

第8章电影网站用户影评分析

(源自:<https://biglab.site>)

(版本:Ver1.0-20231110)

第8章电影网站用户影评分析

任务前言

学习目标

任务背景

软件准备

数据准备

任务8.1了解数据字段并分析需求

任务描述

了解数据字段

统计分析需求描述

任务8.2多维度分析用户影评

任务描述

创建并配置工程项目

计算评分次数最多的10部电影及评分次数并分析

s11连接movie.dat和ratings.dat数据

s12计算所有电影的评分次数

s13统计电影评分次数Top10

MovieRatesTop10Bean类

s13_MoviesRatesTop10类

计算不同性别评分最高的10部电影及评分并分析

s21连接movie.dat,users.dat,ratings.dat数据

s22按性别和电影分组计算每部电影影评的平均评分

MoviesRatesTop10GroupByGenderBean

s22_MoviesRatesAllGroupByGender

s23统计不同性别组内评分Top10的电影及评分信息

计算指定电影各年龄段的平均影评并分析

s31_MoviesAvgScore_GroupByAge

计算影评库中各种类型电影中评价最高的5部电影并分析

s41按类型和电影ID分组并计算每部电影影评的平均评分

s41_MoviesRatesAllGroupByType

s42计算影评库中各种类型电影中评价最高的5部电影并分析

s42_MoviesRatesTop5GroupByType

小结

任务前言

学习目标

1. 掌握根据业务场景设计map()方法和reduce()方法的计算逻辑
2. 掌握编写MapReduce程序解决常见的数据处理问题
3. 掌握编写MapReduce程序实现电影网站用户影评分析的方法

任务背景

1. 对电影的影评进行分析，可以从多维度了解一部电影的质量和受欢迎程度。
2. 常规的数据分析工具在大数据场景下，处理数据的效率低下，显然不适用于大数据处理分析。
3. 分布使用Hadoop分布式框架并结合电影评分数据，编写MapReduce程序实现用户影评分析，从多维度分析用户的观影兴趣偏好。
4. 分布式计算框架的出现，为分析处理大数据的计算提供了很好的解决方案。

软件准备

数据准备

通过百度网盘下载：

下载地址：链接: https://pan.baidu.com/s/1DHGRrnjvi4yMY41Se7_vDA 提取码: vbgu

软件	大小	版本	安装包称
hadoop_data	99.7M	20230925	hadoop_data.tar.gz

下载hadoop_data.tar.gz后，通过xftp工具上传到master主机的/root/hadoop目录下，然后xshell可crt进入master执行：

```
1 cd /root/hadoop
2 tar -zxvf ./hadoop_data.tar.gz
3 hdfs dfs -mkdir /user/myname/movie
4 hdfs dfs -put ./hadoop_data/movies.dat /user/myname/movie
5 hdfs dfs -put ./hadoop_data/ratings.dat /user/myname/movie
6 hdfs dfs -put ./hadoop_data/users.dat /user/myname/movie
```

任务8.1了解数据字段并分析需求

任务描述

进行用户观影兴趣偏好的数据分析之前，需要了解分析对象、数据字段的含义以及数据字段之间的关系。在明确数据字段的含义及其字段与字段之间可能存在的关系后，有助于提出科学的任务诉求，明确需求任务，因此本小节的任务是如下。了解数据字段。统计分析需求描述。

了解数据字段

电影网站提供了与用户信息相关的3份数据，分别为用户对电影的评分数据（ratings.dat）、已知性别的用户信息数据（users.dat）以及电影信息数据（movies.dat），3份数据的介绍说明如下表所示

ratings.dat表：

字段	说明
UserID	用户ID
MovieID	电影ID
Rating	评分
Timestamp	时间戳

users.dat表:

字段	说明
UserID	用户ID
Gender	性别
Age	年龄段
Occupation	职业
Zip-code	编码

movies.dat表:

字段	说明
MovieID	电影ID
Genres	电影类型

统计分析需求描述

通过对电影网站用户及电影评论数据进行分析，结合MapReduce编程知识，分别从评价次数、性别、年龄段、电影类型这4个维度分析用户的观影喜好，具体的统计分析需求如下。

1. 评价次数：计算评价次数最多的10部电影及评分次数。
2. 性别：计算不同性别评分最高的10部电影及评分。
3. 年龄段：计算某给定电影各年龄段的平均电影评分。
4. 电影类型：计算影评库中各种类型电影中评价最高的5部电影。

任务8.2多维度分析用户影评

任务描述

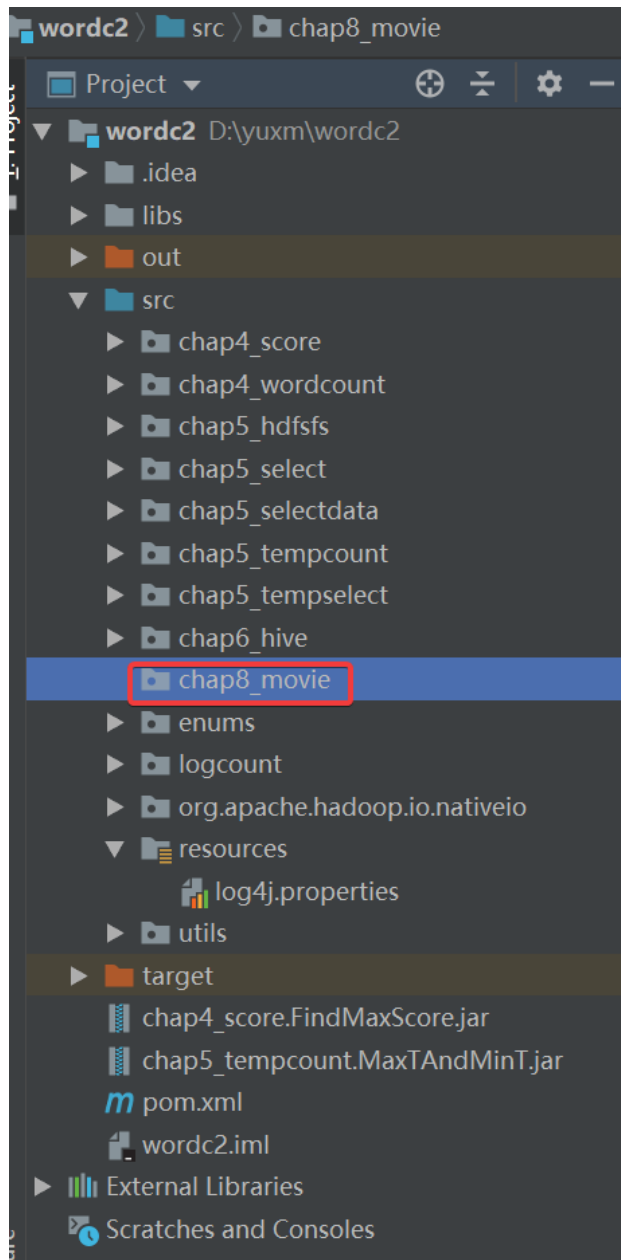
根据任务需求进行MapReduce编程实施方案。将分析需求整合在一个项目中完成，根据不同的分析任务进行任务分析，创建不同的Java类，将每个分析任务分解为若干小的统计任务，分步实现各影评分析任务，本小节任务如下。

