# 摘 要

随着互联网技术的迅速发展，信息爆炸、信息过剩是我们日益面临着的严峻问题。为了提高信息的利用率，推荐系统应运而生。但是随着推荐系统的发展，推荐方式的时效性和准确性要求越来越高。

传统推荐系统大多通过对数据进行定期分析，然后更新推荐模型。这就使得推荐无法保证其实时性，推荐的结果不能反映当前的状态，造成结果不够精准。本系统基于流式大数据实时处理框架Storm，利用其可扩展性、容错性、保证消息被处理等特点。可以使得用户的请求能够得到有效处理并将推荐算法推荐出的音乐实时发送到用户前端以减少用户的等待时间。

实时推荐系统通过使用HBase(Hadoop Database)和Redis作为存储提供推荐系统中的计算结果和推荐列表。使用Bootstrap框架编写用户交互界面，具有良好的浏览器兼容性和用户友好性。

测试结果表明，本文实现的推荐系统在推荐系统评测指标用户满意度、预测准确度、实时性、健壮性上，都达到了设计要求，能在音乐推荐中发挥积极的作用。

**关键词：实时推荐 Storm计算框架 推荐系统 实时计算**

# **ABSTRACT**

With the rapid development of Internet technology, the information explosion, information overload is we are increasingly faced with a serious problem. In order to improve the utilization of information, the recommendation system came into being. But with the development of the recommendation system, the recommended way to the timeliness and accuracy of the increasingly high demand.

Traditional systems are mostly recommended by periodic analysis of the data, and then update the recommended model. This makes the recommendation can not guarantee when in fact, recommended the results do not reflect the current state, causing the results were not accurate. The system is based on streaming real-time big data processing framework Storm, with its scalability, fault tolerance, to ensure that messages are processed and so on. It can make the user's request can be effectively treated and recommendation algorithm recommended by the music sent in real time to the user front-end in order to reduce waiting time for users.

Real-time recommendation system by using HBase (Hadoop Database) and provide recommendations Redis as storage system calculations and recommendation list. Use Bootstrap framework for writing user interface, has a good browser compatibility and user-friendliness.

The test results show that the realization of the system recommended in the recommended evaluation index system user satisfaction, forecast accuracy, real-time, on robustness, have reached the design requirements, it can play an active role in the music recommendation.

**Keyword:Real-time recommendation, Storm real-time computation framework, Recommendation system,Real-time calculation**

# 目录

摘 要 1

ABSTRACT 2

目录 i

第一章 绪论 1

1.1 研究的背景和意义 1

1.2 国内外研究现状 1

1.3 本文研究内容 2

1.4 设计目标 2

1.5 论文的组织架构 2

第二章 系统概述 4

2.1 推荐系统 4

2.2 实时计算框架Storm 5

2.3 数据库 5

2.3.1 Mysql数据库 5

2.3.2 Redis内存数据库 6

2.3.3 HBase数据库 6

2.4 WebSocket协议 6

2.5 本章小结 6

第三章 系统设计 7

3.1 设计目标 7

3.2 系统总体架构设计 7

3.3 音乐推荐算法设计 9

3.4 本章小结 9

第四章 系统实现 10

4.1 VirtualBox与操作系统 10

4.1.1 简要介绍 10

4.1.2 安装 11

4.2 Storm集群搭建 11

4.3 前端页面编写 14

4.4 Storm拓扑实现 18

4.4.1 拓扑处理逻辑 18

4.1.2 实时处理步骤 18

4.1.3 具体代码实现 19

4.5 测试用例编写 20

4.5.1 Junit 21

4.6 本章小结 23

第五章 系统测试 24

5.1 网页前端测试 24

5.2 推荐系统测试 25

5.3 本章小结 28

第六章 总结与展望 29

6.1 工作总结 29

6.2 未来展望 29

致谢 31

参考文献 32

# 第一章 绪论

## 1.1 研究的背景和意义

如今已然进入信息时代，大数据时代已经到来。互联网的迅速发展带来信息过载、数据 量巨大问题。我们每天不得不接受大量的信息，而从中找出我们需要的、对自己有价值的数 据却是不容易的。基于这种情况，实时推荐系统应运而生，解决在信息爆炸时代下的信息过 载问题。为用户过滤信息，呈现给用户有用的部分，从而提高信息使用率。和搜索引擎不同， 搜索引擎是用户主动去寻找信息，实时推荐是可能用户不知道自己需要什么信息，而通过用 户的个人信息、浏览历史、偏好等信息，推荐给用户感兴趣的信息。帮助用户快速发现感兴 趣和高质量的信息，提升用户体验。增加用户使用产品时间。减少用户浏览到重复或者厌恶 的信息带来的不利影响。提供个性化信息，信息的推荐更为精准。 数据作为一种基础资源，可以产生巨大价值。实时推荐系统可以提高信息的使用率，让数据发挥价值，实时推荐系统不同于传统的推荐系统，传统的推荐系统不是实时的，可能用 户原来感兴趣，但是现在已经不感兴趣了，传统的推荐系统的推荐具有过时性。实时推荐系 统，根据用户的实时行为，采用流式数据处理框架 Storm 进行实时处理，然后再把处理结果 实时反馈给用户，达到实时推荐的目的，可以更好的服务用户，达到推荐精确化。

## 1.2 国内外研究现状

早在 1992 年，施乐的科学家为了解决信息负载的问题，第一次提出协同过滤算法，推 荐系统已经以迅雷不及掩耳之势快速发展了二十几年。在电商平台上，国外的亚马逊、 eBay 等在线购物网站很早都在研究并使用，国内的阿里、京东、苏宁等购物网站也在研究和使用， 推荐系统提高了平台的交易额，增加了收益。 目前，实时分布式流计算对于一个互联网公司是至关重要的，所以国内外很多公司都在 就行研究。实时分布流计算具有低延迟、高可靠的特点。目前国内外已有的流处理框架有： 1、 S4 是由 Yahoo 开发并于 2010 年 10 月开源的分布式流数据处理平台。 S4 具有可扩展、部分容错、可插拔等特性。 S4 主要是为了方便开发者开发处理流式处理程序。 2、 Storm 是由 BackType 开发并被 Twitter 与 2011 开源的分布式实时计算系统。 Storm 为了方便计算持续不断的实时流数据而设计，弥补了 Hadoop 批处理所不能满足的实时要求。 3、 Spark 是 UC Berkeley AMP lab 开源的类 Hadoop MapReduce 的通用并行计算框架，是一 个高效的分布式计算系统，将中间输出和结果保存在内存中，具有更高的性能。

