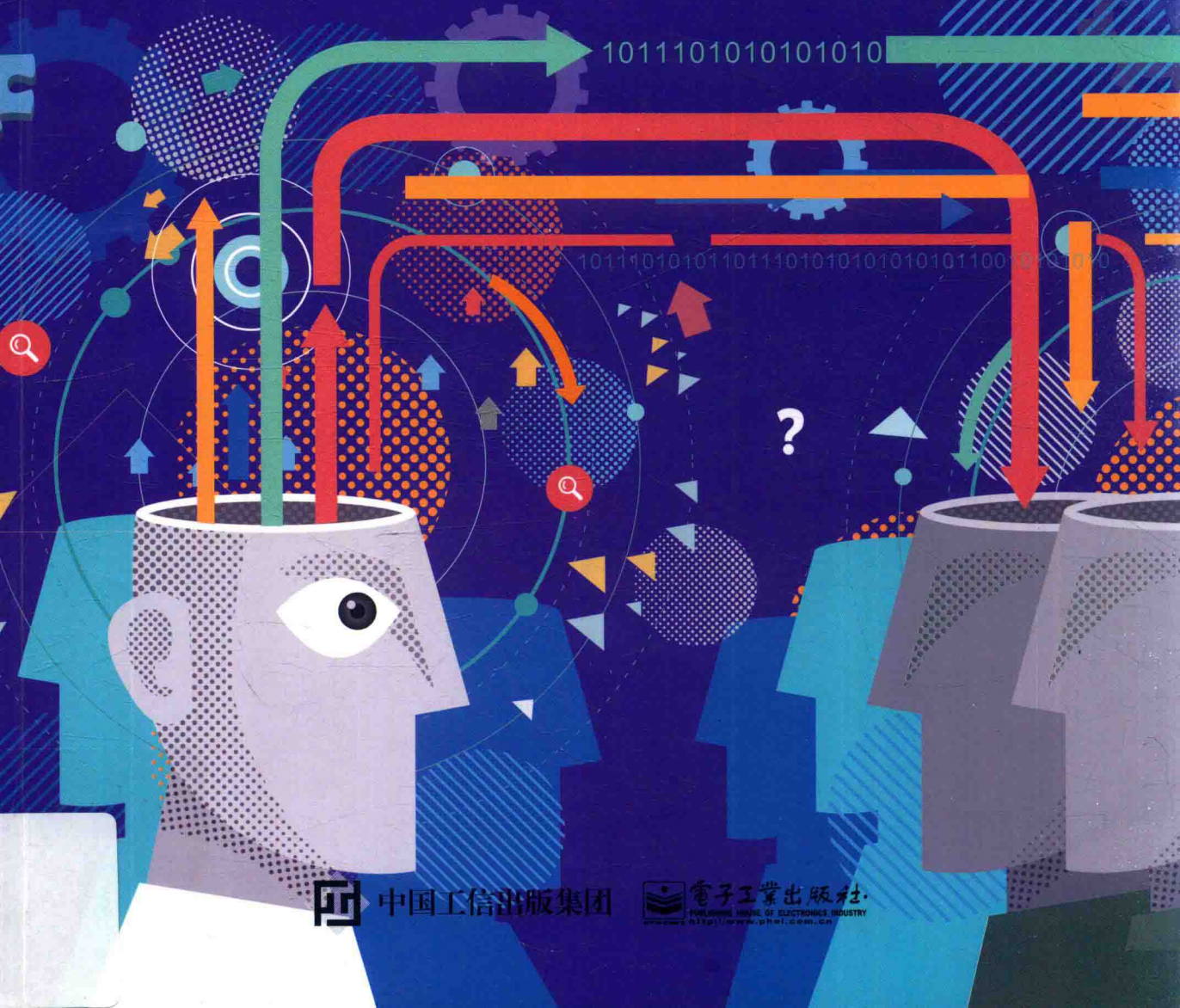
Broadview
www.broadview.com.cn

深入理解并行编程

Is Parallel Programming Hard, and, if so, What Can You Do About It?

[美] Paul E.McKenney 编著
谢宝友 鲁阳 译



中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

深入理解并行编程

Is Parallel Programming Hard,
and,if so,What Can You Do About It?

