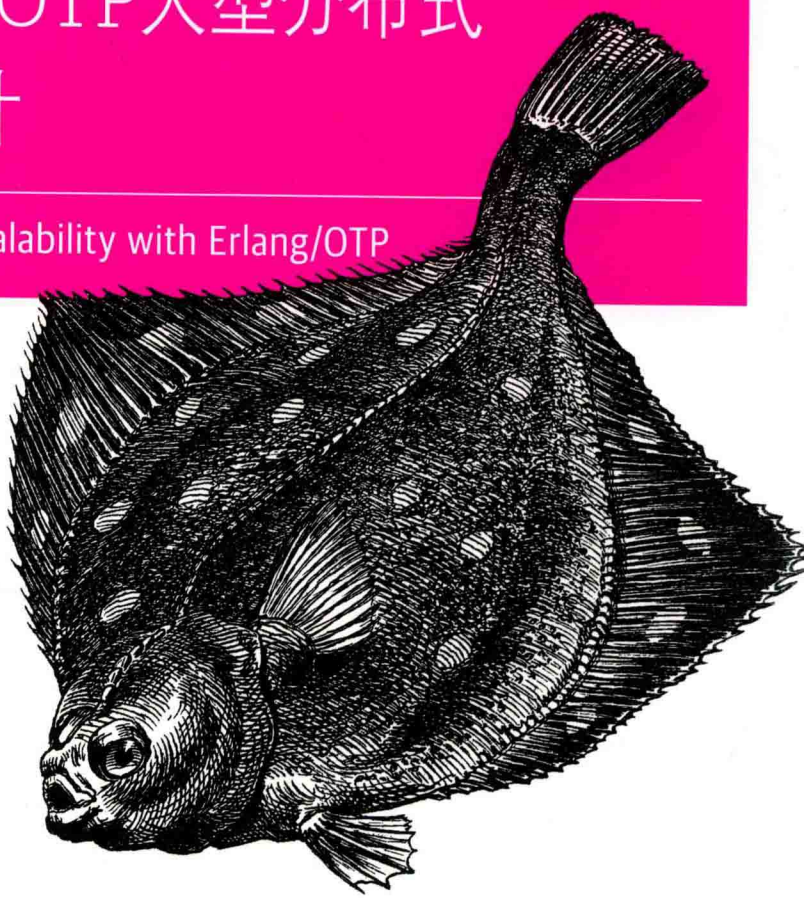
O'REILLY®

Broadview®
www.broadview.com.cn

高伸缩性系统

Erlang/OTP大型分布式容错设计

Designing for Scalability with Erlang/OTP



[瑞典] Francesco Cesarini [美] Steve Vinoski 著
林建入 译

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

O'REILLY®

高伸缩性系统

Erlang/OTP大型分布式容错设计

Designing for Scalability with Erlang/OTP

[瑞典] Francesco Cesarini [美] Steve Vinoski 著
林建入 译

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

内 容 简 介

这是一本罕见的站在核心设计者而非普通开发者角度介绍 Erlang/OTP 系统的权威书籍。两位作者均是深耕分布式计算领域超过20年的专家。本书内容兼具深度与广度，不仅带领读者通过一步步实践的方式深入剖析了 Erlang/OTP 中各类核心进程的行为模式的设计原理，并且还介绍了特殊进程、自定义行为模式、发行包制作等高级主题。除此之外，本书还用了大量篇幅向读者介绍了 Erlang/OTP 系统中的设计原则、构建分布式系统的方法，以及在此基础上实现容错和规模伸缩所需了解的相关知识。

对于任何一位渴望基于 Erlang/OTP 构建出商业级的分布式、高伸缩性、容错型系统的开发者，本书都是不容错过的经典之作。

版权贸易合同登记号　图字：01-2017-5971

图书在版编目（CIP）数据

高伸缩性系统：Erlang/OTP大型分布式容错设计 /（瑞典）弗朗西斯科·切萨里尼（Francesco Cesarini），（美）史蒂夫·温斯基（Steve Vinoski）著；林建入译. —北京：电子工业出版社，2018.6

