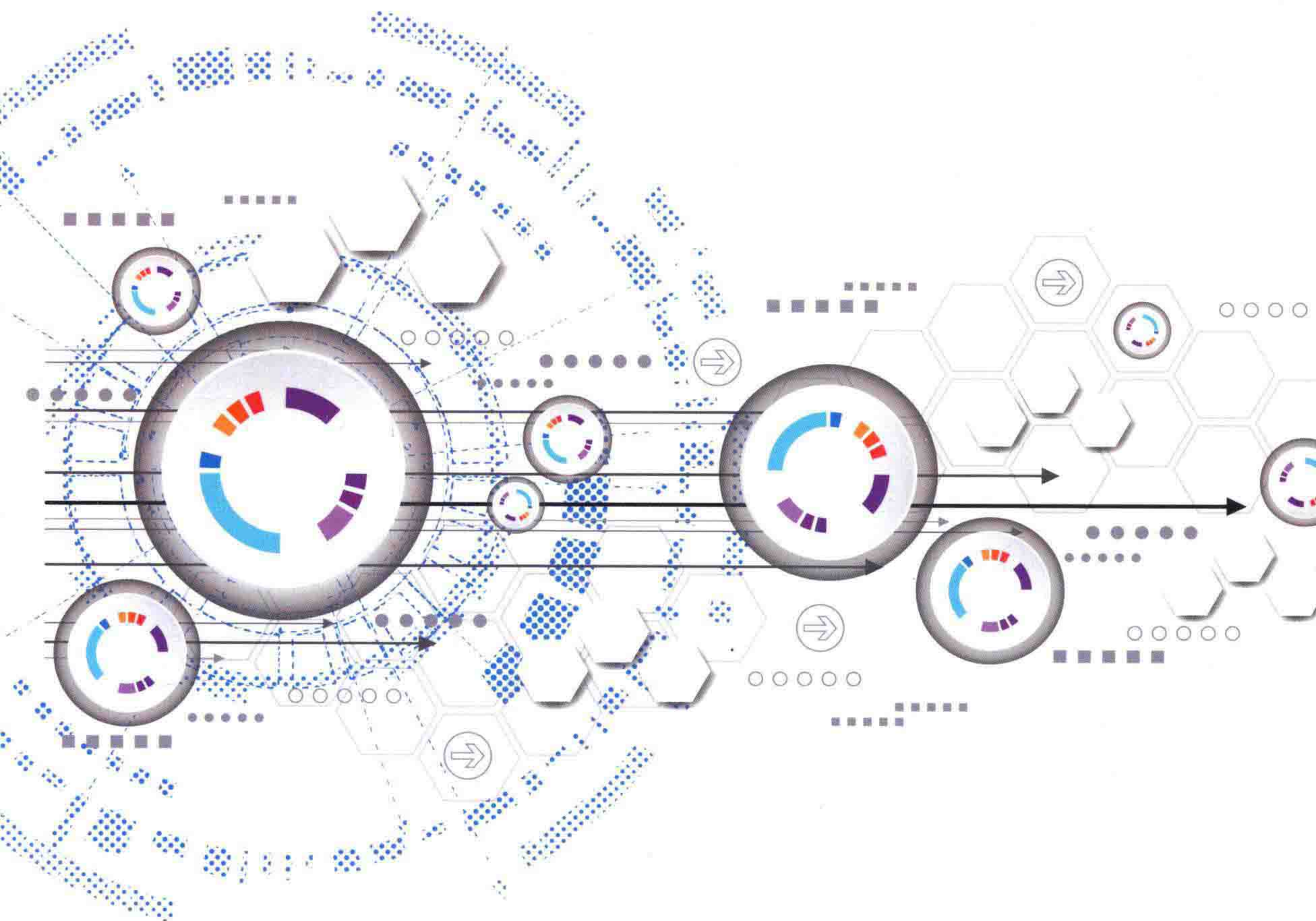




云时代架构系列

**Broadview**<sup>®</sup>  
www.broadview.com.cn



# 可伸缩服务架构

## 框架与中间件

李艳鹏 杨彪 李海亮 贾博岩 刘湫 著



中国工信出版集团



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
http://www.phei.com.cn





云时代架构系列

# 可伸缩服务架构 框架与中间件

李艳鹏 杨彪 李海亮 贾博岩 刘湫 著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING



## 内 容 简 介

本书以高可用服务架构为主题,侧重于讲解高可用架构设计的核心要点:可伸缩和可扩展,从应用层、数据库、缓存、消息队列、大数据查询系统、分布式定时任务调度系统、微服务等层面详细讲解如何设计可伸缩、可扩展的框架,并给出在各个领域解决特定问题的方法论和实践总结。随着本书的出版,我们还开源了4个行之有效的互联网可伸缩框架,包括数据库分库分表 dbsplit、缓存分片 redic、专业的发号器 vesta 和消息队列处理机框架 kclient,每个框架都开箱即用,也可以作为学习互联网平台化框架搭建的素材,更可以作为开发开源项目的示例。

本书的上册《分布式服务架构:原理、设计与实战》详细介绍了如何解决线上高并发服务的一致性、高性能、高可用、敏捷等痛点,本书与上册结合后可覆盖保证线上高并发服务的各个主题:一致性、高性能、高可用、可伸缩、可扩展、敏捷性等,每个主题都是一个方法论。充分理解这些主题,可保障线上服务健壮运行,对实现服务稳定性的  $n$  个 9 有着不可估量的作用。

无论是对于互联网的或者传统的软件工程师、测试工程师、架构师,还是对于深耕于 IT 的其他管理人员,本书都有很强的借鉴性和参考价值,是值得每个技术人员阅读的架构级技术书。

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

可伸缩服务架构:框架与中间件 / 李艳鹏等著. —北京:电子工业出版社, 2018.3  
(云时代架构)  
ISBN 978-7-121-33572-3

I. ①可… II. ①李… III. ①互联网络—网络服务器 IV. ①TP368.5

