第8章电影网站用户影评分析

(源自:https://biglab.site)

(版本:Ver1.0-20231110)

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：

```
cd /root/hadoop
tar -zxvf ./hadoop_data.tar.gz
hdfs dfs -mkdir /user/myname/movie
hdfs dfs -put ./hadoop_data/movies.dat  /user/myname/movie
hdfs dfs -put ./hadoop_data/ratings.dat  /user/myname/movie
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

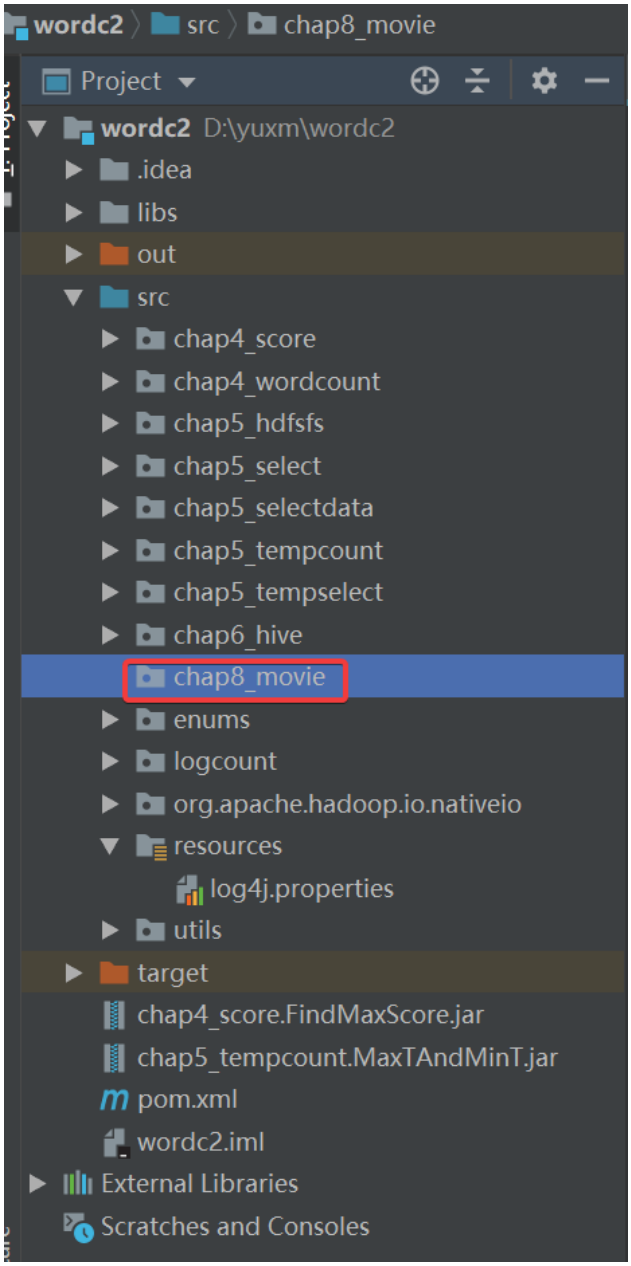
根据任务需求进行MapReduce编程实施方案。将分析需求整合在一个项目中完成，根据不同的分析任务进行任务分析，创建不同的Java类，将每个分析任务分解为若干小的统计任务，分步实现各影评分析任务，本小节任务如下。

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

```
package chap8_movie.s1;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
```

```
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import utils.ConfUtil;
import utils.FinalUtil;

import java.io.*;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.HashMap;
public class s11_Movies_Join_Ratings {
    public static void main(String[] args) throws Exception {
        Configuration conf =
ConfUtil.GetConf(s11_Movies_Join_Ratings.class);
//        Configuration conf = new Configuration();
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf);  // 设置环境参数
        job.setJarByClass(s11_Movies_Join_Ratings.class);  // 设置整个程序的类
名
        job.setMapperClass(Movies_Join_Ratings_Mapper.class);  // 添加Mapper
类
        job.setOutputKeyClass(Text.class);  // 输出类型
        job.setOutputValueClass(NullWritable.class);  // 输出类型
        Path inputPath = new Path(FinalUtil.MovieRatingInputPath);  //
ratings.dat输入路径
        Path outputPath = new Path(FinalUtil.MovieRatingOutputPath);  //
ratings和movies连接后的输出路径
        if (fs.exists(outputPath)) {  // 判断，如果输出路径存在，那么将其删掉
            fs.delete(outputPath, true);
        }
        FileInputFormat.setInputPaths(job, inputPath);
        FileOutputFormat.setOutputPath(job, outputPath);
        job.setNumReduceTasks(0);  // 无Reduce任务
        boolean isdone = job.waitForCompletion(true);
        System.exit(isdone ? 0 : 1);
        job.addCacheFile(new URI(FinalUtil.MovieMoviesInputPath));
//movies.dat的读取路径
    }