1. 创建并配置工程项目。
2. 计算评分次数最多的10部电影及评分次数并分析。
3. 计算不同性别评分最高的10部电影及评分并分析。
4. 计算指定电影各年龄段的平均影评并分析。计

5. 算影评库中各种类型电影中评价最高的5部电影并分析。

创建并配置工程项目

1. 使用前面章节已创建的工程项目。
2. 在src目录下，添加包名为chap8_movie的目录



3. 在chap8_movie包目录下创建java类: Moveie_Join_Ratings、MoviesRatesAll等

计算评分次数最多的10部电影及评分次数并分析

s11连接movie.dat和ratings.dat数据

```
1 package chap8_movie.s1;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.fs.FSDataInputStream;
5 import org.apache.hadoop.fs.FSDataOutputStream;
6 import org.apache.hadoop.fs.FileSystem;
7 import org.apache.hadoop.fs.Path;
8 import org.apache.hadoop.io.LongWritable;
```

```

9 import org.apache.hadoop.io.NullWritable;
10 import org.apache.hadoop.io.Text;
11 import org.apache.hadoop.mapreduce.Job;
12 import org.apache.hadoop.mapreduce.Mapper;
13 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
14 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
15 import utils.ConfUtil;
16 import utils.FinalUtil;
17
18 import java.io.*;
19 import java.net.URI;
20 import java.net.URISyntaxException;
21 import java.util.HashMap;
22 public class s11_Movies_Join_Ratings {
23     public static void main(String[] args) throws Exception {
24         Configuration conf =
25         ConfUtil.GetConf(s11_Movies_Join_Ratings.class);
26         // Configuration conf = new Configuration();
27         FileSystem fs = FileSystem.get(conf);
28         Job job = Job.getInstance(conf); // 设置环境参数
29         job.setJarByClass(s11_Movies_Join_Ratings.class); // 设置整个程序的类
30         名
31         job.setMapperClass(Movies_Join_Ratings_Mapper.class); // 添加Mapper
32         类
33         job.setOutputKeyClass(Text.class); // 输出类型
34         job.setOutputValueClass(NullWritable.class); // 输出类型
35         Path inputPath = new Path(FinalUtil.MovieRatingInputPath); //
36         ratings.dat输入路径
37         Path outputPath = new Path(FinalUtil.MovieRatingOutputPath); //
38         ratings和movies连接后的输出路径
39         if (fs.exists(outputPath)) { // 判断, 如果输出路径存在, 那么将其删掉
40             fs.delete(outputPath, true);
41         }
42         FileInputFormat.setInputPaths(job, inputPath);
43         FileOutputFormat.setOutputPath(job, outputPath);
44         job.setNumReduceTasks(0); // 无Reduce任务
45         boolean isdone = job.waitForCompletion(true);
46         System.exit(isdone ? 0 : 1);
47         job.addCacheFile(new URI(FinalUtil.MovieMoviesInputPath));
48         //movies.dat的读取路径
49     }
50
51     public static class Movies_Join_Ratings_Mapper extends
52     Mapper<LongWritable, Text, Text, NullWritable> {
53         Text kout = new Text();
54         Text valueout = new Text();
55         // 执行map任务之前提前加载movies.dat,将movies.dat加载到movieMap中
56         private HashMap<String, String> movieMap = new HashMap<String,
57         String>();
58
59         @Override
60         protected void setup(Context context) throws IOException,
61         InterruptedException {
62             // FileReader fr = new
63             FileReader(FinalUtil.MovieMoviesInputPath);

```

```

54 //         BufferedReader br = new BufferedReader(fr);
55 //         String readLine = "";
56 //         while ((readLine = br.readLine()) != null) {
57 //             String[] reads = readLine.split("::");
58 //             String movieid = reads[0];
59 //             String movietype = reads[1];
60 //             movieMap.put(movieid, movietype);
61 //         }
62         Configuration conf=new Configuration();
63         FileSystem fs= null;
64         try {
65             fs = FileSystem.get(new
191 URI("hdfs://master:8020/"),conf,"root");
66             } catch (InterruptedException e) {
67                 throw new RuntimeException(e);
68             } catch (URISyntaxException e) {
69                 throw new RuntimeException(e);
70             }
71             //声明查看的路径
72             Path path=new Path(FinalUtil.MovieMoviesInputPath);
73             //获取指定文件的数据字节流
74             FSDataInputStream is=fs.open(path);
75             //读取文件内容并写入到新文件
76             BufferedReader br=new BufferedReader(new
192 InputStreamReader(is,"utf-8"));
77             String line="";
78             while((line=br.readLine())!=null){
79                 String[] reads = line.split("::");
80                 String movieid = reads[0];
81                 String movietype = reads[1];
82                 movieMap.put(movieid, movietype);
83             }
84             //关闭数据字节流
85             br.close();
86             is.close();
87             //关闭文件系统
88             fs.close();
89         }
90
91         @Override
92         protected void map(LongWritable key, Text value, Context context)
193 throws IOException, InterruptedException {
94             // 拿到一行数据将其转换成String类型
95             String line = value.toString().trim();
96             // 对原数据按::进行切分, 可取出各字段信息
97             String[] reads = line.split("::");
98             // 提取电影属性:1::1193::5::978300760
99             String userid = reads[0];
100            String movieid = reads[1];
101            int rate = Integer.parseInt(reads[2]);
102            long ts = Long.parseLong(reads[3]);
103            // 通过movieid 在movieMap中获取电影ID和电影类型
104            String moivetype = movieMap.get(movieid);
            // 将信息组合输出

```

```

105         String kk = userid + "::" + movieid + "::" + rate + "::" + ts +
        "::" + moivetype;
106         kout.set(kk);
107         context.write(kout, NullWritable.get());
108     }
109 }
110 }
111

