Broadview
www.broadview.com.cn

从Paxos到ZooKeeper

分布式一致性原理与实践

倪超 著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

从Paxos到ZooKeeper

分布式一致性原理与实践

倪超 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书从分布式一致性的理论出发，向读者简要介绍几种典型的分布式一致性协议，以及解决分布式一致性问题的思路，其中重点讲解了 Paxos 和 ZAB 协议。同时，本书深入介绍了分布式一致性问题的工业解决方案——ZooKeeper，并着重向读者展示这一分布式协调框架的使用方法、内部实现及运维技巧，旨在帮助读者全面了解 ZooKeeper，并更好地使用和运维 ZooKeeper。全书共 8 章，分为五部分：第一部分（第 1 章）主要介绍了计算机系统从集中式向分布式系统演变过程中面临的挑战，并简要介绍了 ACID、CAP 和 BASE 等经典分布式理论；第二部分（第 2～4 章）介绍了 2PC、3PC 和 Paxos 三种分布式一致性协议，并着重讲解了 ZooKeeper 中使用的一致性协议——ZAB 协议；第三部分（第 5～6 章）介绍了 ZooKeeper 的使用方法，包括客户端 API 的使用以及对 ZooKeeper 服务的部署与运行，并结合真实的分布式应用场景，总结了 ZooKeeper 使用的最佳实践；第四部分（第 7 章）对 ZooKeeper 的架构设计和实现原理进行了深入分析，包含系统模型、Leader 选举、客户端与服务端的工作原理、请求处理，以及服务器角色的工作流程和数据存储等；第五部分（第 8 章）介绍了 ZooKeeper 的运维实践，包括配置详解和监控管理等，重点讲解了如何构建一个高可用的 ZooKeeper 服务。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

从 Paxos 到 Zookeeper：分布式一致性原理与实践 / 倪超著. —北京：电子工业出版社，2015.2
ISBN 978-7-121-24967-9

Ⅰ. ①从… Ⅱ. ①倪… Ⅲ. ①分布式操作系统－研究 Ⅳ. ①TP316.4

中国版本图书馆 CIP 数据核字(2014)第 275697 号

策划编辑：张春雨
责任编辑：徐津平
印　　刷：北京丰源印刷厂
装　　订：河北省三河市路通装订
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：27.25　字数：548 千字
版　　次：2015 年 2 月第 1 版
印　　次：2015 年 2 月第 1 次印刷
定　　价：75.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

问题的提出

在计算机科学领域，分布式一致性问题是一个相当重要，且被广泛探索与论证的问题，通常存在于诸如分布式文件系统、缓存系统和数据库等大型分布式存储系统中。

什么是分布式一致性？分布式一致性分为哪些类型？分布式系统达到一致性后将会是一个什么样的状态？如果失去了一致性约束，分布式系统是否还可以依赖？如果一味地追求一致性，对系统的整体架构和性能又有多大影响？这一系列的问题，似乎都没有一个严格意义上准确的定义和答案。

终端用户

IT 技术的发展，让我们受益无穷，从日常生活的超市收银，到高端精细的火箭发射，现代社会中几乎所有行业，都离不开计算机技术的支持。

尽管计算机工程师们创造出了很多高科技的计算机产品来解决我们日常碰到的问题，但用户只会倾向于选择一些易用、好用的产品，那些难以使用的计算机产品最终都会被淘汰——这种易用性，其实就是用户体验的一部分。

计算机产品的用户体验，可以分为便捷性、安全性和稳定性等方面。在本书中，我们主要讨论的是用户在使用计算机产品过程中遇到的那些和一致性有关的问题。在此之前，我们首先来看一下计算机产品的终端用户是谁，他们的需求又是什么。

火车站售票

假如说我们的终端用户是一位经常做火车的旅行家，通常他是去车站的售票处购买车票，然后拿着车票去检票口，再坐上火车，开始一段美好的旅行——一切似乎都是那么和谐。想象一下，如果他选择的目的地是杭州，而某一趟开往杭州的火车只剩下最后一张车票了，可能在同一时刻，不同售票窗口的另一位乘客也购买了同一张车票。假如说售票系

