

**软件与微电子学院**

**大数据专题大作业报告**

|  |  |
| --- | --- |
| 项目名称： | 分布式电影推荐系统 |
| 姓 名： | 卢意、李延铖、高佳皓、岳晓琴、王景余 |
| 小 组： | 第八组 |
| Github： | <https://github.com/Luyi96/Bigdata_MovieRecommand> |
| 指导老师： | 张齐勋 |

二〇二〇 年 5 月 28日

### 项目介绍

随着经济社会的发展，人们对空闲时间的娱乐活动有了更高的追求，电影成为一种最普遍的娱乐方式。近年来，中国电影产量保持向上向好的发展态势，数据显示2018年我国电影产量总计1082部，全国院线观影人次到达17.16亿人。电影信息越来越多，观众人数也在不断增加，如何为每一位观众迅速找到适合自己的影片成为一个值得关注的问题，为此，电影推荐系统应运而生。顾名思义，电影推荐系统就是根据每一个用户的个人喜好、行为习惯等信息，结合其已有的观影记录，为每一个用户进行个性化的电影推荐。因为这种个性化的内容，系统的首页对于每一个用户来说都是不一样的。

本项目基于华为云平台，建立了一个包括多个节点的分布式电影推荐系统，项目设计数据采集与处理，推荐算法开发和前后端开发，涵盖Linux、MySQL、Hadoop、Spark、IntelliJ IDEA、Kettle、Node.js等系统和软件的安装与使用方法，采用Scala语言编写Spark程序。在系统初始阶段，每个用户对十个不同的电影评分，使用协同过滤推荐算法对不同的用户进行推荐，展示与其类型相似用户的高分电影。

### 系统架构设计

电影推荐系统采用三层框架设计，分别是表现层，服务层和数据层。数据层主要用来处理收集的数据，在我们的项目中，数据主要来源是电影评分数据集MovieLens，因此，数据层的主要功能是将数据集中的数据进行处理修改，并把数据存放到HDFS中。服务层的主要功能包括读取数据，用户查询，电影的管理与推荐，主要包括分布式大数据系统后端的Spark和MySQL开发。表现层的功能主要包括用户登录、注册、评分和观看推荐电影等功能，通过Node.js实现前端网页的开发。



图1 系统架构设计

### 环境搭建

本项目基于5台华为云主机搭建了5节点的hadoop集群和spark集群。

Hadoop搭建过程：

* 选定一台机器作为 master （namenode所在）
* 在 Master 节点上配置 hadoop 用户、安装 SSH server、安装 Java 环境
* 在 Master 节点上安装 Hadoop，并完成配置
* 在其他 Slave 节点上配置 hadoop 用户、安装 SSH server、安装 Java 环境
* 将 Master 节点上的 /usr/local/hadoop 目录复制到其他 Slave 节点上
* 在 Master 节点上开启 Hadoop

Spark集群搭建过程：

* 选定master节点，安装spark
* 配置环境变量
* Spark配置
* 将master中的spark打包复制到4个slave节点上
* 在master节点上启动集群

Mysql服务器搭建：

* 拉取MySQL镜像
* 创建并启动一个MySQL容器：sudo docker run --name pwc-mysql -e MYSQL\_ROOT\_PASSWORD=123456 -p 3306:3306 -d mysql

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 主机名称 | 内网ip | 外网ip | Hadoop节点 | Spark节点 |
| master | 192.168.0.15 | 121.36.165.184 | Namenode | Master |
| Slave1 | 192.168.0.97 | 121.36.154.4 | Datanode | Slave |
| Slave2 | 192.168.0.228 | 121.36.168.176 | Datanode | Slave |
| Slave3 | 192.168.0.116 | 121.36.131.144 | Datanode | Slave |
| Slave4 | 192.168.0.212 | 121.36.162.19 | Datanode | Slave |

