

内容全面，系统地讲解了Cassandra的所有功能特性及其使用方法
深入浅出，结合源代码分析了Cassandra的底层机制和工作原理

国内Cassandra领域的先驱者和实践者亲自执笔，多位数据库专家联袂推荐

Cassandra in Action

Cassandra 实战



郭鹏◎著



机械工业出版社
China Machine Press

数据库 技术丛书

Cassandra in Action

Cassandra 实战

郭 鹏◎著



机械工业出版社
China Machine Press

国内首本 Cassandra 专著，由 Cassandra 领域的先驱者和实践者亲自执笔，多位数据库专家联袂推荐，权威性毋庸置疑。

本书内容全面，基于 Cassandra 最新版撰写，系统地讲解了 Cassandra 的所有功能特性和使用方法；实战性强，不仅包含大量示例代码，而且还设计了一个完整的在线交易系统实例；有一定的深度，不仅结合源代码分析了 Cassandra 的底层机制和工作原理，而且还精心总结了一些关于 Cassandra 的最佳实践。

全书一共分为 13 章，首先简单介绍了 NoSQL 的优势，以及几种具有代表性的 NoSQL 数据库的功能特性；其次详细讲解了 Cassandra 的安装和配置、数据模型和排序规则、编程接口等基础知识；接着以迭代的方式演示了一个基于 Cassandra 的在线交易系统的完整开发过程，很好地将基础理论融入到了实践中；紧接着结合源代码分析了 Cassandra 的集群机制、内部数据存储结构、数据更新机制、数据读取机制、数据压缩机制、启动流程等与 Cassandra 的底层机制和工作原理相关的内容；最后讲解了 Cassandra 在分布式环境中的应用、与 Hadoop 的整合，以及相关的最佳实践。附录中包含了本书示例的源代码以及在 Eclipse 环境中编辑和修改 Cassandra 的源代码方法。

本书适合所有对 Cassandra 感兴趣的读者阅读。通过本书，不仅能全面掌握 Cassandra 的基础知识和使用方法，还能深入理解 Cassandra 的底层机制和工作原理，以及它在复杂现实环境中的应用。

封底无防伪标均为盗版

版权所有，侵权必究

本书法律顾问 北京市展达律师事务所

图书在版编目 (CIP) 数据

.Cassandra 实战/郭鹏著. —北京: 机械工业出版社, 2011.5

ISBN 978-7-111-34164-2

I. C… II. 郭… III. 关系数据库 - 数据库管理系统 IV. TP311.138

中国版本图书馆 CIP 数据核字 (2011) 第 067289 号

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: 陈佳媛

北京市荣盛彩色印刷有限公司印刷

2011 年 6 月第 1 版第 1 次印刷

186mm × 240mm · 20 印张

标准书号: ISBN 978-7-111-34164-2

定价: 59.00 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) 88378991; 88361066

购书热线: (010) 68326294; 88379649; 68995259

投稿热线: (010) 88379604

读者信箱: hzjsj@hzbook.com

前言



为什么写本书

最早开始接触 Cassandra 是在 2010 年年初，那时 Cassandra 才刚刚进入人们的视线。由于传统的数据库已经无法满足项目业务的需求，公司便派我去寻找和尝试其他的数据库解决方案。当时有两种备选方案：Hadoop 项目中的 HBase 和 Facebook 开源的 Cassandra。

在对比了各方面的资料和分析了项目的业务需求之后，我们最终选择了配置简单、部署方便，使用高效的 Cassandra。与 HBase 不同，Cassandra 是一套独立的系统，不需要搭建和了解 HDFS 及 Zookeeper 集群即可开始使用，并且可以在 Windows 系统中直接进行测试。所以，我们在很短的时间之内就完成了功能和性能的评估，为后期的项目实施做好了技术储备。

到 2010 年 6 月份，部门开始正式搭建实时数据中心，需要将公司前台数据库产生的业务数据定期抽取到实时数据中心，提供给业务部门的相关人员使用。于是我们开始设计 Cassandra 的业务数据存储结构，并开始分析 Cassandra 内部的实现机制和源代码，从 Cassandra 的工作原理上指导我们的业务数据存储结构设计和测试方案。在整个设计和测试的过程中，我们遇到的一些问题，都可以通过活跃的 Cassandra 开发社区和开放的源代码找到答案。而且，我们还可以通过修改少量的源代码来修复一些 BUG 和增强系统管理的特性。最终，这些修改也都在开源社区提供的后续版本中得到了体现。得益于活跃的开源社区，

Cassandra 的功能和版本也在不断地更新。尤其是在最新的 Cassandra 0.7.x 版本中，提供了大家都非常期待的二级索引与在线更新 Schema 的功能。