## 1.3 本文研究内容

本文围绕实时推荐系统展开分析研究，采取与实时流式计算框架 Storm 结合的方法，比较现有推荐系统和传统推荐系统的优缺点，在此基础上，研究设计出一种基 于 Storm 的实时推荐系统，给出推荐算法和代码，并对推荐结果进行验证，给出结论。 要解决的问题：针对现有推荐系统时效性不强，信息利用率低等问题，采用现在流行的实时数据处理框架 Storm 并分析现有的推荐系统算法，研究设计出一个音乐实时推荐系统，以解决前面的问题。 本文的初步方案：采用 Hbase 或 HDFS 存储历史的浏览、购买行为信息，用 Hadoop 来进行数据预处理，生成好分析好的离线数据，采用Storm 的实时流式数据处理调用离线数据并 结合当前用户行为进行信息推荐。

## 1.4 设计目标

（1）实现同时提供离线推荐请求和在线推荐请求处理接口， 并且提供接收来自业务平台页面发送的用户实时行为信息的接口， 为在线推荐计算提供信息源。  
（2） 随着个性化推荐系统对实效性要求越来越高，本文设计要求业务平台从发送请求到获取结果中间处理的时间在ms级别。  
（3） 面对逐渐增加的用户量和访问行为， 数据发布平台需要做负载均衡处理高并发的来自业务平台的推荐请求。  
（4）系统能够安全可靠地存储和快速提取推荐结果并且能够随着数据增加有效地线性扩容是非常关键的，因此分布式的存储架构是本系统的选择。

## 1.5 论文的组织架构

本论文共分为六部分，按照如下方式组织：

(1)绪论，介绍了本项目的研究意义、国内外研究现状，说明本作者的主要研究内容和论文组织框架。

(2)系统概述，这一章介绍了推荐系统和Storm的相关概念及原理，主要包括基本概念、分类、以及发展模型；并介绍了本项目主要使用的推荐算法。

(3)系统设计，推荐系统的设计思路，具体分为设计思路、数据库设计、功能模块。

(4)系统实现，主要是推荐系统的具体实现过程。

(5)系统测试，编写系统的测试用例，测试覆盖率达80%以上，并根据推荐系统的测试指标对推荐算法及整个系统进行测试。

(6)总结和展望，最后对整个论文及项目进行总结，并提出下一步规划。

# 第二章 系统概述

## 2.1 推荐系统

Resnick和Varian定义了一个被广泛认同的推荐系统概念：“它是利用电子商务网站向客户提供商品信息和建议，帮助用户决定应该购买什么产品，模拟销售人员帮助客户完成购买过程”。推荐系统是自动联系物品和用户的一种工具，它能够在信息过载的环境中帮助用户发现令他们感兴趣的信息，也能将信息推送给对它们感兴趣的用户。随着信息技术和互联网的发展，人们逐渐从信息匮乏的时代走向信息过载的时代。一类是用户有明确的需求，比如我想看电影《哈利波特》,那我我可能会去百度《哈利波特》，然后找一个链接点进去看。还有，比如我想在淘宝上买小米手环，这样我就会上淘宝，然后在搜索框中输入小米手环，然后进入店铺购买。这种是有明确需求的。但往往我们并没有明确的需求，比如你今天很无聊，想找一部电视剧来看。但当你打开某个下载网站，面对100年来发行的数不胜数的电视剧，你会手足无措，不知道该看哪一部。此时，你遇到了信息过载的问题，需要一个人或者工具来帮助你做筛选，给出一些建议供你选择。推荐系统就是这样一个自动化的工具。它可以帮你分析你的兴趣，找出喜欢你的偏好，推荐给你一些可能的物品供你选择。

自从第一篇关于协同过滤的论文在1990年中期出现后，推荐系统已经成为一个重要的研究领域，在过去的10年，推荐系统在工业界和学术界都得到了很大的发展。诞生了很多优秀的推荐算法，而推荐算法是一个推荐系统的灵魂，推荐系统的好坏通常由推荐算法决定。现在主要分为以下几种推荐算法：

1. 基于内容的推荐（Content-based Recommendation），其主要思路就是根据用户历史记录，分析用户喜好的商品，并通过提取物品特征，查找和用户喜欢物品相似的物品，推荐给用户的过程。
2. 协同过滤推荐（Collaborative Filtering Recommendation）,是利用某些兴趣相投，拥有共同经验群体的喜好来推荐使用者喜好物品的一种方式。

## 2.2 实时计算框架Storm

流计算是目前计算机领域非常热门的技术，而Storm又是其中的佼佼者。Storm是一个分布式的、可靠的实时计算系统，又BackType开发，随后被Twitter收购。与Hadoop的批处理不同，Storm采用流式的消息处理方法，它使得消息可以的得到快速的处理，可以用于实时性要求较高的系统，例如广告点击的在线统计等。Storm弥补了Hadoop在实时处理方面的缺陷。原来的大多数推荐系统就是给予Hadoop来做的，它不能及时处理，要等系统在某个时间点统一进行批处理，然后再把推荐结果呈现出来，而Storm却不需要这样，它可以实时处理，随即将结果呈现出来。

Storm的基本结构如图所示

Storm集群中存在两种类型的节点：运行Nimbus服务的主节点和运行Supervisor服务的工作节点。Storm集群由一个主节点和多个工作节点组成。主节点上运行一个名为“Nimbus”的守护进程，用于分配代码、布置任务及检测故障。每个工作节点则运行一个名为“Supervisor”的守护进程，用于监听工作、开始并终止工作进程。

Worker由Supervisor负责启动，一个Worker中可以有多个Executor线程，每个Executor中又可以包含一个或多个Task。Task为Storm中的最小处理单元。

在逻辑上，Storm中消息的来源节点被称为Spout，消息的处理节点被称为Bolt，在系统中可以存在多个Spout及Bolt，且每个Spout或Bolt都可以设置不同的并行度，如图所示。

Nimbus与Supervisor的协调通过Zookeeper来实现，Nimbus进程和Supervisor进程都是快速失败或者无状态的，所以的状态都保存在Zookeeper中，可以当节点出现故障时，重启即可，保证Storm具有很好的健壮性。

## 2.3 数据库

### 2.3.1 Mysql数据库

MySQL是一个[关系型数据库管理系统](http://baike.baidu.com/view/1450387.htm" \t "http://baike.baidu.com/_blank)，MySQL 最流行的[关系型数据库管理系统](http://baike.baidu.com/view/1450387.htm" \t "http://baike.baidu.com/_blank)，在 WEB 应用方面MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。

MySQL是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。MySQL所使用的 SQL 语言是用于访问[数据库](http://baike.baidu.com/view/1088.htm" \t "http://baike.baidu.com/_blank)的最常用标准化语言。MySQL 软件采用了双授权政策，它分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是[开放源码](http://baike.baidu.com/view/394804.htm" \t "http://baike.baidu.com/_blank)这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

### 2.3.2 Redis内存数据库

Redis是一个开源的高性能键值对数据库，它通过提供多种键值数据类型来适应不同场景下的存储需求，并借助许多高层级的接口使其可以胜任如缓存、队列系统等不同的角色。

### 2.3.3 HBase数据库

Hbase的底层是HDFS文件系统，是一个面向列的非关系型数据库。是一个高可靠性、高性能、可伸缩的分布式存储系统。

## 2.4 WebSocket协议

WebSocket是HTML5开始提供的一种在单个TCP连接上进行全双工通讯的协议，WebSocket是基于TCP协议实现的。通过WebSocket，服务端和客户端可以建立双向连接，并且可以同时向双方发送请求，进行双向通信。通常WebSocket由通信协议和编程API共同组成，使用这些API可以直接从服务器直接发送数据到客户端应用。

