密级：　　保密期限：

北京郵電大學

硕士学位论文

题目：　企业移动协同云办公系统
——工作圈的分析与设计

学　号：　2011127136

姓　名：　赵炜

专　业：　软件工程

导　师：　吴国仕

学　院：　软件学院

2016 年 6 月 10 日

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期：_____

关于论文使用授权的说明

本人完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。

本学位论文不属于保密范围，适用本授权书。

本人签名：_____ 日期：_____

导师签名：_____ 日期：_____

企业移动协同云办公系统——工作圈的分析与设计

摘 要

近几年，随着智能终端迅速的普及，移动互联网用户呈现爆炸式的增长。移动互联网服务极大丰富了我们的日常生活。移动社交（微信、微博）、掌上购物（淘宝、京东、天猫）、移动支付（支付宝、微信、ApplePay）等等。但是想要转型移动互联网，企业的管理首先就要跟得上。首先，大部分的企业在企业协同的软件上投入很少，之前使用的大多数是 ERP、CRM 类型的软件，而且软件更新的速度也很慢，大部分的软件是半年或者一年一次更新，两年或更长时间才有一个大版本的发布，这样的更新速度是远远赶不上用户的需求，赶不上业务的发展的；其次，国内高速的经济发展催生了一大批创业者，他们成立的小微企业人数较少但是数量却已经达到千万级别，对于小微企业的协同软件市场国内也是刚刚起步，还很简陋；数据孤岛也是现代企业急需解决的问题，部门与部门之间难以协同，企业上下游数据不同都会阻碍企业的成长。现在的企业协同市场急需符合现在移动互联网浪潮的一款企业协同服务。云服务的出现让人们看到了解决这些问题希望。将企业协同的服务云化、移动化，让企业协同具备一定的社交属性，将极大的加强了企业内、企业与企业之间的协同能力，减低协同的成本；云化能加快软件服务的更新，使得企业协同服务能力的提升速度能够赶上社会的发展。

本文将通过分析现在的企业协同云办公的发展现状及历史，提出一些眼下急需的企业协同办公的需求，给出一个能解决眼下企业协同痛点的解决方案。然后以本人在公司中的一个企业协同云服务产品——工作圈的设计与实现去满足上面分析的需求。工作圈是畅捷通为了满足企业协同办公对云化需求而开发的一款基于移动互联网的企业协同云服务，重点面对的就是国内所有的小微企业管理与协同的市场。工作圈发布两年以来，已经吸引了 30 万家的企业将自己的业务迁入这个云服务。

关键词：企业协同　移动互联网　云服务　工作圈

ANALYSIS AND DESIGN OF TEAMWORK - THE MOBILE COLLABORATION CLOUD OFFICE SYSTEM OF ENTERPRISE

ABSTRACT

In recent years, with the rapid popularization of intelligent terminals, mobile Internet users show explosive growth. Mobile Internet services greatly enrich our daily lives. Mobile social (WeChat, Weibo), handheld shopping (Taobao, Jingdong, Tmall), mobile payment (Alipay, WeChat, ApplePay) and so on. However the realization of transformation of the mobile Internet is not easy, the company's management should keep up. First, most of the enterprises invest few in the enterprise collaboration software, previously used are mostly ERP, CRM types of software, and software update speed is very slow, most of the software is updated once a year or six months, two years or more have a big release, this update rate is far behind the needs of users, behind the development of the business; secondly, high-speed development of the domestic economy spawned a large number of entrepreneurs, they set up a micro enterprises but the number has already reached the million level. The collaboration software market for micro enterprises in the domestic is just beginning, very simple and crude; data silos is also an urgent problem to solve for the modern enterprises. Difficult collaboration between departments and departments, enterprises have different upstream and downstream data, all of these hinder the growth of enterprises. Now the enterprise collaboration market meet the urgent need of a wave of mobile Internet business collaboration services. The emergence of cloud services let people see the hope to solve these problems. Enterprise cloud collaboration services, mobility, make enterprise collaboration with certain social property, it will greatly enhance the Synergy ability between enterprises, and reduce the cost of collaboration; cloud service can accelerate cloud-based software and services update, making enterprise collaboration service capabilities to enhance the speed to catch up with the development of society.

By the way of analyzing the current enterprise cloud collaboration office development status and history, the paper now put forward some much-needed enterprise collaborative office needs, given a moment to solve the pain points of enterprise collaboration solutions. Then use the design and implementation of the Teamwork to meet the needs of the above analysis which is a product of the author’s company - a business collaboration cloud service offerings. Chanjet Teamwork is to meet the need for collaborative cloud services by enterprise coordination office, it a cloud-based mobile Internet service. It focuses on all domestic micro business management and collaboration markets. Since the release of the Teamwork for two years, it has attracted 300,000 business move into the cloud business services.

KEY WORDS： Enterprise collaboration, Mobile Internet, Cloud Services，Teamwork

目 录

第一章 引言 1

1.1 背景 1
1.2 课题任务 2
1.2.1 课题内容 2
1.2.2 本人承担任务 3
1.3 论文结构 3

第二章 关键技术介绍 4

2.1 分布式系统 4
2.1.1 分布式系统的定义 4
2.1.2 CAP 定律 5
2.1.3 现代分布式系统的特点 5
2.2 开源技术 6
2.2.1 Thrift — RPC Framework 7
2.2.2 分布式服务框架 Zookeeper 7
2.2.3 服务器缓存服务 Redis 8
2.2.4 NoSQL 数据库 MongoDB 8
2.2.5 高性能 HTTP 服务器 Nginx 9
2.3 本章小结 9

第三章 工作圈的需求分析 10

3.1 系统用户角色分析 11
3.2 系统功能需求分析 12
3.2.1 用户账户管理与企业管理 12
3.2.2 圈子与帖子管理 13
3.2.3 用户评论管理 14
3.2.4 用户点赞管理 14
3.2.5 工作功能 14
3.2.6 工作流引擎 15
3.2.7 即时通信（IM） 16
3.3 系统的非功能需求分析 16

3.3.1 数据存储持久性 16
3.3.2 数据可迁移性 16
3.3.3 数据私密性 16
3.3.4 服务可用性 16
3.3.5 故障恢复能力 17
3.4 本章小结 17
第四章 工作圈的总体设计 18
4.1 工作圈的设计目标 18
4.2 工作圈功能模块设计 18
4.2.1 功能总体划分 18
4.2.2 用户账户模块 20
4.2.3 圈子与帖子 21
4.2.4 评论 21
4.2.5 赞 22
4.2.6 工作应用 22
4.2.7 工作流引擎 22
4.3 数据库设计 23
4.3.1 用户账户模块数据设计 23
4.3.2 圈子帖子模块数据设计 23
4.3.3 评论模块数据设计 24
4.3.4 赞模块数据设计 25
4.3.5 工作应用模块数据设计 25
4.3.6 工作流引擎模块数据设计 26
4.4 工作圈技术框架设计 27
4.4.1 工作圈整体技术架构 27
4.4.2 工作圈 Rest 设计 28
4.4.3 分布式服务与自动化配置中心 29
4.4.4 数据分片(Data Sharding)与数据索引中心 32
4.4.5 工作圈部署架构 33
4.5 本章小节 34
第五章 工作圈功能模块的设计与实现 35
5.1 客户端界面设计 35
5.1.1 用户账户界面设计 35

5.1.2 圈子与帖子界面设计 36
5.2 服务详细功能设计与实现 38
5.2.1 用户账户模块详细功能设计与实现 38
5.2.2 圈子帖子模块详细功能设计与实现 40
5.2.3 评论模块详细功能设计与实现 42
5.2.4 赞系统模块详细功能设计与实现 43
5.2.5 工作流引擎模块详细功能设计与实现 44
5.3 本章小结 45
第六章 结束语 46
6.1 论文工作总结 46
6.2 问题和展望 46

第一章 引言

1.1 背景

原始社会到工业作坊到现代企业都存在一定的制度和规矩来协同整个组织的运作，这就是所谓的管理（没有规矩不成方圆）；管理的目的是让每一个人发挥所长而不是限制每一个人，而每一个人的特长发挥需要更好的协同。协同在任何组织天然存在：因为人类社会存在专业分工、既有分工必然需要协同。从远古时代的飞鸽传信、烽火台、飞马传书、驿站、官员奏章、朝廷颁令，到近代的书信邮局、电报传真、电话、红头文件、内部会议，再到现代电子邮件、即时通讯、移动手机、视频通讯、微博微信……所有这些都是为了满足更高效实现一个组织目标而发明的协同工具。从现代文明开始，随着电脑科技的发展、企业组织的发达，也就产生了现在的协同办公软件。协同办公软件的发展与互联网的发展基本保持一致，从刚开始有邮件、电子公告板、BBS，到独立网站的出现、到分类搜索网站、到门户网站（新浪、网易）、到全局搜索引擎的出现、到强化个人分享的博客出现、到实时个人分享的社交化微博出现，到目前微信的兴起，社会互联网的发展一直带动着协同办公软件的发展。互联网的发展脉络是从点对点、点对面、面对面、面对点、点对点直到回归以每一个人为核心的应用。

协同办公软件以应用为核心的阶段发展：

第一阶段：文档电子化（1991-1997）

第二阶段：电子邮件的普及（1998-2003）

第三阶段：公文行政办公和内网建设为核心（2003-2008）

第四阶段：工作流程管理为核心（2007-2013）

第五阶段：信息门户阶段（集成为核心）（2012-至今）

可以看到，协同办公软件发展的前几个阶段中是以 PC 为主要的产生工具，而且数据主要的载体也是在 PC 端。从应用的部署来看，不管是 C/S 结构的方式还是 B/S 结构的方式，目前都是以 PC 端为主来部署和应用的。PC 的优点，计算速度快，可视面积大；缺点，不便携，自备电源待机时间短，不适应长时间的外出工作。这样就提出了智能移动终端的需求，我们需要一个可以随身携带，使用和待机时间长而且功能丰富的终端来代替 PC，智能手机无疑就是最佳的选择了。接下来我们以管理的内容来看看协同办公软件的发展。前三个阶段，管理的内容主要有：文档、邮件、公告、内部新闻等一些静态的资源。第四个阶段，提出了一个新概念：工作流程管理。此时企业的内部就有了很多的协同办公软件。员工

为了完成工作可能需要使用不只一种的协同办公软件，相当的繁锁。在第五阶段，为了解决这个问题，提出了信息门户，重点在于集成这之前提供的众多种类的协同办公软件。起到了一点成效，提高了员工的工作效率。在这个集成的门户中，数据孤岛的问题依然没有能很好的解决。现在，越来越多的企业意识到信息流通的重要性，不只在企业的内部打通信息解决孤岛，更想要加强企业上下游之间的沟通，提升企业的决策能力。云服务，成为了一个最优解决的方案。综上，一个基于云服务的企业协同服务方案成为了现在移动互联网行业中的热门方向。

将所有的企业协同业务放在云上的优点有很多，同样带来的挑战也很大。截止2015年下半年，国内的注册企业数已达7000万家，从业人员相比还要番几番。为如此庞大的用户群提供服务，一台或几台高性能服务器的提供服务是远远不够的。目前服务器性能的增长已经到达一个瓶颈，而分布式服务越来越受到大家的重视。基于分布式的思想，将服务部署在多个服务器上，将所有用户的请求以一个固定的算法平均分布到不同的服务器上去处理。由于企业服务的特殊性，大部分员工并不是 7×24 小时不间断的在线使用，所以这个服务还需要具备弹性扩展的能力，不仅能随着用户的增长来增加自己的服务能力，同时又能在空闲时间来收缩服务器的数量达到对节约成本的目的。

本文的核心是通过 Zookeeper 与 Thrift 技术来构建一个具有高性能以及高可用性的企业协同云服务。云服务的构建是一个相对复杂的过程，需要考虑数据一致性、服务高可用性以及分区等特性的取舍以及实现方式。本文在对协同云服务进行设计的同时，采用分布式的策略来进行构建。使用 Zookeeper 与 Thrift 来实现分布式的服务，指出系统的设计难点与瓶颈，给出相应的解决方案。

1.2 课题任务

1.2.1 课题内容

工作圈是一款基于云服务的协同办公软件，它将为畅捷通公司的所有企业客户提供企业协同的云服务，包括：企业组织架构与员工的管理（企业通讯录）、办公 OA 功能（轻应用下的公告，请示、审批、请假、报销、派活、签到等功能）、沟通功能（IM 聊天，群聊等）、企业上下游协同能力（圈子、帖子，赞等）。工作圈有 PC WEB、Android 以及 iOS 平台的软件包。本文重点首先是设计一个满足上述所有功能的并同时为三个平台的软件包提供服务的协同云后台服务，着重于分析需求，设计服务整体框架与模块等。

1.2.2 本人承担任务

本人在项目中承担了对服务整体技术架构的设计与实现，开发了项目技术架构中一些核心的公共组件与功能模块。开发的公共组件有：数据持久化组件(gongzuoquan-mongo)，缓存服务存取组件(gongzuoquan-cache)。开发的核心服务有：全局唯一标识生成的服务(gongzuoquan-idcenter)，分布式数据索引服务(gongzuoquan-idlist)。开发的功能模块有：评论模块(gongzuoquan-comment)，赞模块(gongzuoquan-favorite)和工作流引擎(gongzuoquan-flow)。

1.3 论文结构

本文共分 6 章，内容安排如下：

第一章 引言，主要介绍本课题的意义、主要任务、以及本人在课题中承担的主要任务等。

第二章 相关技术介绍，介绍本课题中使用到的关键技术。

第三章 工作圈的需求分析，主要介绍企业协同办公云服务的发展以及当下的需求。

第四章 工作圈的总体设计，主要介绍该系统的整体框架设计与功能划分。

第五章 工作圈的核心模块的设计与实现，主要介绍该系统中核心模块的设计思路和实现过程。

第六章 结束语，针对本文所设计到的工作进行总结，并对后期的改进和发展做出期望。

第二章 关键技术介绍

本章将针对在该论文中涉及的分布式系统的思想与使用到的核心技术进行介绍。使用到的核心技术主要包括：Apache 的两个顶级开源项目 Zookeeper 与 Thrift，还有 Redis 服务与 Mongo 数据库。这些技术都是在最近十年的移动互联网中，应对上亿用户的高并发而开发的组件。

2.1 分布式系统

分布式系统并不是什么新鲜词，20 世纪 80 年代之前就已经有出现了很多种不同类型的分布式系统。分布式系统真正普及与出彩的却是在互联网时代。谷歌、亚马逊、Facebook 等行业领先的互联网公司将分布式系统推到了一个新的高度。谷歌的 GFS 文件系统、MapReduce 工具都是构建于分布式的理念上，现在的大数据分析正是建立在这两个软件的思想之上。Apache 旗下的 Zookeeper、Spark、Kafka 等分布式系统，简化了构建分布式计算的门槛，让更多企业客户体会到了分布式系统的好处。

2.1.1 分布式系统的定义

分布式系统是一个其硬件或软件组件分布在连网的计算机上，组件之间通过传递消息进行通信和动作协调的系统。这个简单的定义覆盖了所有可有效部署连网计算机的系统。[6]由一个网络连接的计算机可能在空间上的距离不等。它们可能分布在地球上不同的洲，也可能在同一栋楼或同一个房间里。现代定义的分布式系统有如下显著特征：

并发：现代的软件编写中，多线程处理执行并发是很常见的方式。现在的 CPU 核数越来越多，为了能充分利用 CPU 的运算能力，多线程并发技术是一种很好的方式。在计算机网络中也是同样的，不同用户的请求分配到不同的服务器，不同的 CPU 同时进行处理，系统的处理能力就会随着服务器数据增加而提高。