Datastax 公司作为 Cassandra 的主要开发成员，开始为 Cassandra 的使用者提供相应的服务支持，并开发出了一系列强大的工具，进一步提高了 Cassandra 的实用性。另外，在 2011 年的 5 月，Datastax 将提供一款振奋人心的工具——Brisk。通过 Brisk，Cassandra 将取代 Hadoop 项目中的 HDFS 和 HBase 的角色，直接与 Hadoop 项目中的 MapReduce 和 Hive 项目进行集成，从而提高整个海量数据分析系统的性能和易用性。

相信未来 Cassandra 能够给我们带来更多的惊喜。

本书面向的读者

初中级程序员。他们可以从本书中的实际例子了解 Cassandra 的基本概念，以及如何通过各种编程接口在 Cassandra 中写入和读取数据，如何建立二级索引，如何动态修改 Schema 信息，如何通过 MapReduce 做海量的数据分析，等等。

中高级程序员。他们可以通过本书对源代码的分析和讲解来了解 Cassandra 中的内部实现原理，从而能选用更加高效的实现方式，调整最适合的 Cassandra 集群运行配置，增强程序的整体运行性能和稳定性。

DBA 和系统运维人员。他们可以通过本书中的运维管理等内容，了解如何在 Cassandra 集群中安全高效地增加或减少服务器，以及如何恢复集群的错误，等等。

系统架构师。他们可以根据本书所讲的内容掌握 Cassandra 能做什么，不能做什么，适合做什么，从而为不同项目做出最合适的架构选型。

联系作者

正如大家所知，开源技术的更新速度非常之快，Cassandra 亦是如此。尽管我们已经很努力地将最新的内容融入其中，但仍难免会遗留部分内容。所以，如果你发现本书存在任何错误和问题，或者是你对书中的某些内容不理解需要与我们进一步探讨，再或者是有好的意见或建议，都欢迎随时与我取得联系，我的邮箱是：gpcuster@gmail.com。

另外，我的个人博客也会经常更新关于 Cassandra 的最新动态和使用心得等内容，也欢迎大家来这里与我交流。博客的地址是：<http://gpcuster.cnblogs.com>。

致谢

感谢老张、许玉勤和姜迅，是你们的信任与支持给了我接触和学习 Cassandra 的机会和应用 Cassandra 的平台。

感谢童家旺、何勇、任振中、王海和陈晓峰，在学习和应用 Cassandra 的过程中，你们

给予了我很多帮助与支持。

感谢我的家人，一年来，我的大部分原来属于你们的业余时间都给了这本书，感谢你们的理解。

感谢杨福川和曾珊两位编辑为本书付出的耐心与努力，你们为本书提出了很多宝贵的建议，你们的敬业精神令我钦佩。

感谢本书的所有读者，希望你们能从本书中有所收获，大家的认可是我不断前进的动力。

郭 鹏

2011 年 4 月

目 录



前 言

第1章 认识 NoSQL/1

- 1.1 NoSQL 的起源和发展现状/2
- 1.2 为什么要使用 NoSQL/2
- 1.3 开源 NoSQL 产品介绍/3
 - 1.3.1 Key/Value 的 NoSQL 数据库/3
 - 1.3.2 面向文档的 NoSQL 数据库/4
 - 1.3.3 面向列的 NoSQL 数据库/5
 - 1.3.4 面向图的 NoSQL 数据库/6
- 1.4 本章小结/7

第2章 Cassandra 快速入门/9

- 2.1 在 Windows 环境运行单机版 Cassandra/10
 - 2.1.1 配置 JRE/10
 - 2.1.2 配置运行 Cassandra 0.6.x/11

- 2.1.3 配置运行 Cassandra 0.7. x/12
- 2.2 在 Linux 环境运行单机版 Cassandra/14
 - 2.2.1 配置 JRE/14
 - 2.2.2 配置运行 Cassandra 0.6. x/15
 - 2.2.3 配置运行 Cassandra 0.7. x/16
- 2.3 Cassandra 的数据模型/18
 - 2.3.1 Column/18
 - 2.3.2 SuperColumn/18
 - 2.3.3 ColumnFamily/19
 - 2.3.4 Keyspace/20
- 2.4 Cassandra 的数据排序规则/20
- 2.5 配置数据类型/22
- 2.6 使用命令行工具与 Cassandra 交互/23
 - 2.6.1 与 Cassandra 0.6. x 进行交互/23
 - 2.6.2 与 Cassandra 0.7. x 进行交互/24
- 2.7 本章小结/26

