分类号：　　密级：

兰州大学

研 究 生 学 位 论 文

论文题目（中文）	基于微信公众平台的可扩展架构研究与实现
论文题目（外文）	**The Research and Implementation of Extensible Architecture Based on WeChat Public Platform**
研 究 生 姓 名	吴挺
学 科 、 专 业	计算机科学与技术·计算机系统结构
研 究 方 向	分布式系统
学 位 级 别	硕　　士
导师姓名、职称	周庆国　教授
论 文 工 作 起 止 年 月	2014 年 09 月至 2016 年 05 月
论 文 提 交 日 期	2016 年 5 月
论 文 答 辩 日 期	2016 年 5 月
学 位 授 予 日 期	

校址：甘肃省兰州市

原 创 性 声 明

本人郑重声明：本人所呈交的学位论文，是在导师的指导下独立进行研究所取得的成果。学位论文中凡引用他人已经发表或未发表的成果、数据、观点等，均已明确注明出处。除文中已经注明引用的内容外，不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究成果做出重要贡献的个人和集体，均已在文中以明确方式标明。

本声明的法律责任由本人承担。

论文作者签名：_____ 日 期：_____

关于学位论文使用授权的声明

本人在导师指导下所完成的论文及相关的职务作品，知识产权归属兰州大学。本人完全了解兰州大学有关保存、使用学位论文的规定，同意学校保存或向国家有关部门或机构送交论文的纸质版和电子版，允许论文被查阅和借阅；本人授权兰州大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本学位论文。本人离校后发表、使用学位论文或与该论文直接相关的学术论文或成果时，第一署名单位仍然为兰州大学。

本学位论文研究内容：

□可以公开

□不宜公开，已在学位办公室办理保密申请，解密后适用本授权书。

（请在以上选项内选择其中一项打“√”）

论文作者签名：_____ 导师签名：_____

日 期：_____ 日 期：_____

基于微信公众平台的可扩展架构研究与实现

中文摘要

随着移动互联网的飞速发展，移动应用呈爆发式增长。以微信为代表的优秀应用在生活中扮演着重要的角色。微信推出的公众号为个人、组织、企业提供了创造财富的渠道。通信运营商对于微信公众平台有着独特的优势，全面线下渠道使其公众号推广迅速而广泛。如何更加快速地吸引用户关注公众号，增加公众号关注量已成为通信运营商的考核指标；针对爆发式的用户访问，如何保证高可用的服务质量是亟需解决的问题。

本课题以通信运营商微信公众平台提供可靠服务为目的，提出了一个基于微信公众平台可扩展架构方案，该方案包括两个方面，通信运营商微信公众平台基础架构和可扩展分布式架构。基础架构从功能角度出发对通信运营商微信公众号进行设计与实现，它是提供服务的最小单元，包括消息处理、业务查询办理、活动配置、后台管理系统、日志系统；基础架构采用 LNMP 框架，通过 Redis 内存数据库存储高频访问数据，使用 RabbitMQ 消息队列对消息排队处理。可扩展分布式架构从服务的性能出发对基础架构进行分布式实现，在实现过程中通过 Keepalived 保证前端 Nginx 服务器的可用性，Nginx 根据不同的转发策略把请求转发到多个应用服务器，实时统计 Nginx 的连接数来动态的扩展应用服务器的数量；数据库与应用服务器分离，采用 MySQL Replication 的 dualMaster 技术，并使用 Keepalived 来保证其高可用性；在微信网页授权时，把授权网络请求部署到接口服务器，采用 Session 机制来减少网络请求。

本文使用自动化测试工具对单台服务器和架构的性能进行测试比较；统计分析用户日志优化微信公众号自定义菜单结构和功能，以提高公众号服务质量。

本文实现的基于微信公众平台可扩展架构已经成功应用于某运营商微信公众号中，一直提供着稳定、可用的服务；特别是在开展节日活动爆发式用户访问时，更能体现出可扩展分布式架构的优越性。

关键词：通信运营商，负载均衡，MySQL Replication，Session

The Research and Implementation of Extensible Architecture Based on WeChat Public Platform

Abstract

With the rapid development of mobile Internet, mobile applications have been explosively increased. One of the outstanding applications, such as WeChat, has played an important role in our life. The WeChat public accounts provide a channel for individuals, organizations and enterprises to create wealth. Communication operators have a natural advantage of utilizing WeChat public platform, with their overall of offline channels making the public accounts promoted rapidly and extensively. The increasing amount of users has become a communications operator assessment indicators. It has become an urgent issue that how to ensure the availability of high-performance quality of service with explosive users access.

This thesis proposes an extensible architecture scheme based on WeChat public platform aimed at reliable services of communication operators's WeChat public platforms. The scheme consists of the communications operator WeChat public platform infrastructure and the design and implementation of extensible distributed architecture. The infrastructure, the smallest unit of service, is the design and implementation of the communications operator WeChat public platform from a functional point of view, including message processing, business handing, active configuration, management system, log system, LNMP framework to build services, and Redis database to store high frequency accessing data. The extensible distributed architecture is the distributed implementation for the infrastructure from the point of service performance, with Keepalived to ensure the high availability of Nginx server. According to different forwarding strategies, Nginx load balancing server will forward requests to multiple application servers, and decide the number of application servers by Nginx throughput in real-time. The database and application server are separated. The database is built by dualMaster architecture of MySQL Replication, and Keepalived is used to ensure the high availability. The authorization function is deployed to the network interface service machine, using Session mechanism to reduce network requests.

This thesis tests and compares the single server performance by automated testing tools, and optimizes the structure and function of customized menu by statistical analysis of user log, to

improve the quality of service of communication operator WeChat public platform.

The extensible architecture based on WeChat public platform has been successfully applied by a communication operator. It has been providing the stable and available services. Especially in the condition of explosive user access during festivals, the superiority of extensible architecture can be better reflected.

Key words: Communications operators, load balancing, MySQL Replication, Session

目 录

第一章 绪论

1.1 研究背景

移动互联网是传统互联网与移动通信互相融合的产物[1]。移动互联网通过无线传播技术在空间上扩展了传统互联网，使用户在有数据网络的地方都能访问互联网获取实时信息。据统计分析[2]，截至 2015 年底，国内移动智能终端使用用户量达 6.20 亿，较 2014 年底增加 6303 万人。随着移动互联网用户量的不断增加，以移动智能终端为平台的应用数量也呈爆发式的增长。其中，以微信为代表的优秀应用在日常生活中扮演重要角色。微信是免费使用的，它具有实时沟通、社交、支付、公众号等功能，通过朋友圈的人脉关系为微商户提供了广阔的销售市场，相比短信应用功能更加丰富、更加智能[3]。截至 2015 年微信月活跃用户数已经超过 6.5 亿[4]。同时，基于微信的各种服务类别的公众号数量已经达到 800 万个，通过微信支付来交易的用户量超过了 4 亿。绝大多数的移动智能终端都安装了微信应用。由此可见，微信已经在人们生活扮演着重要的角色，已经成为不可或缺的通信工具。

微信公众平台是创造财富价值的有效渠道[5]。通信运营商可通过微信公众号快速完成用户信息搜集、市场调研、品牌推广，通过挖掘用户数据进行精准的营销，实现增益创收。为了丰富公众号功能，大多运营商公众号都进行了二次开发。随着运营商微信公众账号推广，用户量成倍增加。据运营统计，当公众号推送群发消息时，200 万的用户在同一时间段收到推送消息，如果消息是关于活动或者促销业务的，有超过 20%的用户在群发消息后半小时内访问服务，那么半小时的用户访问量就是 40 万。如何保证这段时间内服务器正常服务是本课题的一个重点内容。通信运营商不定期开展活动，使得短时间内爆发式的用户量增长，对系统性能要求非常高。

目前，许多公众号通过接入第三方平台来提供服务。对于通信运营商不断变化、新增的功能需求，第三方接入平台无法及时响应，况且第三方接入平台也无法满足运营商公众号高并发访问的需求，其后端服务需要定制开发，后端服务是基于 Web 架构来实现的。但是，通信运营商公众号对架构性能的侧重点不同。由于单台服务器的响应速度越来越慢，即使可以增加服务器配置来提高处理能力，但往往会中断服务，况且单一的提高配置无法满足不断增长的负载和高并发访问需求。本课题针对运营商微信公众号的可靠服务以及高并发访问需求提出了

基于微信公众平台的可扩展架构，该架构从通信运营商需求出发，设计和实现了公众号各个模块的功能；通过可扩展分布式架构的可用、可扩展来保证公众号的服务质量，该架构已经成功应用于某运营商微信公众号中。

1.2 发展现状

1.2.1 微信公众平台

微信公众号是微信的一个功能模块，它像一个功能简单的轻应用嵌入在微信中，同时又与微信朋友圈、微信通信录等其他模块密切相连。微信公众平台为个人提供展示自己的方式，为组织提供多渠道服务的途径，为企业提供创造财富的门路。根据《2015 年微信公众号媒体价值研究报告》[6]分析得出，大约 80%的微信用户关注了公众号，说明微信公众号功能也是用户使用微信重要功能之一。微信公众号与微信分享、微信朋友圈密切相关。从以上报告得出，用户阅读的文章内容 20%来自公众号内，80%文章是通过朋友圈好友分享查看的，朋友圈让公众号的推广整整扩大了 4 倍。

微信公众平台目前提供了订阅号、服务号、企业号三种类别的公众号[7]，三种类别的公众号都包含了基本的消息回复、自定义菜单等功能。订阅号主要是简单的发送消息，达到宣传效果，部分接口开放；服务号所有的高级接口均开放，相比订阅号享有更多的高级接口，特有的网页授权、微信支付、周边以及设备等功能对微商户有很好的支持；企业号用于企业内部使用，可用来管理企业员工、团队，企业号的每一个应用对应一个服务号。订阅号和服务号的某些高级接口需要用户缴费审核通过以后才可以使用。微信公众平台提供的 3 类服务：对话服务、功能服务、网页服务。对话服务是消息的输入输出，提供多种样式的消息，并且提供了一些智能接口，如语音识别接口、长链接转短链接接口等；功能服务包括微信支付、小店、卡包、设备等，主要是提供一套微商电子渠道解决方案，结合线上线下创造更多的财富；网页服务提供了一套 JSSDK[8]。JSSDK 是微信公众号的一个能力延展，是将微信原生的一些能力提供给开发者，JSSDK 使开发变得简单高效。JSSDK 的分享接口使开发者能够自定义分享到朋友圈的内容，让用户点击朋友圈里消息时再次回到公众号内，形成了一个闭环系统。

1.2.2 运营商微信公众号发展现状

各大通信运营商都创建的自己的微信公众号，各地的市公司、区县分公司都有申请自己公司的微信公众号。微信公众平台提供的基础功能比较单一，无法满足运营商多样化业务需求，微信公众平台针对有开发能力的用户提供了二次开发

接口。大部分市级以上的运营商类公众号都接入开发者模式来满足自己业务的需求。目前，运营商类公众号主要的应用场景包括下面几个方面：

① 通过引导用户在其微信公众号内绑定手机号码，建立用户公众号 openid 与手机号码的关联，接入运营商业务系统提供业务查询办理服务。通过多渠道自动化方式降低运营商成本，良好的用户体验给运营商带来好的评价。

② 通过订阅号的传播能力（每天一条群发消息）推广新业务和优惠活动。公众号图文消息高阅读量能保证业务的传播，而微信用户主要群体为年轻人，对运营商的新业务、新活动更乐于参与。

③ 承载客服功能。设置常用的关键词回复，提高问题处理效率，同时通过微信公众号平台消息管理功能和多客服在线应答用户。

④ 活动开展。在节假日开发或者接入第三方平台的活动，以小额礼品成本实现公众号推广和传播。

⑤ 增值业务。除了基本的查询和办理业务外，通信运营商往往提供了其他的增值业务，如天气查询、找师傅、高考信息查询等等。可以通过公众号提供给用户，甚至异网用户。

可见，目前的微信公众号已经成为运营商的另一种重要的服务渠道，各运营商都创建了自己的公众号，并且市以上的运营商分公司公众号不止一个，大多分公司同时申请了多个订阅号、服务号。订阅号易于传达资讯，用来群发消息、推广新业务新活动，作为宣传渠道；服务号用来提供服务、缴费、开展活动，作为功能主体。由于服务号的接口权限更多，功能更丰富，通知消息明显，运营商更偏向于服务号；对于用户使用来说，也更趋向于服务号。通信运营商的微信公众号的运营方式与传统运营方式不同，需要运营者有互联网的思维，并且需要实时跟踪微信公众平台新功能、新接口。所以，在通信运营商自有技术人员无法支撑的前提下，有必要引入第三方技术力量，保证微信公众号的服务能力[9]。

1.2.3 微信公众号第三方平台发展现状

为了满足通信运营商公众号多方面的功能需求，公众号需要接入开发者模式进行二次开发。目前，有很多免费的第三方接入平台，如微盟、阁下、点点客等，实现了一些通用的功能。由于通信运营商需求不断新增和其业务系统的特殊性，第三方接入平台不能及时满足其需求，其后端服务需要定制开发，后端服务是基于 Web 架构来实现的，各种 Web 架构的实现侧重点不同。如文献[10]从网络系统出发，以分布式系统的分区、缓存、代理、负载均衡、队列等多方面对可扩展分布式 Web 服务架构进行了介绍；如文献[11]从 Web 缓存的一致性进行讨论。由于通信运营商对性能和服务质量要求高，以及微信公众平台提供的接口的特殊性，

如授权过程需要多次网络请求，因此以满足通信运营商公众号高并发、服务质量、微信公众平台接口等需求为目标，提出基于微信公众平台的可扩展架构，其主要侧重点在于负载均衡服务器高可用、应用服务器扩展、微信授权机制下用户会话状态保存以及数据库的可用性。

后端服务架构通常分为 RPC(Remote Procedure Call Protocol)和 REST(Representational State Transfer)[12]，它们都是把一个个函数封装成接口以供调用。RPC 即远程过程调用，可以通过 HTTP、SOCKET、操作系统自带的管道通信技术来对远程的 API 调用。REST 是 Web Service 的两种实现方式之一，另一种是 Soap。本课题是基于 REST 架构，微信服务器与开发者服务器通过 HTTP 协议通信，开发者服务器与通信运营商业务系统通过 Web Service 的 Soap 来进行通信。

1.3 研究内容及意义

通信运营商如何规划微信公众号自定义菜单结构；如何保证后台服务器在活动开展时爆发式用户访问量情况下的高可用、高性能。本文以此为动机，以某通信运营商微信公众号为例，提出了基于微信公众平台可扩展架构方案，通过运营商微信公众号接入开发者模式，结合通信运营商自身的业务进行二次开发。该方案包括三方面的研究内容，通信运营商微信公众平台基础架构的设计与实现、可扩展分布式架构的设计与实现、架构性能测试与日志分析。

① 通信运营商微信公众平台基础架构的设计与实现

运营商微信公众平台基础架构是提供服务的最小单元，包括消息处理、业务查询办理、活动配置、后台管理系统、客服系统。消息处理模块的主要功能是对微信转发过来的用户请求消息进行解密、解析，并且对返回数据进行封装、加密。业务查询办理模块的主要功能是根据用户请求调用运营商业务系统进行查询办理。活动配置模块主要是用于公众号推广的，分为简单的可配置活动和特定开发类活动，特定开发类活动如“流量红包”、“双十一手机杀价”等。后台管理系统的主要功能包括智能应答消息配置、日志查询、模板消息发送等。在基础架构中采用 LNMP（Linux+Nginx+MySQL+Python）框架搭建后端服务，并通过 Redis 数据库存储高频访问数据，使用 RabbitMQ 来对耗时业务进行排队处理。

② 可扩展分布式架构的设计与实现

可扩展分布式架构是对基础架构的分布式实现，通过 Keepalived 来防止前端 Nginx 服务器单点故障；通过前置 Nginx 负载均衡服务器按照预定的转发策略把请求转发到多个应用服务器；通过对 Nginx 连接数的实时监控，动态的扩展应用服务器的数量。数据库采用 MySQL Replication 的 dualMaster 技术，两台数据库

都是主数据库，同时也是从数据库，在任何一个数据库提交了数据更新操作时，另一个数据库会复制该操作的二进制文件，对其执行相同的操作，以保证数据的一致性[13]。两台数据库都可以做读写操作，从而增加了并发连接数，节约了硬件资源。应用服务侧重于业务逻辑处理。构建高可用的微信页面授权机制，把需要授权网络请求从应用服务器分离出来部署到接口服务机，并且采用 Session 机制来减少网络请求。

③ 架构性能测试与日志分析

采用自动化测试工具对架构的可用性、可扩展性进行测试，分析单节点的服务与分布式服务的性能。通过统计用户日志来优化微信公众号菜单的结构，提高公众号的服务器质量。

本文提出的基于微信公众平台可扩展架构针对爆发式活动开展有一定的可用性，能够满足运营商提出的各种功能需求。提高了用户的体验度和参与性，使得运营商运营公众号简单快捷，同时节约了运营商开发成本，为企业创造了更多的价值和财富。

1.4 论文组织结构

本课题是基于微信公众平台可扩展架构的研究和实现，各章节内容如下：

第一章，绪论，概述了移动用户增长趋势；说明了通信运营商建设微信公众平台的必要性；对通信运营商公众号发展现状进行了分析说明；最后对本文的研究内容及意义进行了详细说明。

第二章，相关技术分析，对架构的开发环境、平台、涉及到的相关理论技术进行简要的介绍，主要包括对公众平台开发者模式、消息队列、版本管理技术；最后对负载均衡技术进行了详细的概述。

第三章，基于微信公众平台可扩展架构需求分析，分别从功能和性能两方面对运营商微信公众平台的需求进行详细说明。

第四章，基于微信公众平台可扩展架构设计与实现，对基础架构中的功能、技术点进行设计与实现；对分布式架构中存在的问题进讨论、设计、实现。

第五章，性能测试与分析，对架构的可用性、可扩展性进行测试。分别测试比较了架构和单台服务器的性能；最后对系统和用户日志进行分析。

第六章，总结与展望。

第二章 相关技术分析

本文主要研究基于通信运营商微信公众平台可扩展架构，给出了相应的实现方案。该章节对架构的开发环境以及相关技术进行介绍，主要包括微信公众平台开发者模式、消息队列、版本管理、负载均衡。