表一：云服务器信息

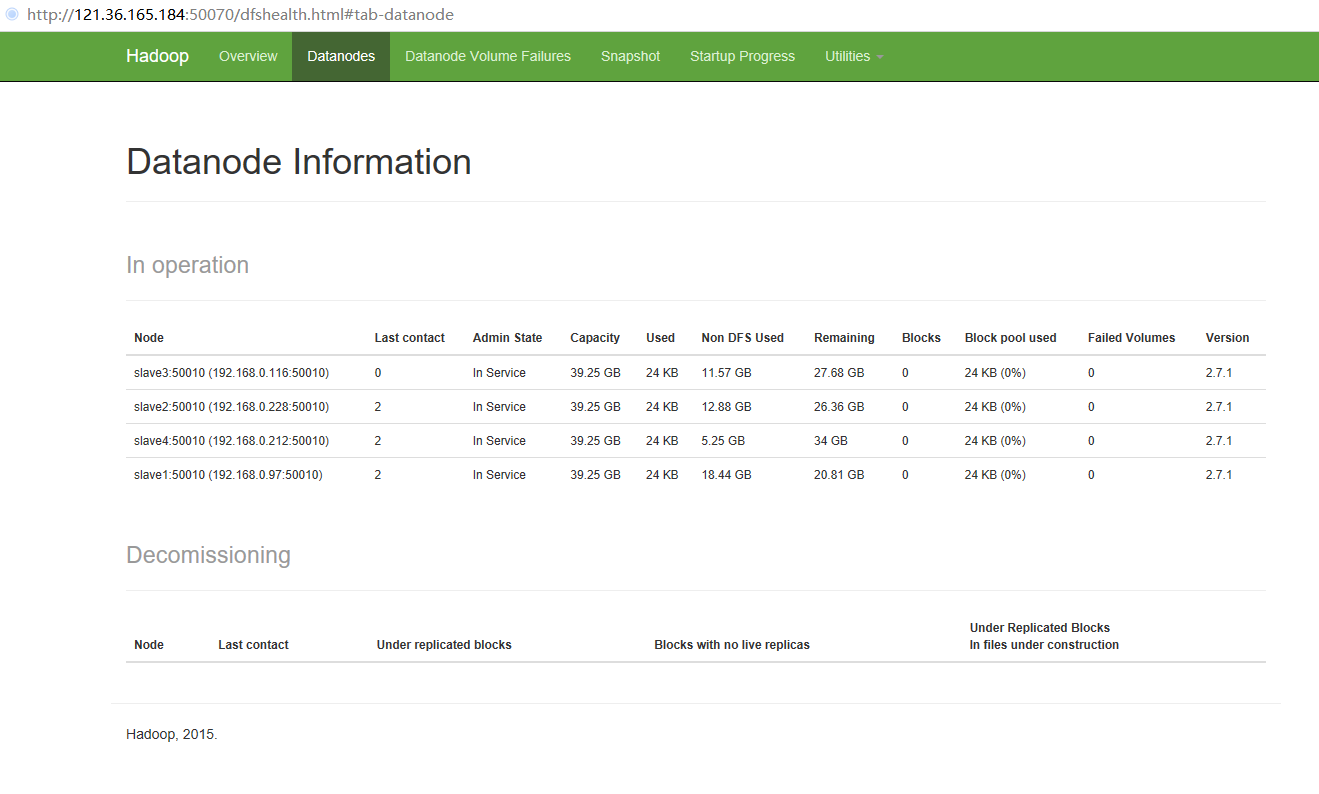


图4 分布式计算节点

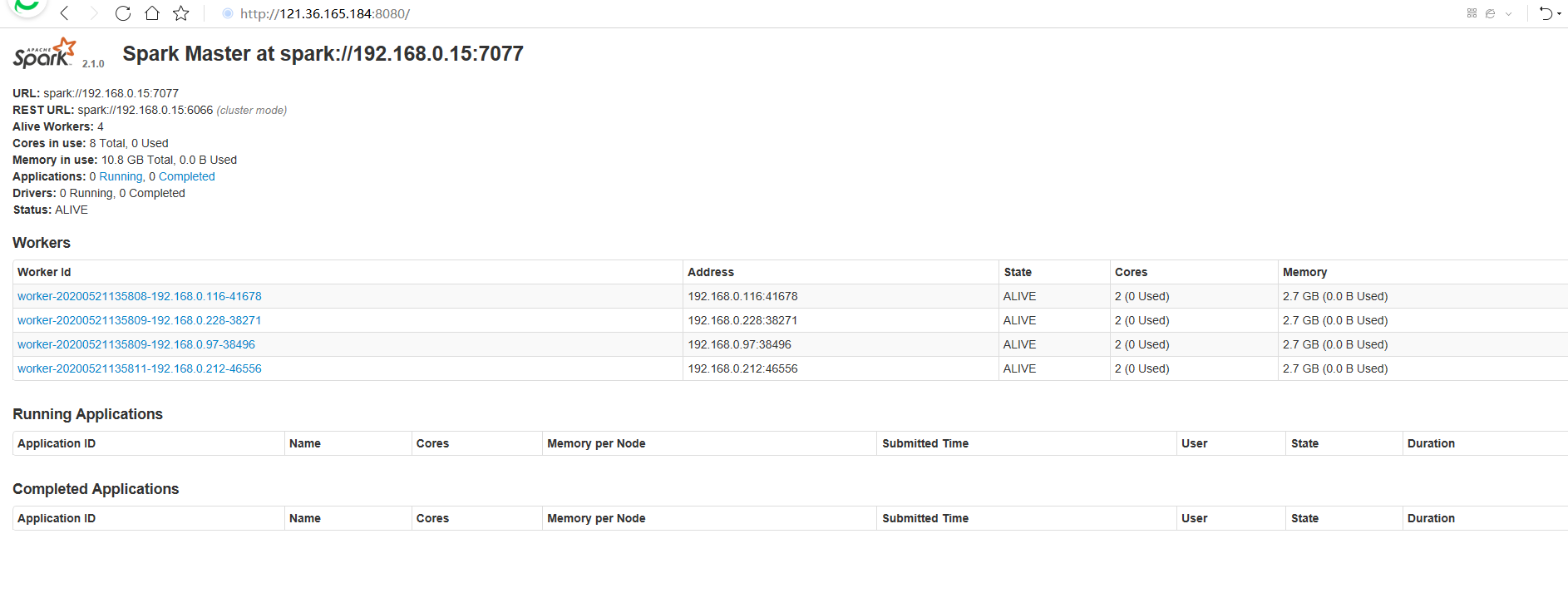


图5 Spark主节点信息

除此之外，我们编写了dockerfile，实现分布式系统的快速部署。只需在五台主机上：①执行构建镜像脚本，构建镜像；②创建docker网络；③启动容器；

### 数据采集

本项目采用MovieLens数据集，将数据集上传到华为用平台，数据集包含三个部分

* 用户评分数据集ratings.dat

1::661::3::978302109

1::914::3::978301968

1::3408::4::978300275

* 样本评分数据集personalRatings.txt

1::8::5::978300719

2::7::7::978302037

2::8::8::978300718

* 电影数据集movies.dat

1::Toy Story (1995)::Animation|Children's|Comedy

2::Jumanji (1995)::Adventure|Children's|Fantasy

3::Grumpier Old Men (1995)::Comedy|Romance

使用Kettle工具将数据ETL到HDFS中，首先建立一个用于存放数据集得HDFS目录input\_spark，分别把ratings.dat、personalRatings.txt、movies.dat、user.dat上传到HDFS的“input\_spark”目录中。

### 数据处理

在使用电影评分数据及进行电影推荐时，对数据集进行预处理，转换成满足需求的指定格式，在我们的项目中，主要是对电影名称进行修正，把movies.dat中电影名称中年份信息删除，主要包括以下3个步骤：

* 在HDFS中新建目录
* 使用ETL工具kettle清洗数据并加载到HDFS中
* 查看HDFS中的数据

将原始数据如下

1::Toy Story (1995)::Animation|Children's|Comedy

修改为

1::Toy Story::1995::Animation|Children's|Comedy

### 算法开发

1.使用基于用户的协同过滤算法，该算法符合人们对于“趣味相投”的任职，即兴趣相似的用户往往有相同的物品喜好，当目标用户需要个性化推荐时，可以先找到和目标用户有相似兴趣的用户群体，然后将这个用户群体喜欢的而目标用户没有听说过的物品推荐给目标用户，这种方法就称为“基于用户的协同滤波算法。”算法实现分为两个步骤。