书名原文：Designing for Scalability with Erlang/OTP

ISBN 978-7-121-33747-6

Ⅰ.①高… Ⅱ.①弗… ②史… ③林… Ⅲ.①程序语言—程序设计 Ⅳ.①TP312

中国版本图书馆CIP数据核字（2018）第036188号

策划编辑：张春雨
责任编辑：刘　舫
封面设计：Karen Montgomery　张　健
印　　刷：北京天宇星印刷厂
装　　订：北京天宇星印刷厂
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：29　　字数：636千字
版　　次：2018年6月第1版
印　　次：2018年6月第1次印刷
定　　价：115.00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。

O'Reilly Media, Inc.介绍

O'Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 Make 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版、在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar 博客有口皆碑。”

——Wired

“O'Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O'Reilly Conference 是聚集关键思想领袖的绝对典范。”

——CRN

“一本 O'Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去 Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

感谢 Alison、Peter 和 Bump 宝宝的耐心和支持。

——*Francesco*

感谢 Dooley、Ed 教授了我做法；感谢 Cindy、Ryan、Erin、Andrew 和 Jake 告诉了我原因。

——*Steve*

感谢 Joe、Mike 和 Robert 当年为拨通那个电话所发明的这一切。

——*Francesco & Steve*

推荐序一

随着IT系统的蓬勃发展，万台级别的机器相互协作组成一个复杂的业务系统已经不再罕见。特别是云计算的普及，使得基础计算资源的获取变得便捷高效，在几个小时内计算资源被申请，各种各样用途的节点被部署，节点被有机互联协作，系统被监控和运维，业务系统被迅速扩容，发现的问题被迅速解决。

而实现这样复杂业务的分布式系统一直是一个很热很大的话题，系统的鲁棒性、可伸缩性、高可用性等每个话题单拎出来都可以讲几天几夜。领域固有的复杂性让很多人望而生畏，更糟糕的是现实对于问题的解法各不相同，很难找到系统讲解这些知识的著作来解释Why、How，更别提最佳实践了。

2005年我在开发VOIP系统的时候，偶然发现我接手的分布式系统，虽然是用C语言编写的，但很奇怪地用了进程、消息传递、状态机的架构，而不是熟悉的线程、并发、switch/case传统解法。当时我负责这套架构的基础设施，为了达到可伸缩、稳定，做得异常痛苦，后来了解到这套架构来自爱立信。2007年，从CSDN的一篇介绍文章开始，爱立信开源的Erlang为人所知，而它号称的系统几个9的可靠性一直是我不能理解的，当我花了很多时间深入了解了Erlang语言、OTP框架、VM实现后突然豁然开朗，疑惑终于被解开了。

Erlang语言有自己的哲学和世界观，其试图构建一个和人类系统很像的系统：每个人都是一个进程，人和人之间是有边界的，人和人是有组织关系的，人和人通过消息来协作，每个人都会犯错，每个人的表现都需要被监督。围绕着这个哲学，Erlang对现实世界描述的进程、消息、协作被具象化，函数式不变语法保证了实体的独立性。它的VM被设计成一个类UNIX的平台可移植虚拟机系统，对于CPU、IO、内存的能效管理接近线性的效率，在节点间屏蔽了消息传递的各种复杂性。特别难能可贵的是，自省第一天就被充分植入系统。

这还不够，毕竟会造车和会开车是两件事情，Erlang用OTP框架来强制对分布式的最佳

实践。分布式系统应该如何被设计，如何实现鲁棒性、高可用性，如何伸缩都有精心的引导，只要遵循这些 OTP 行为，很容易达到你的设计目标。

复杂应用生命周期里会有大量部署、运维、监控、问题排查、升级、下线等烦琐的事情要做，特别是系统规模大了以后，业务平滑变成很大的挑战。OTP 很体贴地来救驾，它提供了各种各样的组件让生命周期管理不那么痛苦，可圈可点的是，成体系的平滑热升级能力是从头到脚设计的。作为一个在电信行业里经受二十多年挑战的系统，系统的稳定可靠一直是被刻意追求的，也是值得信赖的。