## 2.5 本章小结

本章对推荐系统进行了概述，并介绍了Storm及其主要组件，详细阐述了推荐系统的两种主要推荐算法。

# 第三章 系统设计

通过前面章节的介绍，我们已经对推荐系统以及推荐算法有了一定的了解，同时我们也了解实时计算框架Storm，在此基础上，本章将对推荐系统的设计与实现进行介绍，根据设计目标确定系统架构、具体技术等，再通过合适的软硬件完成系统的部署实现。

## 3.1 设计目标

对于要实现的推荐系统，提出以下设计目标：

1. 良好的前端交互页面:设计前台与用户的设计界面，务必简洁大方。给用户一种清新的感觉，且要易用。简单的几个按钮（比如播放、暂停、喜欢、不喜欢）等。
2. 音乐数量：具有海量的音乐作为推荐数据源，这部分可以采用网易云公开的API，编写爬虫程序就行爬取，音乐数量至少30000首。
3. 时效性：本系统为实时推荐系统，则要求系统的延迟时间在ms级别。即是后台实时处理用户的行为，并实时更新推荐结果。
4. 部署简单：采用Maven管理项目工程，编写良好的POM文件，自动化解决项目依赖包的问题，便于移植部署。

## 3.2 系统总体架构设计

本文提出基于Storm的分布式实时推荐系统，本系统是一个关于音乐的推荐系统，根据用户的历史听歌记录，从而推荐给用户可能喜欢的音乐。

由Storm负责实时处理从前台传来的用户的请求，从而更新相似度举证和计算推荐结果。整个系统，将用户数据存储在非关系型数据库Hbase中，同时将一些中间结果存放在内存数据库redis中。前端采用Bootstrap来编写，音乐播放使用HTML5的audio标签实现。

1. 前台交互

前台主体采用Bootstrap框架编写，新用户进来先注册，注册的时候可能会让用户填一些基本信息（比如性别、年龄、职业、爱好音乐种类等），以便后面对用户进行建模。注册成功就登录进登录进系统，主要就一个页面，显示用户昵称，播放歌曲，用户可以对这首歌选择喜欢或者不喜欢。后台系统将实时更新用户的偏好。

1. 数据存储

用户注册时的信息存储在Mysql中，表的结构如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| uid | int unsigned | NO | PRI |  | auto\_increment |
| nick | VARCHAR(50) | NO |  |  |  |
| email | VARCHAR(50) | NO |  |  |  |
| password | VARCHAR(60) | NO |  |  |  |
| age | int unsigned | NO |  |  |  |
| sex | tinyint | NO |  |  |  |
| address | VARCHAR(80) | NO |  |  |  |
| preference | VARCHAR(80) | NO |  |  |  |

用户的浏览数据实时存储进Hbase里面，供实时分析处理模块处理，整个集群的一些计算中间结果存储在redis中。

1. 实时处理

Storm集群上运行着包含着推荐算法的拓扑程序，将收集到的用户的实时行为传入处理，增量更新相似度矩阵，计算实时推荐如果。在redis中存储着每个用户对听过的音乐的评价。同时推荐结果将会写入redis中的缓存推荐队列中，前端从这里面取数据然后将歌曲呈现给用户。

## 3.3 音乐推荐算法设计

1. 对用户进行建模

用户建模解决的是不同用户的特征差异。使用基本信息对用户的音乐品味进行划分。一个成功的音乐推荐系统需要对不同的用户提供满足其不同相应需求的音乐。这一步我们在用户注册的时候根据用户填写的个人资料从而对用户进行建模。例如，不同的地理位置的人，或者不同的年龄段，性别，生活方式，兴趣等等这些因素会影响用户的音乐偏好，决定着他们喜欢哪一种类型的音乐。还要对用户的音乐经历进行建模。不同的用户对音乐的专业知识是不一样的，那么他们对所听音乐的期望也是不一样的。比如我们可以把人群按照对音乐理解程度，专业知识划分为四类：专业音乐人士，喜欢音乐的人，闲的时候听听音乐，听不听音乐无所谓。他们在人群的比例分别是7%，21%,32%,40%.基于用户对于所听音乐的期许，我们需要把那些长尾的未知音乐过滤出来推荐给用户。

1. 对音乐进行建模

要想成功的推荐给用户合适的音乐，我们必须要研究音乐本身的属性。首先音乐的分布者会给音乐一些标签，包括专辑名称、作曲家、标题、流派等等。还有就是对音乐的音频信号进行分析，包括节拍，节奏，高音，乐器，心情等。

## 3.4 本章小结

本章中研究实时音乐推荐系统中所用到的具体技术，同时对整个系统的架构进行了设计。设计了基于Storm的音乐相似度矩阵的计算过程。最后详细介绍了系统所用到的算法。

# 第四章 系统实现

Storm集群需要至少需要三台机器，所以我们需要搭建在虚拟机上，首先安装虚拟机软件VirtualBox，之后在其提供的平台上安装三个相同的操作系统。

## 4.1 VirtualBox与操作系统

### 4.1.1 简要介绍

VirtualBox是一个“虚拟PC”软件，可以在一个虚拟机上同时运行一个或多个操作系统，在用户看来就像对windows应用进程进行切换一样。VirtualBox就像一个模拟器，能为客户操作系统提供了一套完整的虚拟硬件资源，每个虚拟操作系统都拥有自己独立的硬盘分区、网络接口、声卡等各类设备，它们之间互不影响。VirtualBox体积小，且是开源免费的。不像VMware体积大且是收费的。

由于三台虚拟机需要进行通信，所以着重介绍下VirtualBox中网络的连接方式。

表4.1 VirtualBox中虚拟机操作系统网络连接方式

|  |  |
| --- | --- |
| **网络连接方式** | **描述** |
| 桥接（Bridge） | 在这一配置中，主机系统为客户虚拟机充当透明网桥，所有的虚拟机系统都有自己的MAC和IP地址，从而在网络上作为一个完全独立的机器出现，是对外部直接可见的。 |
| 网络地址转换（NAT） | 主机系统充当虚拟网络的网关，在这个虚拟网络中虚拟机被托管，这意味着系统共享主机系统的IP四肢，虚拟网络中的DHCP服务器为虚拟系统分配一个IP地址。虚拟机系统对外部不可见。 |
| 主机（Host-only） | 客户虚拟机和主机系统属于同一个网络，并且没有连接到互联网。宿主机系统模拟出虚拟交换机，所有虚拟机系统的数据包都通过这个交换机进出网络。客户机通常只能拥有私有IP地址。 |
| 自定义（Custom） | 若想自己建立一个复杂的网络，可以使用自定义模式。 |