1. 找到和目标用户兴趣相似的用户集合。
2. 找到该集合中的用户所喜欢的且目标用户没有听说过的物品推荐给目标用户。

假设有用户a,b,c和物品A,B,C,D,其中，用户a,c喜欢物品A,C，因此认为这两个用户是相似用户，于是将用户c喜欢的物品D推荐给用户a。



图2 基于用户的系统滤波算法示例

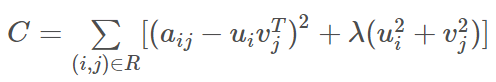
2.基于物品的协同滤波算法，思想和基于用户的一致，推荐那些和用户之前喜欢的物品类似的其他物品，计算方法也类似。

1. 计算物品之间的相似度。

2. 根据物品的相似度和用户的历史行为，给用户生成推荐列表。

**本项目采用一种基于模型的系统滤波算法**，使用Spark MLib中的基于ALS矩阵分解的协同滤波算法。其中用户对物品的打分行为可以表示成一个评分矩阵A(m\*n)，表示m个用户对n各物品的打分情况。但是，用户不会对所以物品打分，表中”？”表示用户没有打分的情况，所以这个矩阵A很多元素都是空的，我们称其为“缺失值（missing value）”。协同过滤提出了一种支持不完整评分矩阵的矩阵分解方法,不用对评分矩阵进行估值填充。在推荐系统中，我们希望得到用户对所有物品的打分情况，如果用户没有对一个物品打分，那么就需要预测用户是否会对该物品打分，以及会打多少分。这就是所谓的“矩阵补全（填空）”。

我们使用用户喜好特征矩阵中的第i个用户的特征向量，和产品特征矩阵第j个产品的特征向量来预测打分矩阵中的对应元素。我们可以得出一下的矩阵分解模型的损失函数为



Spark使用的是交叉最小二乘法（ALS）来最优化损失函数。算法的思想就是：我们随机生成然后固定它求解，再固定求解，这样交替进行下去，直到取得最优解。因为每步迭代都会降低误差，并且误差是有下界的，所以 ALS 一定会收敛。但由于问题是非凸的，ALS 并不保证会收敛到全局最优解。但在实际应用中，ALS 对初始点不是很敏感，是否全局最优解造成的影响并不大。

### 后端服务开发

本项目采用分布式的大数据推荐系统，包括1个master和4个slave，后端服务主要包括根据用户登陆，从数据库中选择该用户对应推荐电影目录。

**①用户登陆**：登录后,根据用户的ID去数据库的电影推荐表（Recommend表）寻找推荐的列表，如果列表为空，则用户是新用户，登录页面跳入电影评分页面，进行电影评分，如果列表不为空，则该用户是老用户，跳转到主页面。

**②推荐算法**：在用户点击提交按钮之后，页面将用户所输入的10部电影评分信息传入相关的JS代码块，进行数据库的插入操作（对数据库的Rating表进行评分数据的插入）。后台推荐算法取得页面用户ID之后，参数（userid）传递至MovieRecommendALS.jar（该jar是基于协同过滤算法的电影推荐）。

MovieRecommendALS.jar 的具体逻辑：

（1）解析传入的参数是否准确，错误类型返回错误信息，成功记录下来作为用户ID

（2）进行SparkConfig配置，创建SparkContext对象

（3）根据SparkContex对象，获取在Hadoop上的Rating.dat文件

（4）解析处理转换从Rating.dat得来的数据

（5）根据用户ID读取相关用户的MySQL数据库的Rating表里的信息

（6）合并处理过的Rating.dat数据与取得用户的数据

（7）创建ALS算法，定义相关参数的设置

（8）执行模型训练，取得ALS算法里的结果

（9）将结果插入Mysql数据库的Recommend表

最后，页面JS代码块根据用户ID取得mysql数据库的Recommend表信息，根据取得的电影ID去MySQL数据库检索出相关的电影详细信息，并显示在页面上。具体的实现过程如下：

1.新建Scala代码文件

装载样本评分数据，其中最后一列Timestamp取除10的余数作为key，Rating为值,即(Int,Rating)，其中ratings.dat原始数据：用户编号、电影编号、评分、评分时间戳。

