**文章编号**　1672-6634(2023)06-0027-08　　**DOI**　10.19728/j.issn1672-6634.2023050011

# 基于 LFM 算法的 SpringCloud 分布式购物系统

邵阳阳,徐子良,姜玉波,李成龙,田　甜

(山东建筑大学 计算机科学与技术学院,山东 济南 250101)

**摘　要**　数据个性化推荐缺乏、无意义流量对系统的影响,以及后期代码维护困难是目前在线购物平台面临的主要问题。基于隐语义模型(Latent Factor Model, LFM)和 SpringCloud 框架,实现了分布式购物系统。针对数据个性化推荐问题,通过数据实时同步工具 Maxwell 采集用户数据并推送至消息中间件 RabbitMQ,日志收集系统 Flume 接收数据并存储至分布式架构 Hadoop 中,由推荐服务读取数据,基于 LFM 的协同过滤算法实现数据个性化推荐;针对系统冗余压力问题,利用微服务网关 Zuul 限制子系统路由访问量,以减少子系统的流量压力;针对代码维护问题,使用 Zuul 和 Nacos 注册中心实现动态路由代理,自动分发流量至新增服务器已注册的子系统中。上述技术提高了在线购物平台的拓展性和应对高并发流量的处理能力,降低了代码维护难度。

**关键词**　购物平台;微服务;推荐系统;协同过滤
**中图分类号**　TP311
**文献标识码**　A　　**开放科学(资源服务)标识码(OSID)**

# SpringCloud Distributed Shopping System Based on LFM Algorithm

SHAO Yangyang, XU Ziliang, JIANG Yubo, LI Chenglong, TIAN Tian

(School of Computer Science and Technology, Shandong Jianzhu University, Jinan 250101, China)

**Abstract**　The main problems facing online shopping platforms are the lack of personal data recommendation, the impact of meaningless traffic on the system, and the difficulty of code maintenance in the later period. A distributed shopping system is implemented based on Latent Factor Model (LFM) and SpringCloud framework. The user data is collected through the real-time data synchronization tool Maxwell and pushed to the message-oriented middleware RabbitMQ. The log collection system Flume receives data and stores it in the distributed Hadoop architecture. The data is taken by the recommendation service. Finally, the LFM-based collaborative filtering algorithm realizes personalized data recommendation. In order to reduce the subsystem's traffic pressure, the micro-service gateway Zuul is used to limit the subsystem's routing access. Zuul and Nacos registries are used to implement dynamic routing proxies to automatically distribute traffic to the subsystem registered by the new server. In this way, the code is easy to maintain. Based on the above techniques, the extensibility of the online shopping platform is improved, the processing ability to deal with high concurrent traffic is guaranteed, and the difficulty of code maintenance is reduced.

**Key words**　shopping platform; micro services; recommendation system; collaborative filtering

**收稿日期**:2023-05-23
**基金项目**:国家自然科学基金项目(62102235)资助
**通讯作者**:田甜,女,汉族,博士,副教授,研究方向:智能软件工程,E-mail:tiantian@sdjzu.edu.cn。

# 1 引言

互联网技术的迅速发展使得人类获得信息的途径更加广泛,推荐算法是从繁杂信息中快速获取个性化需求数据的有力工具[1],以机器学习为代表的新一代人工智能技术已得到了广泛应用[2]。隐语义模型(Latent Factor Model,LFM)算法是一种典型的推荐算法,常被用在推荐系统和文本分类中。利用推荐算法实现用户数据个性化推荐是开发在线购物平台首要考虑的问题。同时,缓解无意义流量产生的冗余访问压力,降低后期代码维护难度也是在线购物平台需要解决的核心问题。

针对在线购物平台的个性化数据推荐、无意义流量和代码维护等问题,本文给出了相应的解决方案。

首先,系统通过数据实时同步工具 Maxwell[3]实时读取 MySQL 二进制日志文件 binlog,采集用户浏览数据和购买数据,并推送至开源消息代理软件 RabbitMQ[4]中;由日志收集系统 Flume[5]接收 RabbitMQ 中的数据并存储至分布式架构 Hadoop[6]中;利用推荐服务读取 Hadoop 中的数据,实现用户喜爱商品的预测和推荐。其次,针对无意义流量对子系统产生冗余压力的问题,系统利用微服务网关 Zuul[7],限制窗口期时间内同一 IP 地址对相同子系统路由的访问量,在网关服务器处拦截流量,以减少子系统的流量压力。最后,为了降低代码维护难度,系统使用微服务网关 Zuul 和 Nacos 注册中心[8]实现动态路由代理,使得在增加服务器时,由动态路由代理自动将流量分发至新增服务器已注册的子系统中,Web 端无需修改任何路由代码。