    public static class Movies_Join_Ratings_Mapper extends
Mapper<LongWritable, Text, Text, NullWritable> {
        Text kout = new Text();
        Text valueout = new Text();
        // 执行map任务之前提前加载movies.dat,将movies.dat加载到movieMap中
        private HashMap<String, String> movieMap = new HashMap<String,
String>();

        @Override
        protected void setup(Context context) throws IOException,
InterruptedException {
//            FileReader fr = new
FileReader(FinalUtil.MovieMoviesInputPath);
```

```
//            BufferedReader br = new BufferedReader(fr);
//            String readLine = "";
//            while ((readLine = br.readLine()) != null) {
//                String[] reads = readLine.split("::");
//                String movieid = reads[0];
//                String movietype = reads[1];
//                movieMap.put(movieid, movietype);
//            }
            Configuration conf=new Configuration();
            FileSystem fs= null;
            try {
                fs = FileSystem.get(new
URI("hdfs://master:8020/"),conf,"root");
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            } catch (URISyntaxException e) {
                throw new RuntimeException(e);
            }
            //声明查看的路径
            Path path=new Path(FinalUtil.MovieMoviesInputPath);
            //获取指定文件的数据字节流
            FSDataInputStream is=fs.open(path);
            //读取文件内容并写入到新文件
            BufferedReader br=new BufferedReader(new
InputStreamReader(is,"utf-8"));
            String line="";
            while((line=br.readLine())!=null){
                String[] reads = line.split("::");
                String movieid = reads[0];
                String movietype = reads[1];
                movieMap.put(movieid, movietype);
            }
            //关闭数据字节流
            br.close();
            is.close();
            //关闭文件系统
            fs.close();
        }

        @Override
        protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
            // 拿到一行数据将其转换成String类型
            String line = value.toString().trim();
            // 对原数据按::进行切分，可取出各字段信息
            String[] reads = line.split("::");
            // 提取电影属性:1::1193::5::978300760
            String userid = reads[0];
            String movieid = reads[1];
            int rate = Integer.parseInt(reads[2]);
            long ts = Long.parseLong(reads[3]);
            // 通过movieid 在movieMap中获取电影ID和电影类型
            String moivetype = movieMap.get(movieid);
            // 将信息组合输出
```

```
            String kk = userid + "::" + movieid + "::" + rate + "::" + ts +
 "::" + moivetype;
             kout.set(kk);
             context.write(kout, NullWritable.get());
         }
     }
}

```

s12计算所有电影的评分次数

```
package chap8_movie.s1;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import utils.ConfUtil;
import utils.FinalUtil;

import java.io.IOException;
public class s12_MoviesRatesAll {
    public static void main(String[] args) throws Exception {
//        Configuration conf = new Configuration();
        Configuration conf = ConfUtil.GetConf(s12_MoviesRatesAll.class);
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf);
        job.setJarByClass(s12_MoviesRatesAll.class);
        job.setMapperClass(MovieRatesAll_Mapper.class);
        job.setReducerClass(MovieRatesAll_Reducer.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(NullWritable.class);
        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/"
        Path outputPath = new Path(FinalUtil.MovieRatingAllOutputPath);  //
输出所有电影的评分次数到该目录下/join/outputAll/
        if (fs.exists(outputPath)) {
            fs.delete(outputPath, true);
        }
        FileInputFormat.setInputPaths(job, inputPath);
        FileOutputFormat.setOutputPath(job, outputPath);
        boolean isdone = job.waitForCompletion(true);
        System.exit(isdone ? 0 : 1);
    }
    public static class MovieRatesAll_Mapper extends Mapper<LongWritable,
Text, Text, Text> {
```

