# 第一章 引言

## 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）

我们看到，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，从业人员还要番几番。为如此庞大的用户群提供服务，一几台高性能服务器的提供服务是远远不够的。眼下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，大部份员工不是7×24在线使用，所以这个服务还需要具务弹性扩展的能力，不仅能随着用户的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。

## 课题任务

### 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、贴子，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。

### 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)。

## 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、以及要达到的预期等。

第二章 相关技术介绍，介绍本课题中使用到的所有关键技术。

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运用到的核心技术进行介绍。运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。

## 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。分布式系统正真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便利。

### 分布式系统的定义

我们把分布式系统定义成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。我们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，多线程处理执行并是是很常见的方式。现在的CPU核心处越来越多，为了能充分利用CPU的运算能力，多线程并发是一种很好的方式。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来过时行的。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的同步状态。

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格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。

我们看一下现代分布式系统的几个例子，包括Web搜索、多人在线游戏和金融交易系统，也考察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。接着本章强调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。

构造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、透明性和提供服务质量的问题。

### 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[3]。）

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[4]。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]

### 现代分布式系统的特点

1.对服务器硬件要求低

• 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。

• 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者分布式系统的性能，再快的服务器也需要等网络络IO。

2.强调横向可扩展性

横向可扩展性（Scale Out）是指通过增加服务器数量来提升集群整体性能。纵向可扩展性（Scale Up）是指提升每台服务器性能进而提升集群整体性能。现代CPU的主频已进到达了4GHz的门槛，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。而且服务节点的冗余和数据的冗余来保证他的可靠性。

通常情况，不要让服务器满负荷运行，服务器长时间满负荷运行的话，出故障的概率显著升高。所以分布式系统采用一大堆中低性能的PC服务器，尽可能把负载均摊到所有服务器上，让每台服务器的负载都不高，保证集群整体稳定性。

4.分布式系统尽可能减少节点间通讯开销

如前所述，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.分布式系统应用服务最好做成无状态的

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。分布式应用服务最好是设计成无状态。因为如果应用程序是有状态的，那么一旦服务器宕机就会使得应用服务程序受影响而挂掉，那存在内存的数据也就丢失了，这显然不是高可靠的服务。把应用服务设计成无状态的，让程序把需要保存的数据都保存在专门的存储上，这样应用服务程序可以任意重启而不丢失数据，方便分布式系统在服务器宕机后恢复应用服务。[8]

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。

## 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。

### 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。

目前流行的服务调用方式有很多种，例如基于 SOAP 消息格式的 Web Service，基于 JSON 消息格式的 RESTful 服务等。其中所用到的数据传输方式包括 XML，JSON 等，然而 XML 相对体积太大，传输效率低，JSON 体积较小，新颖，但还不够完善。Apache 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，其传输数据采用二进制格式，相对 XML 和 JSON 体积更小，对于高并发、大数据量和多语言的环境更有优势。[2]

### 分布式服务框架 Zookeeper

Zookeeper 分布式服务框架是 Apache Hadoop 的一个子项目，它主要是用来解决分布式应用中经常遇到的一些数据管理问题，如：统一命名服务、状态同步服务、集群管理、分布式应用配置项的管理等。

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2.2.1所示：

图 2.2.1 Zookeeper 数据结构[3]

Zookeeper 这种数据结构有如下这些特点：

1. 每个子目录项如 NameService 都被称作为 znode，这个 znode 是被它所在的路径唯一标识，如 Server1 这个 znode 的标识为 /NameService/Server1。
2. znode 可以有子节点目录，并且每个 znode 可以存储数据，注意 EPHEMERAL 类型的目录节点不能有子节点目录。
3. znode 是有版本的，每个 znode 中存储的数据可以有多个版本，也就是一个访问路径中可以存储多份数据。
4. znode 可以是临时节点，一旦创建这个 znode 的客户端与服务器失去联系，这个 znode 也将自动删除，Zookeeper 的客户端和服务器通信采用长连接方式，每个客户端和服务器通过心跳来保持连接，这个连接状态称为 session，如果 znode 是临时节点，这个 session 失效，znode 也就删除了。
5. znode 的目录名可以自动编号，如 App1 已经存在，再创建的话，将会自动命名为 App2。
6. znode 可以被监控，包括这个目录节点中存储的数据的修改，子节点目录的变化等，一旦变化可以通知设置监控的客户端，这个是 Zookeeper 的核心特性，Zookeeper 的很多功能都是基于这个特性实现的。

