HZ BOOKS
华章IT

云计算技术系列丛书

Cloud-Native Distributed Storage Cornerstone

etcd in-depth Interpretation

# 云原生分布式存储基石

## etcd深入解析

华为云容器服务团队 杜军 等编著

**云原生分布式存储领域的里程碑之作，华为云容器服务团队一线专家倾囊打造**

高度概括了分布式系统和一致性算法的理论，完整地介绍了Kubernetes的“心脏”——etcd的设计思想及原理，深度剖析了etcd的架构模块、性能调优以及应用场景

机械工业出版社
China Machine Press

云计算技术系列丛书

Cloud-Native Distributed Storage Cornerstone

etcd in-depth Interpretation



# 云原生分布式存储基石

## etcd深入解析

华为云容器服务团队 杜军 等编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

云原生分布式存储基石：etcd 深入解析 / 华为云容器服务团队等编著 . —北京：机械工业出版社，2018.11
（云计算技术系列丛书）

ISBN 978-7-111-61192-9

I. 云… II. 华… III. 云计算 IV. TP393.027

中国版本图书馆 CIP 数据核字（2018）第 237996 号

# 云原生分布式存储基石：etcd 深入解析

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：赵亮宇 责任校对：殷 虹
印 刷：中国电影出版社印刷厂 版 次：2019 年 1 月第 1 版第 1 次印刷
开 本：186mm×240mm 1/16 印 张：20
书 号：ISBN 978-7-111-61192-9 定 价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

# Preface 前 言

## 为什么要写这本书

近年来，容器和云原生生态蓬勃发展，我们正身处于一波云原生的浪潮中。随着我们习惯于在云端产生和收集的数据，云端积累了海量的数据并继续以惊人的速度增长。如何实现数据分布式、一致性存储，确保云原生环境的可扩展性和高可用性，是各组织亟须解决的现实问题。

云计算时代，etcd 必将成为云原生和分布式系统的基石！而奠定 etcd 基石地位的三个关键因素是 Raft 协议、Go 语言和生态。

etcd 从一开始就摒弃了以复杂和难以理解著称的 Paxos，而是另辟蹊径地通过 Raft 化繁为简，实现了一套健壮的分布式一致性协议的 SDK，这套 SDK 被很多其他分布式数据库 / 系统采用，甚至包括 etcd 兄弟项目 rkt 的竞争对手 Docker。

至于被誉为云时代 C 语言的 Go 语言，具备天然的高并发能力、易安装和可读性好等优点，成就了 etcd 的高性能和项目的易维护性，极大地激发了来自全世界的开源工程师参与 etcd 的热情。云原生领域用 Go 语言编写的重量级项目不胜枚举，例如 Docker、Kubernetes 和 Istio 等。

etcd 相对于 Zookeeper 是一个年轻且更加轻量的项目，它拥有更加健康和有活力的社区。截至这本书出版前夕，etcd 在 Github 上的 star 数是 20 000+，fork 数是 4000+，拥有超过 400 名活跃的代码贡献者。不能忽视的一点是，etcd 已经被 Kubernetes 和 Cloud Foundry 等顶级云原生项目采用，并借势经过了 Google、华为云、Red Hat、IBM、阿里等 IT 巨头大规模生产环境的考验。随着 etcd 进入 CNCF 社区孵化，成为由 CNCF 治理的顶级项目，厂商中立的运作模式将进一步繁荣 etcd 的开源生态。

顺势而为，再加上合理的架构设计，恰如其分的实现，完全让人有理由相信 etcd 的成功。

在我最开始接触 Kubernetes 的时候，就和 etcd 打过交道了。etcd 在华为 PaaS 平台作为关键组件应用在分布式数据协同与更新观察等架构中。犹记得那时 etcd 刚发布，我们希望它提升华为 PaaS 平台的扩展性、性能和稳定性。因此，我们团队还专门成立 etcd 特别攻关小组，吃透了 etcd 的内部运作机制和核心技术。我很荣幸成为这个小组的成员。从那时起，我便对 etcd 着了迷，一口气翻看了 etcd 的源码，同时也向 etcd 社区提交了若干个 patch。