```

s12计算所有电影的评分次数

```

1  package chap8_movie.s1;
2
3  import org.apache.hadoop.conf.Configuration;
4  import org.apache.hadoop.fs.FileSystem;
5  import org.apache.hadoop.fs.Path;
6  import org.apache.hadoop.io.LongWritable;
7  import org.apache.hadoop.io.NullWritable;
8  import org.apache.hadoop.io.Text;
9  import org.apache.hadoop.mapreduce.Job;
10 import org.apache.hadoop.mapreduce.Mapper;
11 import org.apache.hadoop.mapreduce.Reducer;
12 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14 import utils.ConfUtil;
15 import utils.FinalUtil;
16
17 import java.io.IOException;
18 public class s12_MoviesRatesAll {
19     public static void main(String[] args) throws Exception {
20 //         Configuration conf = new Configuration();
21         Configuration conf = ConfUtil.GetConf(s12_MoviesRatesAll.class);
22         FileSystem fs = FileSystem.get(conf);
23         Job job = Job.getInstance(conf);
24         job.setJarByClass(s12_MoviesRatesAll.class);
25         job.setMapperClass(MovieRatesAll_Mapper.class);
26         job.setReducerClass(MovieRatesAll_Reducer.class);
27         job.setMapOutputKeyClass(Text.class);
28         job.setMapOutputValueClass(Text.class);
29         job.setOutputKeyClass(Text.class);
30         job.setOutputValueClass(NullWritable.class);
31         Path inputPath = new Path(FinalUtil.MovieRatingOutputPath); // 将
        movies.dat和rating.dat连接后的结果目录作为输出目录"/join/output/"
32         Path outputPath = new Path(FinalUtil.MovieRatingAllOutputPath); //
        输出所有电影的评分次数到该目录下/join/outputAll/
33         if (fs.exists(outputPath)) {
34             fs.delete(outputPath, true);
35         }
36         FileInputFormat.setInputPaths(job, inputPath);
37         FileOutputFormat.setOutputPath(job, outputPath);
38         boolean isdone = job.waitForCompletion(true);
39         System.exit(isdone ? 0 : 1);
40     }
41     public static class MovieRatesAll_Mapper extends Mapper<LongWritable,
        Text, Text, Text> {

```

```

42     Text kout = new Text();
43     Text valueout = new Text();
44     @Override
45     protected void map(LongWritable key, Text value, Context
context)throws IOException, InterruptedException {
46         String [] reads = value.toString().trim().split("::");
47 // 用户id::电影id::评分::时间戳::电影类型
48         // 1::1193::5::978300760::One Flew Over the Cuckoo's Nest
(1975)::Drama
49         String kk = reads[1]; // 获取Movieid作为key输出
50         String vv = reads[4]; // 获取电影类型作为value值输出
51         kout.set(kk);
52         valueout.set(vv);
53         context.write(kout, valueout);
54     }
55 }
56 // 根据map阶段的结果<k:v>统计value的次数，存入rateNum中，即为某一电影的评分次数
57     public static class MovieRatesAllReducer extends Reducer<Text, Text,
Text, NullWritable> {
58         Text kout = new Text();
59         Text valueout = new Text();
60         @Override
61         protected void reduce(Text key, Iterable<Text> values, Context
context)throws IOException, InterruptedException {
62             int rateNum = 0;
63             String moiveType = "";
64             for(Text text : values){
65                 rateNum++;
66                 moiveType = text.toString();
67             }
68             String kk = key.toString() + "\t" + moiveType + "\t" + rateNum;
69             kout.set(kk);
70             context.write(kout, NullWritable.get());
71         }
72     }
73 }
74

```

s13统计电影评分次数Top10

MovieRatesTop10Bean类

```

1  package chap8_movie.s1;
2
3  import org.apache.hadoop.io.WritableComparable;
4  import java.io.DataInput;
5  import java.io.DataOutput;
6  import java.io.IOException;
7  public class MovieRatesTop10Bean implements
WritableComparable<MovieRatesTop10Bean>{
8      private int MovieYear;
9      private String Movietype;
10     private String Movieid;
11     private double RateNum;
12     public MovieRatesTop10Bean() {

```



```

13     }
14     public MovieRatesTop10Bean(String movietype, String id, double RateNum,
int year) {
15         this.Movietype = movietype;
16         this.Movieid = id;
17         this.RateNum = RateNum;
18         this.MovieYear = year;
19     }
20     public String getMovieId() {
21         return Movieid;
22     }
23     public void setMovieid(String id) {
24         this.Movieid = id;
25     }
26     public double getRateNum() {
27         return RateNum;
28     }
29     public void setRateNum(double RateNum) {
30         this.RateNum = RateNum;
31     }
32     public String getMovietype() {
33         return Movietype;
34     }
35     public void setMovietype(String movietype) {
36         this.Movietype = movietype;
37     }
38     public int getMovieYear() {
39         return MovieYear;
40     }
41     public void setMovieYear(int year) {
42         this.MovieYear = year;
43     }
44     public void write(DataOutput dataOutput) throws IOException {
45         dataOutput.writeUTF(this.Movietype);
46         dataOutput.writeUTF(this.Movieid);
47         dataOutput.writeDouble(this.RateNum);
48     }
49     public void readFields(DataInput dataInput) throws IOException {
50         this.Movietype = dataInput.readUTF();
51         this.Movieid = dataInput.readUTF();
52         this.RateNum = dataInput.readDouble();
53     }
54     public String toString() {
55         return "Scoringtimes{" +
56             "Movieid='" + Movieid + '\'' +
57             ", Movietype='" + Movietype + '\'' +
58             ", RateNum=" + RateNum +
59             '}';
60     }
61     public int compareTo(MovieRatesTop10Bean o) {
62         if (o.getRateNum()== this.RateNum) {
63             return o.getMovietype().compareTo(this.Movietype);
64         } else {
65             return o.getRateNum() > this.RateNum ? 1 : -1;
66         }