```
        Text kout = new Text();
        Text valueout = new Text();
        @Override
        protected void map(LongWritable key, Text value, Context
context)throws IOException, InterruptedException {
            String [] reads = value.toString().trim().split("::");
// 用户id::电影id::评分::时间戳::电影类型
            // 1::1193::5::978300760::One Flew Over the Cuckoo's Nest
(1975)::Drama
            String kk = reads[1];  // 获取Movieid作为key输出
            String vv = reads[4];  // 获取电影类型作为value值输出
            kout.set(kk);
            valueout.set(vv);
            context.write(kout, valueout);
        }
    }
// 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
    public static class MovieRatesAll_Reducer extends Reducer<Text, Text,
Text, NullWritable> {
        Text kout = new Text();
        Text valueout = new Text();
        @Override
        protected void reduce(Text key, Iterable<Text> values, Context
context)throws IOException, InterruptedException {
            int rateNum = 0;
            String moiveType = "";
            for(Text text : values){
                rateNum++;
                moiveType = text.toString();
            }
            String kk = key.toString() + "\t" + moiveType + "\t" + rateNum;
            kout.set(kk);
            context.write(kout, NullWritable.get());
        }
    }
}

```

s13统计电影评分次数Top10

MovieRatesTop10Bean类

```
package chap8_movie.s1;

import org.apache.hadoop.io.WritableComparable;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
public class MovieRatesTop10Bean implements
WritableComparable<MovieRatesTop10Bean>{
    private int MovieYear;
    private String Movietype;
    private String Movieid;
    private double RateNum;
    public MovieRatesTop10Bean() {
```

```
    }
    public MovieRatesTop10Bean(String movietype, String id, double RateNum,
int year) {
        this.Movietype = movietype;
        this.Movieid = id;
        this.RateNum = RateNum;
        this.MovieYear = year;
    }
    public String getMovieId() {
        return Movieid;
    }
    public void setMovieid(String id) {
        this.Movieid = id;
    }
    public double getRateNum() {
        return RateNum;
    }
    public void setRateNum(double RateNum) {
        this.RateNum = RateNum;
    }
    public String getMovietype() {
        return Movietype;
    }
    public void setMovietype(String movietype) {
        this.Movietype = movietype;
    }
    public int getMovieYear() {
        return MovieYear;
    }
    public void setMovieYear(int year) {
        this.MovieYear = year;
    }
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeUTF(this.Movietype);
        dataOutput.writeUTF(this.Movieid);
        dataOutput.writeDouble(this.RateNum);
    }
    public void readFields(DataInput dataInput) throws IOException {
        this.Movietype = dataInput.readUTF();
        this.Movieid = dataInput.readUTF();
        this.RateNum = dataInput.readDouble();
    }
    public String toString() {
        return "Scoringtimes{" +
                "Movieid='" + Movieid + '\'' +
                ", Movietype='" + Movietype + '\'' +
                ", RateNum=" + RateNum +
                '}';
    }
    public int compareTo(MovieRatesTop10Bean o) {
        if (o.getRateNum()== this.RateNum) {
            return o.getMovietype().compareTo(this.Movietype);
        } else {
            return o.getRateNum() > this.RateNum ? 1 : -1;
        }
```

```
    }
}

```

s13_MoviesRatesTop10类

```
package chap8_movie.s1;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import utils.ConfUtil;
import utils.FinalUtil;