全局时钟：分布式服务之间的协作是通过消息互相传送来进行的。服务与服务之间没有一个准备而有效的全局时钟来保证服务器之间状态的同步。

故障独立性：服务器都会有一定的概率发生故障，系统在设计的时候需要考虑到各种各样的故障情况以及应对策略。网络上的分布式系统是以多个服务器节点构成的，当一个节点故障时，并不代表其它的节点同时也发生了故障，同时正在运行的节点也不会很快的了解到故障发生了。

从现代分布式系统的中（包括 Web 搜索、多人在线游戏和金融交易系统等）可以看出今天推动分布式系统发展的关键趋势：现代网络的泛在特性，移动和无处不在计算的出现，分布式多媒体系统不断增加的重要性，以及把分布式系统看成一种实用系统的趋势。资源共享是构造分布式系统的主要动机。资源可以被服务器管理，由客户访问，或者它们被封装成对象，由其他客户对象访问。

构建分布式系统的挑战是处理其组件的异构性、开放性（允许增加或替换组件）、安全性、可伸缩性（用户的负载或数量增加时能正常运行的能力）、故障处理、组件的并发性、透明性等，同时也要提高系统服务的质量。

2.1.2 CAP 定律

在理论计算机科学中，CAP 定理（CAP theorem），又被称作布鲁尔定理（Brewer's theorem），它指出对于一个分布式计算系统来说，不可能同时满足以下三点：

- 一致性（Consistence）（等同于所有节点访问同一份最新的数据副本）
- 可用性（Availability）（对数据更新具备高可用性）
- 容忍网络分区（Partition tolerance）（以实际效果而言，分区相当于对通信的时限要求。系统如果不能在时限内达成数据一致性，就意味着发生了分区的情况，必须就当前操作在 C 和 A 之间做出选择。）

根据定理，分布式系统只能满足三项中的两项而不可能满足全部三项。理解 CAP 理论的最简单方式是想象两个节点分处分区两侧。允许至少一个节点更新状态会导致数据不一致，即丧失了 C 性质。如果为了保证数据一致性，将分区一侧的节点设置为不可用，那么又丧失了 A 性质。除非两个节点可以互相通信，才能既保证 C 又保证 A，这又会导致丧失 P 性质。[7]

2.1.3 现代分布式系统的特点

1.对服务器硬件要求低

允许使用低配置，低性能的服务器来部署分布式的系统。现代服务器的性能已经是非常高了，但是有限的网络带宽在制约着分布式系统的性能，再快的服务器也需要等网络 IO。因为分布式服务器的故障是不可预见的，所以由一个设计良好的软件架构的容错机制来对分布式系统的可靠性做保证。

2.强调横向可扩展性

通过增加节点数来提升服务性能的方式叫横向扩展（Scale Out），通过升级每个节点的性能来提升服务的性能的方式叫纵向扩展（Scale Up）。现代 CPU 的主频已进到达了 4GHz 的门槛，在纵向扩展的方向上想要提升需要花费巨大的代价。服务器整体的吞吐性能更多的是受制于网络与磁盘，所以模向扩展对系统

的整体性能提升相对要明显的多，而且花费的成本就要低不少。增加服务器的数量要比研发一个高频率的 CPU 要容易。

3.不允许单点失效（No Single Point Failure）

单点失效是指，因为一个服务器节点的宕机导致的整个服务不可用的情况。如果这个服务仅运行了一份实例在一台服务器上，很容易就出来单点失效的情况。

之前已提到过，分布式的服务器允许使用低配置，低性能的机器来部署。那么服务器的可用性也很难保证完全可靠。在设计分布式系统时，将所有的节点会故障的可能都要考虑进去，服务节点要有冗余，数据库服务要有热备份等等。服务通过节点的冗余和数据的冗余来保证它的可靠性。

一般的，服务器的运行状态不应该是满负载的，过高或是长时间的高负载会增加服务器的故障率。为了平衡单机上的负载，可以通过增加大量的节点来实现。这些节点的性能不要求很高，一但节点的基数提升了，在总负载不变的情况下平均到每一台机器上的负载就变低了。

4.较少节点间通讯开销

如上所述，分布式系统的整体性能瓶颈在于内部网络开销。所以尽量让服务调用本地的数据减少网络开销，能显著提高服务的整体性能，Hadoop 的 MapReduce 就是一个很好的例子。

5.无状态的分布式系统服务

应用服务的状态是指运行时程序因为处理服务请求而存在内存的数据。在服务有状态的时候，它就不能满足服务不允许单点失效的要求了。因为有状态的服务一但故障了，保存在服务中的状态数据就会丢失，依赖这个数据的业务就会失败，但且不能被其它的节点代替，这样的服务明显不是高可用型的分布式服务。将数据保存在客户端或是持久化的存储中，将服务无状态化。这样服务就不需要担心数据丢失的情况了，同时也让分布式中的所有节点的行为一致，不再有单点失效的问题。

在设计后台服务的时候，用户的登录状态可以持久化到数据库中，也可以保存到缓存中去，不要保存到分布式服务器的内存中。这样就不会因为分布式服务节点的故障导致用户登录状态丢失的情况。

2.2 开源技术

随着开源社区的兴起，开源成为了计算行业中的一股中间力量。越来越多的优秀技术诞生于开源社区，成长于开源社区，开源社区给了程序以更多的选择。本章选择介绍 Apache 社区的两个顶级项目 Thrift 和 Zookeeper，还有优秀的 Redis 与 MongoDB。

2.2.1 Thrift — RPC Framework

Apache Thrift 是 Facebook 实现的一种高效的、支持多种编程语言的远程服务调用的框架和二进制通讯协议。主要用来定义和创建分布式的远程服务。Thrift 的优点有：相比类似于 SOAP 的协议，基于二进行格式的数据传输更加高效、快速；引入库代码少，没有编码框架，没有配置文件；Thrift 的 IDL 对数据定义与各语言之间的翻译很自然，映射的很好；应用层与序列化层的通讯格式分离，独立修改；支持协议的向下兼容，定义良好的协议是可以向之前的版本兼容的。Thrift 它采用接口描述语言定义并创建服务，支持可扩展的跨语言服务开发，所包含的代码生成引擎可以在多种语言中，如 C++，Java，Python，PHP，Ruby，Erlang，Perl，Haskell，C#，Cocoa，Smalltalk 等创建高效的、无缝的服务。Thrift 还可以使用 Java 的 NIO 模式，极大的提高了服务整体的性能和吞吐能力。[2]

2.2.2 分布式服务框架 Zookeeper

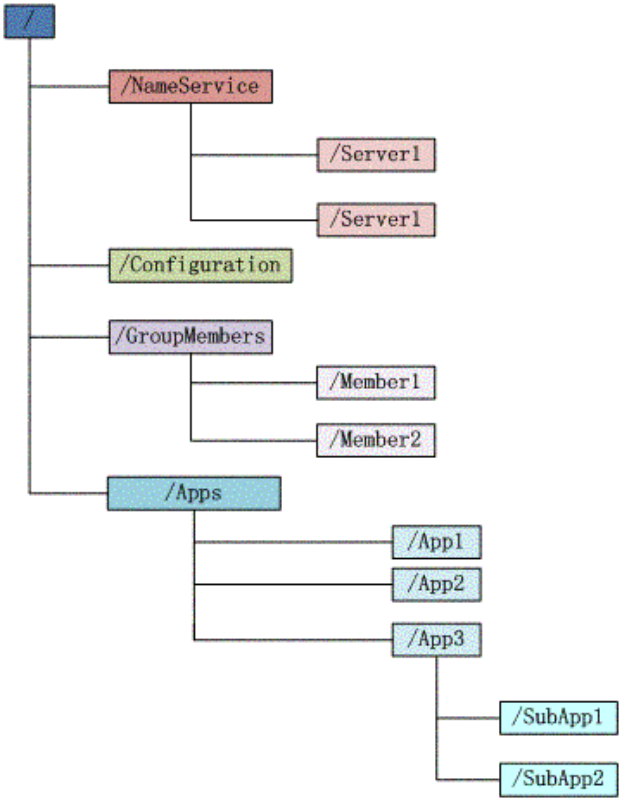


图 2-1 Zookeeper 数据结构[3]

Zookeeper 分布式服务框架是 Apache Hadoop 的一个子项目，主要解决分布式系统中服务命名统一、服务状态同表以及服务配置项管理等问题。

Zookeeper 会维护一个具有树状层次关系的数据结构，它非常类似于一个标准的文件系统，如图 2-1 所示：

Zookeeper 服务的节点是基于路径标识的，每个节点可以有子节点在同一目录下的名称唯一。Zookeeper 提供了对节点变化的监听器，使应用可以随时得到点节点变化的通知。工作圈的环境配置变量就是基于这个监听器来实现的。

2.2.3 服务器缓存服务 Redis

Redis 是一个开源（BSD 许可）的，内存中的数据结构存储系统，它可以用作数据库、缓存和消息中间件。它支持多种类型的数据结构，如字符串（strings），散列（hashes），列表（lists），集合（sets），有序集合（sorted sets）与范围查询，bitmaps，hyperloglogs 和地理空间（geospatial）索引半径查询。Redis 内置了复制（replication），LUA 脚本（Lua scripting），LRU 驱动事件（LRU eviction），事务（transactions）和不同级别的磁盘持久化（persistence），并通过 Redis 哨兵（Sentinel）和自动分区（Cluster）提供高可用性（high availability）。

可以对这些类型执行原子操作，例如：字符串（strings）的 append 命令；散列（hashes）的 hincrby 命令；列表（lists）的 lpush 命令；集合（sets）计算交集 sinter 命令，计算并集 union 命令和计算差集 sdiff 命令；或者在有序集合（sorted sets）里面获取成员的最高排名 zrangebyscore 命令。

为了实现其卓越的性能，Redis 采用运行在内存中的数据集工作方式。根据实际的使用情况，系统可以每隔一定时间将数据集导出到磁盘，或者追加到命令日志中。同时也可以关闭持久化功能，将 Redis 作为一个高效的网络的缓存数据功能使用。Redis 同样支持主从复制（能自动重连和网络断开时自动重新同步），并且第一次同步是快速的非阻塞试的同步。[5]

2.2.4 NoSQL 数据库 MongoDB

MongoDB 是一个开源的文档类型数据库。它具有高性能、高可用性和具备弹性扩展特点。

MongoDB 的文档结构是若干由个 key 和 value 组成键值对构成的。MongoDB 文档类似于 JSON 对象。字段的值可能包括其他文档、数组和数组的文档。在绝大多数的编程语言中，MongoDB 中的文档(即对象)可以对应到本地数据类型。 文档和数组形式的数据组织方式减少非常耗损性能的表的外连接。

MongoDB 提供高性能的数据持久性。对于嵌入式数据模型提供支持，减少了数据库系统的 I / O 活动。MongoDB 索引支持从文档和可以包含密钥和数组更快

的执行查询，支持丰富的查询语言，支持读写操作以及：数据聚合、文本搜索和地理空间查询。

MongoDB 的复制集功能，称为副本集，可以提供：自动故障转移和数据冗余。

MongoDB 提供水平可伸缩性的核心功能：分片——跨一组机器分布数据、标记清楚分片允许数据引导到特定的分片，如考虑地理分布的碎片、支持多种存储引擎。MongoDB 支持多个存储引擎，如：WiredTiger 存储引擎和 MMAPv1 存储引擎。此外，MongoDB 提供可插拔存储引擎 API，允许第三方制定 MongoDB 的存储引擎。

2.2.5 高性能 HTTP 服务器 Nginx

Nginx 是一款开源的高性能 HTTP 服务器，特点是占有内存少、稳定性高、模块丰富、扩展性好。它采用了 Linux 平台的 epoll 事件模型，可支撑大于 2 万以上链接的并发，并且有优异的吞吐能力。工作圈采用 Nignx 来做为其云服务的代理网关实现负载均衡，接受从移动端或是 WEB 端提交的请求并进行分发。

2.3 本章小结

分布式系统是互联网时代下高并发高性能平台构建的主要构建方向。横向可扩展性（Scale Out）使得其构建系统异常灵活，能应对千万级别的用户并发使用，降低了系统对硬件的需求，同时满足了互联网应用的高可用特点。随着开源社区的蓬勃发展，开源的框架也越来越多。在开源的世界里有一句名言：不要重复发明轮子。要充分利用开源的力量来提高我们的工作效率与工作质量。本章所述的组件都是开源社区的优秀成果，本文论述的分布式框架设计，其思想与实现都是与这些开源的组件紧密结合的。

第三章 工作圈的需求分析

在企业中，除了办公自动化系统之外，还有财务、库存、生产、销售、人力等管理系统。由于大量的信息孤岛式的建设，系统之间很少能够紧密协调起来，用户经常需要进行退出一个系统然后再进入另一个系统，并且常常发现数据不一致。因此利用协同办公软件的信息门户与信息协同优势，通过集成的理念去构建岗位工作门户、以及面向对象的 OA 理念开始盛行，同时协同办公软件在移动信息化方面也是首当其冲。并且随着移动互联带来的组织模式的变革，社交化的管理需求开始显露出来，协同办公软件也正在进入自我变革期。作为协同办公软件在当下最为根本的几个特征可以概括如下：基于组织的、基于 WEB 的、基于流程的、基于知识的、基于集成的。协同办公软件的绝大多数用户关注点聚焦在流程管理、知识管理、信息门户、沟通协助和移动应用等方面。现阶段的协同办公软件主流厂商也是多以工作流程为核心，但历史永远是动态发展，目前还有不少用户在关注行政办公、邮件管理和内部信息发布等功能，同时一些走的前面的用户已经十分关注信息门户和业务数据集成、移动办公。更进一步地，部分的厂商已经开始打造以岗位工作目标为核心的信息门户、流程引擎、内容管理、目标任务、移动引擎、融合集成、报表引擎，引导协同办公软件向社交化转型。

协同办公软件绝大多数的用户关心流程管理、知识管理、信息门户和沟通协作等应用，主流协同办公软件也已经形成以工作流程为核心的共识，并且以此为基础，开始引入目标绩效、计划任务、管理报表等与组织管理、日常办公息息相关的内容和方案实现。与此同时，以个人工作门户为标志的平台集成理念，以及依托移动互联的移动办公平台开始引领协同 OA 未来的发展趋势，如移动 APP、微信企业号集成应用等等。所以现代的协同办公服务应该是这样：移动应用为第一媒介；组织战略目标落地为第一要义；流程引擎、内容引擎、集成引擎、报表引擎、建模引擎、消息引擎六大引擎必备；各种应用云端采集部署：降低部署成本，加强智力交换；平台化是大趋势；个性化与标准化的有效统一；更加凸显人为核心的社交化自我管理模式；柔性组织、扩大组织边界；从组织应用为核心到个人发展为核心的模式---任何人都有自己终生唯一的网络办公空间 ID，随时可以和任何企业组织进行对接。

随着人类电脑科技的进步，协同办公软件一定是向着更加还原协同工作自然原貌的方向发展：如果员工想请假、报销、出差，只要对着手机说一下就可以了；如果想获得某些经验，手机上马上有人跳出来告诉我做好这个事情的关键知识点—我可以不用打扰到当前的他，历史的他会来告诉我如何做；如果想召开会议，大家马上可以通过移动设备进行会面交流，系统自动帮助梳理出会议纪要并自动

提醒每一个人要做的事情；任何人所做的事情都可以通过移动设备自动记录下来，并很快反馈到每一个人每一天的工作目标和工作成效中，并告知与整个公司目标的一致性和推进成效；手持设备能够自动记录我对事件的想法和判断，并针对这些想法和判断提供一些信息依据和人员支持。

