> {**  @Override  **protected** **void** reduce(Text key, Iterable<Text> values, Context context)  **throws** IOException, InterruptedException {  **if** (*time* == 0) {  context.write(**new** Text("grandchild"), **new** Text("grandparent"));  }  ArrayList<String> childlist = **new** ArrayList<>();  ArrayList<String> parentlist = **new** ArrayList<>();  Iterator<Text> it = values.iterator();  **while** (it.hasNext()) {  Text text = it.next();  String[] splitstr = text.toString().split(":");  **if** (splitstr[0].equals("child")) {  childlist.add(splitstr[1]);  } **else** {  parentlist.add(splitstr[1]);  }  }  **for** (**int** i = 0; i < childlist.size(); i++) {  **for** (**int** j = 0; j < parentlist.size(); j++) {  context.write(**new** Text(childlist.get(i)), **new** Text(  parentlist.get(j)));  }  }  *time*++;  } **}**  **}**  结果：   1. 多表关联   程序：  **import** java.io.IOException;  **import** java.util.ArrayList;  **import** java.util.Iterator;  **import** java.util.StringTokenizer;  **import** java.util.regex.Matcher;  **import** java.util.regex.Pattern;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.mapreduce.Reducer;  **import** org.apache.hadoop.mapreduce.Mapper;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.Path;  **import** org.apache.hadoop.mapreduce.Job;  **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  **public** **class** MultiTable {  **private** **static** **int** *time* = 0;  **public** **static** **void** main(String[] args) **throws** IOException,  ClassNotFoundException, InterruptedException {  Configuration conf = **new** Configuration();  Job job = Job.*getInstance*(conf, "MultiTable");  job.setJarByClass(MultiTable.**class**);  job.setMapperClass(MyMapper.**class**);  job.setReducerClass(MyReducer.**class**);  job.setOutputKeyClass(Text.**class**);  job.setOutputValueClass(Text.**class**);  FileInputFormat.*addInputPath*(job, **new** Path("/week3/input7"));  FileOutputFormat.*setOutputPath*(job, **new** Path("/week3/output7"));  System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);  }  **public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text> {  @Override  **protected** **void** map(Object key, Text value, Context context)  **throws** IOException, InterruptedException {  StringTokenizer itr = **new** StringTokenizer(value.toString(), "\t");  ArrayList<String> splitlist = **new** ArrayList<String>();  **while** (itr.hasMoreTokens()) {  splitlist.add(itr.nextToken());  }  **if** (!splitlist.get(0).equals("factoryname")  && !splitlist.get(0).equals("addressID")) {  **if** (isAdressId(splitlist.get(0))) {  context.write(**new** Text(splitlist.get(0)), **new** Text(  "addressname:" + splitlist.get(1)));  } **else** {  context.write(**new** Text(splitlist.get(1)), **new** Text(  "factoryname:" + splitlist.get(0)));  }  }  }  **private** **boolean** isAdressId(String str) {  Pattern reg = Pattern.*compile*("[0-9]{1,}");  Matcher matcher = reg.matcher(str);  **return** matcher.matches();  }  }  **public** **static** **class** MyReducer **extends** Reducer<Text, Text, Text, Text> {  @Override  **protected** **void** reduce(Text key, Iterable<Text> values, Context context)  **throws** IOException, InterruptedException {  **if** (*time* == 0) {  context.write(**new** Text("factoryname"), **new** Text("addressname"));  }  ArrayList<String> addresslist = **new** ArrayList<>();  ArrayList<String> factorylist = **new** ArrayList<>();  Iterator<Text> it = values.iterator();  **while** (it.hasNext()) {  Text text = it.next();  String[] splitstr = text.toString().split(":");  **if** (splitstr[0].equals("addressname")) {  addresslist.add(splitstr[1]);  } **else** {  factorylist.add(splitstr[1]);  }  }  **for** (**int** i = 0; i < factorylist.size(); i++) {  context.write(**new** Text(factorylist.get(i)), **new** Text(  addresslist.get(0)));  }  *time*++;  }  }  }  结果：   1. 