**MapReduce自定义排序、分区、分组案例**

字段说明

班级    学号    姓名     语文    数学    英语

1307    7026    邝卓男     95    88    98

1.求每个学生的总分和平均分，并按总分降序排序

2.求每个班级每一门课程的平均分，不同班级的结果输出到不同的结果文件

3.求每个班级的总分最高的前5个学生

**1、求每个学生的总分和平均分，并按总分降序排序**

   思路： 当看到“每个”时，就把后面的字段当成分组字段，需要对总分进行排序，所以需要自定义排序

在开始代码的编写前，首先要确定map、reduce中的key和value各是什么？

1）StudentScore类

public class StudentScore implements WritableComparable<StudentScore> {

private String name;

private int sum;

private double avg;

public StudentScore(){

super();

}

public StudentScore(String name,int sum,Double avg){

this.name=name;

this.sum=sum;

this.avg=avg;

}

public int getSum() {

return sum;

}

public double getAvg() {

return avg;

}

public void setSum(int sum) {

this.sum = sum;

}

public void setAvg(double avg) {

this.avg = avg;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return this.name+"\t"+this.sum+"\t"+this.avg;

}

@Override

public void write(DataOutput out) throws IOException {

out.writeUTF(name);

out.writeInt(sum);

out.writeDouble(avg);

}

@Override

public void readFields(DataInput in) throws IOException {

this.name=in.readUTF();

this.sum=in.readInt();

this.avg=in.readDouble();

}

@Override

public int compareTo(StudentScore o) {

return o.getSum()-this.getSum(); //根据总分进行倒序排序

}

}

2）主类 MyScoreMapReduce

public class MyScoreMapReduce {

public static class MyMapper extends Mapper<LongWritable,Text,StudentScore,Text> {

Text ovalue=new Text();

@Override

protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

// 读取一行数据进行切分

String[] fields = value.toString().split("\t");

int chinese=Integer.parseInt(fields[3]);

int math=Integer.parseInt(fields[4]);

int english=Integer.parseInt(fields[5]);

int sum=chinese+math+english;

Double avg=(1.0)\*sum/3;

StudentScore ss=new StudentScore(fields[2],sum,avg); //创建自定义类对象

ovalue.set(fields[0]+"\t"+fields[1]);

context.write(ss,ovalue);

}

}

public static class MyReduce extends Reducer<StudentScore,Text,Text,StudentScore>{

@Override

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

for(Text v: values){

context.write(v,key);

}

}

}

public static void main(String[] args) {

Configuration conf=new Configuration();

System.setProperty("HADOOP\_HOME\_USER","qyl");

conf .set( "fs.defaultFS" , "hdfs://qyl02:9000" );

try {

Job job=Job.getInstance(conf);

job.setJarByClass(MyScoreMapReduce.class);

job.setMapperClass(MyMapper.class);

job.setReducerClass(MyReduce.class);

job.setMapOutputKeyClass(StudentScore.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(StudentScore.class);

Path inpath=new Path("/student/shuju");

FileInputFormat.addInputPath(job,inpath);

Path outpath=new Path("/student/result01");

if(outpath.getFileSystem(conf).exists(outpath)){

outpath.getFileSystem(conf).delete(outpath,true);

}

FileOutputFormat.setOutputPath(job,outpath);

job.waitForCompletion(true);

} catch (Exception e) {

e.printStackTrace();

}

}

}

3）结果

只列举一下

1304 4021 张晓宸 293 97.66666666666667

1307 7007 刘俊辉 293 97.66666666666667

1307 7019 刘程望 291 97.0

1304 4054 谭凌云 291 97.0

1305 5024 吴思妮 291 97.0

1306 6001 张轩铭 291 97.0

1304 4028 宇佳杨 290 96.66666666666667

1304 4027 张鑫 289 96.33333333333333

1304 4026 胡量 289 96.33333333333333

1303 3003 王凭 289 96.33333333333333

1303 3004 唐翔 289 96.33333333333333

**2.求每个班级每一门课程的平均分，不同班级的结果输出到不同的结果文件**

思路： 分区字段为班级

          排序字段为 班级和课程

          分组字段为 班级和课程

代码编写

1）自定义分区

public class MyPartition extends Partitioner<MySort, IntWritable> {

@Override

public int getPartition(MySort key, IntWritable arg1, int arg2) {

if(key.getClassname().equals("1303")){

return 0;