2.1 开发环境介绍

基于通信运营商微信公众平台可扩展架构的开发环境采用 LNMP 架构，是比较常见的 Web 服务器端开发环境。Linux 系统凭借其自由软件、支持多平台、多任务多用户、网络功能强大、安全稳定等多方面的优点在近几年得到了非常迅猛的发展。在架构中选用 Red Hat Enterprise Linux Server release 6.4 (Santiago)版本[14]作为服务器系统，RHEL(Red Hat Enterprise Linux)是 Red Hat 公司的 Linux 发行版，该发行版本面向商业市场，大约 3 年发布一个新版本，目前最新版本为 REHL7.2。RHEL6.4 是一个面向服务器操作系统。

2.1.1 Nginx 简介

Nginx 是一款开源的 HTTP 服务器，它具有性能高、稳定性强、功能丰富等特点[15]，同时也提供商业技术支持。Nginx 是解决 C10K[16]问题的少数服务器之一，Nginx 不同于传统的服务器依靠线程来处理请求，它使用一个高可用的事件驱动架构，这种架构在负载下使用小的、可预测的内存。Nginx 对静态文件处理得非常好，能够自动索引文件。Nginx 支持动态加载，不需要重启服务即可加载新的配置。Nginx 提供了基于内容的负载均衡转发功能，该功能包括轮询、权重、ip_hash 等 5 种转发策略。通常 Nginx 负责前端转发和静态资源处理，而把动态请求交给后端的应用服务器，在逻辑处理完后返回结果。表 2-1 是 Nginx 与 Apache、Lighttpd Web 服务器的功能特点的比较。

由表 2.1 可知，在传统的小型网站中采用 Apache 这类传统 Web 服务器似乎能够正常运行，但在处理流量爆发的时候很容易出现过载，这种情况下 Nginx 最为合适。在实际应用中，Web 服务器软件的选择并不是固定或者单一的，往往是根据不同功能块的需求，同时使用多种 Web 服务器。从下面 Web 服务器比较来看，Nginx 有许多突出的优点，随着 Nginx 的功能不断丰富，其占有比重不断的增大，很可能成为主流的 Web 服务器。

表 2-1　三种 Web 服务器的比较[17]

Server	Nginx	Apache	Lighttpd
代理	非常好	非常好	一般
Fcgi	好	不好	非常好
Rewriter	非常好	好	一般
热部署	支持	不支持	不支持
稳定性	非常好	好	不好
安全性	一般	好	一般
反向代理	非常好	一般	一般
系统压力	很小	很大	比较小
Session sticky	不支持	支持	不支持
静态文件处理	非常好	一般	好

2.1.2 uWSGI 简介

在动态 Web 开发中，Web Server 处理静态资源请求，然后将动态资源请求通过内置模块和相关的网关协议转发到应用程序容器中处理。常见的网关协议有 CGI（Common Gateway Interface）、SCGI（Simple Common Gateway Interface），FastCGI，WSGI（Python Web Server Gateway Interface），PSGI（Perl Web Server Gateway Interface），rack，WSAPI（Lua Web Server API），JWSGI（Java Web Server Gateway Interface）等。uWSGI 是一个应用服务器容器[18]（用于各种编程语言和协议），其目的是为建设托管服务开发一个完整的堆栈，它实现了 WSGI 协议、uWSGI、HTTP 等协议。在 Python Web 服务搭建中，uWSGI 是一款非常优秀的中间服务器，它具有占用内存小、性能高等特点。在 Nginx 中集成了 uWSGI 模块，使得 Nginx 与 uWSGI 进程可以直接进行通信，Nginx 中 HttpUwsgiModule 模块的作用就是与 uWSGI 服务器交换请求的。uWSGI 与 Nginx 有多种通信方式，常用的通信方式包括 HTTP、Socket，Nginx 和 uWSGI 两端的通信方式需要保持一致，在并发量高的情况下，Socket 的通信方式性能更优。

2.1.3 MySQL 简介

在移动互联网快速发展时代下，每天有大量的用户数据产生，当数据累计到一定程度可以挖掘分析其价值。在架构中使用 MySQL 数据库来存放用户数据。MySQL 是一种小型的、开源的关系型数据库管理系统[19]，它具有速度快、体积小、成本低等特点。在架构中使用 MySQL Replication 技术实现高可用、可扩展数据库存储系统，MySQL Replication 是 MySQL 推出的数据库高可用方案，它的原理是通过复制数据库二进制日志更新操作来同步数据库，它是 MySQL 数据库实现高可用和扩展性的一种简单方案，可用性能够达到 99.99%。

2.2 微信公众平台开发者模式

微信公众平台原生功能仅包含简单的消息收发功能。为了满足各式各样的用户需求，微信公众平台提供了开发者模式，具有开发能力的用户可以通过接入开发者模式实现各自特有的业务功能。在接入开发者模式时，需要在微信公众号管理平台上配置开发者服务器的地址，用来接收用户的请求。同时需要填写用于消息签名的 Token。当公众号成功接入开发者模式后，微信服务器以 HTTP 协议将签名后的用户请求发送到开发者服务器，开发者获得消息进行业务逻辑处理，将结果封装后返回给微信进而实现与用户交互。用户、微信、开发者服务器之间的消息交互过程如图 2-1 所示。

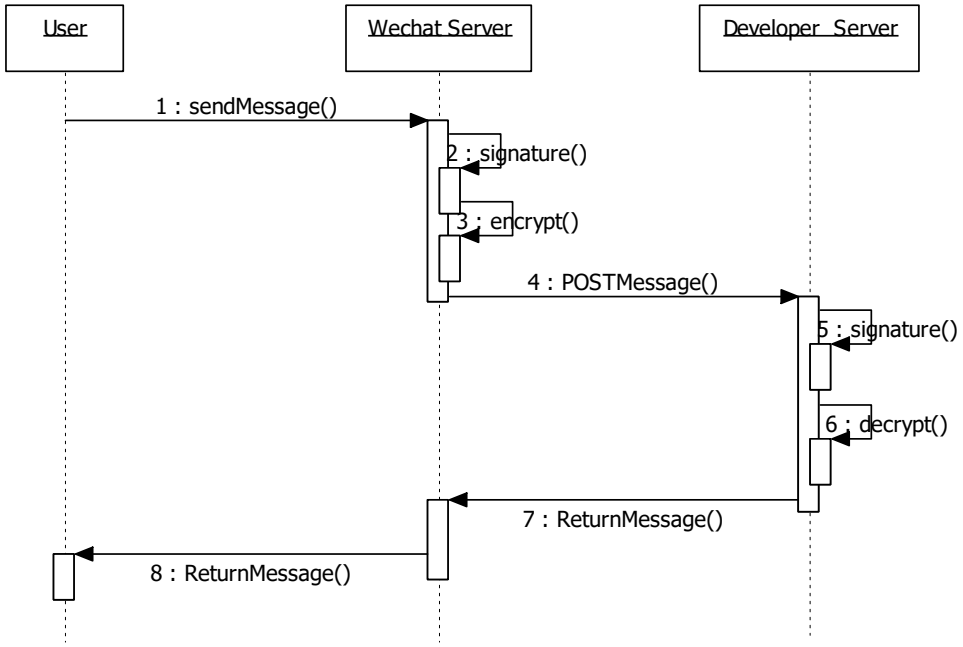

图 2-1 用户、微信服务器、开发者服务器交互图

如图 2-1 所示，用户、微信服务器、开发者服务器交互过程如下：

① 用户向微信公众号发送一个请求，请求发送到微信服务器。

② 微信服务器收到用户的请求后，首先对请求进行签名（根据开发者在微信公众号管理平台上填写的 Token 以及系统的 Timestamp、Nonce 使用 SHA1 进行签名），再根据微信公众平台的安全模式配置决定是否对消息加密。

③ 微信服务器把加工过的用户请求消息 POST 到开发者服务器。

④ 开发者服务器收到请求后，首先对消息进行签名认证（用 SHA1 算法对发送过来的 Timestamp、Nonce 以及自己的 Token 签名得到的 signature 与 URL 地址中的 signature 参数值进行比较，一致则表示消息来源于微信服务器。然后

将消息解密，把 XML 形式的数据解析为易于处理的字典格式。

⑤ 对处理完的请求返回结果，对结果进行封装，如果选择了安全模式，需要对消息进行加密后返回。

2.3 消息队列

由于微信服务器 POST 到开发者服务器的请求响应时长限制在 5 秒内，如果开发者超过 5 秒没有响应，重发机制使得微信服务器重新转发请求，重复 3 次都没有返回，公众号将给出暂时无法提供服务提示，这会导致多种复杂的问题，第一，服务器接收到多次相同的请求，在业务办理时，会导致错误办理；第二，访问量较高的情况下，重试会增加服务器的压力，影响性能；第三，响应时间长，用户长时间收不到返回消息，或者收到多条重复消息，对用户体验造成极大的影响。由于存在这些问题，对于耗时办理类业务，先给用户返回处理提示，然后把请求缓存到消息队列，再按序调用通信运营商业务系统，在完成后使用异步消息给用户展示处理结果。

消息队列是一个常用的组件，它使用了 C/S（Client/Server）结构，提供一种将数据从一个进程到另一个进程传递的方法。在架构中使用 RabbitMQ 消息队列把暂时来不及处理的耗时办理业务请求缓存到队列中，然后从队列中读出请求进行处理。RabbitMQ 是 AMQP（Advanced Message Queuing Protocol）的一个实现，开源的，服务器端用 Erlang 语言编写，支持多种语言的客户端，如：Python、Java、PHP 等。RabbitMQ 有 5 种工作方式：Work queues、Publish/Subscribe、Routing、Topics、RPC。在架构中采用 Work queues[20]工作方式，其原理如图 2-2 所示。

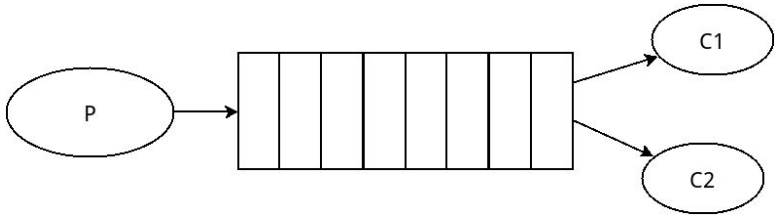

图 2-2　RabbitMQ 的 work queues 工作方式

如图 2-2 中所示，RabbitMQ 作为简单的消息队列使用，多个生产者把系统来不及处理的消息缓存到消息队列排队等待处理，多个消费者可以同时从队列中取出消息执行业务操作。RabbitMQ 在这种模式下起着缓存消息、分发消息的作用。在实际应用中，可能会发生消费者收到 Queue 中的消息，但没有处理完成就中断或出现其他意外的情况，这种情况下可能会导致消息丢失。为了避免这种情况发生，RabbitMQ acknowledgment 选项要求消费者在处理完消息后发送一个回

执（Message acknowledgment）给 RabbitMQ，RabbitMQ 收到消息回执后才将该消息从 Queue 中移除；RabbitMQ 提供了多种客户端语言实现方式，Python 中使用的 RabbitMQ 消息队列库为 pika。

2.4 版本管理

在架构中使用 Git[21]作为版本管理工具。Git 在远端服务器和本地都保存了代码主分支，开发人员需要先将修改提交到本地分支，然后再把代码提交到远程分支。Git 主要的原理[22]如图 2-3 所示，主要分为三个部分，工作区（Working directory）、本地版本库(Local repository)、远程版本库（Remote repostitory。使用 Git 管理时，数据流一般从工作区 Commit 到本地版本库，然后再上传到远程版本库。在本地版本库可以创建多个分支，默认创建了一个本地 Master 分支。在没有创建其他分支时，可以认为本地 Master 分支就是本地版本库。

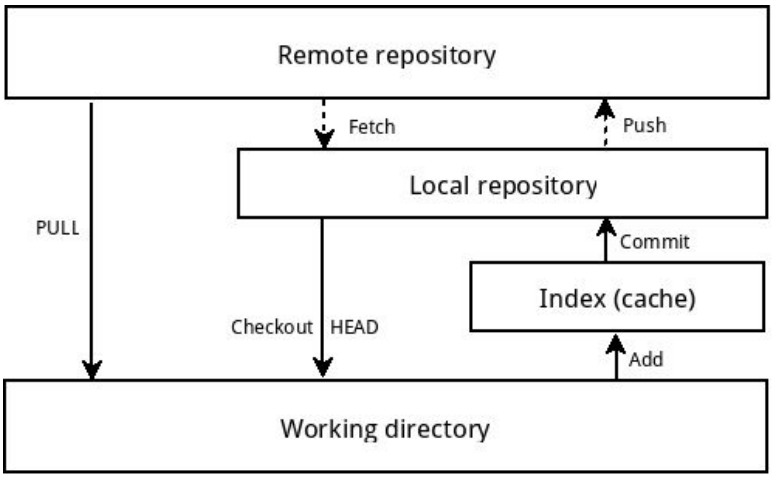

图 2-3　Git Data Flow And Commands

为了在多人协作开发时有效地进行版本管理。开发人员新增、修改代码时，首先在本地新建一个 dev 分支，在 dev 分支上开发和修改，测试通过以后，在 dev 分支上提交更新内容。然后用 local master 合并 dev 分支，再 push 到远程服务器上。结合图形化工具 gitk[23]，可有效地进行代码管理。

2.5 负载均衡

负载均衡是一种动态均衡技术[24]，把请求按照一定的策略转发给后端的应用服务器和缓存服务器。负载均衡对集群系统的性能起着决定性作用，它的特点有：

① 高可用性（HA）：高可用性是利用集群管理软件，如 Keepalived、Heartbeat 来防止单点故障，当主服务器发生故障，按照备份服务器的优先级来动态地选择一个代替主服务器实现不间断的服务。

② 负载平衡：根据不同层面的实现原理，以预定的算法将请求合理的转发到后端的节点，有效的利用各个节点使系统的整体性能达到最大。

负载均衡集群[25]是实现分布式系统常用的有效方法。负载均衡服务器位于集群的接入层，通过识别请求头中的内容以预设的转发策略把用户请求合理地转发给应用节点；应用节点位于系统的中间层，主要负责业务逻辑处理、数据库调用、其他接口功能接入。通过入口层统计系统当前的并发量来实时地增加、减少应用节点的数量；同时可以在负载均衡节点与应用节点之间增加缓存服务器，以减少应用节点的压力，提高整个系统的性能。按照不同的实现机制，负载均衡技术分为以下 4 类。

① 基于 DNS 解析解决方法

基于 DNS 解析解决方法的原理[26]是域名到服务器 IP 地址一对多的映射关系，在域名解析时根据预定的策略解析到不同的服务器 IP 地址，从而实现负载均衡[27]。基于 DNS 解析的解决方法是最早的负载均衡实现方法，该解决方法存在诸多问题：第一，域名解析是一个多层次请求过程，在解析过程中将域名从本地域名服务器提交到上层域名服务器解析，当解析成功后会缓存域名到 IP 地址的映射，从而导致所有的用户都访问同一服务器，出现负载不平衡。第二，缓存的域名到 IP 地址映射存在一个 TTL（Time To Live）时间，如果超过了 TTL 时间，用户访问时需要重新解析域名，TTL 时长设置不当会严重影响系统的性能和响应时间。

② 基于客户端解决方法

基于客户端解决方法的原理是通过客户端浏览器统计服务器的运行状况，然后把用户请求发送到压力最小、性能最高的服务器[28]。该方法解决了基于 DNS 解析方法中域名解析问题，同时服务器端也不需要任何改动。但是，基于客户端的解决方法存在普及通用性问题，并不是所有的客户端程序都有服务器状况采集功能。在客户端应用程序实时收集服务器状况信息时，会占用网络资源，不具有普遍性。缓存的服务器状态信息可能不是最新的，在客户端统计服务器信息时需要实时访问，会导致一定的延迟。

③ 基于四层调度解决方法

基于四层调度[29]解决方法的实现原理是外部 IP 地址与内部 IP 地址的一对多的映射关系，即负载均衡入口层 IP 地址根据请求中的 IP 地址和端口号采用预定的策略映射到系统内部多个 IP 地址来实现负载均衡。由于基于四层的调度解决

方法比较接近网络底层，通常将该方法集成在硬件设备中，如局域网交换机。基于四层调度解决方法基于硬件，其性能非常优秀，但是成本相对较高。针对硬件的替换方案，可以通过开源 Linux，在系统的核心层实现四层交换功能，在比较稳定的核心空间对连接报文进行转发。

④ 基于高层协议内容交换的解决方法

高层协议内容交换[30]又称为第七层调度，其原理是根据应用层报文的内容以及负载均衡策略来决定最终服务器选择。在实际应用中，对报文分析的长度是有限制的，过长的报文会影响处理速度，吞吐量下降。基于高层协议内容交换解决方法能够克服基于四层调度解决方法的缺点。在一次请求当中，需要访问多种资源，如 JS、CSS、IMAGE 等静态资源以及 JSON、XML、TEXT 动态资源，这两种资源的响应方式差异很大，静态资源可以通过缓存进行加速，动态资源可以进行压缩传输。而基于四层调度解决方法无法将这些区分开来。

2.6 小结

本章对架构开发所需的软件环境、编程框架进行了介绍。对微信公众平台开发者模式消息交互进行了详细的概述，对消息队列、课题中用到的版本管理方法进行了说明。最后，重点介绍了负载均衡技术的特点及类型。

第三章　基于微信公众平台的可扩展架构需求分析

本章对基于微信公众平台可扩展架构进行需求分析。通过对用户与系统、管理员与系统的交互场景分析得到系统的关键需求。系统需求分析包括通信运营商微信公众平台基础架构需求分析和可扩展分布式架构需求分析。前者主要是从运营商微信公众号的应用场景出发对系统的功能需求进行分析，包括各个功能模块需求；后者主要是关于可扩展分布式架构性能需求的分析。

3.1 通信运营商微信公众平台基础架构需求分析

运营商微信公众平台基础架构是提供服务的最小单元，它涵盖了运营商公众号的所有功能。根据不同运营商的需求，运营商微信公众平台基础架构可以分为以下功能模块：消息处理、业务逻辑、客服系统、活动配置、后台管理系统、日志系统，如图 3-1 所示。