本文结构如下:第 2 节综述相关工作,第 3 节阐述基于 LFM 的协同过滤推荐算法,第 4 节介绍基于 Spring Cloud 的分布式平台,第 5 节是基于 LFM 算法的 SpringCloud 分布式购物系统的功能实现,第 6 节总结全文并展望下一步工作。

# 2 相关工作

本节从协同过滤推荐算法和分布式架构两方面综述相关工作。协同过滤是最经典且目前应用最为广泛的推荐算法之一[9],该算法无需特定领域知识,主要包括在线协同和离线过滤两部分。所谓在线协同,即通过在线数据寻找用户可能的偏好内容;而离线过滤,则是过滤掉推荐值低或推荐值高但已使用过的数据。基本的协同过滤推荐算法主要分为基于用户的协同过滤(User-Based Collaborative Filtering)[10]、基于物品的协同过滤(Item-Based Collaborative Filtering)[11],以及基于模型的协同过滤(Model-Based Collaborative Filtering)[12]三 种。

本文采用的 LFM 推荐算法,是一种隐含语义分析技术,最早在文本挖掘领域被提出,用于表示文本的潜在语义[13]。LFM 推荐算法不依赖于共同评分矩阵,基本思想如下:将用户和商品的内容分别映射到真实含义未知的特征向量中,通过发掘用户与其所关注的商品间的隐含联系,判断两者之间潜在的关注关系并做出推荐。例如,在推荐系统中能够基于用户的行为对商品进行自动聚类,即将商品划分为不同类别,该类别就可以理解为用户感兴趣的商品。近些年,该技术被不断应用到其它领域中,并得到不错的应用效果,但其性能容易受到数据稀疏和冷启动问题的制约[14]。

冷启动是指系统刚开始运行时,由于后台整合的数据不足,无法合理地分析数据,从而无法给客户提供精准的推荐服务[14]。为解决该问题,郑鹏等提出一种用户隐式信任计算方法,建立用户间接信任,在一定程度上改善了冷启动问题[15]。Kai 等通过融合用户评分项目和其类型计算用户相似度,并设计相应的协同过滤算法,实现了推荐结果的高准确性[16]。Duricic 等将网络科学中的一种度量方法,正则等价,应用到信任网络中生成一个相似度矩阵,用于选 $k$ 最近邻进行推荐,在一定程度上解决了冷启动问题[17]。

用户量的激增给系统的负载能力带来了极大的挑战。对单一应用架构而言,由于应用程序中的所有功能在同一系统内实现,巨大访问量易导致服务崩溃,同时,各种问题也会随着需求迭代接踵而来[18]。作为一种关注全局的分布式框架,SpringCloud 是解决上述问题的有力工具。基于 SpringCloud 架构,蔡勋玮等实现了新型供电系统数据挖掘方法的应用,更好地掌握了供电负荷量的实际变化规律[19];王蓉等设计并实现

医疗信息共享平台，进一步提高医疗信息的共享和交换能力[20]；Yang 等设计在线点餐系统，在准确判断食品配料量的同时满足不同居民的就餐需求[21]。

基于上述工作，本文利用 SpringCloud 以及 LFM 算法，实现了基于 LFM 算法的 SpringCloud 分布式购物系统。针对所有微服务接口，系统采用 Spring Security 框架进行统一鉴权；使用微服务网关 Zuul 进行全局路由代理，配合 Nacos 注册中心和 Zuul 实现动态路由；利用数据实时同步工具 Maxwell 采集用户浏览信息，由 Maxwell 推送信息至开源消息代理软件 RabbitMQ；由日志收集系统 Flume 主动接收 RabbitMQ 中的数据并下沉至分布式架构 Hadoop 中。在该分布式购物系统中，数据采集模块通过模块初始化以保持日常应用中稳定的采集与输出，经过数据预处理后由推荐模块执行推荐算法获得推荐数据。

## 3　基于 LFM 的协同过滤推荐算法

近些年，企业对个性化推荐系统的需求在不断增长，其主要原因是，可以帮助用户过滤信息，进而发掘用户所需内容。无论是影视推荐、电子商务，还是社交平台，推荐算法都扮演着重要角色[22]。