简单倒排索引   **import** java.io.IOException;  **import** java.util.ArrayList;  **import** java.util.StringTokenizer;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.mapreduce.Reducer;  **import** org.apache.hadoop.mapreduce.Mapper;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.Path;  **import** org.apache.hadoop.mapreduce.Job;  **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  **import** org.apache.hadoop.mapreduce.lib.input.FileSplit;  **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  **public** **class** DocSearch {  **public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException, InterruptedException {  Configuration conf = **new** Configuration();  Job job = Job.*getInstance*(conf, "MyAverage");  job.setJarByClass(DocSearch.**class**);  job.setMapperClass(MyMapper.**class**);  job.setReducerClass(MyReducer.**class**);  job.setOutputKeyClass(Text.**class**);  job.setOutputValueClass(Text.**class**);  FileInputFormat.*addInputPath*(job, **new** Path("/bigdata/input8"));  FileOutputFormat.*setOutputPath*(job, **new** Path("/bigdata/output8"));  System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);  }    **public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text>{    @Override  **protected** **void** map(Object key,Text value,Context context) **throws** IOException, InterruptedException{  FileSplit fileSplit = (FileSplit) context.getInputSplit();  String filename = fileSplit.getPath().getName();    StringTokenizer itr = **new** StringTokenizer(value.toString());  **while** (itr.hasMoreElements()) {  String word = (String) itr.nextElement();  context.write(**new** Text(word), **new** Text(filename));  }  }  }    **public** **static** **class** MyReducer **extends** Reducer<Text, Text, Text, Text>{    @Override  **protected** **void** reduce(Text key,Iterable<Text> values,Context context) **throws** IOException, InterruptedException{  ArrayList<String> docs = **new** ArrayList<>();  **for** (Text doc : values) {  String docname = doc.toString();  docs.add(docname);  }  context.write(key, **new** Text(String.*join*(",", docs)));  }  }  }  输出：   1. 专利被引次数统计   **import** java.io.IOException;  **import** java.util.ArrayList;  **import** java.util.StringTokenizer;  **import** org.apache.hadoop.io.IntWritable;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.mapreduce.Reducer;  **import** org.apache.hadoop.mapreduce.Mapper;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.Path;  **import** org.apache.hadoop.mapreduce.Job;  **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  **import** org.apache.hadoop.mapreduce.lib.input.FileSplit;  **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  **public** **class** CountCite {  **public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException, InterruptedException {  Configuration conf = **new** Configuration();  Job job = Job.*getInstance*(conf, "CountCite");  job.setJarByClass(CountCite.**class**);  job.setMapperClass(MyMapper.**class**);  job.setReducerClass(MyReducer.**class**);  job.setOutputKeyClass(Text.**class**);  job.setOutputValueClass(Text.**class**);  FileInputFormat.*addInputPath*(job, **new** Path("/bigdata/input9/cite75\_99.txt"));  FileOutputFormat.*setOutputPath*(job, **new** Path("/bigdata/output9/1"));  System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);  }    **public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text>{    @Override  **protected** **void** map(Object key,Text value,Context context) **throws** IOException, InterruptedException{  String[] strs = value.toString().split(",");  String citing = strs[0];  String cited = strs[1];  **if** (!citing.equals("\"CITING\"")) {  context.write(**new** Text(cited), **new** Text(citing));  }  }  }    **public** **static** **class** MyReducer **extends** Reducer<Text, Text, Text, IntWritable>{    @Override  **protected** **void** reduce(Text key,Iterable<Text> values,Context context) **throws** IOException, InterruptedException{  **int** count = 0;  **for** (Text \_ : values) {  count++;  }  context.write(key, **new** IntWritable(count));  }  }  }  将结果取回本地 hdfs dfs –getmerge /bigdata/input/1 ~/count  结果：   1. 