中国科学技术大学

专业硕士学位论文

（专业学位类型）

**B2C商城及推荐系统设计与实现**

作者姓名：张宁

专业领域：软件工程

校内导师：戴礼荣 教授

企业导师：程宁 博士

完成时间：2018年2月14日

University of Science and Technology of China

A dissertation for master’s degree

（Professional degree type）



**Design and Achievement of B2C Shopping Mall and Recommendation System**

Author： Ning Zhang

Speciality： Software Engineering

Supervisor：Lirong Dai

Advisor： Ning Cheng

Finished time: February 14,2018

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文,是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：\_\_\_\_\_\_\_\_\_\_\_ 签字日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入《中国学位论文全文数据库》等有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

□公开 □保密（\_\_\_\_年）

作者签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 签字日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 摘要

随着互联网经济的蓬勃发展，淘宝、京东等网上购物模式的逐渐兴起，大量线下实体经济转入线上，从而导致商品资源快速膨胀，出现了严重的信息过载问题，这使得具有潜在用户的商品很容易被庞大的商品信息量、同行的恶意竞争、不合理的推广方式等原因在互联网经济浪潮中埋没。

本课题以此为出发点，通过线上征稿和众筹的方式由消费者决定商品的发布与否，商品主要突出时尚、活力、创新、高品质等元素，面向较为年轻、小众的消费群体，为其打造优质的网上购物平台，并结合推荐算法根据用户喜好帮助用户挖掘。

本文设计并实现了商城系统及推荐系统，商城部分主要基于SSM(Spring,

SpringMVC,Mybatis)等主流后端框架，设计实现了用户单点登陆、用户注册、用户订单管理、购物车管理、限时商品秒杀等功能，结合Redis、RebbitMQ等服务中间件，提高系统并发量和响应时间。此外在系统的设计和实现过程中对系统的实时性、稳定性做了相关优化工作。推荐系统部分主要设计了推荐算法的服务接口，采用策略模式灵活调用具体的实现算法，并对三种重要的推荐算法的相关原理和实现进行介绍。

整个系统在商城服务功能基本完善和稳定的前提下，通过推荐算法，根据用户历史行为和特征信息为其提供个性化的推荐服务，提升用户体验，实现三方共赢。

**关键词：**信息过载，商城系统，推荐系统

# ABSTRACT

**Key Words:**Information overload, ecommerce system, recommender systems

# 目录

[摘要 I](#_Toc511319943)

[ABSTRACT II](#_Toc511319944)

[目录 IV](#_Toc511319945)

[第1章 绪论 1](#_Toc511319946)

[1.1系统研究背景 1](#_Toc511319947)

[1.2系统研究意义 2](#_Toc511319948)

[1.3国内外研究概况 2](#_Toc511319949)

[1.4 本人主要研究内容和本人工作 3](#_Toc511319950)

[1.4.1 本人研究内容 3](#_Toc511319951)

[1.4.2 本人主要工作概述 3](#_Toc511319952)

[1.5论文结构安排 4](#_Toc511319953)

[第2章 相关工作 5](#_Toc511319954)

[2.1 Spark 分布式框架 5](#_Toc511319955)

[2.1.1 Spark Core技术 6](#_Toc511319956)

[2.1.2 Spark Streaming 7](#_Toc511319957)

[2.1.3 Spark MLlib 技术 7](#_Toc511319958)

[2.1.4 Spark与Hadoop区别 7](#_Toc511319959)

[2.2 服务端相关技术 8](#_Toc511319960)

[2.2.1 Spring框架简介 8](#_Toc511319961)

[2.2.2 SpringMVC框架简介 8](#_Toc511319962)

[2.2.3 Nginx反向代理 9](#_Toc511319963)

[2.3推荐算法 9](#_Toc511319964)

[2.3.1推荐系统介绍 9](#_Toc511319965)

[2.3.2协同过滤算法概述 10](#_Toc511319966)

[2.3.3基于人口统计学的推荐 11](#_Toc511319967)

[2.3.4基于内容的推荐 11](#_Toc511319968)

[2.4本章小结 12](#_Toc511319969)

[第3章 商城及推荐系统需求分析 13](#_Toc511319970)

[3.1系统业务需求 13](#_Toc511319971)

[3.2系统功能性需求 13](#_Toc511319972)

[3.2.1 商城系统及推荐系统总体功能需求 13](#_Toc511319973)

[3.2.2 登录注册模块需求分析 15](#_Toc511319974)

[3.2.3 购物车模块需求分析 16](#_Toc511319975)

[3.2.4 订单管理模块需求分析 17](#_Toc511319976)

[3.2.5 商品秒杀模块需求分析 18](#_Toc511319977)

[3.2.6 推荐系统计算框架需求分析 20](#_Toc511319978)

[3.2.7 基于user的协同过滤推荐模块需求分析 21](#_Toc511319979)

[3.2.8 基于item的协同过滤推荐模块需求分析 21](#_Toc511319980)

[3.2.9 基于内容的推荐模块需求分析 22](#_Toc511319981)

[3.3 系统非功能性需求 22](#_Toc511319982)

[3.3.1 并发性 22](#_Toc511319983)

[3.3.2 稳定性 22](#_Toc511319984)

[3.3.3 健壮性 23](#_Toc511319985)

[3.3.4 冷启动问题 23](#_Toc511319986)

[3.4 本章小结 23](#_Toc511319987)

[第4章 商城及推荐系统概要设计 24](#_Toc511319988)

[4.1 系统总体结构设计 24](#_Toc511319989)

[4.2 系统功能模块设计 25](#_Toc511319990)

[4.3 数据库设计 26](#_Toc511319991)

[4.4 推荐系统架构设计 30](#_Toc511319992)

[4.5 非功能性需求设计 30](#_Toc511319993)

[4.6 本章小结 33](#_Toc511319994)

[第五章商城及推荐系统详细设计 34](#_Toc511319995)

[5.1 商城基础功能模块设计与实现 34](#_Toc511319996)

[5.1.1 商城系统总体设计与实现 34](#_Toc511319997)

[5.1.2 单点登录模块设计与实现 35](#_Toc511319998)

[5.1.3 购物车模块设计与实现 37](#_Toc511319999)

[5.1.4 订单管理模块的设计与实现 39](#_Toc511320000)

[5.1.5 商品秒杀模块设计与实现 42](#_Toc511320001)

[5.2 推荐系统相关模块设计与实现 44](#_Toc511320002)

[5.2.1 推荐系统计算框架设计与实现 44](#_Toc511320003)

[5.2.2 基于user的协同过滤推荐模块设计与实现 47](#_Toc511320004)

[5.2.3 基于item的协同过滤推荐模块设计与实现 49](#_Toc511320005)

[5.2.4 基于内容的推荐模块设计与实现 51](#_Toc511320006)

[5.3 本章小结 53](#_Toc511320007)

[第6章 实验与测试 54](#_Toc511320008)

[6.1系统开发实现环境 54](#_Toc511320009)

[6.2商场系统测试 54](#_Toc511320010)

[6.2.1测试环境 54](#_Toc511320011)

[6.2.2商城系统功能性测试 55](#_Toc511320012)

[6.3推荐系统测试 58](#_Toc511320013)

[6.3.1测试环境 58](#_Toc511320014)

[6.3.2推荐系统功能性测试 59](#_Toc511320015)

[6.4非功能性测试 63](#_Toc511320016)

[6.4.1并发性测试 63](#_Toc511320017)

[6.4.2兼容性测试 64](#_Toc511320018)

[6.4.3冷启动效果测试 64](#_Toc511320019)

[6.5测试结论 66](#_Toc511320020)

[6.6本章小结 66](#_Toc511320021)

[第7章 总结与展望 67](#_Toc511320022)

[7.1 总结 67](#_Toc511320023)

[7.2 展望 68](#_Toc511320024)

[参考文献 69](#_Toc511320025)

[致 谢 71](#_Toc511320026)

# 第1章 绪论

## 1.1 系统研究背景

随着互联网经济的快速发展和网上购物的日益普及，以淘宝京东为首互联网消费平台日益兴起，众多商家纷纷入驻，商品种类繁多，品质参差不齐，且受众群体更是十分广泛。而本文所设计开发的商城系统与其有一定区别，该系统主要针对部分年轻、时尚、注重消费品质的消费群体，致力于为其提供和推荐新奇、时尚、高颜值、高品质的商品为目标，结合相关推荐算法，为用户挖掘推荐可能感兴趣的商品。

众所周知，大数据时代的今天，互联网经济的快速发展，随之增加的则是大量的用户和信息数据，数据增多的同时也出现了严重的信息过载问题，给人们的生活带来了些许不便，互联网信息的繁杂导致用户很难定位到自己需要的内容，为了解决上述问题，个性化推荐系统应运而生，并且已经得到了越来越多的关注，相关的专家学者们对其研究也更加深入。

商品信息种类数量繁多，导致搜索引擎出现功能缺陷，用户很难通过搜索来挖掘自己可能感兴趣的商品，因此商城个性化推荐系统应运而生，弥补了搜索引擎的部分缺陷。目前，推荐系统主要用于处理信息过载的情况，系统通过分析日志数据、用户历史行为、用户属性等信息创建用户兴趣偏好模型，通过算法根据用户的喜好对物品进行排序，之后经过后端服务向用户推送排序列表。因此推荐系统与搜索引擎相互补充，相辅相成，当用户有明确需求时则采用捜索引擎搜索相关信息，而在用户没有明确自身需求时推荐系统可以帮助他们获取到可能感兴趣的内容。推荐系统也借此在电子商务领域取得广泛应用。

 对于用户而言，由于商品的品牌、种类繁多，使得用户越来越难通过简单的关键字搜索以及商品的结构化属性排序来定位到目标商品。而对于商家而言，特别是对于没有成交量，没有历史交易记录的商家，他们发布的商品难以被用户发觉，需要推荐系统的介入，其次对于精准的广告投放，推荐系统可以根据用户的喜好，较为精准的将商家需要推广的产品推送到可能对它感兴趣的目标群体中，减少商家的广告成本，提高广告转化率。

然而，如何推送高质量推荐列表、快速地响应推荐请求、实时生成推荐列表、海量数据的分析都成为必须面对和解决的难题。

## 1.2 系统研究意义

在本文的以商城系统为主体，并在此基本功能和数据基础上搭建推荐系统模块，推荐系统从宏观上解决了当前互联网经济下信息过载问题，系统主要由数据收集和数据计算为主，通过离线计算和实时计算相结合的方式，加速计算效率和速度，并根据不同的推荐场景使用不同的推荐算法，为用户生成个性化推荐结果，对于消费者而言，推荐系统可以帮助消费者更好的挖掘潜在的兴趣商品，提高用户购物效率，增强用户体验。对于商家，有助于较为精准的投放商品广告信息，增加商品推广命中率，减少广告投放成本，并吸引潜在的客户群体。

    综上所述，该系统无论对于用户还是商家而言都是双赢，具有较高的应用价值和商业意义。

## 1.3 国内外研究概况

推荐系统从上世纪90年代开始被美国科学家研究，最早的推荐项目是对影片进行推荐，主要是通过根据用户点击习惯和一些影片评分综合起来创建用户画像，从而分析出用户可能喜欢感兴趣的电影，然后进行推荐。早期的算法主要是基于内容的推荐，通过对推荐物品本身进行特征提取，该方式会导致如果用户没有对某类产品评价，该类产品就永远不会被推荐。随着科学技术的发展，上述问题得以解决，原始的推荐算法即将被替代。

协同过滤推荐（Collaborative Filtering，简称CF）起源于1992年。该算法的理论基础在目前的推荐算法领域也占有很重要的地位，由于该算法效果明显，所以科学家基于该算法提出了诸多优化，例如基于物质的或者基于模型的协同过滤推荐算法，基本理论都是基于最初的协同过滤算法。推荐方式有很多种，具体如下：一、基于上下文的推荐[5]，即推荐系统会根据用户的访问位置、访问时间、用户心理以及已有信息预测该用户对其他物品可能的估值。由于个体用户的心理是变化莫测的所以预估的结果有可能有偏差，二、基于用户画像的推荐，即将该用户的所有信息进行整合抽象，把用户的所有数据进行展示，建立该用户的一个用户画像，从多个方面进行映射然后进行个性推荐。三、基于内容的推荐、即对数据内容进行分析并得到物品的结构化描述信息，分析用户在过去时间中评分或评论的商品，利用用户画像[7]构建预测模型，从而生成推荐策略。四、基于机器学习的推荐系统、根据数据的多样性选取合适的算法策略，自动的归纳逻辑和规则，并根据归纳总结的结果（模型）来对未来数据进行推荐。但是，上述每种单一方式的推荐系统都有各自的缺点，比如基于内容的推荐有严重的矩阵稀疏问题，基于关联规则的推荐较为耗时只能做离线处理等等。

上述的各个算法在不同的工作场景中都有各自的优势和缺陷，所以不能选取单一的推荐算法进行推荐，目前业界的标准方式是选取多个推荐方法相结合的方式进行推荐，目前比较流行的搭配是推荐算法和基于项目的推荐相结合的方式，上述组合能够达到比较满意的效果。虽然按排列组合的方式计算会有很多种结合方式，但是需要真正有效的结合才有意义，需要提高推荐算法的准确度，使得结合的算法在准确率、实时性和稳定性方面都有提高。

## 1.4 本人主要研究内容和本人工作

### 1.4.1 本人研究内容

    （1）通过对推荐系统的需求分析，进一步了解分析了推荐系统所需要的大数据基础组件，了解了Spark分布式框架及其相关组件的配置和使用，以及HDFS和Hbase的原理和数据存储方式。

    （2）随后研究对比了时下应用广泛的推荐算法，并通过案例分析，为不同的推荐算法选择合适的推荐场景，以获得较好的推荐结果。

    （3）为缩短推荐结果的生成，减轻服务器压力，设计了将离线计算与实时计算相分离的推荐系统计算框架。

    （4）分析研究了推荐系统的冷启动以及数据稀疏性等非功能性需求，设计了改善系统冷启动问题的相关算法。

### 1.4.2 本人主要工作概述

在开发B2C商城及推荐系统过程中，本人参与的具体工作如下：

  （1）B2C商城及推荐系统的设计与实现，主要包括各模块的需求分析、概要设计、详细设计以及相关功能模块的测试。

    （2）基于MVC设计模式，结合Sping、SpringMVC以及MyBatis三大开源框架构建了整个商城系统的基本骨架，并主要完成了商城的登录注册模块、商品的限时秒杀模块、以及用户订单管理和用户购物车管理等。

  （3）设计用户日志收集模块，针对推荐系统的性能和适用场景出发，设计推荐系统的计算框架，并主要实现了基于User的协同过滤推荐模块、基于Item的协同过滤推荐模块、和基于内容的推荐模块，并暴露对外接口供服务层调用。

   （4）对于推荐系统的非功能性需求冷启动部分给出算法层面解决方案。

   （5）针对商城系统和推荐系统进行相关模块的功能性测试。

## 1.5论文结构安排

    本文主要基于JavaWeb相关技术框架搭建网上商城及推荐系统，并在商城系统基本使用功能基础上，实现在不同的应用场景下为用户提供不同维度的商品推荐，提升用户购物体验。本文组织结构如下：

    第一章：绪论，主要基于商城及推荐系统的研究背景和意义进行考察，对国内外推荐系统在商城领域的应用进行阐述，并对论文的主要研究和工作内容做出简要介绍。

    第二章：介绍了论文的相关工作，主要是介绍了论文中使用到的关键技术，如Spark的相关技术栈、HDFS分布式文件系统以及Hbase数据库的原理和应用。并对推荐算法以及协同过滤相关算法做了大致的介绍。

    第三章：商城及推荐系统需求分析，主要针对系统的功能性需求和非功能性需求做出简要的分析和介绍。

    第四章：商城及推荐系统概要设计，对整个系统设计架构做出介绍，并对系统功能模块设计做出简要阐述，并介绍了数据库表的设计，以及对推荐系统冷启动问题的相关非功能性设计的介绍。

    第五章：商城及推荐系统详细设计，包括商城及推荐系统总类图的介绍，以及主要功能模块的设计实现，推荐系统计算框架和具体推荐模块的设计。

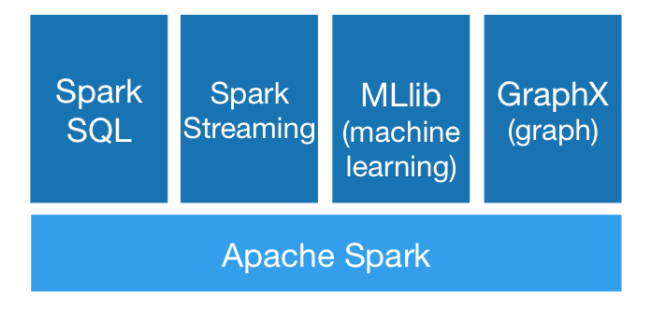
    第六章：商城及推荐系统测试，包括商城系统主要模块的功能性、健壮性测试以及推荐系统相关模块的测试。

    第七章：总结与展望，对本文的相关工作做出总结并对下一步工作进行展望。

# 第2章 相关工作

## 2.1 Spark 分布式框架

作为2009年加州大学伯克利分校AMPLAB研究项目的一部分，Spark与Apache Hadoop仍在积极发展。Spark 2010年正式开源并成为Apache基金项目。 2015年有三个版本，从1.3到1.5，分别包含DataFrames API，SparkR和Project Tungsten等功能。版本1.6在2016年初发布，包括新的Dataset API和数据科学功能的扩展。Spark 2.0于2016年7月发布，这是一个主要版本，有很多新功能和增强功能。



**图2-1 Spark提供的主要计算框架**

图2-1是Spark计算框架包含的主要组成模块，其中Graphx主要进行并行图计算，Spark Graphx性能良好，又具有非常丰富的API，并可以基于海量数据进行复杂的图计算从而得到了较为广泛的应用。MLlib[11]作为Spark生态系统中的一部分，由AMPLab实验室的Mlbase团队开发，是Apache Spark中的核心机器学习库，专门提供机器学习相关的丰富的API，从而让机器学习的门槛更低，使那些不了解机器学习的开发人员也能通过MLlib来处理自己的数据。Spark SQL的引入专门用于处理结构化数据比如（DB、JSON）。Spark SQL不仅是SQL，它对各种数据源做了抽象，能够对各种数据源进行ETL操作，并提供丰富的对复杂数据读写的API，使用非常方便。Spark SQL对多种数据源都提供了很好的支持，其中DataFrame API作为一种具备高伸缩性的数据操作方式，相比RDD API 效率更高[12]。此外Spark SQL还提供了Catalyst Optimizer查询优化器，可以自动优化程序，使得程序更加高效。

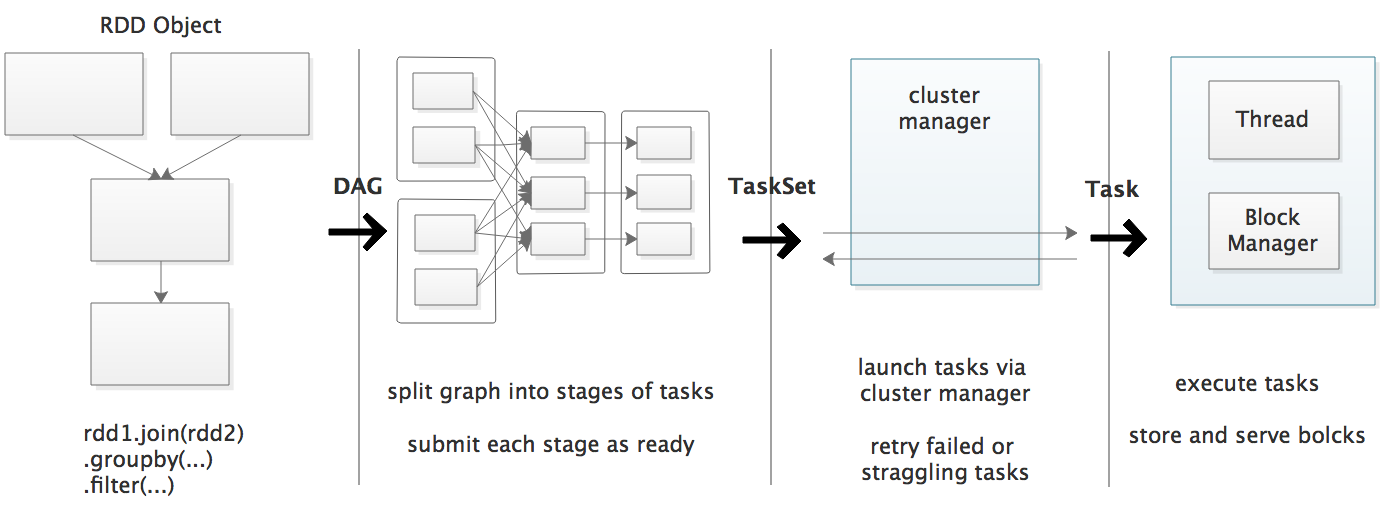
Spark引入的弹性分布式数据集RDD[13]，是一种高度受限的共享内存模型，是基于内存的。这使得其计算效率和速度相比Hadoop 的 MapReduce（基于磁盘，每次执行时都要从磁盘读取数据，计算完毕后都要把数据存放到磁盘上）提升很多。

Apache Spark是围绕速度、易用性和复杂分析构建等方面的大数据处理框架，适用于多种不同的情形的分布式应用场景，包括批处理、流处理、交互式查询等等，并大大减轻了原先需要对各种平台分别管理的负担。统一的RDD抽象和操作，基于统一的技术堆栈，使得Spark称为大数据通用计算平台。在大数据时代逐渐普及的背景下得到了越来越广泛的应用。

### 2.1.1 Spark Core技术

Spark Core是Spark整个生态系统的核心，它主要提供了一套基于RDD的API以及分布式的计算引擎。Spark采用了Master-Slave的分布式计算模型，Master作为主节点，负责任务调度，管理Worker节点，并通过RPC（远程过程调用）通知Worker 启动Executor。Worker则是计算节点，负责管理本节点资源，并向Executor结点发送执行命令，并维持与计算节点的心跳。此外Spark还引入了弹性分布式数据集RDD(Resilient Distributed Datasets)，作为spark计算的基本数据结构，它可以让用户可以显式操作数据，同时，RDD还提供了一组丰富的api来操作这些数据。下面具体介绍spark 执行流程：

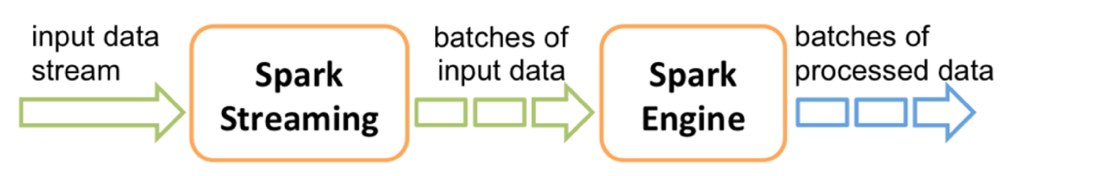
如图2-2所示，当RDD算子遇到Action方法时，DAG图（有向无环图）即确定了边界，DAG图形成，随后将DAG提交给DAGScheduler。DAGScheduler 会将DAG切分成多个stage，切分依据（宽依赖—即RDD数据需要通过网络传递）。多个stage封装到TaskSet后提交给TaskScheduler，随后TaskScheduler把任务提交给worker执行。其中DAGScheduler 和TaskScheduler都在Driver端，main函数创建SparkContext时会使得driver和Master建立连接，Master会根据任务所需资源在集群中找到符合条件的worker。随后Master对worker进行RPC通信，通知worker启动Executor，Executor会和Driver 建立连接，随后的工作worker和Master不再有关系。Driver会直接向Executor提交Task，并启动Executor执行任务。



**图2-2 Spark任务执行流程**

### 2.1.2 Spark Streaming

Spark Streaming类似于Apache Storm，主要应用于流式数据计算，具有高吞吐量和容错能力强等特点。Spark Streaming支持的数据输入源很多，例如：Kafka、Flume、Twitter、ZeroMQ和简单的TCP套接字等等。另外Spark Streaming也能和MLlib（机器学习库）以及Graphx完美融合。



**图2-3 SparkStreaming 工作流程图**

Spark Streaming 主要功能是根据时间片的长短，将数据流转换拆分到一个个的RDD算子中, 通过Spark引擎将RDD分散到各个计算节点进行批处理处理，然后整合处理结果，从而达到大规模处理流数据的作用。同时，Spark Streaming还提供了丰富的API 用于流数据的计算，开发者可以根据具体的使用场景进行灵活的选择。

### 2.1.3 Spark MLlib 技术

机器学习以及相关的统计学技术可以帮助我们从复杂的海量数据中挖掘有用的信息。对于一般开发者来说，其学习成本较大，即使了解其中的相关算法，但是要想具体实现，也是有一定困难的。因此MLlib的愿景就是解决开发者的这些问题，MLlib 构建在apache spark之上，是一个专门针对大量数据处理的通用的、快速的引擎，并实现了一些常见的机器学习算法和实用应用程序，包括分类、回归、聚类、协同过滤、降维以及相关的底层优化。因此对于开发者想要涉足机器学习领域的开发，只需要具体一定的理论知识即可。

### 2.1.4 Spark与Hadoop区别

Hadoop和Apache Spark都是大数据框架，但它们并不真正具有相同的用途。 Hadoop其本质是一个分布式数据基础架构，它在大量服务器群集中的多个节点之间分发海量数据集，这意味着用户不需要购买和维护昂贵的定制硬件。它还可以索引并跟踪数据，从而大大提高了数据处理和分析效率。另一方面，Spark是一种数据处理工具，可以对这些分布式数据集合进行操作，但它本身不具备分布式存储能力，因此需要在Hadoop的分布式文件系统或其他文件系统的基础上工作。

Hadoop不仅包括称为Hadoop分布式文件系统的存储组件，还包含一个名为MapReduce的处理组件，因此我们可以不使用Spark来进行计算。相反，我们也可以使用Spark而不使用Hadoop。Spark是一个多阶段的具有RAM能力的计算框架，具有机器学习，交互式查询和图形分析的实现库。但Spark并没有自己的文件管理系统，所以它需要与一个HDFS（而不是另一个基于云的数据平台）集成。Spark通过内存计算和处理优化加快了批处理速度，因此它的速度通常比MapReduce快得多，这使得它成为处理流式传输工作、交互式查询和基于机器学习的不错选择。Spark实时处理能力使它成为大数据分析的首选。