}

if(key.getClassname().equals("1304")){

return 1;

}

if(key.getClassname().equals("1305")){

return 2;

}

if(key.getClassname().equals("1306")){

return 3;

}else{

return 4;

}

}

}

2）自定义排序

public class MySort implements WritableComparable<MySort> {

private String classname;

private String course;

public MySort() {

super();

}

public MySort(String classname, String course) {

this.classname = classname;

this.course = course;

}

public String getClassname() {

return classname;

}

public String getCourse() {

return course;

}

@Override

public String toString() {

return classname+"\t"+course;

}

public void setClassname(String classname) {

this.classname = classname;

}

public void setCourse(String course) {

this.course = course;

}

@Override

public void write(DataOutput out) throws IOException {

out.writeUTF(classname);

out.writeUTF(course);

}

@Override

public void readFields(DataInput in) throws IOException {

this.classname=in.readUTF();

this.course=in.readUTF();

}

@Override

**public int compareTo(MySort o) {**

int temp=this.getClassname().compareTo(o.getClassname());

if(temp==0){

temp=this.getCourse().compareTo(o.getCourse());

}

return temp;

}

}

3)自定义分组

public class MyGrouping extends WritableComparator {

public MyGrouping(){

super(MySort.class,true);

}

@Override

public int compare(WritableComparable a, WritableComparable b) {

MySort aa=(MySort)a;

MySort bb=(MySort)b;

int i= aa.getClassname().compareTo(bb.getClassname());

if(i==0){

return aa.getCourse().compareTo(bb.getCourse());

}

return i;

}

}

4）编写主类  MyClassMapReduce

public class MyClassMapReduce {

public static class MyMapper extends Mapper<LongWritable,Text,MySort,IntWritable>{

MySort ms=new MySort();

@Override

protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

String[] fields = value.toString().split("\t");

int score=0;

ms.setClassname(fields[0]);

for(int i=3;i<fields.length;i++){

if(i==3){

ms.setCourse("语文");

score=Integer.parseInt(fields[3]);

context.write(ms,new IntWritable(score));

}

if(i==4){

ms.setCourse("数学");

score=Integer.parseInt(fields[4]);

context.write(ms,new IntWritable(score));

}else{

ms.setCourse("英语");

score=Integer.parseInt(fields[5]);

context.write(ms,new IntWritable(score));

}

}

}

}

public static class MyReducer extends Reducer<MySort,IntWritable,MySort,Text>{

@Override

protected void reduce(MySort key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {

int sum=0;int count=0;

for(IntWritable v: values){

sum+=v.get();

count++;

}

context.write(key,new Text(""+1.0\*sum/count));

System.out.println(key.toString()+"---------"+1.0\*sum/count);

}

}

public static void main(String[] args) {

Configuration conf =new Configuration();

System.setProperty("HADOOP\_USER\_NAME", "qyl");

conf .set( "fs.defaultFS" , "hdfs://qyl02:9000" );

try {

Job job=Job.getInstance(conf);

job.setJarByClass(MyClassMapReduce.class);

job.setMapperClass(MyMapper.class);

job.setReducerClass(MyReducer.class);

job.setMapOutputKeyClass(MySort.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(MySort.class);

job.setOutputValueClass(Text.class);

**job.setPartitionerClass(MyPartition.class);**

**job.setGroupingComparatorClass(MyGrouping.class);**

**job.setNumReduceTasks(5);**

//指定需要统计的文件输入路径

Path inpath=new Path("/student/shuju");

FileInputFormat.addInputPath(job, inpath);

//指定输出目录 输出路径不能存在，否则就会报错 默认是覆盖式的输出

Path outpath=new Path("/student/result02");

if(outpath.getFileSystem(conf).exists(outpath)){

outpath.getFileSystem(conf).delete(outpath,true);

}

FileOutputFormat.setOutputPath(job, outpath);

job.waitForCompletion(true);

} catch (Exception e) {

e.printStackTrace();

}

}

}

  5）结果

列举一下

1303 数学 86.78181818181818

1303 英语 90.0

1303 语文 95.12727272727273

  3.求每个班级的总分最高的前5个学生

思路：自定义排序 字段为班级和总成绩

 自定义分组 字段为 班级

1）自定义排序

public class MyClassAndScore implements WritableComparable<MyClassAndScore> {

private String classname;

private int sum;

public String getClassname() {

return classname;