对于我们团队来说，我们很荣幸见证了 etcd 在技术和社区的持续进步并成长为 Kubernetes 项目的一部分。etcd v3 的正式发布延续了这个势头，我们期待将来有更多的功能和特性被引入华为云容器平台的产品中。我们也很高兴过去能够与 etcd 团队和技术社区一起工作，并将持续与 etcd 技术社区协作，将这项技术推到一个更高的层面。

至于为什么要在工作之余抽空写这本书，我们在容器和 Kubernetes 技术布道的过程中发现，国内从事该领域的工程师普遍对 etcd 了解不多，出了问题鲜有定位手段，而 etcd 官网又没有中文资料，因特网上也缺少深入解析 etcd 原理的文章。本着回馈社区和普及云原生技术的原则，我们华为云容器服务团队决定编写这本书，做第一个“吃螃蟹”的人。

毕竟“源码面前，了无秘密”。

## 读者对象

这里我们可以根据软件需求划分出本书的受众：

- 分布式系统工作者
- Raft 算法研究者
- etcd 各个程度的学习者
- Kubernetes 用户与开发者

## 如何阅读本书

本书分为三部分，其中第二部分以接近实战的实例来讲解 etcd 的使用，相较于其他两部分更独立。如果你是一名分布式系统的初学者，请一定从第 1 章的基础理论知

识开始学习。

第一部分为基础篇，包括第 1 章，我们将简单地介绍分布式系统的基本理论，并且详解 Raft 算法的工作原理，帮助读者了解一些掌握 etcd 的基础背景知识。

第二部分为实战篇，包括第 2～7 章，我们将着重讲解 etcd 的常见功能和使用场景，包括 etcd 的架构分析、命令行使用、API 调用、运维部署等。

第三部分为高级篇，包括第 8～11章，我们将直接打开 etcd 的源码，为喜欢刨根问底的读者深度剖析 etcd 的实现原理。

## 勘误和支持

由于作者的水平有限，编写的时间也很仓促，书中不妥之处在所难免，恳请读者批评指正。如果你发现了书中的错误或者有更多的宝贵意见，欢迎发送邮件至我的邮箱 m1093782566@163.com，我很期待能够获得你们的真挚反馈。