## 2.2 服务端相关技术

### 2.2.1 Spring框架简介

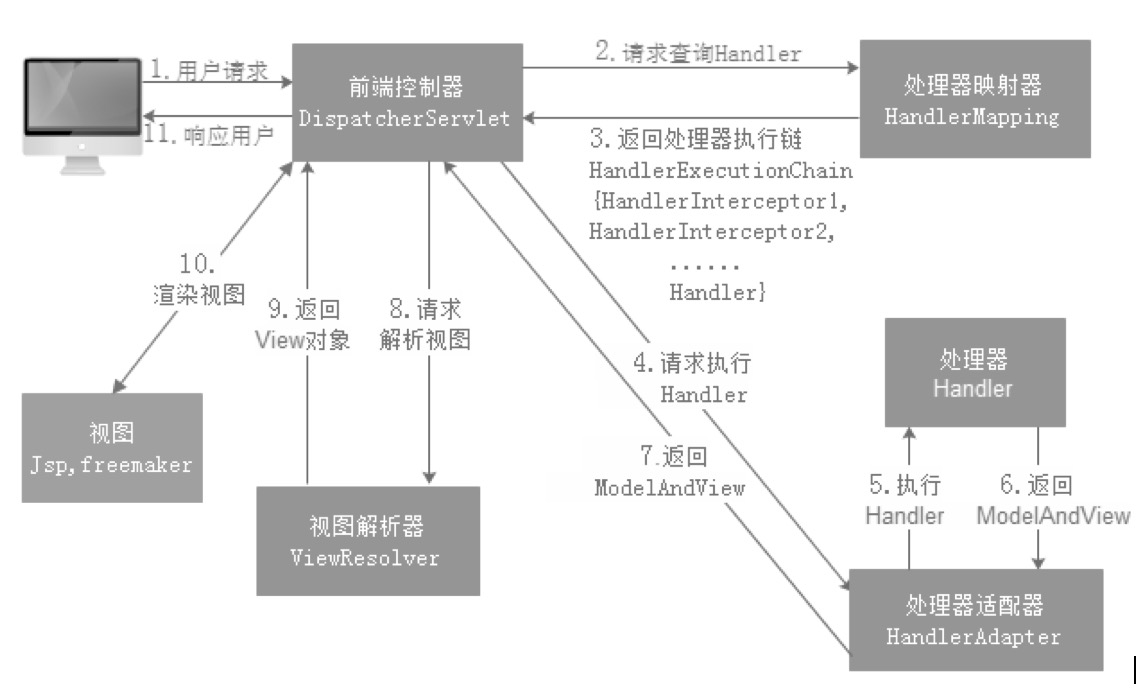
Spring是一个开放源代码的设计层面框架，它主要用户解决的业务逻辑层和其他各层的松耦合问题，因此它将面向接口的编程思想贯穿整个系统应用。商城系统的开发在Spring框架的帮助下使得各个模块的耦合度降到最低，并借助于Spring的Aop面向切面的编程方式，弥补了面向对象变成的不足，AOP的引入使得原来重复的散布在业务逻辑中的代码被横向的分割出来，例如商城系统的权限管理，用户登录检测，日志生成等操作，这些重复的代码通过AOP的方式得到重用，并且只需要简单的配置就可以类似于拔插式的添加和删除，减少了重复的相同逻辑的代码对整体逻辑结构的污染，降低了模块的耦合度。

其次Spring来提供了IOC控制反转的概念，这也是Spring的核心功能并贯穿始终，Spring通过该功能，控制管理对象的生命周期、对象的初始化时机以及对象之间的关系，并可以通过ApplicationContext类来灵活的获取容器中的对象。

Spring的功能远不至于上述几点，在此框架的基础上，使得系统的具有较高的灵活性，低耦合性和易维护。

### 2.2.2 SpringMVC框架简介

SpringMVC作为SpringFrameWork的主要成员，是Spring框架的重要的组成部分，该框架为MVC设计模式提供Controller层的技术解决方案。其主要控制流程如图2-4所示。



**图2-4 SpringMVC 请求流程图**

SpringMVC通过前端控制器DispatcherServlet接受前端的http请求，并根据xml配置去处理器映射器HandlerMapping请求查询指定的Handler，并返回处理器执行链，通过执行链去请求处理器适配器HandlerAdapter调用执行指定的Handler并返回视图对象ModelAndView。DispatcherServlet在根据视图解析器ViewResolver解析视图并返回View对象，在根据执行链将View对象解析渲染返回前端展示给用户。SpringMVC的使用使得开发人员更加轻松的包括服务接口供前端调用，摆脱了web.xml的相关配置，并使程序变得更加简洁易懂。

### 2.2.3 Nginx反向代理

Nginx作为代理服务器，在商城及推荐系统的应用中尤其重要，它可以根据服务器配置隐藏用户服务的真实服务器，间接的提高了内部各个服务器的安全性。其次Nginx可以作为负载均衡器可以根据各个服务器使用情况进行压力的均衡，根据每个服务结点的连接数和负载情况进行请求分发，将用户的请求分配给低负载的服务器来处理，均衡各个服务器的负载压力，此外它还可以灵活的配置权重，对于高性能的服务器适当的增加权重，而对于低性能服务器降低权重，来进一步均衡请求的分发，提高系统资源利用率。

此外Nginx服务器作为所有请求的入口，还具备并发请求数限制、并发连接数限制、数据压缩、数据缓存等相关功能，对于系统的稳定性和服务器性能有很强的辅助作用。

## 2.3推荐算法

### 2.3.1 推荐系统介绍

推荐系统的出现大多数用于为实时互动中的用户提供相关信息、产品的个性化推送服务。在这些系统中，特别是基于k-最邻近法[16]（k-nearest）协同过滤的系统已经在网络上取得了广泛的成功。近年来，可用信息量和网站访问量的巨大增长形成了对推荐系统的一些关键挑战。最关键的技术挑战是产生高质量的建议，并为数百万用户和项目在每秒钟内提供准确有效的建议，在面对较少的数据的情况下实现高覆盖率。在传统的协同过滤系统中，工作量随着系统参与者的数量而增加，因此需要一个即使在非常大规模的问题的情况下，也可以快速产生高质量建议的新的推荐系统技术。为了解决这些问题，我们探讨了并分析了多种推荐算法，并介绍了它们各自的优缺点、原理和应用场景。

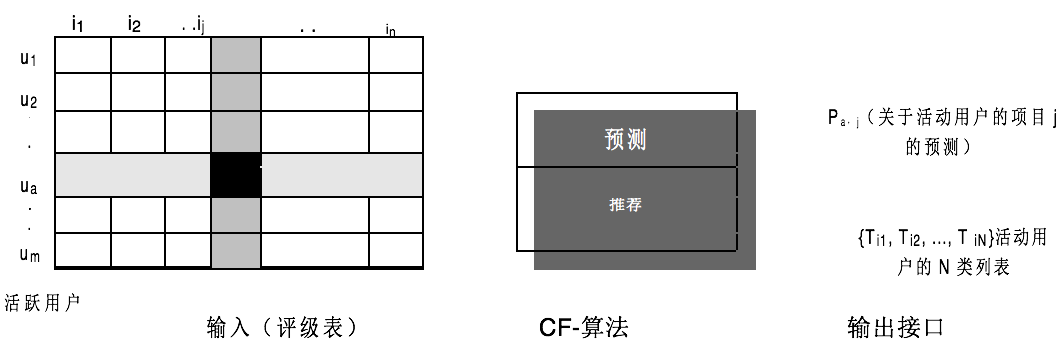
### 2.3.2 协同过滤算法概述

协同过滤算法的目标是根据用户之前的兴趣喜好和他相似用户的兴趣爱好来推荐新项目和预测用户对该项目的评分。在典型的CF场景中，有一个m个用户U={u1,u2,…..um}列表和n个项目I={i1,i2,….in}的列表。每个用户ui都有一个项目列表Iui用于来表达用户对项目的评价。用户可以通过分析定时日志、挖掘Web超链接等来将用户的意见明确地进行评分操作，通常这些分数在一定数值范围内，或者可以隐含地从记录中导出。

预测是一个数值，Pa, j，表示对于活跃用户Ua的项目ij ∉Iua的预测可能性。该预测值与Ua提供的意见值在相同的比例内。

推荐是N个项目的列表，Iτ∈I，是活跃的用户最喜欢的项目。请注意，推荐列表必须是活动用户尚未购买的项目，即。

协同过滤过程的示意图如图2-5。CF算法将整个m x n个用户项目数据表示为评级矩阵A。A中的每个条目ai, j表示第i个用户在第j个项目上的偏好分数（评分）。每个单独的评分都在数字范围内，数值为0，表示用户尚未对该项目作出评价。目前主流的协同过滤算法主要包括基于内存（基于用户）和基于模型（基于项目的）两种[17]。在本节中，我们提供了基于项目的CF的推荐系统算法的详细分析。



**图2-5协同过滤过程**

### 2.3.3 基于人口统计学的推荐

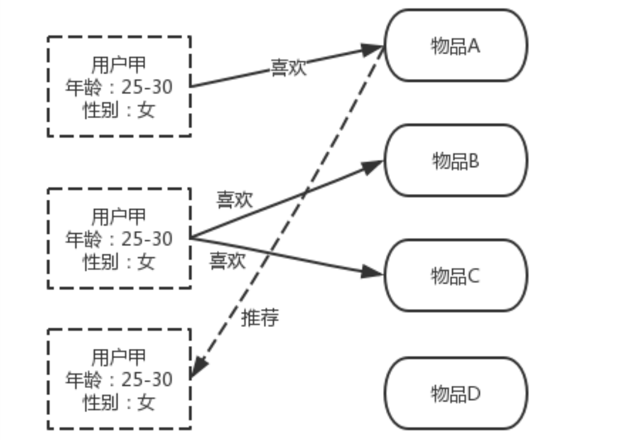
基于人口统计学[20]的推荐方法则是根据人的静态描述信息来进行推荐的方法。它是根据简单的人口统计信息来发现并得到用户的相似度，然后利用相似度寻找与该用户相近的人，并给该用户推荐这些与他相近的人所喜欢的物品。

具体的来说，这种方法首先会收集并利用用户基本的统计信息，比如年龄，性别，籍贯，住址等，其次采用特定的方法计算相异度，计算用户之间的差异。然而实际计算用户属性类别却并不是相同的，比如性别属于二元属性，年龄属于序列属性，职业属于名称属性等等。现假定数据集中一共含有p个混合的属性，那么对于对象i和j的距离，我们将会采用下面的公式计算得出：

(公式2.1)

(公式2.1) 中代表用户的属性f的指示符，则是含有用户的属性f的记录i和j之间的相异度。

如图2-6所示，左边一列分别是用户甲、乙和丙的信息，用户的属性都是名称类和二值类的属性，那么根据公式（2.3）计算后可以得到用户丙与用户甲的距离为0，和用户乙的距离为1，由此可得用户丙和用户甲之间更加相似，所以我们就可以将用户甲喜欢的东西推荐给用户丙。这里假设的邻居数只有一个，然而在实际应用中，可以通过对邻居数目做相应的调整，从而进行覆盖率更高的推荐。



**图2-6基于人口统计学的推荐**

### 2.3.4 基于内容的推荐

基于内容的推荐[21]是利用物品的内容信息，首先算出物品之间的相似度，根据用户以前喜欢的物品信息，向用户推荐与这些物品信息记录相似的类似物品。具体计算相似度的方式有很多，这里可以我们采用Jaccard公式：

(公式2.2)

或者采用简化的余弦相似度公式：

(公式2.3)

公式（2.2）和公式（2.3）中的代表两个电影u与v之间的相似度，其中代表电影u的内容，代表电影v的内容。

如图2-7所示，右侧一列为电影的类型信息，A为“爱情，浪漫”类型的电影，B为“恐怖，惊悚”类型的电影，C为“爱情，浪漫”类型的电影。利用（公式2.3）计算可以得知，电影A与电影B之间的相似度为0，电影A与电影C之间的相似度为1，由于用户甲喜欢电影A，并且电影A与电影C的相似度较高，所以系统会将电影C推荐给甲。



**图 2-7基于内容的推荐**

## 2.4本章小结

本章主要介绍了Spark的生态系统以及相关框架，并阐述了其优越性。介绍了项目中使用相关的后端技术框架以及大数据技术框架以及工作原理，并且简单介绍了几种常见的推荐系统和一些常用的推荐算法。

# 第3章 商城及推荐系统需求分析

## 3.1 系统业务需求

本系统以商城系统为主体，设计实现了商城模块的基本功能，并记录用户在各个模块的操作记录，并在该记录数据基础上通过对用户行为日志的分析，结合协同过滤推荐算法来为用户推送可能感兴趣的商品列表，从而帮助用户在海量商品中快速定位到适合自己的产品，同时也为更好的帮助商家扩展业务渠道，增加业务流水。

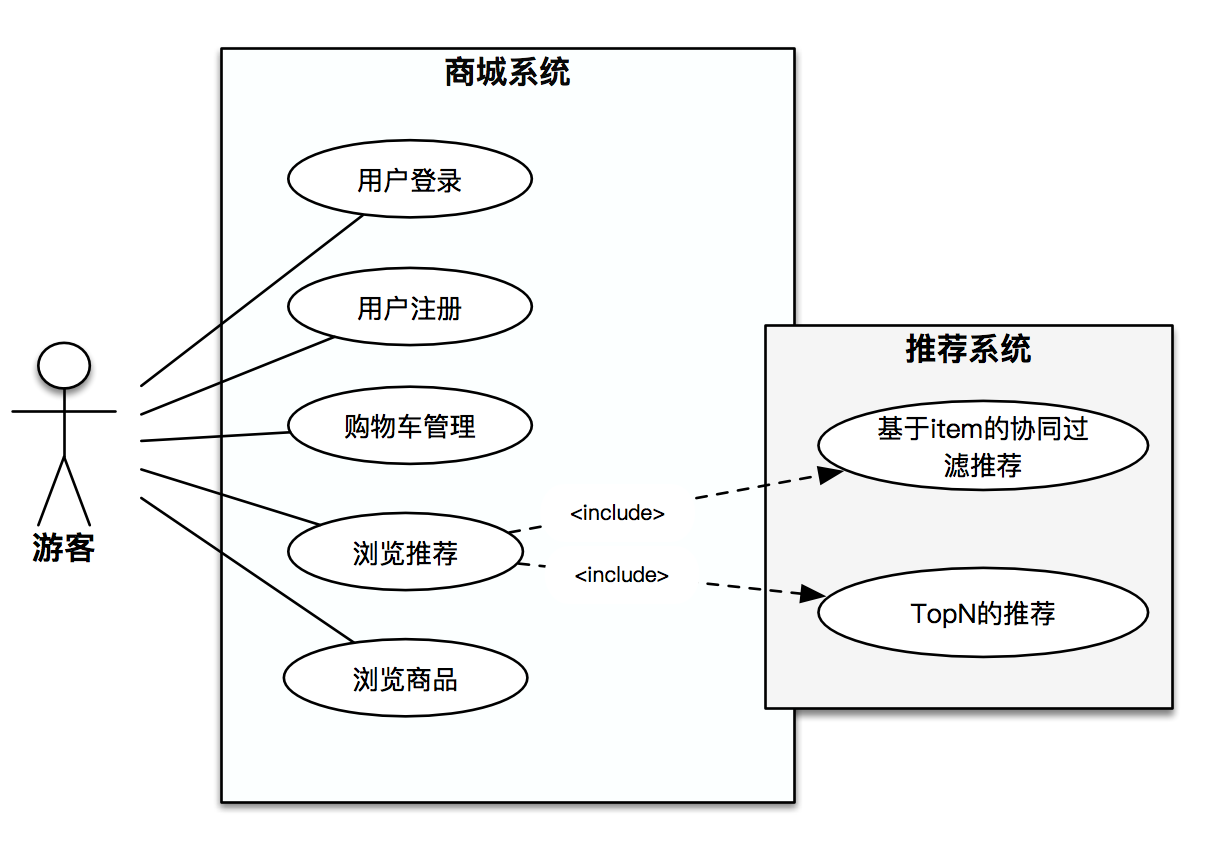
整个商城推荐系统的业务流程如下：

通过用户分析用户的行为日志信息、浏览记录、停留时间、点击数等来预测用户偏好。以用户数据库信息为基础，来进行多维度商品推荐。基于用户注册信息（性别、年龄、地区等）通过基于user的协同过滤算法给出基于用户的推荐结果。通过用户浏览、收藏、以及购物车中的商品同时结合商品标签，通过基于item的协同过滤算法给出基于商品的推荐结果，随着用户量以及商品总数的不断增加，推荐算法的准确性和时效性是考验整个系统好坏的主要标准之一。

## 3.2 系统功能性需求

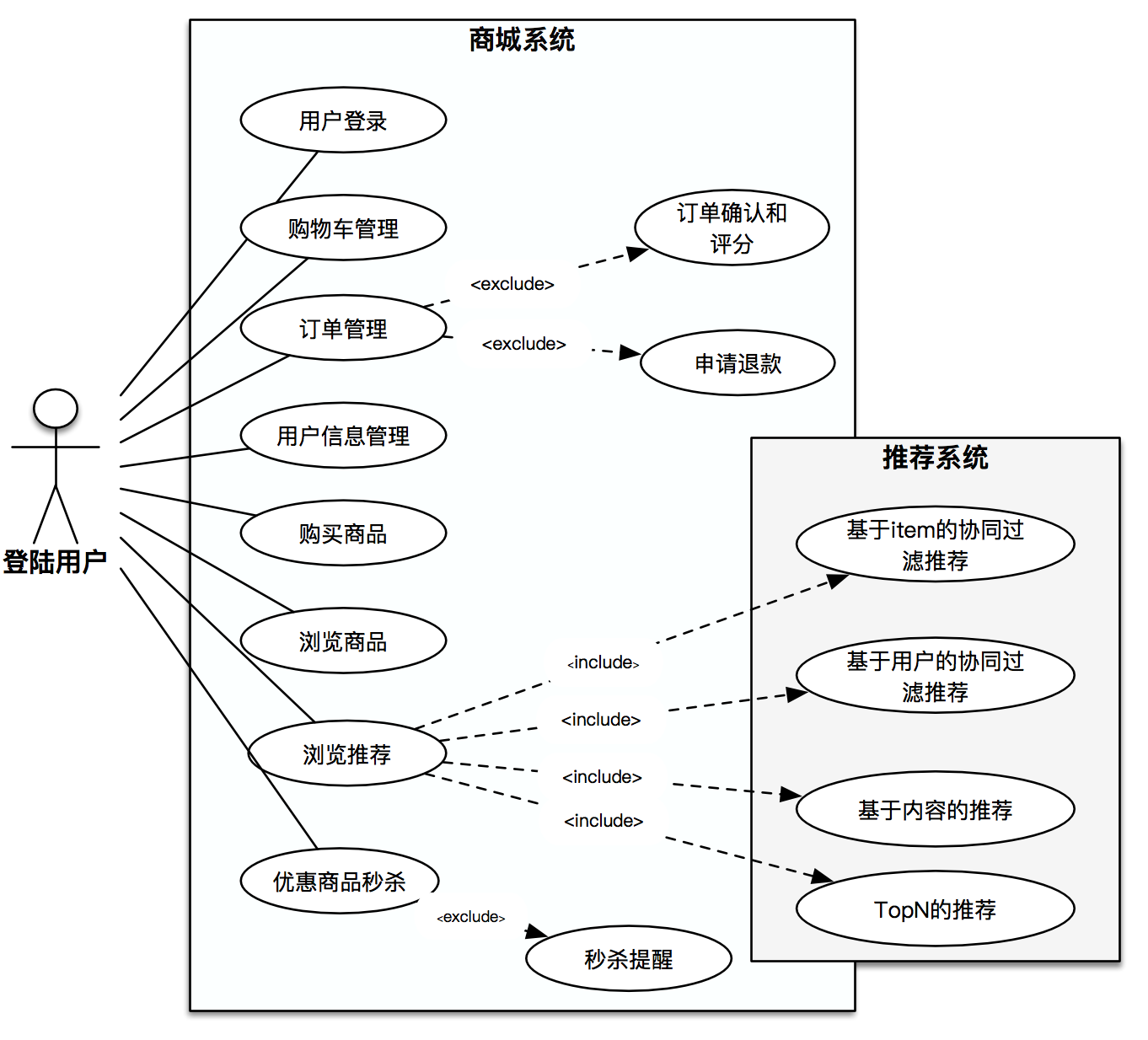
### 3.2.1 商城系统及推荐系统总体功能需求

商城系统需要满足用户日常的基本操作，包括通过注册接口填写个人信息注册账号，用户登录，调用商品详情接口浏览商品的基本信息，并希望可以对喜欢的商品进行收藏、加购物车、购买等操作，并在确认收货后，对商品进行评价和打分，和享受售后的相关服务。对于优惠促销商品，会员用户可以通过限时的秒杀的购买。整个系统的用户类型包括游客，登陆用户。推荐算法会根据不同的用户状态通过使用不同的推荐算法对用户产生的历史行为进行分析，并生成推荐结果。



**图3-1 游客状态下用户用例图**

图3-1是用户状态为游客时的用例图，由此可见，游客时可以进行商品的浏览搜索和购物车管理以及用户的登录注册，推荐系统的推荐方式只有基于item的协同过滤推荐和基于TopN的推荐方式。

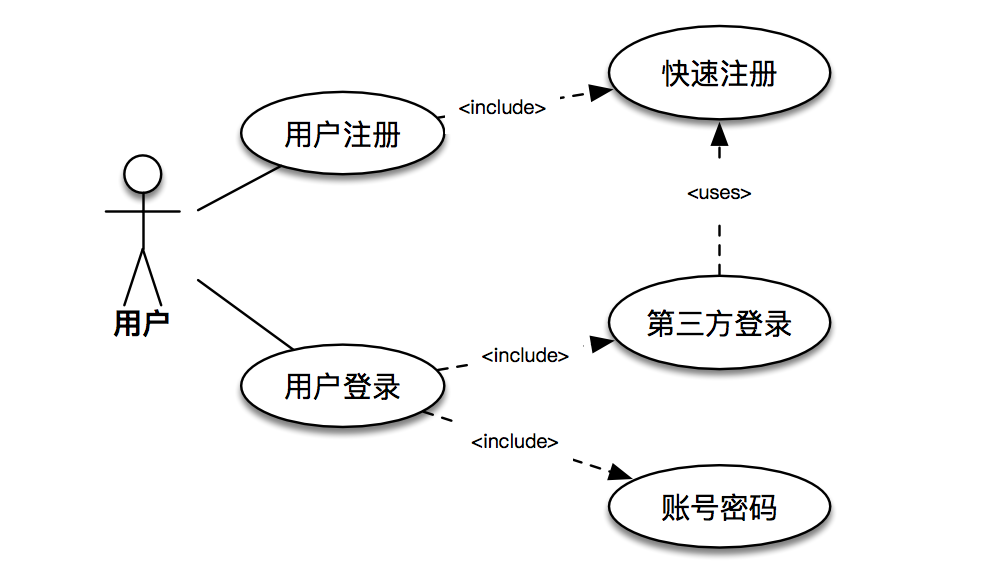


**图3-2用户登录状态用例图**

图3-2是用户已登录状态下的的用例图，相比游客状态而言，登陆后的用户可以进行个人信息编辑、购买商品、收藏商品、管理已生成订单，对于会员或满足条件的用户还可以参与优惠商品的秒杀等活动。由于登陆用户有着更加丰富的功能和行为记录，因此推荐系统也可以根据数据库中用户日志以及偏好来生成更加精准的用户推荐。

### 3.2.2 登录注册模块需求分析

登录注册功能是应用系统的基本功能之一，其中包括登录模块和注册模块，登录模块包括普通账号类型登录和第三方快捷登录，使用第三方快捷登录时，如果检测到第三方账号未绑定商城实体账号，则跳转注册界面，成功注册后，自动绑定第三方账号。具体用例图如图3-4所示。



**图3-4 登录注册模块用例图**

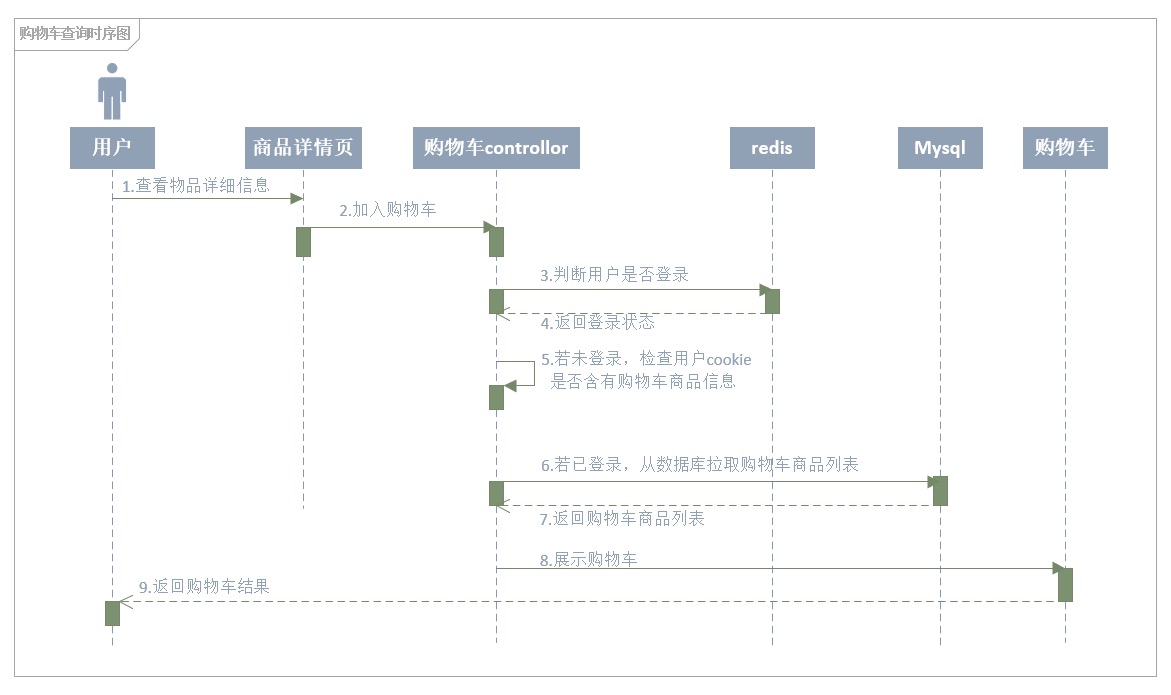
对于单一服务器应用系统而言，用户登录成功后，会在服务器端生成session保存用户相关信息，然后将用户唯一的session\_id返回，并存入客户端对应的cookie中。用户下次访问其他页面会携带此cookie，服务器会根据cookie中的session\_id来验证用户是否登录。

但是对于多服务器应用而言，我们希望用户在某一个模块上登录后，其他跨服务器模块可以获取到当前模块的登陆信息，不需要用户二次登陆，提升用户体验，然而由于每个应用模块可能分布在不同的服务器节点上，而登录产生的session只存储在登录的服务器中，其他模块如果需要获取登录状态就必须通过登录服务器将session进行共享，因此基于该需求的逻辑基础上，我们需要设计并实现类似于session共享的单点登录模块，该模块需要维护了一套用户的登陆认证机制，实现只需要用户登录一次就可以在多服务器的系统中保持登录状态。