public class s13_MoviesRatesTop10 {
    public static void main(String[] args) throws Exception {
        Configuration conf = ConfUtil.GetConf(s12_MoviesRatesAll.class);
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf);
        job.setJarByClass(s13_MoviesRatesTop10.class);
        job.setMapperClass(MovieRatesTop10_Mapper.class);
        job.setReducerClass(MovieRatesTop10_Reducer.class);
        job.setOutputKeyClass(MovieRatesTop10Bean.class);
        job.setOutputValueClass(NullWritable.class);
        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/"
        Path outputPath2 = new Path(FinalUtil.MovieRatingTopOutputPath);  //
输出电影评分次数top10"/join/outputTop10/"
        if (fs.exists(outputPath2)) {
            fs.delete(outputPath2, true);
        }
        FileInputFormat.setInputPaths(job, inputPath2);
        FileOutputFormat.setOutputPath(job, outputPath2);
        boolean isdone = job.waitForCompletion(true);
        System.exit(isdone ? 0 : 1);
    }
    // reduce阶段不能实现排序，所以需要在使用另一个MapReduce进行排序，取前10
    public static class MovieRatesTop10_Mapper extends Mapper<LongWritable,
Text, MovieRatesTop10Bean, NullWritable>{
        Text kout = new Text();
        Text valueout = new Text();
        MovieRatesTop10Bean mrb = new MovieRatesTop10Bean();
        @Override
        protected void map(LongWritable key, Text value,Context
context)throws IOException, InterruptedException {
            String [] reads = value.toString().trim().split("\t");

```

```
            // 1    Toy Story (1995)   2077
            mrb.setMovieid(reads[0]);
            mrb.setMovietype(reads[1]);
            mrb.setRateNum(Integer.parseInt(reads[2]));
            context.write(mrb, NullWritable.get());//指定降序排序
        }
    }
    public static class MovieRatesTop10_Reducer extends
Reducer<MovieRatesTop10Bean, NullWritable, MovieRatesTop10Bean,
NullWritable>{
        Text kout = new Text();
        Text valueout = new Text();
        int count = 0;
        @Override
        protected void reduce(MovieRatesTop10Bean key,
Iterable<NullWritable> values, Context context)throws IOException,
InterruptedException {
            // 1    Toy Story (1995)   2077
            // 返回前10条记录
            for(NullWritable inv : values){
                count ++;
                if (count <= 10) {
                    context.write(key, NullWritable.get());//取前10
                }else {
                    return;
                }
            }
        }
    }
}

```

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

```
package chap8_movie.s2;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.HashMap;

import chap8_movie.s1.s12_MoviesRatesAll;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IOUtils;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import utils.ConfUtil;
import utils.FinalUtil;

public class s21_MapjoinThreeTables {
    public static void main(String[] args) throws Exception {
        Configuration conf =
ConfUtil.GetConf(s21_MapjoinThreeTables.class);
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf);
        job.setJarByClass(s21_MapjoinThreeTables.class);
        job.setMapperClass(MapjoinThreeTables_Mapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(NullWritable.class);
        Path inputPath = new Path(FinalUtil.MovieRatingInputPath);  //
ratings.dat输入路径
        Path outputPath = new Path(FinalUtil.MovieJoinTablesOutputPath); //
结果输出路径，无需创建，将自动生成
        job.setNumReduceTasks(0);  //无reduce任务
        if (fs.exists(outputPath)) {
            fs.delete(outputPath, true);
        }
        FileInputFormat.setInputPaths(job, inputPath);
        FileOutputFormat.setOutputPath(job, outputPath);
        boolean isdone = job.waitForCompletion(true);
        System.exit(isdone ? 0 : 1);
//        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
//        job.addCacheFile(new
URI("hdfs://master:8020/Tipdm/Hadoop/MapReduce/users.dat"));
    }

    public static class MapjoinThreeTables_Mapper extends
Mapper<LongWritable, Text, Text, NullWritable> {
        Text kout = new Text();
        Text valueout = new Text();
        private static HashMap<String, String> movieMap = new
HashMap<String, String>();
        private static HashMap<String, String> usersMap = new
HashMap<String, String>();

        @SuppressWarnings("deprecation")
        @Override
        protected void setup(Context context) throws IOException,
InterruptedException {
//            // 1::Toy Story (1995)::Animation|Children's|Comedy
//            // 通过地址读取电影数据
//            FileReader fr1 = new
FileReader("/opt/data/Hadoop/NO8/movies.dat");
//            BufferedReader bf1 = new BufferedReader(fr1);
//            String stringLine = null;
```