上述的内容不仅是企业协同的现在的需求，也是未来发展的方向。本章接下来就会以这些点切入，来简述工作圈这样一个基于移动云服务的企业协同软件是怎么去满足这些需求的。

3.1 系统用户角色分析

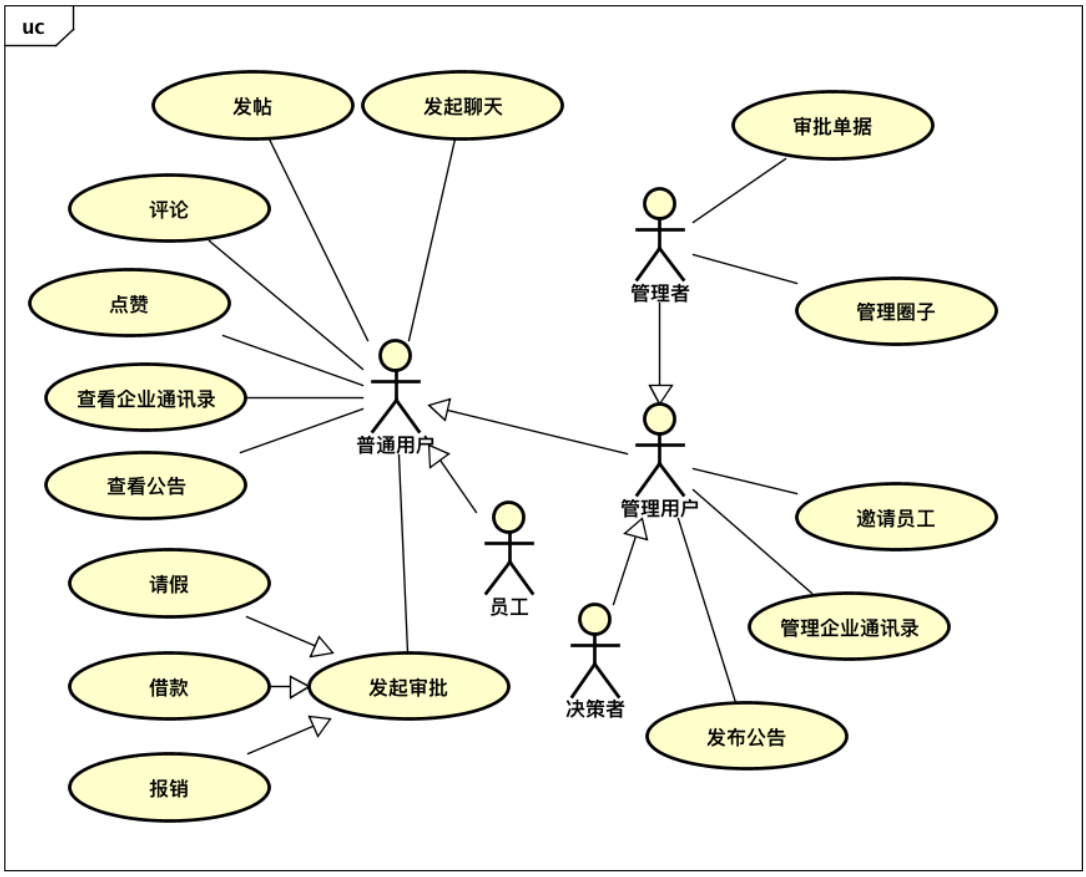


图 3-1 工作圈 UML 用例图

企业内的角色划分，一般分为：决策者、管理者、员工。如果对角色进一步的抽象可以分为：普通用户与管理用户。他们在企业中起到一个什么样的作用，决定了他们在这个系统之中的能力大小。

普通用户，使用工作圈为员工提供的基本的办公协同能力：审批（包括报销、借款、请假等）、查看公告、查看企业通讯录、在圈子中发帖、评论、点赞以及

和他人聊天等协同能力。管理用户在普通用户的基础上扩展了：发布公告、邀请员工、管理企业通讯录等功能。

决策者：一般指的就是企业的领导。他们在企业中仅是很少的一部分人，却是企业的头脑。他们主要的工作就是决定企业发展的方向，决定企业的主要目的与制定企业的管理方式。在工作圈中，他们属于标准的管理用户类型。

管理者：一般是企业员工的管理人员。主要工作是管理员工的日常工作，转达决策者的一些信息。他们不同于决策者，他们需要负责一些很具体的管理工作，所以在角色定义上他们的功能权限要大于一般的管理用户。在工作圈中，对他们的定位是大于管理用户的权限，除了管理用户的功能他们还可以：审批单据、管理圈子等。

员工：就是企业中具本工作的执行者。这个角色代表着企业的绝大多数人。员工主要工作就是完成管理者传达的工作，汇报工作结果。在工作圈中，对他们的定位就是数据的主要生产者。上、下班打卡，每日、每周、每月工作总结、发布工作进度、问题或是解决方案等。

对于大型企业，国有企业来说，他们有完整的 OA 办公系统，同时也有很多的专用服务商为他们提供构件基于他们业务场景下的协同办公系统。也就是说他们使用的系统大多数都是针对他们量身定制的。他们不是工作圈的主要目标。工作圈的主要用户群是一些没有雄厚财力，而且管理流程和主要业务相对简单的中小企业或者是小微企业。小微企业因为人数和规模较小，所以更容易对他们的管理需求和业务需求进行分析和抽象归纳，工作圈的需求定位就会更加的准确，而且适用面也更加地广泛。小微企业的人员较少，对工作圈软件的在企业内部的推广就更加容易。

3.2 系统功能需求分析

3.2.1 用户账户管理与企业管理

工作圈的账户系统是畅捷通 CIA 云平台的子系统，工作圈的用户账户直接使用的 CIA 云平台的用户数据，用户数据与 CIA 通过消息总线保持同步；CIA 云平台下所有子系统注册的账号均可在工作圈直接登录。之所以采用公司的 CIA 云平台用户账户，其一是为了消除数据弧岛，其二是为了工作圈将来的平台化做准备。畅捷通的在企业办公自动化上有很多的优秀的软件和云服务，需要一个协同云服务来帮助它们打通这些数据，而打通数据的第一步就是需要用户模型统一。用户的账户在 CIA 云平台都有一个唯一的 ID 值 userId，可以在不同的系统，不同的业务功能中以 userId 识别用户信息。账户管理模块的主要功能是为工作圈各个

终端提供单点登录、权限校验用户/企业信息查询和变更等功能。并且基于畅捷通云平台的智能平台，为工作圈用户提供智能推荐、搜索服务。

工作圈的企业也是沿用了畅捷通 CIA 云平台的模型，企业的数据同用户一样通过消息总线与 CIA 保持同步。工作圈的企业是以 orgId 这个值作为企业在系统中的唯一标识，同时企业的名称也不能重复。这样可以便于用户在注册进入工作圈的时候，可以直接加入企业。

工作圈企业管理的最主要功能就是企业通讯录。微信就是以手机通讯录为入口，占据了移动互联网时代的入口，这个案例在企业协同市场依然有效，可以说做好了企业的通讯录管理，企业协同能力就完成了一大半。提交审批单的时候，需要企业通讯录才能知道提交给哪个领导，提交报销单，请假单，工作报告也是类似的，开会时发起电话会议需要企业通讯录才能知道参与人的联系方式，更不用提在工作中找人的便利性。最重要的，是挖掘企业与企业之间的协同关系，而目前的协同办公市场上还没有能够提供这样能力的服务。

企业通讯录主要的功能：可以展现企业一个清晰简单的组织结构树；可以通过姓名、电话或是邮件等关键信息搜索公司中同事；可以将同事的电子名片转发给相关的人；可以很方便的和同事通过电话，工作邮件，短信，消息方式进行联系；可以通过工作圈提供的即时通讯功能聊天或是多个人一同发起会话等。企业管理员可以新增，删除企业内的员工，调整企业内的部门组织结构。

3.2.2 圈子与帖子管理

圈子在工作圈中的定位是一组关系相对松散的集合，这个集合是以一个工作上共同的目标为关系来创建的，例如一个项目，一个目标或者是一个团队等。圈子中的角色有圈子管理员与圈子成员。任何人都可以通过圈子内的用户邀请加入圈子。在圈子中，大家可以平等的在圈子中发起话题，这些话题可以是在公司的一些工作成果或是一项讨论等。

帖子承载了圈子中最关键的信息，主要是以文字来构成，辅以图片文件等，同时可以@其他圈子成员关注。帖子可以发表 500 字的文章，可以被评论，可以被点赞。在帖子中可以@圈子中的用户，被@到的用户会收到一条关于这篇帖子的提醒。

圈子的设计思想是一种开放式的功能，对圈子的功能没有一个严格的定义与约束，意味着每家企业，每个员工对这个都可以有不同的使用方式。比如建立一个公司圈，邀请全公司的同事加入，然后领导可以传达信息、员工可以分享成果等等；又比如建立一个客户圈，邀请公司的客户加入，销售就可以很方便的在圈子中推出自己的新产品、或是和客户更加直接的沟通等。个性化非常高的圈子模块，在协同办公中承担着一个不可或缺的作用。下面会介绍工作圈中企业内的办

公应用，它们主要承担着企业的自动化办公与管理的功能，它们的特点就是功能很明确。系统中开发的工作应用的数量毕竟是有限的，总是有一些工作场景不能覆盖，开放式的圈子就很好的为这个做了互补。

圈子中人人平等的特性会为企业带来一种新的管理模型——扁平化管理。在这样的管理模式中领导的指令下达不再是层层下传，而可直达所有的员工。这样的做法有利也有弊：对于小微企业，避免了信息因层层传达而导致的信息缺失；但是对于大型企业，扁平化管理容易造成信息淹没难以分辨有价值的信息。所以圈子内的人员数不是无限增长的。根据公司在小微企业调研的结果，暂时定为一个圈子人数上限不会超过 1000 人。这个设定，在将来工作圈的运行过程中还会根据用户的活跃程度和需求调整。

3.2.3 用户评论管理

评论的功能可以对圈子中的帖子发表评论。评论中也可以上传图片或是附件。评论也可以用来回复他人的评论。当用户评论帖子的时候，发帖人会收到提醒。回复他人的评论时，被回复用户也会收到提醒。评论中可以@圈子中的用户。被@到的用户会收到一条关于这条评论的提醒。

评论的功能看似与帖子相同，然而在工作圈中却把它作为一个单独的服务提供出来是因为其价值需要。在本章的引言中提到了现代的企业协同的一大特点就是社交化，评论则是社交的一个主要的切入点，完成了信息的聚合与传播。在开放式的圈子模块中引入评论可以促进圈子成员之间的交流，同样在工作日报或是报销单之类的其它电子办公的场景中引入评论同样也可以引入评论的机制让更多的用户参与进来。可以看到在企业协同办公的很多场景中都需要评论功能，单独为了每个模块去写评论功能是不合适的。它不仅低效，而且分散了评论的数据，不便于数据的聚合与挖掘。而提供一个独立的服务让所有的模块复用则会提高开发的效率，促进数据的聚合。

3.2.4 用户点赞管理

赞与评论一样，是社交式应用的一个主要的切入点。操作简便而且更加容易吸引用户。同样的，用户的点赞数据也有很多可以挖掘的地方。所以这里也作为一个单独的服务模块提供服务。

3.2.5 工作功能

工作功能是建立在企业通讯录之上，为企业员工用户提供协同办公的服务。可以说工作功能这个模块就是工作圈办公 OA 的入口。为了能覆盖办公场景，工作圈提供的办公应用有：1. 公告，公司内部重大事项的广而告之功能。总是由

公司的管理人员编写，通知给公司的所有员工查看。2. 审批，公司的审批类的办公流程。主要是解决公司自动化，无纸化办公的需求。为了方便员工的办公，工作圈内置了四类常用的审批类型：请示、请假、报销、借款。以及每一类下都有一些常用的审批选项。审批的重要一点，就是审批流程的定义由后面会介绍工作圈的一个智能的、自由的审批流程服务模块——工作流引擎来实现。3. 任务，为了解决对工作进度的跟踪问题。任务的一大特点就是和移动端上的日历提醒相关联，每接到一个限时的任务就自动的在手机的日历上加一个 Deadline 的提醒项。 4. 签到，提供了对公司员工考勤的跟踪。签到可以根据地点划分为外勤签到和内勤签到：内勤签到，方便了员工上下班的加卡，公司设定了办工地点与上下班时间，就可以通过工作圈来跟踪员工的考勤；外勤签到，方便了公司对外勤人员工作的跟踪。签到的主要功能就是利用手机的定位功能与 GIS 信息结合，方便公司对员工的工作情况的掌控。5. 工作报告，提供一个简易的工作日、周、月工作汇总的一个功能。员工填写，领导阅览。6. 文件柜，为公司提供一个办公文件的云存储功能（易美云提供）。7. 邮件，为公司员工提供一个办公邮局，可为入职的员工开通一个基于公司域名下的电子邮箱（由 263 企业邮箱提供）。8. 电话会议，办公电话会议功能，结合企业通讯录来发起一个电话会议（由用友通信公司提供）。9. 企业日报，工作圈每天统计请假、报销以及签到的数据，给公司的领导发送一份工作概要性质的报告。

工作功能的入口，重点在于把控企业协同办公应用的入口。将来的规划可以把企业通讯录的服务能力以 API 的形式开放，提供其他厂商以内部轻应用的形式为工作圈开发一些企业协同办公的功能。上面提到的电话会议与文件柜就是试点应用。积累了足够的用户之后，更进一步的可以以这个入口与企业通讯录为基础来打造畅捷通公司的企业协同办公生态圈。

3.2.6 工作流引擎

工作流管理一直以来都是办公系统中的一个核心业务。不同的业务，不同的企业对于工作流的定义也都不完全一样，从这个角度出发工作流管理的系统曾经也催生了一个很大的业务市场。然而在目前服务云化的背景下，大批量的企业入驻协同办公云服务以后，为一个一个企业单独定制工作流将变的不现实，需要的是一个尽可能通用的服务。于是工作圈设计了一个自由工作流引擎的服务，通过这个服务企业的员工可以自己定制任一流程化业务的处理流程，定制完之后工作圈会记住这个工作流，在下次使用中自动的为用户进行流程的选择。

3.2.7 即时通信（IM）

即时通信服务，作为工作圈企业用户的一个实时通道。可以进行用户之间点对点聊天，群聊、可以向用户发送文字、语音、表情、图片及文件等。工作圈的@人与评论帖子的通知也是通过即时通讯完成的。即时通信服务还可以为工作功能提供相关的消息推送服务，比如发布公告，提交审批等等。

3.3 系统的非功能需求分析

3.3.1 数据存储持久性

定义：承诺在合同期内数据保存不丢的概率，即每月完好数据/_(_每月完好数据+_每月丢失数据)_。

工作圈的数据库设置了 1 个主节点、1 个从节点和 1 个仲裁节点以实现线上的热备份以及故障无缝切换，同时也会对整个数据库进行周期性的全库的备份。这两个备份策略保证了工作圈的数据持久性不低于 99.999999%。数据持久性按服务周期统计，一个服务周期为一个自然月，如不满一个月不计算为一个服务周期。按工作圈的数据持久性计算，如用户在工作圈存储一百亿条工作或是圈子相关的业务数据，每月最多只有 1 条业务数据发生数据丢失的可能性。

3.3.2 数据可迁移性

承诺用户能够控制数据或主机镜像的迁移，保证启用或弃用该云服务时，数据能迁入和迁出。用户可以通过 API，SDK 等工具对存储在工作圈上的数据进行读写操作，并根据需要进行迁移。工作圈不会对用户上传的数据做任何的修改。