[美] Paul E.McKenney 编著
谢宝友 鲁阳 译

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

内 容 简 介

本书首先以霍金提出的两个理论物理限制为引子，解释了多核并行计算兴起的原因，并从硬件的角度阐述并行编程的难题。接着，本书以常见的计数器为例，探讨其不同的实现方法及适用场景。在这些实现方法中，除了介绍常见的锁以外，本书还重点介绍了 RCU 的使用及其原理，以及实现 RCU 的基础：内存屏障。最后，本书还介绍了并行软件的验证，以及并行实时计算等内容。

本书适合于对并行编程有兴趣的大学生、研究生，以及需要对项目进行深度性能优化的软硬件工程师，特别值得一提的是，本书对操作系统内核工程师也很有价值。

图书在版编目（CIP）数据

深入理解并行编程 /（美）保罗·E·麦肯尼（Paul E.McKenney）编著；谢宝友，鲁阳译. — 北京：电子工业出版社，2017.7

书名原文：Is Parallel Programming Hard,and,if so,What Can You Do About It?

ISBN 978-7-121-31508-4

Ⅰ.①深… Ⅱ.①保… ②谢… ③鲁… Ⅲ.①并行程序－程序设计 Ⅳ.①TP311.11

中国版本图书馆 CIP 数据核字（2017）第 105172 号

策划编辑：符隆美
责任编辑：徐津平
印　　刷：三河市华成印务有限公司
装　　订：三河市华成印务有限公司
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：787×1092　1/16　印张：33.25　　字数：850 千字
版　　次：2017 年 7 月第 1 版
印　　次：2017 年 8 月第 2 次印刷
定　　价：129.00 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

声　明

本书代表作者的观点，并不一定代表其雇主的观点。

读者服务

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

- **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/31508

作者序

我希望能够说本书的诞生源于甜蜜和光明，但这无疑是个谎言。和许多需要长年坚持努力的事情一样，本书经过了大量挫折才得以诞生。

你看，大约 10 年前，在并发领域的一个行业专家小组研讨会上，我很荣幸得以提问最后一个问题。一些参会的专家长篇大论地讨论了并行编程的高难度，所以我问为什么并行编程不会在 10 或 20 年内成为司空见惯的事情。大多数小组成员一点都不喜欢这个问题。事实上，第一个小组成员试图用一个简短的回答敷衍了事，但我很容易地做了简短的反驳。无奈，他尝试给出了第二个简短回答，我也继续反驳。几轮之后，他大声喊叫："像你这样的人应该用锤子敲敲头！"我不甘示弱地回答道："那你可要排队才能敲得到。"

我不认为这种交流是特别有启发性的，相反这展示了一个毫无疑问的事实：这位"业内"专家对于并行编程一无所知。不过在场的其他听众却认为这场对话非常有启发性，尤其是那一位感谢我提出这个问题的听众，他的眼里甚至含着泪水。他像学徒一样在 Sequent 计算机系统公司学到了并行编程的诀窍，正如我曾经所做的那样。后来他跳槽去了另一家公司，他的新雇主开始涉足并行编程。出乎他意料之外，事情发展得并非一帆风顺。正如他所说，"我已经足足跟他们说了两年，只要你用正确的办法，这并不是很难，但他们完全不听我的话！"

现在，我们很容易将这个悲伤故事里面的团队作为反面教材。但是在他们的看法中，并行编程等于用你自己的智力伤害自己。除非你了解实现并行性的正确方法，否则在意识到遇到麻烦之前，你越聪明，挖的坑就越深。因此，你越聪明，并行编程看起来就越难。不仅如此，在这件事发生时，极少有人知道如何进行并行编程，这意味着大多数人刚刚开始了解到他们为自己挖的并行编程坑的深度。

即使如此，当这个人用哽咽的声音讲述他的故事，眼泪滑过他的脸庞时，我意识到我不得不做一些事。那就是写眼前这本书，这里面不仅仅浓缩我自己四分之一个世纪的经历，还有其他人加起来数不清几个世纪的经历。

我的母语是英语，英语是我唯一可以声称掌握了的语言。但幸运的是，感谢鲁阳和谢宝友所付出的巨大努力，现在中文版翻译即将面世。我希望这本书不仅可以帮助你学习我所知道的知识，从而不再需要担心并行编程，还能使你能够创建属于自己的并行编程新发现！