作者 Francesco 从 R1 就开始参与系统的设计和实现，是一个资深的开发者，对于系统的设计哲学理解深入骨髓，同时他非常热心地创办了 Erlang Solutions，通过各种会议和书籍教授和帮助用户，积累了丰富的经验。而 Steve 在分布式系统上同样沉浸多年，也是资深 Erlang 用户，开发了大量系统。在这本书中，他们把自己的经验、心得倾囊而出，特别是对于大型复杂系统的设计、权衡和妥协的实践，可以帮助读者将知识迅速转化成生产力。

祝学得开心!

褚霸

阿里云研究员

推荐序二

“多少年过去了”，每次提起 Erlang，我都会如此感慨，也都会想起这些年跟 Erlang 有关的一幕幕场景。

想起第一次使用 Erlang，惊叹于跟面向对象完全不一样的函数式编程范式；想起第一个真正的工作项目，用 Erlang 调用 C++ 搜索库实现的智能问答机器人。也更会想起，邀请 Joe Armstrong 来公司交流的场景，想起请教 Steve Vinoski 关于调试修改 Yaws 的邮件，想起遇到问题时，翻霸爷的博客、看 Erlang Factory 各种分享的那些时刻。

虽然比起工作中遇到的各种各样的挑战，这样的场景并不算多，但也正因为如此，才使得它们在时光里显得格外明亮。

这种感觉，相信很多朋友都深有体会，你对 Erlang 有多少喜爱，就有多少叹息。毕竟它作为一门小众语言存在了太久，而且在可预见的未来一段时间，似乎也不会有太大的改变。

而前面提到的这些人，和未提到的很多社区成员一起，仍然在数年甚至数十年如一日地像灯塔般照耀着这个社区，为这门语言的发展做着自己的贡献。多少年来，无数程序员在他们的感召下，学习及体验并发、容错、函数式编程，又在他们的帮助下，实现了一个又一个永不停机的系统。

“多少年过去了”，也是 Francesco 在写完这本书之后的感慨。好在这份持续贡献的心，终究还是让这本书诞生于世。又借助于本书编辑和译者的辛勤努力，国内的开发者得以看到它的中文版本。

他们都是值得被铭记和感谢的。

也因此，当我看到本书的两位作者是 Francesco 和 Steve 时，我并不意外，而且内心的感觉是，这本书由他们写再好不过了。Steve 不用多说，做 Erlang 许多年，为 OTP、Yaws 等众多知名开源项目做出了重要贡献，之前他是公认的 CORBA 宗师，在分布式系统领

域拥有丰富的实践经验。而 Francesco 是 Erlang Solutions 的创始人和技术总监，后者不仅一直举办 Erlang Factory Conference，还一直在 Erlang 开发和培训方面持续耕耘。

重要的是，这两位作者都有足够的毅力写作，也愿意付出多年的精力来完成这样一本事无巨细的著作。因为 Erlang/OTP 涉及的内容非常广泛，它不仅包含 Erlang 语言的各种特色用法，也有针对不同场景的典型设计模式，还包括在分布式系统设计和实现过程中用到的诸多理论。尤其是后面两者，都是需要一定的基础才能理解透彻从而高效使用的。

在过去的三年半中，我们做了中国最大的即时通信 PaaS 平台，也是整个团队真正大量实践 Erlang 的时期。千万级同时在线的用户，不仅意味着大量的长链接，有缓存和存储组件的普遍使用，也有高并发的内部系统间调用。而我们始终坚持使用最简单的 Erlang，坚持使用 OTP 原生而不是第三方的组件，坚持通过测试和实践检验而不是难辨真假的传说来选型。

事实证明，这样的坚持是值得的，同时也证明了 Erlang/OTP 在现代互联网架构下是经得住考验的。我们做到了核心系统整体 99.99% 的可用性，这一方面要归功于容错监督机制的良好运行，更是因为使用了热更新机制的发行包工具，让我们可以真正快速迭代，系统的升级和回滚都可以在秒级完成。