LFM 作为一种常用的推荐算法，其基本思想为：根据用户的历史行为数据分析其偏好特征，并基于偏好特征为用户 $u$ 推荐相应的商品 $i$ [22,23]。事实上，我们只能获得用户的行为数据共现矩阵 $\boldsymbol{R}$。LFM 算法通过引入潜在因子表示用户和商品，最终将共现矩阵 $\boldsymbol{R}$ 分解为两个低维度的矩阵：潜在因子-用户矩阵 $\boldsymbol{p}_u$ 和潜在因子-商品矩阵 $\boldsymbol{q}_i$。LFM 原理的矩阵表示如图 1 所示。

| | $i1$ | $i2$ | $i3$ | $i4$ |
|---|---|---|---|---|
| $u1$ | $r11$ | $r12$ | $r13$ | $r14$ |
| $u2$ | $r21$ | $r22$ | $r23$ | $r24$ |
| $u3$ | $r31$ | $r32$ | $r33$ | $r34$ |

$\boldsymbol{R}$

=

| | $k1$ | $k2$ | $k3$ |
|---|---|---|---|
| $u1$ | $p11$ | $p12$ | $p13$ |
| $u2$ | $p21$ | $p22$ | $p23$ |
| $u3$ | $p31$ | $p32$ | $p33$ |

$\boldsymbol{p}_u$

×

| | $i1$ | $i2$ | $i3$ | $i4$ |
|---|---|---|---|---|
| $k1$ | $q11$ | $q12$ | $q13$ | $q14$ |
| $k2$ | $q21$ | $q22$ | $q23$ | $q24$ |
| $k3$ | $q31$ | $q32$ | $q33$ | $q34$ |

$\boldsymbol{q}_i$

图 1　LFM 矩阵表示

潜在因子-用户矩阵 $\boldsymbol{p}_u$ 表述为 $\boldsymbol{p}_u = m \times k$，潜在因子-商品矩阵 $\boldsymbol{q}_i$ 表述为 $\boldsymbol{q}_i = k \times n$，其中，$m$ 表示用户的数量，$n$ 表示商品的数量，$k$ 表示潜在因子，即影响用户 $u$ 对商品 $i$ 偏好程度的隐藏特征。

共现矩阵 $\boldsymbol{R}$ 和低维矩阵 $\boldsymbol{p}_u$ 和 $\boldsymbol{q}_i$ 的关系如

$$\boldsymbol{R} = \boldsymbol{p}_u \boldsymbol{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik}, \tag{1}$$

式中 $K$ 表示潜在因子的数量；$p_{uk}$ 表示用户 $u$ 与第 $k$ 个潜在因子的关系；$q_{ik}$ 表示商品 $i$ 与第 $k$ 个潜在因子的关系；$p_{uk}$ 和 $q_{ik}$ 作为参数用于模型训练。LFM 算法通过计算式(1)获得用户对商品的偏好程度。

采用的损失函数为平方误差损失函数 loss1，如

$$\text{loss1} = \sum_{(u,i) \in D} (r(u,i) - r^{\text{LFM}}(u,i))^2, \tag{2}$$

式中 $r(u,i)$ 表示共现矩阵中的值，即训练样本中的标签，点击时值为 1，否则为 0；$r^{\text{LFM}}(u,i)$ 表示模型预估用户对商品喜爱程度的值；$D$ 表示所有训练样本集合；当 $r(u,i)$ 和 $r^{\text{LFM}}(u,i)$ 越接近时，损失函数越小。

在式(2)中，为防止模型过拟合，加入正则化表达式 $\lambda(\| p_u \| + \| q_i \|)^2$ 限制模型参数，以增强其泛化能力。由此得到新的损失函数 loss2，如

$$\text{loss2} = \sum_{(u,i) \in D} (r(u,i) - r^{\text{LFM}}(u,i))^2 + \lambda(\| p_u \| + \| q_i \|)^2, \tag{3}$$

式中 $\lambda$ 表示正则化系数，基于不同场景的重复试验得到。

此外，使用随机梯度下降（Stochastic Gradient Descent，SGD）算法[24]进一步优化求解最小化损失函数，当经过足够次数的迭代时，SGD 可收敛至局部最优解。具体地说，首先，获取未知数 $p_{uk}$ 和 $q_{ik}$ 的偏导值，如式(4)和式(5)所示

$$\frac{\partial \text{loss2}}{\partial p_{uk}} = -2q_{ik}(r(u,i) - r^{\text{LFM}}(u,i)) + 2\lambda p_{uk}, \tag{4}$$

$$\frac{\partial \text{loss2}}{\partial q_{ik}} = -2p_{uk}(r(u,i) - r^{\text{LFM}}(u,i)) + 2\lambda q_{ik}, \tag{5}$$