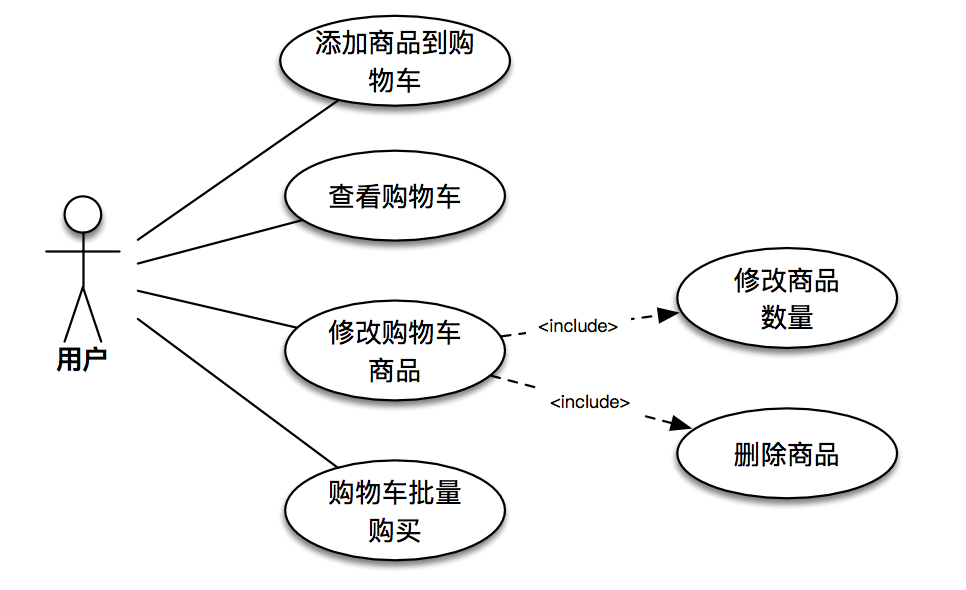
### 3.2.3 购物车模块需求分析

当用户在在商城浏览需要购买的商品时，有时需要多件商品同时购买，或者对于有意向购买的商品进行暂时的保存，可以将商品添加到购物车中。除此之外用户还需要查询购物车商品列表、修改或删除购物车商品、以及购物车商品批量购买等操作。

其中用户添加商品到购物车的具体时序图如图3-5所示：



**图3-5购物车模块时序图**

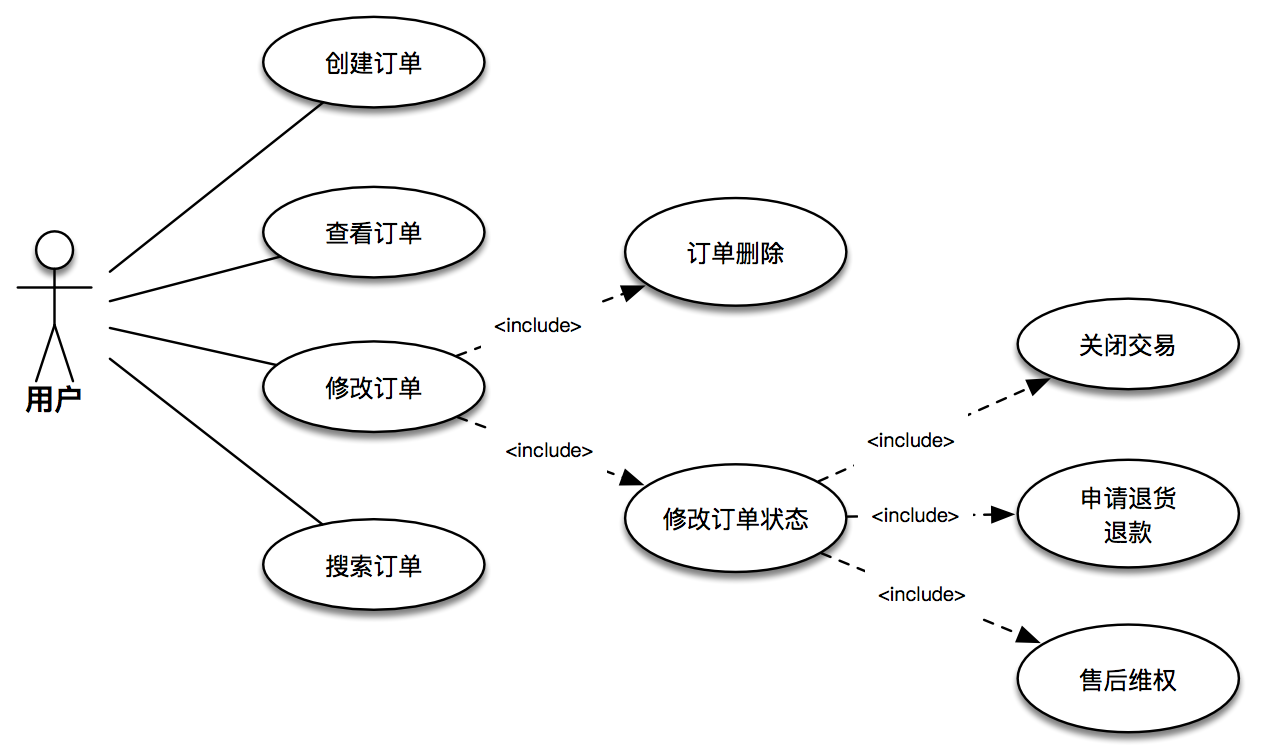


**图3-6 购物车模块用例图**

图3-6是购物车模块用例图，包括用户对购物车的相关操作，其中添加商品到购物车时，若用户未登录，会将购物车信息保存在浏览器的cookie中，若用户已登录，则将购物车信息保存在数据库中。查看购物车详情，会罗列该用户的所有购物车商品。修改购物车中的内容则包括删除购物车中的商品以及修改购物车中商品的数量等。最后我们可以批量的选取购物车中的指定商品进行购买操作。

### 3.2.4 订单管理模块需求分析

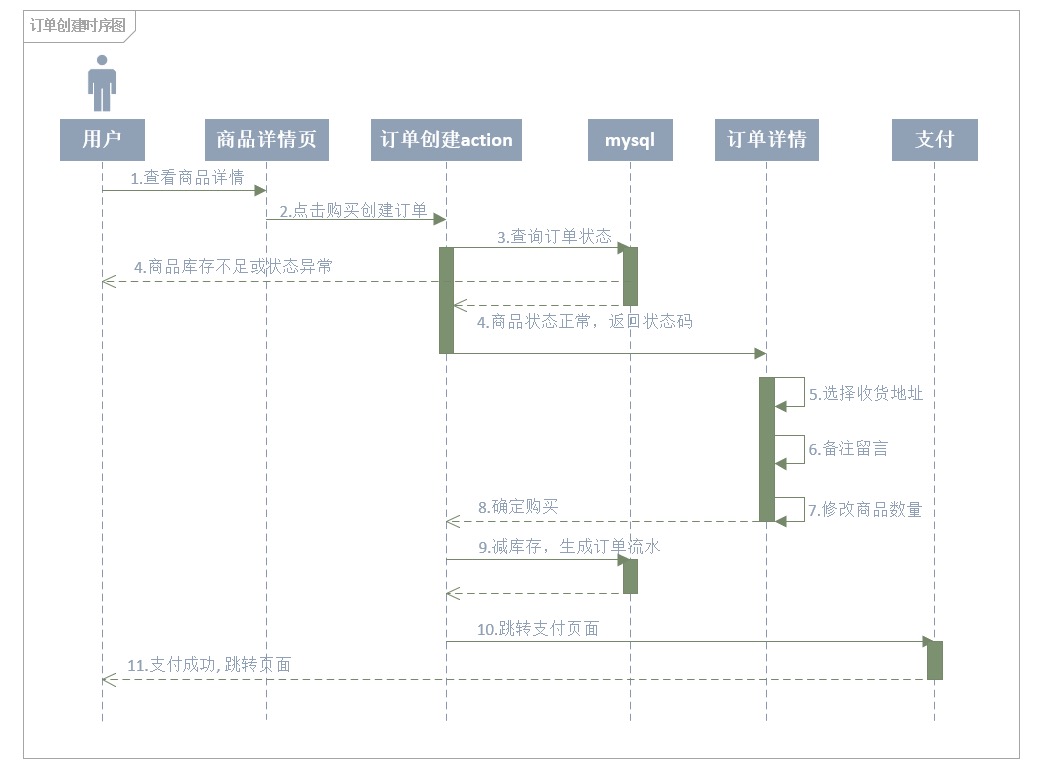
用户往往需要创建和管理商品的订单信息，订单管理模块需要用于完成相关操作，订单管理模块用例图如图3-7所示，主要需要实现的内容包括创建订单，查看用户的历史订单信息，修改当前未完成的订单。修改订单又包括关闭未支付的订单，对于已支付的订单，进行申请退款，和确认收货操作，以及对订单商品进行评价。其中当用户创建订单但没有支付该订单时，系统定时任务会每隔60s扫描用户超过24小时的订单并自定关闭。



**图3-7 订单管理模块用例图**

此外除了订单管理创建等基本功能外，该模块的操作行为会被记录并为推荐系统提供用户偏好计算的数据支持。

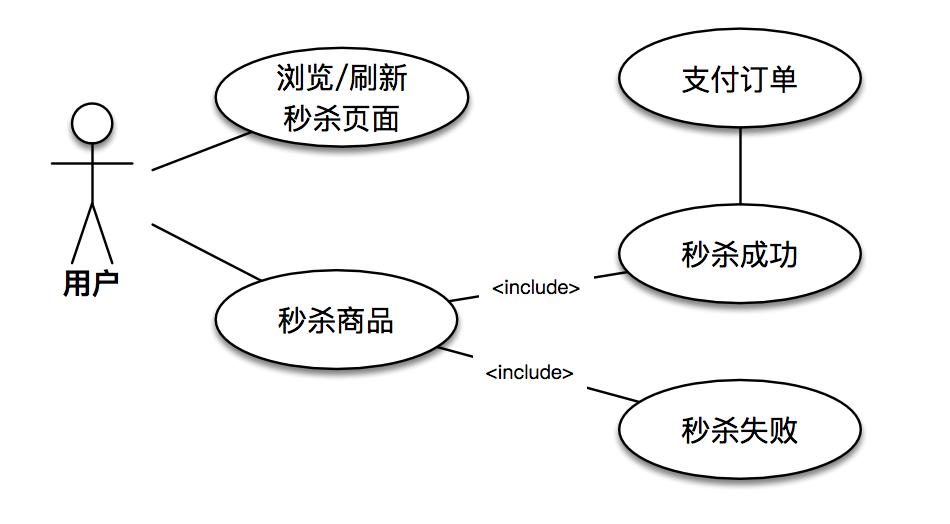
图3-8 是订单创建的流程图，订单创建过程主要需要考虑库存不足，订单创建未支付以及用户余额不足支付不成功等临界情况。



**图3-8 订单创建时序图**

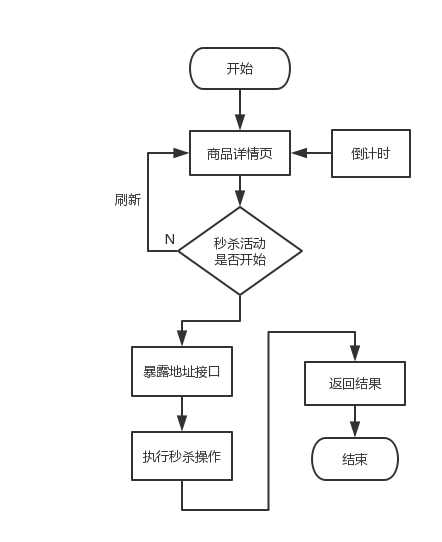
### 3.2.5 商品秒杀模块需求分析

对于限时优惠的促销商品和限量发布商品，用户往往需要通过限时秒杀的方式进行购买，为此秒杀模块需要提供相关功能，用户可以浏览刷新秒杀商品页面，并根据秒杀页面倒计时提示，及时的进行商品的秒杀操作，成功秒杀商品的用户，会生成相关的商品订单，并进入支付页面。具体用例图如图3-9所示。



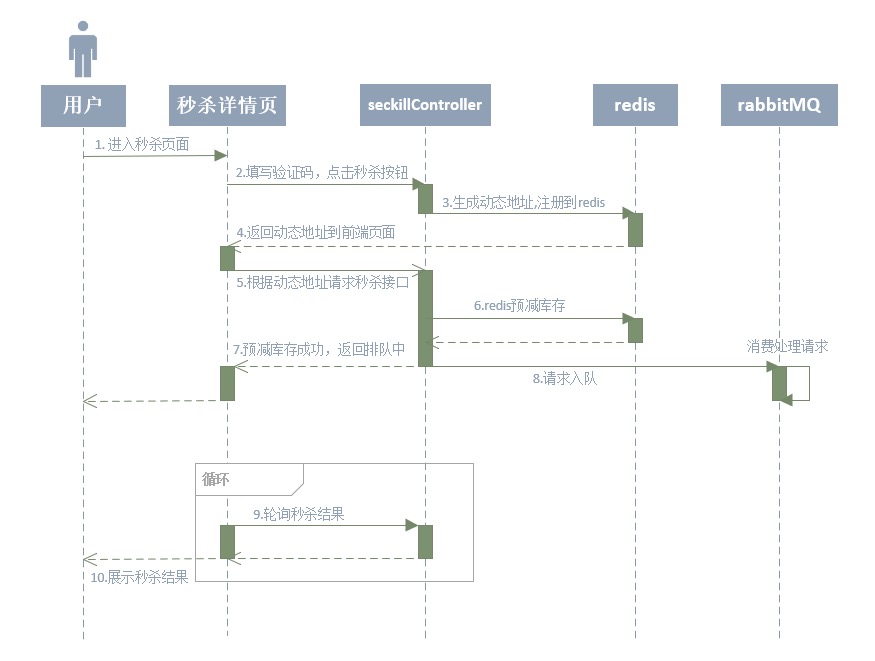
**图3-9 秒杀模块用例图**

由于秒杀服务有着较高的瞬时并发，其特殊性需要我们设计的接口能够承受较高的并发量，为此我们根据秒杀时用户的常规操作流程如图3-10，分析了系统的并发瓶颈。



**图3-10 秒杀服务用户操作流程图**

由流程图可知，系统高并发部分主要出现在商品详情页的频繁刷新和执行秒杀操作上。对于频发刷新操作，希望每次只传输数据变化的部分，尽可能少的数据传输，并将动静态数据分离，对动态数据使用Redis缓存，加速用户获取数据的速度。并使用CDN(内容分发网络)，使用离用户最近的网络节点为用户提供网页资源。对于频繁的秒杀操作，我们将服务瓶颈定位到频繁的update操作上，对于同一用户对同一个商品id进行update减库存操作时压力测试可以达到2w QPS(每秒内可以处理的查询次数)，但是对于多用户并发的修改某一商品的库存时，却变得十分低效，经过分析发现，多用户在进行减库存操作时，mysql会对该行加行级锁，这使得并行操作变为串行化，大量用户线程等待锁的释放，无法达到描述的并发需求。因此我们从减少行级锁的持有时间和减少数据库的角度出发对mysql的相关操作进行优化，从而提高系统并发量。



**图3-11 秒杀模块时序图**

根据对秒杀模块瓶颈分析，尽可能的减少对Mysql数据库的操作，我们引入了缓存，同时为了提升用户体验，减少服务器压力，将请求临时保存在队列中，异步返回秒杀结果。具体秒杀流程图如图3-11所示。

### 3.2.6 推荐系统计算框架需求分析

推荐系统的各种推荐算法需要在用户历史数据的基础上，通过分析用户行为日志和用户基础数据为用户实时的推荐可能感兴趣的商品列表，因此为了使得推荐系统的效果在实时性和准确性达到性能平衡，我们引入和离线计算和实时计算相结合的方式，下面我们对两种计算方式的业务需求进行介绍。

1. 离线计算功能需求

推荐系统在数据基础构建阶段有大量的耗时操作比如模型训练，中间数据计算（线性分类器参数），协同过滤中item的相似矩阵的计算，用户画像构建，关联规则的发现等等，这些计算通常比较复杂并且有大量的数据库查询操作，计算周期长，是算法性能的主要瓶颈，因此需要通过离线计算获得，为实时推荐提供数据基础，最后通过实时数据校正计算结果，从而提高实时推荐的效率。

(2) 实时计算功能需求

对于推荐系统而言，随着应用场景的多样性，以及许多业务场景的特殊性，对推荐系统实时性的要求日益提高，如果推荐系统不能根据用户的行为变化和商品信息的变化做出及时的结果更新，推荐结果的有效性可能会大打折扣，用户可能会因此无法获取有效的符合自身需求的推荐结果。传统的推荐方式采用定时离线计算来更新推荐结果，该方式计算周期往往以小时、天为单位。推荐给用户的商品或信息往往是滞后的、过时的，是基于用户前几个小时或者前一天的行为得到的结果，往往无法满足用户当前的需求，因此这就迫使推荐系统实时性需要有本质的提高。

### 3.2.7 基于user的协同过滤推荐模块需求分析

推荐结果的产生依据往往是多维度的，有时我们需要找到与目标用户的兴趣爱好相似的用户群体，然后将他们感兴趣同时目标用户又没有关注到的商品推荐给用户，这就可以通过基于user的协同过滤推荐算法来实现，该算法其主要思想是找到与该用户相似的用户群，然后将这些相似用户喜好的商品进行过滤筛选，去除掉用户已知和评价低和已失效的商品，然后计算用户对物品集的兴趣度，按排名由高到低给出推荐。

因此我们可以选择合适的应用场景来使用此类算法生成推荐。当新用户首次注册时，我们根据用户填写的注册信息如年龄、地区、以及兴趣爱好、以及购物偏好等通过相似度计算找到与该用户相似度较高的用户群，然后找到该用户群购买收藏度较高的商品推荐给用户。

### 3.2.8 基于item的协同过滤推荐模块需求分析

基于商品的推荐相比于基于用户的推荐而言应用较为广泛，有时我们需要为用户推荐与最近浏览和关注的商品相似的商品，该场景可以使用基于item的协同过滤算法，该算法的主要思想是通过所有用户历史喜好的物品以及最近搜索关注度较高的物品来计算物品之间的相似度，找到相似物品后预测用户评分，然后把与用户的喜好相似度较高的商品推荐给用户。

假设商品A和商品B具有较高的相似度，那么用户甲对商品A进行收藏、加购物车、或者购买操作时，我们猜测用户可能对A商品和其类似商品感兴趣，于是可以将于商品A相似的商品B推荐给该用户，从而帮助用户更好的挖掘自己可能喜欢的商品，提高浏览和购买效率。

由于商品间的相似度较为稳定，我们可以通过离线的方式预先将商品间的相似度计算并入库，算法调用时可以直接通过查表的方式获取相似度，并计算生成推荐结果。

### 3.2.9 基于内容的推荐模块需求分析

由于如今的互联网商品数量庞大，涉及范围广，而平台的商品推广区的位置有限，如何每天将所有需要推广的商品推送给每个人不仅页面无法即时展示，还会将一些用户根本不关注的商品推送给用户，导致推广效果和用户体验极差。为了针对不同的目标客户实现较为有效的商品推广，根据对之前协同过滤推荐的需求描述和介绍可知，协同过滤方式的推荐算法本身不依赖于具体的商品信息和商品描述，因此如果需要对某种类型的商品做定向的针对性的推荐仅仅使用协同过滤的方式是无法满足要求的，因此我们引入基于内容的推荐方式，该方式在商品内容的数据基础上，通过数据挖掘的方式完成商品以及用户的特征提取。从而根据不同用户的历史喜好，按照一定策略学习用户的兴趣偏好，进而从推广商品中筛选中用户可能感兴趣的商品推送给用户。

基于内容的推荐方式，对于平台而言，不同的用户获取到的推广数据各不相同，实现了’千人千面’的展示效果，对于商家而言，该方式有助于较为精准的投放广告，增加商品推广命中率，减少广告投放成本，并吸引潜在的客户群体。对于互联网用户而言，可以为用户提供体验个性化的商城主页，帮助用户挖掘更加优质、感兴趣的商品，提升浏览效率和购物体验。

## 3.3 系统非功能性需求

### 3.3.1 响应时间

整个系统的服务响应时间将会直接影响用户体验，因此要求各个服务接口要有足够快响应速度，包括页面的加载以及其他基本操作。此外服务端的并发处理能力也直接或间接影响服务的响应时间，因此无论是服务器架构还是代码的设计都充分考虑到系统并发性和响应时间这个性能指标。

此外推荐系统的推荐结果也有着较高的时效性，用户可能在某一时间段中对某类商品感兴趣，而超出该时间段后可能会丧失对目标商品的兴趣度，因此需要推荐系统在有效的时间范围内给出推荐结果。

### 3.3.2 稳定性

服务系统的稳定是平台的重要基础，整个系统需要在不同的工作环境维持整成的可靠的工作，但是服务的实际运行过程中往往会出现很多无法预计的突发状况，比如线程池处理缓慢，导致请求等待，请求无响应，又或者数据库服务器宕机、应用服务器宕机、缓存服务崩溃等，因此就要求系统局部的故障不应该影响整体的对外服务，这就需要系统具备较高的稳定性和高可用。

而对于推荐系统的分布式数据集的容错方式一般来说有两种，数据检查点和记录数据的更新。Spark采用记录更新的方式。但是如果更新粒度太细太多，则记录更新成本也会相应的增加。因此spark提供的弹性分布式数据集RDD具有数据流模型的特点：自动容错，位置感知调度和良好的可伸缩性。与分布式共享内存系统需要付出高昂代价的检查点以及回滚机制不同，RDD通过Lineage来创建丢失的分区：RDD记录了单个块上执行的单个操作，每个RDD都记录了他变换的所有操作，以便恢复丢失的分区，这种容错机制有着较低的资源开销和一定的性能提升。

### 3.3.3 健壮性

系统的执行过程时时刻刻都可能有异常现象的发生，发生的因素可能是由于错误的服务调用，也可能是非法的恶意的请求。而当异常发生时，从用户的角度出发，我们需要充分考虑发生异常的因素，并提供友好的页面提示，提升用户体验和用户满意度。 此外还需要详细的记录异常发生的数据执行链路，便于异常的排查和处理，并尽可能规避类似现象的发生。

### 3.3.4 冷启动问题

推荐系统的冷启动问题主要包括新项目冷启动和新用户冷启动两个方面。而对于新用户的冷启动问题一般可以通过利用用户在其他地方已经沉淀的数据和用户兴趣偏好进行冷启动或者制造选项生成粗粒度的推荐等方式进行。这里着重解决新项目的冷启动问题，在此我们引入了用户时间权重的概念，通过对用户评价的时间间隔判断用户活跃度并根据用户对新项目偏爱程度等因素创建构建个性化的预测评分公式。由于新项目不存在任何一个用户对它的评价信息，故无法使用协同过滤算法来实现推荐，因此，本章节提出解决该类冷启动问题推荐算法，该算法不仅仅使用用户对物品的评分信息对来进行推荐，而是要综合考虑项目本身的诸多属性信息、标签对项目的标注信息等较为全面的考虑影响推荐系统的诸多因素。实验证明该方式，在解决推荐系统在新项目的冷启动的问题上是较为有效的。

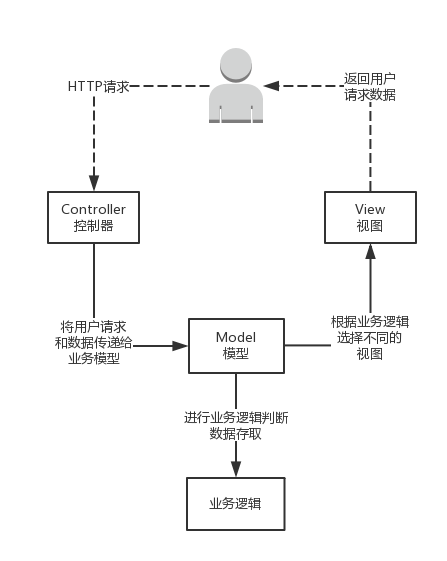
## 3.4 本章小结

本章对于商城及推荐系统的各个模块进行了详细的需求分析，分别包括商城系统的相关模块的介绍，用例图以及流程图的介绍，以及推荐系统的设计和不同推荐模块的应用场景，最后介绍了整个系统的非功能性需求，包括系统的响应时间、稳定性、健壮性、以及推荐系统的冷启动问题。

# 第4章 商城及推荐系统概要设计

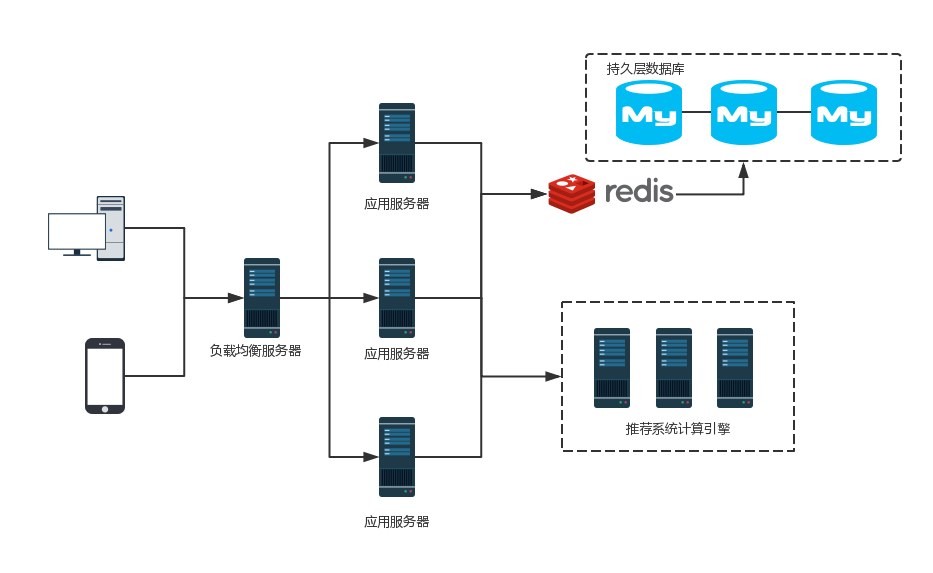
## 4.1 系统总体结构设计

整个商城系统是在Springmvc、Spring、Mybatis三大框架基础上搭建而成，采用MVC(Model View Controller) 的设计模式，其中Model模型层所管理的对象直接与数据库表相关联，负责数据的存取；View层为视图层，主要用于为系统提供图形化页面和数据显示；Controller控制器层负责具体的业务模块流程的控制，为外部请求返回相对应的视图。MVC设计模式层次关系如下图所示。



