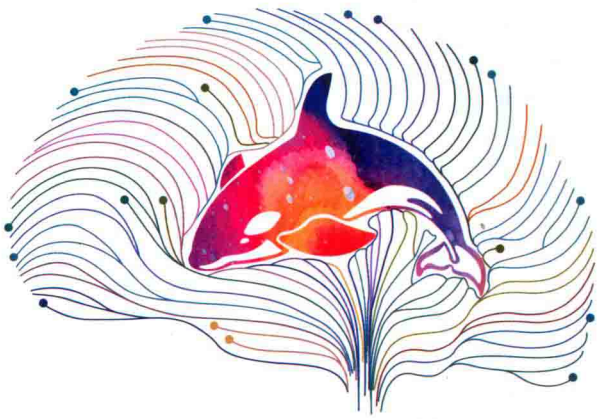
- 本书目的是让读者看了不会睡着的HBase技术书。因为我们坚信看一本非常重要，但是一看就想睡的书是一次痛苦的经历。
- 我们希望这本书能够给读者带来一次愉快而轻松的阅读经历，并在其中顺便学会HBase的安装部署、主要功能、架构设计、性能优化与周边项目。
- 本书适合HBase的初学者，欲深入了解HBase配置、部署、优化和二次开发的软件工程师，以及任何对云计算或者NoSQL相关技术感兴趣的读者。

大数据 技术



An Open-Source, Distributed, Versioned, Non-Relational Database

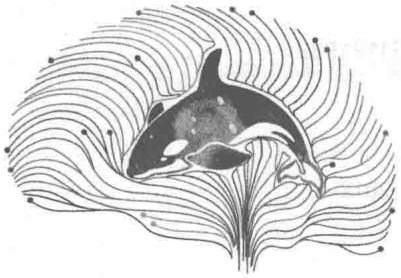
HBase 不睡觉书

杨曦 著

清华大学出版社

HBase

不睡觉书



HBase不睡觉书

杨曦 著

清華大學出版社
北京

内 容 简 介

HBase 是 Apache 旗下一个高可靠性、高性能、面向列、可伸缩的分布式存储系统。利用 HBase 技术可在廉价的 PC 服务器上搭建大规模的存储化集群，使用 HBase 可以对数十亿级别的大数据进行实时性的高性能读写，在满足高性能的同时还保证了数据存取的原子性。

本书共分为 9 章，由浅入深地讲解 HBase 概念、安装、配置、部署，让读者对 HBase 先有一个感性认识，再从应用角度介绍了高级用法、监控和性能调优。既兼顾了初学者，也适用于想要深入学习 HBase 的读者。

本书适合于以前没有接触过 HBase，或者了解 HBase，并希望能够深入掌握的读者，适合 HBase 应用开发人员和系统管理人员学习使用。

图书在版编目（CIP）数据

HBase 不睡觉书 / 杨曦著. — 北京：清华大学出版社，2018

ISBN 978-7-302-49055-5

I. ①H… II. ①杨… III. ①计算机网络－信息存贮 IV. ①TP393

中国版本图书馆 CIP 数据核字（2017）第 295500 号

责任编辑：夏毓彦
封面设计：王　翔
责任校对：闫秀华
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 装 者：三河市金元印装有限公司
经　　销：全国新华书店
开　　本：190mm×260mm　　印　张：26　　字　数：666 千字
版　　次：2018 年 1 月第 1 版　　印　次：2018 年 1 月第 1 次印刷
印　　数：1~3000
定　　价：89.00 元

产品编号：069575-01

前 言

为什么要叫不睡觉书呢？因为我们发现阻碍人们学习新技术最大的障碍不是技术的难度或者条件的限制，而是学习技术时难以抵挡的困意，所以我们的目标就是写一本让人看了不会睡着的 HBase 技术书籍。希望大家可以通过这本书成功地入门 HBase 技术。

为什么要写这本书？

- 目前网上关于 HBase 的知识比较零碎，缺乏系统性。翻译的作品，语言的组织又不符合国人的习惯。
- 目前的资料都很旧。连英文的资料很多都过时了，比如现在很多的书籍和网上的资料都还在介绍三层查询架构，可是 HBase 早已经改成二层查询架构了。实际操作到的跟书上的操作不一样，这很让人沮丧。

如何才能不睡着地看本书

作为本书的作者我强烈不建议大家从头按顺序地读到尾，这不是一种好的读书方式，而且极容易睡着。看书应该是非线性的，先扫一遍目录，然后只看适合自己的，最后再发散式地补看别的章节。

- 如果你手头没有合适的环境，或者你想快速了解 HBase 能干什么，或者你是公司的运维，想知道怎么搭建 HBase，“第 2 章 让 HBase 跑起来”适合你。
- 如果公司的运维帮你搭好了环境，老板催着你赶紧做出项目，那么请直接看“第 4 章 客户端 API 入门”。
- 如果你更关心 HBase 是如何实现它的数据结构的，建议你直接看“第 5 章 HBase 内部探险”。
- 如果你想知道 HBase 如何提升性能，建议你直接看“第 8 章 再快一点”。