3.3.3 数据私密性

承诺用户应有加密或隔离等手段保证同一资源池用户数据互不可见，并且在用户授权的情况下，云服务商才能获得数据。用户存储在工作圈上的数据，在未经用户合法授权的情况下，其他用户无法访问其数据。工作圈为从访问接口上进行权限控制和隔离，保障用户数据的私密性。

3.3.4 服务可用性

承诺用户业务可用性为合同期内每月单个用户云服务业务可用时间的概率，即每月实际可用时间/每月(实际可用时间+不可用时间)。

工作圈的服务可用性不低于 99.99%。可用性按服务周期统计，一个服务周期为一个自然月，如不满一个月不计算为一个服务周期。工作圈所提供的服务在连续的 5 分钟或更长时间不可使用方计为不可用时间，不可使用的服务时间低于

5 分钟的，不计入不可用时间。工作圈不可用时间不包括日常系统维护时间、由用户原因、第三方原因或不可抗力导致的不可用时间。

工作圈服务可用性的计算方法如下：

月总请求次数低于 1 万的用户不做统计；

失败请求数 = 返回错误 5xx 的请求数量；

估算的失败请求数 = 前 7 天用户单位时间请求数的平均值 × 服务不可用时间 （当出现服务不可用且无失败请求返回时）；

工作圈的可用性 =（失败请求数+估算的失败请求数）/总请求数（正常请求数+失败请求数+估算的失败请求数）。

3.3.5 故障恢复能力

告知用户如出现故障时，故障恢复的能力。工作圈为付费用户的云服务提供 7×24 小时的运行维护，并以在线工单和电话报障等方式提供技术支持，具备完善的故障监控、自动告警、快速定位、快速恢复等一系列故障应急响应机制。

3.4 本章小结

本章通过分析企业协同办公发展的历史以及现代企业协同云的特点及诉求，来为企业协同云服务——工作圈定义需求。在本章揭示了企业协同行业的现状与移动互联网的发展，指出现代化的企业协同需要结合这两者才能满足现代企业对协同管理的需求。同时分析了即时通信，社交化办公的场景给出了如何将两者结合的方向。给出了工作圈的模块需求，并且分析了这些模块需求在企业协同中起到的作用与价值，还提供了一些参考的数值。在 3.3 小节中给出了现代云服务的一些非功能性质的要求。下面就会结合之前的开源技术与分布式的思想来进行这个项目的设计，以期能实现本章中的各个需求。

第四章 工作圈的总体设计

结合前两章的关键技术介绍与需求分析，本章将开始进行工作圈的总体设计阶段。这一阶段的主要目标是在满足功能性需求与非功能性需求的基础上完成云服务的整体架构设计。本章的主要内容，首先完成对企业协同办公云服务的业务设计，接下来针对服务的性能需求进行分布式框架的设计。

4.1 工作圈的设计目标

之前提到了工作圈的针对的用户市场是中小型企业与小微企业。那么在国内这个市场有多大呢？根据工伤总局 2014 年发布的《全国小型微型企业发展情况报告(摘要)》：截至 2013 年年底，全国各类企业总数为 1527.84 万户。其中小型微型企业 1169.87 万户，占到企业总数的 76.57%。将 4436.29 万户个体工商户纳入统计后，小型微型企业所占比重达到 94.15%。[6]可以看出工作圈面对的是一个有千万级企业，上亿用户的这么一个市场。所以，对于工作圈服务的规划已经不能再像以前的企业门户那样只考虑几千人的用户去设计实现了，应该朝着移动互联网服务的方向进行设计。在第二章的时候，提到了分布式的概念和技术，就是近几年解决移动互联网并发问题的方案，现在同样也适用于工作圈。结合分布式几个显著特性开始设计工作圈整体的架构。结合分布式几个显著特性开始设计工作圈整体的架构。

4.2 工作圈功能模块设计

4.2.1 功能总体划分

模块划分是分布式服务设计重要的一环。低成本，横向扩展是分布式服务的一个主要特点。从需求分析，可以预见的是工作圈服务的用户账户、即时通信这两个相关的服务访问量会比其他的功能要大一些。为了能做到性能可以弹性扩展，就是在有个别服务的请求量比较大的时候对，为这个服务动态增加计算资源，需要对整个服务的功能合理的划分模块服务。对服务进行模块划分，有助于对业务进行更清晰的梳理。同时在软件设计的层面上，好的模块划分更有利于程序员架构师对项目的维护与扩展，减少开发的代码量。每个模块专注与自己的功能实现，如果有对于其他模块的依赖，则通过标准的 RPC API 调用，这符合软件工程中的高内聚，低耦合的要求。工作圈的功能总体功能模块如图 4-1。

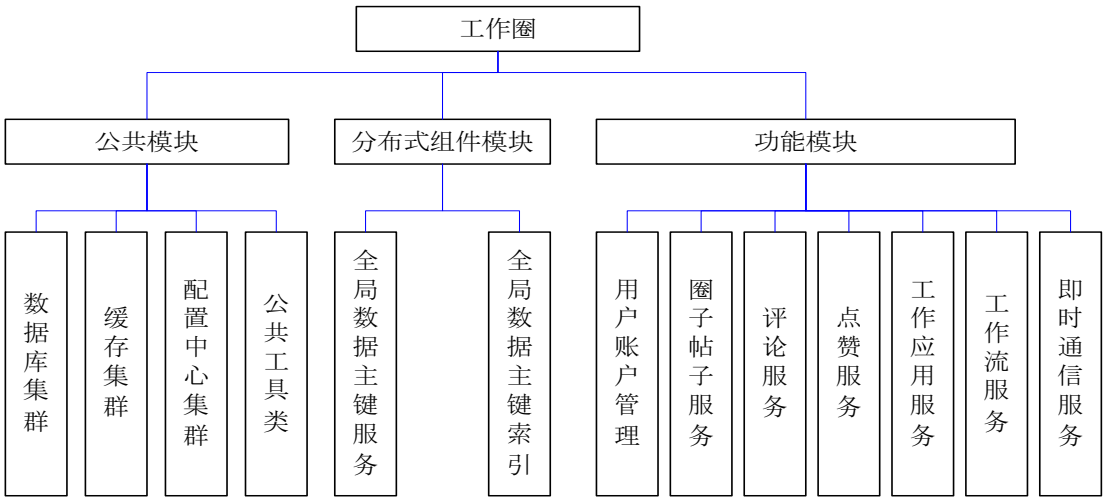


图 4-1 工作圈功能模块图

1. 功能模块划分

根据第三章的功能需求分析，可以进行功能模块划分。根据上面的需求功划分能得到：用户、圈子（帖子）、评论、赞、IM 消息、工作功能等模块服务。因为帖子也是圈子的一个成员属性故划到圈子服务中。而评论和赞的服务，可以通用化用在其它的一些业务上，比如用在工作的审批中。所以作为一个基础的服务可以独立出来。在进行公共模块划分设计需要考虑的要点：1.高内聚，无耦合。模块之前要相互独立，仅数据耦合。2.提取功能相同的代码到一个公共服务包进封装。减少开发代码量。3.模块服务要是无状态的，便于模块服务的模向扩容，同时也为 restful 服务打基础。

对工作圈的功能模块划分如下：

gongzuoquan-account，用户账户服务

gongzuoquan-quan，圈子、帖子服务

gongzuoquan-comment，评论服务

gongzuoquan-favorite，赞服务

gongzuoquan-app，工作应用服务

gongzuoquan-imp，即时通信服务

2. 公共模块划分

在工作圈的规划里，数据持久存储使用 MongoDB 服务，数据内存缓存使用 Redis 服务。所有的业务模块，基本都需要操作数据库与缓存。将 MongoDB 数据库与 Redis 缓存的连接与操作服务封装成公共模块的服务，开发时要求开发人员统一使用这些公共服务完成对数据持久与缓存的操作。这样不仅可以减少在数据持久层上的代码开发量，同时也可以应用链接池技术提升服务的吞吐性能，预防因为人为的疏忽导致的链接泄漏。

工作圈的服务层承担了业务运算的主要工作，这一层服务对系统运算资源敏感，所以这一层的主要考虑就是动态横向扩展能力。这里先不讨论如何去实现这个横向扩展能力，但是需要指出为了实现这个横向扩展能力，需要为每一个模块服务约定一个服务的配置规范，符合这个约定的服务就能实现动态横向扩展能力。将这个公共模块定义为：gongzuoquan-configcenter。

在开发服务的过程中，工程师需要很多的重复的工具类型的功能：比如时间格式化，数据加密，数据计算 MD5 摘要、字符串判空等等的功能，为了每一个不同的模块服务去编写一份工具是不合适的。将所有被重复使用的工具封装到 gongzuoquan-util 中，工程师在开发的时候直接复用即可。最终公共模块划分如下：

gongzuoquan-mongo，提供数据库接入操作的封装。

gongzuoquan-cache，提供对缓存服务的接入操作。

gongzuoquan-configcenter，提供自动化配置中的管理操作。

gongzuoquan-util，提供一些公共的工具类方法

公共模块不同于功能模块，它们不应该成为一个独立的服务，而是成为其它服务所依赖的低层插件。

3. 分布式组件模块服务

工作圈的分布式组件服务是对分布式服务与数据存储提供抽象化服务的模块。在分布式系统之中，数据的流转处理是依赖于一个全局唯一的主键 ID，需要一个 gongzuoquan-idcenter 的服务来为其它的服务生成一个全局唯一的 ID。后面会提到，为了提高存储服务的性能，工作圈会为数据库进行分片，同时在每一个数据库分片中会对其中的数据集合也进行分片。对数据的分片结果就是查询困难，需要一个数据分片的索引中心来维护所有的数据分片。

工作圈的分布式组件模块服务划分如下：

gongzuoquan-idcenter，提供全局主键生成服务。

gongzuoquan-idlist，提供数据主键索引中心。

4.2.2 用户账户模块

工作圈的云服务是畅捷通公司几个云服务中的一个，为了能够在不同的云服务之间打通数据，打通用户信息是关键的第一步。打通用户信息，最简单而有效的做法就是在一开始设计的时候，就沿用同一个用户账户模型，工作圈正是这么做的。工作圈直接使用了公司统一的用户账户模型与数据，所有的用户信息以公司的用户中心服务（以下简称 CIA）为基准。企业数据与用户数据相同，都由公司的 CIA 系统来维护。

因为工作圈的用户体量较大，当在线用户数过多的时候会为CIA带来很大的压力。从另一个角度来讲，当CIA的压力变大的时候可能用户查询的性能会变得很慢，就会反过来影响工作圈的用户体验。所以这里需要采用一个本地缓存的策略，将工作圈所需要的用户和企业的数据从 CIA 同步到工作圈的云上，然后与 CIA 通过一个订阅&消费的模式订阅工作圈用户的数据更新，来达到数据同步的目的。更新的方式如下：

1. CIA上的用户/企业中有一个字段的数据被更新。

2. CIA 将被更新的数据标识（ID），与更新的操作合在一起，发布一条更新消息。

3. 工作圈或是其它的云平台会收到更新的消息，在自己的缓存数据中定位然后更新指定的数据。

4.2.3 圈子与帖子

圈子和帖子之所以设计在一个服务模块里，是因为它们是相辅相成的两个功能。帖子不能离开圈子单独存在，圈子离开了帖子也失去了意义，它们没有必要单独成为一个服务，而且合并之后更为方便开发与维护。

圈子与用户是多对多的关系模型，一个圈子可以有多个用户，同样一个用户也可以有多个圈子。这个关系模型很重要，这个关系是维系着圈子业务的重要一环。简单来讲，用户首先需要加入一个圈子，才可以在圈子中发帖、发评论、为他人点赞等。同样的，当一个用户发帖了之后，另一个用户可不可以查看、评论、点赞也需要根据这个关系来确定。所以对圈子的数据设计应该包含两部分，一个是圈子本身的实体数据，另一个是圈子与用户的关系数据。

帖子是圈子信息流的主要组成部分，帖子与圈子的关系是多对一的关系模型。一个帖子只能从属一个圈子。帖子与用户之前的关系是通过圈子来维护的，所以这里设计的时候只需要考虑帖子的数据模型还有帖子与圈子之前的关系就好。

4.2.4 评论

评论，作为一个单独的服务模块提出来实现，是因为其通用性。帖子中需要评论的功能，在工作应用中同样可以引入评论功能。所以在评论的设计中，不能为每一个业务去维护一个评论与之相关的关系，例如帖子的评论列表或者是审批的评论列表等。可以在这里做一个抽象的定义：“任意一个引入评论功能的业务都视其为一种信息资源，评论是建立这个信息资源之上对其评论的一个信息列表。”有了这么一个定义之后，可以看出不管是帖子，还是审批，还是员工提交的工作报告，在评论或者用评论功能批阅的时候，用户仅仅需要通过这个帖子或者是审批的数据就能发表对其的评论。这里有一个隐含的约束条件，在评论不关

注资源类型的情况下，评论引用的资源其 ID 应该是在云平台全局唯一的。如果不是这样，就有可能出现评论的脏数据。

4.2.5 赞

赞的设计需要注意的要点基本等同于评论。对使用赞功能的业务进行抽象，注意引用的资源 ID 全局唯一。赞唯一的特点就是有一个最近点赞人的头像列表。这个功能的实现相对简单，维护一个最近点赞人的列表以点赞时间倒序排序即可。需要注意的是，这个列表是以资源 ID 为关键词来维护，不是以赞的 ID 维护。即这个数据不能保存在某一条赞当中。

4.2.6 工作应用

工作应用功能是工作圈中比较重要的一个模块，这里从一开始的设计就是为了平台化而进行了。所以这里不针对每一个应用介绍，而是从平台化的角度出发来介绍工作应用平台机制。

现代的移动智能终端为我们带来了更加丰富的信息展现方式，然而不同的移动终端的依托的底层系统不同，开发的语言也不尽相同。但是它们都支持一个语言：JavaScript。HTML5（以下简称 H5）标准的出现更大大增强了这门语言的表达能力。所以 H5+Javascript 成为了平台开放的最佳选择。确定了开放平台的技术方向之后，就要定义开放平台的能力了。开放能力首要任务是打通本地应用与 H5 应用之间的功能接口与数据接口，工作圈 Android 和 iOS 终端采用了开源的 Cordova 框架来实现的接口打通，这里涉及各终端的设计与实现不在本文的讨论范围。

4.2.7 工作流引擎

工作流，一个简单的例子就是审批：由员工 A 发起一项审批单，提交给员工 B 审批；员工 B 审批通过之后在提交员工 C 来审批；假设员工 C 审批完之后就结束了。那么员工 A 就是发起人，员工 B、C 为审批人。而审批的过程就是一个单据流转的过程，到达最后一个人的时候审批完成，归档。

自由工作流，顾名思义就是每一个流程的定义不是一直不变的，而是在使用过程中根据实际的变化来调整的。这么设计的理由一个是因为工作圈面向的小微企业数量比较庞大，不可能为每一个企业定义工作流；另一个原因就因为小微企业的企业结构变动比较频繁，传统以角色定义的模型并不适应，同样的工作圈中企业的模型对员工的角色管理也并不突出。自由工作流的特点就是单据的审批人并不是特定的角色，而是指定的人。那么对于单据的提交人来说，他需要在提交单据的时候指定至少一级的审批人。再有多个审批人的情况下，审批人之间是有

先后次序的，这些也需要在选择审批人的时候指定。这样工作圈就不需要为每一个企业的每一个工作流去指定一个审批人，而是为企业的员工记住他们使用过的审批流即可，在他们下次发起流程的时候能够直接复用上一次的审批流。