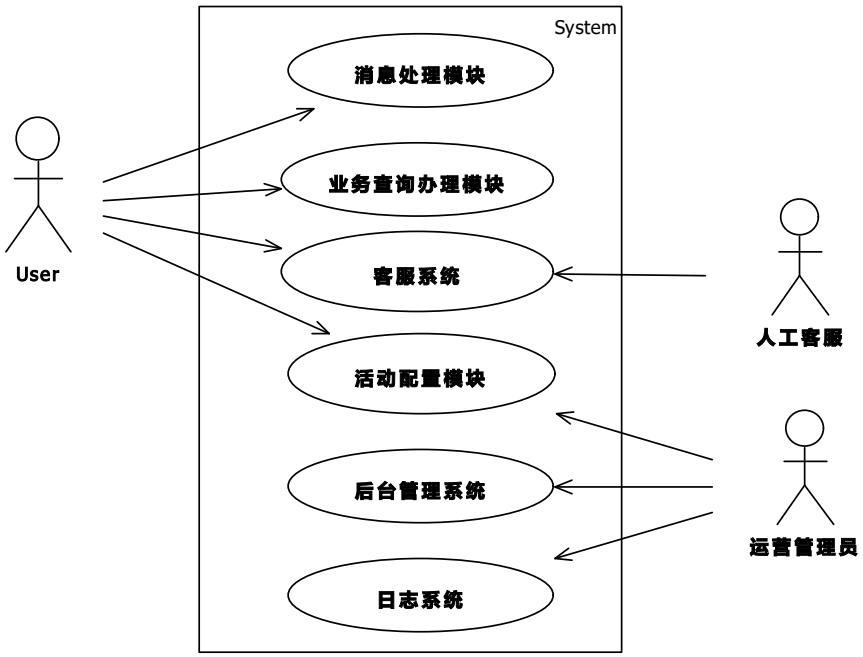

图 3-1　通信运营商微信公众平台基础架构用例图

如图 3-1 所示，与运营商微信公众平台基础架构交互的相关角色包括用户、运营管理员、人工客服，用户向公众号发起服务请求，相反公众号也可以主动给用户推送、群发消息。在运营商微信公众平台基础架构中，消息处理模块主要功

能是对微信转发过来的用户请求消息进行验证、解密、解析，并且对返回给用户的数据进行封装、加密。业务逻辑模处理模块根据用户的请求调用运营商业务系统进行业务办理。客服系统分为智能应答和人工客服两个部分；智能应答是通过关键字智能匹配实现的，当用户输入的内容不是系统保留的功能文本消息或者语音消息时，根据消息库的内容模糊匹配，随机选择一个内容进行回复；在智能应答匹配未命中的情况下，把用户消息转发到人工客服，用户进入聊天语境，人工客服解答完用户的问题后，关闭与用户的连接，用户退出聊天语境。活动配置模块主要用于宣传和推广，分为简单的可配置活动和特定开发类活动，开发类活动包括“双十一手机杀价”、“流量红包”等。后台管理系统是用于运营管理员管理公众号使用的，包括日志查询，消息自动回复配置，模板消息发送等功能。日志系统的功能是记录用户、运营管理员操作记录、系统访问日志、消息日志以及运营商系统调用日志的，对于消息类日志存储在数据库中，并按年分表存储。接口类日志按月分割存储在文件里，把上月前的历史数据压缩存储以节省磁盘空间。

3.1.1 消息处理模块需求分析

消息处理模块需要处理多种用户消息：普通消息、事件消息。普通消息包括文本消息、多媒体消息，多媒体消息包括语音、视频、链接等消息。事件消息包括自定义菜单点击、关注、扫描二维码等。在开发者服务器处理业务逻辑后，返回给用户的消息类型有：文本消息、多媒体消息、图文消息、模板消息。多媒体消息需要调用素材管理接口把多媒体资源上传到微信服务器，获得相应的资源 id，然后以资源 id 的形式封装成 XML 数据返回给微信服务器。模板消息是提供业务办理结果通知的消息。

根据微信公众平台消息接口和通信运营商的功能需求，需要对开发者服务器返回的消息格式进行模板化，消息模板中主要包含发送公众号的 id，接收消息的用户 id，消息创建时间，消息类型、消息的唯一标识等内容。消息模板如表 3-1 所示，下面分别对不同类别的返回消息字段进行分析。

表 3-1　消息模板用例

消息模板字段	描　述
公众号 id	用于标准发送消息的公众号
用户 id	与公众号相关联的用户唯一 id
消息创建时间	消息创建时间，为 Linux 时间戳
消息类型	不同的内容，标志不同的消息类别
消息的标识	消息的唯一 id

① 文本消息

文本消息是公众平台支持的最基本的用户交互方式，当返回消息为文本消息

时，消息模板中的消息类型为 text。消息模板中再增加一个消息内容字段即能够满足微信公众平台接口的需求。文本消息可以用于关键词回复，聊天等场景。

② 多媒体消息

当返回消息为多媒体消息时，消息模板中的消息类型根据不同的多媒体消息类型设置为不同的值，消息类型值可以为图片 image、语音 voice、视频 video、小视频 shortvideo 、位置 location 、链接 link。在多媒体消息中，只包含多媒体资源的 id，其内容是缓存在微信服务器上（存储时间为 3 天），如果需要使用多媒体资源，可调用多媒体文件下载接口下载资源。另外，图片消息中包含有图片的 URL，可以直接使用 URL。当公众号开通了语音识别功能时，开发者服务器接受到的语音消息时会自动带上语音识别结果字段。

③ 事件消息

在微信用户与运营商公众号交互的过程中，用户的事件消息会推送到开发者服务器。事件消息的类型包括点击自定义菜单、关注、扫码等多种事件消息。其中有些事件消息，允许开发者回复用户，如关注事件，部分则不允许，如点击菜单跳转链接的事件推送消息。在事件消息中增加了 EventKey 字段，根据 Event 字段的不同类型，事件消息中 EventKey 字段有不同的含义。当 Event 为 SCAN 时，当前事件为扫描二维码事件，EventKey 为二维码解码后的字符串。当 Event 为 click 时，当前事件为用户点击微信公众号菜单事件，EventKey 为该菜单预设的 Value 值，开发者可以根据消息中 Value 值进行相应的业务处理。当 Event 为 view 时，当前事件为菜单的跳转事件，用户界面由微信公众号跳转到 HTML 页面，在这种事件类型中，开发者不能对用户进行回复。

④ 图文消息

图文消息的 MsgType 为 news，其内容可以为一条或者多条图文，ArticleCount 为图文条数参数，限制为 10 条以内，每条图文内容的字段包括：标题、详细描述、图片地址和图文消息跳转 URL 地址。当图文内容条数为 1 时，整个消息显示一条图文，当图文内容大于 1 时，从第二条图文内容开始不显示详细描述。

⑤ 模板消息

模板消息属于异步通知消息，在用户未主动发起请求的情况下，开发者服务器可以调用微信公众号模板消息下发接口主动给用户发送模板消息。模板消息用于微信公众号向用户发送重要通知。模板消息只能在特定的服务场景中使用，如费用不足、流量不足、活动即将开始等业务。同时，开发者只能编辑模板消息中的部分内容，如果在模板消息库中没有查询符合业务场景的模板，可以向微信公众平台申请新的模板。在发送模板消息时，需要用户的 openid 信息，模板消息各字段如表 3-2 所示。

表 3-2　模板消息用例

参　数	描　述
touser	接收方帐号（收到的 openid）
template_id	模板消息 Id
url	模板消息的连接
data	模板消息内容
ACCESS_TOKEN	开发者凭证

其中，data 的格式如下所示：

{ {first.DATA} }

关键词 1：{ {keyword1.DATA} }

关键词 2：{ {keyword2.DATA} }

关键词 3：{ {keyword3.DATA} }

{ {remark.DATA} }

在上面的格式定义中，关键词的内容不能更改，大括号中的内容可以定义或者缺省。模板消息在通知用户业务办理结果交互时起着重要的作用。

3.1.2 业务查询办理模块需求分析

① 绑定功能

通信运营商标识用户是通过用户手机号码，而在微信公众号里标识用户是通过用户的 openid。微信服务器会为微信公众号关注用户生成一个唯一的 openid 标识（通过用户的账号 id 与微信公众的 id 共同决定），为了通过公众号给用户提供运营商业务服务，需要将用户手机号码与 openid 建立关联关系。

② 查询类功能

查询类业务包括：账单查询、流量查询、套餐查询、积分查询。账单查询返回用户的消费明细，分为日账单、月账单。流量查询包括流量剩余总值、流量使用的明细。套餐查询的主要内容为用户套餐剩余情况，主要包括套餐内短信、语音时长、流量、Wlan。积分查询显示用户当前可兑换积分以及兑换历史记录。

③ 办理类功能

办理类业务包括：套餐办理、数据业务办理、充值缴费。套餐办理，需要支持短彩信、流量套餐、主套餐变更、Wlan 套餐等功能。充值缴费，通过微信支付付款。

3.1.3 客服系统需求分析

智能客服扮演着多种角色，当用户有问题咨询时，可以在公众号里直接发送相应的消息，以满足用户的个性化服务，结合人工客服可以有效的提高客服处理

能力。客服系统分为两个部分：智能应答和客服系统，如图 3-2 所示。

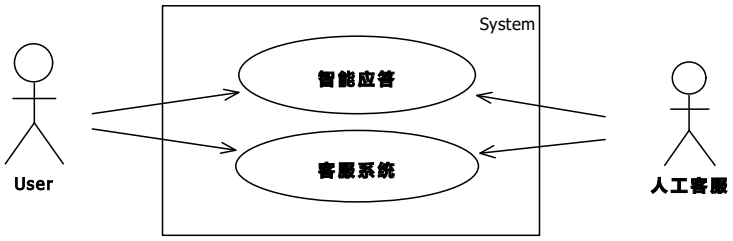

图 3-2　客服系统用例图

在客服系统中，如果用户发送的消息不在系统预设的功能关键字内时，首先通过智能应答系统，如果命中，直接返回结果给用户。如果没有命中，提醒用户点击人工客服进行详情咨询。人工客服要求多个客服可以协同工作，多个客服可以对用户问题进行分流处理，客服系统对客服人员的服务数量以及服务内容记录。客服人员根据人工回答问题的频率统计，把频率最高的问题与其解决方法通过后台管理系统配置智能应答系统中，以减少人工客服的压力。

3.1.4 活动配置模块需求分析

活动包括两种类型，一种是可配置活动，定义一个活动框架，运营人员可以通过后台管理系统的配置生成不同的活动入口 URL，展现出不同活动内容、功能页面，如页面摇一摇、抽奖活动，这种类型活动针对特定用户群。另一种是针对流程复杂的活动，如“双十一手机杀价”、“流量红包”，需要定制开发。本文以“流量红包”活动为例，对运营商活动进行说明，流量红包主要包含四部分的功能，我的账户、发红包、讨红包、拆系统红包，如图 3-3 所示。

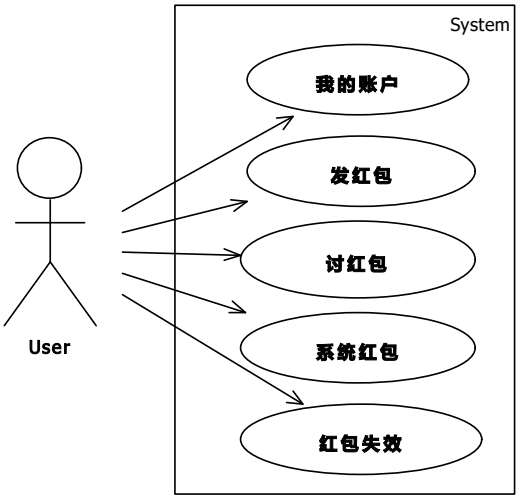

图 3-3　流量红包用例图

系统给每个进入流量红包主页的用户都创建一个虚拟流量账户，在初始状态下，每位用户的账户总余额为零。用户可以发送红包给好友或者到朋友圈，发送的红包可为普通红包或随机流量红包，普通红包是等额流量值红包，随机流量红包要求用户收到的红包流量值是随机的整数值，所有随机流量值之和要与用户发出的总流量值相等。用户也可以通过讨红包的方式邀请好友给自己发流量红包。用户每天可以拆一次系统红包，通过用户的活跃度对用户进行评级，不同等级的用户系统红包额度不同。在虚拟账户中的流量可以兑换充值到用户的手机账户。红包超过 24 小时后自动退回到用户的虚拟流量账户中。

3.1.5 后台管理系统需求分析

后台管理系统主要是给运营管理员、客服人员使用的。其主要功能包括登录、智能应答消息配置、日志查询、模板消息推送、活动配置等，如图 3-4 所示。

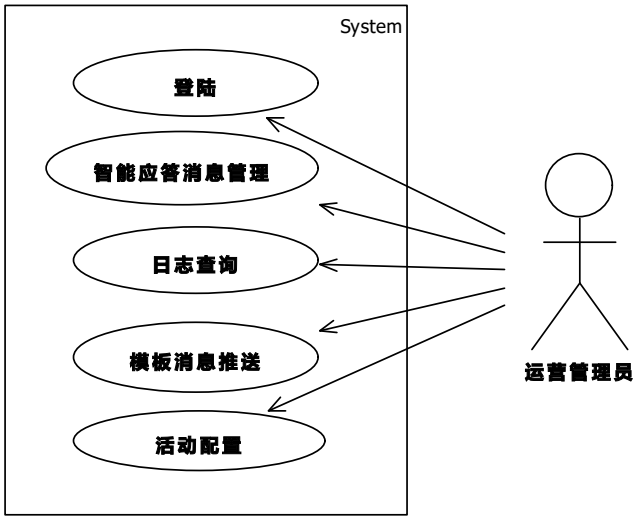

图 3-4　后台管理系统用例图

后台管理系统的用户权限分为普通管理员、超级管理员。普通管理员只能够进行查询、日志下载，只有超级管理员有写权限。智能应答消息管理要求对消息可以进行增、删、改、隐藏等操作。模板消息推送需要管理员先制定模板消息，然后上传需要推送消息的用户号码。在推送时，只有在关联表里存在绑定记录的用户才会收到模板消息通知。在活动配置时，管理员可以上传图片，选择其他功能模块，在活动配置完成以后，可以进行预览以便修改。

3.1.6 日志系统需求分析

日志系统中的日志类型包括用户日志、接口日志、系统访问日志等。每位用

户的任何操作都会记录下来，并保持六个月的日志存储能力。用户操作日志记录在数据库中，以便于查询。用户日志分为消息日志、事件日志。接口日志、系统访问日志保存在文件中，按每月分文件存储，并对上月的文件进行压缩备份。日志可以在后台管理系统或者微信公众平台上下载。

3.2 可扩展分布式架构需求分析

随着用户量的增加，服务器的性能需要不断提升和优化以保证服务质量。当单台服务器无法满足日益增长的用户量和高并发请求时，需要对系统的性能进行扩展。在构建服务前需要对用户量进行估计，长远的计划能够保证服务长久运行。在可扩展架构构建时需遵循以下准则[31]：

① 可用性

可扩展分布式架构的高可用性需要考虑关键组件的冗余，特别是单点故障，达到服务快速恢复的目的，并且在用户无法感知的情况下更换问题组件。一个服务正常的运行时间对于公司的信誉是非常重要的，特别是运营商追求“十分满意”的服务质量。高可用的服务可以增加业务的推广和良好的用户口碑。

② 性能

性能是服务重要的考核指标，在微信服务器与开发者服务器交互过程中要求响应时间不超过 5 秒，当开发者服务器没有及时响应，微信服务器会重新转发用户请求，重试 3 次以后提醒用户该公众号暂时无法提供服务，严重影响了服务器的性能和用户的体验。

③ 可靠性

可靠性是系统在相同的请求始终返回同样的结果。如果用户的数据已经更新，新的请求应该返回新的数据。系统应该对这些数据进行备份，并且提供查询接口给用户检索。

④ 可扩展性

可扩展性[32]可以分为垂直扩展和水平扩展两个方面。垂直扩展是扩大单台应用服务器的硬件配置，如 CPU、内存、带宽。水平扩展是增加服务器的数量。按照系统结构可以分别在前端负载均衡、应用服务器和后端数据库实现扩展。在特定需求下，可以使用多级负载均衡策略来扩展服务器的性能。但是，现在负载均衡服务器一般都可以处理几千上万的并发量。比如，Nginx 可以处理超过 20000 的并发请求。数据库扩展是非常常见的方式，但其会给数据层带来额外的开销并且会增加数据的复杂性。

⑤ 成本

成本包含软硬件成本，以及系统设计、部署、运维费用。

以上需求是设计一个可扩展架构的基本准则，它们可能是互相矛盾的，一般是根据用户的需求选择一个折中的策略。可扩展分布式架构是对基础架构的分布式实现，它的侧重点在于可用性、良好的性能以及可扩展性。

3.3 小结

本章对基于微信公众平台可扩展架构两个部分的需求进行了分析。分析了运营商微信公众平台基础架构的各个子模块需求，包括消息处理模块、业务查询办理、客服系统、活动配置、日志系统，并对可扩展分布式架构的需求进行分析，为后续的设计开发工作提供了基础。

第四章　基于微信公众平台的可扩展架构设计与实现

通信运营商微信公众平台基础架构是可扩展分布式架构的基本单元，通信运营商微信公众平台基础架构是面向业务的，可扩展分布式架构是面向整体性能的。本节按照需求的结构从两个方面对系统进行设计与实现。运营商微信公众平台基础架构的设计与实现包括各个子模块的设计与实现，可扩展分布式架构的设计与实现包括前端负载均衡服务器可用可用性的设计与实现、应用服务器水平扩展的设计与实现、Session 机制的设计与实现、数据库可用性的设计与实现。

4.1 通信运营商微信公众平台基础架构设计与实现

通信运营商微信公众平台基础架构是基于 C/S 与 B/S（Browser/Server）混合架构设计的，用户可以通过两种方式与开发者服务器进行交互，一种是微信客服端里的微信公众号，另一种是微信内置的浏览器，这也是微信公众号与 HTML5 高度融合的优点。服务器端主要负责业务逻辑处理，包括消息处理、业务逻辑，以及与运营商系统、其他 API 接口对接，如图 4-1 中所示。客户端的设计主要包括关键字（多媒体消息）、自定义菜单、HTML5 页面设计。在不考虑性能的情况下，通信运营商微信公众平台基础架构可以在单台服务器上提供服务，基础架构包括了运营商微信公众号的所有功能。

