云环境下实时消息推送服务构建研究

曹鹏飞,李 杰,杨 君
(三江学院 计算机科学与工程学院,江苏 南京 210012)

摘 要:随着移动互联网技术的发展,各类消息纵横交错,人们在获取关联信息时常感到无从选择。传统的消息获取,用户需要主动拉取网络存储的消息,此种方式获取消息的效率极其低下,会产生众多与用户无关的消息。消息推送需要用户不停地拉取刷新,属于被动推送。随着云技术的发展,消息推送技术逐步转变为云推送,告别了单一服务器推送的模式。对现有各类消息推送技术进行研究和分析,针对传统基于主题订阅模式的消息推送服务只能对应单一移动端通信的不足,提出了云实验环境下实现实时消息推送服务的构建方案,设计了消息推送服务(PushWeb)的运行框架,运用 H5 技术中的 WebSocket 技术实现了管理平台和移动客户端应用程序,并将其部署在阿里云中,经测试在良好的网络环境下能够将消息实时准确地推送至移动端。

关键词:云环境;云消息;云消息推送;消息推送;实时消息推送;实时消息
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2020)03-0204-05
doi:10.3969/j.issn.1673-629X.2020.03.039

Research on Construction of Real-time Push Service in Cloud Environment

CAO Peng-fei, LI Jie, YANG Jun
(School of Computer Science and Engineering, Sanjiang University, Nanjing 210012, China)

Abstract: With the development of mobile Internet technology, various kinds of notification are intermingled so that people are often at a loss on how to acquire related notification. As for traditional notification acquisition, users need to actively search notification stored online. This way of notification acquisition is extremely inefficient, and a large number of notifications unrelated to users will emerge. Notification push needs users to constantly pull and refresh notifications, so it belongs to passive push. As cloud technology advances, notification push technology has gradually evolved into cloud push and thus making the model of single server push outdated. On the basis of analysis and research on present notification push technology, the shortcoming that the traditional push service based on topic subscription can only work in single mobile communication terminal is found, targeted at which a solution of realizing real-time notification push service in cloud experimental environment is proposed, and the experiment framework of notification push service (PushWeb) is designed. Besides, management of platform and mobile client applications is realized by means of the WebSocket technology in H5, which is deployed in Alibaba Cloud. After inspection and testing, notifications can be correctly pushed to mobile terminals in real-time in fine network environment.

Key words: cloud environment; cloud notification; cloud notification push; notification push; real-time notification push; real-time notification

0 引 言

随着移动互联网技术的发展,各类消息纵横交错,人们在获取关联信息时常感到无从选择。传统的消息获取,用户需要主动拉取网络存储的消息,此种方式获取消息的效率极其低下,会产生众多与用户无关的消息。消息推送需要用户不停的拉取刷新,属于被动推送[1]。随着云技术的发展,消息推送技术逐步转变为云推送,告别了单一服务器推送的模式。

1999 年,网景公司推出了基于 XML 的简易信息聚合(RSS)[2];2005 年,联合格式(Atom)发布[3],可

收稿日期:2019-04-16 **修回日期**:2019-08-19 **网络出版时间**:2019-12-05
基金项目:江苏省教育自然基金项目(17KJD520007);江苏省现代教育技术重点课题(2017-R-59068);三江学院虚拟仿真实验项目(XL2018002)
作者简介:曹鹏飞(1986-),男,硕士,讲师,研究方向为计算机网络技术。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20191205.1113.038.html

以看作是 RSS2.0 的改进方案。

2009 年,Google 提出了 PubSubHubbub[4],将消息获取由“拉取”模式转变为“推送”模式,客户端不再定时轮询服务器,当有更新时及时推送至客户端,成为了互联网分散式消息传播的开放标准,提供了消息即时更新。

2011 年,H5 标准将 WebSocket 协议纳入 Web 交互,在浏览器客户端与服务器之间通过 TCP/IP 技术实现了无刷新的通信[5]。

随着云计算和移动互联网的发展,消息推送技术也逐渐成熟,苹果公司推出了 APNS(Apple Push Notification Service)[6],安卓系统(Android)从 2.2 版本开始增加了云消息推送模块(C2DM)现更名为(Google Cloud Message,GCM)[7],微软在 WinPhone 系统提供了云消息推送服务(Microsoft Push Notification Service,MPNS)[8]。