4.3 数据库设计

工作圈的后台设计使用的 NoSQL 型的数据库 MongoDB，有别于传统的关系型数据库，NoSQL 的数据库对软件的数据模型设计带来的转变是非常之大的。传统的关系型数据库设计的要点是表之间关系外键的连接，也就是表与表之间的关系是通过关系外键的连接来维护的。系统在做高并发的查询时，外键连接对数据库的性能损耗是非常可观的。同时在扩展需求的时候，表结构会经常变动，程序底层的数据模型也不易扩展。以上的两点未必是关系型数据库的缺点，但是在面对互联网的海量用户并发访问与快速迭代的开发需求，这两点特性并不友好。MongoDB 作为一个 NoSQL 中的优秀产品，解决了这两个痛点。

工作圈选用 MongoDB 的原因有以下几点：

1. BSON 文档式的数据结构，形式上于 JSON 基本一致。因为工作圈的 REST 服务接口统一使用 JSON 数据格式，使用 MongoDB 就更容易理解与分析数据模型，并设计数据结构。

2. BSON 文档的数据易于扩展，不需要 DDL 语句就可以在集合中为数据增加新的属性。

3. NoSQL 数据库没有外键的概念，没有关系的概念。进行查询的时候，响应速度快。

4. MongoDB 有成熟的主从复制集设计和成熟的集群方案，符合现代互联网对数据持久化的高可用与高性能的要求。

4.3.1 用户账户模块数据设计

工作圈的用户账户与企业数据使用的是 CIA 平台的用户模型，并不是自行设计的数据。平台上用户的唯一标识（userId）、企业的唯一标识(orgId)、部门的唯一标识（deptId），这三个标识是工作圈业务模块设计的重要数据。

4.3.2 圈子帖子模块数据设计

圈子的数据设计如下：

{ "_id" : "1953154308116480928", "name" : "Chanjet 圈子", "logo" : "teamlogo.jpg", "createdby" : "60000271637", "createtime" : "1408348065298", "introduction" : "chanjet 圈子",

"invitelevel" : 0, "status" : 0, "masterid" : "60000271637", "qrcodecontent" : "mYnEnq", "qrcodeurl" : "1.jpg" }

圈子的数据有：唯一标识(_id)，圈子名称（name），圈子标志（logo），创建人（createBy），创建时间（createtime），介绍（introduction），状态（status），圈主（masterid），邀请码（qrcodecontent），邀请二维码图片（qrcodeurl）。

工作圈中的唯一标识都是由gongzuoquan-idcenter模块提供的Long类型全局唯一标识，工作圈所有的模块数据都要遵从这一规范。圈子创建的时候，创建人就是圈主，但是圈主是可以移交的，当前圈主不一定是圈子的创建者，所以这里要分开来存。圈子的状态，是表示这个圈子是在用还是解散的状态。工作圈的数据不会物理删除，都是通过类似 status 的字段来表示逻辑删除的状态。邀请二维码图片是通过邀请码生成的，方便用户通过扫码加入这个圈子。

帖子的数据结构设计如下：

{ "_id" : "1954857940643845248" "teamid" : "1952280928296967558", "userid" : "60000004225", "createtime" : "1399517463300", "updatetime" : "1399525936223", "content" : "{@abc,60000001389} 关注一下 IE10 下页面。。目前 IE8 下很多页面的如 footer 等走掉了。。{!lh}{!lh}", "devicetype" : "Web", "location" : "", "latitude" : "", "longitude" : "", "deleted" : false }

帖子的数据有：唯一标识(_id)，发布的圈子(teamid)，发布用户(userid)，创建时间（createtime），更新时间（updatetime），内容（content），设备类型(devicetype)，位置(location)，经度(longitude)，纬度(latitude)。

_id 是全局唯一标识，不多说了。帖子的特点就是根据用户需要可以分享发帖位置，通过 GPS 插件获取手机所在的经纬度，通过地图类 API 查询手机当前的位置并保存。帖子的内容中有几段特殊的文本：{@abc,60000001389}；{!lh}。这两个特殊的文本是工作圈的两种功能性文本：阱一个是@一个名为 abc 的用户，这样用户就能收到一条关于这个帖子的特殊信息；后一个就是工作圈的 emoji 表情了，会在端上解析成一个流汗的 emoji 表情符号。

4.3.3 评论模块数据设计

评论的数据结构设计如下：

{ "_id" : "1954887612579840437", "resourcetype" : 1, "resourceid" : "1952025458524167913", "userid" : "60000000923", "content" : "分几种。。有一种主动回收。。{!zy}", "createtime" : "1402030079657", "replytocomment" : "1952886687211523175",

"replytouser" : "60000000944", "devicetype" : 3, "userip" : null, "status" : 0, "updatetime" : "1402030079657" }

评论的数据有：唯一标识(_id)，被引用的资源（resourceid），发布用户（userid)，创建时间(createtime)，更新时间(updatetime)，内容(content)，回复的用户（replytouser），回复的评论（replytocomment），设备类型（devicetype），IP 地址（userip），状态（status）。

评论是以一个通用开放式的模块来设计的，不仅仅是帖子中可以用到，在其它的功能中也可以使用。将任意一个使用评论的对象抽象为一个 resource，在评论数据里记录 resourceid，就可以表示这个评论是针对这个资源的。回复的评论，这个字段是表示这个评论是回复了另一条评论的评论，也就是现在帖吧常用的盖楼模式。

4.3.4 赞模块数据设计

赞的数据结构设计如下：

{ "_id" : "gzq:favorite:60000004929", "resourceType" : 2, "resourceId" : "231577403934248961", "userid" : "60000004929", "createtime" : "1434526529085", "status" : 0 }

赞的设计同评论差不多，都是通用开放式的服务，抽象了 resource 的概念。这里不再赘述

4.3.5 工作应用模块数据设计

工作应用模块比较特殊，工作圈的工作应用正在走平台化的路线，既有自己研发的应用，比如：审批、签到、工作报告、公告等；又接入了一些第三方的应用：文件柜、电话会议、邮件等，成为了一个开放的应用平台。这里仅介绍一个与工作流引擎结合紧密的核心应用，审批的数据设计。

{ "_id" : "1954908683059207328", "userId" : "60000000930", "orgId" : "90000000821", "departId" : 0, "title" : "test", "content" : "test", "deviceType" : "IPH", "ip" : "118.194.195.5", "createTime" : "1414377546089", "longitude" : 0, "latitude" : 0, "location" : “beijing”, "updateTime" : "1415258230076", "auditId" : "60000001040", "status" : 2, “type”: 2}

审批的数据有：唯一标识(_id)，发起用户（userid），企业（orgId），部门（deptId），标题（title），内容（content），创建时间（createtime），更新时间（updatetime），设备类型（deviceType），IP 地址（userip），位置(location)，经度(longitude)，纬度(latitude)，当前审批人(auditId)，状态（status），审批类型（type）。

审批是一类应用（请示、请假、报销、借款）应用的抽象定义，不同类型的单据通过审批类型这个字段来区分。上面的设计仅能满足基本的请示的需求，有请示的内容（content）。为了满足其它三个业务的需要，还需要在这之上扩展不同的字段：

请假："leaveSheet" : { "beginTime" : "1430976600000", "endTime" : "1430992800000", "sumTime" : 270, "reasons" : "测试测试"}。请假数据有：请假起始时间（beginTime），结束时间（endTime），请假时长（sumTime）/单位：分钟，请假事由：（reasons）

报销："reimburseList" : [{ "reimburseType" : 5, "amount" : "600.00", "remark" : "报销" }], "totalAmountForReimburses" : "600.00"。报销单列表的数据：报销类型（reimburseType），报销金额（amount），备注（remark）。报销总额（totalAmountForReimburses）。

借款："loan" : { "reasons" : "路兔兔借款", "amount" : "2500.00", "createTime" : "1430717854497" }。借款数据：借款金额（amount），借款事由（reasons）。借款时间（createTime）。

可以看出不同的审批业务，所需要的数据也略有不同，同时这几个类型都是互斥的，在审批的数据中以上四种类型的扩展同时只能使用一类。在关系型数据库中必然要冗余很多的字段或者拆分多张表设计才能满足需要，MongoDB 的 BSON 文档数据却能随意的为这个数据扩展相关的属性，相对于关系型数据库设计和扩展功能都更加方便。

4.3.6 工作流引擎模块数据设计

工作流引擎数据结构由工作流实例(FlowInstance)和任务(Task)列表两部分组成，先来看工作流实例设计：

{ "_id" : "1954286430986249633", "resId" : "1953272588472329523", "resType" : 3, "startUserId" : "60000005530", "status" : 0, "createTime" : "1414464172572", "orgId" : "90000001631" }

任务设计：

{ "_id" : "1952454119239682188", "flowInstanceId" : "1953453617233922043", "result" : 2, "handleDesp" : "", "createTime" : "1429692185803", "handleTime" : "1429692218996", "commentId" : 0, "taskUserId" : "60000710782", "orgId" : "60000710783" }

工作流实例数据：唯一标识(_id)，被引用的资源（resId），被引用的资源类型（resType），发起用户（startUserId），创建时间（createtime），企业

（orgId），状态（status）。任务数据：唯一标识(_id)，引用的工作流实例（flowInstanceId),处理结果(result),创建时间(createtime),企业(orgId)，处理人（taskUserId），处理时间（handleTime），评论（commentId）。

一个工作流实例对应一个审批单据，在自由工作流设计中，第一次会让用户指定审批人顺序，然后根据这个顺序创建任务列表并推送给相关的审批人处理。处理的结果与意见通过评论模块生成了一条特殊的评论，展现在这个审批单上。

4.4 工作圈技术框架设计

4.4.1 工作圈整体技术架构

工作圈分布式技术架构的核心就是应用注册中心，所有的服务层都会围绕这个注册中心来设计。工作圈的整体结构是一个标准的 MVC 的三层架构，View 对应工作圈的 Rest 层服务；Control 对应的工作圈的 Server 层服务；Model 对应着数据库与缓存。如图 4-2

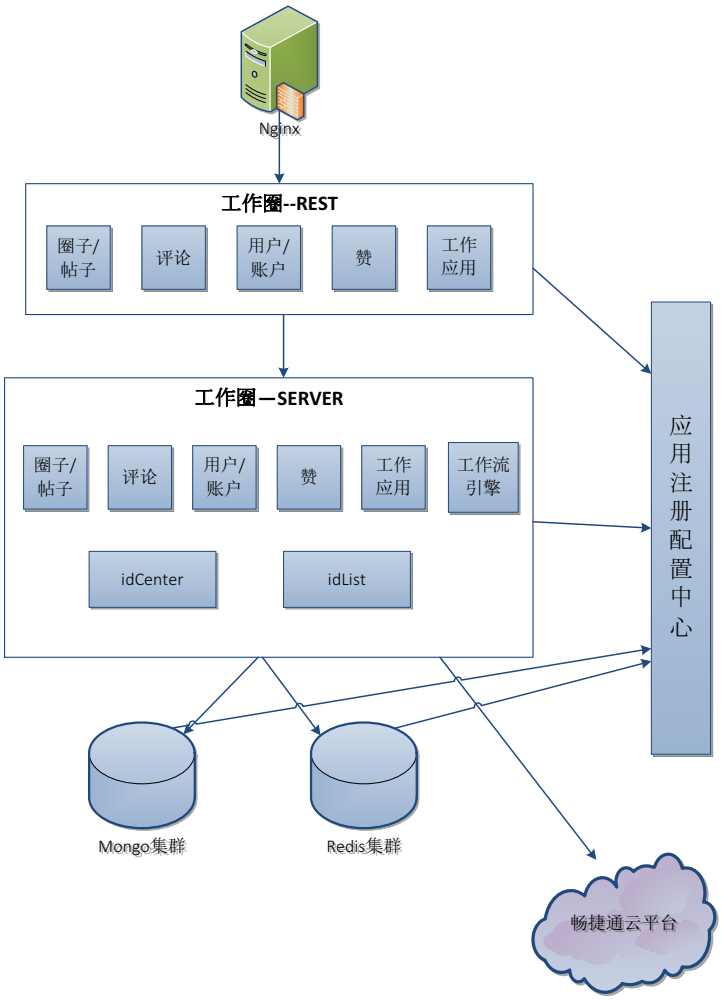


图 4-2 工作圈整体技术架构图

4.4.2 工作圈 Rest 设计

工作圈的用户终端有三种：Android、iOS 和 PC Web。这三个端可以简单的区分成为 PC 端和移动终端。这样区分是因为这两种端对应的场景和对后台的数据交互要求都是不一样的。PC 端的特点：运行在浏览器中（B/S 结构），网络环境信号较强，数据传输几乎没有成本，终端的尺寸较大可以展示的内容丰富，终端更换的频率较高；移动端特点：原生的 APP 程序运行在移动智能操作系统中（C/S 结构），容易处在弱网络环境中，数据传输有成本，终端的尺寸有限，终端不易更换。从 Web 与移动端的对比来看，移动端需要轻量级的增量更新接口，这样意味着移动端的数据在从服务端下行之后需要自行缓存在终端。这样可以避免在弱网环境下程序请求不到数据，同时节省的数据重复传输的成本。PC 端应用因为是 B/S 结构不便于本地缓存数据，所以在查询数据的时候接口应该设计成为全量分页性的接口，根据页码来下行查询数据。然而站在具体的功能业务上，同一个功能的业务逻辑却又是相同的，只是对于不同类型的终端提供的数据交互接口不一样。后台的服务就不能只是一个 server 的设计了，首先应该把业务逻辑封装成为一层，本文中先简称为 server 层。然后在 server 层之上封装一个 rest 层，给不同的终端提供不同的服务。如图 4-3。

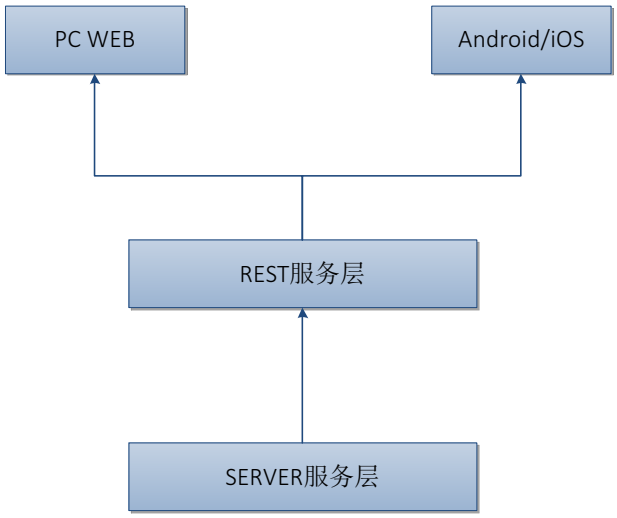


图 4-3　终端交互示意图

1. Nginx 负载均衡

Rest server 是整个工作圈服务的表现层，是和工作圈移动端或是 WEB 端通信的一层。工作圈终端借由 Rest Server 这一层提供的 Rest api，来和工作圈服务进行数据的交换。同样，在面对大规模用户高并发的访问时一台服务器的能力是有限的，云服务就需要多台服务器来分担压力。站在最前面的负载服务器就