由于我们的主机需要相互通信，所以要有自己的IP，且要对外可见，所以选择桥接（Bridge）作为虚拟机连网方式。

### 4.1.2 安装

安装时，软件版本选择如下：

主机系统版本：Mac OS X Yosemite 10.10.5

VirtualBox版本：VirtualBox 5.0.10

虚拟机操作系统：Fedora 20

VirtualBox安装后如图4.1所示：

点击创建新的虚拟机，依次选择合适的硬件配置，并选择下载好的相应版本的操作系统镜像文件，一个新的虚拟操作系统就创建好了。在测试调试环境下，网络连接方式选择了Nat，在后期配置完成后，选择桥接方式。

## 4.2 Storm集群搭建

首先安装了三台虚拟机操作系统，ip分别为192.168.1.102,192.168.1.103,192.168.1.104

在上面安装软件服务分配如下：

Nimbus： ip：192.168.1.102

Zookeeper：ip：192.168.1.102；192.168.1.103；192.168.1.104

Supervisor：ip：192.168.1.103；192.168.1.104

4.2.1 搭建Zookeeper集群

三台机器同时作为Zookeeper集群。

对每台机器均修改conf/zoo.cfg(直接将zoo\_sample.cfg重命名为zoo.cfg)

# The number of milliseconds of each tick

tickTime=2000

# The number of ticks that the initial

# synchronization phase can take

initLimit=10

# The number of ticks that can pass between

# sending a request and getting an acknowledgement

syncLimit=5

autopurge.purgeInterval=24

autopurge.snapRetainCount=5

# the directory where the snapshot is stored.

# do not use /tmp for storage, /tmp here is just

# example sakes.

dataDir=/usr/local/zookeeper/data

dataLogDir=/usr/local/zookeeper/logs

# the port at which the clients will connect

clientPort=2181

server.1=192.168.1.102:2888:3888

server.2=192.168.1.103:2888:3888

server.3=192.168.1.104:2888:3888

# the maximum number of client connections.

# increase this if you need to handle more clients

#maxClientCnxns=60

#

# Be sure to read the maintenance section of the

# administrator guide before turning on autopurge.

#

# The number of snapshots to retain in dataDir

#autopurge.snapRetainCount=3

# Purge task interval in hours

# Set to "0" to disable auto purge feature

#autopurge.purgeInterval=1

maxClientCnxns=50

接下来设置myid

在设置的dataDir目录下新建myid文件，如上面设置的/var/zookeeper目录下。 myid文件中只有一行数字，即与zoo.cfg设置的主机与id对应的id， (默认每台机器通过sudo -s 进入root环境， 缺失的目录通过mkdir 补全)

在192.168.1.102上执行

#echo 1 >> /var/zookeeper/myid

在192.168.1.103上执行

#echo 2 >> /var/zookeeper/myid

在192.168.1.104上执行

#echo 3 >> /var/zookeeper/myid

然后在每台虚拟机上启动zookeeper，这样就搭建好Storm需要的Zookeeper集群。

4.2.2 搭建storm集群

在nimbus、supervisor节点安装依赖包（jdk，python）

下载并解压Storm到nimbus、supervisor节点

修改nimbus、supervisor节点的配置文件（storm.yaml），每台机器相同配置。

Storm的配置文件位于storm主目录下的conf/storm.yaml。

storm.zookeeper.servers：指定zookeeper集群中的主机列表。

nimbus.host:指定nimbus节点对应的主机。配置如下：

Storm.zookeeper.servers:

192.168.1.102

192.168.1.103

192.168.1.104

Nimbus.host 192.168.1.102

supervisor.slots.ports:对于每个supervisor节点，需要指定一些端口，来运行相应数目的JVM进程。下面的配置开发了四个端口，即在supervisor节点上运行了四个JVM进程（4个worker、此处涉及到Storm中的并行化机制）。

Supervisor.slots.ports:

-6700

-6701

-6702

-6703

注：nimbus节点和supervisor节点都需要配置storm.yaml

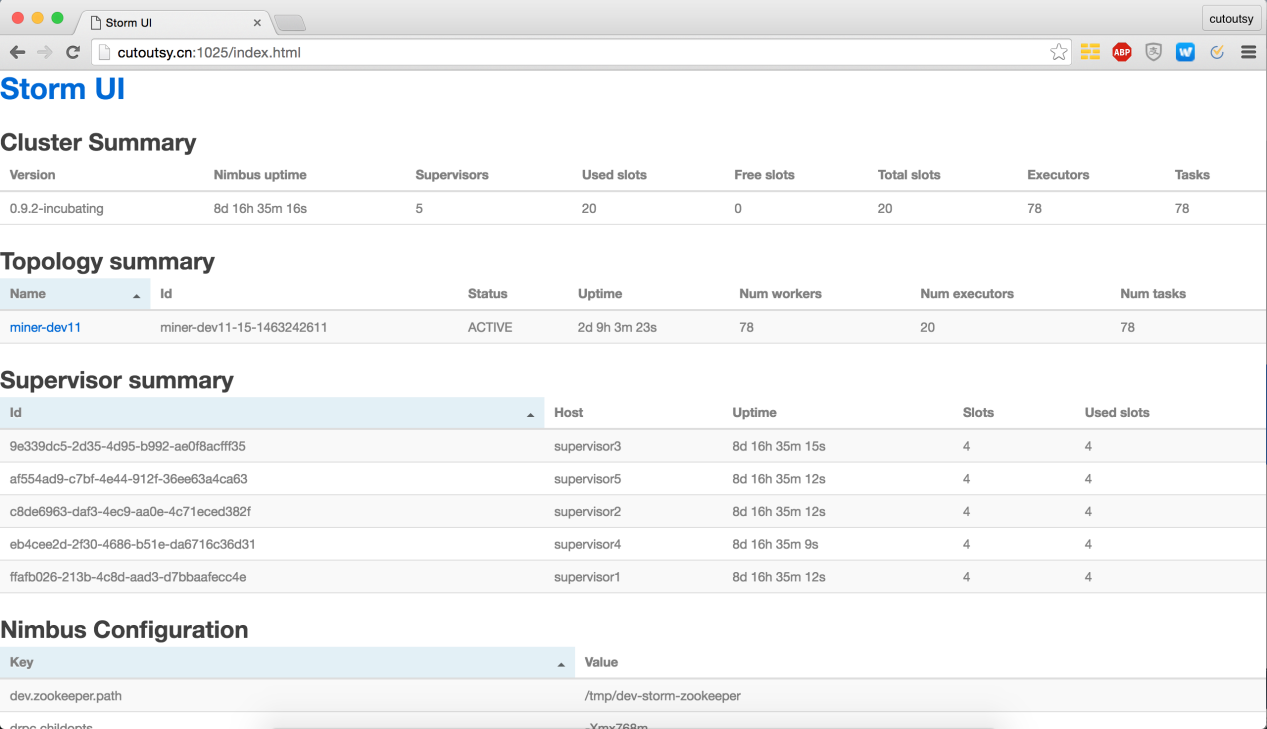
使用storm脚本启动守护进程（包括nimbus、supervisor、ui）

Nimbus:在nimbus节点上运行storm nimbus命令（102）

Supervisor：在supervisor节点上运行storm supervisor命令（103,104）

Ui：在nimbus节点上运行storm ui命令（102）

访问<http://192.168.56.102:8088>，搭建成功的界面如图所示：



## 4.3 前端页面编写

Bootstrap是Twitter推出的一个用于前端开发的开源工具包。是一个CSS/HTML框架。首先我们需要将Bootstrap下载到本地，或者可以直接用存在CDN上的Bootstrap文件。在html文件头部引入：