}

public int getSum() {

return sum;

}

public void setClassname(String classname) {

this.classname = classname;

}

public void setSum(int sum) {

this.sum = sum;

}

public MyClassAndScore() {

super();

}

@Override

public String toString() {

return classname+"\t"+sum;

}

public MyClassAndScore(String classname, int sum) {

this.classname = classname;

this.sum = sum;

}

@Override

public void write(DataOutput out) throws IOException {

out.writeUTF(classname);

out.writeInt(sum);

}

@Override

public void readFields(DataInput in) throws IOException {

this.classname=in.readUTF();

this.sum=in.readInt();

}

@Override

public int compareTo(MyClassAndScore o) {

int temp=this.getClassname().compareTo(o.getClassname());

if(temp==0){

temp=o.getSum()-this.getSum();

}

return temp;

}

}

2）自定义分区

package com.qyl.lt.mapreduce.test03;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;

public class MyGroup2 extends WritableComparator {

public MyGroup2(){

super(MyClassAndScore.class,true);

}

@Override

public int compare(WritableComparable a, WritableComparable b) {

MyClassAndScore aa=(MyClassAndScore)a;

MyClassAndScore bb=(MyClassAndScore)b;

return aa.getClassname().compareTo(bb.getClassname());

}

}

3）编写主类  MyAllSource

public class MyAllSource {

public static class MyMapper extends Mapper<LongWritable,Text,MyClassAndScore,Text>{

@Override

protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

String[] fields = value.toString().split("\t");

int chinese=Integer.parseInt(fields[3]);

int math=Integer.parseInt(fields[4]);

int english=Integer.parseInt(fields[5]);

int sum=chinese+math+english;

MyClassAndScore ms=new MyClassAndScore(fields[0],sum);

context.write(ms,new Text(fields[1]+"\t"+fields[2]));

}

}

public static class MyReducer extends Reducer<MyClassAndScore,Text,Text,Text>{

@Override

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

int count=0;

for(Text v:values){

count++;

if(count<=5) {

context.write(new Text(key.getClassname()), new Text(v.toString() + "\t" + key.getSum()));

}

}

}

}

public static void main(String[] args) {

Configuration conf =new Configuration();

System.setProperty("HADOOP\_USER\_NAME", "qyl");

conf .set( "fs.defaultFS" , "hdfs://qyl02:9000" );

try {

Job job=Job.getInstance(conf);

job.setJarByClass(MyAllSource .class);

job.setMapperClass(MyMapper.class);

job.setReducerClass(MyReducer.class);

job.setMapOutputKeyClass(MyClassAndScore.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setGroupingComparatorClass(MyGroup2.class);

//指定需要统计的文件输入路径

Path inpath=new Path("/student/shuju");

FileInputFormat.addInputPath(job, inpath);

//指定输出目录 输出路径不能存在，否则就会报错 默认是覆盖式的输出

Path outpath=new Path("/student/result03");

if(outpath.getFileSystem(conf).exists(outpath)){

outpath.getFileSystem(conf).delete(outpath,true);

}

FileOutputFormat.setOutputPath(job, outpath);

job.waitForCompletion(true);

} catch (Exception e) {

e.printStackTrace();

}

}

}

 4)结果

1303 3001 谢雨泽 289

1303 3003 王凭 289

1303 3006 钟英杰 289

1303 3010 吕俊刚 289

1303 3013 曹娴瑶 289

1304 4021 张晓宸 293

1304 4054 谭凌云 291

1304 4028 宇佳杨 290

1304 4002 罗斐丹 289

1304 4053 雷磊 289

1305 5024 吴思妮 291

1305 5009 冯志超 289

1305 5053 曹能鑫 289

1305 5052 莫涟欢 289

1305 5046 马小雅 289

1306 6001 张轩铭 291

1306 6017 唐昕 289

1306 6002 邹明慧 289

1306 6043 李君清 289

1306 6042 罗天 289

1307 7007 刘俊辉 293

1307 7019 刘程望 291

1307 7006 苏新兴 289

1307 7001 邓思维 289

数据：由于数据量比较大，放入百度网盘中链接:

https://pan.baidu.com/s/13vHZ1v7Rw2Vbb5wZrWX0cA 提取码: 6qug