我们用到的大多数工具在本书中都有专门的章节讲述，所以对于本书的常规用法，我的建议是通读后留在案头作为参考。不求全部记住，只求用的时候能想起来就好。

对于 Erlang 的初学者来讲，在学会语言后，下一步就是实现业务系统，OTP 无疑是一把现成的武器。事实上，它不仅是现成的武器，更是一把好用的武器，对于大多数资深的开发者来讲都是足够好用的。

一些有抱负的开发者可能会发现，这些工具的实现原理十分简单，手写一个并不是什么难事。这里我也多做一个提醒，魔鬼都在细节里。当你把所有情况都考虑周全，很大可能写出来的跟这个差不多。所以，相信 OTP 的高效，用它，用好它，做更好玩的事情吧，让这个社区继续独立有趣地走下去。

多少年过去了，我依然可以回忆起刚开始用 Erlang/OTP 的感觉，一如古龙书中所述：就像黑暗中忽然有了光，月光，圆月。

一乐（梁宇鹏）

块聊链创始人

译者序

这是一本值得每个 Erlang 程序员阅读的好书，因为它深入透彻地讲解了 Erlang 程序员进阶过程中最为关键的一环——对 OTP 框架的深入理解。

众所周知，与一些火热的流行语言相比，Erlang 书籍一直以来数量不多，并且其中大多数以介绍入门级内容为主，意在引起读者的兴趣。尽管这些书籍也各具特色，不少堪称佳作，但对于真正需要从事 Erlang 进行开发的程序员来说，仅了解基本内容是远远不够的。因此长期以来，要想进一步学习，就需要自己在 Erlang 文档中摸索。客观地说，Erlang 拥有非常完善的文档，并且其源代码很容易读懂，因此只要有好奇心，你可以深入了解任何你感兴趣的细节。但是，了解细节是一回事，了解细节背后的设计动机又是另一回事。从这个角度来看，文档与源代码虽然将核心机制毫无保留地呈现在我们面前，但仍然有所欠缺。欠缺的是一条线索，一条能够贯穿系统设计中重大决策背后动机的线索。而本书的出版，终于补上了这缺失的一环。

我想强调，本书对于 OTP 的讲解，并非局限于讲解其“用法”——如果真是如此，那么阅读文档便足够——而是更注重其“原理”。此原理既是指其工作流程，更是指其设计动机。正因为如此，本书的内容才显得独特而可贵。具体来说，本书的前半部分，在作者的带领下，读者可跟随其指导重新实现 OTP 中核心的构成要件。这一过程并非平庸的代码罗列然后逐句解读，而是首先从背景出发，遵循一定的设计理念，先带领读者设计出一个小型的模型，其虽然看似简陋，但已能够实现基本功能，然后进一步指出其不足，并将其改进为符合 OTP 理念的实现。与 OTP 内部真正的实现相比，显然读者的实现依然是简陋的，但是却深刻地反映了真实系统运作时的核心原理。倘若读者有心，能够认真跟着作者的指点完成整个过程，那么不仅能够轻松理解这些 OTP 框架中核心构件的使用方法，知其然；并且能够明白其背后的工作原理，知其所以然。

完整介绍完 OTP 后，本书的篇幅已过大半。我想，本书内容即使自此戛然而止，也不愧列入经典之列了。但两位作者 Francesco 和 Steve 却选择更进一步，带领读者探索更深的主题。

于是在第 11 章，我们不仅可学习到 OTP 系统的核心设计原则，并且还跟着作者一步步手工完成了 OTP 发行包（Release）的制作。这一章我特别喜欢。因为我和很多读者一样，能使用 rebar3 之类的工具自动完成发行包制作，但对其中的过程却不甚清楚。作者为什么要花费很长的篇幅介绍如何手工制作发行包呢？因为通过这个过程，读者能够深入理解 Erlang 系统的构成及其启动过程。如果不了解这些内容，就无法理解和应对一些比较棘手的启动阶段的问题，同时也丧失了利用这个过程完成一些定制化能力的机会。并且，理解这些内容，对于那些想进一步探索 Erlang 核心机制的硬核程序员来说，也极有帮助。这一章我个人认为是本书中特别重要的一章，并且实践性极强，建议读者跟随作者的指引一步步完成实验。