**图4-1 MVC模式层次关系图**

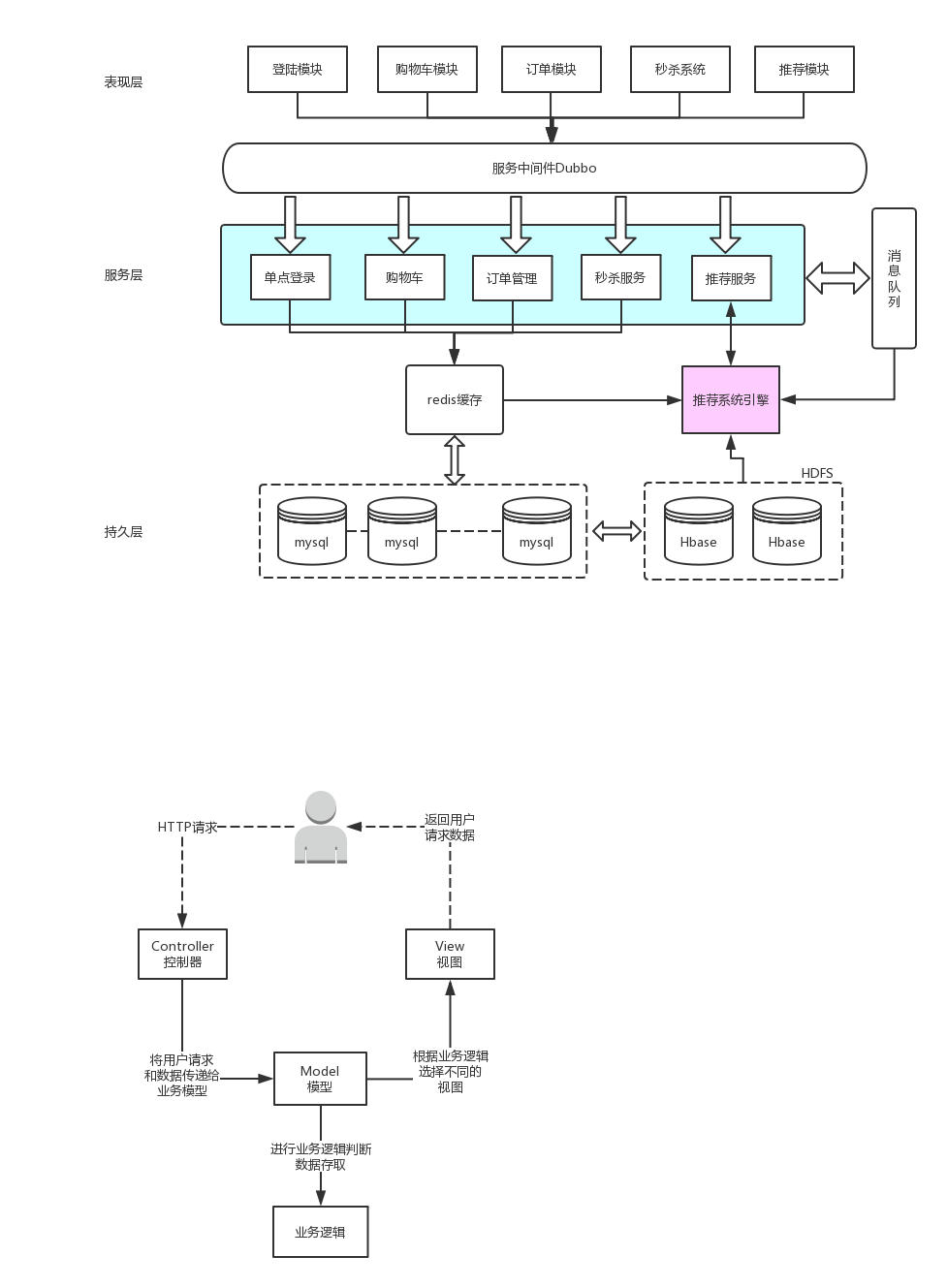
整个系统架构图如图4-2所示。



**图4-2系统架构图**

## 4.2 系统功能模块设计

该商城系统的技术架构主要分为三层，包括表现层、服务层、以及持久层，具体架构图如下图所示：



**图4-3商城系统技术架构图**

其中，表现层主要包括基础的商城功能模块，包括用户注册、登录、以及前端的商品页面显示；用户订单系统，主要用于用户管理商品订单，以及确认收货，申请退款，订单评价等相关操作；会员系统为会员用户提供相关服务和优惠活动；秒杀系统为优惠商品提供高并发的秒杀服务接口；后台管理系统主要用于管理员进行商品发布，订单处理，用户管理等；最后的推荐系统，以用户在商城系统中产生的日志数据为基础，通过对用户的行为分析，结合用户历史标签信息，和当前最新的热门商品帮助用户挖掘可能当前可能感兴趣的商品，并推送给用户，从而实现用户与商家共赢。

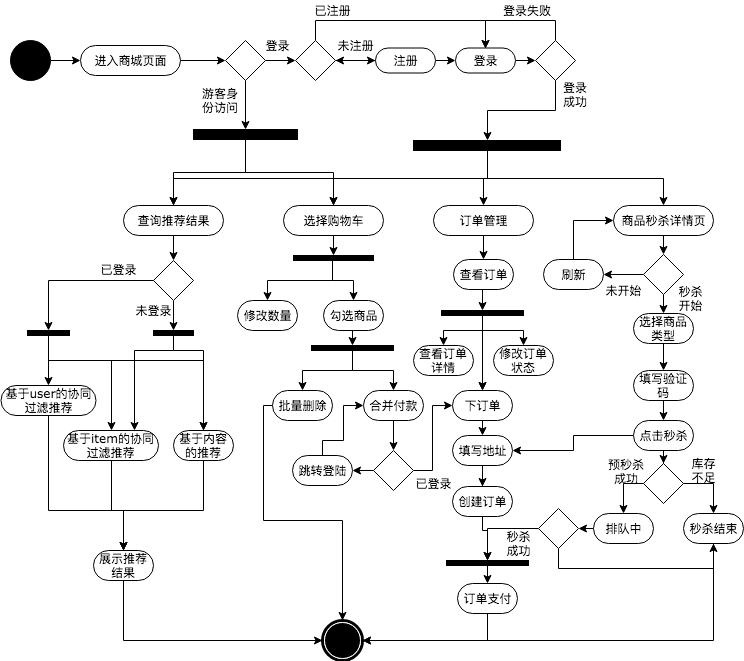
服务层主要用于实现表现层相关的业务逻辑，根据不同的用户类型开放不同的用户权限，对于游客而言只能浏览和搜索商品，并可以添加物品到购物车中，推荐系统会根据短暂的用户的浏览记录以及购物车的商品，生成用户可能感兴趣的商品列表。对于已经登录的普通用户，除了包含游客用户的基本操作功能之外，还可以管理用户个人信息以及商品的订单，此时推荐系统可以掌握更多的用户数据和历史兴趣偏好，推荐的结果比游客状态更为精准。

持久层主要有mysql集群以及分布式存储构成，为服务层提供数据基础。其中mysql集群用来存储主要的商品以及用户数据，而Hbase主要用于存储历史数据和大量的用户日志信息等。

## 4.3 系统处理流程

结合4.2节对系统功能模块的介绍，下面根据图4-4对各个模块间的活动关系和处理流程作出介绍。

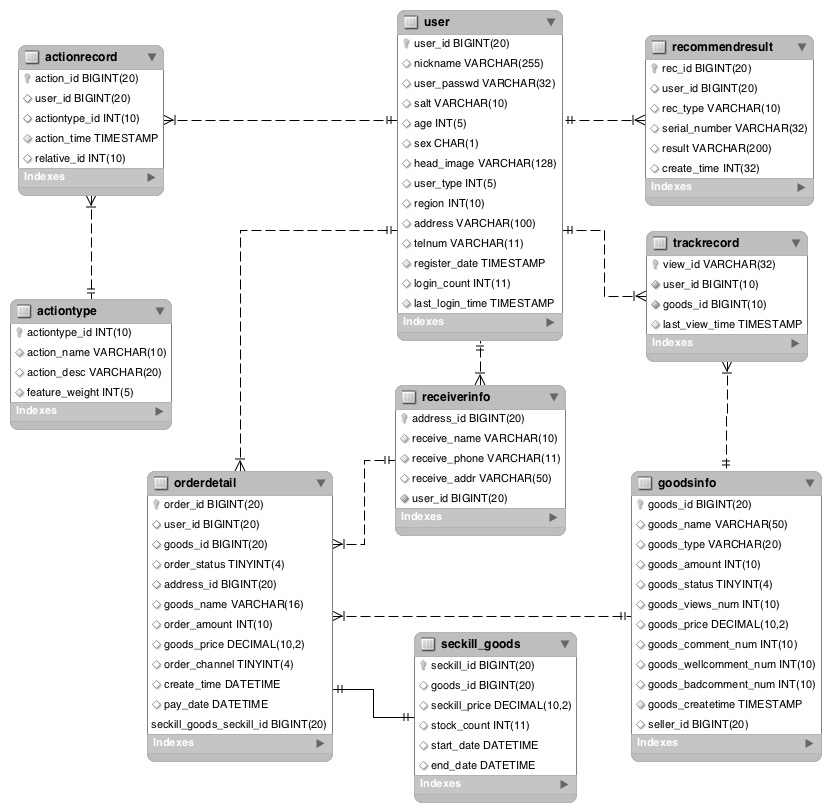
用户可以选择游客和已登录两种状态访问系统，游客状态的用户无法购买商品、访问订单管理页面和参与活动商品秒杀，此外游客状态只支持基于内容的推荐和基于item协同过滤推荐。具体模块的实现会在第五章详细设计部分一一展开。



**图4-4 整体功能模块活动图**

## 4.3 数据库设计

  对于商城的基本数据存储我们使用了mysql数据库，并对于热点数据进行分库分表以增强数据库访问性能，所有的数据库表基于满足三范式的设计要求，但由于实际业务需求的需要，降低对数据库连表查询所带来的性能的降低，适当根据需要在特殊的表中冗余数据，方便查询，并在后续的代码实现中对相关冗余字段的增删改以开启事务的方式进行处理，对数据一致性进行严格要求。此外各个表之间既关联又独立，为了方便分离，放弃使用外键，由于整个系统数据库比较复杂，表数量较多，在此我们仅选取局部重要且相互关系紧凑的表进行介绍，并导出E-R图如图4-5所示。



**图4-5 商场系统相关模块E-R图**

下面我们就对具体的表以及表中字段进行展开介绍。

（1）用户基本信息表，该表中记录了用户基本信息如用户名，用户登陆密码，以及用户的唯一表示user\_id，还有与推荐相关的用户年龄性别以及省市区域等。

**表4.1用户基本信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 非空 | 备注 |
| user\_id | bigint(20) | Y | 用户ID(PK) |
| nickname | varchar(20) | Y | 用户名 |
| user\_passwd | varchar(20) | Y | 用户密码 |
| salt | varchar(10) | Y | 用于md5加密 |
| age | int(5) | N | 用户年龄 |
| sex | char | N | 用户性别 |
| user\_type | int(5) | N | 用户类型 |
| region | int(10) | N | 用户省市区域 |
| address | varchar(100) | N | 用户详细地址 |
| telnum | varchar(11) | Y | 用户手机号 |
| register\_date | timestamp | Y | 用户创建时间 |
| login\_count | int(11) | N | 登陆次数 |
| last\_login\_time | timestamp | N | 最后登陆时间 |

（2）商品信息表，其中goods\_id为自增主键，字段主要包含商品的基本信息，以及用户浏览和评价的相关的相关信息。

**表4.2 商品信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 非空 | 备注 |
| goods\_id | bigint(20) | Y | 商品ID(PK) |
| goods\_name | varchar(50) | Y | 商品名称 |
| goods\_price | decimal(10,2) | Y | 商品价格 |
| goods\_status | tinyint(4) | Y | 商品状态 |
| goods\_type | varchar(20) | Y | 商品类型 |
| goods\_amount | int(10) | Y | 商品总数 |
| goods\_views\_num | int(10) | N | 商品浏览总数 |
| goods\_comment\_num | int(10) | N | 商品评价总数 |
| goods\_wellcomment\_num | int(10) | N | 好评总数 |
| goods\_badcomment\_num | int(10) | N | 差评总数 |
| goods\_createtime | timestamp | Y | 商品创建时间 |
| seller\_id | int(10) | Y | 商家id |

（3）秒杀商品表，seckill\_id字段是秒杀商品的唯一主键，通过goods\_id关联商品信息表来获取秒杀商品的详细信息，seckill\_price是当前商品的秒杀价格，start\_date和end\_date分别是秒杀的开始和结束时间。

**表4.4 秒杀商品信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 非空 | 备注 |
| seckill\_id | bigint(20) | Y | 秒杀商品id(PK) |
| goods\_id | bigint(20) | Y | 关联的商品id |
| seckill\_price | decimal(10,2) | Y | 秒杀价格 |
| stock\_count | int(11) | Y | 秒杀商品库存 |
| start\_date | datetime | Y | 秒杀开始时间 |
| end\_date | datetime | Y | 秒杀结束时间 |

（4）商品浏览记录表，由自增主键view\_id作为记录的唯一表示，由user\_id和goods\_id作为外键来唯一的确定用户所浏览的商品。

**表4.4 商品浏览记录表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 非空 | 备注 |
| view\_id | varchar(32) | Y | 浏览记录id(PK) |
| user\_id | bigint(20) | Y | 用户ID |
| goods\_id | bigint (20) | Y | 商品ID |
| last\_view\_time | timestamp | Y | 最后浏览时间 |

（5）订单记录，该表记录的用户的订单详情，包括商品对应的goods\_id，以及订单的数量和总计以及当前订单的状态等。

**表4.5 订单记录表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 非空 | 备注 |
| order\_id | bigint(20) | Y | 订单ID(PK) |
| user\_id | bigint(20) | Y | 订单所属用户ID |
| goods\_id | bigint(20) | Y | 商品ID |
| order\_status | tinyint(4) | Y | 订单状态 |
| address\_id | int(10) | Y | 收货地址ID |
| goods\_name | varchar(16) | Y | 商品名称（冗余） |
| order\_amount | int(10) | Y | 订单商品数量 |
| order\_price | decimal(10,2) | Y | 订单总价 |
| order\_channel | tinyint | Y | 订单下单渠道 |
| create\_time | timestamp | Y | 订单创建时间 |
| pay\_date | timestamp | N | 订单支付时间 |

（6）收货地址表，如表4.5所示，该表保存的用户的所有收货地址，与用户表构成一对多的关系。

**表4.6 收货地址表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 非空 | 备注 |
| address\_id | bigint(20) | Y | 地址ID(PK) |
| receive\_name | varchar(10) | Y | 收件人姓名 |
| reveive\_phone | varchar(16) | Y | 收件人手机号 |
| reveive\_addr | varchar(50) | Y | 收件地址 |
| user\_id | bigint(20) | Y | 地址所属用户的id |

（7）用户行为记录表，该表记录了用户的操作行为，以actiontype\_id作为外键与行为类型表做关联，来记录用户的购买、收藏、评论、搜索等操作，是推荐系统生成用户喜欢的重要依据。

**表4.7 用户行为记录表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 非空 | 备注 |
| action\_id | bigint(20) | Y | 记录id(PK) |
| user\_id | bigint(20) | Y | 用户id |
| actiontype\_id | int(5) | Y | 操作类型 |
| action\_time | timestamp | Y | 操作时间 |
| relative\_id | int(10) | N | 行为关联id |

（8）行为类型表，该表内容由管理员手动添加，记录了所有需要后续持久化的行为类型，以及对应行为描述，对于不同的行为，设定不同的推荐权重来设置该行为对后期推荐的影响。

**表4.8 行为类型表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 非空 | 备注 |
| actiontype\_id | bigint(20) | Y | 行为类型id(PK) |
| action\_name | varchar(10) | Y | 行为名称 |
| action\_desc | varchar(20) | N | 行为描述 |
| feature\_weight | int(5) | Y | 行为权重 |

（9）推荐结果表，用于保存推荐系统每次生成的推荐结果，user\_id用于关联user表，rec\_type为推荐算法类型，rec\_snapshot用来保存推荐结果快照(json)。

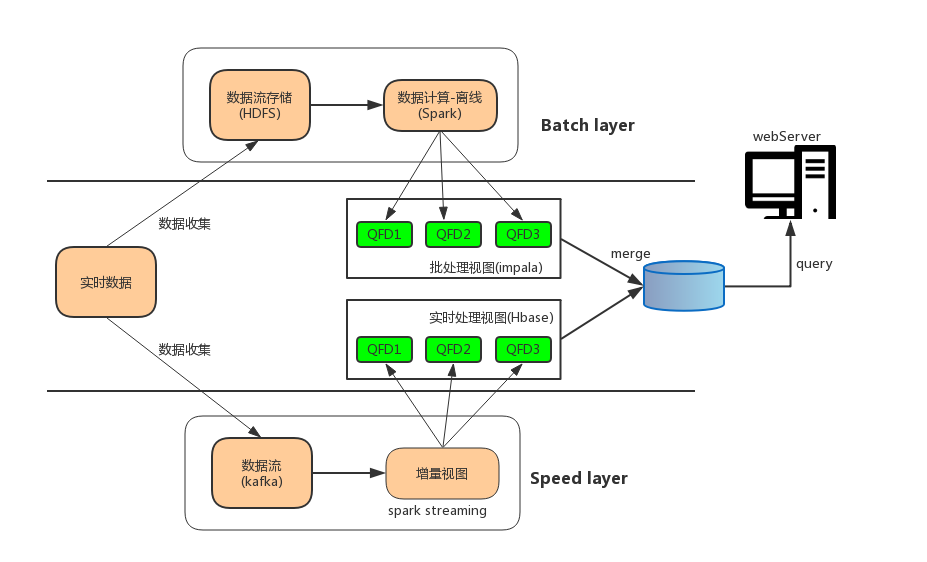
**表4.9 推荐结果表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 非空 | 备注 |
| rec\_id | bigint(20) | Y | 推荐结果id(PK) |
| user\_id | bigint(20) | Y | 用户id |
| rec\_type | tinyint | Y | 推荐类型 |
| serial\_number | varchar(32) | Y | 序列号 |
| rec\_snapshot | varchar(255) | Y | 推荐结果 |
| create\_time | timetimp | Y | 推荐 |

## 4.4 推荐系统架构设计

推荐系统主要用于整合用户数据，进行相关计算，最后为应用层提供推荐服务接口，供应用层调用。该系统整体架构主要包含批处理层、实时处理层，数据存储层、应用层。其中批处理层用户进行复杂的耗时的数据处理，实时处理层主要是从消息队列中选择部分数据通过流计算快速获取一个临时解，从而实现准实时的推荐效果。数据存储层用于读取批处理层和实时处理层通过一定的合并策略（择优选择，累加等）将数据合并，支持随机读，可以在较短的时间内返回查询结果。应用层主要用于推荐结果的展示等一系列的业务逻辑处理。

该架构设计整合了大数据组件，具有高容错、低延时和可扩展性等特性。详细架构图详见图4-6。



**图4-6推荐系统模块架构图**

推荐系统通过实时收集用户相关操作（如浏览、搜索、收藏、评论、购买、加购物车）结合相关的推荐算法，生成推荐结果。

## 4.5 非功能性需求设计

在系统非功能需要分析章节结合系统的实际需求提出了对系统响应时间、稳定性、健壮性、以及推荐系统冷启动等方面需要作出响应的优化和改善下面我们对各个部分的设计作出扼要介绍。

1. 响应时间

对于系统响应时间方面，从两个角度来设计提高系统的响应性能，首先从服务器角度而言，我们对Tomcat服务器进行了相关的优化配置，通过minSpareThreads和maxThreads适当增加服务器空闲连接数和并发处理线程数。此外对于nginx负载均衡服务器端，增加了gzip压缩功能的相关配置，加速数据传输速度。从代码层面而言，使用前后端分离的设计方式，对于频繁访问的数据使用访问缓存的方式获取，数据更新的同时更新缓存，来保证数据的一致性，页面加载采用异步的方式获取，以此来提高页面的响应速度，减少服务器连接数。

1. 稳定性

针对之前对系统稳定性的需求分析，我们针对服务器增加Zookeeper监控，当集群中某个服务节点宕机，集群会动态感知并获取该服务器信息，并发送告警信息，并将该节点从服务节点列表中移除，由其他节点代替该节点继续处理请求。此外我们对部分服务采用熔断降级的方式，引入Netflix的开源组件Hystrix实时监控接口的健康度，若接口调用的失败率很高，到达了熔断条件，则开启服务熔断机制，熔断开启后，以一定的时间间隔试探性的释放请求到失败接口，探测接口是否恢复，如果恢复则关闭熔断，以此来实现对相关请求提供有损服务，保证服务的柔性可用，避免整个系统的崩溃。

1. 健壮性

系统健壮性主要表现在用户非常规操作下的系统能够正常的返回信息，处理请求。在代码层面，我们做了充分的异常处理，并以严格的日志格式做出记录，当系统出现异常，系统并不会崩溃，而是返回用户友好的提示信息，并将异常详情记入日志文件。

此外我们还通过监控反馈的方式，对于重要的异常及时报警，通过人工或自动的方式处理。

1. 冷启动

冷启动问题[31]有很多解决方法，本文采用的方法主要是将时间信息加入到推荐算法中。我们都知道，现实世界中，用户群体一般分为两类：一类用户即所谓的活跃用户，这类用户的特点是喜欢追逐并评价新鲜事物；另一类用户即所谓的消极用户，这类用户更多的是追随角色。在这种情况下，积极用户对项目的评价时间往往在项目发布之后很短的一段时间内，而消极用户的反应则要相对滞后。由于这两类用户的普遍存在，推荐系统应用的时候会考虑把新项目优先推荐给积极用户，以此来带动评价。我们同时把时间信息[33]加入到系统中后，冷启动问题也就得到了解决。

由于时间信息对推荐算法有重要的影响，因此我们对用户的时间权重作了如下定义:

时间权重=

其中sum表示用户评价过的项目总数；timeui表示用户评价的第i个项目的时间，datai表示项目i发布的时间。则用户u的时间权重Wu计算公式如下：

(公式4-1)

由此可知，timeui – datai越小，说明项目评价时间与项目发布时间的距离越近，用户越积极。反之亦然。

与实际项目为例，用户评价时间表如下：

表4.8用户评价时间表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Item | U1 | | U2 | | U3 | |
| Time | Date | Time | Date | Time | Date |
| i1 | 2016 | 2015 | 2015 | 2014 | 2017 | 2017 |
| i2 | 2016 | 2014 | 2016 | 2013 | 2017 | 2014 |
| i3 | 2015 | 2012 | 2017 | 2014 | 2017 | 2016 |

以表一为例，采用公式1计算得到每个用户的时间权重wui即：

根据如上计算可知，用户积极程度由大到小依次是Wu1>Wu3>Wu2，即用户Wu1更偏爱于新发布的产品。

此外我们还根据项目标签信息预测评分值[34]。

1) 基于项目评分信息预测评分值

这里，我们假定有一个用户u，分析用户-项目评分矩阵可以发现用户与项目之间存在评价与未评价，以及评价次数的关系。对于评价与未评价的关系，我们设置一个变量ruj，当ruj的值为1时表示用户评价过该项目，0则表示未做评价。对于评价次数，本文设定以ICk来表示用户评价的个数，UCj表示某一项目被共同评价的次数。在这个关系中，用户评价项目的个数和项目被评价次数可以从用户-项目评分矩阵中得到。根据项目的已评价总数、评分具体信息和用户对项目的评价次数，我们可以定义基于项目评分信息的预测评分值，从而观察和研究他们间的关系。具体公式如下:

(公式4-2)

2) 基于项目属性信息预测评分值

如公式所示表示基于用户u对项目j的预测评分值。如果用户u 对项目 j 做了评价ruj =1, 反之为0。如果项目j具有属性l，则hlj=1，反之为0。

(公式4-3)

3) 基于项目标签信息预测评分值

基于项目标签信息的预测评分计算公式如下：

(公式4-4)

公式(4-4) 中，表示基于标签的用户u对项目j的预测评分值。如果标签g标注的项目j，则bgj=1分之为0。

下面我们基于具体的数据矩阵和公式来围绕研究对象u1计算预测评分值。

**表4.9用户项目评分矩阵**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| rij | i1 | i2 | i3 | i4 |
| u1 | 2 | 0 | 4 | 0 |
| u2 | 0 | 3 | 1 | 0 |
| u3 | 5 | 0 | 2 | 0 |

表4-5 标签-项目标注矩阵

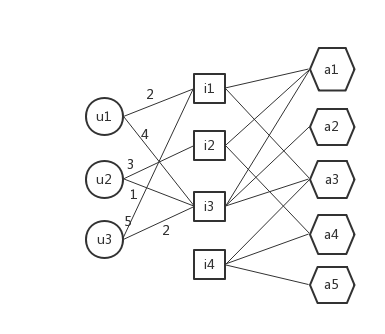
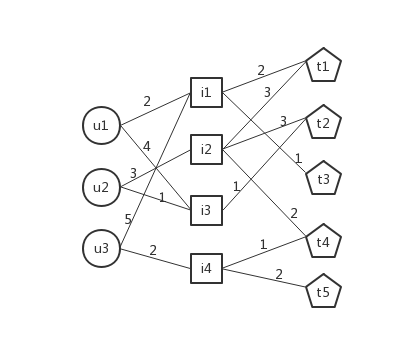
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bij | i1 | i2 | i3 | i4 |
| t1 | 2 | 3 | 0 | 0 |
| t2 | 0 | 3 | 1 | 0 |
| t3 | 1 | 0 | 0 | 0 |
| t4 | 0 | 2 | 1 | 0 |
| t5 | 0 | 0 | 2 | 0 |

**表4.10属性-项目矩阵** **表4.11 标签-项目标注矩阵**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| hij | i1 | i2 | i3 | i4 |  |  | bij | i1 | i2 | i3 | i4 |
| a1 | 1 | 0 | 1 | 0 |  |  | t1 | 2 | 3 | 0 | 0 |
| a2 | 1 | 1 | 1 | 0 |  |  | t2 | 0 | 3 | 1 | 0 |
| a3 | 1 | 0 | 1 | 1 |  |  | t3 | 1 | 0 | 0 | 0 |
| a4 | 0 | 1 | 0 | 1 |  |  | t4 | 0 | 2 | 1 | 0 |
| a5 | 0 | 1 | 0 | 1 |  |  | t5 | 0 | 0 | 2 | 0 |

其中{i1,i2,…in}为n个项目的集合，{u1,u2…um}表示m个用户，rij表示用户ui对项目j的评分。{a1,a2…,an}表示p个属性的集合，hij表示项目j是否具有an属性。{t1,t2…,tq}表示q个标签的集合,bij表示项目j被标注的次数。为了更好的理解公式与表之间的关系，使用两个三分图的形式来表示。

因此我们可以根据上述理论基础，分别根据用户-项目-标签信息和用户-项目属性信息来挖掘他们之间的潜在关系。从而通过预测评分值来解决新项目的冷启动问题。



**图4-7用户-项目-标签信息 图4-8用户-项目-属性信息**

根据之前的分析发现，结合用户、标签、项目属性提出的个性化推荐算法从理论上可以提高推荐系统的准确率。在此，为了更好的观察用户评分信息、项目属性信息，项目标签信息以及用户评分时效性信息在推荐算法中所产生的影响，我们定义权重, 其中得到预测评分公式如下：

(公式4-5)