现有的消息推送大多属于主题式订阅模式,随着 Web 开发技术的发展, WebSocket 作为一种新的技术被应用在消息推送中,有广阔的应用前景,但是针对不同平台终端的推送应用还未有统一标准。

1　实时消息推送服务方案设计

1.1　消息推送服务需求概述

消息推送服务通过云服务器构建消息主体向移动端进行推送,包括:消息推送平台、云服务器、移动客户端(Android)。通过实时消息处理模块将消息发送到移动客户端,客户端接收到消息内容。主要需求如下:

(1)消息推送平台。

提供消息服务接口:采用 Restful API[9] 架构实现即时通信消息推送 API 接口,通过 JSON 进行格式化编码;

消息推送功能:调用 API 接口将消息发送给中间组件,再将消息推送至移动终端;

日志监测功能:将系统运行过程产生的错误记录并保存,监测系统运行状态。

(2)云服务器。

架设在公有云:让用户通过公网 IP 地址直接访问;

存储服务:提供数据的存储和消息的缓存;

云服务器进行相关配置,以实现消息服务的部署。

(3)移动客户端(Android)。

用户管理:提供给管理员用户管理的功能,同时对用户进行权限分配。如:添加、修改、删除用户等,返回操作状态;

业务处理模块:手机终端进行标记,通过算法生成标识符(Token),让手机接收消息时具有唯一标识。手机服务唤醒,一旦检测到消息推送,调用接口程序进行唤醒,提示用户查看消息。实时消息推送,调用消息推送平台提供 API 接口实时推送消息给用户。

1.2　系统结构设计

1.2.1　总体框架设计

系统总体框架由客户端、业务处理、消息推送服务和数据库系统四个模块构成,如图 1 所示。

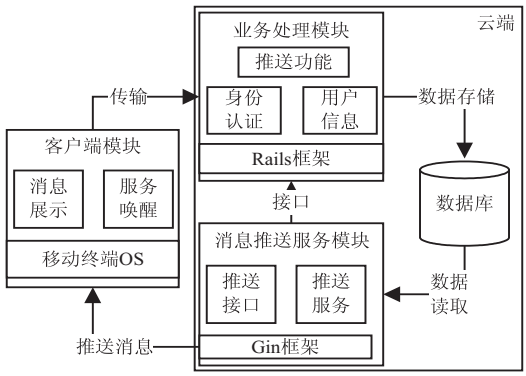

图 1　实时消息推送服务框架

图 1 中显示了系统中四个模块之间的关系,其中:

(1)客户端模块包括消息展示,采用显示展示的方式,并伴随着手机振动和手机铃声;服务唤醒,调用安卓手机系统服务进行唤醒。

(2)业务处理模块包括身份认证、用户管理、推送业务等。身份认证,对所有用户进行分组管理授予不同的权限,区分用户等级;用户信息管理,对用户数据进行增删改查;推送业务,调用消息推送服务 API 接口,将消息主题和内容传递给消息推送服务模块。

(3)消息推送服务模块包括推送接口、推送服务。推送接口,通过构造 HTTP(get、post)请求,将信息发送给推送服务,返回给业务处理模块;推送服务,构建消息传输队列,建立 TCP 长连接,消息推送至移动终端。