国家专利数统计 && 统计一个国家占全球的专利申请比例   **import** java.io.IOException;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.mapreduce.Reducer;  **import** org.apache.hadoop.mapreduce.Mapper;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.Path;  **import** org.apache.hadoop.mapreduce.Job;  **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  **public** **class** CountryPatents {  **public** **static** **int** *patents\_num* = 0;  **public** **static** **int** *time* = 0;  **public** **static** **void** main(String[] args) **throws** IOException, ClassNotFoundException, InterruptedException {  Configuration conf = **new** Configuration();  Job job = Job.*getInstance*(conf, "CountryPatents");  job.setJarByClass(CountryPatents.**class**);  job.setMapperClass(MyMapper.**class**);  job.setReducerClass(MyReducer.**class**);  job.setOutputKeyClass(Text.**class**);  job.setOutputValueClass(Text.**class**);  FileInputFormat.*addInputPath*(job, **new** Path("/bigdata/input9/apat63\_99.txt"));  FileOutputFormat.*setOutputPath*(job, **new** Path("/bigdata/output9/2"));  System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);  }    **public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text>{    @Override  **protected** **void** map(Object key,Text value,Context context) **throws** IOException, InterruptedException{  *patents\_num* ++ ;  String[] strs = value.toString().split(",");  String country = strs[4];  String patent = strs[0];  **if** (!patent.equals("\"PATENT\"")) {  context.write(**new** Text(country), **new** Text(patent));  }  }  }    **public** **static** **class** MyReducer **extends** Reducer<Text, Text, Text, Text>{    @Override  **protected** **void** reduce(Text key,Iterable<Text> values,Context context) **throws** IOException, InterruptedException{  **int** count = 0;  **for** (Text \_ : values) {  count++;  }  **double** ratio = count/(**double**)*patents\_num*;  **if** (*time* == 0) {  context.write(**new** Text(""), **new** Text("Country\tCounts\tRatio"));  context.write(key, **new** Text(String.*valueOf*(count)+"\t"+String.*format*("%.5f", ratio)));  }**else** {  context.write(key, **new** Text(String.*valueOf*(count)+"\t"+String.*format*("%.5f", ratio)));  }  *time*++;  }  }  }  结果：   1. 统计一个国家的专利引用率   **import** java.io.IOException;  **import** org.apache.hadoop.io.IntWritable;  **import** org.apache.hadoop.io.Text;  **import** org.apache.hadoop.mapreduce.Reducer;  **import** org.apache.hadoop.mapreduce.Mapper;  **import** org.apache.hadoop.conf.Configuration;  **import** org.apache.hadoop.fs.Path;  **import** org.apache.hadoop.mapreduce.Job;  **import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  **import** org.apache.hadoop.mapreduce.lib.jobcontrol.ControlledJob;  **import** org.apache.hadoop.mapreduce.lib.jobcontrol.JobControl;  **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  **public** **class** CiteRatio {  **public** **static** **int** *Cited\_nums* = 0;  **public** **static** **int** *time* = 0;  **public** **static** **void** main(String[] args) **throws** IOException,  ClassNotFoundException, InterruptedException {  Configuration conf = **new** Configuration();  Job job1 = Job.*getInstance*(conf, "CiteRatio1");  job1.setJarByClass(CiteRatio.**class**);  job1.setMapperClass(MyMapper1.**class**);  job1.setReducerClass(MyReducer1.**class**);  job1.setOutputKeyClass(Text.**class**);  job1.setOutputValueClass(Text.**class**);    // 多job任务  // job1-1加入控制器  ControlledJob ctrljob1 = **new** ControlledJob(conf);  ctrljob1.setJob(job1);  FileInputFormat.