然后,基于 SGD 进行迭代计算,$p_{uk}$ 和 $q_{ik}$ 沿梯度的负方向不断更新,对应参数更新公式如式(6)和式(7)所示,达到预期目标时,可停止训练。

$$p_{uk} = p_{uk} + \alpha[q_{ik}(r(u,i) - r^{\text{LFM}}(u,i))] - \lambda p_{uk}, \tag{6}$$

$$q_{ik} = q_{ik} + \alpha[p_{uk}(r(u,i) - r^{\text{LFM}}(u,i))] - \lambda q_{ik}, \tag{7}$$

式中 $\alpha$ 为学习率,学习率随迭代次数不断变化。

LFM 推荐算法实现如算法 1 所示。第 2~3 行定义并初始化矩阵 $p_u$、$q_i$ 以及潜在因子个数 $K$,迭代次数 $N$,学习率 $\alpha$,正则化系数 $\lambda$;第 4~11 行是模型训练过程,其中,从第 4 行开始迭代;第 5~6 行通过从数据集中抽取 $u$ 及 $i$ 样本实现优化计算;第 7 行用于获取用户对商品的偏好程度;第 8 行基于当前参数值计算误差值 $E(u,i)$;第 9~11 行基于式(6)和式(7)实现参数优化。

算法 1. LFM 推荐算法

```
1. BEGIN
2. def  LFM(u_i, K, N, α, λ):
3.      [p_u, q_i] = Init(u_i,  K)
4.          for step in range(0,  N):
5.              for u, i in u_i.iterms():
6.                  samples = Rand-Select-Negative-Samples(i)
7.                  for i,  r(u,i)  in samples.i ():
8.                      E(u,i) = (r(u,i) - r^LFM(u,i))
9.                      for k  in range(0,  K):
10.                         p[u][k] += α * E(u,i) * q[i][k] - λ * p[u][k]
11.                         q[i][k] += α * E(u,i) * p[u][k] - λ * q[i][k]
12. END
```

## 4 基于 SpringCloud 的分布式平台

基于 SpringCloud 的分布式平台利用 LFM 的协同过滤推荐算法实现用户推荐,提供了包括前端应用、后台服务、算法实现和平台部署等多方位的闭环业务实现方案,其业务实现流程如图 2 所示。基于 SpringCloud 的分布式平台分为综合业务、数据存储和日志服务三大部分。

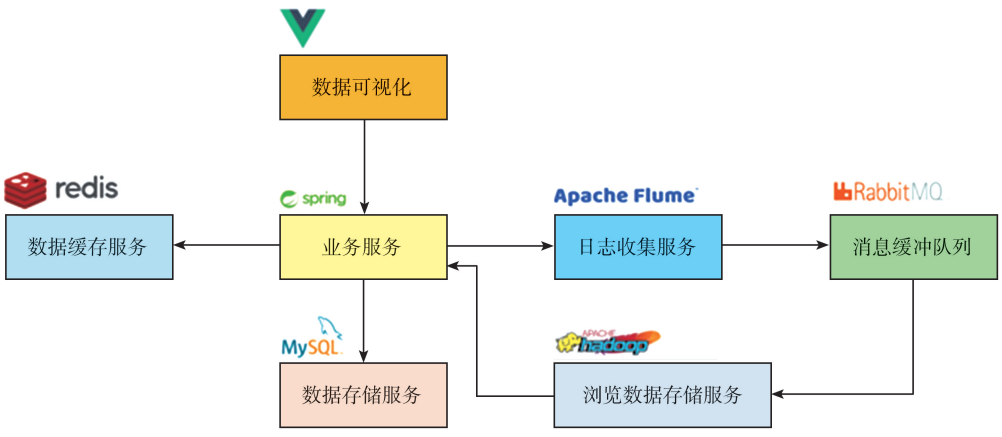


图 2 业务实现方案

综合业务主要包括 3 部分:(1) 用户可视化:实现用户交互和业务数据展示,主体采用 VUE 框架实现并

部署在 Nginx 服务上；(2) 业务服务：通过 Spring 进行构建，对接业务需求，并部署在应用服务器 Tomcat 上，以实现 JavaEE 层面业务逻辑；(3) 推荐服务：利用 Hadoop 中存储的数据，使用推荐算法为用户进行个性化推荐。