1.2.2　数据库系统设计

数据库存储包括用户账户、用户权限、手机注册等数据。数据库各表之间关系如图 2 所示。

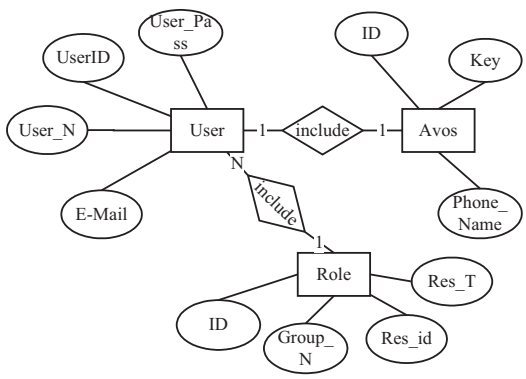

图 2　数据关系

如表 1 所示,User 表存储所有用户的账户数据,包

括:姓名、密码、邮箱、最后 IP、最后登录时间等数据,各表之间通过 Role_id、User_role、User_id 进行关联。

表 1 User 表结构

字段	数据类型	说明
Id	Int	主键,自增编号
User_N	Varchar[20]	用户姓名
E-mail	Varchar[50]	电子邮件
User_pass	Varchar[32]	用户密码(加密)
Create_T	Datetime	创建用户时间
Update_T	Datetime	更新用户时间
Rstpass_token	Varchar[50]	重置 PassToken
Rstpass_T	Datetime	重置 Pass 时间
Create_T	Datetime	账号创建时间
CurrLogin_T	Datetime	当前登录时间
LastLogin_T	Datetime	最后登录时间
Login_count	Int	登录计数
CurrLogin_ip	Varchar[50]	当前登录 IP
LastLogin_ip	Varchar[50]	最后登录 IP

Avos 表,存储用户手机信息,与 User 表外键 User_id 进行关联。User_id 存储用户编号;Key 存储手机序列号,Phone_Name 存储手机名称,Create_T 和 Update_T 是用户创建和更新的时间。

Role 表,存储用户角色,Id 为主键整型自增。Res_id 是资源编号。Group_N 是分组名称,Res_T 是资源类型,与 Group_N 互为唯一索引。Create_T 和 Update_T 是用户创建和更新的时间,不能为空。

2 实时消息推送服务的实现

实时消息推送服务主要由 3 个模块协作运行,包括:管理模块、消息推送服务模块和客户端模块。

管理模块通过框架 Rails 来实现,消息推送服务模块通过框架 Gin 实现,客户端模块采用安卓移动终端,通过 Gin 框架中 WebSocket 组件构造 URL 请求,客户端中请求 URL 地址获取消息的推送。

2.1 管理模块实现

管理模块由用户权限管理、用户信息管理、APP 信息、实时推送四个部分构成。管理模块实现框架如图 3 所示,通过 Ruby on Rails 框架[10]来实现。

其中 Model 模型层处理业务流程,Controller 控制器中有 3 个控制器,分别为:PUSH、App_down 和 User,分别控制推送功能、客户端下载功能、User 资源的操作。View 视图层主要是图形页面展示。

用户权限管理,赋予不同用户不同分组,再将分组赋予不同权限。用户创建时通过 Usermodel 中的 Add_role 事务将用户与用户角色表(Role)关联。

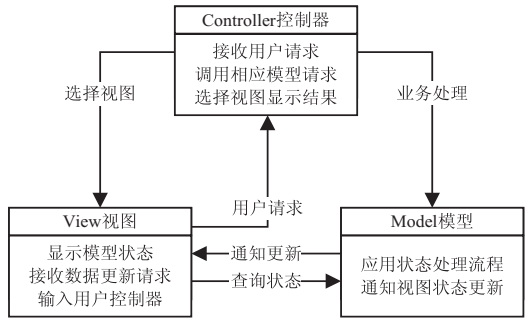

图 3 实现框架

用户信息管理对用户账户信息进行添、删、改、查等操作,通过 JS 代码对用户注册信息进行初步验证,再提交表单与服务器进行校验。

实时推送功能模块使用 HTTP/1.1 协议,采用 POST 方法将信息传递给消息推送服务。使用 RstClient.post 构造 Http_Post 请求,对数据进行格式化处理运用 JSON 输出。如请求失败则通过 Error 获取错误的原因,调用 Deliver 方法将邮件发送给管理员。

前端页面选择响应式浏览方式,让用户可以在多种终端中自动切换,JS 通过 Rails 框架提供的方法将回调信息展示在页面上。

2.2 消息推送服务模块实现

消息推送服务使用 Golang 中 Gin 框架,由协议层、路由分发中间层、数据模型层和业务处理层组成,消息推送服务模块结构如图 4 所示。

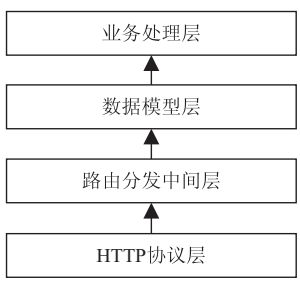

图 4 消息推送服务模块结构

系统底层是 HTTP 协议,通过 Gin 框架 Router 方法路由分发中间层处理,将原本默认端口改为自定义 2333 避免端口被占用情况。在数据模型层中构建数据模型访问 Avos 和 User 表中地数据。业务处理层通过推送接口进行实现。具体实现过程如下:

先建立 TCP 长连接,服务器通过连接通道推送消息。传统的 Socket 连接[11]资源消耗较多,采用心跳机制 MQTT 协议[12]可以降低消耗来维持较长时间的连接。

移动端需要不断地轮询 Publish 端[14]。当有新消息客户端就能监听和获取到消息。消息推送服务实现

过程如图 5 所示。

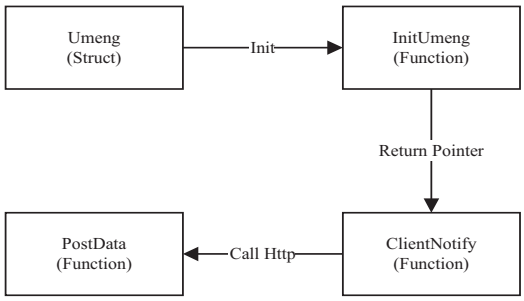

图 5　消息推送服务实现过程

先构建 Struct:Umeng,使用 InitUmeng 方法初始化结构体 Umeng 指针,通过函数指针从 ClientNotify 获取 PostData 数据,发送给消息推送功能 API 接口。

2.3　客户端模块实现

客户端选择安卓(Android)系统作为消息接收端,客户端在后台需要一直监听 Switch 事件,用户在移动端开启推送服务后,系统先对消息服务进行注册,然后调用 SwichPus 方法与手机序列号进行绑定,序列号生成则调用 PushAgent 类中 GetRegistrationId( )方法实现,最后将序列号发送到安卓客户端的消息队列中。

消息展示采用显式的形式,控制消息类将消息推送到移动终端,UmengNotificationClickHandler 构造 LaunchApp 方法来监听消息推送事件,通过 SetNotificationPlaySound( )调用手机系统铃声,并通过 NotifiCationService 类的 onCreate 方法唤醒系统进程。

3　系统运行与测试

3.1　系统运行

3.1.1　云实验环境相关配置

消息推送服务部署在云实验环境下。选用阿里云服务器,通过阿里云直接分配的公网 IP 地址:139.40.62.77。使用 Nginx 为代理服务器[13],将 unicorn 服务使用 8282 端口映射,脚本配置如下:

```
Upstream Back {
Server 139.40.62.77:8282;//云服务器 IP 和端口号}
server{
listen80;//Web 服务转发端口
Root /var/www/push_web;//服务主目录
Location{
proxy_pass http://Back;//地址转发
proxy_set_header X-Real-IP $ remote_ip_addr;//代理 IP 头部设置
proxy_redirect_off;//重定向 url 关闭
proxy_set_header X-Forwarded-For $ remote_ip_addr;
}
}
```

(1)实验云服务器软硬件环境。

服务器软硬件配置:阿里云 ECS 服务器,CPU:2.8G/16 核;内存:8 G;带宽:100 Mbps;操作系统:Centos7 64 位;应用服务:Nginx 1.10.3、MySQL 5.6。

(2)客户端环境。

手机型号:三星 S7;安卓版本:Android 6.0;运行内存:4 G;存储内存:32 GB。

3.1.2　系统运行

云端系统搭建完成后,在手机上安装消息推送服务 APP。服务端发送测试消息,客户端能够收到,系统运行如图 6 所示。

图 6　系统运行

3.2　系统测试

3.2.1　测试指标

系统测试包括功能测试和性能压力测试。

功能测试主要运用黑盒测试[15-16],通过编写系统各功能的测试用例进行测试,验证结果的正确性,包括消息推送功能和用户权限测试。

性能压力测试通过自定义脚本来模拟大规模的消息,测试消息推送的到达率。

3.2.2　主要功能测试

消息推送功能测试用例,用户权限测试用例,如表 2 所示。

表 2　主要功能测试用例

测试内容	消息推送功能测试	用户权限测试
测试目的	验证推送消息是否能成功	验证用户权限是否正确
测试方法	使用安卓手机下载安装 APP 管理员将手机主要信息存储到数据库中 管理员向客移动端推送消息	创建普通用户 打开 URL 输入用户名、密码,点击登录 管理员使用给此用户分配 admin 权限 重新登录,验证是否能登录
预期结果	手机收到消息推送,自动唤醒屏幕	第一次登录提示“用户没有权限” 第二次可以登录系统
结果	通过	通过