```

```

67     }
68 }
69

```

s13_MoviesRatesTop10类

```

1  package chap8_movie.s1;
2
3  import java.io.IOException;
4  import org.apache.hadoop.conf.Configuration;
5  import org.apache.hadoop.fs.FileSystem;
6  import org.apache.hadoop.fs.Path;
7  import org.apache.hadoop.io.LongWritable;
8  import org.apache.hadoop.io.NullWritable;
9  import org.apache.hadoop.io.Text;
10 import org.apache.hadoop.mapreduce.Job;
11 import org.apache.hadoop.mapreduce.Mapper;
12 import org.apache.hadoop.mapreduce.Reducer;
13 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
14 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
15 import utils.ConfUtil;
16 import utils.FinalUtil;
17
18 public class s13_MoviesRatesTop10 {
19     public static void main(String[] args) throws Exception {
20         Configuration conf = ConfUtil.GetConf(s12_MoviesRatesAll.class);
21         FileSystem fs = FileSystem.get(conf);
22         Job job = Job.getInstance(conf);
23         job.setJarByClass(s13_MoviesRatesTop10.class);
24         job.setMapperClass(MovieRatesTop10_Mapper.class);
25         job.setReducerClass(MovieRatesTop10_Reducer.class);
26         job.setOutputKeyClass(MovieRatesTop10Bean.class);
27         job.setOutputValueClass(NullWritable.class);
28         Path inputPath2 = new Path(FinalUtil.MovieRatingAllOutputPath); //
// 将所有电影的评分次数的输出目录作为统计top10的输入目录"/join/outputAll/"
29         Path outputPath2 = new Path(FinalUtil.MovieRatingTopOutputPath); //
// 输出电影评分次数top10"/join/outputTop10/"
30         if (fs.exists(outputPath2)) {
31             fs.delete(outputPath2, true);
32         }
33         FileInputFormat.setInputPaths(job, inputPath2);
34         FileOutputFormat.setOutputPath(job, outputPath2);
35         boolean isdone = job.waitForCompletion(true);
36         System.exit(isdone ? 0 : 1);
37     }
38     // reduce阶段不能实现排序，所以需要使用另一个MapReduce进行排序，取前10
39     public static class MovieRatesTop10_Mapper extends Mapper<LongWritable,
// Text, MovieRatesTop10Bean, NullWritable>{
40         Text kout = new Text();
41         Text valueout = new Text();
42         MovieRatesTop10Bean mrb = new MovieRatesTop10Bean();
43         @Override
44         protected void map(LongWritable key, Text value, Context
// context)throws IOException, InterruptedException {
45             String [] reads = value.toString().trim().split("\t");

```

```

46         // 1    Toy Story (1995)    2077
47         mrb.setMovieid(reads[0]);
48         mrb.setMovietype(reads[1]);
49         mrb.setRateNum(Integer.parseInt(reads[2]));
50         context.write(mrb, NullWritable.get()); //指定降序排序
51     }
52 }
53     public static class MovieRatesTop10_Reducer extends
Reducer<MovieRatesTop10Bean, NullWritable, MovieRatesTop10Bean,
NullWritable>{
54         Text kout = new Text();
55         Text valueout = new Text();
56         int count = 0;
57         @Override
58         protected void reduce(MovieRatesTop10Bean key,
Iterable<NullWritable> values, Context context) throws IOException,
InterruptedException {
59             // 1    Toy Story (1995)    2077
60             // 返回前10条记录
61             for(NullWritable inv : values){
62                 count ++;
63                 if (count <= 10) {
64                     context.write(key, NullWritable.get()); //取前10
65                 }else {
66                     return;
67                 }
68             }
69         }
70     }
71 }
72

```

计算不同性别评分最高的10部电影及评分并分析

s21连接movie.dat,users.dat,ratings.dat数据

```

1  package chap8_movie.s2;
2
3  import java.io.BufferedReader;
4  import java.io.FileReader;
5  import java.io.IOException;
6  import java.io.InputStreamReader;
7  import java.net.URI;
8  import java.net.URISyntaxException;
9  import java.util.HashMap;
10
11 import chap8_movie.s1.s12_MoviesRatesAll;
12 import org.apache.hadoop.conf.Configuration;
13 import org.apache.hadoop.fs.FSDataInputStream;
14 import org.apache.hadoop.fs.FileSystem;
15 import org.apache.hadoop.fs.Path;
16 import org.apache.hadoop.io.IOUtils;
17 import org.apache.hadoop.io.LongWritable;
18 import org.apache.hadoop.io.NullWritable;
19 import org.apache.hadoop.io.Text;

```

```

20 import org.apache.hadoop.mapreduce.Job;
21 import org.apache.hadoop.mapreduce.Mapper;
22 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
23 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
24 import utils.ConfUtil;
25 import utils.FinalUtil;
26
27 public class s21_MapjoinThreeTables {
28     public static void main(String[] args) throws Exception {
29         Configuration conf =
30         ConfUtil.GetConf(s21_MapjoinThreeTables.class);
31         FileSystem fs = FileSystem.get(conf);
32         Job job = Job.getInstance(conf);
33         job.setJarByClass(s21_MapjoinThreeTables.class);
34         job.setMapperClass(MapjoinThreeTables_Mapper.class);
35         job.setMapOutputKeyClass(Text.class);
36         job.setMapOutputValueClass(NullWritable.class);
37         Path inputPath = new Path(FinalUtil.MovieRatingInputPath); //
ratings.dat输入路径
38         Path outputPath = new Path(FinalUtil.MovieJoinTablesOutputPath); //
结果输出路径，无需创建，将自动生成
39         job.setNumReduceTasks(0); //无reduce任务
40         if (fs.exists(outputPath)) {
41             fs.delete(outputPath, true);
42         }
43         FileInputFormat.setInputPaths(job, inputPath);
44         FileOutputFormat.setOutputPath(job, outputPath);
45         boolean isdone = job.waitForCompletion(true);
46         System.exit(isdone ? 0 : 1);
47         // job.addCacheFile(new
URI("hdfs://master:8020/Tipdm/Hadoop/MapReduce/movies.dat")); // 需提前加载
至内存的movies.dat和users.dat的输入路径
48         // job.addCacheFile(new
URI("hdfs://master:8020/Tipdm/Hadoop/MapReduce/users.dat"));
49     }
50     public static class MapjoinThreeTables_Mapper extends
Mapper<LongWritable, Text, Text, NullWritable> {
51         Text kout = new Text();
52         Text valueout = new Text();
53         private static HashMap<String, String> movieMap = new
HashMap<String, String>();
54         private static HashMap<String, String> usersMap = new
HashMap<String, String>();
55
56         @SuppressWarnings("deprecation")
57         @Override
58         protected void setup(Context context) throws IOException,
InterruptedException {
59             // // 1::Toy Story (1995)::Animation|Children's|Comedy
60             // // 通过地址读取电影数据
61             // FileReader fr1 = new
FileReader("/opt/data/Hadoop/NO8/movies.dat");
62             // BufferedReader bf1 = new BufferedReader(fr1);
63             // String stringLine = null;