数据存储主要包括 3 部分：(1) 业务数据库：采用 MySQL 作为主数据库，负责平台业务逻辑数据的存储；(2) 缓存数据库：采用 Redis[25] 作为缓存数据库，为推荐系统实时高速获取需求数据提供支撑；(3) 浏览数据存储服务：使用 Hadoop 利用集群进行高速运算和存储，实现分布式文件存储系统。

日志服务主要包括 2 部分：(1) 日志采集服务：通过数据采集系统 Flume-NG 采集业务平台中用户的评分行为，实时发送到 RabbitMQ 中；(2) 消息缓冲服务：采用 RabbitMQ 作为流式数据的缓存组件，接受来自 Flume 的数据采集及业务请求，将数据推送到 Hadoop 存储系统部分。基于此，实现基于 SpringCloud 框架，兼顾数据存储、日志采集和系统推荐功能的高可用分布式平台架构。

## 5　分布式购物系统功能实现

本节是基于 LFM 算法的 SpringCloud 分布式购物系统的功能实现，主要包括系统架构和主要功能。

### 5.1　系统架构

本系统中服务端包括用户端、商家端、管理端和运维端四个部分，内部系统包括 Web 服务器、商品服务、订单服务、用户管理服务、推荐服务、统一鉴权中心服务以及数据库服务器。

用户通过 Web 服务器域名进行访问，由 Nginx 路由代理分散用户访问流量至不同服务器，从而实现 Web 端负载均衡；再由 Web 端通过统一鉴权中心进行权限验证，以配合 Zuul 和 Nacos 注册中心将流量分配至不同服务器上的微服务模块，由此实现服务端的负载均衡。系统中各模块均与数据库服务器进行数据交互，产生的部分数据由 Maxwell 实时读取并推送至 RabbitMQ，由 Flume 采集数据下沉至 Hadoop 中存储。推荐服务模块基于 LFM 算法实现采集推荐，推荐服务会将产生的推荐信息存储至数据库服务器中，由前端调用接口实现数据交互，完成前后端交互。系统架构及交互方式如图 3 所示。

在推荐服务中，使用 LFM 算法的主要优势包括两方面：一是无需关心分类角度问题，结果均基于用户行为统计自动聚类；二是无需关心分类粒度问题，粒度的大小取决于潜在因子数量 $K$ 。SpringCloud 架构耦合性小，在系统开发过程中，服务器端的多个模块不会相互影响。

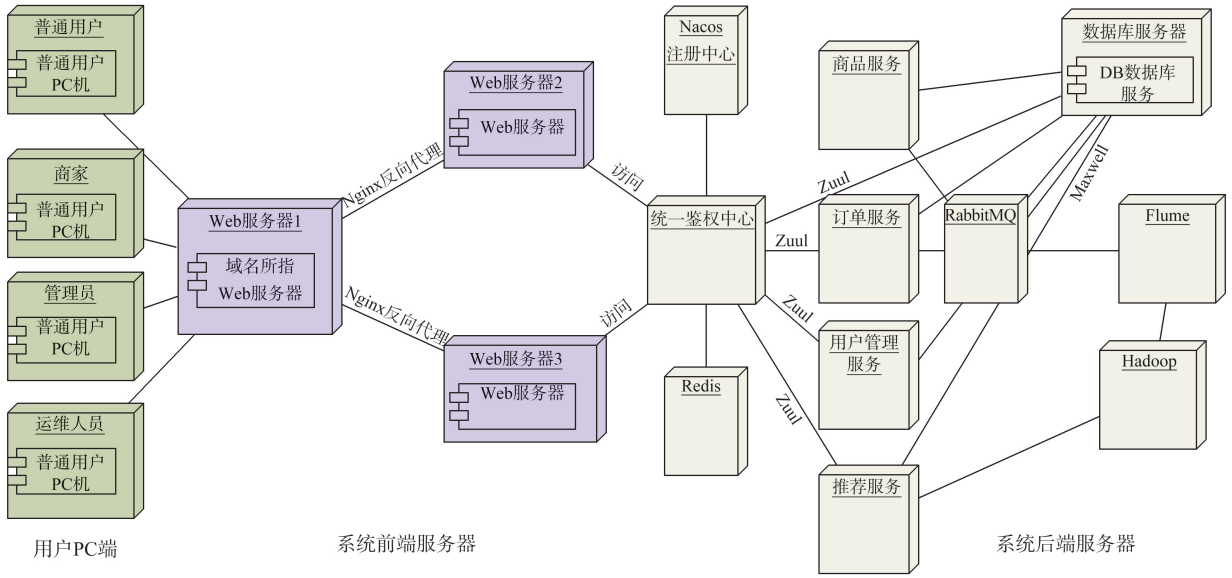


图 3　系统总体架构

### 5.2　主要功能实现

5.2.1　路由代理配置。(1) Zuul+Nacos 动态路由代理。实现动态路由代理需要配置两个中心：配置和注

册中心,其中,配置中心总结各个微服务模块的配置文件,通过路径映射完成配置;注册中心注册各个微服务模块的功能和端口号;系统采用 Nacos 集成注册中心和配置中心。