中国版本图书馆 CIP 数据核字 (2018) 第 018504 号

策划编辑: 张国霞

责任编辑: 徐津平

印 刷: 三河市良远印务有限公司

装 订: 三河市良远印务有限公司

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: 36.25 字数: 812 千字

版 次: 2018 年 3 月第 1 版

印 次: 2018 年 6 月第 3 次印刷

印 数: 7001~12000 册 定价: 109.00 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, 88258888。

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。

本书咨询联系方式: (010) 51260888-819, faq@phei.com.cn。



# 专家评论

能讲明白分布式架构并不容易，本书却做到了。在本书中，不论是对需求场景的剖析，还是对可行方案的仔细研磨及实现，都体现了架构师的专业素养和精益求精。这是一本分布式服务架构方面的好书。

皇包车 CTO 贺伟

本书针对分布式服务架构中常用的缓存分片、数据库分库分表、消息队列、任务调度中间件、RPC、大数据查询系统等技术提供了典型的设计和实现，对我们设计和实现自己的互联网业务系统有重要的参考价值。

《程序员的成长课》作者 安晓辉

互联网业务是爆发式的，其带来的流量压力和对计算能力的要求也是不均衡的，利用廉价的计算机构建分布式计算环境已成为当下的选择。然而，可伸缩技术在带来高性价比的同时，也带来技术上的变革和挑战。大家可以从本书提供的分库分表、缓存分片、消息队列框架、发号器等方案中吸取精髓，快速形成自己的认知，并在工作中积累经验和提升技能，以更好地为公司和团队效力。

企办信息技术有限公司 CTO、云像数字技术顾问 马星光

随着企业业务量的增加，流量洪峰在不断挑战着业务系统的承载能力，设计高并发、可伸缩的系统已成为软件架构师的紧迫任务，而分布式、可伸缩的架构模式已成为抵御洪峰的有效方案之一。本书汇集了作者在多年核心系统开发中的架构及实践经验，以理论与案例相结合的方式展现了分布式系统设计、技术选型、可伸缩架构的设计、框架实现等方面的优秀实践。不管你是在从 0 到 1 构建系统，还是在寻找服务化治理的正确方向，本书都可以帮你解惑。