而第 12 章，更进一步，向读者介绍了如何进行系统升级。我相信很多人都听过 Erlang 支持热更新，但是对它的认识仅限于模块级的热更新。你想知道如何升级 application，甚至升级整个 Erlang 虚拟机吗？事实上一点也不难，作者将告诉你最佳做法，你不用担心升级时 application 间的依赖、数据库模式变化等诸多问题。一切答案都在本书中。剩下的第 13 章到第 16 章同样不容错过。分别介绍了分布式系统架构方案、容错性设计、规模伸缩方法，以及监视与抢救性支持等内容。

每一章都很精彩，我很想向读者一一介绍，但我想更好的做法是让读者自己去领略吧。在这篇译者序里我就不“剧透”了。

交流与反馈

我在 GitHub 上建立了一个项目，如果你希望与我或者其他读者交流，这是一个不错的方式。其中还整理了一些与本书相关的资料（代码、勘误和相关文档等）链接，方便查阅。这个项目会长期维护，欢迎随时来访，共同交流。

https://github.com/Jianru-Lin/scalabilitywitherlangotp

回顾与感谢

作为一篇译者序来说，感谢部分一般的做法是优雅而礼貌的寥寥数语带过即可。少则三两句，多则一两段足矣。先是感谢编辑，然后是感谢家人和朋友。这样做或许没有问题，但我仔细想想倘若多年后自己翻起本书，却看不到自己当时真正想说的话，会很遗憾吧。所以还是想把自己真实的想法写下来。

两年前张春雨老师找到我，问我有没有兴趣翻译一本 Erlang 的书。我当然开心地答应了，因为我很喜欢 Erlang。但由于个人业余时间有限，最终花了两年时间才完成。这期间并

非一帆风顺，有很多波折，主要是我个人工作环境发生变化，业余时间有时候很紧张，而且身体有一段时间也有一些不适，综合各种因素导致翻译的进度时好时坏。拖稿也从偶尔有之到家常便饭，编辑从时不时查阅进度，到不间断地催稿。

刚开始编辑是客气地催稿，我则客气地回应。但是次数多了，有时候确实给编辑着急得不行："这都周三了，说好上周末交的呀？""最迟这周五，不能再拖了！"我也"压力山大"，只能赶紧抽时间处理，有时候一再拖延，真的是很不好意思回编辑的微信了。于是有"林老师，干什么去了？弄完了吗？""稿子什么时候能给，急死了！"刚开始是张春雨老师催，后来和刘舫老师两位一起交替催，催得厉害了，有一天，刘舫编辑自己笑着打趣说："天天追杀啊"。大家都笑了，我也笑了。

我记得很多次，我白天工作忙抽不出时间，只能深夜处理。于是把稿子发给刘舫编辑的时候，已经是凌晨四五点了。可是令我惊奇的是，经常很快就收到了回复。聊了两句后，我准备休息，心里嘀咕着，刘舫老师现在还醒着？深夜交稿尚且如此，周末和节假日更别说了。想想自己尚且有周末休息，可是编辑却一直处于工作状态，顿时觉得自己的辛苦其实和他们还是比不了的。所以对于催稿这件事，也不能说是编辑施压译者，其实编辑同样不容易。

说起来还闹过一个笑话，因为我偶尔会去北京，于是也想见见张春雨老师和刘舫老师。于是有一次就和刘舫老师提起见面吃饭的事，当时文字交谈过程中感觉刘舫老师似乎不太方便。后来才知道原来刘舫老师是女编辑！我和人家沟通了一段时间连对方性别都没搞清楚，真是十分尴尬。但是这也不能完全怪我，因为每次我发的稿件刘舫老师总是细细阅读后给出很多专业的修改意见，让我觉得很厉害，潜移默化习惯性地以为是男同胞。怪只能怪自己有错误的刻板印象。而且后来发现很多技术书籍的编辑都是女性，心里就更惊讶和钦佩了。