Zuul+Nacos 动态路由代理服务在 Nacos 注册中心被发现,Zuul 通过读取对应路由实现 API 网关,新增服务在 Nacos 中完成注册,由微服务网关 Zuul 实现服务的动态代理,对应流程如图 4 所示。

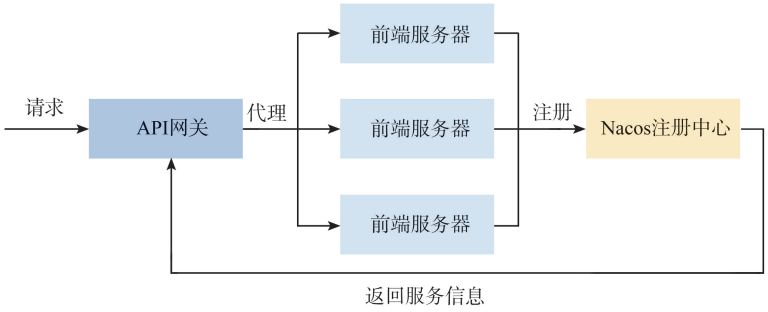


图 4 动态路由代理流程

(2) Nginx 路由代理配置。Nginx 实现客户端的负载均衡,正常情况下,其处理单次请求响应速度更快;高并发请求情况下,Nginx 的响应请求速度也快于其它 Web 服务器。Nginx 单机支持 10 万以上的并发连接,利用 Nginx 的特性,将用户 URL 代理到指定页面,统一管理上传文件位置,实现 Nginx 路由代理配置,对应流程如图 5 所示。

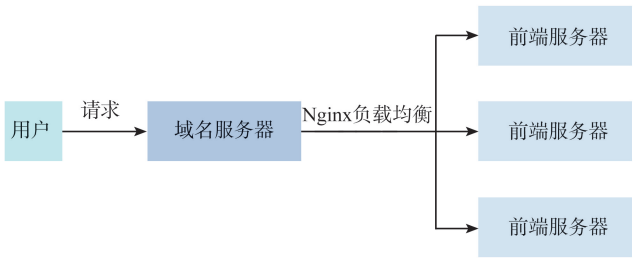


图 5 Nginx 路由代理配置

5.2.2 RabbitMQ 数据源配置。系统通过 Maxwell 采集 MySQL 中用户产生的数据信息,主动推送至 RabbitMQ,再由 RabbitMQ 推送至 Flume,最终下沉至 Hadoop 中。鉴于 Flume 无法直接接收来自 RabbitMQ 的数据,需要针对 Flume 单独开发插件,以允许 Flume 获取 RabbitMQ 中的数据。Flume 采集 RabbitMQ 数据推送至 Hadoop 中。实现流程如图 6 所示。

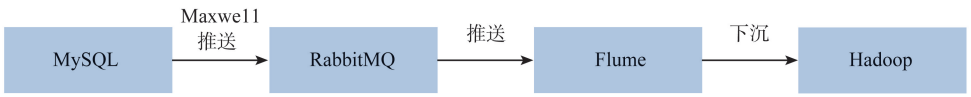


图 6 Flume 采集 RabbitMQ 数据推送至 Hadoop 中实现流程

5.2.3 推荐服务实现。推荐服务实现是从 Hadoop 中读取数据并进行数据清洗,即删除无效信息,并将期望数据信息存储至本地文件。推荐服务模块采用 LFM 算法,通过奇异值矩阵分解(Singular Value Decomposition,SVD)[26]处理评分矩阵,基于用户的潜在特征评价缺失项目。系统通过定时任务每日执行一次,执行过程中推荐程序读取本地文件信息,LFM 算法设置迭代次数为 50,通过迭代后的数据,系统为每个用户推荐 4 条数据,并实例化到数据库中。推荐服务流程如图 7 所示,实现逻辑如图 8 所示。