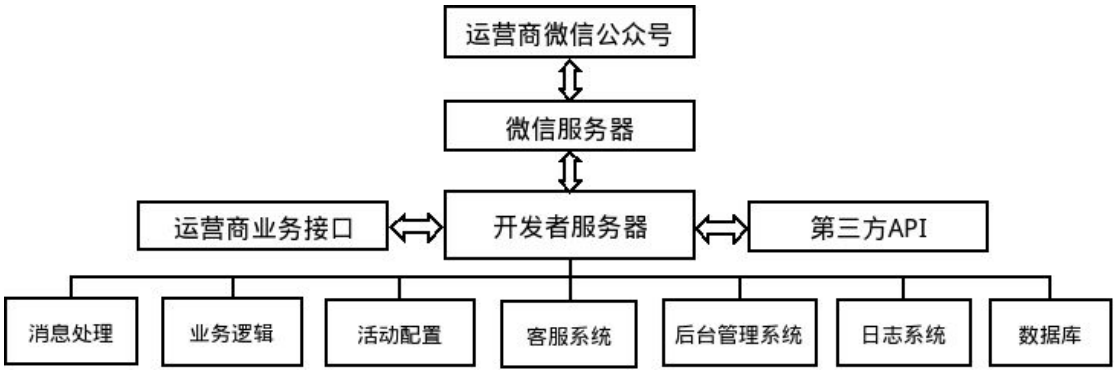

图 4-1　通信运营商微信公众平台基础架构系统架构图

从图 4-1 中可以得出，微信公众号是运营商提供服务的入口，开发者服务器主要接收来自微信服务器转发过来的消息，然后请求运营商业务系统、第三方 API 进行业务逻辑处理，然后把处理完的结果返回给微信服务器，最终返回给用户。由于微信公众号跟 HTML 高度融合，用户也可以直接通过微信内置浏览器

与开发者服务器进行交互。通信运营商微信公众平台基础架构包括消息处理、业务逻辑、活动配置、客服系统、后台管理系统、日志系统、数据库。图 4-2 是通信运营商微信公众平台基础架构逻辑架构图，其描述了该架构各个模块间的消息交互。

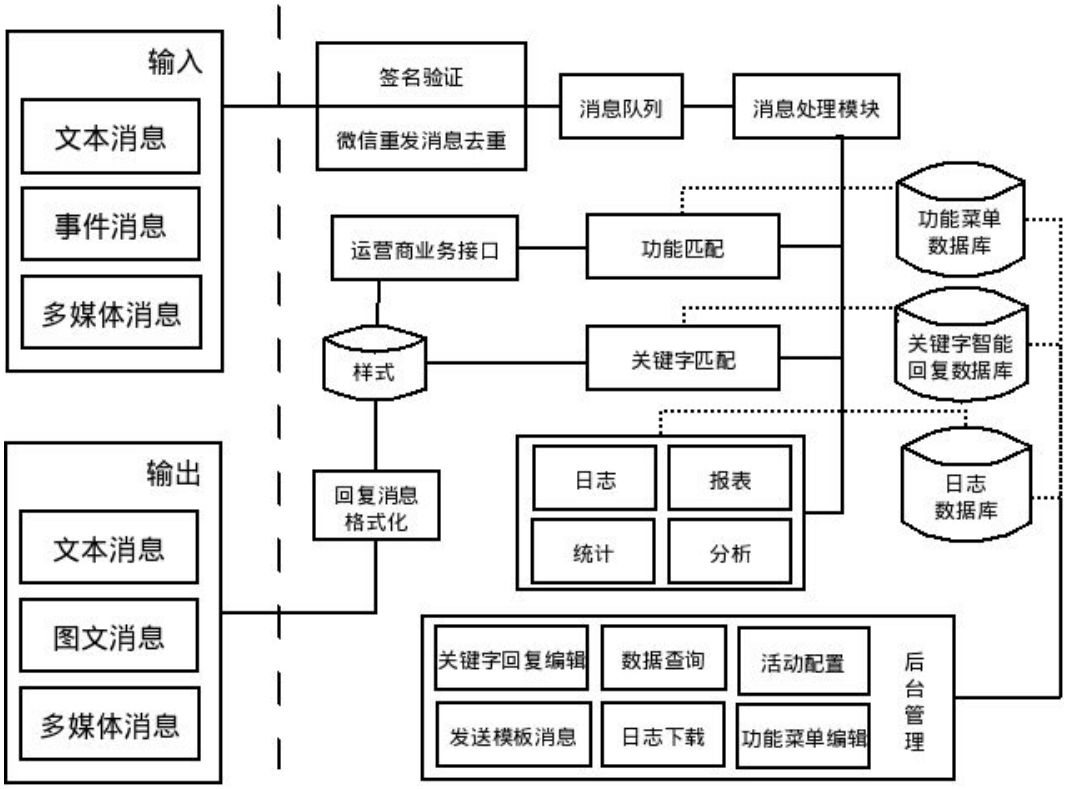

图 4-2　通信运营商微信公众平台基础架构逻辑架构图

在图 4-2 逻辑架构图中，对于不同类型的输入消息，由于其字段参数不一致，不同类别的消息需要分别处理，分为文本、事件、图片、语音、视频 5 大类，并且相同类型的消息不同类别也存在一些差异，比如事件消息中的关注事件与扫描事件的消息在字段的数量上就不相等，因此，相同的消息还需要细分。类似的在返回处理结果时，也需要封装多种消息。在架构中，分别对每种返回消息创建相应的模板类，在每次返回时调用消息模板传入数据结果即可。在逻辑架构中的微信重发消息去重是用于忽略重复消息的，当服务器在 5 秒内没有响应时，微信服务器会重复转发上一条消息，去重机制通过消息标识来识别相同消息。下面分别对各个模块进行设计与实现。

4.1.1 消息处理模块设计与实现

消息处理模块的功能是对微信服务器 POST 给开发者服务器的各种消息进行验证、解析、封装、解密、加密以便后续业务逻辑模块的处理，如图 4-3 所示。

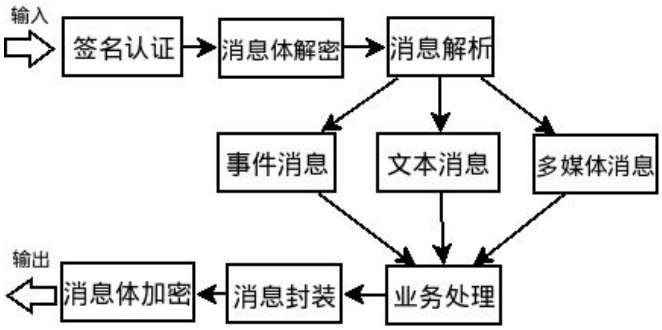

图 4-3　消息处理模块逻辑架构图

① 签名验证

签名验证[33]的主要目的是确认消息是否由微信服务器 POST 过来的，在接入开发者模式以后，微信服务器发送 POST 请求时会在开发者 URL 地址后面携带以下参数，如表 4-1 所示。

表 4-1　签名所需参数

参　数	描　述
signature	微信签名
timestamp	时间戳
nonce	随机数
echostr	随机字符串

签名验证的校验方式由以下 3 个步骤组成：

a)将在微信公众平台接入开发者模式时填写的 token 和 URL 中的 timestamp、nonce 两个参数进行字典排序；

b）将以上排好序三个参数拼接成字符串进行 sha1 加密，得到加密字符串；

c）将加密后字符串与 URL 中的 signature 进行比较，相同则表示请求来自微信服务器。

如果加密后字符串与 signature 完全匹配，校验成功，表示消息源自微信服务器，可进行业务逻辑处理。如果失败，返回为空。Python 代码实现如下：

```
for key in [ 'nonce', ‘timestamp’ ]:
    argsParams.append(token)
    v = request.query.get(key)
    if v: argsParams .append(v)
    argsParams.sort()
    if sha.sha(''.join(argsParams )).hexdigest() == request.query.get('signature'):
        return True
    return False
```

其中，request 是 Python Bottle 框架中 BaseRequest 类，用于操作 HTTP 请求，它定义了许多成员变量和成员方法，方便开发者获取 HTTP 请求中的各类参数，如 COOKIES、POST、body、headers、froms 等。在签名时使用了 Python 的 sha 模块[34]对连接字符串进行签名。

② 消息体的加解密

微信公众平台给开发者提供了多种消息传输方式。为了进一步提高微信公众号的安全保障，在接入开发者模式时，设置消息加密方式为安全模式。在该模式下，requset.body.read()获取到的消息为纯密文。在加解密之前，需要检查消息的真实性，在安全模式和兼容模式下，URL 上会增加两个参数 encrypt_type 和 msg_signature，encrypt_type 表示加密的类型，msg_signature 表示对消息的签名，开发者根据这两个参数来判断是哪以中接入模式。加密的核心代码实现如下：

```
def encrypt(self,text,appid):
    text = self.get_random_str() + struct.pack("I",socket.htonl(len(text))) + text +
appid
    pkcs7 = PKCS7Encoder()
    text = pkcs7.encode(text)
    cryptor = AES.new(self.key,self.mode,self.key[:16])
    ciphertext = cryptor.encrypt(text)
    return ierror.WXBizMsgCrypt_OK, base64.b64encode(ciphertext)
```

在加密 encrypt()函数中，首先调用 get_random_str()函数生成 16 位随机字符串添加到明文的开头，然后使用自定义的填充方式对明文进行补位填充，然后采用加密算法 AES 对消息体进行加密，再使用 BASE64[35]对加密后的字符串进行编码。解密函数 decrypt()是加密函数的逆过程。

③ 消息解析封装

获取到的消息体是 XML 数据形式，为了便于使用消息中的数据，需要把 XML 格式的数据转换成 Python 的字典形式。在返回消息时，需要把数据组织成微信服务器要求的 XML 格式，由于每种消息的 XML 字段不同，XML 的形式也有差异。因此，针对不同的消息定义不同的消息模板类。以下是消息解析的代码实现。

```
import xml.etree.cElementTree as ET
def _parse_message(body):
    root = ET.fromstring(body)
    message = {}
    for child in root:
```

```
            message[child.tag] = child.text
return message
```

在消息解析_parse_message()函数中，引用了 Python 的 XML 模块，该模块把 XML 数据中节点的标签存放在 tag 中，把节点标签中的内容存放在 text 中。在图文消息类模板 News()中，在每次需要返回图文消息时，只需实例化 New()类对象传入对应的数据即可。

4.1.2 业务查询办理模块设计与实现

业务查询办理模块主要包括功能匹配、报文封装、消息队列、运营商业务系统调用，其业务查询办理模块逻辑架构如图 4-4 中所示。

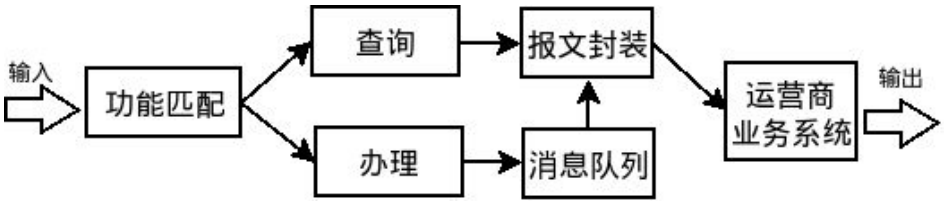

图 4-4　业务查询办理模块逻辑架构图

运营商涉及到多级业务。比如，流量办理分为月度包、季度包、特惠包等，每种类型下面还有不同流量额度，为了统一管理，为业务设置相应的编码。如查询类为 1，办理类为 2，流量办理为 2.1 等。运营商业务主要分为查询和办理两类，不同的业务通过报文封装模块组织报文，最后调用通信运营商的业务系统。在该模块中，详细介绍消息队列和运营商业务系统接口设计。

① 消息队列

在开发者服务器调用运营商业务系统进行业务查询办理时，办理类业务的响应时间长，往往会超过微信公众平台重试机制时长 5 秒时间。因此，对于办理类业务，把业务办理请求先入队进行排队处理，不必等到运营商业务系统返回就对微信服务器返回响应，防止超时。当业务系统处理完以后，通过短信、模板消息等异步通知主动给用户下发办理结果。架构中使用 RabbitMQ 消息队列 Work queues 工作方式，生产者端的代码实现如下：

```
connection = pika.BlockingConnection(pika.ConnectionParameters(''))
channel = connection.channel()
channel.queue_declare(queue="task_queue")
channel.basic_publish(exchange='',routing_key='task_queue',body=msg)
connection.close()
```

消息队列生产者使用 python 的 pika[36]库与消息队列 Server 端建立连接，把

内容发送到 task_queue 中，然后断开连接。消息队列消费者端的代码实现如下：

```
connection=pika.BlockingConnection(pika.ConnectionParameters(host=''))
channel = connection.channel()
channel.queue_declare(queue='task_queue', durable=True)
def callback(ch, method, properties, body):
    ch.basic_ack(delivery_tag = method.delivery_tag)
channel.basic_qos(prefetch_count=1)
channel.basic_consume(callback, queue='task_queue')
channel.start_consuming()
```

消息队列消费者从服务器端获取消息，然后封装成报文，调用通信运营商业务系统，记录业务系统调用日志。当消费者从消息队列服务器端获取消息处理时需要耗费一定的时间。在这个过程中可能会由于某种异常导致本次处理中断，消费者无法通知消息队列服务器端处理结果。Server 端不知道本次处理是否成功，不确定是否执行消息出队。因此，RabbitMQ 消息队列提供了消息确认机制，只有当消费者处理完成功返回确认通知后才从队列中删除该消息。当然，该机制会占用服务器端额外的内存，却保证了消息不丢失。在调用业务系统处理返回结果后，给消息队列返回一个回执消息，并且通过模板消息异步通知用户办理成功或失败，最后记录接口结果日志。

② 运营商业务系统接口设计与实现

开发者服务器与运营商业务系统采用 SOAP 协议进行交互，SOAP 是一种简单的基于 XML 的协议。运营商业务系统通过 WSDL（Web Services Description Language）描述接口方法，在调用时，需要按照 WSDL 描述组织参数，并调用对应的方法。以下是请求返回的消息实例。

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:ns1="" xmlns:soapenv="">
    <soapenv:Body>
        <ns1:query>
            <commInput>
                <ns1:_client_ip>192.168.0.100</ns1:_client_ip>
                <ns1:_phone_id>15293000000</ns1:_phone_id>
            </commInput>
        </ns1:query>
    </soapenv:Body>
</soapenv:Envelope>
```

在调用运营商业务系统时，使用 Python 的 PySimpleSOAP 库来调用业务系统的 Web Service 接口。由于运营商业务系统的接口类型多，接口参数复杂。因此，为每一大类的接口分别封装其接口类以方便调用。

4.1.3 客服系统设计与实现

客服系统是用于帮助用户解决问题的，分为智能应答和人工客服两个模块，在智能应答库中未匹配问题时，提醒用户接入人工服务来解决问题。客服系统的逻辑架构图如图 4-5 所示。

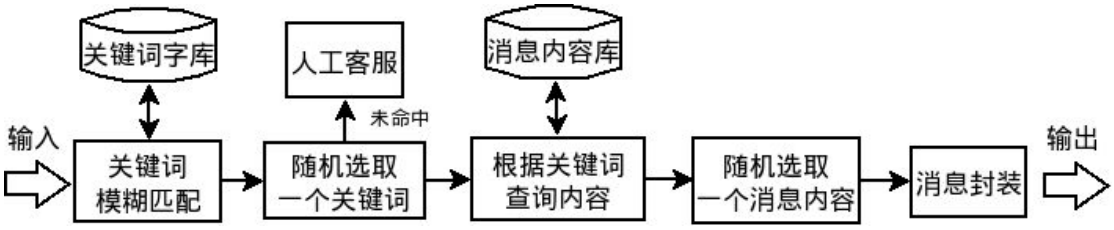

图 4-5　客服系统逻辑架构图

上图中根据用户输入的消息，查询智能应答关键字库中所有的关键词，与用户的输入消息进行模糊匹配，只要关键词包含或者与用户消息相同即命中。如果未命中，返回接入人工服务的提示消息。否则，从命中的关键词中随机选取一个，查询消息内容库，随机选取一个内容，经过封装以后返回给用户。智能回复的部分实现代码如下所示。

```
db.execute('SELECT replyindex FROM autoreply WHERE usermsg=%s', ( ch
oice  (matchitemlist  )))
replyindextuple = db.fetchall()
replyindexlist = []
for i in replyindextuple:
        replyindexlist.append(int(i.get('replyindex')))
        randomone = choice(replyindexlist)
return _get_autoreplycontent(randomone, db, message)
```

在客服系统智能应答模块中，主要使用 Python 的 random 模块中的 choice 函数来随机选取智能应答内容。在实现中，需要构建两张表，关键词表和关键词内容表，一个关键词可对应多个关键词内容，运营管理员可以通过人工客服回答问题的频数来丰富关键词库，把常见问题添加到关键词库中。

4.1.4 活动配置模块设计与实现

利用微信服务号的网页授权接口和分享接口，客户可以有两种获取活动页面

的方式，一是在服务号内点击自定义菜单或者发送关键字返回链接获取；二是从朋友分享或者朋友圈获取。分享功能可以有效地促进活动信息在用户间传播，可获得很好的推广效果。该模块提供两种类别的活动，一种是可配置简单活动，定义一个通用的页面架构，在后台管理系统中可以新建活动主题，对页面中的图片、文字进行配置，在配置好后，将活动链接配置到关键词回复内容，在用户点击链接时，通过查询数据库得到的图片地址和文字内容对页面进行渲染，实现简单活动页面可配置管理。另一类是复杂的活动，需要定制开发，如“双十一活动”、“流量红包”，本文以流量红包为例进行设计与实现。

在流量红包活动中，为每个参与流量红包活动的用户创建一个账户，该账户对应着用户微信 openid，账户中的流量可以兑换到已经绑定微信公众号的手机号账户中。系统为用户发送的每一个红包生成一个唯一 id，红包有效时间为 24 小时，超过 24 小时未领取完的红包流量值将做退账处理。流量红包的类型分为普通流量红包、随机流量红包两种。以下对随机流量红包算法进行设计。该算法的输入为流量红包的个数 fsgs、总流量值 fsll。返回为 fsgs 个随机分配的整数流量值，保存在数组 res[]中，fsgs 个随机流量值之和为 fsll。以下是随机流量红包算法的实现步骤。

① 输入随机流量红包的总个数 fsgs、总流量 fsll，fsgs 与 fsll 均为整数，已发送流量记为 ll，已发送个数为 gs，ll 与 gs 的初始值为零。

② 计算该红包的平均流量值 bs，bs = fsll/fsgs + 1。

③ 如果总发送个数 fsgs 减去已发送个数 gs 小于 2，把剩余的流量全部赋值给最后一个随机值，算法结束。否则，执行（4）。

④ 计算剩余总流量剩余发送个数的平均值 avg，avg = (fsll - ll))/(fsgs - gs)。

⑤ 根据 bs 的值生成一个大于 0，小于 1 的随机数 rand。

⑥ 本次随机流量的值为 int(round(rand*avg))， int(round())是对生成的随机值进行四舍五入计算。

代码实现为：