3.2.3 性能压力测试

本测试针对系统进行压力测试,计算系统消息推送的到达率,具体测试用例如表 3 所示。

表 3 消息推送到达率测试用例

测试内容	消息推送到达率测试
测试目的	验证消息推送服务的到达率
测试方法	使用安卓手机下载安装 DemoAPP 编写压力测试脚本,运行后批量进行消息推送 统计消息到达情况,计算到达率
预期结果	每千次消息推送到达率大于 90% 每次推送消息响应时间小于 50 ms
结果	在每千次消息推送中实际到达为 968 次 平均响应时间为 33.9 ms

测试用例脚本如下:

```
For (i=0;i<1000;i++){
RstC.post 'http://139.40.62.77:81/push_phone',{'msg'->i.to_s,'User_id' -> 2,'titl'-> "msg:#{i.to_s}" }
}//获取系统 Post 接口
```

根据测试数据看出,系统整体性能良好,每千次消息推送中到达率为 96.8%,在网络状态良好的情况下延迟较低,平均响应时间为 33.9 ms,能正常运行。

4 结束语

提出了在云实验环境下实现实时消息推送服务的构建方案,对消息推送服务框架进行了设计,运用 WebSocket 技术实现了管理平台和移动客户端应用程序,并将其部署在阿里云实验环境中。经过测试,在良好的网络环境下消息能够实时、准确地推送至移动端。

参考文献:

[1] 倪红军. 基于 Android 平台的消息推送研究与实现[J]. 实验室研究与探索,2014,33(5):96-100.

[2] COLD S J. Using really simple syndication (RSS) to enhance student research[J]. ACM SIGITE Newsletter,2006,3(1):6-9.

[3] SAYRE R. Atom:the standard in syndication[J]. IEEE Internet Computing,2005,9(4):71-78.

[4] WEIS T,WACKER A. Federating websites with the Google wave protocol[J]. IEEE Internet Computing,2011,15(3):51-58.

[5] WANG Z,DENG H,HU L,et al. HTML5 web worker transparent offloading method for web applications[C]//2018 IEEE 18th international conference on communication technology (ICCT). [s.l.]:IEEE,2018:1319-1323.

[6] WARREN I,MEADS A,SRIRAMA S,et al. Push notification mechanisms for pervasive smartphone applications[J]. IEEE Pervasive Computing,2014,13(2):61-71.

[7] YILMAZ Y S,AYDIN B I,DEMIRBAS M. Google cloud messaging (GCM):an evaluation[C]//2014 IEEE global communications conference. Austin,TX:IEEE,2014:2807-2812.

[8] BELL K M,BLEAU D N,DAVEY J T. Push notification service:U.S.,8064896[P]. 2011-11-22.

[9] GOSSETT E,TOHER C,OSES C,et al. AFLOW-ML:a RESTful API for machine-learning predictions of materials properties[J]. Computational Materials Science,2018,152:134-145.

[10] BÄCHLE M,KIRCHBERG P. Ruby on rails[J]. IEEE Software,2007,24(6):105-108.

[11] 曹文彬,谭新明,刘 备,等. 基于事件驱动的高性能 WebSocket 服务器的设计与实现[J]. 计算机应用与软件,2018,35(1):21-27.

[12] MANSO M,JOHNSEN F T,LUND K,et al. Using MQTT to support mobile tactical force situational awareness[C]//2018 international conference on military communications and information systems (ICMCIS). Warsaw:IEEE,2018:1-6.

[13] 张 云,许江淳,李玉惠,等. 基于 Nginx 服务器负载均衡技术的研究与改进[J]. 软件,2017,38(8):6-12.

[14] 徐 丹,艾文凯. 高性能时间序列数据库网络中间件研究[J]. 计算机系统应用,2018,27(12):262-267.

[15] 潘文婵,李 超. 基于 UFT 的软件测试实验教学研究[J]. 实验室科学,2017,20(5):92-94.

[16] 王 强,田 涛,刘昕昀. 软件测试能力评定模型研究[J]. 计算机技术与发展,2018, 28(8):75-79