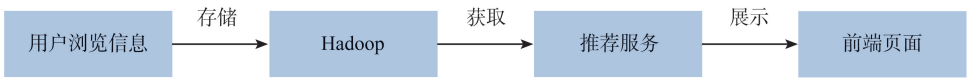


图 7 推荐服务流程

5.2.4 用户购买商品实现。当用户购买商品时,由商品服务先对用户购买的商品加分布式锁,并核验库存信息。当库存容量足够时,商品服务将购买数据发送至 RabbitMQ 中并使库存减 1,消息队列获取数据后等待订单服务消费,消费完成后将数据存入数据库中,对应流程如图 9 所示。

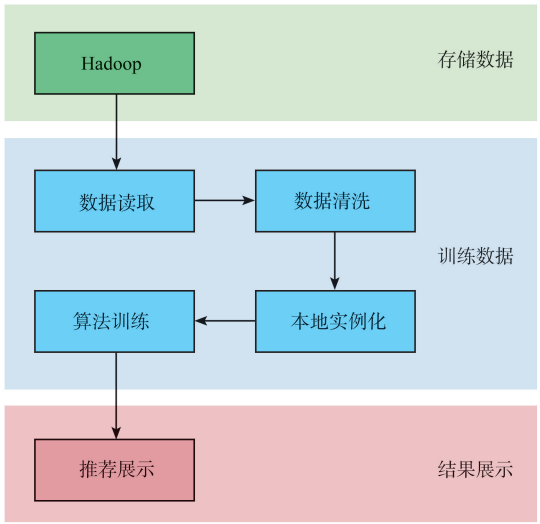


图 8　推荐服务实现逻辑

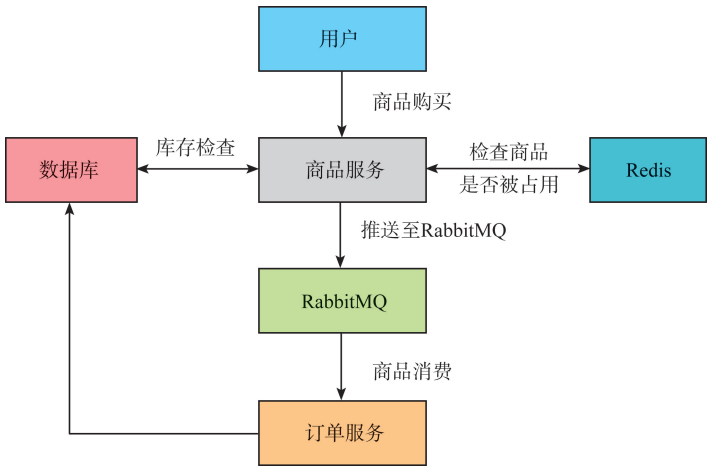


图 9　用户购买商品流程

5.2.5　服务上传部署。首先，后台管理端获取并读取用户上传的文件信息，基于文件名创建对应文件夹，文件路径为“Linux 服务器/opt”，并将用户上传的文件存储至该文件夹下，使用 SSH 协议包连接至 Linux 服务，通过执行对应 Linux 语句运行 jar 包，对应流程如图 10 所示。

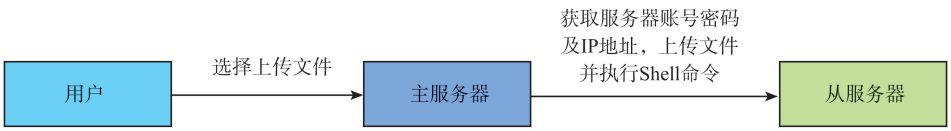


图 10　服务上传部署

## 6　结语

本系统基于 Java 语言、SpringCloud 后端开发框架、Hadoop 分布式架构、Flume 日志收集系统、Maxwell 数据实时同步工具、MySQL 数据库和 Redis 临时数据库，实现了基于 LFM 算法的分布式购物系统。系统采用 LFM 算法，综合考虑用户与历史浏览商品特征间的相似性，实现了用户偏好数据的个性化推荐。为应对高并发情况，采用分布式系统，使用微服务网关进行限流处理，减少了高并发环境下无意义流量对系统的压力。当微服务系统压力过大时，系统主动为响应过慢的服务降级，从而减少访问压力，当该服务响应时间恢复正常时，系统通过提高服务级别的方式降低服务宕机的可能性。此外，系统还使用微服务网关 Zuul 和 Nacos 注册中心实现动态路由代理，自动将流量分发至新增服务器已注册的子系统中，改善了后期系统维护中代码修改困难的问题。在未来的研究工作中，我们将对基于 LFM 算法的分布式购物系统做进一步优化。

## 参　考　文　献

[1]　李雪婷，杨抒，赛亚热·迪力夏提，等. 融合内容与协同过滤的混合推荐算法应用研究[J]. 计算机技术与发展，2021，31(10)：24-29.