```
def get_Random_ll(fsll , fsgs):
    gs = 0, ll = 0
     res = [], bs = fsll/fsgs +1
    if bs >= 10:
        bs = 9
    for gs in range(fsgs):
        a = 0
        if fsgs - gs == 1:
```

```
            a = fsll - sum(res)
        else:
            a = int(round(float((fsll - ll))/(fsgs - gs)*(float(random.randint(1
0-bs, 10+bs))/10)))
        res.append(a); ll = ll +a; gs = gs+1;
    return res
```

算法是根据当前 hbll、hbgs 的平均流量值乘以一个大于 0 小于 2 的随机数来实现的，当 bs 的值越大，得出的各流量随机值波动越大，bs 等于 2 时，各随机流量值相等。在代码实现时主要用到 Python 的 random 库来生成随机数，使用 round()函数来进行四舍五入计算。在第五章测试中，将会对流量随机算法的稳定性进行测试。

4.1.5 后台管理系统设计与实现

① 后台管理系统的设计

后台管理系统是运营管理员用于管理通信运营商微信公众号的，包括登陆、智能应答关键字配置、可配置活动配置、日志查询、模板消息推送等功能。管理员可以创建文本、图文、页面消息，并配置相应的关键字。在该平台上，客服、运营人员可以对绑定记录、活动记录进行查询，以减少投诉的处理。在查询到的业务办理没有成功时，管理员可以根据业务补办模块开通业务。

② 后台管理系统的实现

消息管理模块是对智能应答关键词及内容配置管理，可以对每一项进行编辑、删除、隐藏操作，一次性只能更新一条消息。可以对所有消息进行隐藏、显示操作。只有显示的消息才对用户是可见的，界面如图 4-6 所示。

图 4-6　后台管理系统的消息管理实现界面

模板消息推送模块是用于向用户主动推送模板消息的。管理员可以创建模板消息、更改已创建的模板消息，并通过上传号码文件发送模板消息。在确认消息推送后，后台会把号码文件中的信息入队列进行排队处理。当队列中的数据处理完成以后，以邮件的方式通知运营管理员，其操作界面如图 4-7 所示。

图 4-7　后台管理系统的模板消息推送界面

4.1.6 日志系统设计与实现

日志根据产生对象分为用户日志、系统日志。为了方便查询，用户日志存放于数据库中，系统日志以文件的形式分月保存。根据用户的不同消息日志创建不同的日志表，通过系统脚本每天定时的给运营管理员发送日志内容，写入数据库的语句如下所示：

① 图片多媒体日志

```
db.execute("insert into logimage (uid, mediaid, picurl, area) values (%s, %s, %s, %s)",(fromusername, message['MediaId'], message['PicUrl'], area))
```

② 事件日志

```
db.execute('INSERT INTO eventlog (content, logtype, openid,area) VALUES (%s, %s, %s, %s)', (eventkey, 'Event', message.get('FromUserName'), area))
```

③ 文本日志

```
db.execute('insert into log (content, logtype, openid, area) values (%s, %s, %s, %s)', (contentencode, messagekey, message.get('FromUserName'), area))
```

系统日志包括消息体日志、Web 服务器访问日志。Web 服务器访问日志包括 Nginx 日志、uWSGI 日志、网络监控日志、CPU 平均负载日志。Nginx 日志包括所有请求的记录，静态资源请求和动态资源请求。uWSGI 日志记录了当前应用接收的动态请求日志。消息体日志可以查看到当前 HTTP 请求体中的内容，结合 uWSGI 日志，开发者可方便地进行调试、错误排查，以下是消息体日志的实现。

```
import logging
FORM = '%(asctime)-15s %(levelname)s %(message)s'
logging.basicConfig(filename='/srv/log/message.log',level=logging.DEBUG,
format=FORM)
def log(message, level="info"):
    logging.info(message)
```

在日志系统实现中直接引用了 Python 的 logging 日志库，logging.basicConfig 是配置日志的保存目录和显示格式。日志的输出级别设置为 DEBUG，只有 DEBUG 级别及以上的日志才会输出。在 log()函数中，输出的是 INFO 信息，消息体日志格式如下所示：

```
2016-02-02 03:03:10,044 INFO post area is wx-kf & body is
<CreateTime>1454353373</CreateTime>
<MsgType><![CDATA[event]]></MsgType>
<Event><![CDATA[TEMPLATESENDJOBFINISH]]></Event>
<MsgID>412230141</MsgID>
<Status><![CDATA[success]]></Status>
</xml>
2016-02-02 03:03:10,045 INFO post msgtype is event & area is wx-kf
```

以上示例详细的显示了该消息类型为事件类型，事件的内容为模板消息推送后的系统返回消息，模板消息的推送状态为 success。另外，根据 area 可以判断是哪一个公众号在进行模板消息推送。

4.1.7 数据库设计与实现

在架构中使用的是 MySQL 数据库。MySQL 的 UTF-8 编码只支持 1-3 个字节，有些用户的昵称可能包含 Emoji 表情符号，Emoji 表情符号需要 4 个字节来存储。如果使用 UTF-8 编码表创建数据库和表，在插入 Emoji 数据时会报“Incorrect string value: '\xF0\x9F\x98\x8A\xF0\x9F...' 错误，为了正确的显示中文内容，放在数据丢失，需要在数据库配置文件里对客户端、服务器端的编码设置 utf8mb4，数据库重启后配置生效。以下为创建应用数据库示例：

```
CREATE DATABASE `wechat` /*!40100 DEFAULT CHARACTER SET utf8mb4*/ ;
```

根据第三章中的需求分析设计对应的数据库表以存放相应的数据。涉及到不同类别的表，主要包括 BIND（关联表，用户微信 openid 与手机号码关联表）、USER（用户信息表，存放微信个人资料信息）、LOG（日志类表，其中包括事件

消息日志、多媒体消息日志）、AUTOREPLEY（智能应答关键词字表）、ACTIVITY（活动配置表）、PUSHMSG（模板消息配置表）、DUOKF（人工客服接入表）、MENU_STRUCT（菜单子菜单配置表）、MENU（菜单内容表）、HB_USERINFO（流量红包用户信息表），以下直接通过表结构来说明数据对象。

① BIND 关联表

关联表属性包括主键 id、用户微信 openid、手机号码、用户号码区域、公众号标识、时间，关联表是存储用户微信 openid 与用户手机号码关联关系的表，以便通过用户手机号码为用户提供服务。id 是 Bind 表的唯一主键，为了有效的提高查询效率，为其创建 bindindex、bindindex2 索引。关联表的创建语句如下所示：

```
CREATE TABLE `bind` ( `id` int(11), `phonenum` varchar(20), `openid` varchar(100), `datetime` timestamp, `passwd` varchar(10), `attcode` varchar(10), PRIMARY KEY (`id`), KEY `bindindex` (`openid`(10), `phonenum`), KEY `bindindex2` (`phonenum`));
```

② USER 用户信息表

用户信息表属性包括主键 id、用户微信 openid、用户微信昵称、用户是否关注公众号标识、公众号标识、时间，用户信息表保存用户的微信个人信息，当用户关注公众号时，开发者服务器就通过用户微信 id 获取用户的个人信息（昵称、头像等），以便在活动中使用，减少不必要的网络请求。其建表语句如下：

```
CREATE TABLE `user` ( `id` int(11) AUTO_INCREMENT, `uid` varchar(100), `nickname` varchar(100), `subscribe` int(2) DEFAULT '1', `datetime` timestamp DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (`id`), KEY `userindex` (`uid`(10), `area`));
```

③ LOG 日志类表

日志表属性包括主键 id、用户微信 openid、日志内容、日志类型、时间。日志表是存储用户访问信息的表，根据消息类型的不同，存储在不同的日志表中，以便后续查询。其建表语句如下：

```
CREATE TABLE `log` (`id` int(11) AUTO_INCREMENT, `content` text, `date` timestamp DEFAULT CURRENT_TIMESTAMP, `logtype` text, `openid` varchar(100), PRIMARY KEY (`id`));
```

④ AUTOREPLEY 智能应答关键词字表

智能应答关键词表属性包括主键 id、关键词、关键词索引、消息回复类别、时间，它是关键词库，系统能够匹配用户关键词的百分比由该表的内容丰富程度决定。其建表语句：

```
CREATE TABLE `autoreply` ( `id` int(11) AUTO_INCREMENT, `usermsg` text, `replyindex` int(11), `date` timestamp DEFAULT CURRENT_TIMESTAMP, `type` text, PRIMARY KEY (`id`));
```

⑤ PUSHMSG 模板消息配置表

模板消息配置表是存储管理员配置模板消息的内容，其表属性跟模板消息的字段一一对应，包括主键 id、模板消息 id、模板消息 url、模板消息概述、模板消息内容、模板消息尾注、时间。

```
CREATE TABLE `pushmsg` ( `id` int(11) AUTO_INCREMENT, `tempid` varchar(100), `tempurl` text, `firstdata` text, `keyword1` text, `keyword2` text, `keyword3` text, `keyword4` text, `remarkdata` text, `date` timestamp);
```

数据库中的表还包括活动、业务系统接口等相关表。在创建表时，需要考虑表的使用情况，根据表查询、更新的频率对表创建合适的索引，能够有效的减少数据库操作时间。

4.2 可扩展分布式架构设计与实现

可扩展分布式架构是基于微信公众平台基础架构的分布式实现，保证服务的可用性和可扩展性。以下对可扩展分布式架构的相关问题进行讨论，对可扩展分布式架构设计与实现进行介绍。

4.2.1 可扩展分布式架构面临问题讨论

从可扩展分布式服务的可用性出发，可扩展分布式架构存在单点服务器可用性、应用服务器水平扩展、分布式环境中会话状态一致性、数据库可用性等问题，以下从 4 个方面进行讨论。

① 负载均衡高可用性讨论

高可用技术属于容错技术的范畴，冗余是实现系统高可用的常用方法[37]。在实际应用中，很多服务器是用于备份的，它们处在等待的状态。当前端某台负载均衡服务器宕机的情况下，另一台备份机自动接管其转发服务。负载均衡服务器位于服务的入口层，他负责把请求转发给应用服务器，保证其高可用是应用服务器提供服务的基础。在架构中，负载均衡服务器采用的是 Nginx，保证其高可用的应用很多，如 Keepalived、Heartbeat、corosync、pacemaker 等，架构中使用 Keepalived[38]来保证负载均衡服务器的高可用性。Keepalived 基于 VRRP 协议[39]。VRRP 协议将多台服务器组成一个组，给这个组分配一个虚拟 IP[40]，占用虚拟 IP 的主服务器负责转发请求，其他服务器作为备份服务器处于等待状态，主服务器

定时发送组播消息 vrrp 包，当某个时间点备份服务器没有收到来自主服务器的组播消息时，Keepalived 解除虚拟 IP 与主服务器的绑定，重新指向一个优先级最高的备份服务器来保证服务的高可用。Keepalived 实现的是前端轻量级的高可用，不需要共享存储。它一般与负载均衡技术（Lvs、Nginx、haproxy）一起工作实现集群高可用。

② 应用服务器水平扩展讨论

架构中的应用服务器全部采用相同配置、相同系统的虚拟服务器，它们的处理能力相当，所以 Nginx 采用静态负载均衡轮询方法来转发请求。通过统计当前负载均衡服务器的连接数来动态的扩展应用服务器的数量。统计 Nginx 的方法可通过 Nginx 的 ngx_http_stub_status_module 模块、Nginx 日志分析、自定义模块等方法来获取，当然也可以统计当前系统的 tcp 连接数间接的统计 Nginx 的连接数。Ngxtop 是通过分析 Nginx 日志来实时监控 Nginx 的简单命令行工具，它类似 top 命令展示出实时的统计内容。Nginx 自带的 ngx_http_stub_status_module 模块可以统计出当前的连接总数，可以根据该项判断架构的负载量，该模块是被动的，当需要 Nginx 主动汇报服务器状态时，可以通过 Nginx 自定义模块来实现。

③ 数据库可用性和可扩展性讨论

提高数据库的高可用性需要根据系统的具体应用场景、业务需求、运营能力、现有的基础设施进行具体的分析和设计。使用 MySQL 作为数据库的应用都要求实现数据库可用性和可扩展性[41]。数据库的可用性是指系统的应对能力，当数据库出现问题故障以后仍然能提供正常的服务。数据库可扩展性指数据库查询、更新操作跨越了多个 MySQL 数据库。MySQL 提供了不同级别的可用性解决方案，这些方案分为三类：数据复制、集群和虚拟系统、集群共享和复制，包括：MySQL Replication、MySQL Fabric、MySQL Cluster 等。在架构中采用 MySQL Replication 机制中的 dualMaster 方式来保证后端数据库的可用性和可扩展性。

④ 会话保存讨论

由于 HTTP 是无状态的协议，为了支持客服端与服务器端的交互，需要保存用户的交互状态信息，常用的技术包括 Cookies 和 Session[42]。Cookies 实现机制是把用户状态保存在客户端，用户向服务器发送请求时，携带了客户端存储的用户状态信息。Session 是把状态信息保存在服务器端，相对 Cookies 更加安全，能够有效阻止 Cookies 禁用、篡改。在架构中使用 Session[43]技术来保存用户的状态信息。在分布式系统中，请求可能被发送到应用服务器的不同节点，当用户多次请求被发送到不同的服务器时，将为用户生成不同的 Session 数据，各应用服务器保存的用户状态信息不一致可能会带来多种异常问题。

4.2.2 可扩展分布式架构设计与实现

在服务器实际部署中，每个单节点服务器都采用 Keepalived 技术保证其高可用性，通过负载均衡转发服务器对当前连接数实时统计，动态地扩展应用服务器的数量，可扩展分布式架构部署如图 4-8 所示。架构前端是两台 Nginx 负载均衡转发服务器，一台作为备份服务器，中间是两台应用服务器，应用服务器的数量可以水平扩展，后端是两台数据库服务器，使用 MySQL Replication 机制保证数据库的可用性。针对微信的网页授权接口，增加了网页授权服务器。负载均衡服务器与应用服务器之间可通过 uWSGI、HTTP 协议进行通信，对于 uWSGI 应用服务器容器使用 uWSGI 协议进行通信速度更快，并且 uWSGI 协议的一些特殊特性有利于 Nginx 与 uWSGI 间的通信。

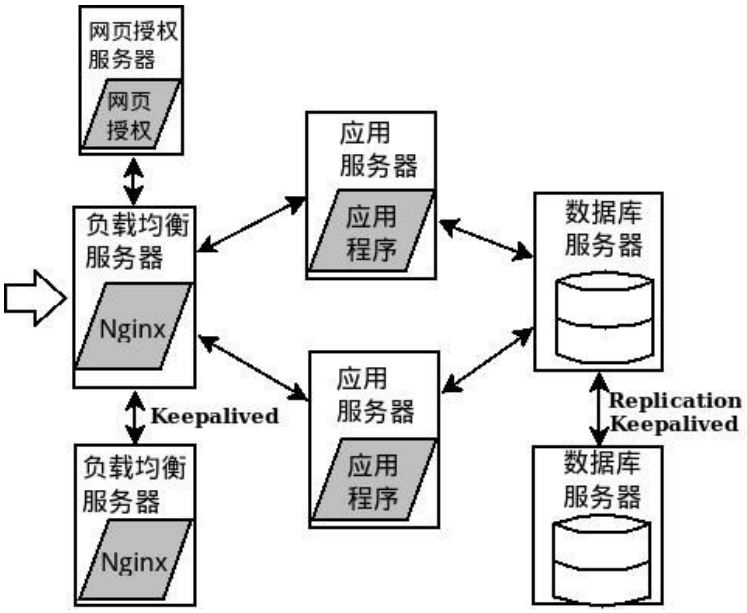

图 4-8　可扩展分布式架构部署图

大部分用户选择从微信朋友圈寻找阅读内容，为了获取从微信朋友圈进入通信运营商的活动页面的用户信息 openid，微信公众平台提供了网页授权机制。在网页授权过程中，其入口是微信服务器，微信服务器添加 code 参数将请求重定向到开发者服务器，开发者服务器通过 code 去微信服务器获取用于请求用户信息的 access_token，开发者服务器通过 access_token 再次访问微信服务器获取用户信息。在网页授权过程中，至少需要两次网络请求才能获取到用户的基本信息。在并发访问量高的情况下，网络请求过多会占用应用进程的连接数，导致应用服务器性能下降。在上面架构中，使用两台服务器作为网页授权接口服务器，并采用 Session 机制，有效的减少了网络请求的次数。以下是 4.2.1 中各问题解决方案的设计与实现。

4.2.3 负载均衡服务器高可用性设计与实现

通过 Keepalived 来保证 Nginx 负载均衡服务器的高可用性，Nginx 与 Keepalived 有主从配置、双主配置两种方案，在架构中采用主从配置方案来实现 Nginx 高可用性。正常情况下，主 Nginx 服务器提供服务，另一台备份服务器处于等待状态。在主服务器出现故障后，通过邮件方式通知系统管理员，同时备份服务器自动接管负载均衡转发服务，其结构如图 4-9 所示。

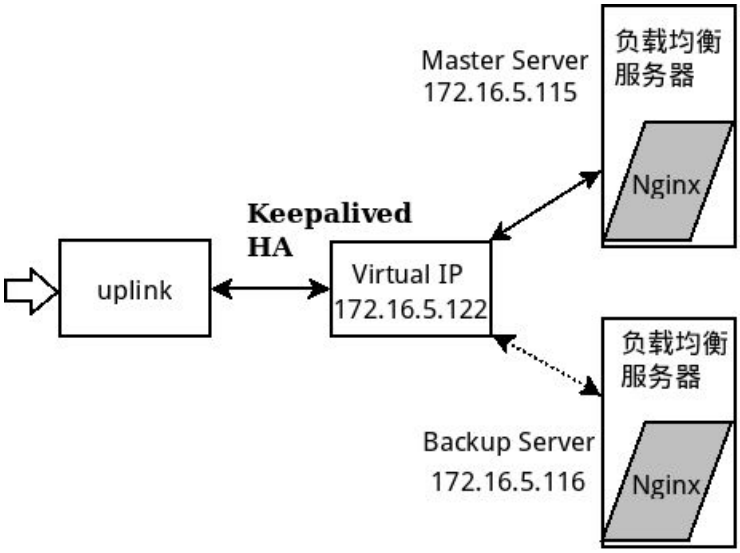

图 4-9　Nginx 高可用部署图

如图 4-9 所示，在 Nginx 高可用方案中，通过虚拟 IP 地址 172.16.5.122 绑定到主 Nginx 负载均衡服务器 172.16.5.115 的 eth0 网卡上。同时，在备份服务器中配置 Keepalived，其中 vrrp_instance 的 state 值设置为 BACKUP。

以下是具体实现步骤。

① Keepalived 安装