```

```

64 //         while ((stringLine = bf1.readLine()) != null) {
65 //             String[] reads = stringLine.split("::");
66 //             String movieid = reads[0];
67 //             String movieInfo = reads[1];
68 //             moviemap.put(movieid, movieInfo);
69 //         }
70 //         // 1::F::1::10::48067
71 //         // 通过地址读取用户数据
72 //         FileReader fr2 = new
FileReader("/opt/data/Hadoop/N08/users.dat");
73 //         BufferedReader bf2 = new BufferedReader(fr2);
74 //         String stringLine2 = null;
75 //         while ((stringLine2 = bf2.readLine()) != null) {
76 //             String[] reads = stringLine2.split("::");
77 //             String userid = reads[0];
78 //             String userInfo = reads[1] + "::" + reads[2] + "::" +
reads[3] + "::" + reads[4];
79 //             usersmap.put(userid, userInfo);
80 //         }
81 //         // 关闭资源
82 //         IOUtils.closeStream(bf1);
83 //         IOUtils.closeStream(bf2);
84
85         Configuration conf=new Configuration();
86         FileSystem fs= null;
87         try {
88             fs = FileSystem.get(new
URI("hdfs://master:8020/"),conf,"root");
89         } catch (InterruptedException e) {
90             throw new RuntimeException(e);
91         } catch (URISyntaxException e) {
92             throw new RuntimeException(e);
93         }
94         //声明查看的路径
95         Path path=new Path(FinalUtil.MovieMoviesInputPath);
96         //获取指定文件的数据字节流
97         FSDataInputStream is=fs.open(path);
98         //读取文件内容并写入到新文件
99         BufferedReader br=new BufferedReader(new
InputStreamReader(is,"utf-8"));
100         String line="";
101         while((line=br.readLine())!=null){
102             String[] reads = line.split("::");
103             String movieid = reads[0];
104             String movietype = reads[1];
105             movieMap.put(movieid, movietype);
106         }
107         //关闭数据字节流
108         br.close();
109         is.close();
110
111         //声明查看的路径
112         Path pathusers=new Path(FinalUtil.MovieUsersInputPath);
113         //获取指定文件的数据字节流
114         FSDataInputStream isusers=fs.open(pathusers);

```

```

115         //读取文件内容并写入到新文件
116         BufferedReader brusers=new BufferedReader(new
InputStreamReader(isusers,"utf-8"));
117         String lineuses="";
118         while((lineuses=brusers.readLine())!=null){
119             String[] reads = lineuses.split("::");
120             String userid = reads[0];
121             String userInfo = reads[1] + "::" + reads[2] + "::" +
reads[3] + "::" + reads[4];
122             usersMap.put(userid, userInfo);
123         }
124         //关闭数据字节流
125         brusers.close();
126         isusers.close();
127         //关闭文件系统
128         fs.close();
129     }
130
131     @Override
132     protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
133         String[] reads1 = value.toString().trim().split("::");
134         // 1::1193::5::978300760 :用户ID、电影ID、评分、评分时间戳
135         // 通过电影ID和用户ID在对应的map中获取信息, ratings不存在空信息, 如果存
在空信息, 那么需要进行map.contains判断
136         String struser = usersMap.get(reads1[0]);
137         String strmovie = movieMap.get(reads1[1]);
138         // 进行多表连接, 数据格式为userid、movieId、rate、ts、sex、age、
occupation、zipcode、movieType
139         String[] userinfo = struser.split("::");//sex, age, occupation,
zipcode
140         String kk = reads1[0] + "::" + reads1[1] + "::" + reads1[2] +
"::" + reads1[3] + "::"
141             + userinfo[0] + "::" + userinfo[1] + "::" + userinfo[2]
+ "::" + userinfo[3] + "::"
142             + strmovie;
143         kout.set(kk);
144         context.write(kout, NullWritable.get());
145     }
146 }
147 }
148

```

s22按性别和电影分组计算每部电影影评的平均评分

MoviesRatesTop10GroupByGenderBean

```

1 package chap8_movie.s2;
2
3 import java.io.DataInput;
4 import java.io.DataOutput;
5 import java.io.IOException;
6 import org.apache.hadoop.io.WritableComparable;
7

```

```

8 public class MoviesRatesTop10GroupByGenderBean implements
WritableComparable<MoviesRatesTop10GroupByGenderBean> {
9     private String sex;
10    private String mID;
11    private double rate;
12    public String getSex() {
13        return sex;
14    }
15    public void setSex(String sex) {
16        this.sex = sex;
17    }
18    public String getmID() {
19        return mID;
20    }
21    public void setmID(String mID) {
22        this.mID = mID;
23    }
24    public double getRate() {
25        return rate;
26    }
27    public void setRate(double rate) {
28        this.rate = rate;
29    }
30    public MoviesRatesTop10GroupByGenderBean(String sex, String mID, double
rate) {
31        super();
32        this.sex = sex;
33        this.mID = mID;
34        this.rate = rate;
35    }
36    public MoviesRatesTop10GroupByGenderBean() {
37        super();
38        // TODO Auto-generated constructor stub
39    }
40    @Override
41    public String toString() {
42        return sex + "\t" + mID + "\t" + rate;
43    }
44    public void write(DataOutput out) throws IOException {
45        out.writeUTF(sex);
46        out.writeUTF(mID);
47        out.writeDouble(rate);
48    }
49    public void readFields(DataInput in) throws IOException {
50        sex = in.readUTF();
51        mID = in.readUTF();
52        rate = in.readDouble();
53    }
54    public int compareTo(MoviesRatesTop10GroupByGenderBean o) {
55        int diff = this.sex.compareTo(o.sex);
56        double diff2 = this.rate - o.rate;
57        if (diff == 0) {
58            return diff2 > 0 ? -1 : 1;
59        }else {
60            return diff;