如果你还是觉得困，那肯定不是这本书的关系，是你的确缺乏睡眠，请马上去睡觉，有精神了再来看书。看得慢，看得少都没有关系，千万别困着看！

如何才能不睡着地看所有书

为什么我们看技术书籍总是犯困呢？

因为技术书籍必须把方法和知识点都写全面，否则容易误导读者，你可以把技术书籍看

成是一本电话黄页。我们总是错误地以为既然要学习，那么每一个知识点、每一个方法都不能错过，所以认真地精读每一本技术书籍。你想象一下，如果你精读一本电话黄页，会不会感到疲劳？会不会忍不住睡去？

其实不光是读本书，学习所有的技术书籍都应该掌握正确的方法。那就是：跳着看，具体地说就是不要针对每一个 API 方法都精读，这样很容易迷失在一长串的 API 方法列表中，感到疲劳，导致无法坚持下去；而是针对某个知识点精细地掌握某一个方法，亲自实践这一个方法，然后别的方法快速略读过去，等回头需要用的时候再回来查阅。我们需要把每一本技术书籍都看成入门教程+技术手册，第一遍阅读的时候把每个知识点挑出一个方法作为入门，把其他方法当作技术手册来查阅，你总不会想细读一本电话黄页吧。

本书在很多地方都给出了阅读提示，提醒大家不要精读，该略过的部分就要勇敢地略过。

这本书不是 HBase 知识大全

这本书的目的只是让你学会 HBase。有些知识点并没有涉及，比如集群备份、ACL 权限控制、REST 客户端等，所以想学习这些知识的同学们可能要失望了。我只能让你们愉快地入门，更深层次的知识就看你们自己的努力了！

技术支持与致谢

如果你在看本书的时候发现了一些问题或者不足之处，请发邮件给 alexyang11@qq.com 告诉我。

部分彩色图片可以到下面网址（注意数字与字母大小写）下载：

https://pan.baidu.com/s/1slqjJnZ

最后感谢我的家人、朋友、同事对我编写本书的帮助，感谢清华大学出版社的夏毓彦编辑，感谢 HBase Team 的 Ted Yu，没有他们的帮助，我不可能完成本书！

著者

2017 年 11 月于硅谷

目 录

第 1 章

◂ 初识HBase ▸

1.1 海量数据与 NoSQL

1.1.1 关系型数据库的极限

想必大家都用过类似 MySQL 或者 Oracle 这样的关系型数据库。一个网站或者系统最核心的表就是用户表，而当用户表的数据达到几千万甚至几亿级别的时候，对单条数据的检索将花费数秒甚至达到分钟级别。实际情况更复杂，查询的操作速度将会受到以下两个因素的影响：

- 表会被并发地进行插入、编辑以及删除操作。一个大中型网站的并发操作一般能达到几十乃至几百并发，此时单条数据查询的延时将轻而易举地达到分钟级别。
- 查询语句通常都不是简单地对一个表的查询，而有可能是多个表关联后的复杂查询，甚至有可能有 group by 或者 order by 操作，此时，性能下降随之而来。

因此，当关系型数据库的表数据达到一定量级的时候，查询的操作就会慢得无法忍受。姑且不论聘请经验丰富的 DBA 进行深度优化的成本多少，实际情况是，哪怕是进行了深度的优化，情况仍然不容乐观。原本这种情况只发生在某些垄断行业中，但是现在随着越来越多的“独角兽公司”（估值达到 10 亿美元以上的公司）的出现，在海量数据下进行快速开发，并进行高效运行的需求越来越多。这可难倒了全世界的关系型数据库专家，世界的数据库技术似乎达到了瓶颈。怎么办呢？

1.1.2 CAP 理论

有的专家尝试将关系型数据库做成分布式数据库，把压力分摊到了多个服务器上，但是，随之而来的问题则是很难保证原子性。原子性可是数据库最根本的 ACID 中的元素啊！如果没有了原子性，数据库就不可靠了，这样的数据库还能用吗？如果增加一些必要的操作，那么原子性是保证了，但是性能却大幅下降了。专家们始终没有办法构建出一个既有完美原子性又兼具高性能的分布式数据库。

就在一筹莫展的时候，有人突然想起，20 世纪 90 年代初期 Berkerly 大学有位 Eric Brew

er 教授提出了一个 CAP 理论，如图 1-1 所示。

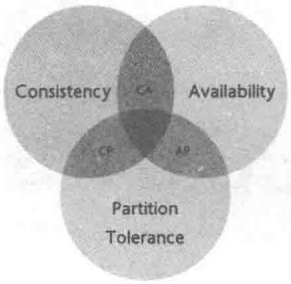


图 1-1

全称是 Consistency Availability and Partition tolerance：

- Consistency（一致性）：数据一致更新，所有数据变动都是同步的。
- Availability（可用性）：良好的响应性能。
- Partition tolerance（分区容错性）：可靠性。