由公式4-15可知，该个性化推荐充分利用了用户、标签以及项目属性的相关信息。当, 会综合考虑用户、标签与项目属性信息进行预测评分计算，即用户、标签与项目属性的个性化推荐算法(consider user,tag,and item-attribute personalized recommendation algorithm,简称CUTA)。同理，当时为用户项目属性推荐算法（简称CUA），当时，为用户标签信息推荐算法（简称CUT）。

为了解决推荐系统冷启动问题，我们在上述基础上加入了用户时间权重信息，得到CUTATime算法，该算法会在获取基于项目评分(CUTA)、基于标签评分(CUT)、基于项目属性信息(CUA)得到的预测评分值的基础上通过分析用户评价项目的时间信息，获取用户时间权重，并结合之前得到的预测评分实现推荐。

## 4.6 本章小结

本章介绍了系统的整体架构以及各个功能模块间的关系，并给出了相关模块的数据库表结构设计，最后对推荐系统的整体架构以及功能进行简要介绍。本章让我们从宏观上对整个系统有了大致了解，为后续详细设计奠定基础。

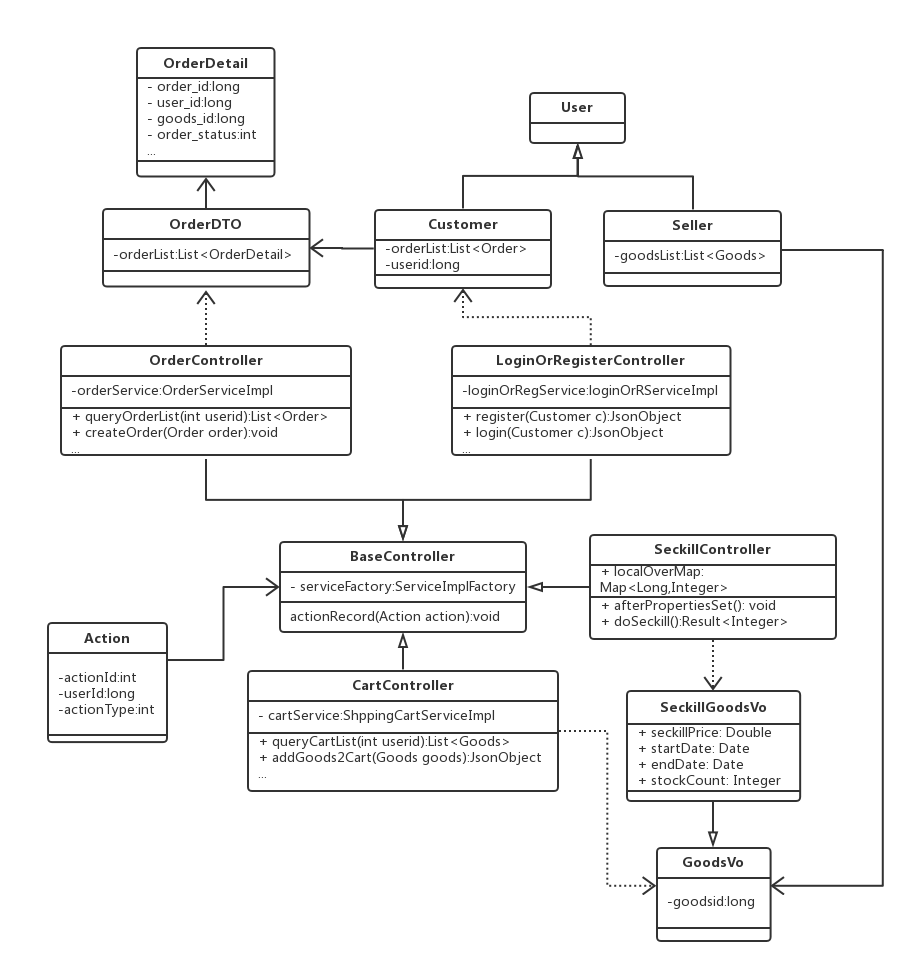
# 第五章商城及推荐系统详细设计

## 5.1 商城基础功能模块设计与实现

本章节主要对整个商城系统做出详细介绍，包括各个模块间的关系，以及核心模块的实现细节等。

### 5.1.1 商城系统总体设计与实现

商城系统的核心模块是用户的登陆注册模块、订单管理模块、以及购物车模块、以及为推荐系统提供数据支持的相关服务。其中各个模块与对象间的关系详见图5-1。



**图5-1商城系统总类图**

为了较为清晰的阐述系统的总体设计结构，我们在图5-1中主要展示了系统的核心模块和相关对象间的基本关系，详细的类图会在后续的章节具体展开。下面我们对主要类的关系做出如下介绍：

(1) BaseController为三大模块提供通用的方法，如生产service具体实现的工厂类。服务接口需要继承此类以获取公共的通过方法和成员变量。其中CartController、OrderController、LoginOrRegisterController 的相关行为会根据用户登录状态通过actionRecord记录到相关日志表中。为推荐模块提供数据基础。

(2) CartController 为购物车的相关操作提供服务接口，包括用户购物车信息的查询，添加商品到购物车，以及修改/删除购物车内商品。

(3) OrderController 为订单模块提供服务接口，其中订单查询接口通过获取请求信息用的user\_id将用户的订单数据封装，并返回到前端展示。当用户从商品详情页或购物车页面下单时，前端页面会将下单的商品数据封装到json中并请求订单创建接口，该接口通过SpringMvc解析json数据并封装成到数据传输对象中，通过具体的service方法完成数据的持久化工作。

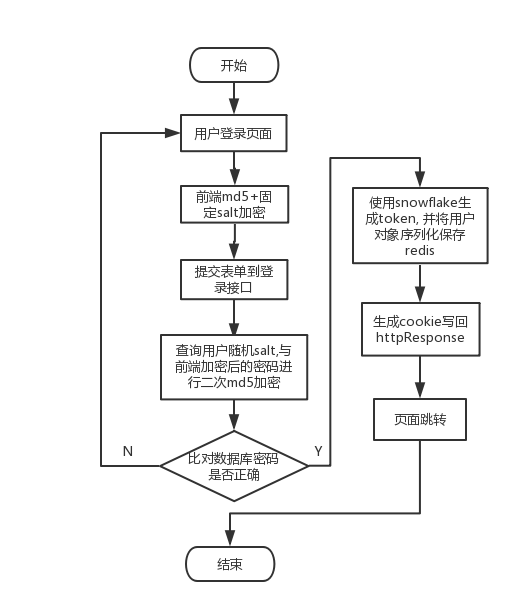
(4) LoginOrRegisterController 主要为用户提供注册和登陆服务。

(5) SeckillController 为秒杀模块提供服务接口，该接口的afterproperties()方法会在服务初始化时将秒杀商品加入缓存，publicSeckillGoods()方法用于发布秒杀商品。其余与处理器映射器绑定的url包括获取秒杀地址，执行商品秒杀操作和查询秒杀结果等。

### 5.1.2 单点登录模块设计与实现

由于整个系统由多个服务器节点组成，每个节点独立的运行系统的某个模块，因此设计了单点登录系统来实现多服务器共享session，考虑到session的查询需要较高的查询效率并且其本身具有时效性的要求，因此我们借助基于内存的数据库Redis来实现类似于session的功能。

单点登录模块流程图如图5-2所示，当用户进行登录操作时，为了防止密码在传输过程中泄露，前端页面会将用户密码拼接一个固定的字符串(salt)进行一次md5加密，然后通过post请求发送到服务器端，服务器端收到请求后，将其通过UserArgumentResoler检查字段合法性，并将登陆信息并封装到LoginVo中，然后判断用户账号是否存在，如果存在，则接收到的加密后的密码与用户注册时服务端随机生成的salt进行拼接，再进行一次Md5加密，然后比对加密后的字符串是否与数据库中的一致，如果一致，则使用snowflake生成随机ID作为token，毕竟用户对象通过protostuff序列化后，保存在Redis中，并生成cookie写入HttpResponse对象中返回给前端。



**图5-2单点登录模块流程图**

整个Redis缓存作为认证服务器，可以被多个系统访问，当用户访问需要用户登录认证的服务接口，如购买、收藏、查看订单时，接口会访问认证服务器查看当前用户token是否有效，若没有未登录，或登录token过期，则自动跳转到登录界面，如果用户已登录，则刷新Redis中数据的销毁时间并刷新用户cookie。

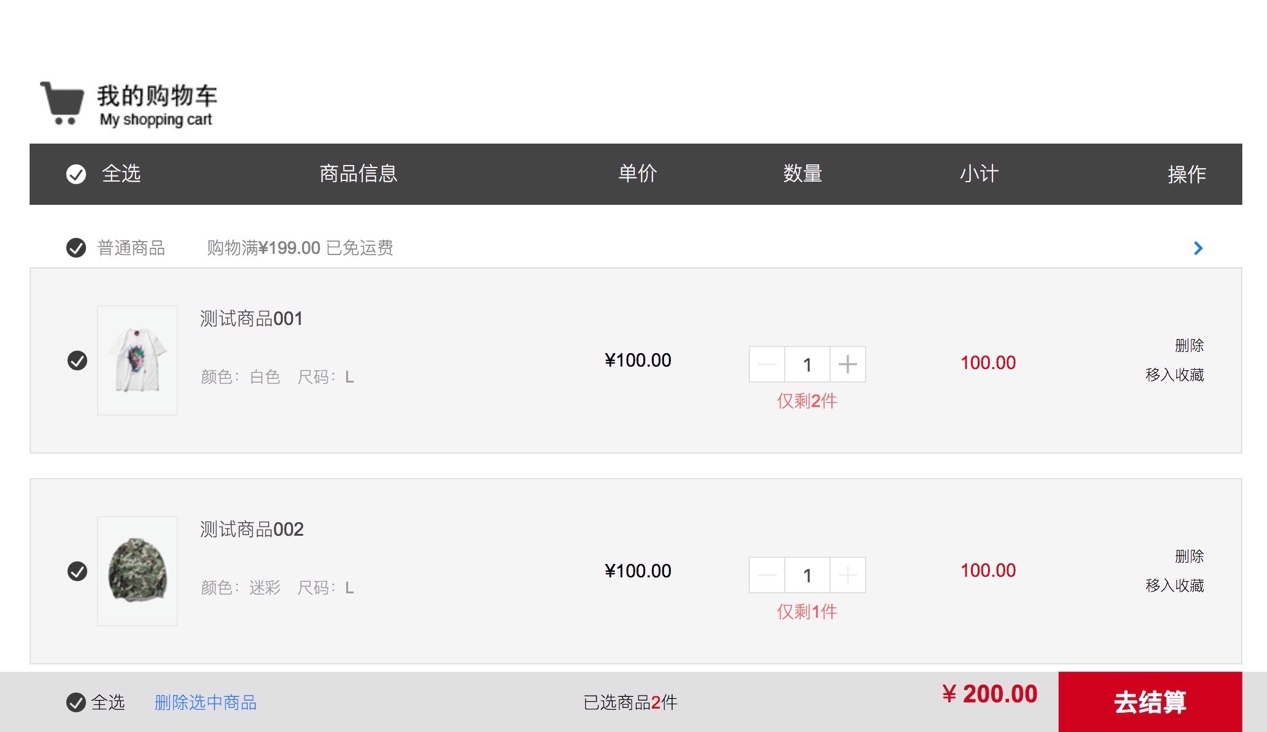
具体登陆注册页面如图5-3所示。

**图5-3用户登录注册页面原型图**

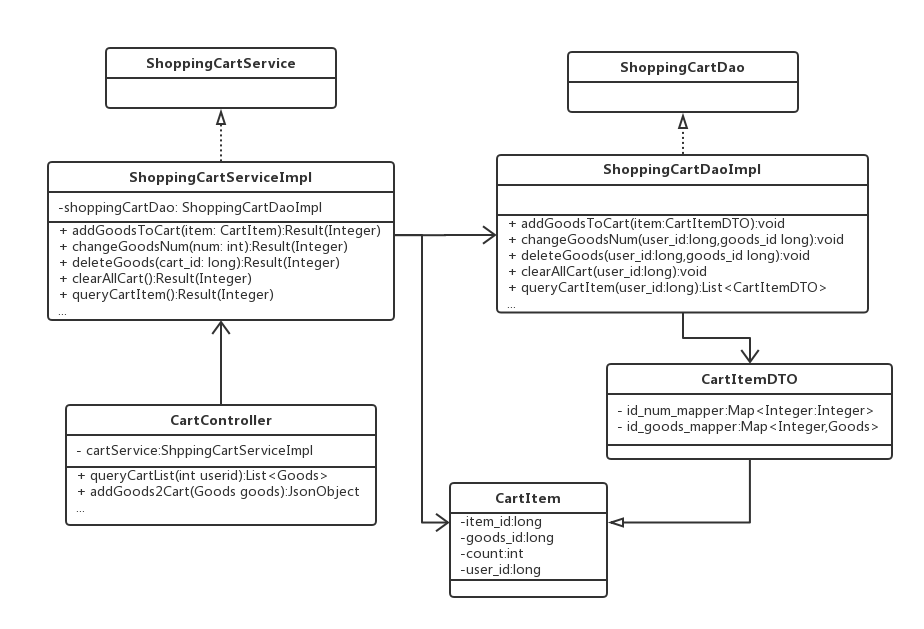
### 5.1.3 购物车模块设计与实现

购物车模块主要用户临时保存用户有意愿购买的商品记录，其主要操作包括将商品加入购物车、查看购物车商品列表，修改商品数量，批量结算购物车中商品，清空购物车等，实际页面如图5-4所示。



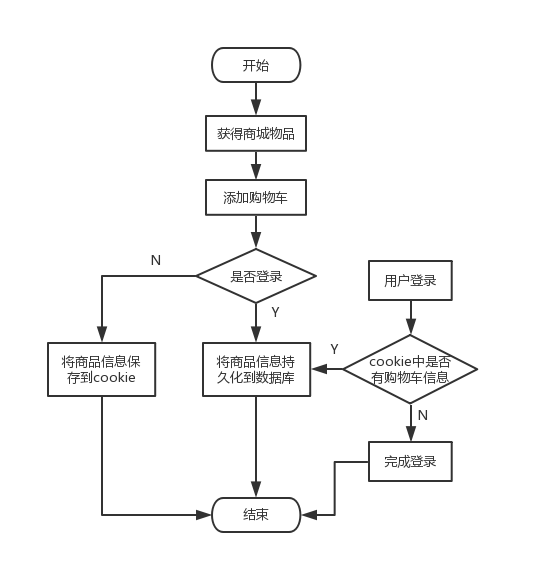
**图5-4 购物车模块原型图**

除此之外，当物品库存小于5时，页面会给出剩余库存的提示。下面结合该模块类图(图5-5)，对相关类的功能和类间关系进行详细介绍。



**图5-5 购物车模块类图**

1. CartController 是购物车模块的入口类，主要用于接收并处理前端的http请求并返回相关数据。
2. ShoppingCartServiceImpl 是购物车模块相关接口的具体实现类，由CartController调用该类来完成接口的具体实现。并通过SpringAOP的面向切面技术，将相关用户行为操作持久化到数据库中。
3. ShoppingCartImpl 是ShoppingCartDao接口的具体实现类，主要完成购物车数据的增删改查等操作。
4. CartItemDTO 是对购物车数据的封装，保存购物车商品和数量的关系映射，并与用户购物车数据的传输和前端展示。



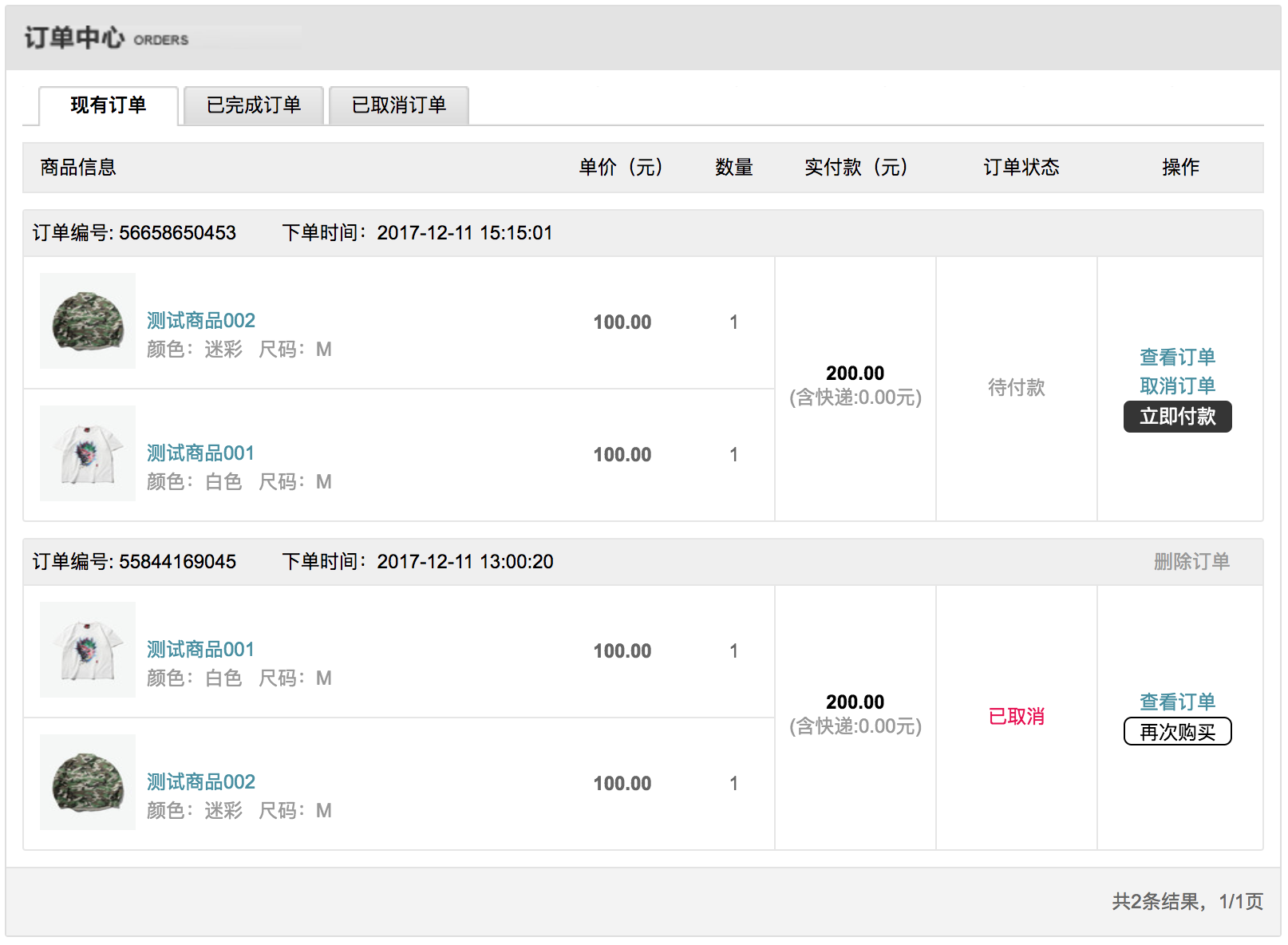
**图5-6 购物车模块流程图**

根据图5-6模块流程图所示，进一步介绍购物车操作的详细流程，当用户进入购物车详情页面时，通过前端页面发送http请求，该请求由Springmvc的前端控制器DispatcherServlet接收，此时前端控制前会根据xml的具体配置去查询HandlerMapping并返回处理执行链，然后去请求处理器适配器去执行具体的Handler CartController，CartController通过shoppingCartService的实现类去根据用户id查询用户购物车详情，并返回购物车的详情的数据传输对象ShoppingCartDTO，并由CartController处理后封装成json返回给前端展示。当用户修改购物车数量，删除购物车商品，则会通过ajax异步的发送请求给CartContoller来调用具体的实现，并异步返回状态码通知前端执行的响应结果。当接收到添加购物车操作时，首先需要判断用户是否登录，当用户没有登录，我们将用户的购物车内容以json的格式存储在cookie中，反之，我们先检查cookie购物车字段对应的value是否为空，如果不为空，我们会将购物车中的商品信息持久化到用户数据库中。

此外，添加/删除购物车的行为对于已登录用户，会影响该用户的商品喜好评分表goods\_attention\_degree中商品id对应的值，而对于未登录的用户，该行为只会对基于item的推荐结果产生影响。

### 5.1.4 订单管理模块的设计与实现

用户订单管理模块主要为用户提供订单管理功能，包括商品的购买(订单生成)、订单取消、订单确认、未支付订单自动关闭、以及查看订单等其他相关服务。具体页面原型图如图5-7所示。



**图5-7 订单管理页面**

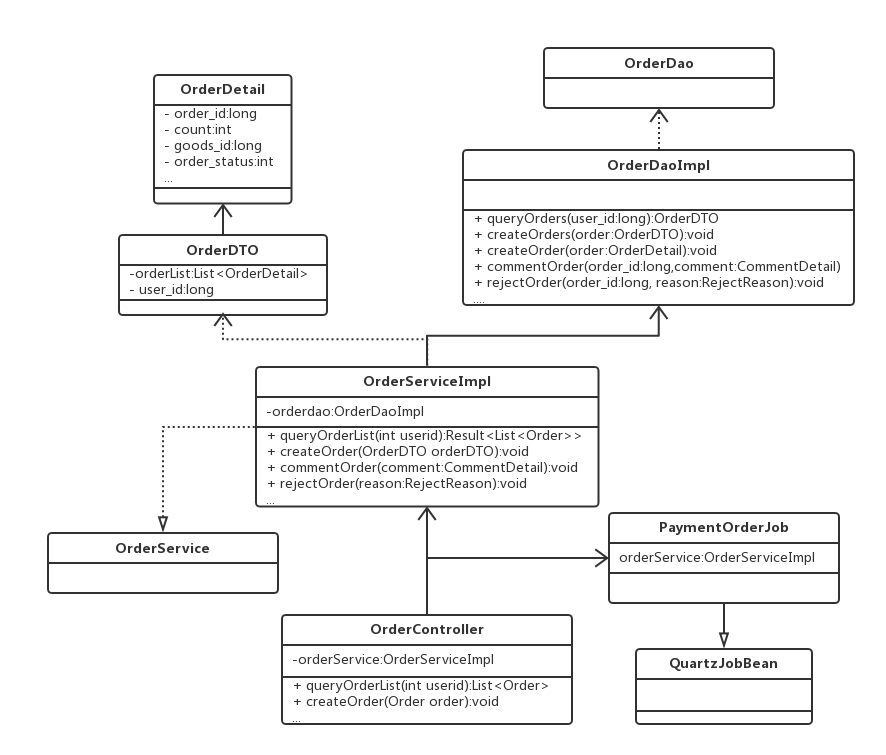
根据图5-8对该模块中所涉及到的重要的类以及类之间关系做出详细介绍。

（1）OrderController是订单功能的入口类，用于接收并处理订单操作的相关http请求，并返回请求结果供前端页面展示。

（2）PaymentOrderJob主要用户启动未完成订单的计时任务，当订单被创建且该任务会被创建，当2小时内未支付，该任务会调用OrderService的closeOrder方法来关闭该订单。

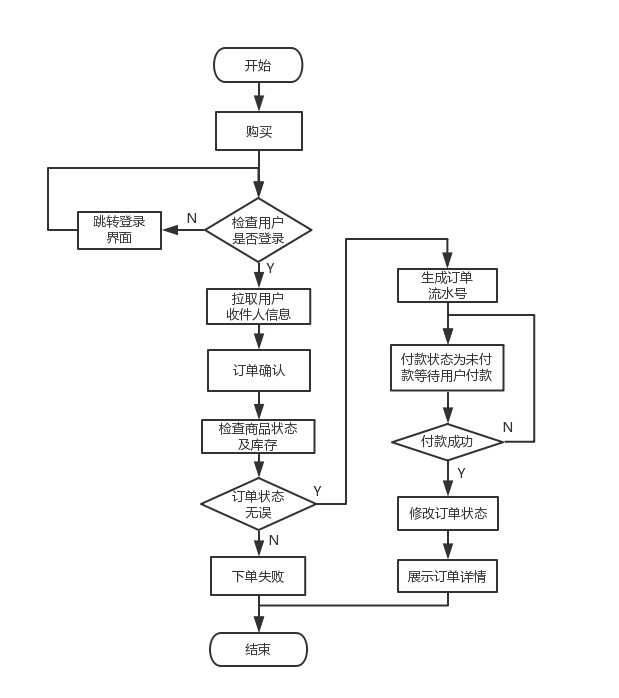
（3）OrderServiceImpl 是OrderService的实现类，主要包括业务方法的具体实现，包括订单创建，订单确认，订单关闭，订单查询等。

（4）OrderDaoImpl是订单模块数据层的相关功能的实现，通过mybatis框架结合xml文件动态生成sql，实现订单模块数据的持久化工作。



**图5-8订单管理模块类图**

结合类图下面主要针对订单生成功能进行详细介绍，具体流程如图5-9所示。当用户点击购买按钮或者通过购物车合并付款来下订单时，会判断用户此时的登陆状态，若用户未登录，则跳转到用户登录界面，登陆成功后跳转的下单界面，选择收货地址等相关信息，确认订单。此时，OrderController会调用OrderService的createOrder方法，该方法由实现类OrderServiceImpl实现具体的逻辑，OrderServiceImpl会遍历OrderDTO中的List<OrderDetail> orders，OrderDetail类的成员变量包括标识该订单具体商品的goods\_id和该订单所对象的商品数量count，Service层会将OrderDetail封装到数据传输对象中，传递给OrderDao，并将该记录持久化到OrderDetail表中，完成订单创建操作。订单管理的其他请求操作如订单查询、搜索、删除、修改等的基本数据流向与之相类似。



**图5-9订单模块流程图**

此外订单ID采用snowflake算法生成，该Id根据创建时间按照一定规则自增，使其在整个系统中不会发生ID冲突。订单超时取消由PaymentOrderJob类实现，该类维护了一个线程安全的无界队列LinkedBlockingQueue，每秒生成一个job\_id加入的该队列中，在这一秒内生成的订单会与 job\_id为key，order\_id为value，追加到Redis的set集合中，详细存储结构如图5-10所示。



**图5-10 未支付订单数据存储结构**

当新的订单被创建时，该订单的order\_id追加到Redis当前时刻的job\_id所对应的set集合中，并对该事件注册支付监听，若成功支付，该监听会调用RedisService的方法将该job\_id中的order\_id移除。每秒会从该地队列队首移除两小时以前的job\_id，并查询Redis中返回该job\_id对应的set集合，若集合不为空则调用OrderDao中的closeOrderBitch(List<String> orders)批处理关闭所有未支付全部订单。