<link cheet”href=”//cdn.bootcss.com/bootstrap/3.3.5/css/bootstrap.min.css”>

引入Bootstrap核心CSS文件

<script cheet”href=”//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js”>

引入Bootsrap核心的JavaScript文件

下面是一个最简单也是最基本的Bootstrap页面

<!DOCTYPE html><html lang="zh-CN">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- 上述3个meta标签\*必须\*放在最前面，任何其他内容都\*必须\*跟随其后！ -->

<title>Bootstrap 101 Template</title>

<!-- Bootstrap -->

<link href="css/bootstrap.min.css" rel="stylesheet">

<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->

<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

<!--[if lt IE 9]>

<script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"></script>

<script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"></script>

<![endif]-->

</head>

<body>

<h1>你好，世界！</h1>

<script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></script>

<!-- Include all compiled plugins (below), or include individual files as needed -->

<script src="js/bootstrap.min.js"></script>

</body></html>

接着我用其中的表单控件编写了登录页面，登录页面主要就是一个账号（用户注册时的email）和登录密码。

表单代码如下：

<form>

<div class="form-group">

<label for="exampleInputEmail1">Email address</label>

<input type="email" class="form-control" id="exampleInputEmail1" placeholder="Email">

</div>

<div class="form-group">

<label for="exampleInputPassword1">Password</label>

<input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">

</div>

<div class="form-group">

<label for="exampleInputFile">File input</label>

<input type="file" id="exampleInputFile">

<p class="help-block">Example block-level help text here.</p>

</div>

<div class="checkbox">

<label>

<input type="checkbox"> Check me out

</label>

</div>

<button type="submit" class="btn btn-default">Submit</button></form>

最后的登录页面如图所示：

同样的利用表单组件也编写了用户注册页面，注册页面如图所示。

HTML5提供了音频播放的标准，现在在网页上播放音乐不用像以前那样通过使用插件flash来播放，并不是所有的浏览器都支持这个插件。HTML5规定了一种通过audio元素来包含音频的标准方法。Audio元素能够播放声音文件和音频流。

当前audio元素支持三种音频格式以及它和浏览器的兼容性，如表格所示

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | IE9 | Firefox3.5 | Opera 10.5 | Chrome | Safari |
| Ogg Vorbis |  | √ | √ | √ |  |
| MP3 | √ |  |  | √ | √ |
| Wav |  | √ | √ |  | √ |

音乐文件一般都是MP3文件，所以大多数浏览器都支持。在网页代码里插入如下代码就可以播放音频：

<audio src=”song.mp3” controls=”controls”></audio>

Control属性供添加播放、暂停和音量控件。

<audio>与</audio>之间插入的内容是供不提供audio元素的浏览器显示的，当用户无法播放音乐时可以告诉用户更换浏览器，这样对用户更友好，比如像下面这样。

<audio src=”song.mp3” controls=”controls”>你的浏览器不支持HTML5音乐播放，请更换浏览器。</audio>

<audio>标签的属性

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 描述 |
| autoplay | autoplay | 如果出现该属性，则音频在就绪后马上播放 |
| controls | controls | 如果出现该属性，则向用户显示控件，比如播放按钮 |
| loop | loop | 如果出现该属性，则每当音频结束时重新开始播放 |
| preload | preload | 如果出现该属性，则音频在页面加载时进行加载，并预备播放，如果使用“autoplay”，则忽略该属性。 |
| src | urlcccc | 要播放的音频的URL。 |

最后的音乐播放主界面如图所示：

## 4.4 Storm拓扑实现

### 4.4.1 拓扑处理逻辑

如图所示，实时处理过程中的Storm拓扑有三部分组成。

DataSpout为数据源，负责从redis消息队列中读取用户的行为数据，并将这些数据作为数据源传入拓扑结构中。

EstimateBolt为“行为-评分”转化Bolt，负责将用户对物品的行为转换为对物品的评分。

由数据源DataSpout到“行为-评分”转换Bolt采用的分组策略为ShuffleGrouping，这样可以使得每一个“行为-评分”转化Bolt得到相同数量的数据。

RecommendBolt为推荐Bolt，通过用户最近的评分记录以及系统启动时得到的音乐相似度计算并推荐相似物品。由“行为-评分”转换Bolt

到推荐Bolt采用的分组策略为FieldGrouping，按UserId字段进行分组，这样相同用户的行为将被分发到同一个推荐Bolt中进行处理。

图 实时处理拓扑图

### 4.1.2 实时处理步骤

步骤1：数据源DataSpout从redis消息队列中取得数据，并将数据发送到下一个Bolt中。

步骤2：Bolt在接收到数据源发动的用户的行为事件时，将这些事件放入内存队列中，并使用用户标识（UserId）和音乐标识（ItemID）。定义过期策略，将过期的事件从队列中删除。

步骤3：计算用户对此音乐的评分。扫描内存中的事件队列，从中选择用户听取最多的音乐以及听取这个音乐的次数。通过音乐类型、听取次数以及“行为-评分”元数据决定用户对此音乐的评分。

步骤4：更新音乐相似度矩阵并查找相似物品。通过物品标识（ItemID），在redis中查找其相似度矩阵，更新相似度矩阵，并根据用户对当前音乐的评分（即步骤3中计算得到的评分）以及相似度矩阵计算其他歌曲的预计评分。

步骤5：对预计评分进行降序排列，并选取评分最高的K个物品作为推荐歌曲。

步骤6：将被推荐的歌曲写入redis中的歌曲推荐队列中，推荐给相应用户。

### 4.1.3 具体代码实现

下面是TopologyMain的具体代码，代码中分别设置了DataSpout,EstimateBolt,RecommendBolt及其对应的并发与

分组关系，其中各个Spout与Bolt对应逻辑架构图中的各个部分。

TopologyBuilder topologyBuilder = new TopologyBuilder();

topologyBuilder.setSpout(“dataSpout”, new DataSpout(), 2);

topologyBuilder.setBolt(“estimateBolt”,new EstimateBolt(),8).shufferGrouping(“dataSpout”);

topologyBuilder.setBolt(“recommendBolt”,new RecommendBolt(), 8).fieldsGrouping(“estimateBolt”);