```

```

61     }
62     }
63 }
64

```

s22_MoviesRatesAllGroupByGender

```

1  package chap8_movie.s2;
2
3  import java.io.IOException;
4  import org.apache.hadoop.conf.Configuration;
5  import org.apache.hadoop.fs.FileSystem;
6  import org.apache.hadoop.fs.Path;
7  import org.apache.hadoop.io.DoubleWritable;
8  import org.apache.hadoop.io.LongWritable;
9  import org.apache.hadoop.io.Text;
10 import org.apache.hadoop.mapreduce.Job;
11 import org.apache.hadoop.mapreduce.Mapper;
12 import org.apache.hadoop.mapreduce.Reducer;
13 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
14 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
15 import utils.ConfUtil;
16 import utils.FinalUtil;
17
18 public class s22_MoviesRatesAllGroupByGender {
19     public static void main(String[] args) throws Exception {
20         Configuration conf = ConfUtil.GetConf(s21_MapjoinThreeTables.class);
21         FileSystem fs = FileSystem.get(conf);
22         Job job = Job.getInstance(conf);
23         job.setJarByClass(s22_MoviesRatesAllGroupByGender.class);
24         job.setMapperClass(MoviesRatesAllGroupByGender_Mapper.class);
25         job.setReducerClass(MoviesRatesAllGroupByGender_Reducer.class);
26         job.setMapOutputKeyClass(Text.class);
27         job.setMapOutputValueClass(Text.class);
28         job.setOutputKeyClass(Text.class);
29         job.setOutputValueClass(DoubleWritable.class);
30         Path inputPath = new Path(FinalUtil.MovieJoinTablesOutputPath);
31         Path outputPath = new
32 Path(FinalUtil.MovieRatesAllGroupByGenderOutputPath);
33         if (fs.exists(outputPath)) {
34             fs.delete(outputPath, true);
35         }
36         FileInputFormat.setInputPaths(job, inputPath);
37         FileOutputFormat.setOutputPath(job, outputPath);
38         boolean isdone = job.waitForCompletion(true);
39         System.exit(isdone ? 0 : 1);
40     }
41     public static class MoviesRatesAllGroupByGender_Mapper extends
42 Mapper<LongWritable, Text, Text, Text>{
43         Text kout = new Text();
44         Text valueout = new Text();
45         @Override
46         protected void map(LongWritable key, Text value, Context
47 context) throws IOException, InterruptedException {
48             String [] reads = value.toString().trim().split("::");

```



```

46         // 1::1193::5::978300760::F::1::10::48067::Drama
47         // 性别、电影ID、评分
48         String sex = reads[4];
49         String mID = reads[1];
50         int rate = Integer.parseInt(reads[2]);
51         // 每部电影的评分, 组内每部电影的总评分/每部电影的评分次数
52         // 按照性别和电影ID进行分组
53         String kk = sex + "\t" +mID;
54         String vv = reads[2];
55         kout.set(kk);
56         valueout.set(vv);
57         context.write(kout, valueout);
58     }
59 }
60 public static class MoviesRatesAllGroupByGender_Reducer extends
Reducer<Text, Text, Text, DoubleWritable>{
61     Text kout = new Text();
62     Text valueout = new Text();
63     @Override
64     protected void reduce(Text key, Iterable<Text> values, Context
context)throws IOException, InterruptedException {
65         int totalRate = 0; // 初始化总评分为0
66         int rateNum = 0; // 初始化总评分次数为0
67         double avgRate = 0; // 初始化每部电影的平均评分为0
68         for(Text text : values){ // 计算每部电影的总评分及评分次数
69             int rate = Integer.parseInt(text.toString());
70             totalRate += rate;
71             rateNum ++;
72         }
73         avgRate = 1.0 * totalRate / rateNum; // 计算每部电影的平均评分
74         DoubleWritable vv = new DoubleWritable(avgRate);
75         context.write(key, vv);
76     }
77 }
78 }
79

```

s23统计不同性别组内评分Top10的电影及评分信息

```

1 package chap8_movie.s2;
2
3 import org.apache.hadoop.conf.Configuration;
4 import org.apache.hadoop.fs.FileSystem;
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.LongWritable;
7 import org.apache.hadoop.io.NullWritable;
8 import org.apache.hadoop.io.Text;
9 import org.apache.hadoop.mapreduce.Job;
10 import org.apache.hadoop.mapreduce.Mapper;
11 import org.apache.hadoop.mapreduce.Reducer;
12 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
14 import utils.ConfUtil;
15 import utils.FinalUtil;
16

```

```

17 import java.io.IOException;
18 public class s23_MoviesRatesTop10GroupByGender {
19     public static void main(String[] args) throws Exception {
20         Configuration conf =
21         ConfUtil.GetConf(s23_MoviesRatesTop10GroupByGender.class);
22         FileSystem fs = FileSystem.get(conf);
23         Job job = Job.getInstance(conf);
24         job.setJarByClass(s23_MoviesRatesTop10GroupByGender.class);
25         job.setMapperClass(MoviesRatesTop10GroupByGender_2_Mapper.class);
26         job.setReducerClass(MoviesRatesTop10GroupByGender_2_Reducer.class);
27         job.setMapOutputKeyClass(MoviesRatesTop10GroupByGenderBean.class);
28         job.setMapOutputValueClass(NullWritable.class);
29         job.setOutputKeyClass(MoviesRatesTop10GroupByGenderBean.class);
30         job.setOutputValueClass(NullWritable.class);
31         job.setGroupingComparatorClass(GroupByGender.class); // 按性别设置分
组
32         Path inputPath = new
33         Path(FinalUtil.MovieRatesAllGroupByGenderOutputPath); // 将上一结果的输出路径作
为本次计算的数据输入
34         Path outputPath = new
35         Path(FinalUtil.MovieRatesTop10GroupByGenderOutputPath); // 设置本次计算结果的
输出路径
36         if (fs.exists(outputPath)) {
37             fs.delete(outputPath, true);
38         }
39         FileInputFormat.setInputPaths(job, inputPath);
40         FileOutputFormat.setOutputPath(job, outputPath);
41         boolean isdone = job.waitForCompletion(true);
42         System.exit(isdone ? 0 : 1);
43     }
44     public static class MoviesRatesTop10GroupByGender_2_Mapper extends
45     Mapper<LongWritable, Text, MoviesRatesTop10GroupByGenderBean, NullWritable>{
46         Text kout = new Text();
47         Text valueout = new Text();
48         MoviesRatesTop10GroupByGenderBean mrt = new
49     MoviesRatesTop10GroupByGenderBean();
50         @Override
51         protected void map(LongWritable key, Text value, Context
52     context) throws IOException, InterruptedException {
53             String [] reads = value.toString().trim().split("\t");
54             mrt.setSex(reads[0]);
55             mrt.setmID(reads[1]);
56             mrt.setRate(Double.parseDouble(reads[2]));
57             context.write(mrt, NullWritable.get());
58         }
59     }
60     public static class MoviesRatesTop10GroupByGender_2_Reducer extends
61     Reducer<MoviesRatesTop10GroupByGenderBean, NullWritable,
62     MoviesRatesTop10GroupByGenderBean, NullWritable>{
63         Text kout = new Text();
64         Text valueout = new Text();
65         @Override
66         protected void reduce(MoviesRatesTop10GroupByGenderBean key,
67     Iterable<NullWritable> values, Context context) throws IOException,
68     InterruptedException {