菜鸟网络技术专家 高春东







# 推荐序一

从传统互联网到移动互联网再到物联网，中国乃至全球的互联网技术在近十年得到了高速发展。作为架构师，我们非常乐意把这些技术传播出去，让更多的人享受互联网技术的红利，让技术拓展商业的边界。阿里巴巴的双 11 技术已经越来越成熟，因为阿里巴巴已经逐步具备了基于云的计算能力，可以轻松应对各种业务压力。

本书的内容包括缓存分片、消息队列框架、数据库分库分表、分布式发号器、大数据查询系统、分布式定时调度任务系统、微服务和 RPC 等；并配备了 4 个开箱即用的开源项目，包括分库分表框架 dbsplit、专业的发号器 vesta、消息队列框架 kclient 和缓存分片的 redic，每个项目都是独立的互联网高并发框架，是构建互联网级项目的重要组件。本书对行业内许多流行的开源项目也有很多介绍，读者可以在其中学到平台架构设计的方方面面，也可以了解开源项目在构思和实现方面的优秀实践。

互联网的技术核心其实源于分布式，这是互联网具备高性能、高并发特性的基础，掌握这些核心内容后，你会发现驾驭技术变得如此轻松！希望艳鹏的这本书可以带领大家掌握分布式的精髓，这本书非常棒！也感谢艳鹏邀请我为本书作序！

技术的世界是如此美妙，作为一名深耕架构的技术人，我由衷地希望有更多的人加入，一起为中国的技术添砖加瓦！

未达科技 CEO、原阿里巴巴中文站首席架构师 焦英俊

## 推荐序二

近十年来，互联网服务在社交网络、搜索、电商、O2O、视频、移动和云计算等领域呈现了井喷式发展，伴随而来的是数千万的日订单量、数亿的日活跃用户、数百亿的日消息发送量等海量的业务规模。支撑这些海量的业务规模的则是基于廉价服务器集群的高可用、可伸缩的分布式互联网技术。

本书以可伸缩服务架构为重点，从理论基础、架构设计、一线行业的实践经验和代码实现细节等方面，系统化地介绍了分布式互联网的高可用、可伸缩技术的核心要点，是一本兼具深度和广度的技术参考书。

虽然本书的主题是“架构”，但是这并不影响本书的易读性，它比大多数同类书都要讲得透彻、明白，也适合有想法、有目标的初中级开发人员阅读。分库分表、Dubbo 源码解析等方面的内容，更可以帮助高级技术人员提升自己的技术实力，以及发挥更大的价值。

架构是在长期的生产活动中经过深度思考所积累下来的优秀实践和可复用的合理抽象，希望你不要错过本书的精彩内容。

开心网副总裁 杨延峰



# 推荐序三

杨彪兄弟找我为本书写序，我欣然从命。我阅读了这本佳作的若干个章节，发现它有几个特点：紧扣常见的问题域；结合了开源产品；代码很讲究。这里，我想谈一谈开源与写作的关系，因为我觉得这就是一场修行。

什么是开源？我的粗浅理解就是，开源就是开源产品，是开放了源代码的产品。我从业 16 年，用过不少开源产品，但发现被广泛使用的项目并不都拥有好的代码和充分的测试用例，可见知易行难。可喜的是，国人的开源产品在逐渐增多，也有不少好作品出现，艳鹏的 `vesta-id-generator` 解决的就是分布式系统中常见的 ID 生成问题。

古人谈学习有“眼到”、“手到”、“心到”之说。

“眼到”指的是阅读，但阅读面广并不见得知识就是自己的，例如一个早晚听张靓颖歌曲的粉丝不见得就能发出海豚音。

“手到”指的是要不断练习，当然，这并不是指简单、重复地练习。佛罗里达州立大学心理学家 K. Anders Ericsson 首次提出了“刻意练习”的概念，该练习方法的核心假设是：专家级水平是逐渐练出来的，而有效进步的关键在于找到一系列小任务让受训者按顺序完成，这些小任务必须是受训者正好不会做但又可以学习和掌握的。