### 5.1.5 商品秒杀模块设计与实现

商品秒杀模块主要为用户抢购优惠促销商品提供稳定可靠的高并发服务，商品秒杀详情页面如图5-11所示。



**图5-11 商品秒杀详情页面**

下面对该模块具体实现所涉及的相关类以及类间关系做出如下详细介绍，秒杀模块类图详见图5-12。

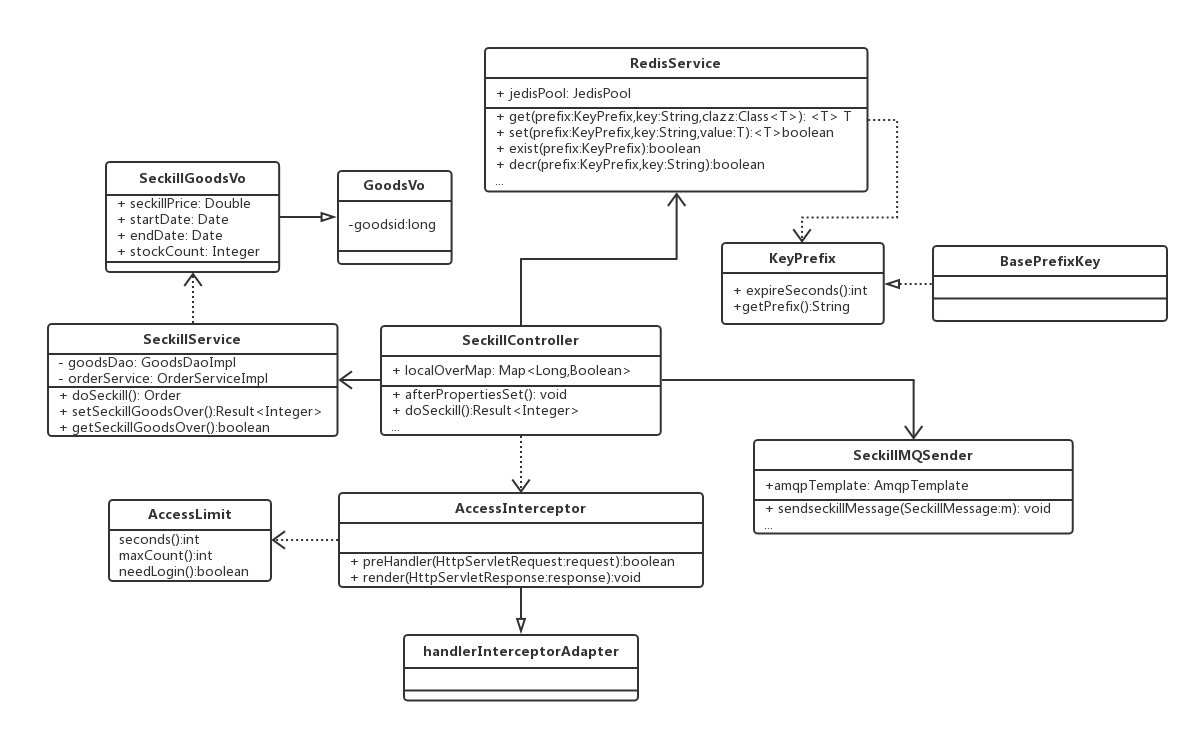
（1）SeckillController 该类是秒杀服务的入口类，用于接收前端的http请求，并调用具体实现类完成秒杀操作，查询秒杀结果等。

（2）SeckillService 该类包含了秒杀服务的具体实现，其中包括秒杀的执行，秒杀结果的查询。

（3）RedisService 主要实现了对Redis缓存的操作具体实现，包括将对象序列化到缓存，从缓存中获取对象，判断某个key在缓存中是否存在，以及对商品进行预减库存等操作。

（4）AccessInterceptor 该类继承自HandlerInterceptorAdapter，并在WebConfig将该拦截器注册到Spring容器中，用于实现对秒杀接口的限流操作，将需要限流的方法加AccessLimit注解，于是该方法的调用会被该类拦截并通过缓存记录用户id在最近时间点的访问次数，当用户访问超过预设值，请求会被拒绝。

（5）SeckillMQSender 该类主要用于向RabbitMQ消息队列发送预秒杀成功的订单对象，该对象会由消费者SeckillMQReceiver按照入队顺序逐一处理。

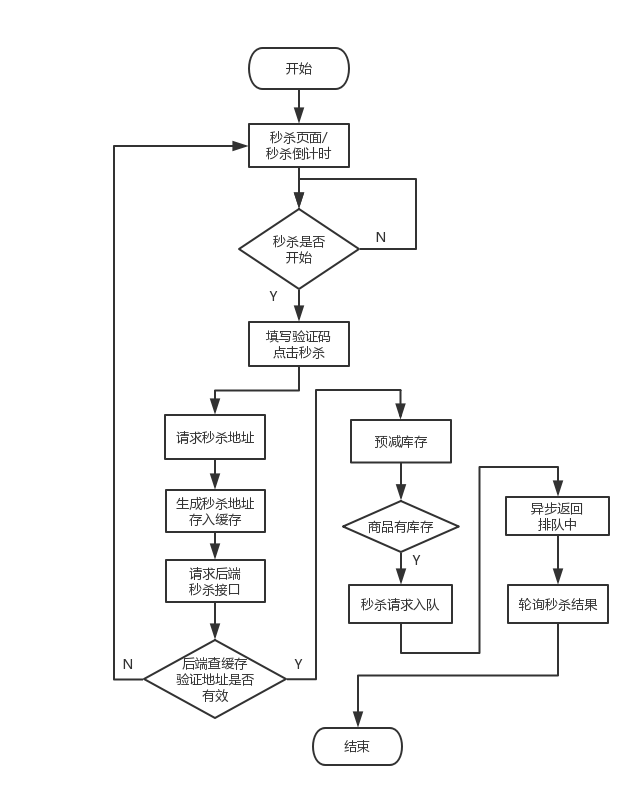


**图5-12 秒杀模块类图**

结合图5-8秒杀模块类图下面对该模块详细设计进行介绍，当用户进入商品秒杀详情页面，根据商品id查询SeckillController提供的接口返回该商品的详细信息，包括商品库存，秒杀的起止时间等，当秒杀活动开始，用户输入验证码并点击秒杀按钮，前端页面发送http请求到SeckillController中的getSeckillPath方法动态获取秒杀地址链接，该方法会根据用户id和salt值通过MD5加密生成动态地址，并用用户唯一标识作为key保存到Redis中，并将地址返回到前端页面。前端页面收到后请求真正的秒杀接口即SeckillController的doSeckill方法，并通过@ PathVariable注解获取到请求url中的path值并查询Redis缓存判断地址合法性，如果该地址合法，则进入秒杀流程，具体流程图详见图5-13 。

当系统初始化或者新的秒杀商品发布时，会通过SeckillController中的afterPropertiesSet() 方法将秒杀商品id作为key库存作为value存入Redis，并将该商品状态保存的ThreadLocal中。为了避免所有秒杀请求直接操作数据库而带来的性能瓶颈，采用Redis预减库存的方式，秒杀执行会通过RedisService的decr方法对相关goodsid对应的库存进行减一，如果减一后库存仍然大于等于0，则将该请求包含的相关信息（如goodsId，userId）封装到SeckillMessage中并通过SeckillMQSender将该对象放入RabbitMQ中，并返回状态码0，前端显示排队中。否则，减库存失败，前端显示秒杀结束。

对于通过Redis预减库存成功请求，SeckillMQReveiver类会依次消费队列中的请求信息，为用户创建秒杀订单并写入数据库中，并写入Redis秒杀结果，前端页面会轮询Redis接口，秒杀成功后提示用户进入下单流程。



**图5-13 秒杀模块流程图**

对于秒杀系统后端性能优化设计总结：

1. 前后端分离，静态页面缓存，减少页面刷新带来的数据传输。
2. 通过Redis预减库存的方式减少对数据库的访问次数。
3. 通过ThreadLocal内存标记，减轻对Redis的访问压力。
4. 通过RabbitMQ记录待处理的秒杀请求，消费消息并落地。

对于秒杀模块的安全性设计总结：

1. 通过二维码方式防止秒杀请求过于集中。
2. 使用隐藏秒杀地址的方式，提高秒杀接口安全性。
3. 通过Spring拦截器和Redis实现接口限流防刷。
4. 通过user\_id和goods\_id作为唯一索引，进一步防止重复秒杀。

## 5.2 推荐系统相关模块设计与实现

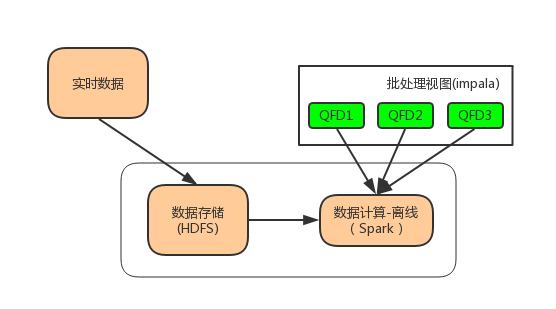
根据第三章的需求分析可知，商品推荐有着不同的应用场景，因此推荐系统需要通过对用户行为数据的分析从不同的维度使用不同的推荐算法生成满足需求条件的推荐结果。本章节会对不同的推荐模块的实现做出具体阐述。

### 5.2.1 推荐系统计算框架设计与实现

(1) 离线计算部分

离线数据的存储过程是当新数据产生时，我们通过flume/sqoop（用于关系型数据库）将其收集到hdfs中，每半小时存放到hdfs的一个目录中，然后进行批处理（可以使用mapReduce或者spark）处理完后倒入impala, impala是一个sql引擎，能够通过sql语句查询存储在Hadoop的HDFS和Hbase中的PB级大数据。

批处理层特点，因为写到了hdfs系统中，因此数据不可修改，只可以追加，可通过spark或map-reduce进行任意计算，支持水平扩展（通过扩展机器），缺点是高延迟，根据计算量或者数据量的不同，运行时间可能是几分钟到几个小时。具体设计架构见图5-14。



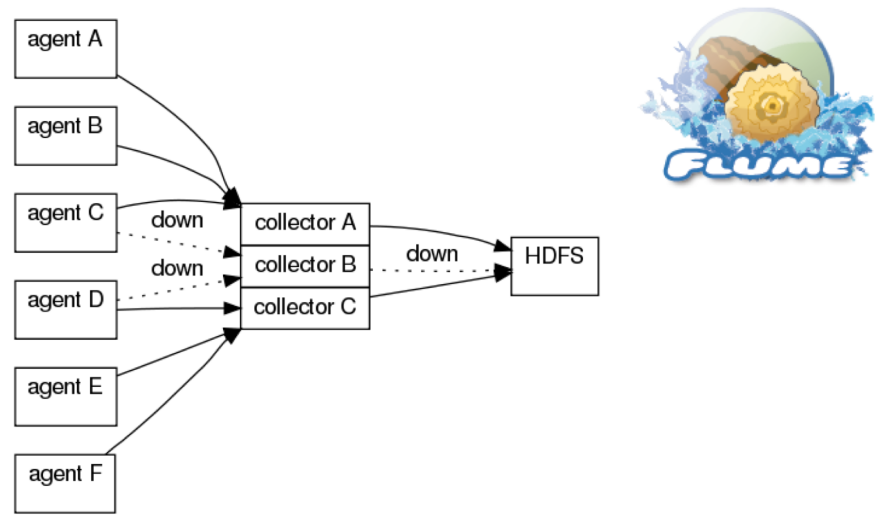
**图5-14离线计算系统架构**

该模块主要承担两个主要职责，一、存储Master DataSet, 即一个持续增长的数据集。数据来源于系统的实时数据采集模块。二、对Master DataSet 进行预计算。离线计算的模式与批处理为主，我们使用Spark的RDD算子方式进行：

|  |
| --- |
| 批处理计算伪代码： |
| function runOffLineCompute():  while(true)  recomputeBatchViews() |

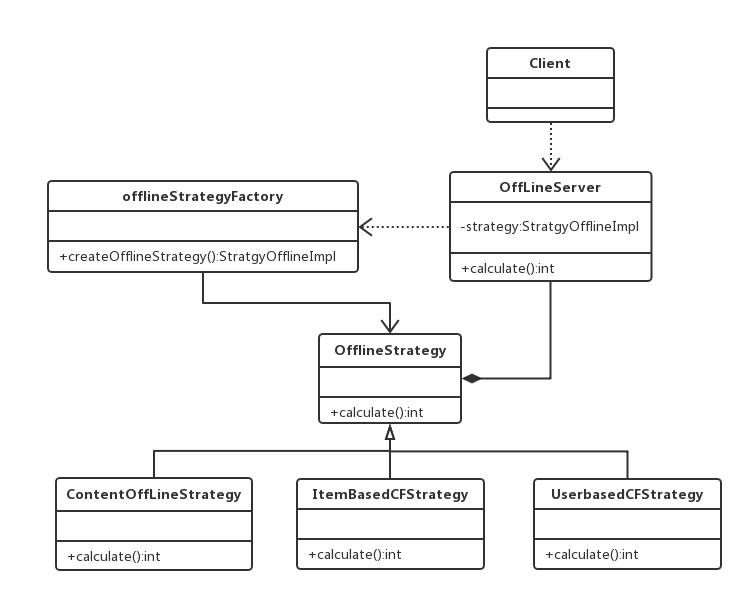
利用离线计算模块进行数据预处理的主要作用是将实时计算的数据量尽可能的减小，从而有效地利用资源，改善实时计算的性能。但这里有一个前提，就是我们需要预先知道实时计算需要的数据，如此才能在离线计算中安排执行计划，定期对数据进行批量处理。此外，还要求这些预运算的统计数据是支持合并。

离线计算的数据采集我们使用的是flume框架，该框架主要用于采集非关系型数据-如日志信息等，flume为收集、移动大量的日志数据提供了一种可靠的可用的分布式框架，它具有基于流式数据简单灵活的架构，并具有故障转移和恢复机制，为系统提供一定的可靠性和容错能力。Flume需要在产生日志的agent结点上部署,用于监听节点产生的数据，agent节点可以将收集的数据传输给collector,然后collector对数据流进行汇总（该设计的主要目的是为了减少hdfs中小文件的个数，减轻hdfs的并发压力），并将其写入hdfs中。



**图5-15 flume数据采集流程图**

推荐系统离线计算框架的设计实现采用了策略设计模式，该模式下将不同的推荐算法独立封装实现，并暴露对外接口通过入参来自由的切换使用的算法，该模式可以在服务运行或调试期间灵活的更换使用的具体算法，各个算法时间相对独立封闭，而在实际的项目中，我们通过工厂设计模式来获取策略类的实现，离线计算具体类图如图5-16。



**图5-16 离线计算模块基本类图**

其中OfflineSever对外提供服务接口，OfflineStrategy将具体的策略进行抽象，该类不关注算法层面的具体实现方式，每个算法的实现相互独立，如基于内容的推荐的离线计算部分的具体实现由ContentOffLineStrategy类来完成，而基于Item的协同过滤推荐算法则是由ItemBasedCFStrategy完成。

（2）实时计算部分

要完成用户实时推荐的计算任务，第一步，要对数据进行采集和清洗操作。对于实时推荐系统的计算需要两部分数据集，一部分是用户之前的离线的数据，第二部分是用户最新的实时数据。其中实时数据主要包括用户ID,用户最新的行为日志（如：用户对商品和评价操作，即用户-商品信息，包括用户ID，商品ID，商品分类，评分等级，时间戳字段等）。

实时流计算是指对源源不断的数据流进行系统的连续计算，并实时的返回结果。为了确保计算的精确性和有效性，就要求系统提供实时的用户行为信息，然后通过相关计算，将结果写入Hbase中，为后续推荐操作提供推送和查询。

实时流计算作为加速层主要用于实时处理，缺点是不能进行复杂计算。

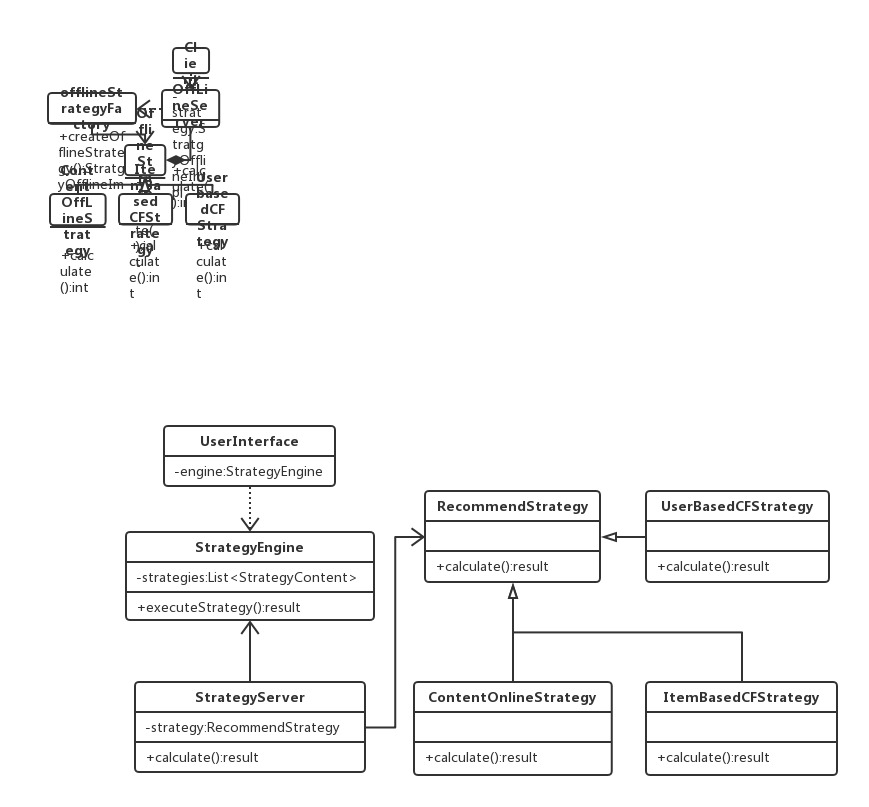
实时计算层的特点主要是对数据进行流式处理，持续性计算，存储和分析某个窗口内的数据，该方式通常是基于实时性和准确性的一种折中，因此准确性有时并不高（临时解），后期最终的结果，会通过批处理获取，批处理会覆盖掉流式处理的结果。具体架构图详见图5-17：



**图5-17实时计算系统架构图**

其中对于新产生的数据会收集用于离线计算部分，因为实时计算不允许进行耗时的复杂计算。因此其中一小部分数据会选择性的提交给kafka消息总线，用于实时计算。对于数据收集部分会通过节点上部署的flume上进行，而且flume与kafka也有很好的兼容性。

在线计算具体的设计逻辑与离线计算有本质上的不同，离线计算的计算方式是在固定的时间点或间隔点执行。而在线计算则是由通过暴露的对外接口调用来执行。具体实现类图如图5-18所示。



**图5-18实时计算模块基本类图**

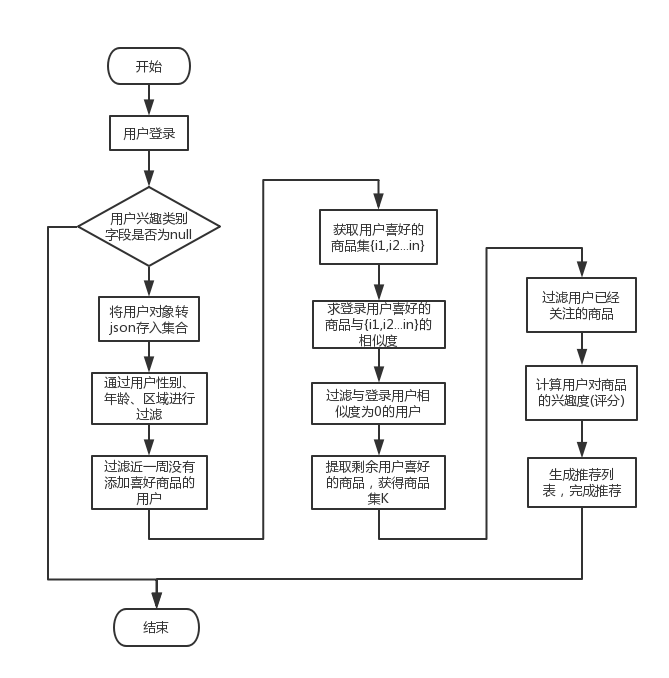
实时计算模块的依然通过策略模式来实现，StrategyServer是算法具体实现使用的类，StrategyEngine则是用来对外暴露的服务接口，通过调用executeStrategy() 来调用具体的算法实现类来获取计算结果，再将结果通过封装返回给用户。下面小节中将具体介绍每种算法的原理、流程和实现。

### 5.2.2 基于user的协同过滤推荐模块设计与实现

由于注册用户的数据量比较庞大，用户-商品的特征矩阵非常稀疏，因此需要使用离线计算对用户数据做预处理来减少在线实时推荐的计算压力。离线计算时，通过查询用户登录次数，总订单数，计算用户历史活跃度，当活跃度满足条件，我们根据历史行为为用户进行分类，并将用户类别标签记录到用户标签表中。该标签会随着用户行为变化而变化。通过聚类，对相同类别用户展开计算，可以大大减少计算量。

当用户登录后，先查询用户标签是否为null，如果不为null，说明该用户活跃度满足条件，我们就可以使用基于用户的协同过滤推荐算法。

首先我们先把与该用户具有相同用户标签的用户中的对象转json存入List<String> userList中，我们使用jdk8的并发流处理List中的数据，首先通过userList.stream()获取去流对象，根据userList.stream().filter( )对用户性别，年龄，所在地区，做进一步过滤，最后，我们对剩余用户做相似度计算，并找到与目标用户相似度较大的前n个用户，然后提取该用户最近喜好的商品，剔除出用户已经关注的商品，然后进行推荐。具体流程图见图5-19。



**图5-19基于user的协同过滤算法流程图**

其中用户间相似度计算我们可以使用余弦或者Jaccard公式，其中N(u)为用户u喜欢的物品集，N(v)是用户v喜欢的物品集，那么用户u与用户v的相似度计算详见第二章公式2.3和公式2.3。

根据相似度排名后取前N个与目标用户u最相似的用户集,用公式S(u,N)表示，并获取这N个用户喜好的商品集合，并去除登录用户已经喜欢的商品，获取剩余商品集S(i)，对于用户u对商品i的兴趣度(计算评分)可通过公式(5.1)计算。

（公式5.1）

然后我们根据目标用户对相关商品的兴趣度由高到低进行排序，然后选取前k个商品推荐给用户，完成推荐。

具体实现所涉及的主要函数方法介绍如下：

（1）queryStandardScoreList(): 该方法用户查询用户对商品的评价集，以商品id和评分值来构建评分矩阵。

（2）similarityDegreeCompute(): 用于用户维度和商品维度的相似度计算，通过入参similarityType来选择不同的相似度计算方式，相似度计算包括余弦相似度，皮尔逊相似度等。

（3）generateNeighbor(): 根据相似度计算结果进行排序，通过预设的相似度阈值P以及最大邻居数K，生成符合要求的用户集。

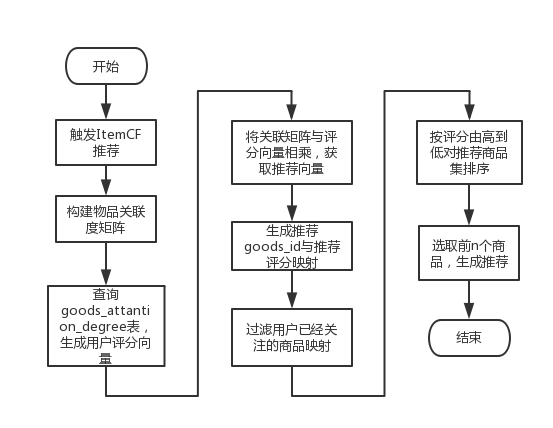
（4）buildCandidateList(): 过滤最近邻居集中重复或目标用户已购买的商品。

（5）score(): 通过加权平均对候选集做预测评分，评分结果将直接影响最终的推荐结果。

### 5.2.3 基于item的协同过滤推荐模块设计与实现

相比于从用户角度出发的User-Based 协同过滤算法，Item-Based 协同过滤则是从商品角度出发来生成推荐，寻找与用户喜欢的、最近关注的、或者曾经购买并好评的商品相似的商品推荐给用户，其主要思想是分析用户-项目矩阵以识别不同项目之间的关系，然后使用这些关系来计算给定的某对用户的预测分数。找到用户有兴趣购买与用户之前购买项目相似的项目，并且倾向于避免/不予以推荐类似于用户之前不喜欢的项目。由于该算法是基于商品的推荐，因此无论用户是否登录均适用，只要当用户对商品进行收藏、加购物车等表现出对该商品关注的行为时，我们就可以基于ItemCF生成推荐。

首先我们会通过离线的方式计算用户历史行为偏好的商品间的相似性关系，并添加都用户商品推荐表中，并标注推荐生成时间。当用户将某商品加入购物车后，我们需要为用户在购物车加入成功的提示底部生成商品推荐列表，具体推荐流程图如图5-20所示。



**图5-20基于item的协同过滤算法流程图**

基于ItemCF推荐算法，需要维护一张物品-用户关注度表（goods\_attention\_degree）,该表主要用户记录goods\_id与user\_id相关联的物品关注度，用户对商品的行为通过对应评价规则表将直接影响该关注度的值，具体评价规则表如表5-1所示。

**表5.1用户商品喜好度评价规则表**

|  |  |  |  |
| --- | --- | --- | --- |
| **操作描述** | **评价规则** | **操作描述** | **评价规则** |
| 收藏 | +2 | 购买 | +2 |
| 浏览 | +0.5 | 评价(好评/带文字) | +2/+4 |
| 加购物车 | +1 | 评价(差评) | -4 |
| 购物车删除商品 | -1.5 | 商品详情页停留时间<5s | -1 |
| 收藏删除商品 | -1.5 | 商品详情页停留时间>15s | +1 |

对于物品相似度矩阵，我们可以将矩阵的值按照最大值进行归一化处理，以此来提高推荐结果的准确度，归一化公式如下:

其中是归一化后得到的目标相似矩阵。

  随后查询用户关注度表，生成推荐向量，然后将归一化后的物品相似矩阵与推荐向量相乘，获取推荐向量，选取推荐向量中前n个值对应商品进行推荐。

### 5.2.4 基于内容的推荐模块设计与实现

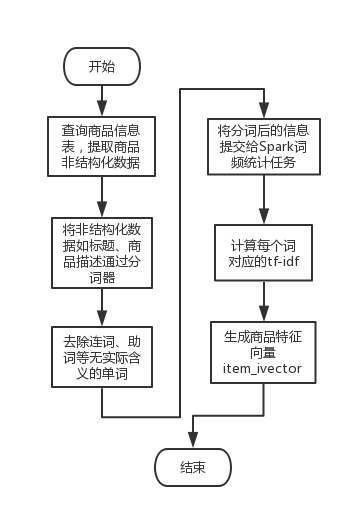
基于内容的推荐，主要是根据用户历史喜好的商品，如已购买并好评的商品，寻找与这些商品相类似的商品。基于内容的推荐算法主要包括三个步骤，第一、为商品提取特征信息(item-vector)。第二、利用用户历史数据(对商品的评价信息)来计算该用户喜好的特征信息(profile-vector)。第三、根据候选商品的item-vector和用户的profile-vector选取出相关性较大的前N个商品推荐给用户。