Brewer 教授给出的定理是：

任何分布式系统只可同时满足二点，没法三者兼顾。

Brewer 教授给出的忠告是：

架构师不要将精力浪费在如何设计能满足三者的完美分布式系统，而是应该进行取舍。

这回人们可以不用纠结于如何设计一个拥有完美原子性的高性能分布式数据库了。现在的问题是，我们究竟要舍弃哪一个特性？

1.1.3 NoSQL

对 CAP 特性的放弃带来了一种全新类型的数据库——非关系型数据库。和关系型数据库正好相反，非关系型数据库 NoSQL 对事务性的要求并不严格，甚至可以说是相当马虎。

有些数据库是保证最终一致性，信息不会立即同步，而是经过了一段时间才达到一致性。比如你发了一篇文章，你的一部分朋友立马看到了这篇文章，另一部分朋友却要等到 5 分钟之后才能刷出这篇文章。虽然有延时，但是对于一个娱乐性质的 Web 2.0 网站又有谁会在乎这几分钟的延时呢？如果你用传统关系型数据库，网站可能早就宕掉了。

有些数据库可以在部分机器宕机的情况下依然可以正常运行，其实原理就是把同一份数据复制成了好几份放到了好几个地方，正应了那句老话：

不要把鸡蛋同时放在一个篮子里。

读者在之后的篇章中可以看到，HBase 正是这种类型的 NoSQL 数据库。

总之，数据库的世界从此开始百花齐放起来了，如图 1-2 所示。

图 1-2

很多人以为 NoSQL 是非 SQL 的意思，其实它是 Not Only SQL 的缩写，意思是不只是 SQL。

1.2 HBase 是怎么来的

2006 年 Google 技术人员 Fay Chang 发布了一篇文章 *Bigtable: A Distributed Storage System for Structured Data*。该文章向世人介绍了一种分布式的数据库，这种数据库可以在局部几台服务器崩溃的情况下继续提供高性能的服务。

2007 年 Powerset 公司的工作人员基于此文研发了 BigTable 的 Java 开源版本，即 HBase。刚开始它只是 Hadoop 的一部分。

2008 年 HBase 成为了 Apache 的顶级项目。HBase 几乎实现了 BigTable 的所有特性。它被称为一个开源的非关系型分布式数据库。

2010 年 HBase 的开发速度打破了一直以来跟 Hadoop 版本一致的惯例，因为 HBase 的版本发布速度已经超越了 Hadoop。它的版本号一下从 0.20.x 跳跃到了 0.89.x。

HBase Logo 的演化过程如图 1-3 所示。

图 1-3

看来我不是唯一一个觉得以前的 Logo 很丑的人。

1.3 为什么要用 HBase

HBase 的存储是基于 Hadoop 的。Hadoop 是这些年崛起的拥有着高性能，高稳定，可管理的大数据应用平台。Hadoop 已经快要变为大数据的代名词了，基于 Hadoop 衍生出了大量优秀的开源项目。

Hadoop 实现了一个分布式文件系统（HDFS）。HDFS 有高容错性的特点，被设计用来部署在低廉的硬件上，而且它提供高吞吐量以访问应用程序的数据，适合那些有着超大数据集的应用程序。基于 Hadoop 意味着 HBase 与生俱来的超强的扩展性和吞吐量。

HBase 采用的是 Key/Value 的存储方式，这意味着，即使随着数据量增大，也几乎不会导致查询的性能下降。HBase 又是一个列式数据库（对比于传统的行式数据库而言），当你的表字段很多的时候，你甚至可以把其中几个字段放在集群的一部分机器上，而另外几个字段放到另外一部分机器上，充分分散了负载压力。然而，如此复杂的存储结构和分布式的存储方式带来的代价就是：哪怕只是存储少量数据，它也不会很快。所以我常常跟人说：

HBase 并不快，只是当数据量很大的时候它慢的不明显

凡事都不可能只有优点而没有缺点。数据分析是 HBase 的弱项，因为对于 HBase 乃至整个 NoSQL 生态圈来说，基本上都是不支持表关联的。当你想实现 group by 或者 order by 的时候，你会发现，你需要写很多的代码来实现 MapReduce。

因此，请不要盲目地使用 HBase。

当你的情况大体上符合以下任意一种的时候：

- 主要需求是数据分析，比如做报表。
- 单表数据量不超过千万。

请不要使用 HBase，使用 MySQL 或者 Oracle 之类的产品可以让你的脑细胞免受折磨。

当你的情况是：

- 单表数据量超千万，而且并发还挺高。
- 数据分析需求较弱，或者不需要那么灵活或者实时。

请使用 HBase，它不会让你失望的。

1.4 你必须懂的基本概念

1.4.1 部署架构

在安装 HBase 之前，有几个概念你必须弄懂，否则你可能就不知道自己在装什么。

我们先从大到小介绍一下 HBase 的架构。从 HBase 的部署架构上来说，HBase 有两种服务器：Master 服务器和 RegionServer 服务器。

一般一个 HBase 集群有一个 Master 服务器和几个 RegionServer 服务器。Master 服务器负责维护表结构信息，实际的数据都存储在 RegionServer 服务器上，如图 1-4 所示。

HBase 有一点很特殊：客户端获取数据由客户端直连 RegionServer 的，所以你会发现 Master 挂掉之后你依然可以查询数据，但就是不能新建表了。