*addInputPath*(job1, **new** Path(args[0]));  FileOutputFormat.*setOutputPath*(job1, **new** Path(args[1]));      Job job2 = Job.*getInstance*(conf, "CiteRatio2");  job2.setJarByClass(CiteRatio.**class**);  job2.setMapperClass(MyMapper2.**class**);  job2.setReducerClass(MyReducer2.**class**);  job2.setOutputKeyClass(Text.**class**);  job2.setOutputValueClass(IntWritable.**class**);  // job1-2加入控制器  ControlledJob ctrljob2 = **new** ControlledJob(conf);  ctrljob2.setJob(job2);  // 意思为job2的启动，依赖于job1作业的完成  ctrljob2.addDependingJob(ctrljob1);    FileInputFormat.*addInputPath*(job2, **new** Path(args[1]));  FileOutputFormat.*setOutputPath*(job2, **new** Path(args[2]));  // 主的控制容器，控制上面的总的两个子作业  JobControl jobctrl = **new** JobControl("myctrl");  // 添加到总的JobControl里，进行控制  jobctrl.addJob(ctrljob1);  jobctrl.addJob(ctrljob2);  // 在线程启动，记住一定要有这个  Thread t = **new** Thread(jobctrl);  t.start();  **while** (**true**) {  **if** (jobctrl.allFinished()) {// 如果作业成功完成，就打印成功作业的信息  System.*out*.println(jobctrl.getSuccessfulJobList());  jobctrl.stop();  **break**;  }  }  }  **public** **static** **class** MyMapper1 **extends** Mapper<Object, Text, Text, Text> {  @Override  **protected** **void** map(Object key, Text value, Context context)  **throws** IOException, InterruptedException {  String[] strs = value.toString().split(",");  **if** (!strs[0].equals("\"CITING\"") && !strs[0].equals("\"PATENT\"")) {  **if** (strs.length == 2) {  *Cited\_nums*++;  String citing = strs[0];  String cited = strs[1];  context.write(**new** Text(cited), **new** Text("citing:" + citing));  } **else** {  String patent = strs[0];  String country = strs[4];  context.write(**new** Text(patent), **new** Text("country:"  + country));  }  }  }  }  **public** **static** **class** MyReducer1 **extends**  Reducer<Text, Text, Text, IntWritable> {  @Override  **protected** **void** reduce(Text key, Iterable<Text> values, Context context)  **throws** IOException, InterruptedException {  **int** count = 0;  Text country = **new** Text();  **for** (Text text : values) {  String[] strs = text.toString().split(":");  **if** (strs[0].equals("citing")) {  count++;  } **else** {  country.set(strs[1]);  }  }  **if** (country.toString().length() != 0) {  context.write(country, **new** IntWritable(count));  }  }  }  **public** **static** **class** MyMapper2 **extends**  Mapper<Object, Text, Text, IntWritable> {  @Override  **protected** **void** map(Object key, Text value, Context context)  **throws** IOException, InterruptedException {  String[] strs = value.toString().split("\t");  context.write(**new** Text(strs[0]),  **new** IntWritable(Integer.*parseInt*(strs[1])));  }  }  **public** **static** **class** MyReducer2 **extends**  Reducer<Text, IntWritable, Text, Text> {  @Override  **protected** **void** reduce(Text key, Iterable<IntWritable> values,  Context context) **throws** IOException, InterruptedException {  **int** sum = 0;  **for** (IntWritable value : values) {  sum += value.get();  }  **double** ratio = (**double**)sum/*Cited\_nums*;  **if** (*time* == 0) {  context.write(**new** Text("Country"), **new** Text("CitedRatio"));  context.write(key, **new** Text(String.*format*("%.6f", ratio)));  }**else** {  context.write(key, **new** Text(String.*format*("%.6f", ratio)));  }  *time*++;  }  }  }  结果： |
| 实验中碰到的问题、  体会 | 1、import相关包时尤其注意，例如Text类，有多个对应的包，import com.sun.jersey.core.impl.provider.entity.XMLJAXBElementProvider.Text;如上导入包则报错，正确的包应为import org.apache.hadoop.io.Text;所以可以复制还是尽量复制吧。  2、iterable对象在迭代一次后就无法使用了，我重新创建iterator也没用。  3、time==0多做一步  4、多job时参数的设定只能在run configuration中指定。 |
| 得分 |  |

注：学生做每个实验都必须写实验报告。实验报告可以是纸质的，也可以是电子形式的。后面的表3“实验报告”仅作为参考。任课教师可根据实验的特殊情况自行设计实验报告。实验报告（如是电子形式的必须刻成光盘）在学期结束时交给教务科存档。

实验指导教师签名\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_年\_\_\_\_月\_\_\_\_日