我认为，“心到”的一个环节是写作。写作能帮我们梳理得更通畅。从自己懂，到给别人讲明白，再到写出来且别人还能明白，是进阶的关系。

所以，从项目产品实践，到开源，再到写作，就是一场修行。修行是外在的表现、是与同行交流、是与世界对话；修行也是内观，是收获若干思想的结晶、汲取若干灵感的泉水、沉淀若干系统的对白，在其中收获更多的就是作者自己。

开卷有益，愿读者也能在阅读和实践上收获知与行的快乐！

蚂蚁金服高级技术专家 于君泽



# 前言

本书的上册《分布式服务架构：原理、设计与实战》详细介绍了如何解决线上高并发服务的一致性、高性能、高可用、敏捷等痛点。本书延续了高可用服务架构的主题，侧重于讲解高可用架构设计的核心要点：可伸缩和可扩展，从应用层、数据库、缓存、消息队列、大数据查询系统、分布式定时任务调度系统、微服务等层面详细讲解如何设计可伸缩、可扩展的框架，并给出在各个领域解决特定问题的方法论和实践总结。随着本书的出版，我们还开源了 4 个行之有效的互联网可伸缩框架，包括数据库分库分表 `dbsplit`、缓存分片 `redic`、专业的发号器 `vesta` 和消息队列处理机框架 `kclient`，每个框架都开箱即用，且都是独立的互联网高并发框架，是构建互联网级项目的重要组件，也可以作为学习互联网平台化框架搭建的素材，更可以作为开发开源项目的示例。

在写作本书的过程中，我们的多位作者根据自身在互联网大规模、高并发项目中遇到的实际问题，总结了切实有效的方法论和解决方案，最后沉淀出一套适合高并发服务的优秀开源技术框架，其中的方法论可以帮助读者解决实际问题，开源框架可以帮助读者快速搭建可伸缩的互联网高并发项目。无论是对于互联网的或者传统的软件工程师、测试工程师、架构师，还是对于深耕于 IT 的其他管理人员，本书都有很强的借鉴性和参考价值，是值得每个技术人员阅读的架构级技术书。

感谢我的作者团队，杨彪、海亮、刘湜、博岩等无数次地与我一起通宵达旦地赶稿子；感谢汪勤平、史先斌、潘运鹏、张诚、高亮、李继、周伟、曲源等同学在编辑阶段为我阅稿，并提出专业意见；感谢 IT 行业内的重量级大咖焦英俊、杨延峰、右军（于君泽）在百忙之中抽出时间为本书作序；也感谢大作者开涛和安晓辉等同学为本书写评语；感谢各个行业的重量级朋友们对本书的大力支持，包括：冯沁原、贺伟、张义明、马星光、高春东、黄福伟、李雪冰、付红雷等；更感谢加入云时代架构技术社区的小伙伴们持续关注和支持！

在写书的过程中，我们每个人都兢兢业业、勤勤恳恳，每增加一部分内容，都互相审核，尽量保证书中案例的准确性和时效性，确保能为读者带来很有价值的设计经验，并且我们想做



到“授人以渔”，而不是“授人以鱼”。在写作的过程中有辛苦、有快乐、有价值、有成果，我们希望能持续地为读者带来经验、解决方案、架构思路和快乐。

感谢电子工业出版社博文视点张国霞编辑的认真态度和辛勤工作，本书在经过作者们及国霞编辑大半年的努力后最终顺利出版。

最后，感谢我的妻子和父母在我写书期间对我的全力支持和帮助。

李艳鹏

2018年3月

轻松注册成为博文视点社区用户（[www.broadview.com.cn](http://www.broadview.com.cn)），扫码直达本书页面。

- **下载资源：**本书如提供示例代码及资源文件，均可在 下载资源 处下载。
- **提交勘误：**您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动：**在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/33572>