在DataSpout中，由于我们要从redis用户行为队列中取数据，所以我们要连接redis数据库，我们采用连接池技术进行redis数据库连接，这样可以减少连接创建时间，简化编程模式，也使得资源可控。主体代码如下：

private JedisPool initJedisPool(String ip, int port, String password) {

if (pool == null) {

int max\_active = 1000;

int max\_total = 1000;

int max\_idle = 1000;

int max\_wait = 100;

JedisPoolConfig config = new JedisPoolConfig();

config.setMaxWaitMillis(1000 \* max\_wait);

config.setMaxIdle(max\_idle);

config.setMaxTotal(max\_total);

config.setTestOnBorrow(true);

config.setTestOnReturn(true);

try {

pool = new JedisPool(config, ip, port, 1000 \* 2, password);

} catch (Exception e) {

e.printStackTrace();

}

}

return pool;

}

public JedisPool getJedisPool() {

return this.pool;

}

/\* 从JedisPool中得到Jedis实例，并使引用计数加1 \*/

public Jedis getJedisInstance() {

Jedis jedis = pool.getResource();

jedis\_instance\_count++;

if (jedis != null) {

return jedis;

}

return null;

}

## 4.5 测试用例编写

为了保证软件的正确性、完整性和质量的过程。我们要在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估。

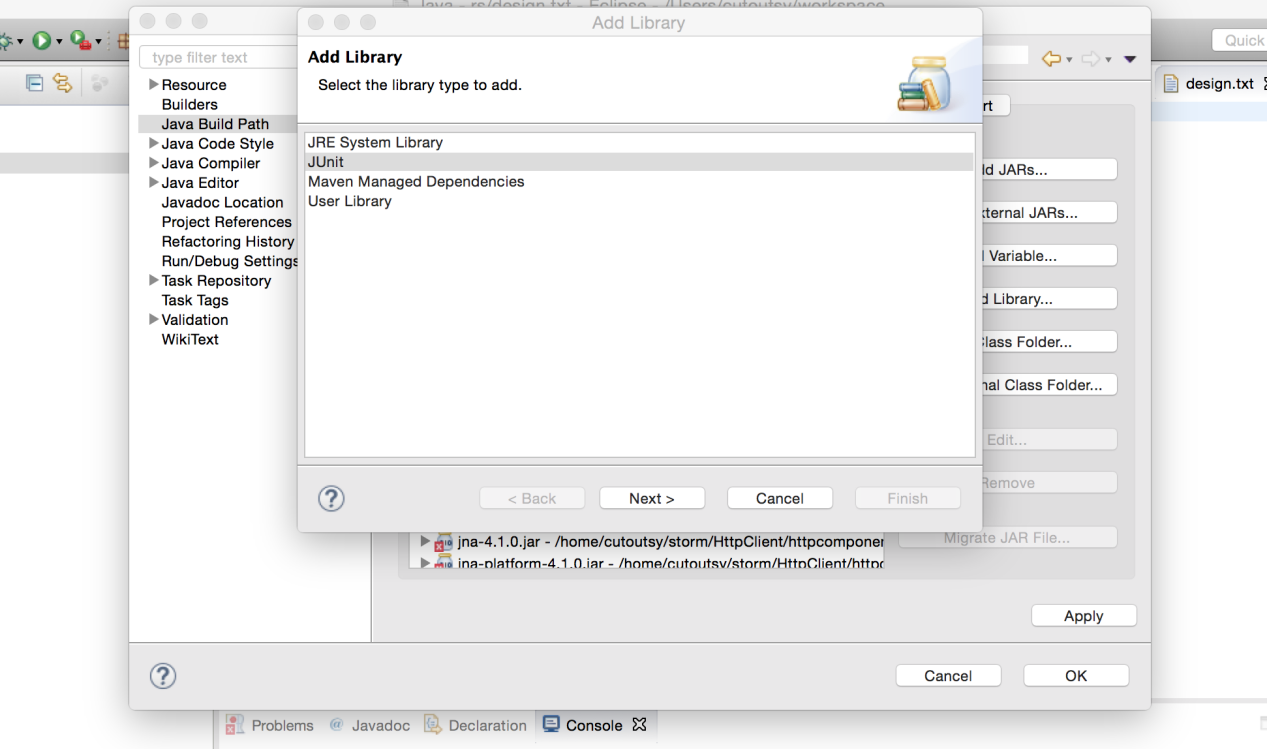
本系统主要是Java编写，所以主要采用Junit进行单元测试。

### 4.5.1 Junit

JUnit是一个Java语言的单元测试框架。JUnit有它自己的JUnit扩展生态圈。多数Java的开发环境都已经集成JUnit作为单元测试的工具。Junit是一套框架，继承TestCase类，就可以用Junit进行自动测试了。尤其是结合项目管理工具Maven，可以很好的保证项目的质量，写出代码质量高，系统可靠的项目。

4.5.2 Junit使用

由于在大多数JAVA IDE都集成了Junit，所以我们要想使用Junit，只需在工程里add Library...然后在弹出的对话框中选择Junit即可，如下图所示：



