|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名：崔文帅 | | 学号：2025060163 | 专业年级：  网工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：2023-4-26 |
| **实验的准备阶段** | **课程名称** | **大数据基础** | | |
| **实验名称** | **电话号码分装及流量统计** | | |
| **实验目的** | 1）理解mapreduce执行原理      2）理解map,reduce阶段      3）熟悉mapreduce的排序和分区 | | |
| **实验内容** | 1）  需求：      对所给的所有电话号码产生的流量记录进行封装并把封装类作为map输出键进行指定电话号码前3为的大小进行排序升序后输出。      原始数据：      18610117315 200 300      13718855152 300 500      18610117315 100 300      18610117315 500 700      13718855152 400 900      13121521297 100 800      原理：         首先按电话号码记录信息的封装类作为键进行排序，同时为电话记录的封装类型指定所属的数据类型（Writable），重写排序方法和规则。  2）  需求：      对所给的所有电话号码产生的流量记录进行按电话号码进行汇总，求出所有相同电话号码产生的流量和。      原始数据：      18610117315 200 300      13718855152 300 500      18610117315 100 300      18610117315 500 700      13718855152 400 900      13121521297 100 800      原理：         首先按电话号码作为键进行排序，相同键的内容形成一个集合，然后把相同键的所有内容值进行流量相加，最后按照指定分区条件进行分区输出。 | | |
| **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分区实现  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，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 本案例如果在集群田需要用到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“FlowAccount” 。 如图4所示    图4          2.2 在项目src目录下，右键点击，选择”New”—“Class”创建一个类文件名称为“Flow”并指定包名” com.flow”,该类是对给定数据的三列值的封装，并作为mapper的输出键值对象 。实现代码如下：   |  | | --- | | package  com.flow;  import  java.io.DataInput;  import  java.io.DataOutput;  import  java.io.IOException;  import  org.apache.hadoop.io.Text;  import  org.apache.hadoop.io.WritableComparable;  public  class Flow  implements WritableComparable<Flow>{       private String tel;       private int down;       private int up;       public Flow(){       }       public Flow(String tel, int down, int up) {           super();           this.tel  = tel;          this.down = down;           this.up = up;       }       public String getTel() {           return  tel;       }       public void setTel(String tel) {           this.tel  = tel;       }       public int getDown() {           return  down;       }       public void setDown(int down) {           this.down = down;       }       public int getUp() {           return  up;       }       public void setUp(int up) {          this.up = up;       }       //按自定义字段顺序写入到数据输出流中       @Override       public void write(DataOutput out) throws IOException  {           out.writeUTF(tel);          out.writeInt(down);           out.writeInt(up);       }      //按写入中的顺序从数据输入流读取字段数据       @Override       public void readFields(DataInput in) throws  IOException {           this.tel  = in.readUTF();           this.down = in.readInt();           this.up = in.readInt();       }  //设置排序方式       @Override       public int compareTo(Flow o) {           int telnum1 = Integer.parseInt(this.getTel().substring(0, 3));           int telnum2 = Integer.parseInt(o.getTel().substring(0, 3));           if(telnum1>=telnum2){                return  1;           }else{                return  -1;           }       }       @Override       public String toString() {           return  this.tel+":("+this.down+","+this.up+")";       }  } |           2.3 在项目src目录下，右键点击，选择”New”—“Class”创建一个类文件名称为“FlowMapper”并指定包名” com.flow” 。如图5所示    图5          2.4 在编写“FlowMapper”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          2.5 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          2.6 让类“FlowMapper”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.flow;  import java.io.IOException;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  public class FlowMapper  extends Mapper<LongWritable, Text, Flow, NullWritable> {           @Override           protected  void map(LongWritable key, Text value,                              Mapper<LongWritable, Text, Flow, NullWritable>.Context  context)                              throws  IOException, InterruptedException  {                     //读取一行文本信息                     String  lineMsg = value.toString();                     //对读取记录进行切分                     String  arrLine[] = lineMsg.split("  ");                     Flow  flow = new Flow(arrLine[0],Integer.parseInt(arrLine[1]),Integer.parseInt(arrLine[2]));                     //指定map输出的键值                     context.write(flow, NullWritable.get());           }  } |           2.7 在项目src目录下指定的包名” com.flow”下右键点击，新建一个类名为“FlowReducer “并继承Reducer类，然后添加该类中的代码内容如下所示。   |  | | --- | | package com.flow;  import java.io.IOException;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.mapreduce.Reducer;  public class FlowReducer  extends Reducer<Flow, NullWritable, Flow, NullWritable> {       @Override      protected void reduce(Flow  key, Iterable<NullWritable>  vs2,                 Reducer<Flow,  NullWritable, Flow, NullWritable>.Context  context)                 throws  IOException, InterruptedException  {       context.write(key, NullWritable.get());      }  } |           2.8 在项目src目录下指定的包名” com.flow”下右键点击，新建一个测试主类名为” TestFlow ”并指定main主方法。如图8所示    图8          2.9 测试代码如下所示。   |  | | --- | | package com.flow;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.mapreduce.Job;  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  public class TestFlow  {           public  static void main(String[] args) throws Exception{                     Configuration  conf = new Configuration();                     Job  job = Job.getInstance(conf);                     job.setJarByClass(TestFlow.class);                     //设置map-reduce类                     job.setMapperClass(FlowMapper.class);                     job.setReducerClass(FlowReducer.class);                     //设置map输出键值对类型                     job.setMapOutputKeyClass(Flow.class);                     job.setMapOutputValueClass(NullWritable.class);                     //设置reduce输出键值对类型                     job.setOutputKeyClass(Flow.class);                     job.setOutputValueClass(NullWritable.class);                     //设置本地输入输出路径                     FileInputFormat.setInputPaths(job, new  Path("file:///simple/source.txt"));                     FileOutputFormat.setOutputPath(job, new Path("file:///simple/output"));                     //可以动态设置路径参数  //               FileInputFormat.setInputPaths(job, new Path(args[0]));  //               FileOutputFormat.setOutputPath(job, new Path(args[1]));                     job.waitForCompletion(true);           }  } |           2.10 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” TestFlow “，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          2.11 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  **【实验目的】**      1）理解mapreduce执行原理      2）理解map,reduce阶段      3）熟悉mapreduce的分区  **【实验原理】**      需求：      对所给的所有电话号码产生的流量记录进行按电话号码进行汇总，求出所有相同电话号码产生的流量和。      原始数据：      18610117315 200 300      13718855152 300 500      18610117315 100 300      18610117315 500 700      13718855152 400 900      13121521297 100 800      原理：         首先按电话号码作为键进行排序，相同键的内容形成一个集合，然后把相同键的所有内容值进行流量相加，最后按照指定分区条件进行分区输出。  **【实验环境】**      本次环境是：centos6.5 + jdk1.7.0\_79 + hadoop2.4.1 + eclipse      日志文件source.txt存放在/home/txtcode/目录下  **【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，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 本案例如果在集群田需要用到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“FlowAccount” 。 如图4所示    图4          2.2 在项目src目录下，右键点击，选择”New”—“Class”创建一个类文件名称为“Flow”并指定包名” com.flow”,该类是对给定数据的三列值的封装，并作为mapper的输出键值对象 。实现代码如下：   |  | | --- | | package  com.flow;  import  java.io.DataInput;  import  java.io.DataOutput;  import  java.io.IOException;  import  org.apache.hadoop.io.Text;  import  org.apache.hadoop.io.WritableComparable;  public  class Flow  implements WritableComparable<Text>{       private String tel;       private int down;       private int up;       public Flow(){       }       public Flow(String tel, int down, int up) {           super();           this.tel  = tel;           this.down = down;           this.up = up;       }       public String getTel() {           return  tel;       }       public void setTel(String tel) {           this.tel  = tel;       }       public int getDown() {           return  down;       }       public void setDown(int down) {           this.down = down;       }       public int getUp() {           return  up;       }       public void setUp(int up) {           this.up = up;       }       @Override       public void write(DataOutput out) throws IOException  {           out.writeUTF(tel);           out.writeInt(down);           out.writeInt(up);       }       @Override       public void readFields(DataInput in) throws  IOException {           this.tel  = in.readUTF();           this.down = in.readInt();           this.up = in.readInt();       }       @Override       public int compareTo(Text o) {           return  1;       }  } |           2.3 在项目src目录下，右键点击，选择”New”—“Class”创建一个类文件名称为“FlowPartitioner”并指定包名”com.flow”,该类是对数据处理后的结果进行分区设置 。