首先我们进行商品特征信息的提取，我们将商品信息中的非结构化数据(如: 商品标题、商品描述信息等)通过分词统计的方式提取中一个词在该商品中出现的频率，大部分情况下，一个商品的非结构化信息包含多个词，因此该商品i可以通过向量di=(w1i,w2i,w3i...) 表示，其中w1i表示第一个词在商品i中出现的权重，该值越大，表示这个词对于该商品的意义越重要。

随后我们通过词频-逆文档频率（term frequency–inverse document frequency， tf-idf） 计算向量di的各个权重，其中第i篇商品的第k个词对应的tf-idf为：

其中TF(tk，di)是第k个词在商品描述中出现的次数，nk所有商品描述中出现第k个词的商品的数量，最后，第k个词在商品k中的权重可以通过公式5.4将tf-idf做归一化获得。

商品特征提取流程图如图5-21所示。



**图5-21商品特征向量提取流程图**

接着我们根据用户对历史商品的评价数据，构建用户喜好模型，我通过查询商品的结构化数据，如商品的分类、属性等，构造训练集，并将其提交给spark计算任务，输入数据如表5.2所示：

**表5.2 商品结构化数据类型表**

|  |  |  |  |
| --- | --- | --- | --- |
| **商品一级分类** | **商品二级分类** | **商品三级分类** | **是否好评(浏览相当于好评)** |
| 数码产品id | 手机id /配件id | 品牌id | 好评1/差评0 |

我们通过Spark Mllib提供的机器学习库来进行模型的构建，首先通过API LabeledPoint将原始数据转化成向量标签，然后设置决策树深度depth，以及最大分支树maxBranch,设置分类变量val categoricalFeaturesInfo = Map[Int, Int]()，最后就可以通过val model = DecisionTree.trainClassifier来进行模型训练。

当需要从一批候选商品中，找到用户可能喜欢的商品时，就可以通过之前训练的决策树模型调用model.predict方法做初步的过滤和筛选，然后提取剩余的商品特征向量item-ivector，与用户的偏好向量profiler-ivector做余弦相似度计算，选取topN的商品，推荐给用户。

## 5.3 本章小结

本章主要介绍商城系统和推荐系统的主要模块的设计与实现，包括商城系统的登陆注册模块、购物车模块、订单管理模块，商品秒杀模块，以及推荐系统的日志收集、计算框架、和具体的推荐模块的功能实现。每个模块的设计与实现从模块的流程图，以及系统的类图分别进行介绍。

# 第6章 实验与测试

## 6.1系统开发实现环境

  网上商城系统和推荐系统和具体实现环境、开发工具和部署环境见表6.1。

**表6.1 开发测试环境信息表**

|  |  |
| --- | --- |
| 说明 | 开发工具 |
| 开发操作系统 | MacOS 10.12 |
| 开发工具 | IntelliJ IDEA、Vim、Sublime Text2 |
| 设计工具 | Microsoft Visio2016、MySQLWorkbench |
| 代码部署环境 | Ubuntu 16.04 / Centos 6.5 |
| Web服务器 | Nginx 、Tomcat7 |
| 版本控制 | SourceTree |

## 6.2商场系统测试

### 6.2.1测试环境

我们分别从软硬件角度介绍测试环境，其中商城系统测试的硬件环境配置表如表6.2所示，软件环境配置表如表6.3所示。

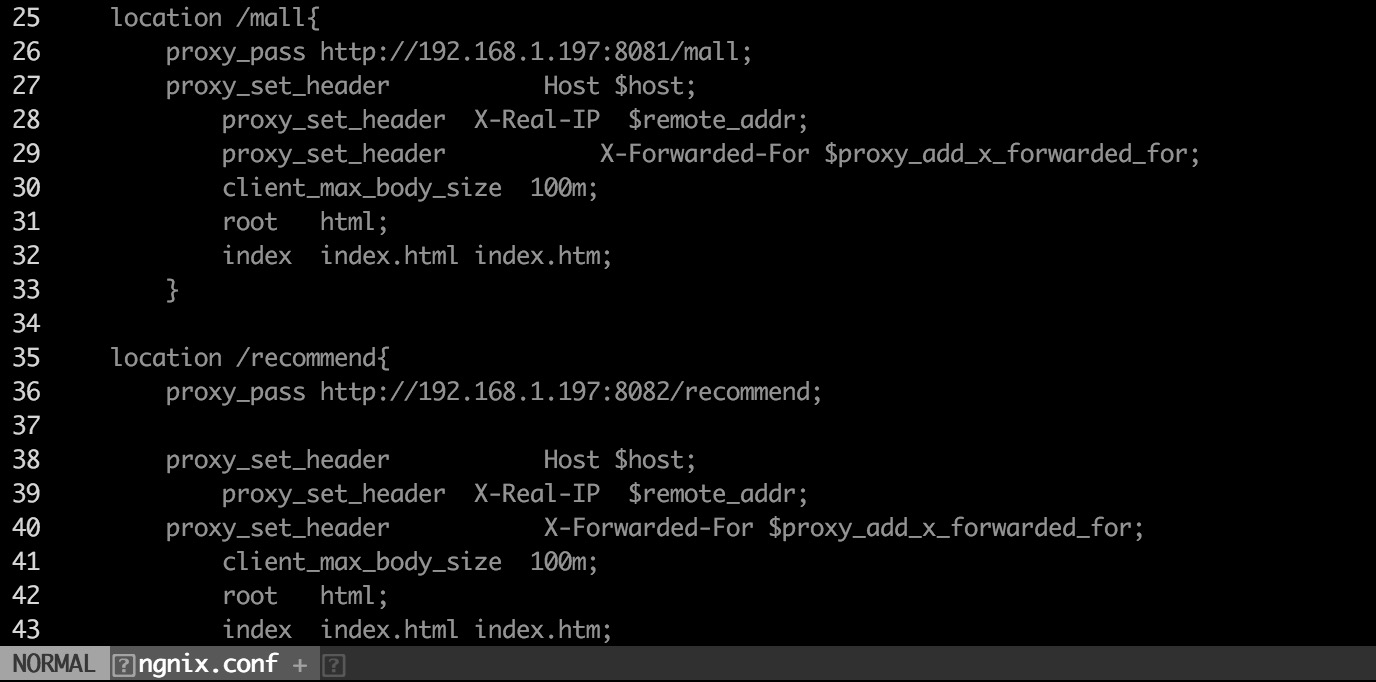
**表6.2 商城系统硬件环境配置表**

|  |  |  |  |
| --- | --- | --- | --- |
| **类别** | **基本配置** | **数量** | **单位** |
| 客户端PC | 3.2GHz Intel Core i7 ,16G 内存 | 1 | 台 |
| 应用服务器 | 8核CPU ，16G内存  1T硬盘 | 1 | 台 |

**表6.3商城系统软件环境信息表**

|  |  |
| --- | --- |
| **类型** | **名称/版本** |
| 操作系统 | ubuntu16.04 |
| web服务器 | tomcat7 |
| 负载均衡服务器 | nginx 1.12.2 |
| 数据库 | mysql 5.7 |
| no-sql数据库 | redis 3.2.5 |
| 消息队列 | RabbitMq |

我们在ubuntu上分别部署了两个tomcat web服务器，并将商城系统部署在tomcat1上，将推荐系统的服务模块部署在tomcat2上，然后通过ngnix做反向代理访问各个服务接口，修改/usr/local/nginx/conf/nginx.conf ,相关配置如图6-1所示。



**图6-1 Nginx反向代理配置**

然后保存，通过nginx –s reload 重启服务，就可以通过http://192.168.1.197/mall 和http://192.168.1.197/recommend 来分别访问相关服务。

### 6.2.2商城系统功能性测试

启动tomcat服务器，并部署商城服务代码，按系统模块设计测试用例，并通过postMan发送http请求，观察http返回json结果是否符合实际要求。

1. 用户登录注册模块测试

根据测试用例对该模块进行测试，注册接口URL: ${domain}/test/to\_register,

登录接口URL:${domain}/test/to\_login，测试结果与结论详见表6.4。

**表6.4登录注册模块用例表**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试模块** | **用例描述** | **返回值情况** | **结论** |
| 用户登录注册模块 | 用户注册信息不完整、格式错误 | 无法提交表单，提示错误信息 | 正常 |
| 已注册用户提交注册表单 | 返回状态码和提示信息 | 正常 |
| 未注册用户提交注册信息 | 返回注册成功 | 正常 |
| 单一IP频繁注册 | 无法注册、返回提示信息 | 正常 |
| 用户登录格式信息不完整 | 无法提交表单，返回提示信息 | 正常 |
| 用户登录信息完整，用户不存在 | 返回状态码和提示信息 | 正常 |
| 用户登录信息完整，用户已注册 | 提示登录成功，页面跳转 | 正常 |

1. 购物车模块测试

购物车模块测试主要从用户是否登录出发，测试商品数据是否能够成功保存，其次考虑商品库存边界情况，商品库存不足时是否能够成功添加购物车。

具体测试用户及测试结果详见表6.5。

**表6.5购物车模块测试用例表**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试模块** | **用例描述** | **返回值情况** | **结论** |
| 购物车模块 | 未登录状态下，查看购物车内容 | 页面正常跳转，购物内容无误 | 正常 |
| 已登录状态下，查看购物车内容 | 页面正常跳转，购物内容无误 | 正常 |
| 添加商品到购物车，当商品库存数量<=添加购物车的商品数量 | 商品添加成功，页面正常跳转 | 正常 |
| 添加商品到购物车，当该商品库存数量>添加购物车的商品数量 | 无法添加商品到购物车，提示库存不足 | 正常 |
| 修改购物车商品数量时，当更改的商品数量>=商品库存时 | 修改成功 | 正常 |
| 修改购物车商品数量时，当修改商品数量>商品库存 | 提示库存不足,无法修改 | 正常 |
| 成功添加购物车的商品已售罄(库存为0) | 该商品变灰色，无法选中或者发起下单付款操作 | 正常 |
| 在未登录状态下删除购物车中商品 | 成功删除 | 正常 |
| 已登录状态删除商品 | 验证数据库，用户购物车对应商品已删除 | 正常 |
| 用户登录后合并未登录时的购物车商品 | 可以将登录前cookie中的商品数据提取并合并到用户历史购物车中。 | 正常 |

1. 订单模块测试

用户在商品详情页或者购物车详情页面购买商品，支付，创建订单，并可以在订单详情页查看订单状态，并对订单进行修改查询等操作。

测试用例主要考虑订单创建时商品的边界情况，保证线程安全性，防止超买超卖状况发生。订单创建后，还需要对订单管理的各个功能模块进行测试，观察返回结果以及数据库修改情况，保证数据的一致性。具体测试用例及测试结果详见表6.6。

**表6.6订单模块测试用例表**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试模块** | **用例描述** | **返回值情况** | **结论** |
| 订单管理模块 | 未登录状态查询订单 | 页面跳转到登录界面。 | 正常 |
| 已登录状态查询订单 | 订单数据显示正常，多个订单已分页显示。 | 正常 |
| 搜索用户历史订单 | 根据关键字模糊查询，返回结果准确。 | 正常 |
| 查询订单商品详情 | 跳转商品详情界面  1、若商品已下架，展示商品详情快照。  2、若商品状态正常，正常显示商品详情。 | 正常 |
| 用户在不同的订单状态下修改订单状态(删除订单，关闭订单，申请退款，确认收货，评价等操作) | 1、订单未确认状态不允许删除订单操作。  2、处于申请退款状态的订单无法进行确认收货操作。  3、关闭订单，申请退款，评价等操作返回状态均正确无误。 | 正常 |

（4）商品秒杀模块测试

商品秒杀模块测试与之前的模块有些不同，除了对常规的功能测试外，主要是针对秒杀操作进行高并发的测试并采用断点追踪的方式查看秒杀过程中缓存、消息队列数据的正确性，具体测试用例详见表6.7。

**表6.7商品秒杀模块测试用例表**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试模块** | **用例描述** | **返回值情况** | **结论** |
| 商品秒杀模块 | 秒杀未开始 | 秒杀按钮为灰色不可点击 | 正常 |
| 进入秒杀详情页，秒杀商品 | 正常秒杀场景，可以成功秒杀商品 | 正常 |
| 库存不足时秒杀商品 | 返回库存不足，页面更新为秒杀结束 | 正常 |
| 多用户同时并发秒杀 | 无超卖现象发生 | 正常 |
| 无session情况，脚本访问秒杀链接 | 返回异常状态信息Session无效 | 正常 |
| 已秒杀该商品的用户进行二次秒杀 | 预减库存成功，但秒杀结果显示不允许重复秒杀 | 正常 |
| 商品库存已售罄或到达秒杀截止时间 | 显示秒杀活动已结束提示 | 正常 |

## 6.3推荐系统测试

### 6.3.1测试环境

推荐系统的测试环境仍然从软硬件角度介绍，具体配置表详见表6.8和表6.9。

**表6.8 推荐系统硬件环境信息表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **类别** | **基本配置** | **数量** | **单位** |
| PC | 客户端 | 3.2GHz Intel Core i7 ,16G 内存 | 1 | 台 |
| 服务器 | 应用服务器 | 4核CPU ，8G内存  512M硬盘 | 3 | 台 |

**表6.9 推荐系统软件环境配置表**

|  |  |
| --- | --- |
| **类型** | **名称/版本** |
| 操作系统 | centos 6.5 |
| 分布式文件系统 | hadoop-2.7.5 |
| 分布式计算框架 | spark-2.3.0 |
| web服务器 | tomcat7 |

如配置表所示，我们使用3台centos系统的服务器构建测试集群，并在master节点上部署tomcat  web服务器用户部署计算服务框架，用于接收外部请求并返回处理结果。分布式文件系统我们使用hadoop提供的hdfs，分布式计算则使用spark进行。

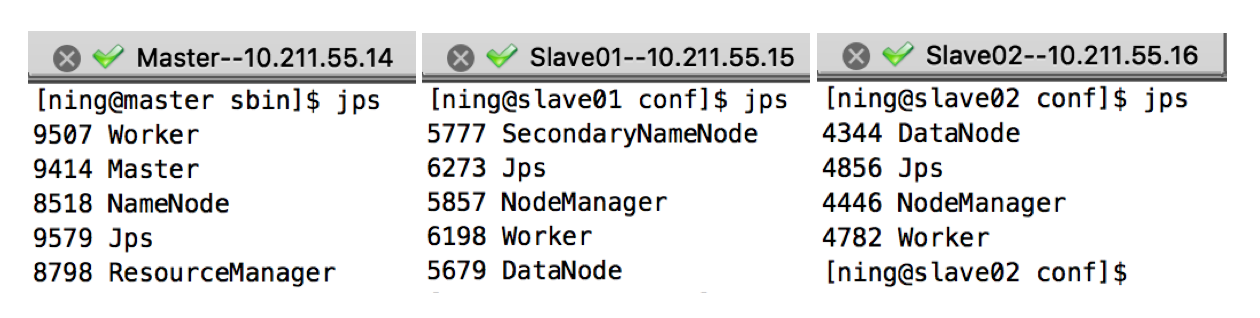
根据测试环境需求，我们分别配置安装hadoop和spark环境，其中需要为各个服务器配置JDK环境，关闭防火墙，修改/etc/hosts文件配置相关ip的映射，并配置各个服务器的rsa免密登陆。

随后我们在master节点将hadoop-2.7.5解压，分别修改hadoop-2.7.5/etc/hadoop下的hadoop.env.sh、hdfs-site.xml、core-site.xml、mapred-site.xml、mapred-env.sh以及yarn-site.xml。并在slaves中添加slave01和slave02，最后通过执行./bin/hdfs namenode –format 格式化namenode。

接着我们将spark解压，并配置spark-env.sh和slaves。

最后将配置好的hadoop和spark 通过scp命令复制到slave01节点和slave02节点。

部署好后启动hadoop和spark服务，通过start-all.sh 启动集群。启动后具体的服务进程如图6-2所示。



**图6-2 hadoop、spark服务进程详情**

Spark 分布式计算各个计算节点资源以及Worker的详情如图6-3所示。



**图6-3** spark 计算监控页面

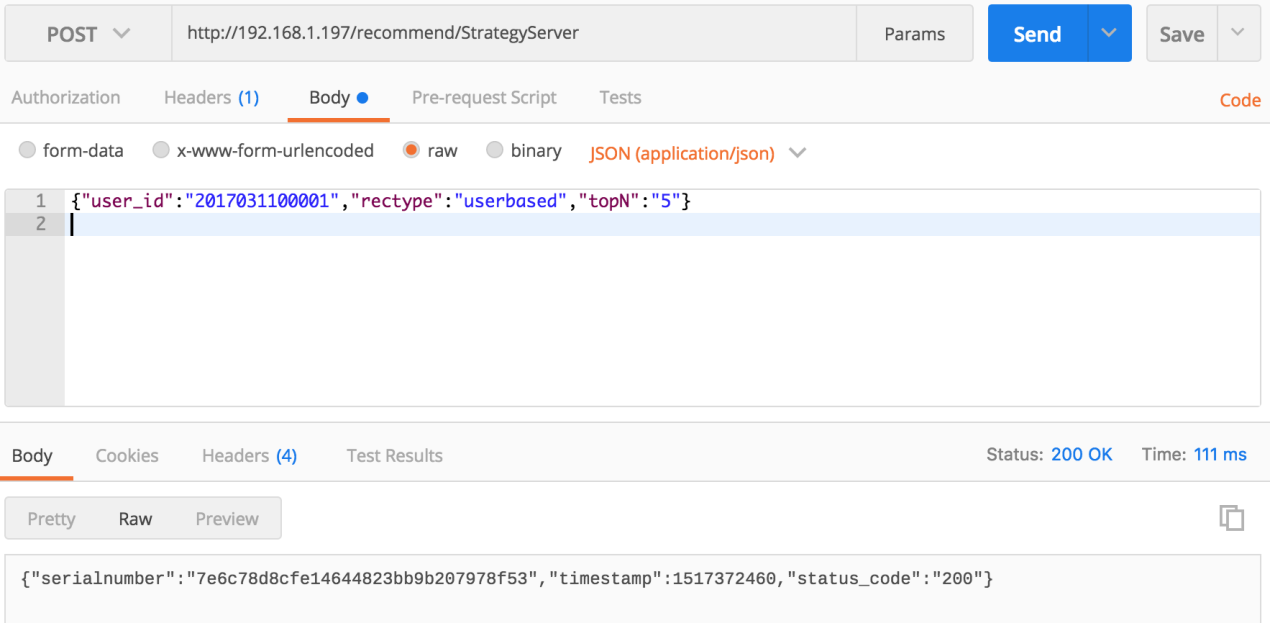
### 6.3.2推荐系统功能性测试

推荐系统功能测试主要采用黑盒的方式进行，测试的主要方面包括测试计算服务框架接收请求并触发spark任务的正确性，基于user的协同过滤推荐算法的返回结果，基于item的协同过滤推荐算法的返回结果和基于内容的推荐算法的返回结果，并进行并发性测试，评估系统抗压能力。其中功能性测试仍然使用黑盒测试的方式，对不同应用场景下的三种推荐算法分别展开。具体测试用例见表6.10。

**表6.10推荐算法功能测试**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试模块** | **输入用例** | **期望结果** | **测试结论** |
| 基于user的协同过滤推荐模块 | 发送HTTP请求到StrategyServer服务接口，入参为：{user\_id=”2017031100001”, rectype=”userbased”，topN=”5”} | 根据该用户历史记录返回推荐结果。 | 符合预期 |
| 基于item的协同过滤推荐模块 | 发送HTTP请求到StrategyServer服务接口，入参为：{user\_id=”2017031100001”, goods\_id:”1002001192”, rectype=”contentbased”, topN=”5”} | 根据该用户历史记录和商品标签返回推荐结果。 | 符合预期 |
| 基于内容的推荐模块 | 发送HTTP请求到StrategyServer服务接口，入参为：{user\_id=”2017031100001”, rectype=”contentbased”, topN=”5”} | 根据该用户历史记录返回推荐结果。 | 符合预期 |

基于用户的协同过滤测试，请求入参为 user\_id=”2017031100001”, rectype=”userbased” ,测试结果如下图所示。

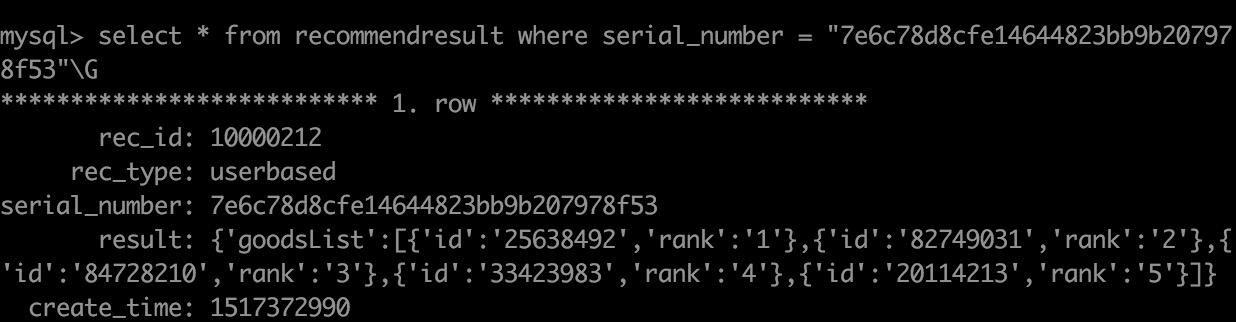


**图6-4 基于user的协同过滤推荐模块异步返回结果**

如图所以，我们使用postman发送http请求到推荐系统的应用服务器StrategyServer上，入参如图所示，返回结果是一个json字符串包括流水号serialnumber，当前时间戳timestamp和状态吗status\_code。

由于计算结果的返回需要一定时间，为了使得计算的速度不影响到StrategyServer的并发量，我们将请求设为异步，即当我们访问/recommend/StrategyServer接口时，该接口会生成流水号和spark任务提交的状态码以及时间戳并返回给前端，当前请求结束连接关闭。前端通过流水号发送查询请求，获取推荐结果。

我们根据返回的serialnumber查询数据库，获取推荐结果如图6-5所示。



**图6-5基于user推荐数据库详情**

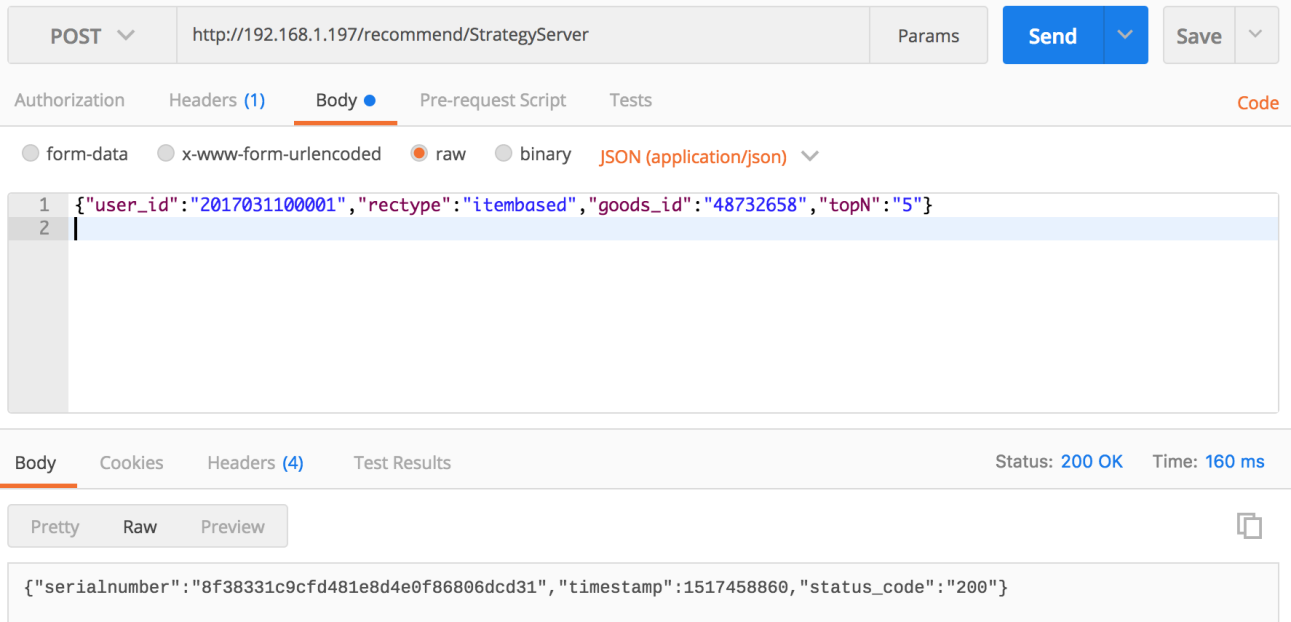
id对应的具体商品信息如表6.11所示。

**表6.11 基于user的协同过滤推荐结果**

|  |  |
| --- | --- |
| **goods\_id** | **商品标题** |
| 54334921 | Cherry樱桃MX8.0机械键盘RGB背光 |
| 14359823 | 音乐脉动BL Pulse2无线蓝牙迷你音响炫彩 |
| 14763491 | NIID slash 多功能运动防水背包 |
| 36124122 | Matador NANODRY 旅行快干毛巾 |
| 12342177 | 美安迪折叠U型枕 |

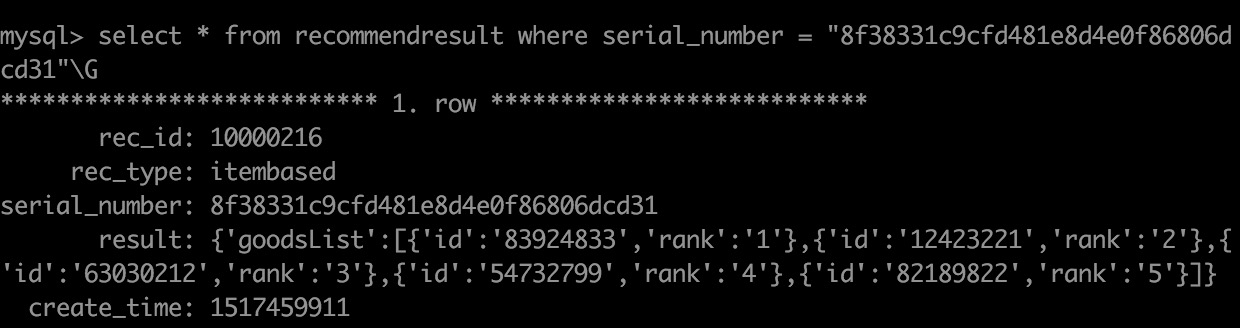
经过数据库核实，推荐结果与该用户近一周的商品喜好记录描述基本吻合，推荐结果符合预期效果。

基于item的协同过滤的测试流程与此类似，请求入参user\_id=”2017031100001”, goods\_id = “48732658”, rectype=”itembased”。测试结果如下图所示。