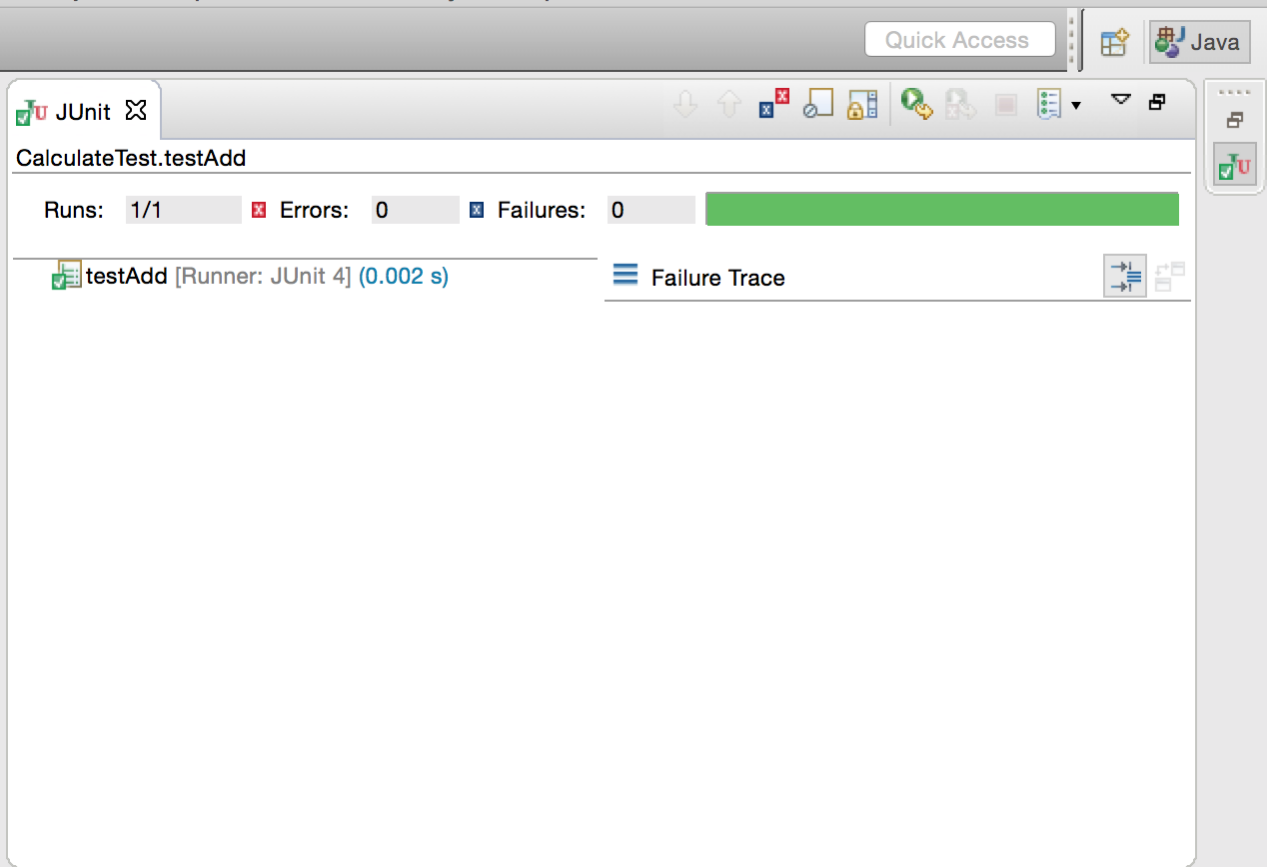
这样就把Junit引入当前项目库中了。

测试代码最后是不会出现在最终产品中的。所以我们不能把测试代码与被测试代码（源代码）放在一起，而是在源代码的同级目录创建测试代码目录（test），并保证测试代码和被测试代码使用相同的包名。这样既保证了代码的分离，同时还保证了查找的方便。

为了更好的区分测试类和被测试类。测试方法需要使用“Test”开头，同时测试方法需要按照一定的规范书写：

1. 测试方法必须使用注解org.junit.Test修饰。
2. 测试方法必须使用public void修饰，而且不能带有任何参数。

看测试运行结果如何，在测试类上点击右键，在弹出的菜单中选择Run As Junit Test。运行成功结果如图所示：



绿色的进度条提示我们测试运行通过了。在测试的过程中，我们要对代码可能出现的问题进行全面预测。

测试套件

随着我们系统的代码越来越多，单元测试的类也越来越多，如果我们一个一个的单独运行测试类，这肯定是不行的。Junit提供了一种批量运行测试类的方法，叫做测试套件，这样，每次需要验证系统功能正确性时，只执行一个或几个测试套件便可以了。测试套件的编写需要遵循以下规则：

1. 创建一个空类作为测试套件的入口。
2. 使用主键org.junit.runner.RunWith和org.junit.runners.Suite.SuiteClasses修饰这个空类。
3. 将org.junit.runners.Suite作为参数传入注解RunWith，以提示JUnit为此类使用套件运行器执行。
4. 将需要放入此测试套件的测试类组成数组作为注解SuiteClasses的参数。
5. 保证这个空类使用public修饰，而且存在公开的不带任何参数的构造函数。

## 4.6 本章小结

# 第五章 系统测试

在实现推荐系统的部署后，需要对系统功能进行测试，来验证其可靠性，因此，针对系统的不同模块以及推荐算法的性能和准确性进行测试。

## 5.1 网页前端测试

本小节的内容是为了提升用户体验而设计的，并且实现了跨平台的功能，是本文不可或缺的一部分内容。本文的主体框架是采用Bootstrap[20]框架，后台采用Java[21]语言编写，前端采用html+css+js的混合语言进行编写的。这个网页界面主要分为三部分功能：1)、用户注册。2)、用户登录。3）音乐推荐。具体设计过程在前面已经详细叙述，以下是本界面的测试结果。

1. 注册界面

界面如图所示，在提交前JavaScript在数据被送往服务器前对表单中的这些输入数据进行验证。主要验证的表单数据有：

用户是否已填写表单中的必填项目。

用户输入的邮件地址是否合法。

用户是否输入合法的日期。

用户是否在数据域输入了文本。

图 注册页面

1. 登录界面

登录界面如图所示，也要验证表单数据，输入正确的用户名密码可以进入音乐推荐页面，用户错误的用户名密码会提示用户名或密码错误。

图 登录界面图

1. 音乐推荐页面

主要验证了音乐是否能正确播放，推荐音乐是否能正确的显示出来以及按钮点击是否起作用。

图 音乐推荐页面

1. 兼容性测试

兼容性测试主要是测试前端页面在各个不同的浏览器下显示、行为是否正确、一致，以及在同一浏览器不同版本下的兼容性。经过测试，兼容性结果如表12所示。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | IE9 | Firefox3.5 | Opera 10.5 | Chrome | Safari | IE10 |
| 推荐系统 |  | √ | √ | √ | √ | √ |

## 5.2 推荐系统测试

推荐系统有各种评测指标。这些评测指标可用于评价推荐系统各方面的性能。这些指标有些可以定量计算，有些只能定性描述。本系统主要测试推荐系统的用户满意度、预测准确度、实时性、健壮性。

5.2.1 用户满意度

用户作为推荐系统的重要参与者，其满意度是评测推荐系统的最重要指标。但是，用户满意度没有办法离线计算，只能通过用户调查或者在线实验获得。本测试主要通过调查问卷的形式来获得用户满意度。把用户对推荐系统的满意度分为不同层次。设计了一系列的问题，比如有个问题我是这样设计的：

> 推荐的音乐都是我非常想看的。

> 推荐的音乐很多我都听过了，确实是符合我兴趣的不错音乐。

> 推荐的音乐和我的兴趣是相关的，但我并不喜欢。

> 不知道为什么会推荐这些音乐，它们和我的兴趣丝毫没有关系。

通过对回收回来的问卷进行数据分析，得到的分析结果如图88所示：

从图中可以看出，超过70%的用户对推荐的音乐是非常满意的。

除了问卷调查，我们在音乐播放页面放置了两个按钮（喜欢或者不喜欢）。可以通过统计两种按钮的单击情况来度量系统的用户满意度。通过统计，超过80%的用户对推荐的音乐是表示喜欢的。

5.2.2 预测准确度

预测准确度度量一个推荐系统或者推荐算法预测用户行为的能力。这个指标是最重要的推荐系统离线评测指标。在计算该指标是需要一个离线的数据集，该数据集包含用户的历史行为记录。然后将该数据集通过时间分为训练集合测试集。通过在训练集上建立用户的行为和兴趣模型预测用户在测试集上的行为，并计算预测行为和测试集上的实际行为的重合度作为预测准确度。可以通过评分预测和TopN推荐来测量此指标。本文采用TopN的方式。

