Broadview®
www.broadview.com.cn

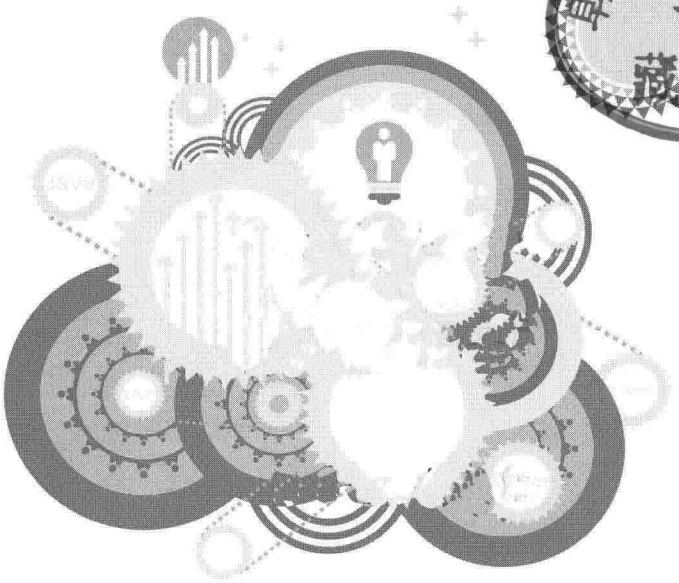
分布式服务框架
原理与实践

李林锋 / 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

分布式服务框架
原理与实践

李林锋 / 著



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

内 容 简 介

本书作者具有丰富的分布式服务框架、平台中间件的架构设计和实践经验，其主导设计的华为分布式服务框架已经在全球数十个国家成功商用。书中依托工作实践，从分布式服务框架的架构设计原理到实践经验总结，涵盖了服务化架构演进、订阅发布、路由策略、集群容错和服务治理等多个专题，全方位剖析服务框架的设计原则和原理，结合大量实践案例与读者分享作者对分布式服务框架设计和运维的体会；同时，对基于 Docker 部署微服务以及基于微服务架构开发、部署和运维业务系统进行了详细介绍。

本书适合架构师、设计师、软件开发工程师、测试工程师，以及其他对互联网分布式架构感兴趣的相关人士阅读。

图书在版编目（CIP）数据

分布式服务框架原理与实践 / 李林锋著. —北京：电子工业出版社，2016.1
ISBN 978-7-121-27919-5

Ⅰ. ①分… Ⅱ. ①李… Ⅲ. ①分布式计算机－研究 Ⅳ. ①TP338.8

中国版本图书馆 CIP 数据核字（2015）第 308395 号

责任编辑：董　英
印　　刷：北京京师印务有限公司
装　　订：北京京师印务有限公司
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：19.5　　字数：415 千字
版　　次：2016 年 1 月第 1 版
印　　次：2016 年 3 月第 2 次印刷
印　　数：4001~7000 册　　定价：65.00 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。

序一

IT 的体系架构在历史上经历了几次大的变化。从主机瘦客户机时代，到 Client Server 兴起，然后过渡到 Browser Server 的架构，再到移动+云计算+大数据的大热。

总结起来，IT 的核心变迁轨迹是在客户端不断提升体验，易联易用，而在服务器端则是不断追求性能和成本优化改进。近几年，还有一个非常明显的趋势是技术的成熟度和融合度不断提高，移动、云计算领域平台型的公司（Android、iOS、AWS）使得整个 IT 能力的使用成本很低，进入速度非常快，现在的高中生也可以利用手头的工具非常快速方便地参与到软件构建中来，这在以前是不可想象的。移动互联网兴起以后，大概在短短 5 年内，世界上绝大部分原来在 PC 端可以满足的需求都由移动端的应用实现了一遍。IT 已经变成了一个快速消费品，而不是一个奢侈品。

技术的进步使得 IT 的敏捷性大大提升，但是对于一个大型系统来说，如何能够降低系统的复杂度，提升敏捷性是关键而又头疼的问题。我们也看到一些通用的标准和最佳实践已经建立起来了，降低模块之间的耦合度，提升组件的内聚性，规范对外的接口，实现分布式的系统架构，把一个大型系统通过服务化的方式规划治理起来，已经成为一个共识。一个现代的大型 IT 系统，服务可以多至十万、百万级，如此众多的服务，从设计、开发、运行、编排、维护到治理，每一个环节都需要大量深入仔细的考虑，才能够运转起来。我们可以把这样一个系统比喻成一个城市，城市里面有成千上万的公司，每一个公司都有自己的业务来往，同时又需要现代化的交通、电力、通信、金融等体系的支持。无论是小公司还是大公司，都依赖于整个城市的运作和治理体系。公司和城市是相辅相成的关系。IT 系统里面的业务模块和服务化框架也是相辅相成的关系。服务化框架对于一个大型 IT 系