2.装载电影目录对照表（电影ID->电影标题），其中movies.dat原始数据：电影编号、电影名称、电影类别。

将样本评分表以key值切分成3个部分，分别用于训练 (60%，并加入用户评分), 校验 (20%), and 测试 (20%)。

训练不同参数下的模型，并在校验集中验证，获取最佳参数下的模型

1. 修改pom.xml文件
2. 生成应用程序JAR包

为了能够把应用程序部署到Spark环境中运行，需要使用IDEA工具对程序进行打包，生成应用程序JAR包。

1. 把JAR包提交到Spark中运行

运行Spark\_Recommend.jar程序，包括5个参数，5个参数，其中，第1个参数“/input\_spark”是HDFS文件系统中的目录，该目录下包含了两个文件movies.dat和ratings.dat（如果不存在该目录和文件，请使用HDFS命令创建目录并上传数据文件）。第2个参数是personalRatings.dat文件路径（这里是一个放在Linux本地文件系统中的文件，没有存放在HDFS中），第3、4和5个参数分别是隐语义因子个数、ALS正则化参数、迭代次数，运行结果如下。

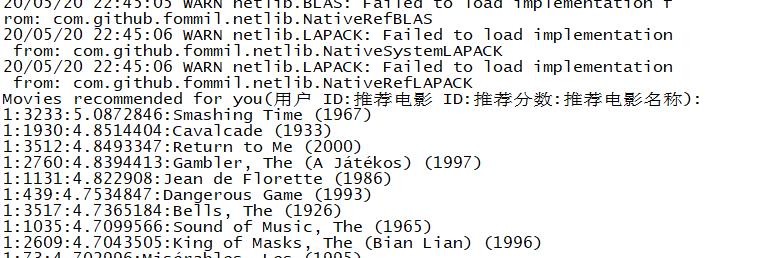
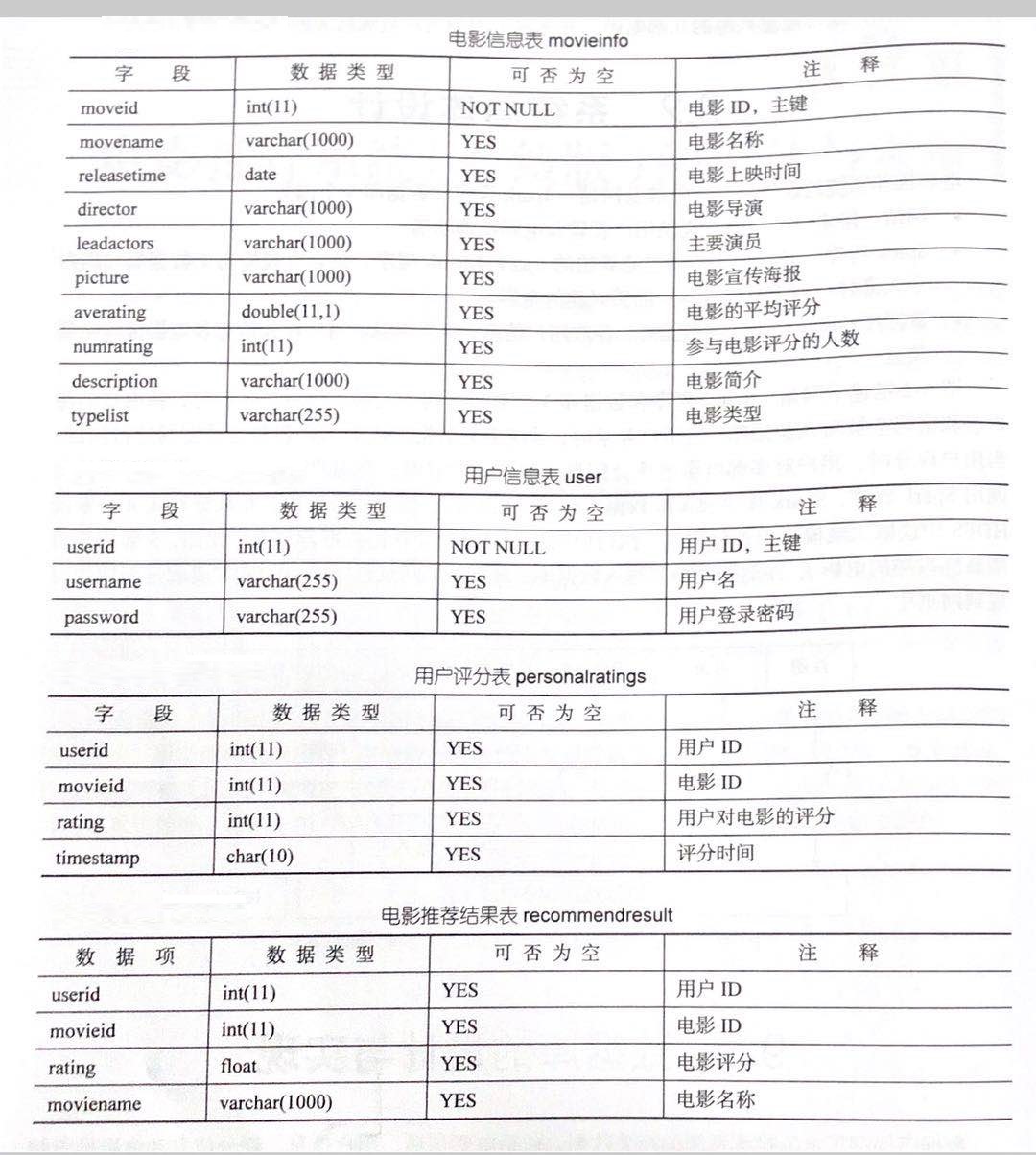