Paul E. McKenney

推荐语

在我所看过的各种关于操作系统概念和并行编程的书籍中，我对 Paul 的书评价最高，它不是对学术方法的简单罗列，而是对现代硬件上运行并行系统的各种现实世界问题和面临挑战的细致分析，这一切都源于 Paul 在这一领域的丰富经验和巨大的贡献。

——Opersys CEO，《Embedded Android》作者 Karim Yaghmour

并行编程很难，但阅读 Paul 的书是掌握并行编程最简单的（当然也是最有趣的）办法之一！

——Linux 内核 x86、sched 和 rt-patches 分支的维护者 Ingo Molnar

编程的至高无上境界是毫不费力地驯服 CPU。你正在阅读的是关于各任务在 CPU 进行战争的伟大著作，一旦你开始翻阅，再多的编程挑战也不用怕！

——Linux 内核防火墙 ipchains 和 iptables 的作者，网络货币 prettycoin 的作者 Rusty Russell

对程序员而言，想要了解并行编程中涉及的问题，以及如何正确解决这些问题，本书是不可或缺的。

——《Linux 内核驱动》作者，Linux 内核 stable 分支和其他大量分支的维护者 Greg Kroah-Hartman

这是一本每个并行编程专业人员案头必备的参考书，浓缩了作者在该领域数十年的丰富实践经验。它也是一本学习并行编程的优秀教材，在涵盖主题的广度和深度方面表现优异。极具吸引力的写作风格使得本书的阅读成为非常愉快的体验。

——Facebook 资深工程师，危险指针和无锁内存分配器的发明者 Maged M. Micheal

Linux 内核社区里高手云集，并且里面的人经常个性鲜明，以至于有很多人认为内核社区很不友好。但 Paul 是一个特别亲切、友善和耐心的人，不管是在内核邮件列表里还是面对面交流时。而这本书也体现了 Paul 的这些品质，他以最详尽易懂的方式解释并行编程方方面面的知识。这不表示这本书看起来很轻松，因为并行编程本身就很难。但真正有用的知识大概都没能够轻松获得。

——Linux 内核 cgroups 和 cpuset 分支的维护者，华为 Linux 内核高级工程师 李泽帆

刚看到书名时我在想，并行编程这样一个在计算机领域“古老”且成熟的话题还有什么值得多写的。翻看几页目录后便改变了想法。

该书从并行编程问题的历史背景讲起，一步步引入问题的挑战并带读者游历硬件与软件交互的发展，最后阐述当下并行编程的复杂性。

本书囊括所有系统编程的要素，不仅仅是概念层面的解释，更重要的是深入分析了每个要素存在的必要性及底层原理。对于喜欢钻研的同学或是在业界工作的工程师甚至架构师都是非常好的学习资源。

尽管我在业界有多年的开发设计经验，依然从书中学到很多实用的知识。作者 Paul E. McKenney 用深入浅出地方式将自己在并行编程领域数十年的经验归纳在这五百多页中。译者谢宝友和鲁阳在系统编程上有着扎实的功底，用流畅的语言将本书翻译给广大国内读者。这是一本难得的技术好书！

——VoltDB 研发部总监 石宁

并行编程并没有那么难，如果你花点时间在这本书和它里面的小问题上的话。

——Linux 内核 RCU 代码贡献者 冯博群

Paul 是 Linux 顶级黑客，也是 Linux 社区 RCU 模块的领导者和维护者。他的著作《Is Parallel Programming Hard, And, If So, What Can You Do About It?》首版在 9 年前就发布了。本书主要陈述了在适应多核硬件下提升并行软件的扩展性，避免由于锁竞争所引起的产品性能急剧下降，以及开展多核系统的设计、优化工作。

Paul 所维护的 RCU 模块在 Linux kernel 各个子系统中被大量应用，是保障 kernel 扩展性的基础技术，没有 RCU 就没有 Linux 现在优秀的多核性能和扩展性；在并行计算方面，Paul 对于锁、RCU、SMP、NUMA、内存屏障等并行技术有深刻的了解，兼具近 20 年解决问题的实践经验。中兴同仁翻译此书，对于提升我国开源系统软件的设计水平和开发高端产品，均有重大意义。

——中国开源软件推进联盟主席 陆首群

宝友的“自学成才”路径一直很让我印象深刻，贡献及收获在中兴这样一个正规军遍布的大型通讯上市企业，并通过一年的努力帮助中兴在开源社区提升代码贡献率和质量，又再次让我竖起大拇指！不忘初心的工程师梦想、学术上的坚持，以及职业生涯中的成就，宝友身上的这些闪光的品质都是怀揣梦想的年轻一代工程师们学习的榜样。