统没有进行一致性保障，两人都购票成功了。而在检票口检票的时候，其中一位乘客会被告知他的车票无效——当然，现代的中国铁路售票系统已经很少出现这样的问题了，但在这个例子中，我们可以看出，终端用户对于我们的系统的需求非常简单：

> “请售票给我，如果没有余票了，请在售票的时候就告诉我票是无效票的。”

这就对购票系统提出了严格的一致性要求——系统的数据（在本例中指的就是那趟开往杭州的火车的余票数），无论在哪个售票窗口，每时每刻都必须是准确无误的！

银行转账

假如说我们的终端用户是一名刚毕业的大学生，通常在拿到第一个月工资之后，都会选择向家里汇款。当他来到银行柜台，完成转账操作后，银行的柜台服务员会友善地提醒他：“您的转账将在 N 个工作日后到账！”此时这名毕业生有一些沮丧，会对那名柜台服务员叮嘱：“好吧，多久没关系，钱不要少就行了！”——这也成为了几乎所有的用户对于现代银行系统最基本的需求。

网上购物

假如说我们的终端用户是一名网上购物狂，当他看到一件库存量为 5 的心仪商品，会迅速地确认购买，写下收货地址，然后下单——然而，在下单的那个瞬间，系统可能会告知该用户：“库存量不足！”此时，绝大部分的消费者往往都会抱怨自己动作太慢，使得心爱的商品被其他人抢走了！

但其实有过网购系统开发经验的工程师一定明白，在商品详情页面上显示的那个库存量，通常不是该商品的真实库存量，只有在真正下单购买的时候，系统才会检查该商品的真实库存量。但是，谁在意呢？

在上面三个例子中，相信读者一定已经看出来了，我们的终端用户在使用不同的计算机产品时对于数据一致性的需求是不一样的：

> 有些系统，既要快速地响应用户，同时还要保证系统的数据对于任意客户端都是真实可靠的，就像火车站的售票系统。
>
> 还有些系统，需要为用户保证绝对可靠的数据安全，虽然在数据一致性上存在延时，但最终务必保证严格的一致，就像银行的转账系统。
>
> 另外的一些系统，虽然向用户展示了一些可以说是“错误”的数据，但是在整个系统使用过程中，一定会在某一个流程上对系统数据进行准确无误的检查，从而避免用户发生不必要的损失，就像网购系统。

试读结束：需要全本请在线购买： www.ertongbook.com

更新的并发性

在计算机发展的早期阶段，受到底层硬件技术的制约，同时也是由于人们对于计算机系统的实际使用需求比较简单，因此很多上层的应用程序架构都是单线程模型的。以 C 语言为例，其诞生于上世纪 70 年代，当时几乎所有使用 C 语言开发的应用程序都是单线程的。从现在来看，单线程应用程序虽然在运行效率上无法和后来的多线程应用程序相比，但是在编程模型上相对简单，因此能够避免多线程程序中出现的不少并发问题。

随着计算机底层硬件技术和现代操作系统的不断发展，多线程技术开始被越来越多地引入到计算机编程模型之中，并对现代计算机应用程序的整体架构起到了至关重要的作用。

多线程的引入，为应用程序带来性能上的卓越提升，同时也带来了一个最大的副作用，那就是并发。《深入理解计算机系统》[注1]一书对并发进行了如下定义：如果逻辑控制流在时间上重叠，那么它们就是并发的。这里提到的逻辑控制流，通俗地讲，就是一次程序操作，比如读取或更新内存中变量的值。

在本书后面的讨论中，我们提到的“并发”都特指更新操作的并发，即有多个线程同时更新内存中变量的值——我们将这一现象称为更新的并发性。

分布式一致性问题

在分布式系统中另一个需要解决的重要问题就是数据的复制。在我们日常的开发经验中，相信很多开发人员都碰到过这样的问题：假设客户端 C_1 将系统中的一个值 K 由 V_1 更新为 V_2，但客户端 C_2 无法立即读取到 K 的最新值，需要在一段时间之后才能读取到。读者可能也已经猜到了，上面这个例子就是常见的数据库之间复制的延时问题。