图3 Spark推荐系统计算结果

1. 需要注意的是，ALS（最小交替二乘法）算法里面包含了随机的过程，因此，每次运行Spark\_Recommend程序，得到的推荐结果不会完全相同。

③数据库设计

在mysql中创建了一个名称为“movierecommend”的数据库，再在此数据库中创建了4个表，即电影信息表movieinfo、用户信息表user、用户评分表personalratings以及电影推荐结果表recommendresult。其中movieinfo表中插入了电影评分的训练数据集。



### 前端开发

使用Node.js在网页中展现结果，前端的主要功能包括用户的注册、登陆、评分和推荐，登录页面包括2个输入框（用户名、密码），一个按钮（登录按钮）以及一个注册页面的链接。界面如图6所示。



图6 登录界面

注册页面主要包括3个输入框（用户名、密码、确认密码），一个按钮（注册按钮）以及一个去登录页面的链接。界面如图7所示。

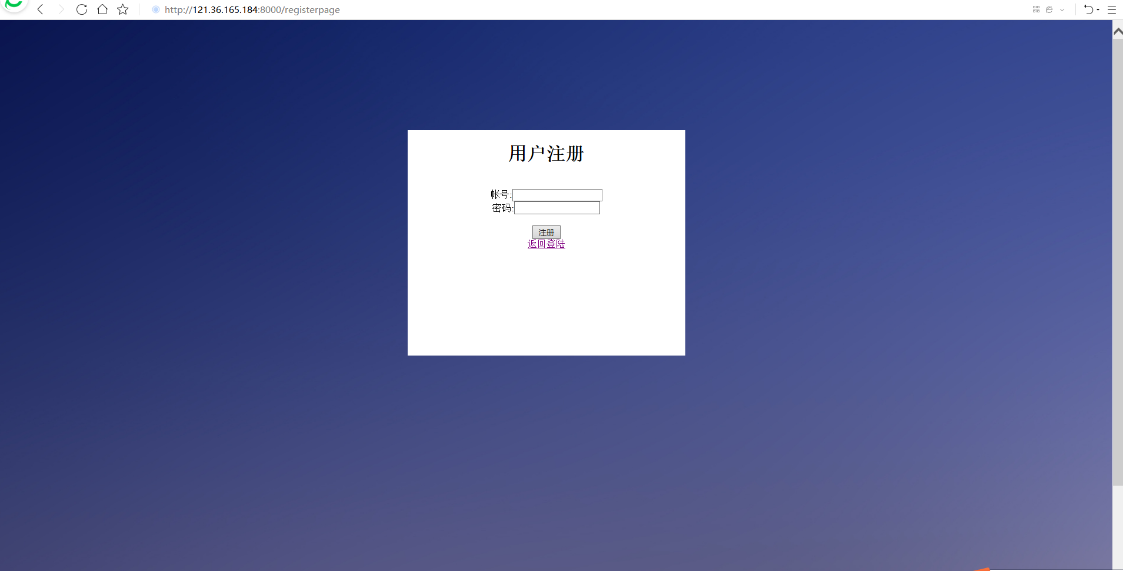


图7 登录界面

首页是未登录前的首页，主要就是含有登录、注册的入口。界面如图8所示。