代码实现如下：   |  | | --- | | package  com.flow;  import  java.util.HashMap;  import  org.apache.hadoop.io.Text;  import  org.apache.hadoop.mapreduce.Partitioner;  public  class FlowPartitioner extends Partitioner<Text,  Flow> {       //定义电话号码对应的标号       public static HashMap<String,Integer> hm = new HashMap<String,Integer>();       static{           hm.put("13718855152", 1);           hm.put("18610117315", 2);       }       //根据不同的键值返回响应的编号       @Override       public int getPartition(Text key, Flow  value, int numPartitions)  {           String  tel = key.toString();           Integer  code = hm.get(tel);           if(code  == null){                code  = 0;           }           return  code;       }  } |           2.4 在项目src目录下，右键点击，选择”New”—“Class”创建一个类文件名称为“FlowMapper”并指定包名” com.flow” 。如图5所示    图5          2.5 在编写“FlowMapper”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          2.6 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          2.7 让类“FlowMapper”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.flow;  import java.io.IOException;  import org.apache.hadoop.io.LongWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Mapper;  public class FlowMapper  extends Mapper<LongWritable, Text, Text,  Flow> {           @Override           protected  void map(LongWritable key, Text value,                              Mapper<LongWritable, Text, Text, Flow>.Context context)                              throws  IOException, InterruptedException  {                     //读取一行文本信息                     String  lineMsg = value.toString();                     //对读取记录进行切分                     String  arrLine[] = lineMsg.split("  ");                     //指定map输出的键值                     context.write(new Text(arrLine[0]),  new  Flow(arrLine[0],Integer.parseInt(arrLine[1]),Integer.parseInt(arrLine[2])));           }  } |           2.8 在项目src目录下指定的包名” com.flow”下右键点击，新建一个类名为”FlowReducer”并继承Reducer类，然后添加该类中的代码内容如下所示。   |  | | --- | | package com.flow;  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 FlowReducer  extends Reducer<Text, Flow, Text, LongWritable>  {       @Override      protected void reduce(Text  key, Iterable<Flow> vs2,                 Reducer<Text,  Flow, Text, LongWritable>.Context context)                 throws  IOException, InterruptedException  {        //获取map输出键对应的值的集合        Iterator<Flow>  it = vs2.iterator();       int total = 0;       //对集合进行遍历统计每个键对应流量的和        while(it.hasNext()){                 Flow f  = it.next();                 int sum = f.getUp()+f.getDown();                 total+=sum;       }       context.write(key, new LongWritable(total));      }  } |           2.9 在项目src目录下指定的包名” com.flow”下右键点击，新建一个测试主类名为” TestFlow ”并指定main主方法。如图8所示    图8          2.10 测试代码如下所示。   |  | | --- | | package com.flow;  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 TestFlow  {           public  static void main(String[] args) throws Exception{                     Configuration  conf = new Configuration();                     Job  job = Job.getInstance(conf);                     job.setJarByClass(TestFlow.class);                     //设置map-reduce类                     job.setMapperClass(FlowMapper.class);                     job.setReducerClass(FlowReducer.class);                     //设置map输出键值对类型                     job.setMapOutputKeyClass(Text.class);                     job.setMapOutputValueClass(Flow.class);                     //设置分区类                     job.setPartitionerClass(FlowPartitioner.class);                     //设置3个reduce任务，产生三个文件                     job.setNumReduceTasks(3);                     //设置reduce输出键值对类型                     job.setOutputKeyClass(Text.class);                     job.setOutputValueClass(LongWritable.class);                     //设置本地输入输出路径                     FileInputFormat.setInputPaths(job, new  Path("file:///simple/source.txt"));                     FileOutputFormat.setOutputPath(job, new  Path("file:///simple/output"));                     //可以动态设置路径参数  //               FileInputFormat.setInputPaths(job, new Path(args[0]));  //               FileOutputFormat.setOutputPath(job, new Path(args[1]));                     job.waitForCompletion(true);           }  } |           2.11 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” TestFlow “，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          2.12 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，cat part-r-00001，cat part-r-00002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** |  | | |