## 致谢

我首先要感谢 etcd 的工程师团队，他们编写并开源了这么一款足以成为云原生基石的分布式存储系统——etcd。

感谢华为云容器服务团队的高级架构师、Kubernetes 社区核心维护者 Kevin 老师，他为这本书的出版提供了良好的技术氛围和宝贵的实战经验支持。

感谢 CMU 在读硕士研究生梁明强同学，在写作过程中为我提供了犀利而宝贵的意见和文字。

感谢机械工业出版社华章公司的编辑杨绣国老师，感谢你的魄力和远见，在这一年多的时间中始终支持我的写作，你的鼓励和帮助引导我能顺利完成全部书稿。

我要感谢我的爸爸、妈妈、外公、外婆，感谢你们将我培养成人，并时时刻刻为我灌输着信心和力量！

我要感谢我的爱人，你的陪伴和鼓励使得这本书得以顺利完成。

谨以此书，献给我最亲爱的家人，以及众多热爱云原生与分布式技术的朋友们。

杜军

中国，华为杭州研究所，2018 年 9 月

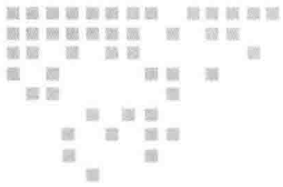
# 目 录 Contents

第一部分 Part 1

# 基 础 篇

本部分简单介绍分布式系统的基本理论，详细讲解 Raft 算法的工作原理，帮助读者了解 etcd 的基础知识，主要包括以下章节：

- 第 1 章　分布式系统与一致性协议



Chapter 1 第 1 章

# 分布式系统与一致性协议

现如今，摩尔定律的影响已经严重衰减甚至近于失效，但我们却实实在在地看到了计算能力的大幅度提升。在围棋人机大战里，人工智能 AlphaGo 打败李世石、柯洁的事实仍历历在目。计算能力的提升在很多时候都是源于系统（大数据、人工智能、云计算、区块链等）采用了分布式架构。《分布式系统概念与设计》一书中对分布式系统概念的定义如下：

> 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，彼此之间仅仅通过消息传递进行通信和协调的系统。

简单来说，分布式系统就是一组计算机节点和软件共同对外提供服务的系统。但对于用户来说，操作分布式系统就好像是在请求一个服务器。因为在分布式系统中，各个节点之间的协作是通过网络进行的，所以分布式系统中的节点在空间分布上几乎没有任何限制，可以分布于不同的机柜、机房，甚至是不同的国家和地区。

分布式系统的设计目标一般包括如下几个方面。

- 可用性：可用性是分布式系统的核心需求，其用于衡量一个分布式系统持续对外提供服务的能力。
- 可扩展性：增加机器后不会改变或极少改变系统行为，并且能获得近似线性的性能提升。
- 容错性：系统发生错误时，具有对错误进行规避以及从错误中恢复的能力。
- 性能：对外服务的响应延时和吞吐率要能满足用户的需求。

虽然分布式架构可以组建一个强大的集群，但实际工作中遇到的挑战要比传统单体架构大得多，具体表现如下所示。

1）节点之间的网络通信是不可靠的，存在网络延时和丢包等情况。
2）存在节点处理错误的情况，节点自身随时也有宕机的可能。
3）同步调用使系统变得不具备可扩展性。

## 1.1　CAP 原理

提到分布式系统，就不得不提 CAP 原理。CAP 原理在计算机科学领域广为人知，如果说系统架构师将 CAP 原理视作分布式系统的设计准则一点也不为过。

下面让我们先来回顾一下 CAP 的完整定义。

- C：Consistency（一致性）。这里的一致性特指强一致，通俗地说，就是所有节点上的数据时刻保持同步。一致性严谨的表述是原子读写，即所有读写都应该看起来是“原子”的，或串行的。所有的读写请求都好像是经全局排序过的一样，写后面的读一定能读到前面所写的内容。

- A：Availability（可用性）。任何非故障节点都应该在有限的时间内给出请求的响应，不论请求是否成功。
- P：Tolerance to the partition of network（分区容忍性）。当发生网络分区时（即节点之间无法通信），在丢失任意多消息的情况下，系统仍然能够正常工作。

相信大家都非常清楚 CAP 原理的指导意义：在任何分布式系统中，可用性、一致性和分区容忍性这三个方面都是相互矛盾的，三者不可兼得，最多只能取其二。本章不会对 CAP 原理进行严格的证明，感兴趣的读者可以自行查阅 Gilbert 和 Lynch 的论文[1]，下面将给出直观的说明。

1）AP 满足但 C 不满足：如果既要求系统高可用又要求分区容错，那么就要放弃一致性了。因为一旦发生网络分区（P），节点之间将无法通信，为了满足高可用（A），每个节点只能用本地数据提供服务，这样就会导致数据的不一致（!C）。一些信奉 BASE（Basic Availability，Soft state，Eventually Consistency）原则的 NoSQL 数据库（例如，Cassandra、CouchDB 等）往往会放宽对一致性的要求（满足最终一致性即可），以此来换取基本的可用性。

2）CP 满足但 A 不满足：如果要求数据在各个服务器上是强一致的（C），然而网络分区（P）会导致同步时间无限延长，那么如此一来可用性就得不到保障了（!A）。坚持事务 ACID（原子性、一致性、隔离性和持久性）的传统数据库以及对结果一致性非常敏感的应用（例如，金融业务）通常会做出这样的选择。