[2]　周新鹏，徐凌伟. 5G 移动通信系统性能分析与预测方法研究[J]. 聊城大学学报(自然科学版)，2021，34(4)：23-28.

[3]　张洁辉. MaxWell 测试仿真数据的自动生成[D]. 北京：北京邮电大学，2008.

[4]　吴璨，王小宁，肖海力，等. 分布式消息系统研究综述[J]. 计算机科学，2019，46(S1)：1-5.

[5]　朱涛，孙知信，宫婧. 基于改进的 Flume 的实时数据采集系统[J]. 科技资讯，2021，19(11)：73-75.

[6]　DI M G，TOMARCHIO O. A hierarchical Hadoop framework to process Geo-Distributed big data[J]. Big Data and Cognitive Compu-

ting，2022，6(1)：1-5.

[7] 李浪. 基于微服务网关 Zuul 的 TCP 功能扩展和限流研究[D]. 武汉：武汉理工大学，2019.

[8] 张卓鹏. 基于微服务架构的高可用教务系统的设计与实现[D]. 长春：吉林大学，2021.

[9] 黄勃，严非凡，张昊，等. 推荐系统研究进展与应用[J]. 武汉大学学报(理学版)，2021，67(6)：503-516.

[10] CHEN J，WANG X，ZHAO S，et al. Deep attention user-based collaborative filtering for recommendation[J]. Neurocomputing，2020，383：57-68.

[11] NAAK A，HAGE H，AÏMEUR E. A multi-criteria collaborative filtering approach for research paper recommendation in Papyres[J]. Lecture Notes in Business Information Processing，2009，26：25-39.

[12] KOREN Y，BELl R，VOLINSKY C. Matrix factorization techniques for recommender systems[J]. Computer，2009，42(8)：30-37.

[13] 刘旋. 基于谱聚类和 LFM 的选课推荐算法设计[J]. 现代信息科技，2020，4(1)：14-16.

[14] 程苗. 基于用户行为数据的个性化推荐算法研究[D]. 哈尔滨：哈尔滨理工大学，2021.

[15] 郑鹏，王应明，梁薇. 基于信任和矩阵分解的协同过滤推荐算法[J]. 计算机工程与应用，2018，54(13)：34-40.

[16] KAI Z，LU P Y. Improved collaborative filtering approach based on user similarity combination[C]. // Interna-tional Conference on Management Science & Engineering，IEEE，2014：238-243.

[17] DURICIC T，LACIC E，KOWALD D，et al. Exploiting weak ties in trust-based recommender systems using regular equivalence[J]. arXiv:1907.11620，2019.

[18] 王璐，姜宇轩，李青山，等. 微服务故障检测研究综述[J]. 计算机学报，2023,46(11):2342-2369.

[19] 蔡勋玮，赵俊，赵丽,等. SpringCloud 微服务框架下新型供电系统数据挖掘方法[J]. 电子设计工程，2022，30(16)：164-168.

[20] 王蓉，李晗，周国海，等. 基于 SpringCloud 框架的医疗信息共享平台设计与实现[J]. 中国医学装备，2022，19(5)：133-137.

[21] YANG Y. Design and implementation of online food ordering system based on Springcloud[J]. Information Systems and Economics，2022，3(4)：66-71.

[22] ZHAO X，KANG H，FENG T，et al. A hybrid model based on LFM and BiGRU toward research paper recommendation[J]. IEEE Access，2020，8：188628-188640.

[23] HOFMANN T，PUZICHA J. Latent class models for collaborative filtering[C]. //Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2. 1999.

[24] 史加荣，王丹，尚凡华，等. 随机梯度下降算法研究进展[J]. 自动化学报，2021，47(9)：2103-2119.

[25] 刘俊龙，刘光明，张黛，等. 基于 Redis 的海量互联网小文件实时存储与索引策略研究[J]. 计算机研究与发展，2015，52(S2)：148-154.

[26] ZHOU X，HE J，HUANG G，et al. SVD-based incremental approaches for recommender systems[J]. Journal of Computer and System Sciences，2015，81(4)：717-733.

(责任编辑:刘庆松)