其实张春雨老师联系我之前，我早就知道他了，因为我读过的不少优秀引进书籍的策划编辑都是他，我书架上的《游戏引擎架构》和《Clojure 编程》就是（后者的责任编辑还是刘舫老师），这些书都属高水准作品。而其中每一本的译者序里都有"感谢张春雨老师"的话语，这就是为什么我对他有印象的原因。提到这一点，张春雨老师幽默地开玩笑说"呵呵，他们没有感谢我，是我自己加进去的"，把我和刘舫老师都逗乐了。

好吧，不管怎么说本书终于翻译完了。我写了这么长的一段，其实只是希望下次您看到书籍时，不仅要注意到作者和译者的名字，也应当留意编辑们的名字。作为译者，我可以留下一些文字。但作为编辑，就很少让读者意识到他们辛苦的付出了。所以，感谢张春雨老师和刘舫老师，你们辛苦了。

另外，要特别感谢我的妻子，是你一直在催稿，催得比编辑还紧（二位编辑万万没想到吧？

其实你们有一个不花钱的手下天天跟着我，我逃得过你们却逃不过她），所以现在终于完成了，而不是再多三个月。当然，当我完成这一切时，你比我还要开心。说起来我还欠你一条比目鱼，咱们说好了完成这本书后就买一条尝尝的。你还说，很期待书印刷出来后，捧在手里的感觉，你要看看我在里面是怎么感谢你的。仔细想想这些年我成功做到的每一件事情背后其实都离不开你的支持，但我觉得这还不够，我们还要一起再翻译更多书，一起完成更多想做的事，一起去更多想去的地方。我写下这些文字的时候你就躺在我身后，不亦乐乎地玩着手机。我没有让你看到我写的内容，不过我猜你看到这段的时候一定会高兴的。因为我也是。

最后，感谢我的父母和家人，尤其是保慈林女士、保慈芬女士、张绍光先生，是你们令我能接受好的教育，并教会我勇敢追求渴望的人生。而我的丈母娘在我工作繁忙期间，在生活上给了我很多关照，减轻了我的很多负担，为我节约了很多时间，对顺利交稿功不可没，我心里十分感激。

说得有些冗长，深感抱歉，但这些都是我的真实想法。因为我想即使再过很多年，读起这段文字还是会很快乐。我很满足。

林建入

2018 年 5 月 6 日深夜于海口

序言

本书为你提供的，是一名自 1996 年从 R1 版便开始接触 Erlang 的爱好者，钻研十多年后终于成长为一位分布式系统专家，在这一过程中他所获得的宝贵知识和经验，让你明白为何 Erlang/OTP 能够使你更容易地专注应对系统开发中那些真正的挑战。

通过描述如何构建 OTP 行为模式（behavior）以及为什么需要 OTP 行为模式，我们向你展示了如何使用它们构建独立节点。这就是最初我们向 O'Reilly 提供的草案，内容仅限于此。但是在编写本书时，我们决定将内容更进一步，记录下我们的实践经验、设计决策过程和架构分布式系统时常见的一些问题。通过我们所做的这一系列设计选择和折中，这些模式为我们提供了 Erlang/OTP 众所周知的可伸缩性、可靠性和可用性。与流行的观点相反，这一切并非魔法般地开箱即得的，但获得它们确实比其他任何——非语义级别模拟 Erlang 的，或者不是运行在 BEAM 虚拟机上的——编程语言要容易得多。

Francesco：为什么写这本书

有人曾告诉我，写书有点像生孩子。一旦你写完一本书，拿到纸质图书的那一刻，脑子里有的只是兴奋和激动，而曾经付出的艰辛将统统被忘掉，只渴望着赶快开始写另一本。自从 2009 年 6 月首次拿到纸质书以来，我一直有编写 *Erlang Programming*（O'Reilly）续作的打算。在我开始这个项目时，我还没有自己的孩子，但最终这一项目花了如此长的时间以至于我的第二个孩子都已经快出生了。美好的事物值得我们等待，谁说不是呢？