**图6-6 基于item的协同过滤推荐模块异步返回结果**

通过数据库查询流水号对应的推荐结果如图6-7。



**图6-7基于item推荐数据库详情**

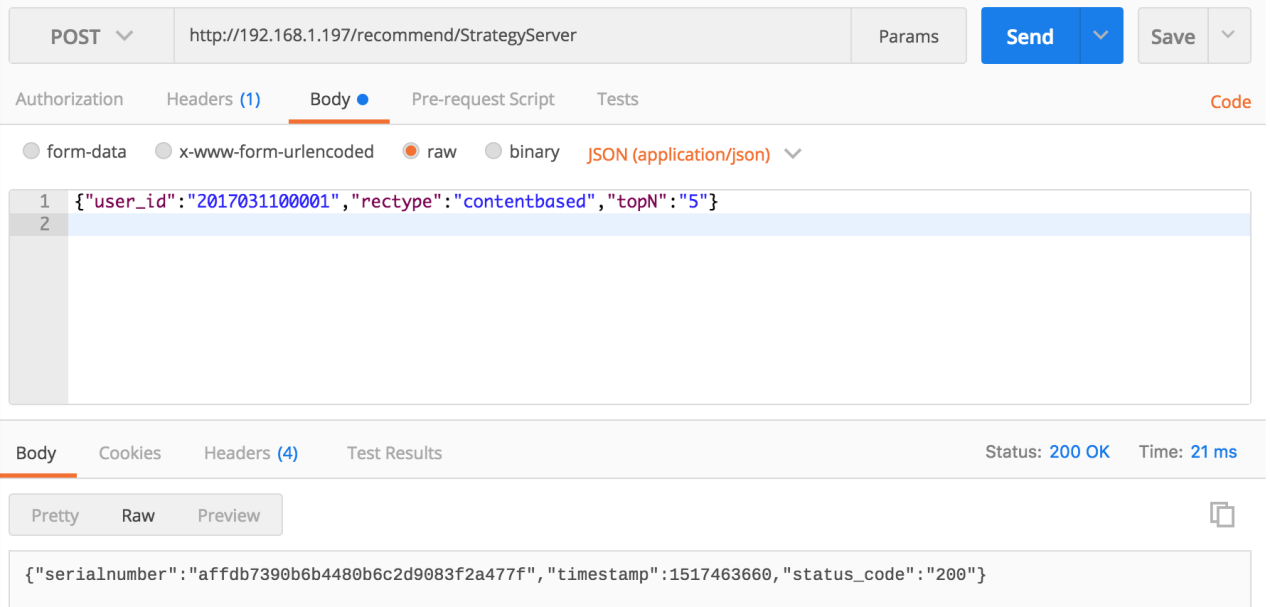
通过查询goodsinfo可知，请求的goods\_id 以及推荐结果中goodsList的对应的商品如表6.12所示。

**表6.12 基于item的协同过滤推荐结果**

|  |  |
| --- | --- |
| **goods\_id** | **商品标题** |
| 48732658(请求时的id) | 旗舰超薄科沃斯家用扫地机器人规划一体机 |
| 83924833 | 海尔Haier全自动扫地机器人智能家用 |
| 12425221 | 飞利浦S系列吸尘器FC6812夏季新品 |
| 63030212 | [日本**正负直**吸尘器XJC-Y010静音小型](https://item.taobao.com/item.htm?id=547652847103&ns=1&abbucket=11#detail)款 |
| 54732799 | 德国ElfBot擦窗机器人全自动智能电动清洁窗宝 |
| 82189822 | [美的手持无线吸尘器超静音迷你吸尘机H7-L021E](https://detail.tmall.com/item.htm?id=561756952872&standard=1) |

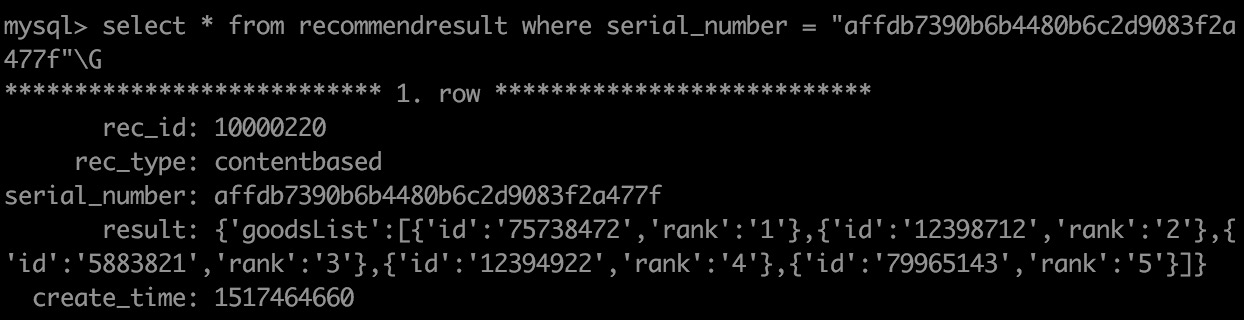
由返回数据可知，推荐结果与请求的商品数据具有较高的相关性，符合推荐预期。

基于内容的推荐模块测试，其入参与基于用户的协同过滤模块相似，只需要用户id，推荐类型和推荐记录数即可，Http请求结果如图6-8所示。



**图6-8 基于内容的推荐模块异步返回结果**

通过数据库查询流水号对应的推荐结果如图6-9。



**图6-9基于内容的推荐数据库详情**

通过查询goodsinfo可知，请求的goods\_id 以及推荐结果中goodsList的对应的商品如表6.13所示。

**表6.13 基于内容的推荐结果**

|  |  |
| --- | --- |
| **goods\_id** | **商品标题** |
| 75738472 | [天猫魔盒3Pro网络高清播放器家用电视机顶盒子](https://detail.tmall.com/item.htm?id=545676619032&ad_id=&am_id=&cm_id=140105335569ed55e27b&pm_id=&abbucket=11) |
| 12398712 | 定制原创小米5c/5x/note2小米手机壳软 |
| 58831821 | DJI大疆精灵智能航拍Phantom 3无人机 |
| 12394922 | Sphero 2.0 SPRK+遥控机器人蓝牙智能小球可编程 |
| 19965143 | iPhone6数据线苹果6s充电线器7plus智能断电加长 |

基于内容的推荐通过对物品特征和用户特征的匹配，从候选的待推广的商品列表中过滤筛选，提取topN的推荐结果，推荐给用户。推荐结果符合预期要求。

## 6.4非功能性测试

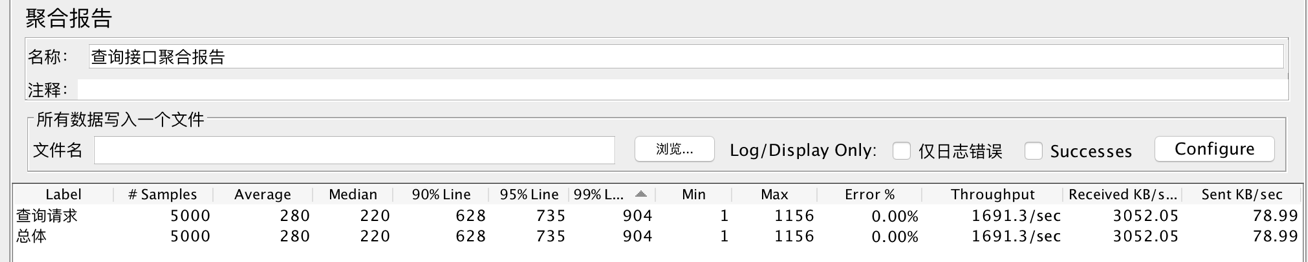
### 6.4.1并发性测试

我们使用Jmeter对服务接口做压力测试，测试模块包括订单的查询，购物车查询，商品详情页和秒杀系统的秒杀效果。

首先我们分别对${domain}/mall/queryOrder?user\_id=“2017031100001”,

${domain}/mall/queryCart?user\_id=”2017031100001”,以及${domain}/mall/goodsDetail?goods\_id=”10002001”分别发起5000并发量的压力测试。

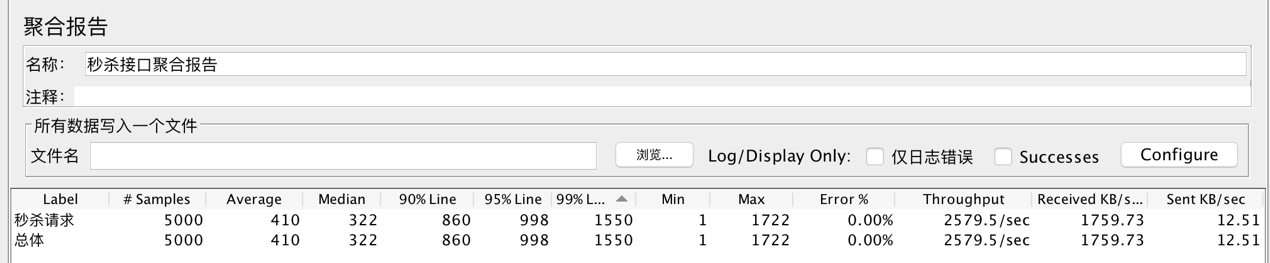
测试聚合报告如图6-10所示.



**图6-10 查询接口压力测试报告**

由测试报告可知，5000并发的请求服务器平均每秒可以处理1691个，错误率0%，基本满足预期要求。

同样，我们对秒杀模块接口进行了5000并发的压力测试，goods\_id为10002002，该商品库存500件，测试报告如图6-11所示。



**图6-11 秒杀接口压力测试报告**

由测试报告可知，由于引入缓存和消息队列，并且秒杀结果异步返回的情况下，该接口有着较小的数据传输量，且服务器每秒可以处理2579个请求，满足预期要求。

### 6.4.2兼容性测试

商城系统的兼容性问题主要体现在对浏览器的支持上，我们分别对表6.14几种常用的浏览器进行测试，测试对系统各个页面的兼容性支持良好，符合预期要求。

**表6.14**

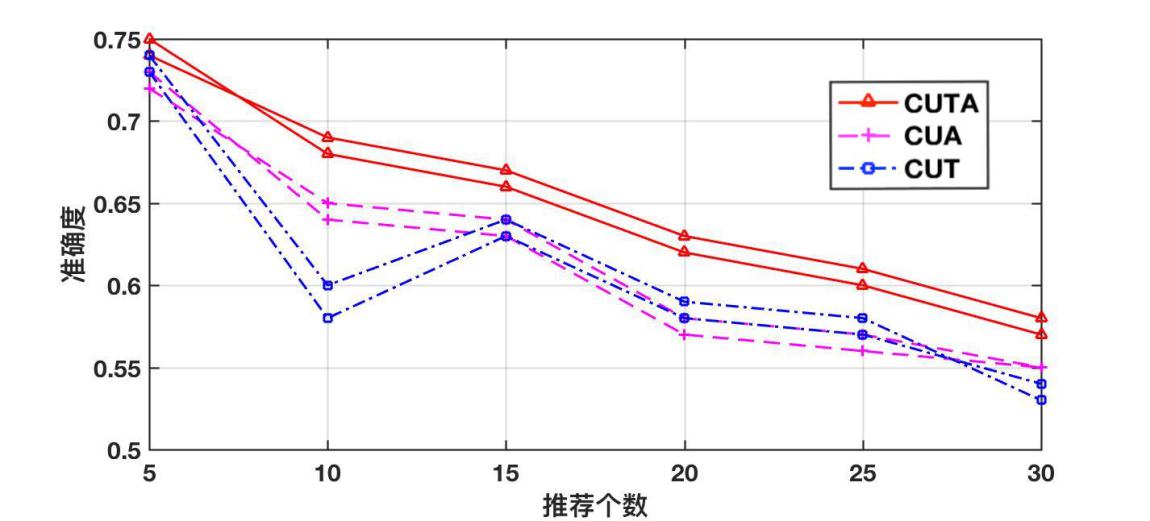
|  |  |
| --- | --- |
| **浏览器** | **兼容性** |
| Chrome浏览器 | 兼容 |
| 360浏览器 | 兼容 |
| Firefox火狐浏览器 | 兼容 |
| QQ浏览器 | 兼容 |

### 6.4.3冷启动仿真测试

(1) 不考虑新项目的不同推荐算法的准确度对比

为了验证我们提出的冷启动算法的性能，我们对4.5节提出的的三类推荐算法进行了相关的实验对比，我们使用MovieLens数据集，该数据集主要包括用户对电影的评分信息，电影的属性信息以及电影的标签信息等。我们从该数据集中选取至少被30个用户同时评论的电影，以及至少评论过10部电影的用户.经过筛选我们得到2035个用户以及3200部电影并将其50%划分为训练集，另外50%为测试集。

通过调整不同的的取值，并针对近邻个数为5，10，15，20，25，30的情况下进行了相关实验，实验结果如图6-11所示：



**图6-11不同推荐算法的准确度对比**

通过观察我们可以发现，CUTA算法的准确度明显高于CUA和CUT算法，而当邻域个数大于15时，调整的取值对CUA和CUT推荐的准确率并无明显影响。因此我们可以发现，在不考虑新项目的前提下，CUTA的准确率明显优于CUA和CUT算法。

(2) 改善冷启动问题的相关实验对比

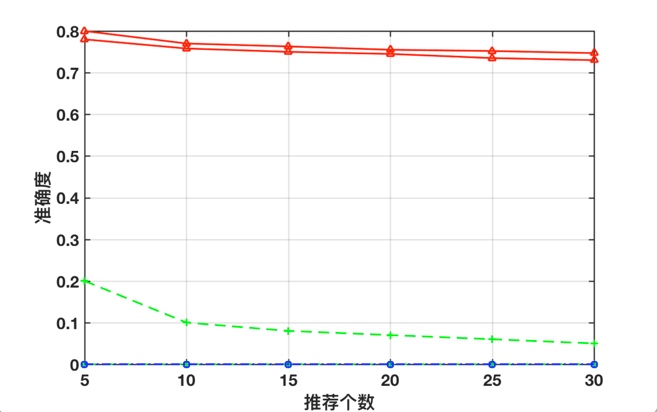
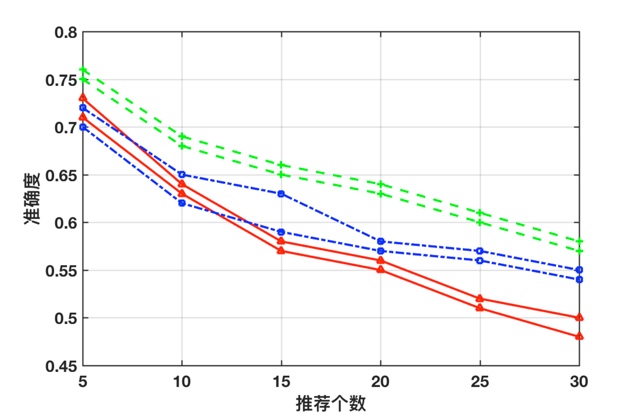
对于冷启动的问题，即通常用户并未曾对项目做出过任何评价，所以在数据集中并不会存在用户对项目的任何评分信息。因此当面临冷启动问题，我们主要会通过项目的基本属性信息和用户的时间权重两项综合来为用户推荐项目。

这里我们借鉴了一个新的概念：新颖度[35]，这里所说的新颖度是指在给目标用户推荐的列表中，新项目的个数占推荐项目的总数的度，即一个比值：

novelty = Recommended\_SUM /SUM (公式6.1)

其中novelty表示新颖度，Recommended\_SUM表示推荐列表中包含的新项目的个数，SUM表示项目总数。

首先我们将实验一中的测试集分成两部分，其中一部分是随机抽取150个项目当作新的项目，同时将训练集中这些对应项目的评分信息和标签信息全都置为0，其余未被抽取的部分为另一部分。的取值和实验一取值相同，并在不同邻域的情况下进行相关实验。



**图6-12解决冷启动问题算法的准确度对比** **图6-13解决冷启动问题算法的新颖度对比**

从实验结果(图6-12，图6-13)可以看出，本文提出的CUTA算法的准确率略高于CUT算法，当近邻值取5左右时，CUTATime的准确度略高于CUT，但是当近邻值大于5时，CUT算法的准确率依然可观。而对于新项目的推荐情况如图6-12，可知CUT算法所推荐的项目的新颖度为0，不能解决冷启动问题，综合考虑，CUTATime算法的新颖度最高，因此在处理新项目冷启动问题时其效果最佳。

## 6.5测试结论

我们对商城系统以及推荐系统按照测试用例进行了相关测试，对于商城系统对于不同状态下的用户请求能够，能够友好的与用户进行交互，各系统模块具有较好的健壮性。对于推荐系统，不同场景下各种推荐算法所给出的推荐结果也基本符合预期要求。

## 6.6本章小结

本章针对于论文中的两大系统的各个模块分别进行测试，主要介绍了测试环境、测试用例、以及测试结果。最后根据测试结果结合用例来判断功能模块是否达到预期要求。

# 第7章 总结与展望

## 7.1 总结

目前，大数据背景下，互联网海量信息中蕴含了很多有价值的信息，如何挖掘这些信息为人所用变得越来越重要，同时也是大数据平台首要需要解决的问题，其中推荐系统的出现，其本质正是通过挖掘用户的行为信息，更好的了解用户的喜好，从而帮助用户了解自己，并帮助用户从海量数据中找到自己需要的内容。如今推荐系统已经广泛的应用于众多大型互联网公司中，并成为公司战略推广的主要技术手段。通过深度挖掘用户行为信息和偏好信息，为用户提供有效的高质量的推荐结果，可以大大提高互联网公司运营服务能力，进一步提高公司营业额，因此优秀的高质量的推荐系统具有广泛的发展和服务场景。本文以设计实现了基于分布式计算框架spark的推荐系统，介绍了大数据的相关技术以及推荐系统的相关概念和面临的问题。并基于商场系统的历史数据的基础下进行了相关测评。

首先我们介绍了推荐系统所解决的问题，即大数据背景下的信息过载问题。然后我们详细介绍了Spark生态系统的相关框架以及所使用到的大数据处理的相关模块。使用Spark来代替MapReduce，使用Scala编程语言，从而实现更高的编程效率以及算法的执行效率。

在设计和实现部分，介绍了系统的整体架构，包括系统各层次的作用以及相关的功能，接着我们介绍了论文的主要内容，商城系统的主要模块的设计以及推荐系统的相关算法的实现。并针对推荐系统最常见的冷启动问题进行深入分析，并提出时间权重的概念，改善了推荐系统的冷启动问题。具体工作如下：

1. 在本文的开始部分介绍了商城系统及推荐系统的背景和存在意义，并介绍了国内外主流推荐算法的应用场景及优缺点。
2. 本文的第二部分主要介绍了推荐系统的计算部分所需要的技术支持，主要包括Spark分布式计算框架以及相关模块，通过spark实现算法的并行计算。介绍了商城后端使用的主流框架以及推荐系统使用的相关算法。
3. 本文的第三部分针对系统的需求分析、概要设计以及详细设计进行分别阐述，包括商城系统的登陆模块、购物车模块、订单管理模块。以及推荐系统的用户日志收集、计算框架设计、以及相关推荐模块的分析与实现。
4. 第四部分主要是针对商城系统和推荐系统进行测试，主要包括模块的功能性测试、健壮性测试、并发性测试等。测试结果表明各模块基本满足设计要求，符合预期效果。

本人通过上述工作，完成了B2C商城及推荐系统的设计与实现，本在此基础上完成论文撰写。

## 7.2 展望

本文基于B2C商城系统的数据基础上，设计了推荐系统根据策略模式来为不同的应用场景提供推荐服务，通过测试，商城系统和推荐系统在功能性、健壮性等方面基本符合预期要求。不过该系统仍存在很多不足和需要改进的方面。

(1) 整个系统如何在数据快速增长的情况下，有效的调配计算资源，进行有效的推荐。

(2) 系统设计本身没有考虑到安全因素，例如商家恶意操作影响推荐结果以谋取利益的情况。

(3) 针对算法提出相应的优化方案，考虑扩展更多的应用场景，并考虑在应用环境下设计相应的特征向量。

(4) 本文的所有实验连接线上数据库进行，具体推荐结果的准确性暂时通过历史记录和查看用户喜好倾向判断，没有标准的量化数据。

# 参考文献

1. 朱扬勇，孙婧.推荐系统研究进展[J].计算机科学与探索，2015，9( 5) : 52-61.
2. 赵良辉，熊作贞.电子商务推荐系统综述及发展研究[J].电子商务，2013( 12) : 58-60.
3. 杨凤萍. 基于神经网络集成和用户偏好模型的协同过滤推荐算法研究[D]. 华中师范大学, 2016.
4. 吴颜, 沈洁, 顾天竺,等. 协同过滤推荐系统中数据稀疏问题的解决[J]. 计算机应用研究, 2007, 24(06):94-97.
5. 吴奕, 乐嘉锦. 基于上下文的分布式协同过滤推荐技术[J]. 计算机工程, 2010, 36(12):90-93.
6. R. Burke, M. O'Mahony, and N. Hurley. Robust collaborative recommendation. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, Recommender Systems Handbook, chapter 25, pages 805-835.
7. 刘海, 卢慧, 阮金花,等. 基于"用户画像"挖掘的精准营销细分模型研究[J]. 丝绸, 2015, 52(12):37-42.
8. Li G, Li. A new algorithm of cold-start in a collaboration filtering system. Journal of Shandong University (engineering science),2012,2(24):12-17 (in Chinese with English abstract).
9. 梁彦.基于分布式平台Spark和YARN的数据挖掘算法的并行化研究[D].中山大学,2014.
10. 孙冬婷, 何涛, 张福海. 推荐系统中的冷启动问题研究综述[J]. 计算机与现代化, 2012(05):59-63.
11. Kraska T, Talwalkar A, Duchi J, et al. MLbase: A Distributed Machine-learning System[J]. Cidr, 2013.
12. 冯琳.集群计算引擎Spark中内存优化研究与实现[D].清华大学，2013.
13. Huang C Q, Yang S Q, Tang J C, et al. RDDShare: Reusing Results of Spark RDD[C]// IEEE International Conference on Data Science in Cyberspace. IEEE, 2016:370-375.
14. 李崇欣. 分布式数据库Hbase: the definitive guide[M]. Sebastopol: O’Reilly Media, Inc. 2015.
15. 刘冬鸣. 分布式组件Zookeeper事务一致性探析[J]. 信息通信, 2017(06).
16. Maillo J, Ramírez S, Triguero I, et al. kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors Classifier for Big Data[J]. Knowledge-Based Systems, 2017.
17. 郭艳红. 推荐系统的协同过滤其法与应用研究[D]. 大连:大连理工大学.2014.
18. 陈清浩.基于SVD的协同过滤推荐算法研究[D].西南交通大学.2015.
19. Kim J K, Cho Y H. Using Web Usage Mining and SVD to Improve E-commerce Recommendation Quality[J]. 2003, 2891:86-97.
20. 王媛媛, 李翔. 基于人口统计学的改进聚类模型协同过滤算法[J]. 计算机科学, 2017, 44(03):63-69.
21. Yoshida T, Kitayama D. A feature terms extraction method based on polarity analysis of customer reviews for content-based recommendation[C]// The, International Conference. 2016:7-11.
22. Cao J,Wu ZWang Y, et al. Hybrid collaborative filtering algorithm for bidirectional Web service recommendation[J]. Knowledge and Information Systems,2013,36(3):607-627.
23. YIN H Z，CUI B，SUN Y Z，et al.Lcars: a spatial item recommender system[J].Journal of the ACM Transactions on Information Systems，2014，32 ( 3) : 1-11，37.
24. 张亮，赵娜.改进的协同过滤推荐算法[J].计算机系统应用，2016，25( 7) : 45-52.
25. Polatidis N, Georgiadis C K. A multi-level collaborative filtering method that improves recommendations[M]. Pergamon Press, Inc. 2016.
26. Huang CG, Yin J, Wang J, Liu YB, Wang JH. Uncertain neighbors collaborative filtering recommendation algorithm. Chinese Journal of Computers,2010,8(33):1370-1377(in Chinese with English abstract).
27. Park S T, Chu W.Pairwise preference regression for cold-start recommendation[M].
28. 程飞，贾彩燕.一种基于用户相似性的协同过滤推荐算法[J].计算机工程与科学，2013，35( 5) : 161-165.
29. He X, Zhang H, Kan M Y, et al. Fast Matrix Factorization for Online Recommendation with Implicit Feedback[J]. 2017:549-558.
30. W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A learning-rate schedule for stochastic gradient methods to matrix factorization. In Advances in Knowledge Discovery and Data Mining, pages 442–455. Springer, 2015.
31. Jesús Bobadilla,Fernando Ortega,Antonio Hernando,Jesús Bernal.  [A collaborative filtering approach to mitigate the new user cold start problem](http://www.cnki.net/kcms/detail/detail.aspx?filename=SJES13011501719153&dbcode=SSJD&v=Mjc0MTdvOUVZK29HRFhrNm9HY1U2VDU4UUh1VXJtUXpDOGZpVEw3d1plWnVFU25zVTdqSUlWd1JhQnM9TmlmT2ZiSzdIdEROcQ==)[J]. Knowledge-Based Systems . 2011.
32. 丁少衡，姬东鸿，王路路.基于用户属性和评分的协同过滤推荐算法[J].计算机工程与设计，2015(2):487-491.
33. 李改. 一种解决协同过滤系统冷启动问题的新算法[J]. 山东大学学报(工学版),2012.
34. Zheng N, Li QD. A recommender system based on tag and time information for social tagging systems. Expert Systems with Applications, 2011,38(4):457-485.
35. 刘建国，周涛.个性化推荐系统的研究进展[J].自然科学进展,2009,19(1);1-15.

# 致 谢

首先，感谢我的母校中国科学技术大学给了我读研深造的机会，并且给了我良好的教育。在读研期间，学校的学习氛围以及严谨的治学态度给我留下深刻的印象，同时学校尽可能的为学生提供学习资料与良好的环境，增长了学生的见识，我以我的母校而自豪。

其次，要感谢我的班主任朱洪军老师，感谢他在读研期间对我的生活和学习上的关心与帮助，朱洪军老师在教学上的认真和对学生负责的态度给我留下深刻印象，同时也是我学习的榜样。最重要的是要感谢戴礼荣老师作为我的校内导师，感谢戴礼荣老师细心指导我的开题报告及论文的撰写，戴礼荣老师无论多忙多晚都会耐心为我解决疑惑，同时给予技术上的支持以及论文修改的建议。若没有戴礼荣老师的细心指导，论文的完成非常的困难。在这里非常感谢戴礼荣老师的指导，戴礼荣老师对学术要求严谨，同时又平易近人，给我留下深刻印象，衷心祝愿她永远健康快乐，工作顺利。

再次，要感谢平安科技(深圳)有限公司给了我实习的机会，使我了解了企业工作和学校学习的区别，让我提前进入工作氛围。同时感谢企业导师程宁，感谢他在工作中给予我的指导，为我解决了很多工作中的疑惑，同时为我的论文选题以及工作学习提供很多宝贵的建议，感谢她对我的工作和生活的关心。也要感谢同事们在工作上给予的技术指导，使我的技术得到很大提升。

最后，感谢我的家人和朋友，是他们一直在背后默默的支持和鼓励我，在遇到挫折时他们总会为我加油。感谢父母对我的栽培和付出，希望他们永远健康快乐。

2017年