分布式系统对于数据的复制需求一般都来自于以下两个原因。

- 为了增加系统的可用性，以防止单点故障引起的系统不可用。
- 提高系统的整体性能，通过负载均衡技术，能够让分布在不同地方的数据副本都能够为用户提供服务。

数据复制在可用性和性能方面给分布式系统带来的巨大好处是不言而喻的，然而数据复制所带来的一致性挑战，也是每一个系统研发人员不得不面对的。

所谓的分布式一致性问题，是指在分布式环境中引入数据复制机制后，不同数据节点间

注1：*Computer Systems: A Programmer's* 是 Randal E.Bryant 和 David O'Hallaron 合著的一本计算机科学领域非常经典的基础性著作，其中文版本《深入理解计算机系统》由龚奕利和雷迎春共同翻译。

可能出现的，并无法依靠计算机应用程序自身解决的数据不一致情况。简单地讲，数据一致性就是指在对一个副本数据进行更新的同时，必须确保也能够更新其他的副本，否则不同副本之间的数据将不再一致。

那怎么来解决这个问题呢？顺着上面提到的复制延时问题，很快就有人想到了一种解决办法，那就是：

> “既然是由于延时引起的问题，那我可以将写入的动作阻塞，直到数据复制完成后，才完成写入动作。”

没错，这似乎能解决问题，而且有一些系统的架构也确实直接使用了这个思路。但这个思路在解决一致性问题的同时，又带来了新的问题：写入的性能。如果你的应用场景有非常多的写请求，那么使用这个思路之后，后续的写请求都将会阻塞在前一个请求的写操作上，导致系统整理性能急剧下降。

总的来讲，我们无法找到一种能够满足分布式系统所有系统属性的分布式一致性解决方案。因此，如何既保证数据的一致性，同时又不影响系统运行的性能，是每一个分布式系统都需要重点考虑和权衡的。于是，一致性级别由此诞生。

强一致性

> 这种一致性级别是最符合用户直觉的，它要求系统写入什么，读出来的也会是什么，用户体验好，但实现起来往往对系统的性能影响比较大。

弱一致性

> 这种一致性级别约束了系统在写入成功后，不承诺立即可以读到写入的值，也不具体承诺多久之后数据能够达到一致，但会尽可能地保证到某个时间级别（比如秒级别）后，数据能够达到一致状态。弱一致性还可以再进行细分：
>
> - **会话一致性**：该一致性级别只保证对于写入的值，在同一个客户端会话中可以读到一致的值，但其他的会话不能保证。
> - **用户一致性**：该一致性级别只保证对于写入的值，在同一个用户中可以读到一致的值，但其他用户不能保证。

最终一致性

> 最终一致性是弱一致性的一个特例，系统会保证在一定时间内，能够达到一个数据一致的状态。这里之所以将最终一致性单独提出来，是因为它是弱一致性

中非常重要的一种一致性模型，也是业界在大型分布式系统的数据一致性上比较推崇的模型。

本书将会从分布式一致性的理论出发，向读者讲解几种典型的分布式一致性协议是如何解决分布式一致性问题的。之后，本书则会深入介绍分布式一致性问题的工业解决方案——ZooKeeper，并着重向读者展示这一分布式协调框架的使用方法、内部实现以及运维技巧。

致谢

首先要感谢现在的部门老大蒋江伟先生。第一次接触蒋江伟是在 2011 年，当时参加了他的一个讲座“淘宝前台系统优化实践——吞吐量优化”，对其中关于“编写 GC 友好代码”的内容有不解之处，于是私下请教。他耐心的讲解令我至今记忆犹新。两年前，他全面负责中间件团队之后，给予了我更大的帮助和鼓励，使我得到了极大的进步，真的非常感谢。本书的问世，离不开他的推荐。也正是这一份写作的责任感，让我有决心和毅力来对整个 ZooKeeper 内容进行了一次全面的整理。在这里，衷心祝福蒋江伟先生带领中间件团队走向新的高度。