```
//            while ((stringLine = bf1.readLine()) != null) {
//                String[] reads = stringLine.split("::");
//                String movieid = reads[0];
//                String movieInfo = reads[1];
//                moviemap.put(movieid, movieInfo);
//            }
//            // 1::F::1::10::48067
//            // 通过地址读取用户数据
//            FileReader fr2 = new FileReader("/opt/data/Hadoop/NO8/users.dat");
//            BufferedReader bf2 = new BufferedReader(fr2);
//            String stringLine2 = null;
//            while ((stringLine2 = bf2.readLine()) != null) {
//                String[] reads = stringLine2.split("::");
//                String userid = reads[0];
//                String userInfo = reads[1] + "::" + reads[2] + "::" + reads[3] + "::" + reads[4];
//                usersmap.put(userid, userInfo);
//            }
//            // 关闭资源
//            IOUtils.closeStream(bf1);
//            IOUtils.closeStream(bf2);

            Configuration conf=new Configuration();
            FileSystem fs= null;
            try {
                fs = FileSystem.get(new URI("hdfs://master:8020/"),conf,"root");
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            } catch (URISyntaxException e) {
                throw new RuntimeException(e);
            }
            //声明查看的路径
            Path path=new Path(FinalUtil.MovieMoviesInputPath);
            //获取指定文件的数据字节流
            FSDataInputStream is=fs.open(path);
            //读取文件内容并写入到新文件
            BufferedReader br=new BufferedReader(new InputStreamReader(is,"utf-8"));
            String line="";
            while((line=br.readLine())!=null){
                String[] reads = line.split("::");
                String movieid = reads[0];
                String movietype = reads[1];
                movieMap.put(movieid, movietype);
            }
            //关闭数据字节流
            br.close();
            is.close();

            //声明查看的路径
            Path pathusers=new Path(FinalUtil.MovieUsersInputPath);
            //获取指定文件的数据字节流
            FSDataInputStream isusers=fs.open(pathusers);
```

```
            //读取文件内容并写入到新文件
            BufferedReader brusers=new BufferedReader(new
InputStreamReader(isusers,"utf-8"));
            String lineuses="";
            while((lineuses=brusers.readLine())!=null){
                String[] reads = lineuses.split("::");
                String userid = reads[0];
                String userInfo = reads[1] + "::" + reads[2] + "::" +
reads[3] + "::" + reads[4];
                usersMap.put(userid, userInfo);
            }
            //关闭数据字节流
            brusers.close();
            isusers.close();
            //关闭文件系统
            fs.close();
        }

        @Override
        protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
            String[] reads1 = value.toString().trim().split("::");
            // 1::1193::5::978300760 :用户ID、电影ID、评分、评分时间戳
            // 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.contain判断
            String struser = usersMap.get(reads1[0]);
            String strmovie = movieMap.get(reads1[1]);
            // 进行多表连接，数据格式为userid、movieId、rate、ts、sex、age、
occupation、zipcode、movieType
            String[] userinfo = struser.split("::");//sex, age, occupation,
zipcode
            String kk = reads1[0] + "::" + reads1[1] + "::" + reads1[2] +
"::" + reads1[3] + "::"
                    + userinfo[0] + "::" + userinfo[1] + "::" + userinfo[2]
+ "::" + userinfo[3] + "::"
                    + strmovie;
            kout.set(kk);
            context.write(kout, NullWritable.get());
        }
    }
}

```

s22按性别和电影分组计算每部电影影评的平均评分

MoviesRatesTop10GroupByGenderBean

```
package chap8_movie.s2;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import org.apache.hadoop.io.WritableComparable;

```

```
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;
```

```
        }
    }
}

```

s22_MoviesRatesAllGroupByGender

```
package chap8_movie.s2;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import utils.ConfUtil;
import utils.FinalUtil;

public class s22_MoviesRatesAllGroupByGender {
public static void main(String[] args) throws Exception {
        Configuration conf = ConfUtil.GetConf(s21_MapjoinThreeTables.class);
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf);
        job.setJarByClass(s22_MoviesRatesAllGroupByGender.class);
        job.setMapperClass(MoviesRatesAllGroupByGender_Mapper.class);
        job.setReducerClass(MoviesRatesAllGroupByGender_Reducer.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);
        Path inputPath = new Path(FinalUtil.MovieJoinTablesOutputPath);
        Path outputPath = new
Path(FinalUtil.MovieRatesAllGroupByGenderOutputPath);
        if (fs.exists(outputPath)) {
            fs.delete(outputPath, true);
        }
        FileInputFormat.setInputPaths(job, inputPath);
        FileOutputFormat.setOutputPath(job, outputPath);
        boolean isdone = job.waitForCompletion(true);
        System.exit(isdone ? 0 : 1);
    }
    public static class MoviesRatesAllGroupByGender_Mapper extends
Mapper<LongWritable, Text, Text, Text>{
        Text kout = new Text();
        Text valueout = new Text();
        @Override
        protected void map(LongWritable key, Text value,Context
context)throws IOException, InterruptedException {
            String [] reads = value.toString().trim().split("::");

```