```
wt@wt-pc:~$ sudo aptitude install keepalived
```

② Nginx 监控脚本

该脚本监控 Nginx 的运行状态，如果 Nginx 进程不存在，尝试重新启动，如果失败，则停止 Keepalived，由 Backup 服务器接管服务。

```
wt@wt-pc:~$ sudo vim /etc/keepalived/check_nginx.sh
#!/bin/bash
counter=$(ps -C nginx --no-heading|wc -l)
if [ "${counter}" = "0" ]; then
      /usr/local/bin/nginx
      sleep 2
```

```
    counter=$(ps -C nginx --no-heading|wc -l)
    if [ "${counter}" = "0" ]; then
        /etc/init.d/keepalived stop
    fi
fi
```

③ Keepalived 配置

在主服务器和备份服务器上都需要配置 Keepalived，其配置文件区别在于以下参数，state MASTER - state BACKUP，priority 101 - priority 100，mcast_src_ip 172.16.5.115 - mcast_src_ip 172.16.5.116，配置内容详见[44]。在 Keepalived 配置中，主、备服务器需要配置相同的虚拟 IP 地址，虚拟 IP 地址需要是相同网段的没有被占用 IP。备份服务器的优先级别应该小于主服务器。另外，两台 Nginx 负载均衡转发服务器的 Nginx 配置需要相同。

4.2.4 应用服务器水平扩展设计与实现

通过 Nginx 自带的 status 模块，编写相应的 Python 脚本获取 Nginx 当前的平均连接数。根据连接数的大小设置多个阈值，根据不同的阈值使 Nginx 动态加载不同的配置文件实现架构中应用服务器的水平扩展。Status 模块提供的参数如下表 4-2 所示。

表 4-2　Status 模块监控参数

参数名称	参数描述
Active connections	当前活跃的用户连接
accepts	接收到的用户连接总数
handled	Nginx 处理的用户连接总数
requests	用户请求总数
Reading	连接中 Nginx 读取请求首部的个数
Writing	连接中 Nginx 写返回给用户的个数
Waiting	没有请求的活跃用户连接数

在新版本的 Nginx 中，ngx_http_stub_status_module 模块是默认加载的。只需要修改配置文件添加一个 location 即可通过 Web 查看。以下是应用服务器水平扩展的实现步骤。

① 获取 Nginx 状态信息

修改 Nginx 的配置文件，在 server 节点里添加一个 location，并且限制访问地址为本机 IP，如下所示：

```
wt@wt-pc:~$ sudo vim /etc/nginx/sites-enabled/default
        location /basic_status {
```

```
            stub_status;
            allow 172.16.5.116;
            deny all;
        }
```

重新加载 Nginx 配置文件。可以通过浏览器 http://172.16.5.116/basic_status 查看当前 Nginx 的状态信息，如图 4-10 所示。

```
Active connections: 1
server accepts handled requests
 413826 413826 413837
Reading: 0 Writing: 1 Waiting: 0
```

图 4-10 Nginx 连接数示例图

② 编写脚本

编写脚本来获取 Nginx 的状态信息，周期性获取图 4-10 中所示的的 requests 参数值来计算出单位时间平均连接数，根据单位时间平均连接数来增加、减少应用服务器的数量，实现应用服务器的水平动态扩展，部分代码实现如下：

```
requestNum1 = 0; requestNum2 = 0; reloadValue = 0
requestInfo = urllib2.urlopen( "http://172.16.5.116/basic_status" ).read()
requestNum1 = int( requestInfo.split( "server accepts handled requests" )[1].split(" ")[3].strip() )
while True:
    time.sleep(5)
    requestInfo = urllib2.urlopen( "http://127.0.0.1/basic_status" ).read()
    requestNum2 = int( requestInfo.split( "server accepts handled requests" )[1].split(" ")[3].strip() )
    requstAvg = (requestNum2 - requestNum1)/5
    requestNum1 = requestNum2
    if requstAvg >10000:
        if reloadValue != 5:
            os.system("rm /etc/nginx/sites-enabled/default")
            os.system("ln -s /etc/nginx/sites-available/default10000 /etc/nginx/sites-enabled/default")
            os.system("/etc/init.d/nginx reload")
            reloadValue = 5
        print requstAvg, "    LOAD CONFIG FILE default10000"
```

在以上代码中，每隔 5 秒通过 Python 网络请求库 urllib2 来获取当前 Nginx 的 requests 参数，使用 split()函数分割获取当前的 requests 参数值，计算 5 秒内两次平均值与区间阈值比较。当平均值不在相同的区间内时，使 Nginx 加载不同的配置文件实现应用服务器的水平扩展。同时，使用 reloadValue 作为标志位来防止相同状态下 Nginx 无效的加载配置文件。

4.2.5 MySQL 数据库的可用性设计与实现

MySQL Replication 机制是基于复制 MySQL 数据库二进制日志文件中更新操作（更新、删除等）记录来实现的。从数据库从主数据库的二进制文件中复制更新记录，再解析日志并应用到自身。为了保证数据库的高可用性，减少在特定场合下需要进行 Master 切换，采用 dualMaster 复制架构，并且采用双主 Keepalived 方案来保证高可用，如图 4-11 所示。

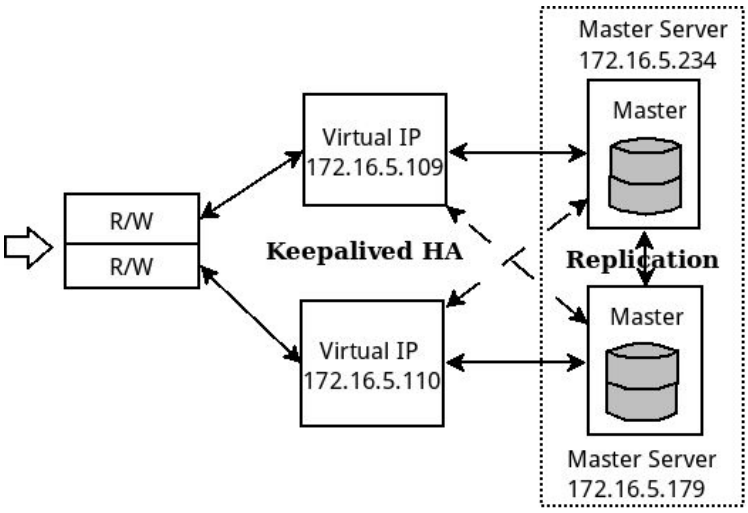

图 4-11 Mysql 高可用部署图

如图 4-11 所示，两台数据库通过配置 Replication 机制来保证数据库的可用性。dualMaster 复制架构是两个 MySQL 数据库互相作为 Master 数据库进行复制，任何一方的更新操作都会同步到另一方。MySQL 的 BinaryLog 中记录了当前 MySQL 数据库的 server-id，可以避免两台数据库间的循环复制。在图 4.11 中，为了利用备份数据库资源，使两个数据库都能提供读写服务，使用 Keepalived 双主模式来保证数据库高可用性。但是，当两数据库同时往同一张表写入数据时，复制机制可能会导致该表中的数据不一致。因此，在 Keepalved+dualMaster 架构下，约定相同表的写操作只在一端进行可有效的避免复制冲突，并且可提高数据库服务器的利用率。MySQL 高可用实现步骤如下，数据库服务器地址：Master1：172.16.5.234，Master2：172.16.5.179。

① 首先，修改 MySQL 数据库的配置文件，设置 server-id，声明需要同步

的数据库 test，以及不需要同步的数据库，在配置文件中添加以下内容：

```
server-id = 1
log-bin =  /var/log/mysql/mysql-bin.log
binlog-do-db=test
replicate-ignore-db=information_schema
replicate-ignore-db=mysql
replicate-ignore-db=performance_schema
auto-increment-increment= 2 # 整个结构中服务器的总数
auto-increment-offset = 1 # 设定数据库中自动增长的起点
```

同样的，在 Master2 数据库配置文件中添加除 server-id 的相同内容，在 Master2 中 server-id 设置为 2，以避免两台数据库间的循环复制。log-bin 是声明需要复制的数据库日志文件。

② 添加数据库用户，在 Master1 数据库上创建用户 test12，并授予其同步复制的权限，设置访问密码，更新配置。相同的，在 Master2 数据库上创建用户 test34，执行 Master1 上相同的操作。

```
Master1：
CREATE USER 'test12'@'172.16.5.179';
GRANT REPLICATION SLAVE ON *.* TO 'test12'@'172.16.5.179' IDENTIFIED BY 'test12';flush privileges ;
Master2：
CREATE USER 'test34'@'172.16.5.234';
GRANT REPLICATION SLAVE ON *.* TO 'test34'@'172.16.5.234' IDENTIFIED BY 'test34';flush privileges ;
```

③ Master1 上指定 Master2 数据库为同步数据库，首先停止 SLAVE 机制，把 Master1 的主数据库设置为 Master2，访问用户为 test34，启动同步复制机制。

```
STOP SLAVE; CHANGE MASTER TO MASTER_HOST='172.16.5.179', MASTER_USER='test34'[45], MASTER_PASSWORD='test34'; START SLAVE;
```

④ Master2 上指定 Maste1 数据库为同步数据库，执行与 Master1 相同的操作，并启动同步复制机制。

```
STOP SLAVE; CHANGE MASTER TO MASTER_HOST='172.16.5.234', MASTER_USER='test12'[46], MASTER_PASSWORD='test12'; START SLAVE;
```

在实现了 MySQL 可用性后，通过 Keepalived 来提高 MySQL 数据库的利用率。Keepalived 实现 MySQL 数据库的高可用性同负载均衡服务器高可用性实现步骤，因为是双主配置，在 Keepalived 配置文件中需要配置主从 vrrp_instance。

4.2.6 分布式环境中会话保存设计与实现

微信的网页授权需要开发者服务器用 code（一次有效，过期时间为 5 分钟）去请求微信服务器获取用户 openid 和 access_token 值，然后通过 access_token 获取用户的详细信息，在这个过程中至少需要有 2 次网络请求，如果不与应用服务器分开，在高并发的情况下，很容易占用应用服务器连接数，造成应用服务器的拥塞。因此，使用接口服务器来负责网页授权，并采用 Session 机制来保存用户的状态，减少网络请求，其结构部署如图 4-12 所示。

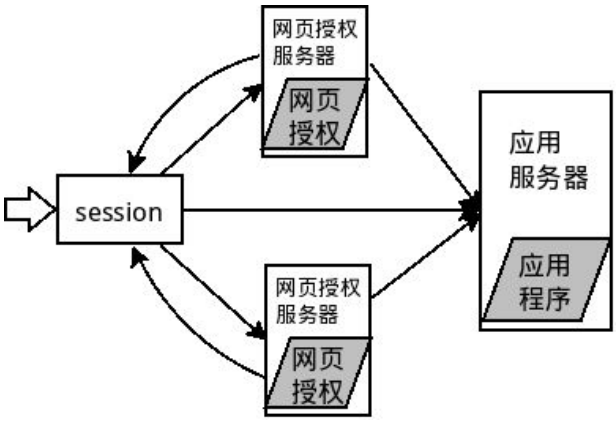

图 4-12 会话保存机制部署图

在以上部署图中，使用多台网页授权接口服务器分流授权，多台服务器中的 Session 中保存的用户状态可能不一致，解决的方法分为两种，一种是多台接口服务器共享 Session 数据；另一种方法使得相同的 IP 请求由同一服务器来处理。在架构中使用第二种方法。以下是会话保存实现步骤。

① Session 的实现

在 Python 的 Web 框架 Bottle 中使用 beaker 插件来实现 Session 机制，beaker 是一个缓存和会话的 WSGI 中间件，Session 实现的代码如下所示：

```
from beaker.middleware import SessionMiddleware
session_opts = {
        'session.type':'file',
        'session.data_dir':'./sessions',
        'session.cookie_expires':100000,
        'session.auto':True
        }
application = bottle.default_app()
application = SessionMiddleware(application, session_opts)
```

Session 需要在 Web 应用程序运行前进行配置。在上面代码中，设置 Session

的数据存储类型为文件，存储位置为当前应用目录下的 sessions 目录中，Session 数据存储时间为 100000 秒。

② Session 内容一致性实现

通过配置 Nginx 负载均衡转发策略为 ip_hash 把相同用户 IP 的请求转发到同一网页授权服务器中，以保证用户会话状态的一致性。Nginx 配置如下：

```
upstream rd {
    ip_hash;
    server 172.16.5.204:9000;
    server 172.16.5.205:9000;
    server 172.16.5.200:9001;
    server 172.16.5.201:9001;
}
```

Nginx 的 ip_hash 转发策略使得相同的用户授权请求根据其 IP 转发到同一网页授权服务器。如果用户更换了 IP 地址，需要重新授权。因此，设置 session 的过期时间为一个有限值。

4.3 小结

本章实现了通信运营商微信公众平台基础架构和可扩展分布式架构。前者实现了通信运营商微信公众平台基础架构的各个子模块功能，设计了系统数据库。后者设计且实现了可扩展分布式架构中单节点的高可用、应用服务器的水平扩展、分布式环境中的会话保存、数据库的可用性。

第五章　测试与分析

在第四章设计与实现中，主要从两个方面实现了基于微信公众平台的可扩展架构。该章节先对可扩展分布式架构的各个方面进行测试，然后对通信运营商微信公众平台基础架构中的随机流量算法进行稳定性测试，再分析用户的使用日志来重新规划微信公众号的自定义菜单结构，以提高公众号的服务器质量。

5.1 可扩展分布式架构测试

5.1.1 负载均衡服务器高可用性测试

在 4.2.3 节中，两台负载均衡服务器通过 Keepalived 实现高可用性。当 Master 服务器中的 Nginx 进程不存在时，终止 Keepalived 服务，虚拟 IP 会自动绑定到 Backup 服务器，保证负载均衡服务器高可用。为了测试其可用性，通过主动暂停 Master 服务器上 Nginx 服务，比较暂停前和暂停后的 Master、Backup 服务器的 IP 配置信息来验证。以下是两台 Nginx 服务器的测试结果。

① 在正常情况下，Master 服务器默认绑定虚拟 IP 地址 172.16.5.122，通过 Linux 下的网络配置工具 ip 查看，172.16.5.122 虚拟 IP 地址成功绑定到网卡 eth0 上，如图 5-1 所示。

```
wt@wt-pc:~$ ip a| grep eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    inet 172.16.5.115/24 brd 172.16.5.255 scope global eth0
    inet 172.16.5.122/32 scope global eth0
```

图 5-1　Master 服务器绑定虚拟 IP 结果图

② 当强制暂停 Master 服务器上 Nginx 服务时，Keepalived 监听脚本检测不到 Nginx 进程的存在，关闭 Keepalived 并解除虚拟 IP 与 eth0 的绑定关系，其网络配置信息如图 5-2 所示。

```
wt@wt-pc:~$ sudo /etc/init.d/nginx  stop
wt@wt-pc:~$ ip a| grep eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    inet 172.16.5.115/24 brd 172.16.5.255 scope global eth0
wt@wt-pc:~$
```

图 5-2　Master 服务器虚拟 IP 解除绑定结果图

③ 当虚拟 IP 与 Master 服务器解除绑定后主动与最高级别 Backup 服务器进行绑定，接管负载均衡转发服务，Backup 服务器网络配置信息如图 5-3 所示。

```
az@az:~$ ip a|grep eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    inet 172.16.5.116/24 brd 172.16.5.255 scope global eth0
    inet 172.16.5.122/32 scope global eth0
az@az:~$
```

图 5-3　Backup 服务器绑定虚拟 IP 结果图

在上面的测试中，Master 无法提供负载均衡转发服务时，虚拟 IP 自动绑定到 Backup 服务器以接管转发服务，保证了 Nginx 负载均衡服务器的高可用性。

5.1.2 应用服务器扩展性测试

在 4.2.4 节应用服务器水平扩展实现中可以获得 Nginx 每秒平均请求数。为了测试应用服务器的可扩展性，通过 Webbench[47]工具来模拟大量的并发请求，在不同的并发量访问下 Nginx 加载不同的配置文件来改变应用服务器的数量。从监控脚本日志输出结果中验证了应用服务器实现了水平可扩展。如图 5-4 所示，使用 Webbench 模拟并发用户量为 10000 个，持续访问时间为 30 秒。

```
wt@wt-pc:~$ webbench -c 10000 -t 30 http://127.0.0.1/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Benchmarking: GET http://127.0.0.1/
10000 clients, running 30 sec.