# 目 录

|       |                      |    |
|-------|----------------------|----|
| 第 1 章 | 如何设计一款永不重复的高性能分布式发号器 | 1  |
| 1.1   | 可选方案及技术选型            | 2  |
| 1.1.1 | 为什么不用 UUID           | 2  |
| 1.1.2 | 基于数据库的实现方案           | 2  |
| 1.1.3 | Snowflake 开源项目       | 3  |
| 1.1.4 | 小结                   | 4  |
| 1.2   | 分布式系统对发号器的基本需求       | 4  |
| 1.3   | 架构设计与核心要点            | 6  |
| 1.3.1 | 发布模式                 | 6  |
| 1.3.2 | ID 类型                | 7  |
| 1.3.3 | 数据结构                 | 7  |
| 1.3.4 | 并发                   | 9  |
| 1.3.5 | 机器 ID 的分配            | 9  |
| 1.3.6 | 时间同步                 | 10 |
| 1.3.7 | 设计验证                 | 11 |
| 1.4   | 如何根据设计实现多场景的发号器      | 11 |
| 1.4.1 | 项目结构                 | 12 |
| 1.4.2 | 服务接口的定义              | 14 |
| 1.4.3 | 服务接口的实现              | 15 |
| 1.4.4 | ID 元数据与长整型 ID 的互相转换  | 22 |
| 1.4.5 | 时间操作                 | 25 |
| 1.4.6 | 机器 ID 的生成            | 27 |



|       |                                         |    |
|-------|-----------------------------------------|----|
| 1.4.7 | 小结                                      | 32 |
| 1.5   | 如何保证性能需求                                | 32 |
| 1.5.1 | 嵌入发布模式的压测结果                             | 33 |
| 1.5.2 | 中心服务器发布模式的压测结果                          | 33 |
| 1.5.3 | REST 发布模式（Netty 实现）的压测结果                | 33 |
| 1.5.4 | REST 发布模式（Spring Boot + Tomcat 实现）的压测结果 | 34 |
| 1.5.5 | 性能测试总结                                  | 34 |
| 1.6   | 如何让用户快速使用                               | 35 |
| 1.6.1 | REST 发布模式的使用指南                          | 35 |
| 1.6.2 | 服务化模式的使用指南                              | 38 |
| 1.6.3 | 嵌入发布模式的使用指南                             | 41 |
| 1.7   | 为用户提供 API 文档                            | 43 |
| 1.7.1 | RESTful API 文档                          | 44 |
| 1.7.2 | Java API 文档                             | 45 |

## 第 2 章 可灵活扩展的消息队列框架的设计与实现

49

|       |       |    |
|-------|-------|----|
| 2.1   | 背景介绍  | 50 |
| 2.2   | 项目目标  | 50 |
| 2.2.1 | 简单易用  | 50 |
| 2.2.2 | 高性能   | 51 |
| 2.2.3 | 高稳定性  | 51 |
| 2.3   | 架构难点  | 51 |
| 2.3.1 | 线程模型  | 51 |
| 2.3.2 | 异常处理  | 53 |
| 2.3.3 | 优雅关机  | 53 |
| 2.4   | 设计与实现 | 54 |
| 2.4.1 | 项目结构  | 54 |



|       |               |    |
|-------|---------------|----|
| 2.4.2 | 项目包的规划        | 55 |
| 2.4.3 | 生产者的设计与实现     | 57 |
| 2.4.4 | 消费者的设计与实现     | 58 |
| 2.4.5 | 启动模块的设计与实现    | 67 |
| 2.4.6 | 消息处理器的体系结构    | 76 |
| 2.4.7 | 反射机制          | 79 |
| 2.4.8 | 模板项目的设计       | 80 |
| 2.5   | 使用指南          | 82 |
| 2.5.1 | 安装步骤          | 82 |
| 2.5.2 | Java API      | 83 |
| 2.5.3 | 与 Spring 环境集成 | 84 |
| 2.5.4 | 对服务源码进行注解     | 85 |
| 2.6   | API 简介        | 87 |
| 2.6.1 | Producer API  | 87 |
| 2.6.2 | Consumer API  | 88 |
| 2.6.3 | 消息处理器         | 88 |
| 2.6.4 | 消息处理器定义的注解    | 90 |
| 2.7   | 消息处理机模板项目     | 91 |
| 2.7.1 | 快速开发向导        | 91 |
| 2.7.2 | 后台监控和管理       | 92 |