其次，本书的写作，离不开各位小伙伴们的支持和帮助，他们是各领域的资深专家，我向他们征集了很多有营养的内容。在这里，按照章节顺序，依次表示感谢：许泽彬参与了“问题提出”的写作；侯前明对 Paxos 算法的前世今生进行的整理；段培乐对晦涩的 Paxos 协议进行了细致的讲解；姜宇向我提供了他对于分布式事务的见解；徐伟辰参与了分布式锁服务 Chubby 相关的写作；叶成旭提供了他在上家公司时对 Hypertable 的学习和研究成果；高伟细致地向我展示了 Curator 这一 ZooKeeper 客户端的使用；陈杰提供了他在“自动化的 DNS 服务”场景中的经验总结；曹龙参与了 Hadoop 相关内容的写作；邓明鉴则贡献了他对 HBase 的深刻见解；作为产品的开源负责人，庄晓丹和王强提供了对消息中间件 Metamorphosis 技术架构的讲解；李鼎则向我全面展示了 RPC 服务框架 Dubbo 的技术细节；楼江航向我提供了 Canal 和 Otter 这两个分布式产品中的 ZooKeeper 应用场景；李雨前、柳明和温朝凯则一起写了终搜在产品演进过程中对 ZooKeeper 的使用和改进；封仲淹参与了对其自主产品 JStorm 的技术剖析……是你们一遍又一遍地对内容进行修改，才使得本书内容更为丰满。

另外，也要感谢温文鎏、王林、许泽彬、高伟和段培乐等人对全书的审阅，正是你们提出的宝贵建议，对完善本书提供了非常大的帮助。

感谢现在的同事陆学慧先生，从 2013 年下半年开始，他全面接手对 ZooKeeper 的开发和运维，在他身上感受到的专业和创新精神让我备受鼓舞。

另外，感谢我的第一个主管马震先生，是他的帮助为我指引了方向，让我有机会进入ZooKeeper的世界，并负责这个产品在公司的发展。尽管由于业务调整，马震先生已经转岗到其他部门，但依然由衷祝福他工作顺利。

还要感谢我的同事，阿里巴巴店铺平台的侯前明先生。本来该书作者应该是我们两个人，但是由于期间他的家庭又增加了一个小生命，导致其不得不中途退出。从本书的选题到写作大纲的制定，他都倾注了不少心血，相信如果有他一起创作，本书内容会更加丰满、深刻。这里表达遗憾的同时，也向这位两个孩子的父亲送去祝福，祝愿他生活美满。

感谢本书的责任编辑刘芸女士，是她反复审稿和编排，才能让本书的内容趋于完美。

感谢本书的封面设计吴海燕女士，她的努力已经无需言表，在技术书上的这一前卫、极富视觉冲击力的封面设计，深深震撼到了我，也希望读者朋友们能够喜欢。

尤其感谢本书的策划编辑张春雨先生。作为一个南方人，我很少有机会和那些有着一口北方腔的朋友交谈，第一次接到张春雨先生电话的时候，我才真正领略了北京腔，也正是他的邀请，才能让我有机会进行本书的撰写，同时在前后将近1年半的漫长写作过程中，也是他的帮助和鼓励，才让我坚持完成并不断完善本书的内容。在这里，也衷心祝愿张春雨先生事业更上一层楼。

最后，还有我的父母，在过去的1年时间里，多次放假没有回家，尽管父母一直鼓励我专注工作，专注于自己的事业，但我深知他们内心对儿子的牵挂，在这里也深深地向他们道一声："谢谢"，**也谨以此书献给我最亲爱的爸爸妈妈。**

倪　超

2014年12月于杭州淘宝城

目录

试读结束：需要全本请在线购买： www.ertongbook.com

第 1 章

分布式架构

随着计算机系统规模变得越来越大，将所有的业务单元集中部署在一个或若干个大型机上的体系结构，已经越来越不能满足当今计算机系统，尤其是大型互联网系统的快速发展，各种灵活多变的系统架构模型层出不穷。同时，随着微型计算机的出现，越来越多廉价的 PC 机成为了各大企业 IT 架构的首选，分布式的处理方式越来越受到业界的青睐——计算机系统正在经历一场前所未有的从集中式向分布式架构的变革。