```

```

59         int count = 0;
60         // 求取前10
61         for(NullWritable inv : values){
62             count ++;
63             if (count <= 10) {
64                 context.write(key, NullWritable.get());
65             }else {
66                 return;
67             }
68         }
69     }
70 }
71 }
72

```

计算指定电影各年龄段的平均影评并分析

根据users.dat中数据的描述信息得知，字段Age并不是用户的真实年龄，而是年龄段。查看users.dat中的年龄段，该文件Age的取值共有7个，分别为0、1、2、3、4、5、6，分别表示7个年龄段，具体如下表所示。

Age	说明
0	18岁以下 (不包含18岁)
1	18~24岁
2	25~34岁
3	35~44岁
4	45~49岁
5	50~55岁
6	56岁及以上

s31_MoviesAvgScore_GroupByAge

```

1 package chap8_movie.s3;
2
3 import java.io.IOException;
4 import java.text.DecimalFormat;
5
6 import chap8_movie.s2.s21_MapjoinThreeTables;
7 import org.apache.hadoop.conf.Configuration;
8 import org.apache.hadoop.fs.FileSystem;
9 import org.apache.hadoop.fs.Path;
10 import org.apache.hadoop.io.LongWritable;
11 import org.apache.hadoop.io.Text;
12 import org.apache.hadoop.mapreduce.Job;
13 import org.apache.hadoop.mapreduce.Mapper;
14 import org.apache.hadoop.mapreduce.Reducer;

```

```

15 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
16 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
17 import utils.ConfUtil;
18 import utils.FinalUtil;
19
20 public class s31_MoviesAvgScore_GroupByAge {
21     public static void main(String[] args) throws Exception {
22         Configuration conf =
23         ConfUtil.GetConf(s31_MoviesAvgScore_GroupByAge.class);
24         FileSystem fs = FileSystem.get(conf);
25         Job job = Job.getInstance(conf);
26         job.setJarByClass(s31_MoviesAvgScore_GroupByAge.class);
27         job.setMapperClass(MovieAvgScore_GroupByAge_Mapper.class);
28         job.setReducerClass(MovieAvgScore_GroupByAge_Reducer.class);
29         job.setMapOutputKeyClass(Text.class);
30         job.setMapOutputValueClass(Text.class);
31         job.setOutputKeyClass(Text.class);
32         job.setOutputValueClass(Text.class);
33         Path inputPath = new Path(FinalUtil.MovieJoinTablesOutputPath); //
以3表连接的输出路径作为本次任务的输入路径
34         Path outputPath = new
Path(FinalUtil.MovieAvgScoreGroupByAgeOutputPath); // 设置输出路径
35         if (fs.exists(outputPath)) {
36             fs.delete(outputPath, true);
37         }
38         FileInputFormat.setInputPaths(job, inputPath);
39         FileOutputFormat.setOutputPath(job, outputPath);
40         boolean isdone = job.waitForCompletion(true);
41         System.exit(isdone ? 0 : 1);
42     }
43     public static class MovieAvgScore_GroupByAge_Mapper extends
Mapper<LongWritable, Text, Text, Text>{
44         Text kout = new Text();
45         Text valueout = new Text();
46         // 以求movieid = 2858这部电影各年龄段的平均影评
47         // userid, movieId, rate, ts, gender, age, occupation, zipcode,
movieType
48         // 用户ID、电影ID、评分、评分时间戳、性别、年龄段、职业、邮政编码、电影类型
49         @Override
50         protected void map(LongWritable key, Text value,Context
context)throws IOException, InterruptedException {
51             String [] reads = value.toString().trim().split("::");
52             String movieid = reads[1];
53             String age = reads[5];
54             String rate = reads[2];
55             if (movieid.equals("2858")) { // 判断电影id是否为2858, 进行过滤
56                 kout.set(age); // 输出k值为age
57                 valueout.set(rate + "\t" + movieid); // v值为电影评分和电影id
58                 context.write(kout, valueout); // 输出到reduce端
59             }
60         }
61     }
62     public static class MovieAvgScore_GroupByAge_Reducer extends
Reducer<Text, Text, Text, Text>{
63         Text kout = new Text();

```

```

63     Text valueout = new Text();
64     @Override
65     protected void reduce(Text key, Iterable<Text> values, Context
context)throws IOException, InterruptedException {
66         int totalRate = 0; // 初始化电影总评分
67         int rateNum = 0; // 初始化电影评论次数
68         double avgRate = 0; // 初始化平均评分
69         String movieid = "";
70         for(Text text : values){
71             String[] reads = text.toString().split("\t");
72             totalRate += Integer.parseInt(reads[0]);
73             rateNum ++; // 累加评分次数
74             movieid = reads[1]; // 仅仅为了验证一下
75         }
76         avgRate = 1.0 * totalRate / rateNum; // 计算电影平均评分
77         DecimalFormat df = new DecimalFormat("#.##"); // 设置评分格式
78         String string = df.format(avgRate);
79         String vv = string + "\t" +movieid; // 将电影平均评分与电影id连接
80         valueout.set(vv);
81         context.write(key, valueout);
82     }
83 }
84 }
85

```

计算影评库中各种类型电影中评价最高的5部电影并分析

s41按类型和电影ID分组并计算每部电影影评的平均评分

s41_MoviesRatesAllGroupByType

```

1  package chap8_movie.s4;
2
3  import chap8_movie.s3.s31_MoviesAvgScore_GroupByAge;
4  import org.apache.hadoop.conf.Configuration;
5  import org.apache.hadoop.fs.FileSystem;
6  import org.apache.hadoop.fs.Path;
7  import org.apache.hadoop.io.IntWritable;
8  import org.apache.hadoop.io.LongWritable;
9  import org.apache.hadoop.io.Text;
10 import org.apache.hadoop.mapreduce.Job;
11 import org.apache.hadoop.mapreduce.Mapper;
12 import org.apache.hadoop.mapreduce.Reducer;
13 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
14 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
15 import utils.ConfUtil;
16 import utils.FinalUtil;
17
18 import java.io.IOException;
19 import java.text.DecimalFormat;
20 public class s41_MoviesRatesAllGroupByType {
21     public static void main(String[] args) throws Exception {
22         Configuration conf =
ConfUtil.GetConf(s41_MoviesRatesAllGroupByType.class);
23         //求各类型的电影平均评分