——Linaro 全球执行副总裁 大中华区总经理 郭晶

在多核处理器已经成为主流计算架构的今天，理解和掌握并行编程技术，对于相关软件开发人员来说至关重要。《深入理解并行编程》一书系统讲述了并行计算的要点和难点，堪称经典，是入门和学习并行编程的不二推荐。

——Linux IMX 平台维护者 Shawn Guo

这本书举重若轻地将并行编程涉及的软、硬件各个方面的基本原理透彻地呈现在读者面前，相信读者研读和实践后可以对并行编程有庖丁解牛之感。

——Linaro 资深内核工程师 聂军

《深入理解并行编程》全方面讲述了高速缓存、内存屏障、锁、RCU、并发性、实时性等知识，如同少林寺的“洗髓经”，是迈向“武林高手”的必修内功，值得对并行编程感兴趣的计算机从业者、尤其是操作系统底层软件从业者细读。

——Red Hat 资深 Linux 内核工程师 庞训磊

并行编程一直是程序设计的难题，这个难题来源于硬件系统，也来源于人类本身的思维模式。人类的思考模式是线性的，很难做到一心二用，很难在程序设计的过程中自如处理并行化的算法和结构。

此外，并行编程的作用越来越大，AI 的涌现和大数据对计算量的要求导致 GPU、FPGA 及 ASIC 之类异构计算的兴起。这些异构计算都以并行计算为根基，并行计算很可能成为计算领域的下一个风口。

本书探讨了并行计算的根源。从硬件、锁机制、数据分割和 RCU 等多个方面，对并行计算的本质和如何应用做了很多分析工作，对读者理解并行计算和提高对并行计算的掌控力有很大的帮助。

——腾讯高级技术专家 高剑林

推荐序

读着《深入理解并行编程》的样章，我的脑海里不断地浮现出9年前的一幕幕。我在网上寻找操作系统的志同道合者，看到一个税收专业中专毕业者的自荐信，其时他已具有10年的IT行业工作经验，从事过大量手机、通信行业软件研发工作，担任过项目总监研发管理工作，在电信应用开发方面已经做得比较成功。但他对操作系统有浓厚的兴趣、执着的追求，放弃了在高层应用软件方面的既有优势，专注于操作系统的研究。离职在家，利用半年时间开发出一个嵌入式操作系统模型，计划两年内研发一款自研操作系统。有感于他的执着和热爱，我向公司争取破格录取他。我认为做一个操作系统不难，但做生态难，做商业成功难，建议他深入学习开源Linux的技术，站到巨人肩膀上，再结合操作系统团队的商业模式探索，争取把操作系统做成功。于是，他如痴如醉地研究Linux内核，在一年时间里，每天晚上坚持花三个小时以上的时间钻研《深入理解Linux内核》这本书，还将自己的读书心得笔记共享到团队论坛上，并对开源内核进行注解，分享到开源论坛上。

2008年正是多核架构快速发展之时，操作系统的支持参差不齐，驱动、应用开发模式不成熟，既有单态单核单进程的业务应用如何进行重构和演进，方案设计、开发联调、故障排查、系统调优又会遇到很多复杂和棘手的问题，中兴通讯操作系统团队需要支撑公司所有产品、各种CPU架构、各种复杂业务场景，团队面临着前所未有的技术和进度压力。团队成员除了在研发一线通过不断实践进行被动积累和提升外，也加强了主动的理论知识提升，阅读《深入理解并行编程》就是其中之一。令我印象非常深刻的是，多核故障往往比较随机和复杂，难以复现和理解，但以谢宝友为代表的团队成员往往可以通过阅读业务、驱动、内核代码就定位到故障根源，整理出故障逻辑，我认为这与他们的系统理论水平提升是分不开的。非常欣慰的是，我们成功地解决了这个过渡时期涌现的诸如多核内存序相关故障，利用无锁并行编程优化了系统性能。时至今日，我们团队已经从30人发展到数百人，嵌入式操作系统已全面应用于公司所有产品，在全球稳定商用，并且扩展应用到电力、铁路、汽车等领域，2016年获得了第四届中国工业大奖。