1.1 从集中式到分布式

自 20 世纪 60 年代大型主机被发明出来以后，凭借其超强的计算和 I/O 处理能力以及在稳定性和安全性方面的卓越表现，在很长一段时间内，大型主机引领了计算机行业以及商业计算领域的发展。在大型主机的研发上最知名的当属 IBM，其主导研发的革命性产品 System/360 系列大型主机，是计算机发展史上的一个里程碑，与波音 707 和福特 T 型车齐名，被誉为 20 世纪最重要的三大商业成就，并一度成为了大型主机的代名词。从那时起，IT 界进入了大型主机时代。

伴随着大型主机时代的到来，集中式的计算机系统架构也成为了主流。在那个时候，由于大型主机卓越的性能和良好的稳定性，其在单机处理能力方面的优势非常明显，使得 IT 系统快速进入了集中式处理阶段，其对应的计算机系统称为集中式系统。但从 20 世纪 80 年代以来，计算机系统向网络化和微型化的发展日趋明显，传统的集中式处理模式越来越不能适应人们的需求。

首先，大型主机的人才培养成本非常之高。通常一台大型主机汇集了大量精密的计算机组件，操作非常复杂，这对一个运维人员掌握其技术细节提出了非常高的要求。

其次，大型主机也是非常昂贵的。通常一台配置较好的 IBM 大型主机，其售价可能在

上百万美元甚至更高，因此也只有像政府、金融和电信等企业才有能力采购大型主机。

另外，集中式系统具有明显的单点问题。大型主机虽然在性能和稳定性方面表现卓越，但这并不代表其永远不会出现故障。一旦一台大型主机出现了故障，那么整个系统将处于不可用状态，其后果相当严重。最后，随着业务的不断发展，用户访问量迅速提高，计算机系统的规模也在不断扩大，在单一大型主机上进行系统的扩容往往比较困难。

而另一方面，随着 PC 机性能的不断提升和网络技术的快速普及，大型主机的市场份额变得越来越小，很多企业开始放弃原来的大型主机，而改用小型机和普通 PC 服务器来搭建分布式的计算机系统。

其中最为典型的就是阿里巴巴集团的“去 IOE”运动。从 2008 年开始，阿里巴巴的各项业务都进入了井喷式的发展阶段，这对于后台 IT 系统的计算与存储能力提出了非常高的要求，一味地针对小型机和高端存储进行不断扩容，无疑会产生巨大的成本。同时，集中式的系统架构体系也存在诸多单点问题，完全无法满足互联网应用爆炸式的发展需求。因此，为了解决业务快速发展给 IT 系统带来的巨大挑战，从 2009 年开始，阿里集团启动了“去 IOE”计划，其电商系统开始正式迈入分布式系统时代。

1.1.1 集中式的特点

所谓的集中式系统就是指由一台或多台主计算机组成中心节点，数据集中存储于这个中心节点中，并且整个系统的所有业务单元都集中部署在这个中心节点上，系统的所有功能均由其集中处理。也就是说，在集中式系统中，每个终端或客户端机器仅仅负责数据的录入和输出，而数据的存储与控制处理完全交由主机来完成。

集中式系统最大的特点就是部署结构简单。由于集中式系统往往基于底层性能卓越的大型主机，因此无须考虑如何对服务进行多个节点的部署，也就不用考虑多个节点之间的分布式协作问题。

1.1.2 分布式的特点

在《分布式系统概念与设计》[注1]一书中，对分布式系统做了如下定义：

分布式系统是一个硬件或软件组件分布在不同的网络计算机上，彼此之间仅仅

注1：*Distributed Systems: Concepts and Design* 是 George Coulouris、Jean Dollimore、Tim Kindberg 和 Gordon Blair 合著的一本分布式领域经典著作，其中文版本《分布式系统概念与设计》由金蓓弘和马应龙共同翻译。

试读结束：需要全本请在线购买： www.ertongbook.com