与第一本书一样，本书是围绕我在 Erlang Solutions 公司所做的 OTP 培训材料中的示例编写的，我将使用这些示例时我的讲解和教学过程转化为文字。每当完成一章后，我都会回顾并确保那些学生较难理解的部分我的讲解是清晰的。最好的学生通常会问的那些问题最终被放到了补充材料部分，而篇幅较长的章则被分解成一些较短小的章。原本一切都很顺利，直到我们到达第 11 章和第 12 章时，因为发行包制作和软件升级没有一种统一的方法，而是存在许多种工具。有些工具需要集成到客户的构建和发布过程中，而其他一些则是开箱即用的，还有一些已无法使用。对于任何想要理解系统的发行包制作

和软件升级包括其幕后工作原理的人，我们希望这两章成为他们的终极指南。此外，如果你必须对现有工具进行故障排除或编写自己的工具，其中还介绍了你所需要了解的内容。

但真正的麻烦从第 13 章才开始。由于没有任何示例和培训材料，我发现自己必须将头脑中的内容形式化，将构建 Erlang/OTP 系统时所采取的方法落实为文档，并尝试将其与分布式计算理论结合起来。最终第 13 章变成了 4 章，并且花了写出本书前 10 章那么长的时间才完成。对于那些购买了早期访问（early access）的读者，我希望没有辜负你们的等待。对于那些明智地等我们写完才购买的朋友，希望你们喜欢这些内容!

Steve：为什么写这本书

我第一次发现 Erlang/OTP 是在 2006 年，当时我正在研究如何能更快、更便宜、更好地开发企业集成软件。无论我从哪个方面考察，Erlang/OTP 都明显优于我和我的同事当时一直使用的 C++ 和 Java 语言。2007 年，我加入了一家新公司，开始在商业产品中使用 Erlang/OTP，事实证明，我之前考察所发现的一切优势都是真的。我教一些同事使用了这种语言，不久后，我们开发的软件比其他大多数人开发的都更强大、更可靠、更容易演进，并且能更快地投入生产环境，甚至与人员规模大得多的 C++ 团队相比优势依然明显。直到今天，我仍然完全信赖 Erlang/OTP 在实践中表现出的令人印象深刻的高效性。

多年来我发表了不少技术资料，而我的目标读者一直都是像我这样的其他从业者。这本书也不例外。在前面的 12 章中，我们提供了许多深入的实践性细节，使开发人员能够充分理解 OTP 的基本设计原则。在这些细节中包含了大量极具实用价值的知识——各类模块、函数和方案等——它们将为你的日常设计、开发和调试工作节省大量时间和精力。在最后的 4 章中，我们将转变方向，聚焦于宏观的层面，探讨可伸缩分布式应用在开发、部署和运行时涉及的各种权衡取舍。由于与分布式系统、容错和 DevOps 相关的知识、方案和需考虑的权衡数量着实庞大，所以要想将这些章节简洁地写出来难度可想而知，但我相信本书为此达到了适当的平衡，既为读者提供了大量好的建议，同时又能避免读者迷失于其中。

我希望这本书能帮助读者提高所开发的软件和系统的质量及效用。

本书的读者对象

本书的目标受众主要针对 Erlang、Elixir 开发人员和架构师——已经完整阅读过一本以上入门书籍，并准备将知识提升到一个新的水平的人。这不是一本带你入门的初等水平的书籍，而是一本涵盖了许多其他书籍未涉及的高级内容、让你远超同行水平的书。其中第 3 章至 第 12 章存在依赖关系，应该顺序阅读，第 13 章至第 16 章也是如此。如果你不需要回顾 Erlang 初级知识，可以跳过第 2 章。

如何阅读本书

本书中的内容兼容 Erlang 18.2。书中涉及的绝大部分功能同样适用于先前版本的 Erlang；针对不适用的功能在书中均有指出。对于未来版本的不兼容性虽然在写书时尚不可知，但将会在本书的勘误页面上进行详细说明，并修复本书 GitHub 仓库中的对应代码。我们鼓励你从我们的 GitHub 仓库下载本书的示例代码，并亲自运行以更好地理解相关知识。