第3章 理解 Cassandra 编程接口/27

- 3.1 多语言服务开发框架 Thrift/28
- 3.2 Cassandra 的数据类型/28
 - 3.2.1 Column/28
 - 3.2.2 SuperColumn/29
 - 3.2.3 ColumnOrSuperColumn/29
 - 3.2.4 ColumnParent/29
 - 3.2.5 ColumnPath/30
 - 3.2.6 SliceRange/30
 - 3.2.7 SlicePredicate/30
 - 3.2.8 Deletion/31
 - 3.2.9 Mutation/31
 - 3.2.10 KeyRange/31
 - 3.2.11 KeySlice/32
 - 3.2.12 TokenRange/32
 - 3.2.13 AuthenticationRequest/32
 - 3.2.14 ConsistencyLevel/33
 - 3.2.15 NotFoundException/33

- 3.2.16 InvalidRequestException/34
- 3.2.17 UnavailableException/34
- 3.2.18 TimedOutException/34
- 3.2.19 AuthenticationException/34
- 3.2.20 AuthorizationException/35
- 3.3 Cassandra 的编程接口/35
 - 3.3.1 get/35
 - 3.3.2 get_slice/36
 - 3.3.3 multiget_slice/36
 - 3.3.4 get_count/37
 - 3.3.5 get_range_slices/37
 - 3.5.6 insert/38
 - 3.3.7 remove/38
 - 3.3.8 batch_mutate/39
 - 3.3.9 describe_keyspaces/39
 - 3.3.10 describe_keyspace/39
 - 3.3.11 describe_cluster_name/40
 - 3.3.12 describe_version/40
 - 3.3.13 describe_ring/40
- 3.4 Cassandra 0.7.x 版本新增功能/40
 - 3.4.1 二级索引/40
 - 3.4.2 动态修改 Schema/44
 - 3.4.3 自动清除过期数据/46
- 3.5 本章小结/47

第4章 基于 Cassandra 的在线交易系统/49

- 4.1 需求分析/50
- 4.2 数据模型设计/50
 - 4.2.1 Seller/50
 - 4.2.2 Buyer/51
 - 4.2.3 Product/51
 - 4.2.4 ProductCategory/52
 - 4.2.5 Comment/53
- 4.3 编码实现/54
 - 4.3.1 修改 Keyspace 设置/54
 - 4.3.2 建立 Eclipse 项目/54

- 4.3.3 实体对象实现/55
- 4.3.4 Cassandra 数据操作接口实现/56
- 4.4 系统功能验证/60
 - 4.4.1 BuyerDao 功能验证/60
 - 4.4.2 SellerDao 功能验证/61
 - 4.4.3 ProductDao 功能验证/62
- 4.5 迁移到 Cassandra 0.7.x/65
 - 4.5.1 建立 Eclipse 项目/65
 - 4.5.2 修改编译错误代码/65
 - 4.5.3 新增 Schema 在线定义功能/69
 - 4.5.4 功能验证/70
- 4.6 本章小结/71

第5章 Cassandra 的集群机制/73

- 5.1 一致性哈希/74
 - 5.1.1 理解一致性哈希/74
 - 5.1.2 一致性哈希在 Cassandra 中的应用/77
- 5.2 Gossip: 集群节点之间的通信协议/81
 - 5.2.1 FailureDetector/82
 - 5.2.2 Gossiper/83
- 5.3 集群的数据备份机制/88
 - 5.3.1 EndpointSnitch/88
 - 5.3.2 ReplicationStrategy/91
- 5.4 集群状态变化的处理机制/96
 - 5.4.1 StorageLoadBalancer/96
 - 5.4.2 StorageService/97
 - 5.4.3 MigrationManager/98
- 5.5 本章小结/99

第6章 Cassandra 的内部数据存储结构/101

- 6.1 Cassandra 中的数据存放规则/102
- 6.2 Commilog/102
- 6.3 Memtable/103

- 6.4 SSTable/105
 - 6.4.1 Filter 文件/105
 - 6.4.2 Index 文件/107
 - 6.4.3 Data 文件/109
 - 6.4.4 Statistics 文件/113
- 6.5 系统表空间/113
- 6.6 本章小结/114

第7章 Cassandra 的数据更新机制/115

- 7.1 数据更新流程/116
- 7.2 集群数据更新策略/116
 - 7.2.1 ANY/120
 - 7.2.2 ONE/121
 - 7.2.3 QUORUM/121
 - 7.2.4 LOCAL_QUORUM/121
 - 7.2.5 EACH_QUORUM/121
 - 7.2.6 ALL/121
- 7.3 二级索引/122
 - 7.3.1 为什么需要二级索引/122
 - 7.3.2 Cassandra 二级索引更新过程/123
- 7.4 本章小结/124