Speed=1711122 pages/min, 8727374 bytes/sec.
Requests: 855413 susceed, 0 failed.
wt@wt-pc:~$
```

图 5-4　Webbench 并发请求测试图

根据图 5-4 中 webbench 的执行结果所示，在 30 秒内总共发送了 855413 个请求，成功 855413 个，失败为 0 个。在 Nginx 监控脚本中，设置了 5 个档位的每秒平均请求数，分别是大于 10000/sec、5000~10000/sec、1000~5000/sec、500~1000/sec、小于 500/sec。监控脚本输出日志如图 5-5 所示。

```
wt@wt-pc:~/wt$
wt@wt-pc:~/wt$ sudo python monitor.py
1   LOAD CONFIG FILE default500
0   LOAD CONFIG FILE default500
 * Reloading nginx configuration nginx                                  [ OK ]
19180   LOAD CONFIG FILE default10000
25839   LOAD CONFIG FILE default10000
20034   LOAD CONFIG FILE default10000
27419   LOAD CONFIG FILE default10000
24901   LOAD CONFIG FILE default10000
24300   LOAD CONFIG FILE default10000
10007   LOAD CONFIG FILE default10000
 * Reloading nginx configuration nginx                                  [ OK ]
3303   LOAD CONFIG FILE default1000
 * Reloading nginx configuration nginx                                  [ OK ]
0   LOAD CONFIG FILE default500
1   LOAD CONFIG FILE default500
```

图 5-5　应用服务器扩展性测试结果图

如图 5-5 所示的监控脚本测试结果中，第一列表示每秒请求数，LOAD CONFIG 表示当前加载的配置文件，Reloading 表示动态加载 Nginx 配置文件的执行结果，在并发请求数阈值超过 1000 和 10000 的时候，Nginx 都成功加载了不同的配置文件。由上测试可知，在 Nginx 并发量高的情况下，系统会自动加载不同的配置文件来增加应用服务器的数量，以满足高并发的需求。在并发量降低后，又自动加载低配置文件，实现了应用服务器的水平扩展。在相同连接数阈值内，使用 reloadValue 标志位来标志当前的状态，防止无效的配置文件加载。

5.1.3 MySQL 数据库 dualMaster 可用性测试

在 4.2.5 节中数据库采用 MySQL dualMaster 复制架构。在两端都配置完成后，执行 start slave 通过 SHOW SLAVE STATUS\G 可查看同步日志。通过查看更新操作的同步日志来测试 MySQL 数据库 dualMaster 架构的可用性。

在图 5-6 中显示了 Master1 172.16.5.234 数据库上的 SLAVE 状态：

```
mysql> show slave status\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.16.5.179
                  Master_User: test34
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000001
          Read_Master_Log_Pos: 107
```

图 5-6　Master1 同步状态日志截图

在图 5-7 中显示了 Master2 172.16.5.179 数据库上的 SLAVE 状态：

