|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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** | **MR调优学生信息区间统计** | | |
| **实验目的** | 1）理解mapreduce执行原理      2）理解map,reduce阶段      3）熟悉mapreduce的分区原理 | | |
| **实验内容** | 1）需求：      对所给的所给的学生信息，包括用户名和年龄，通过map-reduce模拟对海量学生信息进行处理，即实现把所有的学生信息按年龄界限为18进行分区。      原始数据：      张三 13      李四 28      王五 34      赵六 22      钱七 17      刘八 18      胡三 29      原理：         首先把学生的信息进行封装成新的类型，作为map’阶段的输出值，因为不需要按键排序，所以键只需要设置为NullWritable类型即可。在reduce阶段对所有的值进行遍历之后直接输出即可 | | |
| **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“StuAgeCata” 。 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“StudentWritable”并指定包名” com.stu”,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.stu;  import  java.io.DataInput;  import  java.io.DataOutput;  import  java.io.IOException;  import  org.apache.hadoop.io.WritableComparable;  public  class StudentWritable implements WritableComparable<StudentWritable>  {       private String name;       private int age;       public String getName() {           return  name;       }       public void setName(String name) {           this.name  = name;       }       public int getAge() {           return  age;       }       public void setAge(int age) {           this.age = age;       }       public StudentWritable(){       }       public StudentWritable(String name,int  age){           this.name  = name;           this.age = age;       }       @Override       public void write(DataOutput out) throws IOException  {            out.writeUTF(name);            out.writeInt(age);       }       @Override       public void readFields(DataInput in) throws  IOException {           this.name  = in.readUTF();           this.age = in.readInt();       }       @Override       public String toString() {           return  "StudentWritable [name=" + name + ",  age=" + age + "]";       }       @Override       public int compareTo(StudentWritable o) {           return  1;       }  } |           2.3 在项目src目录下，右键点击包名” com.stu”选择”New”—“Class”创建一个类文件名称为“StuPartitioner”,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.stu;  import  org.apache.hadoop.io.NullWritable;  import  org.apache.hadoop.mapreduce.Partitioner;  public  class StuPartitioner extends Partitioner<NullWritable, StudentWritable>  {       @Override       public int getPartition(NullWritable key, StudentWritable  value,                int numPartitions) {           //按年龄进行分区，分区条件为大于18岁和小于18岁           if(value.getAge()>=18){                return  1;           }else{                return  0;           }       }  } |           2.4 在项目src目录下，右键点击包名” com.stu”选择”New”—“Class”创建一个类文件名称为“StudentMapper”。如图5所示    图5          2.5 在编写“StudentMapper”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“StudentMapper”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.stu;  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 StudentMapper  extends Mapper<LongWritable, Text, NullWritable, StudentWritable>  {        @Override      protected void map(                 LongWritable key,                 Text  value,                 Mapper<LongWritable, Text, NullWritable,  StudentWritable>.Context context)                 throws  IOException, InterruptedException  {       String stuArr[] = value.toString().split("  ");       context.write(NullWritable.get(),  new StudentWritable(stuArr[0],Integer.parseInt(stuArr[1])));      }  } |           2.8 在项目src目录下指定的包名” com.stu”下右键点击，新建一个类名为“StudentRedcer“并继承Reducer类，然后添加该类中的代码内容如下所示。   |  | | --- | | package com.stu;  import java.io.IOException;  import java.util.Iterator;  import org.apache.hadoop.io.NullWritable;  import org.apache.hadoop.mapreduce.Reducer;  public class StudentRedcer  extends Reducer<NullWritable, StudentWritable, NullWritable, StudentWritable>{           @Override           protected  void reduce(                              NullWritable key,                              Iterable<StudentWritable>  iter,                              Reducer<NullWritable, StudentWritable, NullWritable, StudentWritable>.Context  context)                              throws  IOException, InterruptedException  {                       //遍历数据                       Iterator<StudentWritable>  it=  iter.iterator();                       while(it.hasNext()){                                context.write(NullWritable.get(), it.next());                       }           }  } |           2.9 在项目src目录下指定的包名” com.stu”下右键点击，新建一个测试主类名为” TestStuMapReducer”并指定main主方法。如图8所示    图8          2.10 最后在项目src目录下指定的包名” com.stu”下右键点击，新建一个测试主类名为” TestStuMapReducer “添加测试代码如下所示。   |  | | --- | | package com.stu;  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 TestStuMapReducer  {           public  static void main(String[] args) throws Exception{                     Configuration  conf = new Configuration();                     //conf.set("fs.default.name", "hdfs://192.168.0.xxx:9000");                     //  step1 : get a job                     Job  job = Job.getInstance(conf);                     //step2:  set jar main class                     job.setJarByClass(TestStuMapReducer.class);                     //step3:  set map class and redcer class                     job.setMapperClass(StudentMapper.class);                     job.setReducerClass(StudentRedcer.class);                     job.setPartitionerClass(StuPartitioner.class);                     job.setNumReduceTasks(2);                       //step4:  set map reduce output type                     job.setMapOutputKeyClass(NullWritable.class);                     job.setMapOutputValueClass(StudentWritable.class);                     job.setOutputKeyClass(NullWritable.class);                     job.setOutputValueClass(StudentWritable.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/output"));                     //CombineFileInputFormat.setInputPaths(job, inputPaths);                     //step6:  commit job                     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** |  | | |