尤其重要，这里选择了前面介绍过的 Nginx 服务器来做为 HTTP 请求的代理网关与负载。如图 4-4。

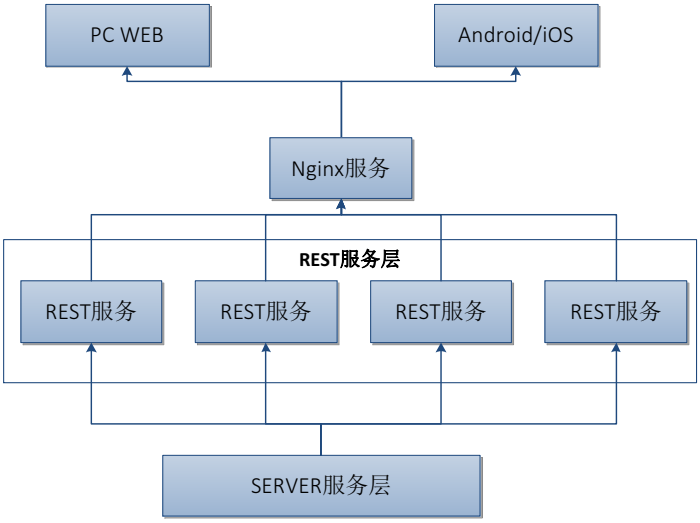


图 4-4 Rest 集群示意图

2. 无状态的服务设计

其实提出这点反而是为了解决用户的登录状态问题，假设用户登录的请求发到了 Rest A 服务上，对用户的数据请求又发送到了 Rest B 服务上。那 Rest B 怎么能知道用户的登录状态呢？在过去的单点式系统中，有一个 Session 的概念。用户登录后建立在服务器，请求的时候服务端会查询这个 Session，登出后销毁。但是在分布式的 Rest server 上主机之前内存的数据不共享，无法得知用户的 Session 状态。

工作圈的解决方式是登录后返回一个加密的 token 给终端。不是在服务端保存用户的登录状态，而是在用户使用的终端上保存这个 token。在 Android iOS 系统上是由工作圈 App 软件保存在手机的数据库中，PC WEB 因为没有数据库，将这个 token 保存在 Cookie 中。在每次请求的时候将 WEB 端会自动的带上 Cookie 供服务器查询，手机端的 App 需要程序从手机上读取这个请求加到 Http Header 中来便于服务检查。

4.4.3 分布式服务与自动化配置中心

模块划分完成之后，这个框架还不能被称为分布式框架。因为它还没有满足前面（2.1.3 现代分布式系统的特点）中提到的特性。下面来尝试进一步的满足这些特性。

1. 分布式服务远程调用（RPC）

服务层的模块规划完成了，接着要解决 Rest 层对服务层的业务服务调用。在第二章介绍的 Apache Thrift 就是一个优秀的 RPC 框架。Thrift 使用自己的 IDL 语言来定义远程服务的接口，然后使用 Thrift 工具生成基于定义的服务端接口以及可以调用服务的客户端 Client。

前面介绍了工作圈的两层结构：REST、Server。Client 就是 REST 与 SEVER 之间的桥梁。工作圈的 Server 层是分布式部署的，所以 Client 层这个桥梁就需要支持分布式的 RPC 调用。Client 首先要通过 ZkHelper（工作圈封装的 Zookeeper 统一操作接口）来读取服务注册的情况，简单说就是拿到已经注册在 Zookeeper 中心的可用服务器 IP 列表。然后随机的从这个列表中挑出一个服务，尝试与其建立连接。连接成功就调用服务，失败说明服务是有问题的需要把这个问题 IP 从 Zookeeper 中删掉然后取列表中的下一个 IP。这样就形成了一完整的分布式服务的负载与监控。Client 层封装 Server 层通过 Thrift 定义的 RPC 方法，将其暴露给其它的服务。需要使用模块的服务的时，引用相应的 Client 即可。

以用户账户服务 gongzuoquan-account 举例，Thrift 会为开发者生成一个接口定义程序 gongzuoquan-account-core。gongzuoquan-account-server 在这个包的基础上实现它定义的服务接口；gongzuoquan-account-client 在这个包的基础上封装的客户端的调用方法。gongzuoquan-account-rest 通过引用 client 实现了访问 server 的目的，结构如图 4-5。

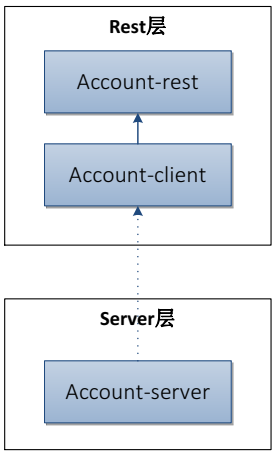


图 4-5 Server-Rest 通信示意图

2. 横向可扩展性（Scale Out）

横向可扩展（Scale Out）意味着一个业务服务模块是可以部署很多个副本的，这样就带来一个问题，如何保证所有的请求都能平均的分发到所有的服务上去？ 前面完成了 REST 与 SERVER 层的剥离与远程调用，在已知 server 服务的 IP 前提下就可以进行调用。在有多台服务实例情况下，简单的办法就是维护一

个服务可用的 IP 数组，然后随机的从数组中选出一个服务 IP 来进行调用。这样就需要一个自动化配置中心，在一个模块服务启动时，将启动的服务器在网络上的 IP 地址注册到自动化配置中心服务中，这样就完成了服务状态的发布。然后任一客户端在需要调用这个服务的时候，先在自动化配置中心中读取一下这个服务的服务状态，取得了可以提供这个服务的服务器 IP 列表，然后再以一个算法随机的选择其中一个地址发起请求，完成一次调用。这样就满足了服务的横向可扩展性。

Zookeeper 是这个自动化中心实现的最佳选择。服务以一个约定的路径：/gongzuoquan/account/cluster/在 Zookeeper 服务中注册一个节点，然后把服务自身的 IP 地址写入到这个节点下，如图 4-6。

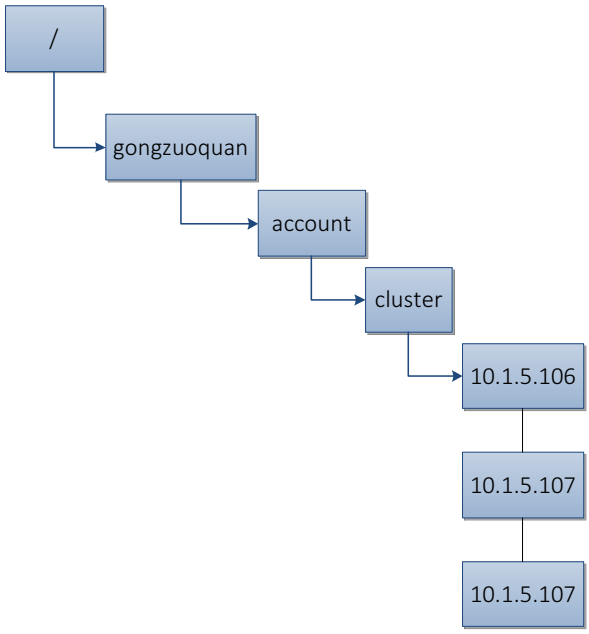


图 4-6 模块服务注册中心数据节点示意图

3. 不允许单点失效（No Single Point Failure）

不允许单点失效（No Single Point Failure）。只要对自动化配置中心的功能稍加改动，便能满足这一特性。只要能将服务状态列表中的故障节点检测到，并从中剔出除们，客户端便不会受到影响。只要这个服务状态列表中仍有一台服务器上的服务是正常的，就会不造成整个服务失效。当然，这里也要考虑到这个自动化配置中心服务本身会出问题，需要保证这组服务也是高可用的服务。

对故障列表的检测，有两种方式：1.在启动的服务与自动化配置中心之间建立一个心跳，心跳消失即失效；2.从客户端来检测，如果客户端调用服务失败后，将其从自动化配置中心中剔除然后访问下一个服务地址。

第一种方式，因为心跳会有延时。如果在延时中发生问题，那么在这一段时间内就会有一定机率的请求会请求到故障节点上。（如下图 4-7）第二种方式，需要做到客户端实时监听自动化配置中心的服务器状态变化，这样虽然会避免请求故障节点，但是在客户端数量过大的时候会大量的占用自动化配置中心服务的链接。工作圈目前的设计，客户端服务还是可控的，所以选择了第二种实时性较高的处理方式。

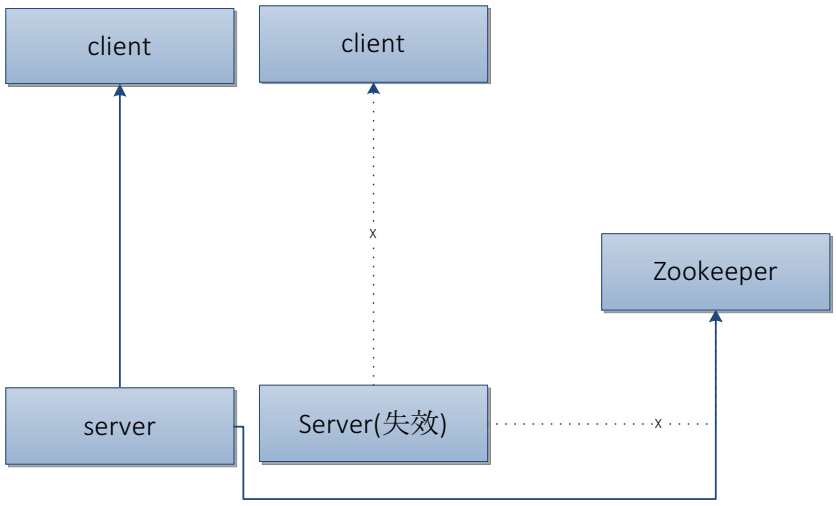


图 4-7 服务-配置中心心跳示意图

在实际的开发中，也可以将数据库和缓存服务的地址与链接配置存到 Zookeeper 中相应的节点下。当服务器的数据库或是缓存进行迁移数所据的时候，监听节点的变化，重新建立指向新服务的链接池就可以了。

4.4.4 数据分片(Data Sharding)与数据索引中心

大数据时代，数据的持久化性能会影响服务的整体的性能。如果一个模块下的业务都存储在一个表中，那么随着数据量的上升性能必然会下降。解决的办法是对数据库中的数据进行水平扩展(分表，分库等)，控制住单表中的数据量不会超过一个阀值即可。这个阀值的界定不是固定的，会根据数据库的选择以及运行的环境不同而不同。比如系统需要数据库插入一条业务数据的响应时间要在 1ms 以内，需要在实际的运行环境中对数据库进行一个 insert 压力测试，得到在多少条数据的情况下单表的插入响应时间超过了 1ms（这个时间需要根据业务场景来设定），那么这个阀值就是估算的一个单表数据量上限的指标。假设这个值是 f，需求预估的整个系统容量是 m，那个分表数 t = m / f。如果数据库服务器不只一台，那么可以再将数据散列到不同的数据库上去，此时假设有 n 台数据库服务，那么 t = m / (f * n)。

对数据进行分表，分库的插入的过程称为数据分片(Data Sharding)。常见的分片策略有几种：1.根据 ID 特征：例如对记录的 ID 取模，得到的结果是几，那么这条记录就放在编号为几的数据分区上；2.根据时间分区：例如以年/月为单位，某一年/月中产生的数据放在一起；3.基于索引表：根据 ID 先在一个复合索引中查到分区信息，然后再到分区中去查找具体的数据。这些分片的策略没有好坏之分，实际应用中需要根据具体的业务需求或是数据特征来选择设计。需要注意的是：数据分片不是银弹，它对系统的性能和伸缩性（Scalability）带来一定好处的同时，也会对系统开发带来许多复杂度。例如，有两条记录分别处在不同的分区中，如果有另一条数据与这两条记录其中之一有关联，那么关联的信息就需要在这两个分区中各保存一次。在对数据完整性有要求的场合，也就是对数据操作需要事务的情况下，跨分区的事务会变成“性能杀手”。另一方面，数据分片对全局扫描的需求也没有带来提升，反而增加了扫描的复杂度。

工作圈的数据分片使用了 ID 取模和索引表两种的方式，但是对这个策略进行了改进。首先引入一个 gongzuoquan-idcenter 的服务，用来随机生成数据在工作圈服务内全局唯一的主键 id。这样，id 在进行取模分片的时候就可以较为均匀的分布在所有的分表中。同时为了解决跨区数据查询的问题，服务需要一个数据索引中心（类拟于 Hadoop 的 nameService 的概念)。将一组有关系的数据 id 索引在一起保存起来，在查询的时候通过这个索引可以得到这些数据的 id，然后再通过 id 取模来得到分库分表的数据，从而查询到需要的业务数据。工作圈中承担这部分工作的核心服务是 gongzuoquan-idlist。

4.4.5 工作圈部署架构

工作圈完整的部署架构如图 4-8 所示。工作圈主体的服务是由 Rest 层和 Server 层来承担的。为了提高性能，Rest 和 Server 层都有不止一台的服务器来提供服务，每一台 Rest 服务器中都部署着工作圈所有 Rest 模块，同样 Server 中的每一台服务器也都部署着工作圈所有的 Server 模块。这样部署，使得整个系统中只要有一个 Rest 服务器和 Server 服务器还在工作，都可以继续为用户提供服务。集群中任意一个节点的故障都不会影响到系统的完整功能，仅会影响整个集群的性能。Nginx 承担着网关与 Rest 层服务的负载，图中可以看到 Rest 集群一共层部署了 4 台主机，Server 集群一共部署了 8 台主机。注册配置中心用到的 Zookeeper 集群服务与 Server 集群共享了 5 台主机。MongoDB 集群使用了一个 3×2 的方案，集群使用了三组服务做分布式数据存储，每一组服务都是由 Master/Slave/仲裁节点做的高可用配置。

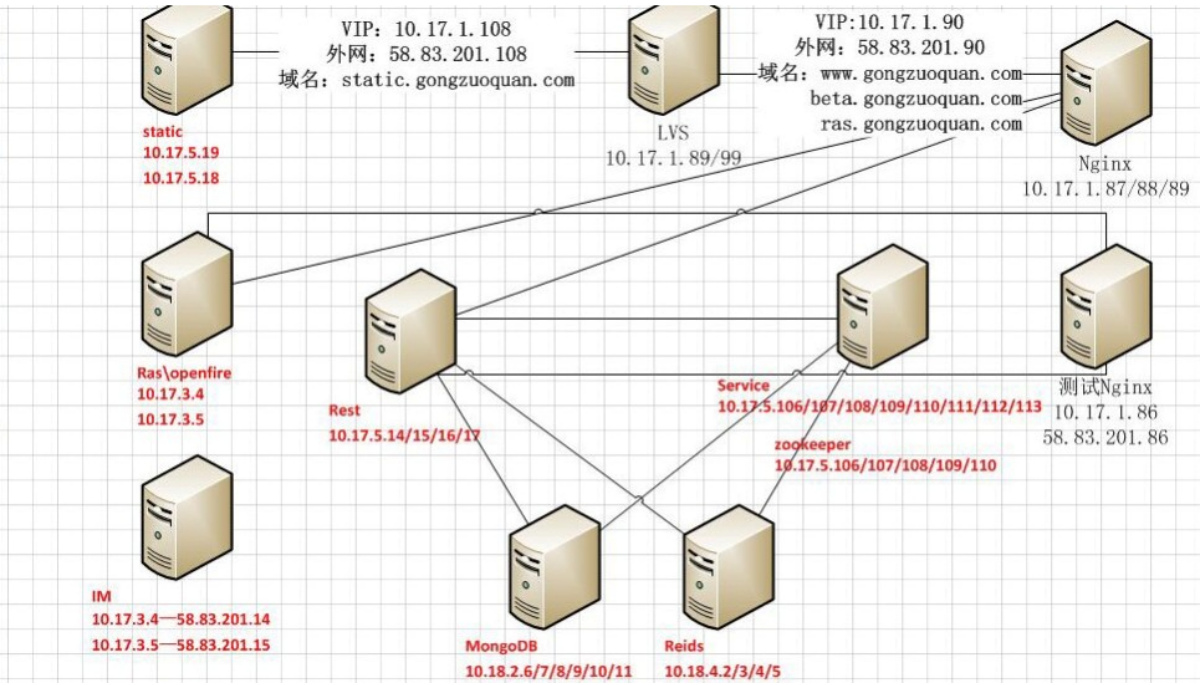


图 4-8 工作圈部署网络拓扑图

4.5 本章小结

本章重点描述了工作圈的 Rest server 层、模块化的服务层的设计。以前一章的需求作为入口，对工作圈的服务进行了模块的划分，并且对公共功能的模块进行了封装设计，减少开发人员的工作量。结合第二章的分布式思想为 Rest 和 Server 层设计了负载均衡的解决方案。同时也对持久化的数据存储进行了数据分片的设计，以期能够在大量用户并发的场景下能有高性能的表现。综上所述为本章的核心内容，本章虽然没有具体提及某个功能的具体设计与实现，但从宏观的角度出发以系统运转主线为基础简明的描述了整个系统的业务逻辑，在后面的章节中将选择这些逻辑或模块中的核心内容作为重点阐述。