## 第 3 章 轻量级的数据库分库分表架构与框架 93

|       |              |    |
|-------|--------------|----|
| 3.1   | 什么是分库分表      | 94 |
| 3.1.1 | 使用数据库的三个阶段   | 94 |
| 3.1.2 | 在什么情况下需要分库分表 | 95 |
| 3.1.3 | 分库分表的典型实例    | 96 |



|       |                                     |     |
|-------|-------------------------------------|-----|
| 3.2   | 三种分而治之的解决方案 .....                   | 97  |
| 3.2.1 | 客户端分片 .....                         | 97  |
| 3.2.2 | 代理分片 .....                          | 100 |
| 3.2.3 | 支持事务的分布式数据库 .....                   | 101 |
| 3.3   | 分库分表的架构设计 .....                     | 102 |
| 3.3.1 | 整体的切分方式 .....                       | 102 |
| 3.3.2 | 水平切分方式的路由过程和分片维度 .....              | 106 |
| 3.3.3 | 分片后的事务处理机制 .....                    | 107 |
| 3.3.4 | 读写分离 .....                          | 119 |
| 3.3.5 | 分库分表引起的问题 .....                     | 119 |
| 3.4   | 流行代理分片框架 Mycat 的初体验 .....           | 123 |
| 3.4.1 | 安装 Mycat .....                      | 123 |
| 3.4.2 | 配置 Mycat .....                      | 124 |
| 3.4.3 | 配置数据库节点 .....                       | 128 |
| 3.4.4 | 数据迁移 .....                          | 129 |
| 3.4.5 | Mycat 支持的分片规则 .....                 | 129 |
| 3.5   | 流行的客户端分片框架 Sharding JDBC 的初体验 ..... | 138 |
| 3.5.1 | Sharding JDBC 简介 .....              | 138 |
| 3.5.2 | Sharding JDBC 的功能 .....             | 139 |
| 3.5.3 | Sharding JDBC 的使用 .....             | 141 |
| 3.5.4 | Sharding JDBC 的使用限制 .....           | 152 |
| 3.6   | 自研客户端分片框架 dbsplit 的设计、实现与使用 .....   | 153 |
| 3.6.1 | 项目结构 .....                          | 154 |
| 3.6.2 | 包结构和执行流程 .....                      | 155 |
| 3.6.3 | 切片下标命名策略 .....                      | 159 |
| 3.6.4 | SQL 解析和组装 .....                     | 167 |
| 3.6.5 | SQL 实用程序 .....                      | 168 |
| 3.6.6 | 反射实用程序 .....                        | 173 |
| 3.6.7 | 分片规则的配置 .....                       | 177 |



|        |                                                                |     |
|--------|----------------------------------------------------------------|-----|
| 3.6.8  | 支持分片的 SplitJdbcTemplate 和 SimpleSplitJdbcTemplate 接口 API ..... | 179 |
| 3.6.9  | JdbcTemplate 的扩展 SimpleJdbcTemplate 接口 API .....               | 184 |
| 3.6.10 | 用于创建分库分表数据库的脚本工具 .....                                         | 187 |
| 3.6.11 | 使用 dbsplit 的一个简单示例 .....                                       | 192 |
| 3.6.12 | 使用 dbsplit 的线上真实示例展示 .....                                     | 199 |