是不可或缺的。

业界有很多介绍服务化理念和技术点的文章和书籍。但真正能够在理论、实践、技术要点、眼界多方面全面覆盖的资料,还是比较缺乏的。我很高兴看到林锋能够总结自己在理论、产品和客户实践多方面的认知和经验，为读者奉献《分布式服务框架原理与实践》一书，深入浅出地介绍分布式服务的概念、体系和关键技术点。希望这本书能够帮助你了解分布式服务框架，掌握分布式服务体系和技术要点，同时也能实践服务化给你的 IT 系统带来的敏捷。

黄省江

华为软件 PaaS 平台&云中间件技术总监

序二

容器 SDN 技术与微服务架构实践从 20 世纪末期的第一波互联网浪潮开始，软件架构的主流就逐步从 Office 这类纯客户端软件逐步过渡到服务端的架构设计。与传统的客户端设计相比，服务端的架构设计更关注伸缩性、可用性和可维护性。很可惜的是，现在市面上讲语言、讲算法、讲设计模式或者讲某一门独立的技术的书都不少，但服务端架构设计的书却寥寥无几。

从私下沟通的结果来看，大部分互联网企业都没有解决好上面的几个问题。正如建筑设计的现代化是从结构工程师开始的，软件设计的现代化会从架构师开始，希望李林锋这本书能够帮助广大的架构师或者有志于成为架构师的人掌握好这些知识，让架构师能够带领开发团队构建出稳定、安全、可维护、可伸缩的合格产品，让架构师在软件开发现代化这条路上起到领路人的作用。

本书最末一章讲述了最近很火的微服务架构，微服务架构的思想包含了对以前种种架构模式的反思，也通过 Docker、Mesos 等技术变成第一个可以轻松产品化的架构思想。我相信再过几年对于开发人员，特别是服务端的开发人员，他们所面临的开发模式将与现在的开发模式有很大的差异，这种差异我觉得甚至会大过程序可以有服务端这个概念的引入。

在微服务架构下，开发的门槛将进一步降低，分布式将更加自然而不是依赖于艰难的设计，运维的负担也将降低至一个极低的水平。我们可以期望通过微服务架构，从想法到产品的距离将更短，也能期待涌现出更多让人叹为观止的产品，还能期待能出现些我们现在完全无法预测的技术和产品。

李道兵

七牛云首席架构师

前　言

2008 年 9 月份，我有幸参与了一个华为软件公司的国内 Top3 项目，作为一名有经验的开发，项目经理安排我负责整个系统的实时交易和后台服务端设计。

尽管有 ERP 软件开发经验，但是第一次参与这么大项目的架构设计和开发，压力还是非常大，经常夜不能寐。

好在公司当时已经研发了 Java Web 框架，它基于 Spring + Struts + iBatis 构建，利用公司成熟的 MVC 框架我们很顺利地完成了项目的开发和交付，并最终成功上线运行。

随着业务的发展，用户数和需求逐步增多，团队规模越来越大，我们遇到了很多棘手的问题：

1）代码重复率高：一些业务层的公共功能，被多个模块重复开发，导致研发成本上升，代码质量下降，架构腐化，为后续系统的运维和新功能的开发带来巨大的挑战。

2）需求变更困难：由于长流程无法有效拆分、代码重复率高等因素，导致每次需求变更就影响一大片，需要做大量的回归测试来保证质量，需求的交付周期被拉长。

3）部署效率低：业务没有拆分，很多功能模块都打到同一个 war 包中，一旦有一个功能发生变更，就需要重新打包和部署；巨无霸应用由于包含功能模块过多，编译、打包时间比较长，一旦编译过程出错，需要根据错误重新修改代码再编译，耗时较长。

4）学习成本高：业务流程是由一长串本地接口或者方法调用串联起来的，臃肿而冗长，而且往往由一个人负责开发和维护。随着业务的发展和需求变化，本地代码在不断的迭代和变更，最后形成了一个个垂直的功能孤岛，只有原来的开发者才理解接口调用关系和功能需求，一旦原有的开发者离职或者调到其他项目组，这些功能模块的运维就会变得非常困难。