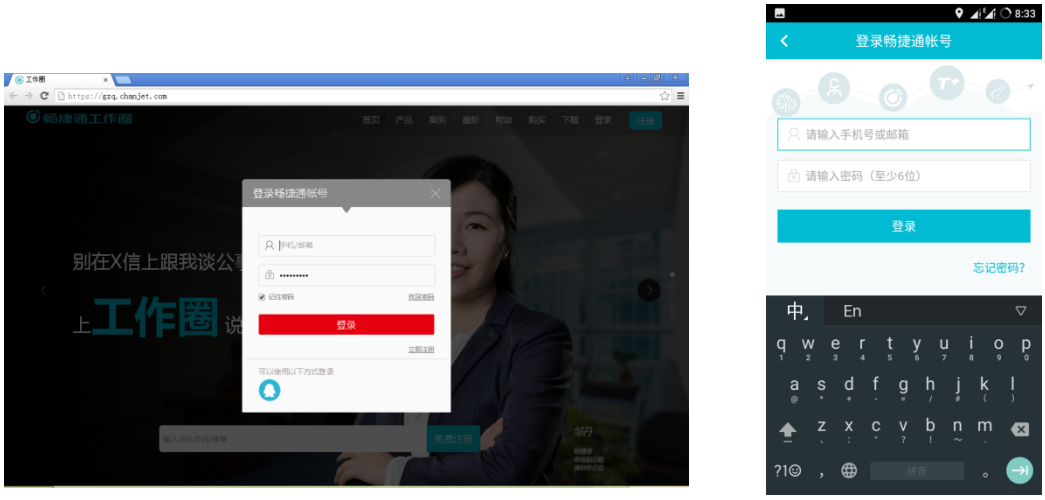
第五章 工作圈功能模块的设计与实现

上一章讲述了工作圈的总体框架的设计思路与服务划分。本章结合第三章章的需求以及第四章的设计，给出了用户账户、圈子\帖子、评论服务、工作流引擎模块的设计与实现。

5.1 客户端界面设计

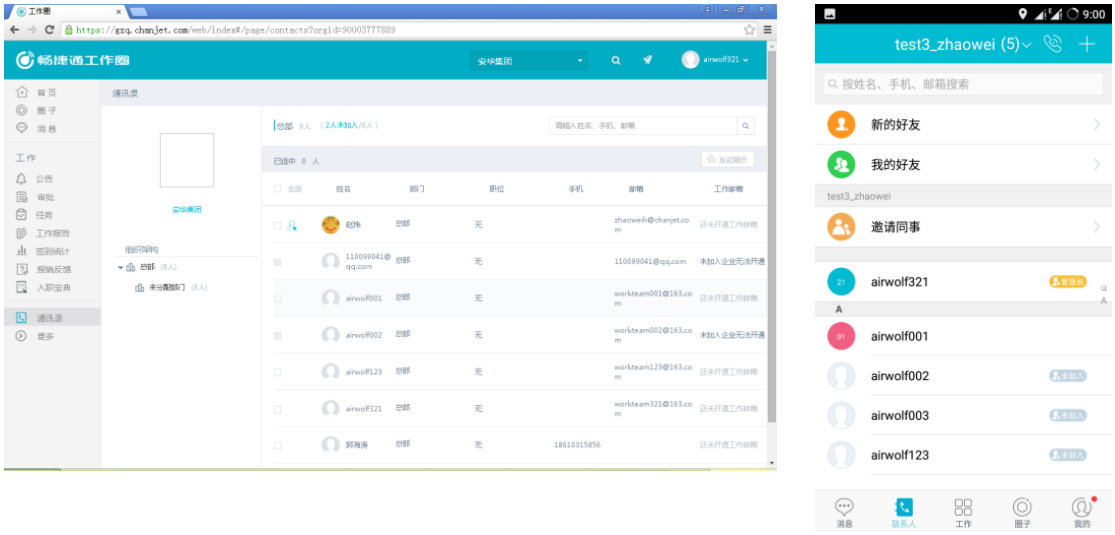
5.1.1 用户账户界面设计

用户账户的主要功能有注册与登录工作圈、浏览管理企业通讯录等。WEB 端可以支持手机号和电子邮箱的注册，同时还支持通过 QQ 号绑定注册的快速通道。而手机端仅允许手机号注册。如图 5-1



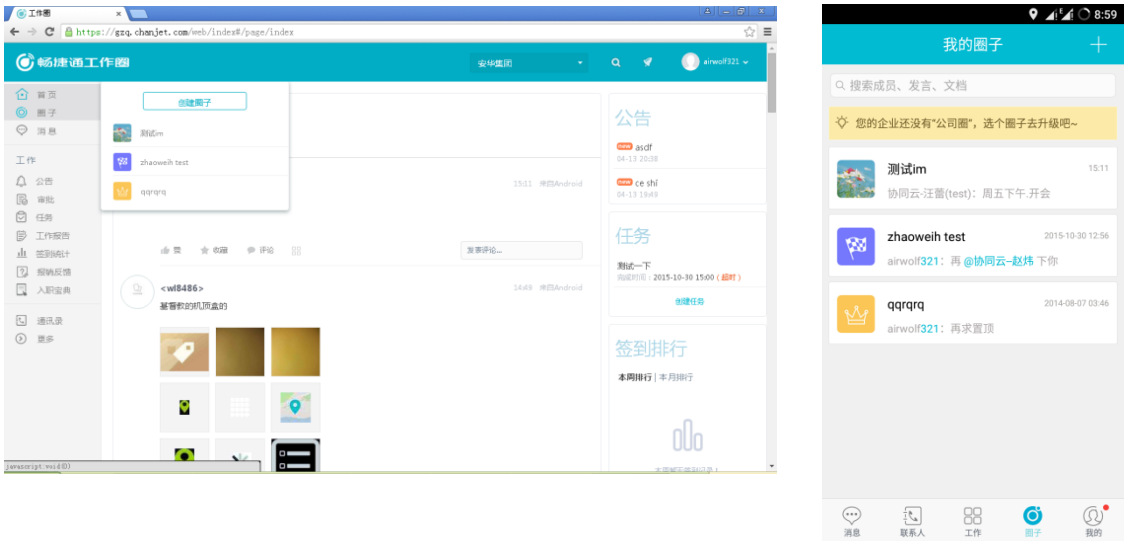
5-1 PC WEB 与手机的登录界面

在企业通讯录界面，用户可以搜索和查看企业中的员工信息和部门信息，企业管理员可以添加可删除企业中的员工、部门。企业通讯录的 WEB 和 PC 界面设计如图 5-2。



5-2 PC WEB 与手机的企业通讯录界面

5.1.2 圈子与帖子界面设计



5-3 PC WEB 与手机的圈子列表界面

圈子界面的主要功能就是编辑圈子信息和邀请用户加入圈子等。帖子界面的功能有：发帖，浏览圈子中的帖子，给帖子点赞和评论等。如图 5-3。

WEB 端可视化的窗口较大，展示的信息也全面，帖子列表、帖子内容、评论和点赞的功能的交互都展示在了一个界面中。右图的手机端仅能展示一个圈子的列表，需要点击某一个圈子才能查看到这个圈子的帖子列表。如图 5-4。

5-4 手机的帖子列表界面

在帖子列表的下方，用户可以发表纯文本的帖子，也可以通过使用手机的摄像头或者选用手机已有的图片来编辑一个图文并茂的帖子。在每一个帖子中都有点赞、评论和收藏的按钮。查看帖子详情，用户需要点击帖子列表中的某一个帖子，评论的列表会展示在帖子的详情中。如图 5-5。

5-5 手机的帖子详情界面

5.2 服务详细功能设计与实现

5.2.1 用户账户模块详细功能设计与实现

前面介绍了，工作圈的用户体系是畅捷通 CIA 云平台用户管理的一个子集，同时工作圈账户系统支撑工作圈各个端的单点登录、权限校验、注册、用户/企业信息查询和变更，并且基于畅捷通云平台的智能平台，为工作圈用户提供智能推荐、搜索服务。

工作圈的注册流程十分简便，用户可以通过有效的手机号或者是电子邮箱进行注册。工作圈通过向用户注册的手机号或邮箱发送一条含用验证码的信息，然后在注册的过程中输入正确的验证码即可确认这个手机号或者电子邮箱是用户本人的并且是有效的。因为工作圈使用的是CIA的用户，所以注册的过程需要调用 CIA 的注册接口，确认 CIA 注册成功之后方可以写入工作圈自己的用户数据库中，如图 5-6。

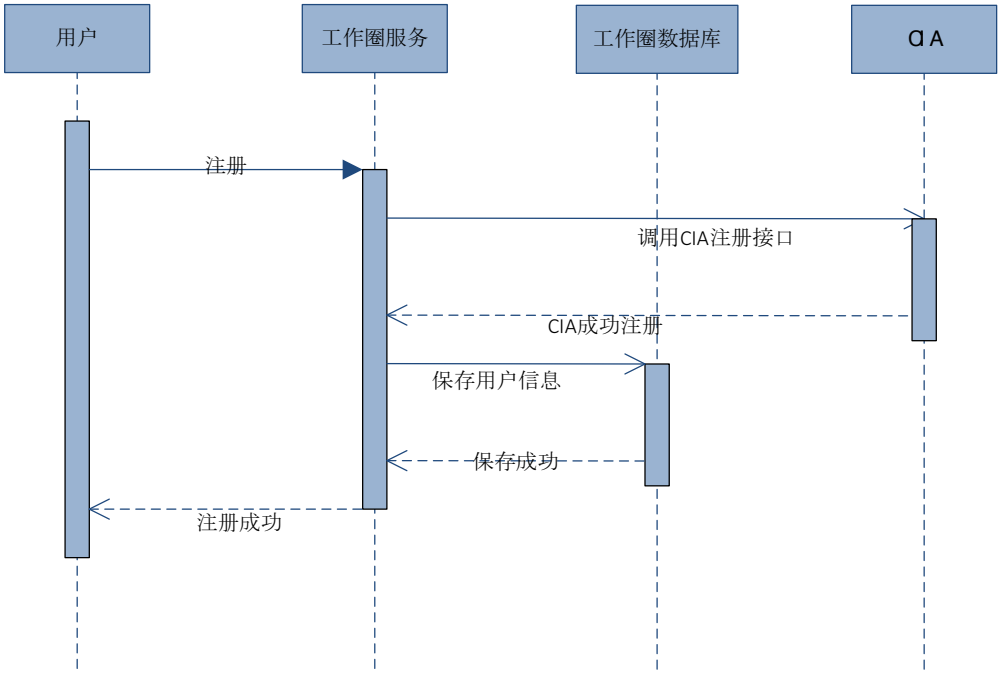


图 5-6 用户注册时序图

用户注册的伪代码实现如下：

```
// 发送短信验证码
if 是有效的手机号 then 调用 CIA 平台短信验证码发送接口
else if 是有效的邮箱账号 then 调用 CIA 平台邮件验证码发送接口
else 既不是手机号也不是邮箱账号
```

return 提示用户：‘您输入的账号信息不正确，请输入有效的手机号码或者邮箱账号’

// 注册账号
if 验证码不正确
  return 提示用户：‘您输入的验证码有误，请重新输入’
else
  do 调用 CIA 平台注册接口
  if CIA 平台注册成功 then 保存到工作圈的数据库
    return 注册成功
  else
    return 注册失败

由于用户信息是 CIA 的，工作圈的用户账户仅是一个子集。那么工作圈用户信息与 CIA 用户信息的同步就很重要，否则会出现用户修改了用户信息之后，登录工作圈信息还是过去的状态。工作圈和 CIA 用户同步信息是通过一个消息总线的服务实现的，当 CIA 有一个用户信息被更新或是删除的时候，CIA 服务会生成一条格式化的消息放在消息总线上，消息的内容包括这个用户的标识(userId)、操作类型、操作的字段、操作的结果。工作圈用户账户模块需要从这个消息总线上订阅 CIA 发出的用户变更类型的消息，当有新的消息时会自动的推送到工作圈的用户账户模块，从而更新工作圈的用户信息。

工作圈的用户登录的流程与注册类似，调用 CIA 系统的鉴权接口，成功校验用户名密码之后，再为将工作圈的用户置为登录状态。工作圈的页面与服务端是同过 REST 接口进行交互的，前面已经介绍过了 REST 下的无状态的用户登录设计，具体实现如下图 5-7。 图中的拦截器是一个所有模块通用的拦截器，每一个需要校验用户登录信息。

用户登录状态鉴权的伪代码实现如下：

// 用户鉴权拦截器实现
If 是移动端发起的请求 then 从 Http RequestHeader 中获取工作圈 Token
else
  do 从 Http Cookie 中获取工作圈 Token
if Token 有效 then 调用用户实际请求的业务接口
else
  return 提示用户登录过期，跳转登录界面

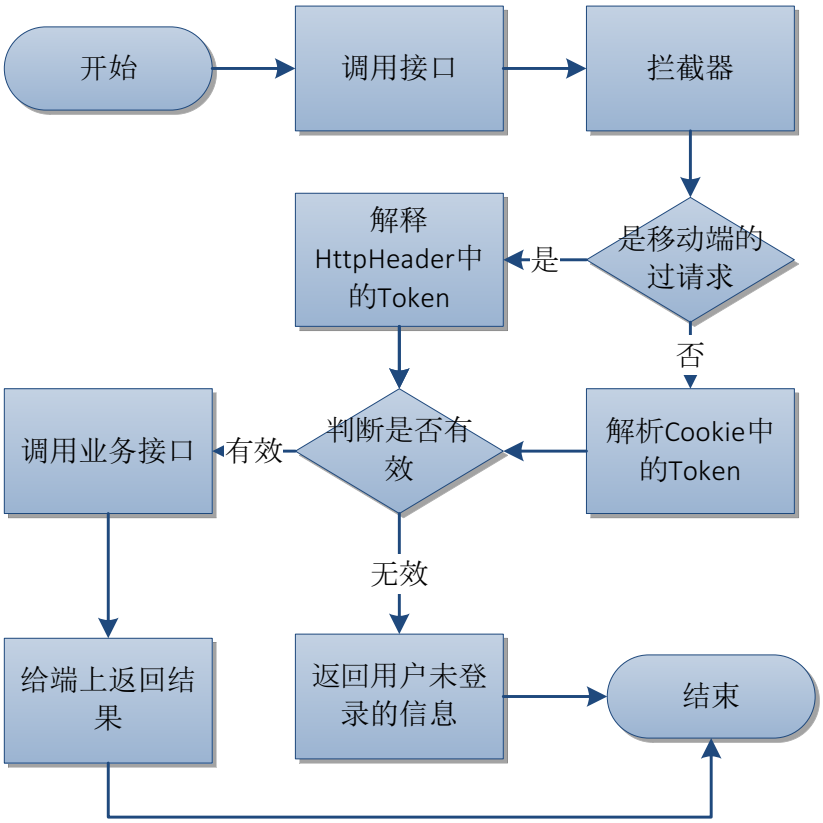


图 5-7 用户登录状态校验流程

5.2.2 圈子帖子模块详细功能设计与实现

工作圈的圈子系统是创建编辑圈子和帖子，并邀请或申请加入圈子的模块。工作圈圈子系统，包括创建编辑删除圈子、创建删除展示帖子，邀请加入圈子，申请加入圈子等功能。

用户创建圈子之后即成为圈子的管理员（圈主），可以修改圈子的信息（名称、LOGO）等，用户创建圈子和编辑圈子信息的流程如图 5-8 所示。

圈子成员都是圈主或者已加入圈子的成员邀请加入的，邀请的来源可以是用户的手机通讯录，也可以是用户所在企业的企业通讯录，也可以是用户手动输入的手机号或是电子邮箱账号等。