## 第4章 缓存的本质和缓存使用的优秀实践 201

|       |                            |     |
|-------|----------------------------|-----|
| 4.1   | 使用缓存的目的和问题 .....           | 202 |
| 4.2   | 自相似, CPU 的缓存和系统架构的缓存 ..... | 203 |
| 4.2.1 | CPU 缓存的架构及性能 .....         | 205 |
| 4.2.2 | CPU 缓存的运行过程分析 .....        | 206 |
| 4.2.3 | 缓存行与伪共享 .....              | 208 |
| 4.2.4 | 从 CPU 的体系架构到分布式的缓存架构 ..... | 218 |
| 4.3   | 常用的分布式缓存解决方案 .....         | 221 |
| 4.3.1 | 常用的分布式缓存的对比 .....          | 221 |
| 4.3.2 | Redis 初体验 .....            | 225 |
| 4.4   | 分布式缓存的通用方法 .....           | 229 |
| 4.4.1 | 缓存编程的具体方法 .....            | 229 |
| 4.4.2 | 应用层访问缓存的模式 .....           | 233 |
| 4.4.3 | 分布式缓存分片的三种模式 .....         | 235 |
| 4.4.4 | 分布式缓存的迁移方案 .....           | 238 |
| 4.4.5 | 缓存穿透、缓存并发和缓存雪崩 .....       | 244 |
| 4.4.6 | 缓存对事务的支持 .....             | 246 |
| 4.5   | 分布式缓存的设计与案例 .....          | 248 |
| 4.5.1 | 缓存设计的核心要素 .....            | 248 |
| 4.5.2 | 缓存设计的优秀实践 .....            | 250 |
| 4.5.3 | 关于常见的缓存线上问题的案例 .....       | 253 |



|       |                        |     |
|-------|------------------------|-----|
| 4.6   | 客户端缓存分片框架 redic 的设计与实现 | 257 |
| 4.6.1 | 什么时候需要 redic           | 258 |
| 4.6.2 | 如何使用 redic             | 258 |
| 4.6.3 | 更多的配置                  | 258 |
| 4.6.4 | 项目结构                   | 260 |
| 4.6.5 | 包结构                    | 261 |
| 4.6.6 | 设计与实现的过程               | 261 |

## 第 5 章 大数据利器之 Elasticsearch 268

|       |                     |     |
|-------|---------------------|-----|
| 5.1   | Lucene 简介           | 269 |
| 5.1.1 | 核心模块                | 269 |
| 5.1.2 | 核心术语                | 270 |
| 5.1.3 | 检索方式                | 271 |
| 5.1.4 | 分段存储                | 273 |
| 5.1.5 | 段合并策略               | 275 |
| 5.1.6 | Lucene 相似度打分        | 278 |
| 5.2   | Elasticsearch 简介    | 286 |
| 5.2.1 | 核心概念                | 286 |
| 5.2.2 | 3C 和脑裂              | 289 |
| 5.2.3 | 事务日志                | 291 |
| 5.2.4 | 在集群中写索引             | 294 |
| 5.2.5 | 集群中的查询流程            | 295 |
| 5.3   | Elasticsearch 实战    | 298 |
| 5.3.1 | Elasticsearch 的配置说明 | 298 |
| 5.3.2 | 常用的接口               | 300 |
| 5.4   | 性能调优                | 305 |
| 5.4.1 | 写优化                 | 305 |



|       |          |     |
|-------|----------|-----|
| 5.4.2 | 读优化      | 308 |
| 5.4.3 | 堆大小的设置   | 313 |
| 5.4.4 | 服务器配置的选择 | 315 |
| 5.4.5 | 硬盘的选择和设置 | 316 |
| 5.4.6 | 接入方式     | 318 |
| 5.4.7 | 角色隔离和脑裂  | 319 |

## 第 6 章 全面揭秘分布式定时任务 321

|       |               |     |
|-------|---------------|-----|
| 6.1   | 什么是定时任务       | 322 |
| 6.2   | 分布式定时任务       | 341 |
| 6.2.1 | 定时任务的使用场景     | 342 |
| 6.2.2 | 传统定时任务存在的问题   | 342 |
| 6.2.3 | 分布式定时任务及其原理   | 344 |
| 6.3   | 开源分布式定时任务的用法  | 347 |
| 6.3.1 | Quartz 的分布式模式 | 347 |
| 6.3.2 | TBSchedule    | 356 |
| 6.3.3 | Elastic-Job   | 365 |

## 第 7 章 RPC 服务的发展历程和对比分析 377

|       |               |     |
|-------|---------------|-----|
| 7.1   | 什么是 RPC 服务    | 378 |
| 7.2   | RPC 服务的原理     | 379 |
| 7.2.1 | Sokcet 套接字    | 379 |
| 7.2.2 | RPC 的调用过程     | 380 |
| 7.3   | 在程序中使用 RPC 服务 | 382 |