推荐系统在推荐给用户音乐时，会给用户一个个性化的推荐列表。TopN推荐的预测准确率一般通过准确率（precision）/召回率（recall）度量。

令R（u）是根据用户在训练集上的行为给用户的推荐列表，而T(u)是用户在测试集上的行为列表。那么，推荐结果的召回率定义为：

Recall=

推荐结果的准确率定义为：

Precision=

通过对推荐系统的测试，得到准确率（precision）/召回率（recall）的曲线，如图89所示：

如图可知，本推荐系统具有良好的预测准确率。

5.2.3 实时性

在很多推荐系统中，因为物品具有很强的实效性。而本系统时基于Storm的实时推荐系统，实时性就显得显得尤为重要。

推荐系统的实时性包括两个方面。首先，推荐系统需要实时更新推荐列表来满足用户新的行为变化。这个可以通过推荐用户列表的变化速率来评测。如果推荐列表在用户有行为后变化不大，或者没有变化，说明推荐系统的实时性不高。通过测试，本系统在用户切换一首歌曲时，就会立即更新音乐相似度矩阵，同时更新用户的推荐列表。系统的实时处理在ms级。具有很好的实时性，可以达到预期的设计目标。

实时性的第二个方面是推荐系统需要能够将新加入系统的物品推荐给用户。这主要考验了推荐系统处理物品冷启动的能力。通过测试，本系统在加入新的音乐歌曲时，推荐系统能够很快将新音乐加入音乐相似度矩阵中，并且加入到给用户的推荐列表中，这个时间根据加入的音乐数量和实时计算拓扑所占有系统资源数量来决定。在测试中，在三台虚拟机搭建的集群中，加入5000首歌曲，系统跟新的时间在3s左右，还会很快的，具有很好的实时性。对于这个处理时间，我们可以通过给Storm集群增加节点来缩短这个时间，Storm节点越多，处理速度就越快，多新加入的音乐处理速度就越快，更新时间就越短。同时当用户的行为发生改变时，响应的速度也更快。

5.2.4 健壮性

本测试主要测试推荐系统的稳定性，由于系统选择Storm作为实时计算处理框架。而Storm具有可扩展性、高容错性和保证消息被处理等特点。从而保证整个推荐系统具有很好的健壮性。经过测试，系统可以连续运行30天以上而不出现错误。同时通过模拟大量用户在线使用的场景，对系统进行高并发的场景，测试结果表明系统在高并发的情况下依然工作良好。

## 5.3 本章小结

本章主要是对本文设计的推荐进行测试，从而验证该推荐系统是否具有良好健壮性，能否应付高并发的场景，同时针对推荐系统，测试了用户满意度、预测准确度和实时性。通过一系列的测试，从而证明了该推荐系统具有良好的推荐功能，实现了本文的预期目标。本章第二小节主要测试为了提升用户体验而设计了前端网页界面，根据测试结果显示，前端界面具有良好的显示效果和兼容各大主流浏览器，结果符合预期设想。

# 第六章 总结与展望

本章对整篇文章进行全面的总结，同时提出对未来的展望。

## 6.1 工作总结

实时推荐系统是退浆领域中比较新颖的一种推荐手段，将传统的离线计算转化为实时计算，极大提高了用户的满意度和预测准确度，从根本上改变了传统推荐系统滞后于推荐的状况。但是，实时推荐系统都还处于不断发展的过程，还有诸多可以进一步完善的地方。本文从研究分析实时数据处理框架Storm的角度出发，首先了解了推荐系统及Storm的概念、不同类型的推荐算法以及它的优缺点。然后介绍了本项目所要部署推荐系统的设计目标、设计的功能模块与拓扑结构。接着，开始部署实现推荐系统，具体介绍了部署推荐过程的过程与要注意的细节。最后，对系统功能进行测试。

## 6.2 未来展望

在实现推荐系统的过程中，由于作者的能力与时间限制，推荐系统还有很多不足之处需要继续研究探索并加以完善。总结如下：

1. 在线算法的改进

由于实时性的要求，在线部分的计算要求不能过于复杂，同时为了保证推荐结果能够以近期的行为为依托，故推荐系统采用基于当前时间的行为来计算推荐结果。不同的更新策略会有不同的推荐成功率，故应该继续优化好的更新策略，使得推荐更加准确。

1. 推荐算法的选择

系统中选择了基于物品的协同过滤算法，但不同的推荐算法有不同的优势，应该继续试验其他算法在系统中的推荐效率。同时可以考虑结合多种算法的优点，来使得推荐效果更好。

1. 系统的可配置性

现在系统基本是不可配置的，一个好的推荐系统应该是可以灵活配置的。后面将会增加数据库、相似度算法选择、Storm运行时的并行度等。使得系统更加完善。

# 致谢

本文是在我的导师的亲切关怀和悉心指导下完成的。张老师严谨的治学态度，精益求精的工作作风，渊博的学术知识都使我受益匪浅。从整个论文的选题、项目实现到最后的整篇论文完成，都给予了我耐心的指导，为我指点迷津，帮助我开拓思路。由于他的严格要求与悉心指导，我才能够顺利的完成毕业论文。在此，我以诚挚的心情向他表示衷心的感谢与崇高的敬意。

其次，感谢在我项目实现过程中对我提出宝贵意见的张伟师兄。也谢谢我的舍友们，在朝夕相处的生活中的照顾。感谢实验室的师兄师姐在日常学习和生活中，对我的无私指导和帮助。另外感谢吴杰、薛鹤宇从开始毕设设计到最后结束过程一路的陪伴，使我保持着积极向上的心态，这段时间也是我永生难忘的记忆。

感谢在大学四年中，所有教过我、指导过我、帮助过我的老师和学长，是你们让我拥有了丰富而美好的大学生活。

感谢本文所引用文献的各位作者，他们之前杰出的工作为本文的研究给予了非常大的帮助和启发。感谢在百忙之中抽出时间参与本文评审和出席答辩的各位老师和专家给予的指导。

感谢养育我长大的父母，感谢他们对我无微不至的照顾和一路的陪伴。

最后，再次感谢所有帮助过我，关心过我的人，谢谢!

# 参考文献

[1]刘建国, 周涛, and 汪秉宏. "个性化推荐系统的研究进展." 自然科学进展 19.1(2009):1-15.

[2]许海玲等. "互联网推荐系统比较研究." 软件学报 20.2(2009):350-362.

[3]李川. 实时个性化推荐系统的设计与实现. MS thesis. 北京邮电大学, 2015.

[4]项亮. 推荐系统实践. 人民邮电出版社, 2012.

[5]莫萍燕, and 王玉龙. "一种实时推荐系统数据发布平台的设计与实现." 电信网技术 9(2015).

[6]Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." Knowledge and Data Engineering, IEEE Transactions on 17.6 (2005): 734-749.

[7]Toshniwal, Ankit, et al. "Storm@ twitter." Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM, 2014.

[8]Jain, Ankit, and Anand Nalya. Learning Storm. Packt Publishing, 2014.

[9]http://blog.csdn.net/a130737/article/details/45081327

[10]http://storm.apache.org/

[11]http://www.bootcss.com/