1. 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。

1. 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

3.3 关键模块的设计与实现

（1）Bootstrap前端

（2）音乐评分

（3）处理流程

3.4 音乐推荐算法设计

3.5 本章小结

1. 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

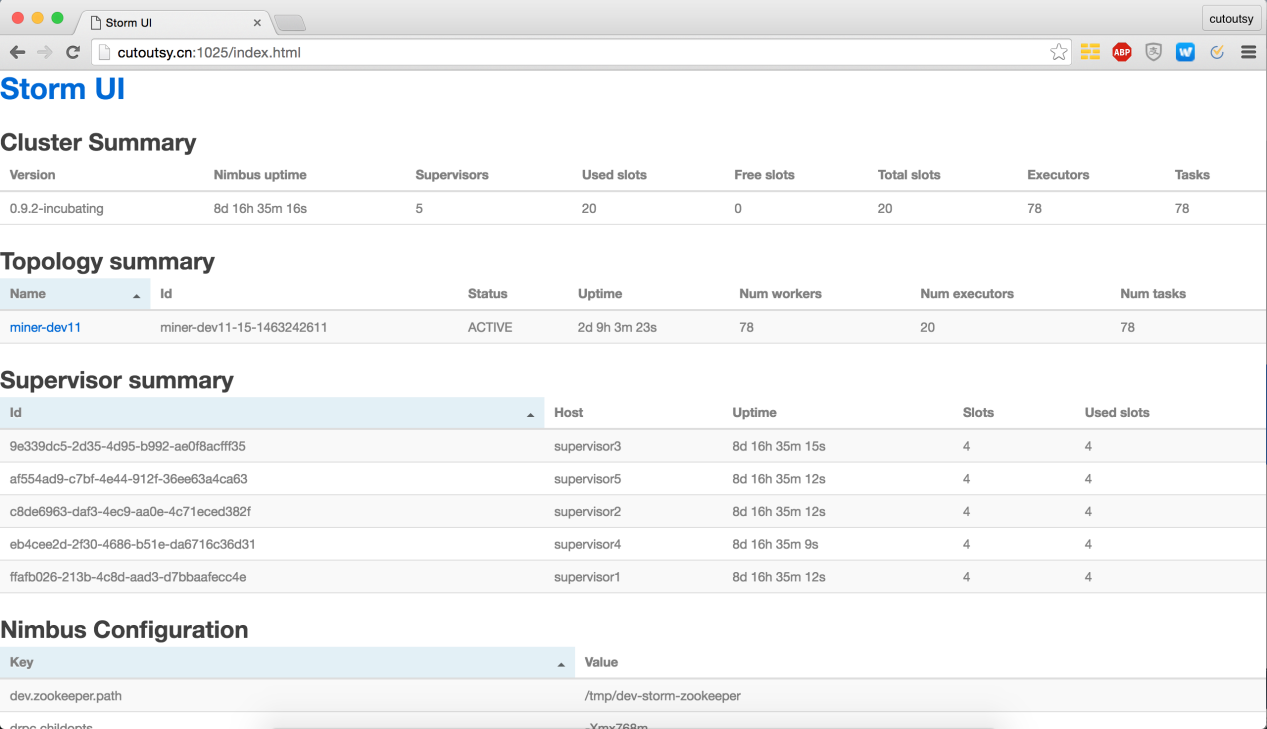
使用storm脚本启动守护进程（包括nimbus、supervisor、ui）

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

4.4 实时拓扑编写

4.5 测试用例编写