用户创建圈子的伪代码实现如下：

```
if 圈子名称为空
    return 要求用户填写圈子名称
else
    do 保存圈子数据
if 保存成功 then 邀请人员加入圈子
    return 圈子创建成功
```

else

return 圈子创建失败

加入圈子的成员就可以在圈子中发帖子。帖子中可以发布 500 个文字，9 张图片与用户的当前位置信息等。工作圈的帖子的一大特点就是可以@某一个特定的用户，被@的用户必须是圈子的成员之一，他可以接受到关于这个帖子@他的消息。@用户的信息是发布在帖子的文本内容中，帖子的数据提交到服务端之后，服务端使用正则对这个文本中的@信息进行解析，得到被@用户的唯一标识（userId），然后通过调用即时消息的接口为被@的用户推送一条圈子消息，提示有人在帖子中@过他。

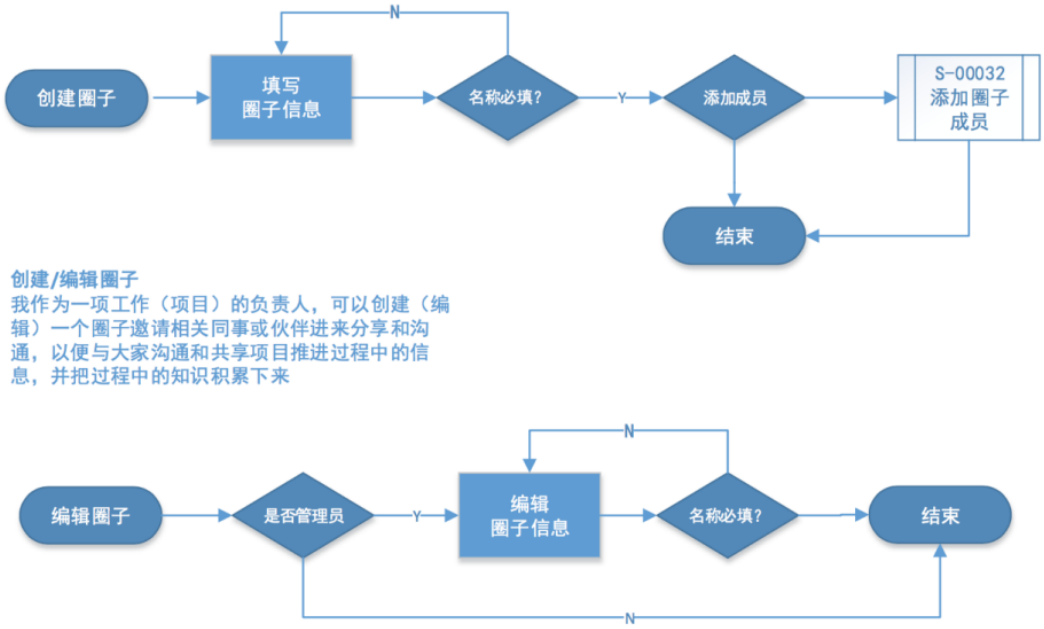


图 5-8 圈子创建编辑流程图

用户发帖的伪代码实现如下：

if 发帖人在圈子内

if 帖子信息校验合法 then 保存帖子

if 帖子内容中有@人的内容 then 为被@的用户推送通知

return 发帖成功

else

return 提示用户帖子信息有误的地方需要修改

else

return 提示用户：‘您不在圈子中，不能发帖’

5.2.3 评论模块详细功能设计与实现

工作圈的评论系统为工作圈提供了一个公共评论/回复服务。工作圈中涉及评论与回复相关业务的功能都可以使用这个服务来发布相关的评论信息。评论系统为工作圈中的帖子，轻应用中的审批、请假、任务等功能提供了评论和回复评论的功能。

评论之所以能为任意一个功能提供评论的服务，是因为评论将所有使用评论模块的功能视为一个 Resource，然后以这个 Resource 的唯一标识 resourceid 为主来聚合所有关于这个 Resource 的评论。所以在已知 resourceid 的情况下，就可以新增，删除和查询一个 Resource 的评论了。

评论功能相关的伪代码实现如下：

```
// 新增评论
if ResourceId 为空
   return 评论保存的 ResourceId 不能为空
else
   do 将评论保存在这个 ResourceId 对应的评论列表中
   if 评论是回复他人的评论 then 给被回复人发送一条回复类型的通知
   else
      do 给 Resource 的创建者发送一条评论类型的通知
   if 评论的内容中有@他人的内容 then 给被@的用户发送一条评论@类型的通知
   do 将评论的数据以异步处理的方式发送到智能平台
   return 评论成功

// 删除评论
if 操作用户是评论的发起者 then 删除评论
   return 删除成功
else if 操作用户是管理员 then 删除评论
   return 删除成功
else
   return 提示用户:'您不能删除他人的评论'

// 查询评论列表
if ResourceId 不为空 then 从数据库中查询这个 Resource 下的评论列表
   return 返回查询结果
else
```

return 查询参数不正确

5.2.4 赞系统模块详细功能设计与实现

工作圈的赞系统为工作圈提供了一个公共平台性质的点赞服务。并且基于畅捷通云平台的智能平台为工作圈用户提供智能推荐、搜索服务。赞与评论提供的功能性接口非常相似，其中一个的区别是它们的数据结构有差异，点赞的数据实体中是没有内容的，也不可以@其他用户；另一个区别是点赞没有删除，但是有取消赞的功能，也不能为这个 Resource 重复点赞。

赞功能相关的伪代码实现如下：

```
// 点赞
if ResourceId 信息为空
   return 点赞的 ResourceId 不能为空
else if 用户没有为这个 ResourceId 点过赞 then 将赞保存在这个 ResourceId 对应的点赞列表中
     do 给被点赞人发送一条点赞类型的通知
        将赞的数据以异步处理的方式发送到智能平台
   return 成功
else
   return 提示用户:'您已经点过赞了'

// 取消赞
if 操作用户是赞的发起者
   if 用户为这个 ResourceId 点过赞 then 取消赞
      return 取消成功
   else
      return 提示用户:'您还没有点过赞'
else
   return 不能取消他人的点赞

// 查询赞列表
if ResourceId 不为空 then 从工作圈的数据库中查询这个 Resource 下的赞列表
   return 返回查询结果
else
   return 查询参数不正确
```

5.2.5 工作流引擎模块详细功能设计与实现

工作流引擎，为工作圈中的工作应用提供一个可定制的，自由的工作流程服务。这个服务模块是主要是为了工作应用服务提供的一个服务，并不面向实际用户，所以不需要 REST 层和 Restful 接口。工作流引擎服务层的设计类似于圈子模块的设计，设计了两个数据模块 Task 和 FlowInstance。

FlowInstance，就是对一个特定的工作流的定义实例。比如某企业的一个员工需要使用审批，那么就需要一个 FlowInstance 来保存一个企业的审批工作流。FlowInstance 保存的数据包括定义这个审批流的企业信息，发起人信息，还有审批人列表。

Task，是工作流任务的定义。在工作中，领导交代的任务都可以自上而下的被分解成为一个一个的任务，有时可同时并行工作，有时需要按部就班的一个一个的处理，有时需要多个人来协作，有时只能一个人来完成。在工作流定义的这个 Task 就是为了描述这个分解后的任务设计的。Task 保存了当前处理人，下一步处理人和任务信息的引用(FlowInstanceId)。

有了 FlowInstance 和 Task，自由工作流就容易实现了。当用户发起一个自由工作流的时候，创建一个没有 TaskList 的 FlowInstance，而实际的处理人则由这个流程的发起人来制定，同样的处理人可以指定和修改下一步的处理人信息，最后一步的处理人处理完单据之后，为单据打上完成的标签。这个看似复杂的步骤，其实是在为后面的智能推荐做准备。用户的每一次发起和完成的工作流都将被记录到智能平台，然后根据每一次的单据的处理记录和用户与处理人之间的关系来推导出这个类型的工作流可能的处理过程，然后为其他相同角色的用户直接生成处理人列表。也就是说，在收集到足够多的数据之后，这个公司的用户在发起工作流的时候将不再需要指定处理人，工作圈将为其自动的添加处理人列表。让用户来完善他们的工作流定义，这就是自由工作流的最终目的，解决了如何为上千万家企业提供工作流的问题。

工作流功能相关的伪代码实现如下：

```
// 发起工作流
do 从智能平台获取为用户推荐的工作流数据
if 有推荐 then 用户确认是否符合实际情况
  if 符合 then 通过推荐的数据来创建工作流
  else
    do 用户修改一下错误信息
       将新的参考数据推送智能平台
       通过用户修改后的结果来创建工作流
else
```

```
    do 让用户自己新创建一个工作流
       将新的工作流推送到智能平台
       保存新的工作流
  do 推送处理信息到工作流的下一个处理人

  // 处理工作流
  填写处理结果
  if 处理成功
    if 工作流中是否有下一个处理人 then 推送处理信息到工作流的下一个处理人
      else
        do 给之前的所有处理人推送信息，提示工作流处理完毕
           关闭这个工作流
  else
    do 给之前的所有处理人推送信息，提示工作流处理失败
       关闭这个工作流
```

5.3 本章小结

本章主要讲解了工作圈基础服务与业务模块的具体设计与实现。文中给出了的基于第四章总体设计思路下的类设计与接口设计，同时探讨了用户账户、圈子、帖子、评论等关键业务的实现思路，具体的代码没有再进行赘述。一个系统的从设计到开发，是一个极其复杂的过程，程序员很容易在开发的时候迷失或是犯错。一个好的系统框架为程序员带来的价值是显而易见的，使得程序员能高效率，高质量的交付模块是本章设计的重点。

第六章 结束语

6.1 论文工作总结

本文介绍了企业协同办公发展的历史于现状。从当下的移动互联网出发，探讨了现在企业协同的不足之处以及改进的方向，并且对企业协同的未来发展给出了分析。在提出意见之后，立足于移动互联网的云服务，本文尝试给出一个基于云服务的企业协同办公的解决方案。

本文分析了分布式计算的思想，并结合现代开源社区组件设计了一个高性能，高可用的分布式系统框架。通过这个分布式的框架实现了基于分布式开源技术下的企业移动协同云办公服务。

本文取得的主要成果有：

1. 分析企业协同软件的发展方向
2. 阐述了分布式思想
3. 设计了一个以开源技术为基础的分布式服务框架
4. 以分布式服务框架为基础，完成了一个基于分布式开源技术下的企业移动协同云办公服务。

6.2 问题和展望

关于企业协同云服务的探讨，本文仅是一个基础性质的实现。在开头提出的设想还有很多没有来得及完成。企业帐户已经实现了，企业协同办公的应用也有了。下一步宏大的设想，也是公司正在实现的就是基于这个云服务与企业用户群实现的一个企业办公应用平台与生态圈，联合更多的开发商来为工作圈的企业用户提供服务，当平台的服务越加完善的时候就会吸引更多的企业用户来使用这个云平台，形成良性循环。

本文探讨的这个分布式框架，有两个不太明显的性能瓶颈。说不太明显，是因为这两个问题是在两个数据量非常大的情况下会出现的。一个是 Zookeeper 服务，当注册的服务非常之多的时候，它的性能会有问题。另一个是分布式数据索引服务，当业务数据量过大，而且没有很好的进行分片的时候会对这个服务有较大的影响。互联网与移动互联网将人类的沟通拉进了一个前所未有的地球村，从世界的一个地方发送一个信息到遥远的另一方不过短短的数秒，人们之前的交流变的非常便利，分布式的服务在其中做了很大的贡献。本文所描述的框架还有许多可以扩展的地方。

Docker，如果说近两年虚拟化最火的热词无疑就是这个技术了。它以最轻量级的方式为程序运行打造一个完整的虚拟机环境，可以快速的分发、部署与运行。这样，分布式服务就能方便的在业务高峰时段进行弹性扩展与收缩。

Spark，一个基于 Hadoop 的流数据处理组件。互联网的分布式服务和大数据处理两者是相辅相成的，当用户量达到千万级别的时候，产生的数据是非常可观的，同时其中的价值也是巨大的。

参考文献

[1] 吴强 分布式系统一致性的发展历史（一） 点融黑帮 2015 年 9 月 8 日

[2] L Lamport Time, Clocks, and the Ordering of Events in a Distributed System Operating Systems July 1978 Fig.1

[3] 黄晓军,张静，张凯 Apache Thrift - 可伸缩的跨语言服务开发框架 IBM developWorks 2012 年 1 月 16 日

[4] 许令波 分布式服务框架 Zookeeper -- 管理分布式环境中的数据 IBM developWorks 2010 年 11 月 18 日

[5] canezk, icbd, 许瑞等 译 Redis 介绍 Redis.io 2016 年 4 月 27 日

[6] 胡艳辉 小微企业金融服务创新与完善 经济与管理 2014 年第 20 期

[7] 金蓓弘,马应龙等 译 分布式系统:概念与设计 原书第 5 版 机械工业出版社 2013 年 3 月 1 日

[8] Alexander Misel, Abadcafe, ZéroBot 等 CAP 定理 Wikipedia.org 2015 年 12 月 11 日

[9] 王璞 分布式系统的特点以及设计理念 InfoQ 2015 年 6 月 22 日

[10] 洛若曼尼斯 智能 WEB 算法 电子工业出版社 2011 128-200

[11] 刘鹏 云计算 电子工业出版社 2010 38-108

致 谢

四年的研究生学习生活马上就要结束了，在毕业之际首先要感谢我的导师吴国仕，吴老师从一开始审阅论文初稿到完成都付出了大量的宝贵时间和心血，无论是从具体内容还是创新观点都给了我悉心的指导。在与吴老师学习的过程中，增强了我的分析能力和对行业未来发展的洞察能力。吴老师对学术的深刻理解和严谨的态度让我感触颇深，受益匪浅。在此谨向吴老师表达由衷的感谢，谢谢您在我的工作、学习和生活中的帮助和支持。

同时我还要感谢北京邮电大学，感谢计算机软件学院，为我们配备一流的师资力量和系统专业的知识体系，让我们的专业有了更大的提高和进步。感谢所有给我们上课的辛勤的老师们，不辞疲惫的为我们传授知识，在老师们的身上我学到了专业、耐心、坚持、毅力。而这些也必将影响我以后的工作和学习。

时光飞逝岁月荏苒。当我回首往昔，那些在北邮的上课的情形还历历在目，北邮的班级、老师讲坛上的身影、同学的笑容、还有放学时走在校园里偶带文艺风的音乐就像电影画面一样定格在脑海里。最后再次感谢北邮所有老师和同学的陪伴和帮助，谢谢你们给了我研究生生活美好的回忆。祝愿北邮的老师们，工作顺利，桃李满天下。祝愿同学们事业有成，前程似锦。