致谢

撰写本书是一个漫长的旅程。在进行这项工作时，我们得到了许多优秀人士的帮助。编辑 Andy Oram 给予我们无数的想法和建议，耐心地指导我们，给予我们反馈，并且不断鼓励我们。Andy，谢谢你，没有你，我们无法完成此书！ Simon Thompson——*Erlang Programming* 一书的合著者帮助了本书的构思和起草，并为第 2 章奠定了基础。非常感谢 Robert Virding 贡献的一些例子。我们还得到了很多读者、审稿人、贡献者的帮助，为我们提供了许多反馈，有了这些，我们才能让每一章的内容充实。在此我们列出他们的姓名，并忐忑地希望没有遗漏任何一位：Richard Ben Aleya、Roberto Aloi、Jesper Louis Andersen、Bob Balance、Eva Bihari、Martin Bodocky、Natalia Chechina、Jean-François Cloutier、Richard Croucher、Viktória Fördős、Heinz Gies、Joacim Halén、Fred Hebert、Csaba Hoch、Torben Hoffmann、Bob Ippolito、Aman Kohli、Jan Willem Luiten、Jay Nelson、Robby Raschke、Andrzej Śliwa、David Smith、Sam Tavakoli、Premanand Thangamani、Jan Uhlig、John Warwick、David Welton、Ulf Wiger 和 Alexander Yong。如果我们在名单中遗漏了您，我们真诚地向您道歉！给我们发一封电子邮件，我们会将您添加进去。对 Erlang Solutions 职员中那些阅读了本书早期撰写过程中的手稿，以及其他早期为本书提供勘误的人，我们必须大声地向你们表示感谢。此外，还要特别感谢所有通过社交媒体渠道鼓励过我们的人，特别是其他作者。你知道我说的就是你们！ 最后，同样重要的一点是，感谢 O'Reilly 的制作、营销和会议团队，不断提醒我们“只要尚未付印，工作就不算结束”。我们非常感谢你们的支持！

本书所用的排版约定

本书中使用了以下排版约定：

斜体（*Italic*）

用于表明新术语、应用程序、URL、电子邮件地址、文件名、目录名和文件扩展名。

等宽字体（`Constant width`）

用于程序清单，以及在段落中对变量、函数名、数据库、数据类型、环境变量、语

句和关键字等程序元素的引用。还用于行为模式（behavior）、命令和命令行选项等。

等宽加粗字体（**Constant width bold**）

用于显示命令或用户手工输入的文本。

等宽斜体（*Constant width italic*）

用于显示应该由用户提供或者根据上下文确定的值。

该图标表示提示或建议。

该图标表示一般性的备注说明。

该图标表示提醒或警告。

中文版书中切口以“<⬭>”表示原书页码，便于读者与英文原版图书对照阅读，本书的索引中所列的页码也为英文原版图书中的页码。

使用示例代码

补充材料（示例代码、练习材料等）可以从此网址下载：*https://github.com/francescoc/scalabilitywitherlangotp*。

本书的初衷是帮助你完成工作。一般而言，你可以在你的程序和文档中使用本书中的代码。除非涉及大量引用代码，否则你无须联系我们获得许可。例如，编写程序时使用到了本书中的几个代码块，这不需要取得我们的许可。但其他情况例如销售或分发来自O'Reilly 书籍中的示例代码则需要先取得许可。为了回答问题而引用本书中的内容和示例代码不需要许可。将本书中的大量示例代码整合到产品文档中则需要获得许可。

我们感谢但不要求注明出处。出处通常包括标题、作者、出版商和 ISBN。例如：“*Designing for Scalability with Erlang/OTP* by Francesco Cesarini and Steve Vinoski (O'Reilly). Copyright 2016 Francesco Cesarini and Stephen Vinoski, 978-1-449-32073-7.”

如果你觉得你对示例代码的使用不属于上述一般性合理情形，或者对许可范围存疑，欢