第8章 Cassandra 的数据读取机制/125

- 8.1 数据读取流程/126
 - 8.1.1 弱读取/126
 - 8.1.2 强读取/128
- 8.2 集群数据读取策略/131
 - 8.2.1 ONE/132
 - 8.2.2 QUORUM/132
 - 8.2.3 LOCAL_QUORUM/132
 - 8.2.4 EACH_QUORUM/132
 - 8.2.5 ALL/133
- 8.3 读修复/133
- 8.4 数据缓存/134

8.4.1 RowCache/134

8.4.2 KeyCache/134

8.5 二级索引/135

8.6 本章小结/135

第9章 Cassandra 的数据压缩机制/137

9.1 为什么要进行数据压缩/138

9.2 如何控制数据压缩/138

9.3 数据压缩流程/139

9.4 维护 Cassandra 中的数据/143

9.4.1 数据清理压缩/143

9.4.2 数据一致性校验压缩/144

9.5 本章小结/144

第10章 Cassandra 的启动流程/145

10.1 Cassandra 启动脚本/146

10.2 Cassandra 启动流程/149

10.2.1 配置 log4j/150

10.2.2 读取校验配置文件信息/150

10.2.3 加载所有的数据文件/152

10.2.4 修复数据/154

10.2.5 启动 Gossiper 服务/155

10.2.6 判断是否需要 Bootstrap 操作/156

10.2.7 监听 Thrift 端口，提供 Thrift 服务/157

10.3 本章小结/157

第11章 在分布式环境中使用的 Cassandra/159

11.1 在 Linux 环境中搭建与使用 Cassandra 集群/160

11.1.1 配置 JRE/160

11.1.2 部署 Cassandra 可执行文件/161

11.1.3 修改 Cassandra 配置文件/162

11.1.4 启动 Cassandra/163

11.2 Cassandra 运行配置项详解/166

- 11.3 Cassandra 集群的运行和维护/175
 - 11.3.1 查看集群的运行情况/176
 - 11.3.2 添加节点/179
 - 11.3.3 删除节点/181
 - 11.3.4 移动节点/183
 - 11.3.5 数据维护/185
- 11.4 本章小结/187

第12章 Cassandra 与 Hadoop 的整合/189

- 12.1 Hadoop 快速入门/190
 - 12.1.1 Hadoop 简介/190
 - 12.1.2 HDFS/190
 - 12.1.3 Map/Reduce/192
 - 12.1.4 配置单机版 Hadoop/194
 - 12.1.5 编写 Map/Reduce 程序/195
- 12.2 为什么要整合 Cassandra 与 Hadoop/200
- 12.3 使用 Map/Reduce 导入数据到 Cassandra 中/200
- 12.4 将 Cassandra 中的数据作为 Map/Reduce 输入/205
- 12.5 本章小结/209

第13章 Cassandra 最佳实践/211

- 13.1 避免 Cassandra 自身的限制/212
 - 13.1.1 不要盲目使用 Super Column/212
 - 13.1.2 硬盘的容量大小限制/212
 - 13.1.3 注意系统大小限制/212
- 13.2 数据压缩策略/213
- 13.3 使用高级的客户端/213
 - 13.3.1 Pycassa/213
 - 13.3.2 Hector /215
 - 13.3.3 FluentCassandra/218
 - 13.3.4 Cassandra /220
 - 13.3.5 phpcassa /221
- 13.4 负载均衡/222
 - 13.4.1 随机选取/222

13.4.2 缓存集群信息/222

13.5 谨慎使用二级索引/223

13.6 通过 JMX 监测 Cassandra/223

13.7 调整 JVM 启动参数/229

13.8 使用适合的系统配置参数/231

13.9 本章小结/232

附录 A 在 Eclipse 中修改 Cassandra 源代码/233**附录 B CassSeller 代码/243****附录 C CassSeller - 0.7 代码/273**



第 1 章

认识 NoSQL

本章内容

- NoSQL 的起源和发展现状
- 为什么要使用 NoSQL
- 开源 NoSQL 产品介绍
- 本章小结

1.1 NoSQL 的起源和发展现状

对于 NoSQL 这个新兴的名词，大家的理解不尽相同。在网站 <http://nosql-database.org/> 上对 NoSQL 有一个较为全面的解释：“下一代的数据库产品应该具备这几个特点：非关系型的、分布式的、开源的、可以线性扩展的。”这类数据库最初的目的在于提供现代网站可扩展的数据库解决方案，它开始于 2009 年年初，目前正在快速发展。这种类型的数据库具有以下特点：自由的 Schema，数据多处备份，简单的编程 API，数据的最终一致性等。所以，将这种类型的数据库称为 NoSQL（不仅仅是 SQL，全称为“not only sql”）数据库。