```

```

24     FileSystem fs = FileSystem.get(conf);
25     Job job = Job.getInstance(conf);
26     job.setJarByClass(s41_MoviesRatesAllGroupByType.class);
27     job.setMapperClass(MoviesRatesAllGroupByType_Mapper.class);
28     job.setReducerClass(MoviesRatesAllGroupByType_Reducer.class);
29     job.setMapOutputKeyClass(Text.class);
30     job.setMapOutputValueClass(IntWritable.class);
31     job.setOutputKeyClass(Text.class);
32     job.setOutputValueClass(Text.class);
33     Path inputPath = new Path(FinalUtil.MovieJoinTablesOutputPath);
34     Path outputPath = new
Path(FinalUtil.MovieRatesAllGroupByTypeOutputPath);
35     if (fs.exists(outputPath)) {
36         fs.delete(outputPath, true);
37     }
38     FileInputFormat.setInputPaths(job, inputPath);
39     FileOutputFormat.setOutputPath(job, outputPath);
40     boolean isdone = job.waitForCompletion(true);
41     System.exit(isdone ? 0 : 1);
42 }
43
44     public static class MoviesRatesAllGroupByType_Mapper extends
Mapper<LongWritable, Text, Text, IntWritable> {
45         Text kout = new Text();
46         Text valueout = new Text();
47         @Override
48         protected void map(LongWritable key, Text value, Context
context) throws IOException, InterruptedException {
49             // 该影评库中各种类型电影中评价最高的5部电影（类型、电影ID、平均影评分）
50             // 用户ID、电影ID、评分、评分时间戳、性别、年龄、职业、邮政编码、电影类型
51             String [] reads = value.toString().trim().split("::");
52             String moiveID = reads[1];
53             int rate = Integer.parseInt(reads[2]);
54             String type = reads[8];
55             context.write(new Text(type + "\t" + moiveID), new
IntWritable(rate));
56         }
57     }
58     public static class MoviesRatesAllGroupByType_Reducer extends
Reducer<Text, IntWritable, Text, Text> {
59         Text kout = new Text();
60         Text valueout = new Text();
61         @Override
62         protected void reduce(Text key, Iterable<IntWritable> values,
Context context) throws IOException, InterruptedException {
63             int num = 0;
64             int total = 0;
65             double avg = 0;
66             for(IntWritable in : values){
67                 num ++;
68                 total += in.get();
69             }
70             avg = 1.0 * total / num;
71             DecimalFormat df = new DecimalFormat("#.##");
72             String format = df.format(avg);

```

```

73         context.write(key, new Text(format));
74     }
75 }
76 }
77

```

s42计算影评库中各种类型电影中评价最高的5部电影并分析

s42_MoviesRatesTop5GroupByType

```

1  package chap8_movie.s4;
2
3  import java.io.IOException;
4  import org.apache.hadoop.conf.Configuration;
5  import org.apache.hadoop.fs.FileSystem;
6  import org.apache.hadoop.fs.Path;
7  import org.apache.hadoop.io.LongWritable;
8  import org.apache.hadoop.io.NullWritable;
9  import org.apache.hadoop.io.Text;
10 import org.apache.hadoop.mapreduce.Job;
11 import org.apache.hadoop.mapreduce.Mapper;
12 import org.apache.hadoop.mapreduce.Reducer;
13 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
14 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
15 import utils.ConfUtil;
16 import utils.FinalUtil;
17
18 public class s42_MoviesRatesTop5GroupByType {
19     public static void main(String[] args) throws Exception {
20         Configuration conf =
21         ConfUtil.GetConf(s41_MoviesRatesAllGroupByType.class);
22         FileSystem fs = FileSystem.get(conf);
23         Job job = Job.getInstance(conf);
24         job.setJarByClass(s42_MoviesRatesTop5GroupByType.class);
25         job.setMapperClass(MoviesRatesTop5GroupByType_2_Mapper.class);
26         job.setReducerClass(MoviesRatesTop5GroupByType_2_Reducer.class);
27         job.setOutputKeyClass(MoviesRatesTop5GroupByTypeBean.class);
28         job.setOutputValueClass(NullWritable.class);
29         job.setGroupingComparatorClass(GroupByType.class); // 按电影类型进行分
30         组
31         Path inputPath2 = new
32         Path(FinalUtil.MovieRatesAllGroupByTypeOutputPath); // 数据输入路径
33         Path outputPath2 = new
34         Path(FinalUtil.MovieRatesTop5GroupByTypeOutputPath); // 输出目录
35         if (fs.exists(outputPath2)) {
36             fs.delete(outputPath2, true);
37         }
38         FileInputFormat.setInputPaths(job, inputPath2);
39         FileOutputFormat.setOutputPath(job, outputPath2);
40         boolean isdone = job.waitForCompletion(true);
41         System.exit(isdone ? 0 : 1);
42     }
43     public static class MoviesRatesTop5GroupByType_2_Mapper extends
44     Mapper<LongWritable, Text, MoviesRatesTop5GroupByTypeBean, NullWritable>{
45         Text kout = new Text();

```

```

41     Text valueout = new Text();
42     MoviesRatesTop5GroupByTypeBean mrb = new
MoviesRatesTop5GroupByTypeBean();
43     @Override
44     protected void map(LongWritable key, Text value, Context
context) throws IOException, InterruptedException {
45         String [] reads = value.toString().trim().split("\t");
46         mrb.setType(reads[0]);
47         mrb.setMID(reads[1]);
48         mrb.setNum(Double.parseDouble(reads[2]));
49         context.write(mrb, NullWritable.get());
50     }
51 }
52 public static class MoviesRatesTop5GroupByType_2_Reducer extends
Reducer<MoviesRatesTop5GroupByTypeBean, NullWritable,
MoviesRatesTop5GroupByTypeBean, NullWritable>{
53     Text kout = new Text();
54     Text valueout = new Text();
55     @Override
56     protected void reduce(MoviesRatesTop5GroupByTypeBean key,
Iterable<NullWritable> values, Context context) throws IOException,
InterruptedException {
57         int num = 0;
58         for(NullWritable in : values){
59             num ++;
60             if (num <= 5) {
61                 context.write(key, NullWritable.get());
62             }else {
63                 return;
64             }
65         }
66     }
67 }
68 }
69

```

小结

本章首先介绍了用户影评分析的背景及影评数据字段的含义，再根据影评数据从评价次数、性别、年龄段、电影类型这4个维度提出4个分析任务。

针对每个任务分别分析其计算过程，使用mapjoin()方法实现多数据连接，同时详细介绍了每个任务有关map()方法和reduce()方法的计算逻辑和数据类型要求，并且通过定义compareTo()方法，实现两个对象的比较和排序，最后通过MapReduce编程实现各分析任务。

通过电影网站用户硬盘分析的实例，可以加深对MapReduce框架的理解并熟悉MapReduce程序的编写逻辑。