```
            // 1::1193::5::978300760::F::1::10::48067::Drama
            // 性别、电影ID、评分
            String sex = reads[4];
            String mID = reads[1];
            int rate = Integer.parseInt(reads[2]);
            // 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
            // 按照性别和电影ID进行分组
            String kk = sex + "\t" +mID;
            String vv = reads[2];
            kout.set(kk);
            valueout.set(vv);
            context.write(kout, valueout);
        }
    }
    public static class MoviesRatesAllGroupByGender_Reducer extends
Reducer<Text, Text, Text, DoubleWritable>{
        Text kout = new Text();
        Text valueout = new Text();
        @Override
        protected void reduce(Text key, Iterable<Text> values, Context
context)throws IOException, InterruptedException {
            int totalRate = 0;  // 初始化总评分为0
            int rateNum = 0;  // 初始化总评分次数为0
            double avgRate = 0;  // 初始化每部电影的平均评分为0
            for(Text text : values){  // 计算每部电影的总评分及评分次数
                int rate = Integer.parseInt(text.toString());
                totalRate += rate;
                rateNum ++;
            }
            avgRate = 1.0 * totalRate / rateNum;  // 计算每部电影的平均评分
            DoubleWritable vv = new DoubleWritable(avgRate);
            context.write(key, vv);
        }
}
}

```

s23统计不同性别组内评分Top10的电影及评分信息

```
package chap8_movie.s2;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import utils.ConfUtil;
import utils.FinalUtil;

```

```
import java.io.IOException;
public class s23_MoviesRatesTop10GroupByGender {
    public static void main(String[] args) throws Exception {
        Configuration conf =
ConfUtil.GetConf(s23_MoviesRatesTop10GroupByGender.class);
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf);
        job.setJarByClass(s23_MoviesRatesTop10GroupByGender.class);
        job.setMapperClass(MoviesRatesTop10GroupByGender_2_Mapper.class);
        job.setReducerClass(MoviesRatesTop10GroupByGender_2_Reducer.class);
        job.setMapOutputKeyClass(MoviesRatesTop10GroupByGenderBean.class);
        job.setMapOutputValueClass(NullWritable.class);
        job.setOutputKeyClass(MoviesRatesTop10GroupByGenderBean.class);
        job.setOutputValueClass(NullWritable.class);
        job.setGroupingComparatorClass(GroupByGender.class);  // 按性别设置分
组
        Path inputPath = new
Path(FinalUtil.MovieRatesAllGroupByGenderOutputPath);  // 将上一结果的输出路径作
为本次计算的数据输入
        Path outputPath = new
Path(FinalUtil.MovieRatesTop10GroupByGenderOutputPath);  // 设置本次计算结果的
输出路径
        if (fs.exists(outputPath)) {
            fs.delete(outputPath, true);
        }
        FileInputFormat.setInputPaths(job, inputPath);
        FileOutputFormat.setOutputPath(job, outputPath);
        boolean isdone = job.waitForCompletion(true);
        System.exit(isdone ? 0 : 1);
    }
    public static class MoviesRatesTop10GroupByGender_2_Mapper extends
Mapper<LongWritable, Text, MoviesRatesTop10GroupByGenderBean, NullWritable>{
        Text kout = new Text();
        Text valueout = new Text();
        MoviesRatesTop10GroupByGenderBean mrt = new
MoviesRatesTop10GroupByGenderBean();
        @Override
        protected void map(LongWritable key, Text value,Context
context)throws IOException, InterruptedException {
            String [] reads = value.toString().trim().split("\t");
            mrt.setSex(reads[0]);
            mrt.setmID(reads[1]);
            mrt.setRate(Double.parseDouble(reads[2]));
            context.write(mrt, NullWritable.get());
        }
    }
    public static class MoviesRatesTop10GroupByGender_2_Reducer extends
Reducer<MoviesRatesTop10GroupByGenderBean, NullWritable,
MoviesRatesTop10GroupByGenderBean, NullWritable>{
        Text kout = new Text();
        Text valueout = new Text();
        @Override
        protected void reduce(MoviesRatesTop10GroupByGenderBean key,
Iterable<NullWritable> values, Context context)throws IOException,
InterruptedException {
```