NoSQL 数据库并不是要取代现在广泛应用的传统数据库，而是采用一种非关系型的方式解决数据的存储和计算的问题。

由于 NoSQL 中没有像传统数据库那样定义数据的组织方式为关系型的，所以只要内部的数据组织采用了非关系型的方式，就可以称之为 NoSQL 数据库。

目前，可以将众多的 NoSQL 数据库按照内部的数据组织形式进行如下分类：

- ☐ Key/Value 的 NoSQL 数据库
- ☐ 面向文档的 NoSQL 数据库
- ☐ 面向列的 NoSQL 数据库
- ☐ 面向图的 NoSQL 数据库

不同的数据组织适合于不同的应用场景，后面将进行介绍。

1.2 为什么要使用 NoSQL

SQL 语言和关系型数据库（MySQL、PostgreSQL、Oracle 等）是通用的数据解决方案，占有绝大多数的市场。不过在最近兴起的 NoSQL 运动中，涌现出一批具备高可用性、支持线性扩展、支持 Map/Reduce 操作等特性的数据产品，它们具有如下特性：

- ☐ 频繁的写入操作、相对较少的读取统计信息的操作（如网站访问计数器），应该使用基于内存的 Key/Value（键/值）存储系统（如 Redis）或者是具备本地更新特性的文档存储系统（如 MongoDB）。
- ☐ 海量数据（如数据仓库中需要分析的数据）适合存储在一个结构松散、分布式的文件存储系统中，如 Hadoop。
- ☐ 存储二进制文件（如 mp3 或者 pdf 文档）并且能够直接为用户的浏览器提供下载功能，可以使用 Amazon S3。
- ☐ 临时性的数据（如网站的 session、缓存 HTML 页面信息等）适合存储在 Memcache 中。
- ☐ 如果希望数据具备高可用性，并且能够将数据丢失的风险降到最低，同时整个系统具备线性扩展的能力，可以考虑使用 Cassandra 和 HBase。

使用这些数据产品并不是要取代原有的数据产品，而是为不同的应用场景提供更多的选择。NoSQL 代表着：**选择合适的方案处理合适的业务场景**。上面介绍的几种 NoSQL 应用场景也许能够帮助我们选择合适的数据存储方案，网上也有不少值得参考的资源。和其他的技术方案一样，选择适合的业务场景才是最重要的。

绝大多数的应用都有非常复杂的应用场景，怎样才能找出一款 NoSQL 产品能够适用于所有的需求？答案是搭配使用多款 NoSQL 产品。传统数据库中的通用（One For All）的情况在 NoSQL 中是不存在的。

如图 1-1 所示，可以在一个网站中使用 4 款数据产品来提供服务。

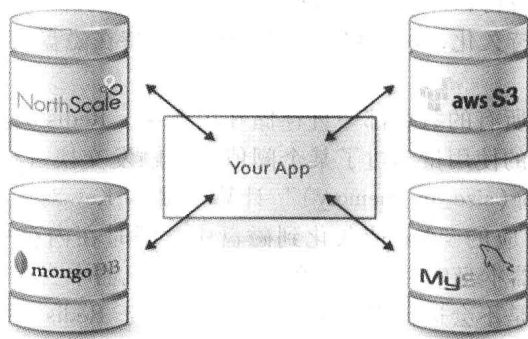


图 1-1 NoSQL 产品组合

- MySQL 用于存储敏感的数据，比如用户的资料、交易的信息等。
- MongoDB 用于存储大量的、相对不敏感的数据，比如博客文章的内容、文章访问次数等。
- Amazon S3 用于存储用户上传的文档、图片、音乐等数据。
- Memcached 用于存储临时性的信息，比如缓存 HTML 页面等。

选择多样的数据存储方案同样有利于提升我们对 NoSQL 的数据产品的理解，帮助我们大量的解决方案中选择最适用的产品，而不是把眼光仅仅放在某一款产品上。

核心的思想是：最适用的才是最好的。

1.3 开源 NoSQL 产品介绍

1.3.1 Key/Value 的 NoSQL 数据库

1. Memcached

Memcached 是国外社区网站 LiveJournal 开发的高性能的内存 Key/Value 缓存服务器，目的是通过缓存数据库查询结果，减少数据库访问次数，以提高动态 Web 应用的速度，从而提高系统的可扩展性。