Zookeeper 作为一个分布式的服务框架，主要用来解决分布式集群中应用系统的一致性问题，它能提供基于类似于文件系统的目录节点树方式的数据存储，但是 Zookeeper 并不是用来专门存储数据的，它的作用主要是用来维护和监控你存储的数据的状态变化。通过监控这些数据状态的变化，从而可以达到基于数据的集群管理。

### 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）。

你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。根据您的使用情况，您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。您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]

### NoSql数据库 MongoDB

1、存储形式，基于BSON文档结构的非关系性的数据。可以方便的扩展字段。相较于传统的关系型数据库，它可以使用一个表存储原来需要使用多表关联才能存储的数据库。

2、存储动态性，相较于传统的数据库当要增加一个属性值的时候要对表大动，MongoDB的面向BSON文档的形式可以使其属性值轻意的增加和删除。而原来 的关系型数据库要实现这个需要有很多的属性表来支持。

3、Mongodb有两种数据写入模式，

• 一种是高效的fire-and-forget模式就是只管向数据库服务器提交数据不等数据库服务器的回应。这个是数据库默认模式。

• 另外一种是安全模式，就是写入的同时还要与服务器同步，当数据的安全性要求高的时候适合用安全模式。

4、Mongodb的日志功能。

• 日志功能是帮助在系统Down机的时候恢复数据完整性做的，默认情况下Mongodb是开启日志功能的。每一个数据库操作都会先记录日志，所以当down机重启动服务器的时候数据库服务器能够通过日志文件恢复之前未完成的操作。

• 当然，开启日志必然影响性能，所以用户需要根据自己的实际应用来选择一个合适的工作模式。

• 如果你打算使用无日志模式，最好使用拷贝集群模式，这样可以降低数据丢失的风险。

5、Mongodb支持关系型数据库的所有索引模式（升序，降序，唯一，复合，地理空间索引），同时也支持二级索引（通过B-tree实现）。每一个collection支持64个索引

6、支持ad hoc queries用户自己可以定义自己的查询。

7、Mongodb的集群模式是主从模式的集群，其中主服务器只有一个，支持读写，而从服务器可以有多个只支持读取操作。集群的主要目的是通过数据的冗余实现failover模式，从而提高数据库服务器的可用性。当主服务器断掉后其中的一个从服务器会自动升级为主服务器，其它drive client只要通过failover就可以自动切换到新的主服务器，正常工作。而当原来的主服务器再恢复工作的时候，它将自动转为从服务器。

8、Mongodb采用了最新的低成本的横向扩展模式，相对于传统的单结点纵向扩展，可以节约成功，而且有更好的可靠性，更好的数据处理性能。

### 高性能HTTP服务器Nginx

Nginx是一个高性能的 HTTP 和 反向代理 服务器，也是一个 IMAP/POP3/SMTP 代理服务器。 Nginx 是由 Igor Sysoev 为俄罗斯访问量第二的 Rambler.ru 站点开发的，它已经在该站点运行超过两年半了。 Igor 将源代码以类 BSD 许可证的形式发布。尽管还是测试版，但是，Nginx 已经因为它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名了。在本文中将使用Nginx作为云服务的前端请求的代理与负载均衡。

## 本章小结

总而言之，分布式系统是大数据时代企业级应用的首选平台，它有良好的可扩展性，尤其是横向可扩展性（Scale Out），使得分布式系统非常灵活，能应对千变万化的企业级需求，而且降低了企业客户对服务器硬件的要求，真正能做到应用服务层面的弹性扩展（auto-scaling）。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。我们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。