图8 推荐系统首页

评分页面，主要就是向登录的用户显示电影，然后让用户自行选择电影进行评分，点击提交后跳转推荐页面。评分页面如图9所示。

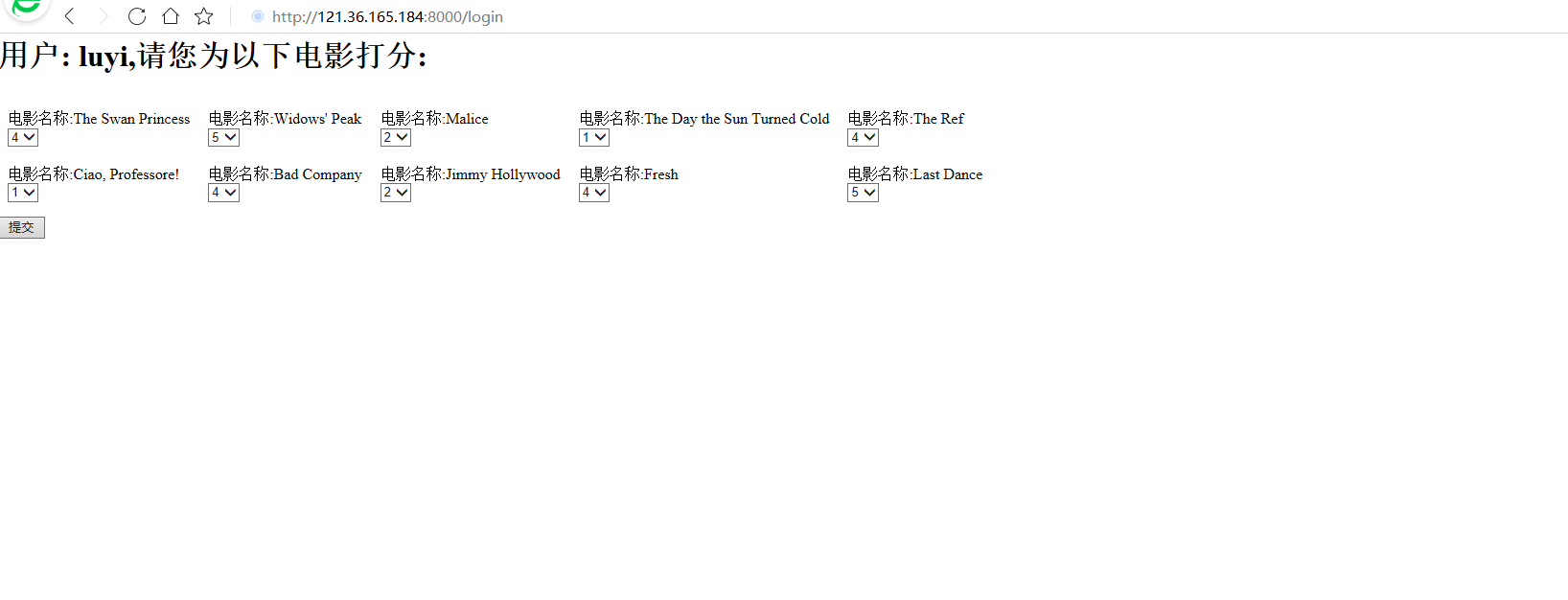


图9 评分页面

推荐页面是针对评分页面后的跳转页面，主要是显示用户评分提交后经过系统运行后的推荐电影。页面如图10所示。



图10 推荐页面

### 系统集成

综上所述，本系统的可分为用户管理、电影推荐两个主要部分，具体描述如下

（1）用户管理：用户可以进行登录和注册操作。

登录：用户登录系统。

注册：新用户注册成为系统用户。

（2）电影推荐：已登录的用户可在系统中进行电影评分操作和获得电影推荐服务。

电影评分：针对首次登录用户可以选择十部电影进行评分。

成功推荐：针对已参与电影评分的用户，系统会对用户进行相应的电影推荐。

前端页面通过MySQL与后端数据库交互，获得用户信息和对应的电影条目，利用Spark实现实时的推荐计算，与数据的读取和存储，最终将推荐结果显示在前端网页上。



图11 系统实现过程