```
mysql> show slave status\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.16.5.234
                  Master_User: test12
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000001
          Read_Master_Log_Pos: 107
```

图 5-7　Master2 同步状态日志截图

根据日志显示，两数据库都是对方的 Master 和 Salve，在进行数据更新操作时会同步到另一台，在其中任何一台数据库出现故障时，另一台的数据总是最新的。Keepalived HA 机制可有效的利用了两台数据库，数据库的高可用测试同负载均衡服务器高可用性测试。两种机制的结合，同时保证了数据库性能和可用性。

5.1.4 基于微信公众平台的可扩展架构的性能测试

为了测试基于微信公众平台的可扩展整体架构的性能，分别从并发数、响应时间两个方面对架构与单应用节点进行测试。为了不影响现有业务，重新搭建一套测试环境。架构的测试环境为下列 3 台服务器的分布式部署，单台服务器的测试环境为分布式环境中的一个应用节点。测试环境服务器配置如表 5-1 所示：

表 5-1　基于微信公众平台可扩展架构测试环境

类　型	发行版内核	CPU 架构	内　存	网 卡
Nginx(N1)	Ubuntu 15.10，4.2.0	Intel，64bit，3.10GHz	DDR3，4G	100M
Server(S2)	Ubuntu 14.04，3.13.0	Intel，64bit，3.10GHz	DDR3，4G	100M
Server(S3)	Ubuntu 14.04，3.13.0	Intel，64bit，3.10GHz	DDR3，4G	100M

首先采用 NetPIPE[48]工具对测试环境各个节点的网络吞吐量进行测试，3 台服务器都是 100M 网卡。其测试结果如图 5-8 所示。

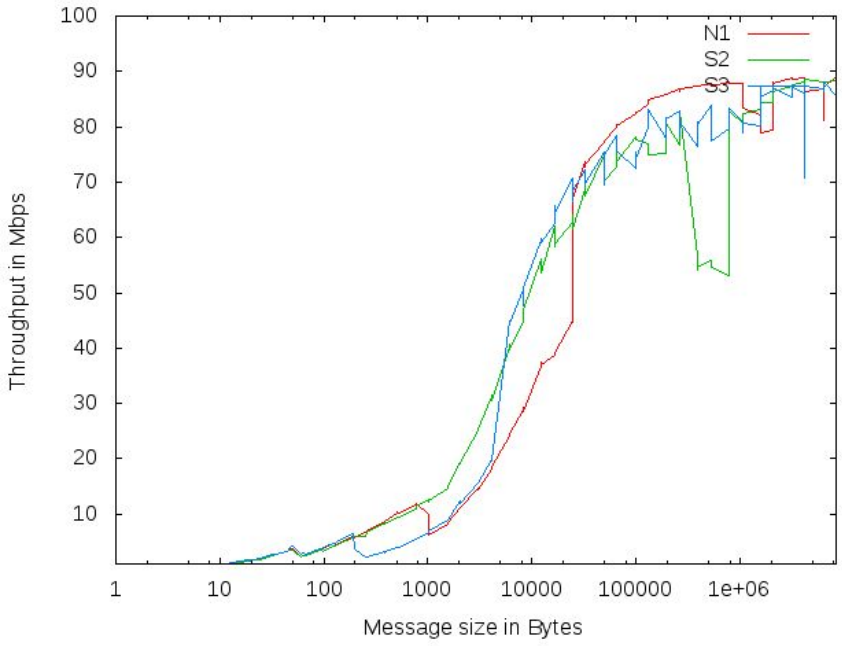

图 5-8　单台服务器的网络吞吐量图

根据图 5-8 所示，随着传输的数据包的大小不断增大，3 台测试服务器的最大网络吞吐量接近 90Mbps。同时，从 NetPIPE 其他信息得出，随着数据包的不断增大，传输的延时成指数增加。在此网络环境下，使用自动化测试工具 http_load[49]对单应用节点和分布式架构的单位响应请求量、连接平均响应时间进行测试比较。

①　单位时间响应请求量

单位时间响应请求量是对应 QTP[50]中的每秒响应用户数，该参数能较好的反

应出架构的性能，数值越大，单位时间内响应的请求数越多。

为了比较单台服务器与分布式架构服务器的响应请求量，通过 http_load 工具分别对两类服务器进行测试。每次测试的总访问次数为 1000 次，并发访问进程的数量由 1 递增到 500 个，每增加 1 个进程对服务器进行一次性能测试。总测试次数为 500 次。每台应用服务器开启了 100 个并发应用进程来处理请求，其测试结果如图 5-9 所示。

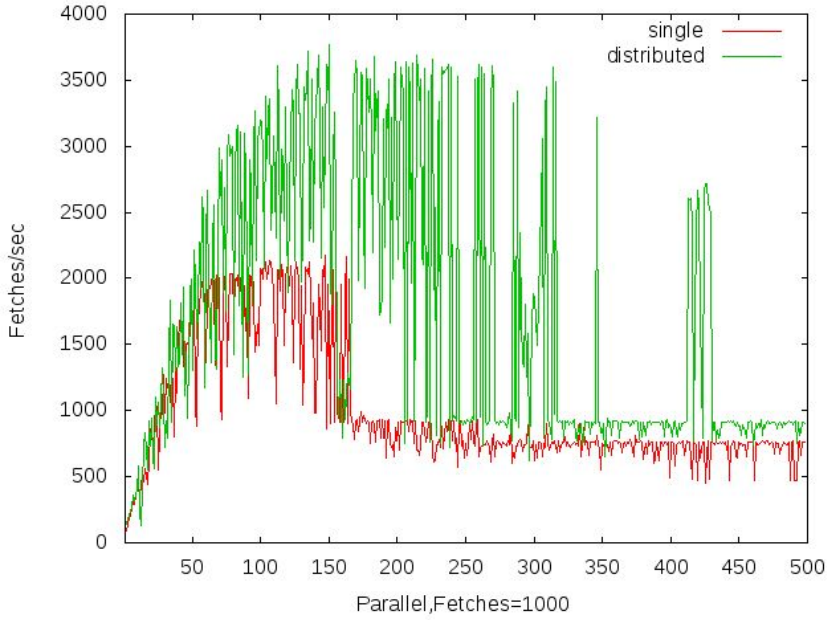

图 5-9　单位时间响应请求量比较结果

图 5-9 中，红色曲线表示单台服务器的单位时间响应请求量，绿色曲线表示分布式架构服务器的单位时间响应请求量。单台应用服务器开启了 100 个应用进程来处理请求，所以在并发访问进程小于 100 时，单位时间响应请求量是递增的。当并发请求量超过 100 时，单台应用服务器单位时间响应请求量达到 2000，接近了饱和。而分布式架构的单位时间请求量继续递增，当并发进程超过 250 时，分布式架构服务器的单位时间请求量达到最大。在达到最大单位时间请求量时，分布式架构服务器能承载的并发进程量大约是单点服务器的 2.5 倍。在一定程度上表明，分布式架构的单位时间请求量达到饱和时的最大用户承载量高于同等数量的单点应用服务器。随着并发请求量的继续增加，两类服务器性能都达到了饱和，单位时间请求量趋于稳定。

② 平均响应时间

平均响应时间对应 QTP 中的响应时间，该参数反映出服务质量，连接平均响应时间越短，服务质量越好，单节点应用服务器与分布式架构的平均响应时间测试结果如图 5-10 所示。

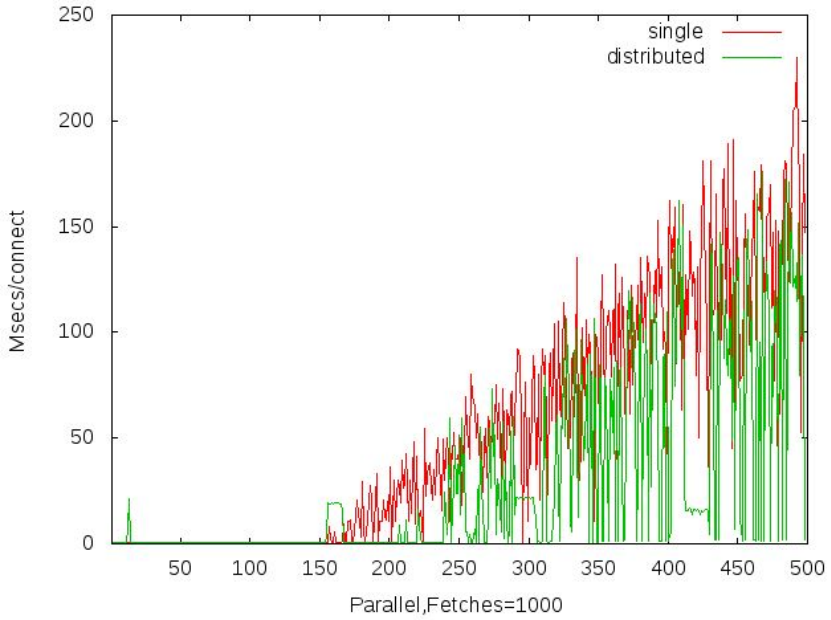

图 5-10　平均响应时间比较结果

由图 5-10 所示，在并发访问量低于 150 的情况下，响应时间几乎为零，随着并发访问量的增加，响应时间也成线性增长。但是，总体来说，可扩展架构比单台服务器的响应时间短。

结合单位时间响应请求量和平均响应时间分析，在服务器并发量达到饱和（服务器中应用的进程数）时，其单位时间响应请求量最高，平均响应时间也很短。随着并发访问量的增加，单位时间响应请求量降低后趋于稳定，响应时间呈线性增长。总的来说，分布式架构对服务的性能有一定的提升。

5.1.5 随机流量红包算法稳定性测试

为测试 4.1.4 节中随机流量红包算法的稳定性。在输入参数 fsll=500，fsgs=20 时，分别对算法进行 10 次、1000 次、10000 次测试，计算每次测试数据的平均值序列并比较，以测试随机流量红包算法的稳定性，10 次测试的数据如下：

[33, 47, 44, 11, 30, 18, 34, 9, 7, 12, 3, 45, 23, 16, 11, 47, 25, 14, 18, 53]

[5, 13, 19, 16, 17, 20, 38, 52, 16, 28, 28, 30, 46, 42, 39, 16, 4, 12, 21, 38]

[25, 30, 47, 37, 43, 17, 26, 8, 27, 7, 37, 30, 2, 5, 32, 28, 42, 4, 5, 48]

[25, 38, 39, 28, 28, 18, 19, 38, 31, 11, 23, 16, 26, 2, 45, 11, 10, 34, 46, 12]

[15, 31, 3, 48, 40, 17, 5, 10, 39, 42, 38, 31, 34, 19, 32, 29, 15, 19, 23, 10]

[48, 31, 9, 32, 45, 36, 38, 24, 20, 18, 34, 24, 32, 11, 31, 23, 13, 5, 8, 18]

[15, 3, 51, 48, 38, 30, 2, 39, 9, 39, 5, 29, 36, 33, 27, 33, 11, 17, 14, 21]

[25, 3, 26, 24, 37, 31, 33, 25, 39, 33, 11, 40, 15, 27, 7, 17, 16, 52, 21, 18]

[30, 37, 17, 20, 47, 19, 40, 40, 12, 26, 25, 31, 12, 10, 22, 43, 3, 20, 18, 28]

[28, 45, 31, 21, 38, 2, 2, 13, 8, 14, 12, 6, 60, 6, 68, 23, 46, 41, 25, 11]

计算以上 10 次数据的平均值如下序列所示。

[24.9,27.8,28.6,28.5,36.3,20.8,23.7,25.8,20.8,23,21.6,28.2,28.6,17.1,31.4,27,18.5,21.8,19.9,25.7]

当测试次数为 1000 次时，1000 组数据平均值序列为：

[25.906,24.814,25.836,24.785,24.550,25.082,25.424,24.416,25.142,25.259,25.167,25.182,24.599,24.707,24.638,24.941,25.494,24.989,25.564,23.505]

当测试次数为 10000 次时，10000 组数据平均值序列为：

[25.2884,24.9612,25.0371,25.0315,25.1142,24.976,25.0709,24.8581,24.9926,24.7901,24.8802,24.8473,25.0819,25.1265,25.0774,24.7169,25.1568,24.8023,25.1617,25.0289]

10 次、1000 次、10000 次流量平均随机值序列的对比折线图如 5-11 所示。

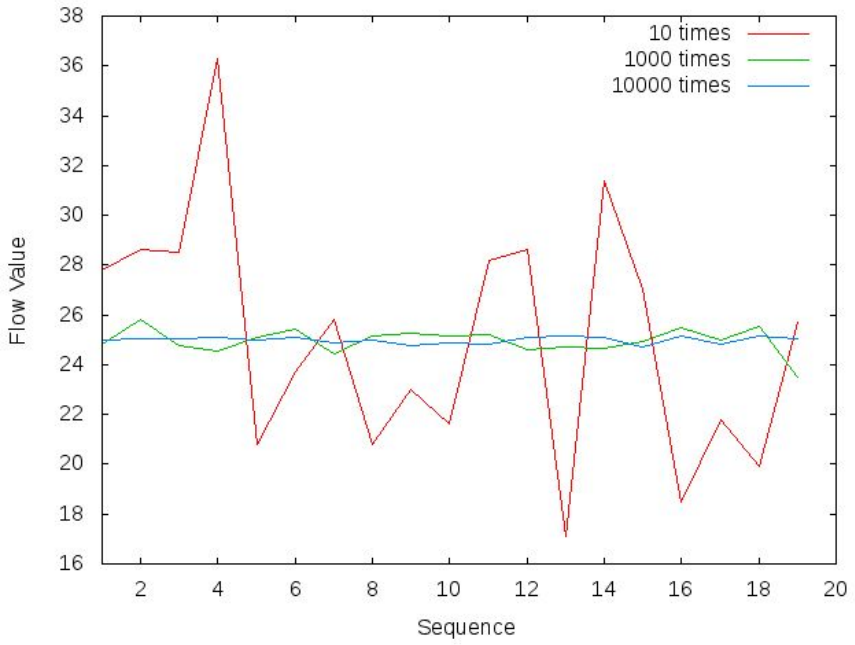

图 5-11　随机流量红包算法 10000 次、1000 次、10 次数据平均值对比折线图

在输入参数 fsll=500，fsgs=20 时，算法的平均流量值为 25。由图 5-11 随机流量红包算法 10000 次、1000 次、10 次数据平均值对比折线图可知，随机流量值在其平均值周围上下波动。当测试次数越多时，多次随机流量红包平均值越接近算法的流量平均值。多次随机流量平均值面额是大致均匀的，10000 次流量红包序列值已经趋近于一条直线。因此，每个用户在多次领取流量后的平均值是相当的，随机流量红包算法是相对比较稳定的。

5.2 日志分析

为了提高服务质量，在公众号服务一段时间后，积累了大量的日志文件，通过日志分析来优化微信公众号的功能。日志分析包括用户日志和系统日志分析。

5.2.1 用户日志分析

通过统计用户使用微信公众号菜单的点击次数来分析用户使用菜单的情况，从而优化公众号菜单功能结构。从公众号的菜单中随机挑选 6 个菜单项，统计某月各菜单每天的点击总量。图 5-12 是公众号 6 个菜单某月点击次数的统计图。

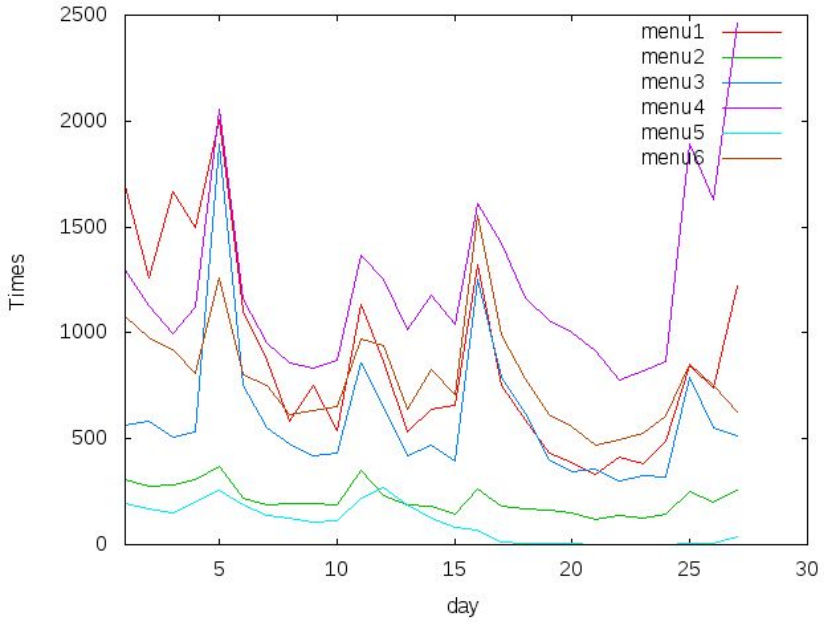

图 5-12　微信公众号某月菜单点击次数统计图

图 5-12 中统计了某月 6 个菜单项每天的用户点击菜单次数，从图中可以看出各菜单用户访问量不同，但是各个曲线走势几乎一致。说明每个菜单的点击比例是相当的。通过计算，菜单 Menu1 点击次数占总点击次数的 29.91%，而 Menu5 只占菜单总点击次数的 4.75%，Menu2 占 2.65%。因此，运营商可以考虑合并菜单 Menu5 和 Menu2 的功能，把空余出来的菜单项用于新的业务。

5.2.2 系统日志分析

系统监控日志包括 CPU 监控日志，网络监控日志。为了监控应用服务器的性能，对 CPU 平均负载、网络请求量进行采样分析。在应用服务器上，每分钟读取一次内核文件/proc/net/dev 来获得网络流量值。通过两次的网络流量值差来记录每分钟的网络上行、下行流量。同时也对 CPU 的平均负载进行采样统计，每 15 分钟对/proc/loadavg 进行一次读取。某月的采样结果如图 5-13 和 5-14 所示。

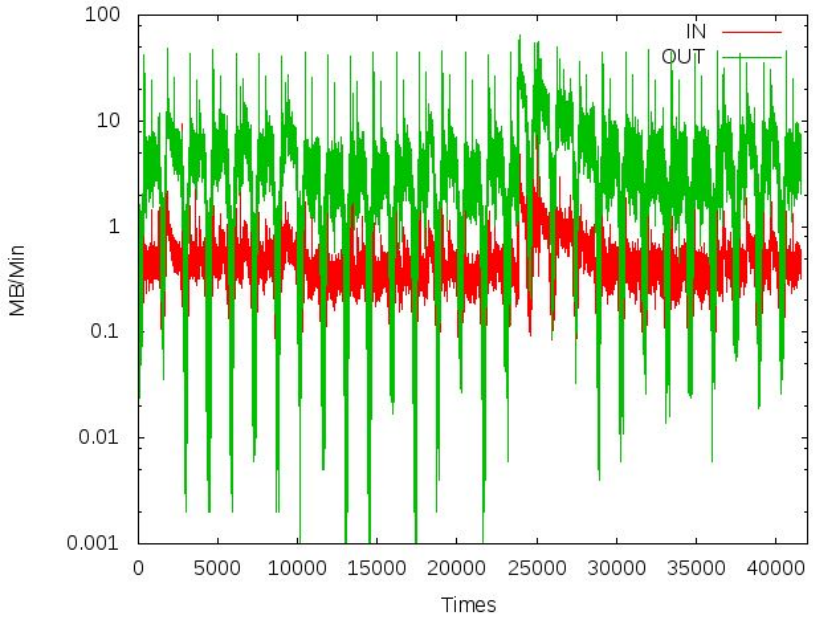

图 5-13　应用服务器某月网络流量图

在图 5-13 中，红色曲线代表接收流量，绿色曲线表示发送流量。接收流量平均值为 0.434MB/Min，最大值为 9.934MB/Min，发送流量平均值为 4.396MB/Min，最大值为 64.094MB/Min。可见，在高并发请求时，网络的吞吐量增大，网络延迟比较严重。

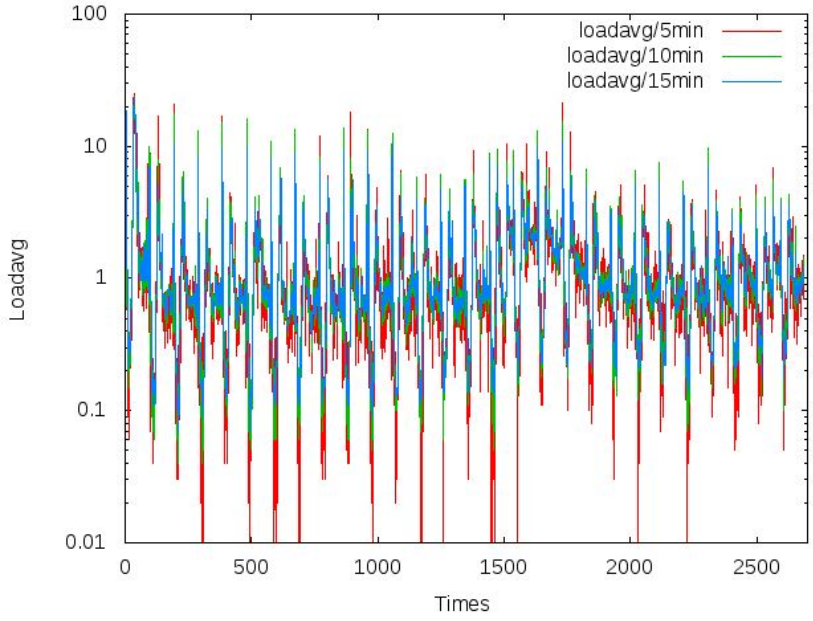

图 5-14　应用服务器某月 CPU 平均负载图

根据图 5-14，应用服务器 CPU 在某月的平均负载为 1.23234，最大负载为 25.75，CPU 平均负载大于 2 的占 13.34%。从以上分析得出，高并发请求时，应

用服务器网络流量波动大，CPU 负载较高，很大程度上影响了应用的服务质量。

5.3 小结

本章对基于微信公众平台的可扩展架构的性能进行了测试，包括节点的可用性、可扩展性测试。对用户日志、系统日志进行了分析，形成闭环优化系统来优化运营商微信公众号的功能。通过以上测试，该架构实现了高并发用户访问量的下的扩展性，同时实现了单点服务器的可用性。最后，通过性能测试对比，可扩展分布式架构的在一定程度上提高服务的性能，保证了服务质量。

第六章 总结与展望

本章对课题中的相关内容进行了总结，归纳出架构的不足方面，针对不足之处提出改进方案，指明接下来的工作内容。

6.1 工作总结

本文主要介绍了基于微信公众平台的可扩展架构的相关内容；首先概述了架构中涉及到的相关技术；详细分析了架构的功能以及性能需求；对架构中的基础架构的功能进行设计、实现，同时对整个架构中的各个节点进行了可用性、可扩展性设计与实现；最后对架构进行测试与分析，并做出相应的总结。对工作内容概括为以下 4 个方面：

① 对基于微信公众平台的可扩展架构的需求分别从通信运营商微信公众平台基础架构和可扩展分布式架构进行了分析。运营商微信公众平台基础架构分别从功能方面进行了需求分析，包括消息处理、业务逻辑等；可扩展分布式架构的需求包括可用性、可扩展性、性能等方面。

② 根据通信运营商微信公众平台基础架构的需求进行功能的设计和实现，包括消息处理、运营商业务接入、客服系统、后台管理系统、业务系统接口封装，设计了数据库中各表的结构，使得基础架构可以提供完整的服务器。

③ 根据可扩展分布式架构需求进行设计与实现。首先讨论了可扩展分布式架构存在的技术问题；然后从负载均衡服务器高可用性、应用服务器水平扩展、数据库的可用性、会话状态一致性 4 个方面进行设计和实现。

④ 对可扩展分布式架构的各节点进行了测试；然后对架构的整体性能进行测试，比较了在高并发请求下分布式架构的优越性；最后对用户日志、系统日志进行了分析。

最后，得出基于微信公众平台的可扩展架构在一定范围内的高并发用户访问量情况下能够提供高可用、可扩展的服务结论，保证了通信运营商微信公众号的服务质量。

6.2 展望

本课题在研究与实现过程中发现以下不足，下面对不足之处加以标注，并且

在后续的工作中改进。

① 分布式架构中负载均衡服务器高可用性是采用备份的方式实现的，备份服务器处于闲置状态，浪费了硬件资源。可以通过两台服务器实现主主模式，使得同一域名解析到两个虚拟 IP 来改进。

② 会话保存是通过 Nginx 的 ip_hash 来把请求转发到相同的服务器，如果有更换服务器，可能会导致部分用户状态数据丢失，可通过共享 Session 来改进。

③ 通过测试发现，分布式架构的性能是以某最大用户量为前提的。当用户量继续增加，需要扩展和优化架构，采用的机制往往更复杂，部署也更加的繁琐。用户可以考虑把应用部署到云服务平台，不用考虑复杂的服务器性能，根据业务需求和策略，自动调整其服务器资源。

参考文献

[1] Chigona, W., Beukes, D., Vally, J., & Tanner, M.Can mobile Internet help alleviate social exclusion in developing countries?.The Electronic Journal of Information Systems in Developing Countries,2009,36,1–16.

[2] 中国互联网络发展状况统计报告（2016 年 1 月)[EB/OL]. http://www.cnnic.cn/.

[3] 覃凯. 微信在企业营销中的利与弊[J]. 电子商务, 2012 (11): 28-29.

[4] 腾讯：2015 年微信平台数据研究报告[EB/OL]. http://tech.qq.com/a/20150127/018482.htm.

[5] 赵敬, 李贝. 微信公众平台发展现状初探[J]. 新闻实践, 2013 (8): 8-10.

[6] 2015 年微信公众号媒体价值研究报告[EB/OL]. http://www.iresearch.com.cn/report/2393.html.

[7] 微信公众平台. https://mp.weixin.qq.com.

[8] 微信公众平台的设计与开发之道[EB/OL]. http://www.infoq.com/cn/articles/wechat-design-dev.

[9] 夏超. 通信运营商微信运营策略[J]. 信息通信技术, 2015, 9(4): 47-51.MLA.

[10] Matsudaira K. Scalable web architecture and distributed systems[J]. The Architecture of Open Source Applications, 2012, 2.

[11] Gwertzman J, Seltzer M I. World Wide Web Cache Consistency[C]//USENIX annual technical conference. 1996: 141-152.

[12] Vinoski S. Rpc and rest: Dilemma, disruption, and displacement[J]. Internet Computing, IEEE, 2008, 12(5): 92-95.

[13] Maxia G. Advanced MySQL replication techniques[J]. Viitattu, 2006, 20: 2010.

[14] Red Hat Enterprise Linux 6.4 Release Notes[EB/OL]. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/6.4_Release_Notes/index.html.

[15] Reese W. Nginx: the high-performance web server and reverse proxy[J]. Linux Journal, 2008, 2008(173): 2.

[16] Kegel D. The C10K problem, 2011[J]. http://www. kegel. com/c10k. Html.

[17] Apache,lighttpd,nginxs[EB/OL].http://www.blogjava.net/daniel-tu/archive/2008/12/29/248883.html.

[18] The uWSGI project. http://uwsgi-docs.readthedocs.org/en/latest/.

[19] MySQL A B. MySQL[J]. 2001.

[20] RabbitMQ. http://www.rabbitmq.com/.

[21] Git. http://git-scm.com/about.

[22] WT'Blog[EB/OL]. http://blog.w2j.xyz/2.

[23] Gitk. https://git-scm.com/docs/gitk.

[24] 邓明琳. 基于面向内容交换的 Linux 集群系统负载均衡策略研究[D]. 重庆大学, 2008.

[25] Apostolopoulos G, Aubespin D, Peris V, et al. Design, implementation and performance of a content-based switch[C]//INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. IEEE, 2000,3: 1117-1126.

[26] Katz E D, Butler M, McGrath R. A scalable HTTP server: The NCSA prototype[J]. Computer

Networks and ISDN systems, 1994, 27(2): 155-164.

[27] Wen-zheng L I. RESEARCH ON NETWORK LOADING BALANCING BASED INTELLIGENT DNS [J][J]. Journal of Beijing Technology and Business University (Natural Science Edition), 2008, 3: 014.

[28] Cardellini V, Colajanni M, Philip S Y. Dynamic load balancing on web-server systems[J]. IEEE Internet computing, 1999, 3(3): 28.

[29] Adelman K A, Kashtan D L, Palter W L, et al. Method and apparatus for a TCP/IP load balancing and failover process in an internet protocol (IP) network clustering system: U.S. Patent 6,078,957[P]. 2000-6-20.

[30] Brendel J, Kring C J, Liu Z, et al. World-wide-web server with delayed resource-binding for resource-based load balancing on a distributed resource multi-node network: U.S. Patent 5,774,660[P]. 1998-6-30.

[31] Kate Matsudaira[EB/OL]. Scalable Web Architecture and Distributed Systems. http://www.aosabook.org/en/distsys.html#.

[32] Venkatesh CM[EB/OL]. 4 Architecture Issues When Scaling Web Applications: Bottlenecks, Database, CPU, IO. http://venkateshcm.com/2014/05/Architecture-Issues-Scaling-Web-Applications/.

[33] 微信公众平台文档. http://mp.weixin.qq.com/wiki/8/f9a0b8382e0b77d87b3bcc1ce6fbc104.html.

[34] Python sha. https://docs.python.org/2/library/sha.html?highlight=sha.

[35] Python base64. https://docs.python.org/2/library/base64.html?highlight=base64#module-base64.

[36] RabbitMQ pika. http://www.rabbitmq.com/tutorials/tutorial-one-python.html.

[37] 骆宗阳, 王澄, 杨宇航. 具有高可用性的负载均衡技术的研究与实现[J]. 计算机工程与应用, 2003, 39(27): 193-195.

[38] Keepalived. http://www.keepalived.org/documentation.html.

[39] Hinden R. Virtual router redundancy protocol (VRRP)[J]. 2004.

[40] Rosen E C, Rekhter Y. BGP/MPLS IP virtual private networks (VPNs)[J]. 2006.

[41] Schwartz B, Zaitsev P, Tkachenko V. High performance MySQL: Optimization, backups, and replication[M]. " O'Reilly Media, Inc.", 2012.

[42] Kristol D M, Montulli L. HTTP state management mechanism[J]. 2000.

[43] Low A, Roychoudhry A, Chin H C. Systems and Methods to Adaptively Load Balance User Sessions to Reduce Energy Consumption: U.S. Patent Application 12/255,197[P]. 2008-10-21.

[44] High Availability Support for NGINX Plus. https://www.nginx.com/resources/admin-guide/nginx-ha-keepalived/.

[45] Replication Statements[EB/OL]. https://dev.mysql.com/doc/refman/5.5/en/change-master-to.html.

[46] SQL Statements for Controlling Master Servers.[EB/OL]. https://dev.mysql.com/doc/refman/5.5/en/replication-master-sql.html.

[47] Haddad I F. Open-source web servers: performance on carrier-class Linux platform[J]. Linux Journal, 2001, 2001(91): 1.

[48] Snell Q O, Mikler A R, Gustafson J L. Netpipe: A network protocol independent perf ormance evaluator[C]//IASTED International Conference on Intelligent Information Man agement and Systems. 1996, 6.

[49] Kargl F, Maier J, Weber M. Protecting web servers from distributed denial of service attacks[C]//Proceedings of the 10th international conference on World Wide Web. AC M, 2001: 514-524.

[50] Mallepally S R. QuickTest Professional (QTP) Interview Questions and Guidelines: A Quick Reference Guide to QuickTest Professional[J]. 2009.

在学期间的研究成果

一、发表论文

1. Zhou Q, Wu J, Wu T, et al. Learning network storage curriculum with experimental case based on embedded systems[J]. Computer Applications in Engineering Education, 2015.
2. Zhou, Q., Zhang, Y., Sun, H., Wu, T., Yang, M., Zhou, R., ... & Li, K. C. (2015, September). The Design and Implementation of Embedded Online Laboratory. In Parallel Processing Workshops (ICPPW), 2015 44th International Conference on (pp. 86-89). IEEE.
3. Li, Y., Sun, H., Yuan, J., Wu, T., Tian, Y., Li, Y., ... & Li, K. C. (2015, September). Accelerating 3D Digital Differential Analyzer Ray Tracing Algorithm on the GPU Using CUDA. In Parallel Processing Workshops (ICPPW), 2015 44th International Conference on (pp. 61-66). IEEE.
4. Analysing and Implementing Data Integrity of iSCSI Network.(审核).
5. 发明专利. 基于 iSCSI 的数据完整性存储系统. 201510196539.3.
6. 发明专利. 一种基于智能终端的老年人桌面系统. 201510019755.0.

二、参与课题

1. 甘肃省省级科技计划项目 基于移动智能终端的云存储系统 项目编号：1204GKCA061 2013.10-2014.01
2. 老年桌面系统开发 2014.04-2014.08
3. 微信公众平台开发 2014.04-2015.10

三、所获奖项

1. 兰州大学 2014-2015 学年一等奖学金
2. 兰州大学 2014-2015 学年“三好研究生”

致　　谢

在这三年的学习期间里，不管是在学习上还是在生活上，许多人都给予了我莫大的帮助，在此向他们表示由衷的感谢。

首先，感谢我的导师周庆国教授。在学习方面，他给我们提供了接触新技术、新知识的渠道，总是在第一时间把专业领域的新产品、新动态推荐给我们。在刚进入实验室时，给迷茫的我们指明了研究方向。在论文定题时给予了很多很好的参考意见，同时在写作时，给予了详细的专业性修改意见。在生活中，周老师平易近人，给予了我们不小的资助。周老师博学多才、对科研精益求精时刻激励着我们，让我受益终生。

然后，感谢周睿副教授，不管是在我的小论文，还是毕业论文中都给予了很多的帮助。特别是在毕业论文中，给予了详细的指导。

还有，感谢 DSLab 实验室的兄弟姐妹们，他们给我带来了欢乐，同时又替我排解忧愁。在学习方面，给了我许多的帮助，从他们身上学到了许多有用的技巧。同时，感谢我的家人、亲朋好友，他们给予我大量的激励、支持、关心，是我坚强的后盾。

此外，感谢在百忙之中对本论文进行评审的专家学者。

最后，再次向所有帮助、关心我的老师同学、亲朋好友表示最由衷的感谢。