```
            int count = 0;
            // 求取前10
            for(NullWritable inv : values){
                count ++;
                if (count <= 10) {
                    context.write(key, NullWritable.get());
                }else {
                    return;
                }
            }
        }
    }
}

```

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

```
package chap8_movie.s3;

import java.io.IOException;
import java.text.DecimalFormat;

import chap8_movie.s2.s21_MapjoinThreeTables;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
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.Mapper;
import org.apache.hadoop.mapreduce.Reducer;

```

```
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import utils.ConfUtil;
import utils.FinalUtil;

public class s31_MoviesAvgScore_GroupByAge {
    public static void main(String[] args) throws Exception {
        Configuration conf =
ConfUtil.GetConf(s31_MoviesAvgScore_GroupByAge.class);
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf);
        job.setJarByClass(s31_MoviesAvgScore_GroupByAge.class);
        job.setMapperClass(MovieAvgScore_GroupByAge_Mapper.class);
        job.setReducerClass(MovieAvgScore_GroupByAge_Reducer.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        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
        Path outputPath = new
Path(FinalUtil.MovieAvgScoreGroupByAgeOutputPath);  // 设置输出路径
        if (fs.exists(outputPath)) {
            fs.delete(outputPath, true);
        }
        FileInputFormat.setInputPaths(job, inputPath);
        FileOutputFormat.setOutputPath(job, outputPath);
        boolean isdone = job.waitForCompletion(true);
        System.exit(isdone ? 0 : 1);
    }
    public static class MovieAvgScore_GroupByAge_Mapper extends
Mapper<LongWritable, Text, Text, Text>{
        Text kout = new Text();
        Text valueout = new Text();
        // 以求movieid = 2858这部电影各年龄段的平均影评
        // userid, movieId, rate, ts, gender, age, occupation, zipcode,
movieType
        // 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
        @Override
        protected void map(LongWritable key, Text value,Context
context)throws IOException, InterruptedException {
            String [] reads = value.toString().trim().split("::");
            String movieid = reads[1];
            String age = reads[5];
            String rate = reads[2];
            if (movieid.equals("2858")) {  // 判断电影id是否为2858，进行过滤
                kout.set(age);  // 输出k值为age
                valueout.set(rate + "\t" + movieid);  // v值为电影评分和电影id
                context.write(kout, valueout);  // 输出到reduce端
            }
        }
    }
    public static class MovieAvgScore_GroupByAge_Reducer extends
Reducer<Text, Text, Text, Text>{
        Text kout = new Text();
```

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