3）CA 满足但 P 不满足：指的是如果不存在网络分区，那么强一致性和可用性是可以同时满足的。

CAP 原理最初的提出者 Eric Brewer 在 CAP 猜想提出 12 年之际（2012 年）对该原理重新进行了阐述[2]，明确了 CAP 原理只适用于原子读写的场景，而不支持数据库事务之类的场景。同时指出，只有极少数网络分区在非常罕

见的场景下，三者才有可能做到有机的结合。无独有偶，Lynch 也重写了论文“ Perspectives on the CAP Theorem”[3]，引入了活性（liveness）和安全属性（safety），并认为 CAP 是活性与安全性之间权衡的一个特例。CAP 中的 C（一致性）属于活性，A（可用性）属于安全性。

正如热力学第二定律揭示了任何尝试发明永动机的努力都是徒劳的一样，CAP 原理明确指出了完美满足 CAP 三种属性的分布式系统是不存在的。了解 CAP 原理的目的在于，其能够帮助我们更好地理解实际分布式协议实现过程中的取舍，比如在后面的章节中将会提到的 lease 机制、quorum 机制等。

## 1.2　一致性

在阐述一致性模型和一致性协议之前，我们先来了解下什么是一致性。分布式存储系统通常会通过维护多个副本来进行容错，以提高系统的可用性。这就引出了分布式存储系统的核心问题——如何保证多个副本的一致性？

“一致性”这个中文术语在计算机的不同领域具有不同的含义，不同的含义所对应的英文术语也是不一样的，例如，Coherence、Consensus 和 Consistency 等。就这三个术语而言，简单来说，它们之间存在的区别具体如下：

- Coherence 这个单词只在 Cache Coherence 场景下出现过，其所关注的是多核共享内存的 CPU 架构下，各个核的 Cache 上的数据应如何保持一致。
- Consensus 是共识，它强调的是多个提议者就某件事情达成共识，其所关注的是达成共识的过程，例如 Paxos 协议、Raft 选举等。Consensus 属于 replication protocol 的范畴。
- Consistency 表达的含义相对复杂一些，广义上说，它描述了系统本身

的不变量的维护程度对上层业务客户端的影响，以及该系统的并发状态会向客户端暴露什么样的异常行为。CAP、ACID 中的 C 都有这层意思。

本书将要重点讨论的分布式系统中的一致性问题，属于上文中提到的 Consensus 和 Consistency 范畴。分布式系统的一致性是一个具备容错能力的分布式系统需要解决的基本问题。通俗地讲，一致性就是不同的副本服务器认可同一份数据。一旦这些服务器对某份数据达成了一致，那么该决定便是最终的决定，且未来也无法推翻。


一致性与结果的正确性没有关系，而是系统对外呈现的状态是否一致（统一）。例如，所有节点都达成一个错误的共识也是一致性的一种表现。

一致性协议就是用来解决一致性问题的，它能使得一组机器像一个整体一样工作，即使其中的一些机器发生了错误也能正常工作。正因为如此，一致性协议在大规模分布式系统中扮演着关键角色。

同时，一致性协议也是分布式计算领域的一个重要的研究课题，对它的研究可以追溯到 20 世纪 80 年代，一致性协议衍生出了很多算法。衡量一致性算法的标准具体如下。

- 可终止性：非失败进程在有限的时间内能够做出决定，等价于 liveness。
- 一致性：所有的进程必须对最终的决定达成一致，等价于 safety。
- 合法性：算法做出的决定值必须在其他进程（客户端）的期望值范围之内。即客户端请求回答“是”或“否”时，不能返回“不确定”。

一致性协议是在复制状态机（Replicated State Machines，RSM）的背景下提出来的，通常也应用于具有复制状态机语义的场景。在了解复制状态机之前，让我们先简单了解下一致性模型。