5）缺乏统一的 RPC 框架：由于 Web 框架只提供了 HTTP/HTTPS 协议，例如 SOAP、SMPP 等协议栈需要业务自行集成第三方的开源框架，超时重发、网络断连等底层故障需要在应用上层统一封装和处理，工作繁琐而且容易出错，对业务开发人员的技能要求也非常高。

随着公司业务的不断发展，传统 MVC 架构已经无法再满足业务对平台的诉求，因此在 2010 年下半年我们开始研发新的 SOA 中间件，它包括企业集成总线 ESB、流程编排引擎 BPM、RPC 通信框架等。新的 RPC 通信框架底层封装了 Java NIO 通信框架 Netty、常用的序列化/反序列化框架，以及为应用层提供线程池和消息调度器，基于 RPC 通信框架，业务可以快速的实现跨进程的远程通信，而不需要关心底层的通信细节，例如链路的闪断、失败重试等，极大的提升了应用的开发效率。

在很长一段时间，自研的 RPC 框架成为业务首选的 Java 服务端框架。

随着 RPC 框架的推广和使用日益深入，一些新的公共需求被反馈过来：

1）依赖管理：当服务越来越多时，服务 URL 配置管理变得非常困难，希望有一个统一的服务注册中心管理服务的依赖关系。

2）透明路由：通过订阅发布机制，消费者只需要关心服务本身，并不需要配置具体的服务提供者地址，实现服务的自动发现。

3）服务治理：业务失败之后的放通处理，超时时间控制、流控等常用运维功能，希望能够独立出一个服务治理中心，统一对集群各节点的服务做在线治理，提升治理效率，保障服务 SLA。

4）其他……

为了解决这些问题，以 RPC 框架为核心，我们构建了全新的分布式服务框架，相比于传统 RPC 框架，它提供了如下新特性：

1）基于注册中心的服务订阅/发布机制，支持服务自动发现和健康状态检测。

2）集群容错。

3）依赖解耦，全配置化开发，对应用零侵入。

4）服务治理，包括服务降级、服务调用链跟踪、服务上线审批和下线通知等。

5）服务化最佳实践等。

分布式服务框架不仅仅包含核心的运行时类库，还包括服务划分原则、服务化最佳实践、服务治理、服务监控、服务开发框架等，它是一套完整的解决方案，用来协助应用做服务化改造，以及指导用户如何构建适合自己业务场景的服务化体系，将服务化的价值发挥到极致。

基于分布式服务框架，业务终于可以把全部精力都放到应用层的逻辑开发，研发效率、系统可靠性都得到了极大的提升。目前，华为电信软件主要解决方案几乎所有的 Java 系统都基于分布式服务框架构建，底层的基础框架实现了统一。

最近一年多来，随着 DevOps 和以 Docker 为首的容器技术的发展，微服务架构逐渐流行起来，微服务架构的流行有其必然的历史原因，它是敏捷开发、基础设施服务化、DevOps 和互联网行业快速发展的综合产物。亚马逊 AWS、Netflix 等都是微服务的成功实践者，相信未来国内越来越多的大型应用也会演进到微服务架构。

华为软件公司的 Java 架构经历了传统的 MVC 垂直架构-RPC 框架-分布式服务框架，目前正在向 Docker + 微服务方向演进，整个服务化架构的演进历程也是业界技术变迁的一个缩影。

在这 7 年的演进历程中，我有幸全程参与了相关框架的架构设计和核心模块的开发，深有感触。在此希望将自己设计、开发和运维的相关经验分享出来，为初学者和相关经验人士提供一些启发，汲取相关经验，少走些弯路。

能够完成本书需要感谢很多人，首先感谢华为公司给我提供了足够大的舞台，感谢华为 PaaS 平台&中间件团队领导和同事莫晓军、望岳和王世军等，以及与我在分布式服务框架团队一起战斗过的开发、测试和资料。

其次要感谢我的家人，你们一直在背后默默的支持我。感谢参与本书编辑的英姐、美工以及其他人员，你们的辛苦换来了本书的如期上市。

最后要感谢所有的读者，你们的支持和鼓励是我写作本书的动力源泉。

李林锋

2015 年 12 月于南京

目　录