```

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

```
package chap8_movie.s4;

import chap8_movie.s3.s31_MoviesAvgScore_GroupByAge;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import utils.ConfUtil;
import utils.FinalUtil;

import java.io.IOException;
import java.text.DecimalFormat;
public class s41_MoviesRatesAllGroupByType {
public static void main(String[] args) throws Exception {
    Configuration conf =
ConfUtil.GetConf(s41_MoviesRatesAllGroupByType.class);
        //求各类型的电影平均评分
```

```
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf);
        job.setJarByClass(s41_MoviesRatesAllGroupByType.class);
        job.setMapperClass(MoviesRatesAllGroupByType_Mapper.class);
        job.setReducerClass(MoviesRatesAllGroupByType_Reducer.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        Path inputPath = new Path(FinalUtil.MovieJoinTablesOutputPath);
        Path outputPath = new
Path(FinalUtil.MovieRatesAllGroupByTypeOutputPath);
        if (fs.exists(outputPath)) {
            fs.delete(outputPath, true);
        }
        FileInputFormat.setInputPaths(job, inputPath);
        FileOutputFormat.setOutputPath(job, outputPath);
        boolean isdone = job.waitForCompletion(true);
        System.exit(isdone ? 0 : 1);
    }

    public static class MoviesRatesAllGroupByType_Mapper extends
Mapper<LongWritable, Text, Text, IntWritable> {
        Text kout = new Text();
        Text valueout = new Text();
        @Override
        protected void map(LongWritable key, Text value,Context
context)throws IOException, InterruptedException {
            // 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）
            // 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
            String [] reads = value.toString().trim().split("::");
            String moiveID = reads[1];
            int rate = Integer.parseInt(reads[2]);
            String type = reads[8];
            context.write(new Text(type + "\t" + moiveID), new
IntWritable(rate));
        }
    }
    public static class MoviesRatesAllGroupByType_Reducer extends
Reducer<Text, IntWritable, Text, Text> {
        Text kout = new Text();
        Text valueout = new Text();
        @Override
        protected void reduce(Text key, Iterable<IntWritable> values,
Context context)throws IOException, InterruptedException {
            int num = 0;
            int total = 0;
            double avg = 0;
            for(IntWritable in : values){
                num ++;
                total += in.get();
            }
            avg = 1.0 * total / num;
            DecimalFormat df = new DecimalFormat("#.#");
            String format = df.format(avg);
```

```
            context.write(key, new Text(format));
        }
    }
}

```

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

```
package chap8_movie.s4;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import utils.ConfUtil;
import utils.FinalUtil;

public class s42_MoviesRatesTop5GroupByType {
    public static void main(String[] args) throws Exception {
        Configuration conf =
ConfUtil.GetConf(s41_MoviesRatesAllGroupByType.class);
        FileSystem fs = FileSystem.get(conf);
        Job job = Job.getInstance(conf);
        job.setJarByClass(s42_MoviesRatesTop5GroupByType.class);
        job.setMapperClass(MoviesRatesTop5GroupByType_2_Mapper.class);
        job.setReducerClass(MoviesRatesTop5GroupByType_2_Reducer.class);
        job.setOutputKeyClass(MoviesRatesTop5GroupByTypeBean.class);
        job.setOutputValueClass(NullWritable.class);
        job.setGroupingComparatorClass(GroupByType.class);  // 按电影类型进行分
组
        Path inputPath2 = new
Path(FinalUtil.MovieRatesAllGroupByTypeOutputPath);  // 数据输入路径
        Path outputPath2 = new
Path(FinalUtil.MovieRatesTop5GroupByTypeOutputPath);  // 输出目录
        if (fs.exists(outputPath2)) {
            fs.delete(outputPath2, true);
        }
        FileInputFormat.setInputPaths(job, inputPath2);
        FileOutputFormat.setOutputPath(job, outputPath2);
        boolean isdone = job.waitForCompletion(true);
        System.exit(isdone ? 0 : 1);
    }
    public static class MoviesRatesTop5GroupByType_2_Mapper extends
Mapper<LongWritable, Text, MoviesRatesTop5GroupByTypeBean, NullWritable>{
        Text kout = new Text();
```

```
        Text valueout = new Text();
        MoviesRatesTop5GroupByTypeBean mrb = new
MoviesRatesTop5GroupByTypeBean();
        @Override
        protected void map(LongWritable key, Text value,Context
context)throws IOException, InterruptedException {
            String [] reads = value.toString().trim().split("\t");
            mrb.setType(reads[0]);
            mrb.setMID(reads[1]);
            mrb.setNum(Double.parseDouble(reads[2]));
            context.write(mrb, NullWritable.get());
        }
    }
    public static class MoviesRatesTop5GroupByType_2_Reducer extends
Reducer<MoviesRatesTop5GroupByTypeBean, NullWritable,
MoviesRatesTop5GroupByTypeBean, NullWritable>{
        Text kout = new Text();
        Text valueout = new Text();
        @Override
        protected void reduce(MoviesRatesTop5GroupByTypeBean key,
Iterable<NullWritable> values, Context context)throws IOException,
InterruptedException {
            int num = 0;
            for(NullWritable in : values){
                num ++;
                if (num <= 5) {
                    context.write(key, NullWritable.get());
                }else {
                    return;
                }
            }
        }
    }
}

```

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。