另一方面，站在技术的角度来看，在计算机领域，并行编程的困难是众所周知的。

有4、5年编程经验的读者，可能或多或少遇到过并行编程的问题，最著名的问题可能就是死锁。读者需要掌握调试死锁问题的技巧，以及避免死锁问题的编程技术。

喜欢深入思考的读者，在理解并解决死锁问题之后，可能还会阅读并行编程方面的书籍，进一步接触到活锁、饥饿等更有趣的并行编程问题。中兴通讯操作系统团队的同事，就曾经在开源虚拟化软件中遇到过类似的问题：虚拟机容器在互斥锁的保护下，轮询系统状态并等待状态变化。这样的轮询操作造成了进程调度不及时，系统状态迟迟不能变化。这是一个典型的活锁问题。在多核系统越来越普及的今天，类似的活锁问题更容易出现。解决这类问题，需要经验丰富的工程师，借助多种调试工具，花费不少的时间。

但是，并行编程仅仅与锁相关吗？

在摩尔定律尚未失效时，并行编程确实主要与锁紧密相关。但是，我们看看霍金向IT工程师所提出的两个难题：

1．有限的光速；

2．物质的原子特性。

这两个难题最终会将CPU频率的理论上限限制在10GHz以内，不可避免地使摩尔定律失效。要继续提升硬件性能，需要借助于多核扩展。

要充分发挥多核系统的性能，必须提升并行软件的扩展性。也就是说，并行软件需要尽量减少锁冲突，避免由于锁竞争而引起性能急剧下降。这不是一件简单的事情！我们知道，Linux 操作系统在接近 20 年的时候内，一直受到大内核锁的困扰。为了彻底抛弃大内核锁，开源社区近几年内做出了艰辛的努力，才实现了这个目标。即使如此，Linux 内核仍然大量使用不同种类的锁，并且不可能完全放弃锁的使用。

也许你会说，在多核系统中，有一种简单的避免锁的方法，就是原子变量。在某些架构中，原子变量是由单条指令实现的，性能“想必”不差，使用方法也简单。曾经有一位具有十多年编程经验的工程师也表达过类似的观点。在此，有两个问题需要回答。

1．这样的原子操作指令，其性能真的不差？它的执行周期是否可能达到上千个时钟周期？

2．对于多个相互之间有逻辑关联的变量，原子操作是否满足要求？

实际上，多核系统中的并行软件，除了常见的锁之外，还需要使用冒险指针、RCU、内存屏障这样的重量级并行编程工具。这些编程工具都属于“无锁编程”的范畴。

即使在 Linux 内核开源社区工作 10 年以上的资深工程师，也不一定能真正灵活自如地使用RCU、内存屏障来进行并行编程。因此，真正了解并行编程的读者，难免在面对并行编程难题时，有一种“抚襟长叹息”的感觉。

然而，我们知道，有很多重要的应用依赖于并行——图形渲染、密码破解、图像扫描、物理与生物过程模拟等。有一个极端的例子，在证券交易所，为了避免长距离传输引起的通信延迟（理论上，光束绕地球一周需要大概 130ms），需要将分析证券交易的计算机放到更接近证券交易的地方，并且压榨出计算机的所有性能。这样，才能保证达成有利的证券交易。可以毫不夸张地说，对软件性能有苛刻需求的软件工程师和大型软件开发企业，都需要真正掌握并行编程的艺术，特别是“无锁编程”的艺术。一旦真正掌握了，它就会为你带来意想不到的性能提升。曾经有一位著名企业的高级专家，在应用了本书所述的 RCU 后，软件性能提升了大约 10 倍。

本书正是这样一本深入讲解多核并行编程，特别是无锁编程的好书。

首先，本书作者 Paul 具有 40 年软件编程职业生涯，他大部分的工作都与并行编程相关。即使在领导 IBM Linux 中心时，他仍然坚持每天编程，是一名真正的“工匠”。同时，作者也是Linux 开源社区 RCU 模块的领导者和维护者。认真阅读本书后，不得不钦佩于作者在并行编程方面的真知灼见和实践能力。例如作者亲自编写了一个软件用例，来考察 CPU 核之间原子操作和锁的性能，得出一个结论，原子操作和锁可能消耗超过 1000 个 CPU 时钟周期；作者也编写过另外一个关于全局变量的用例，其中一个 CPU 核递增操作一个全局变量，同时在不同的 CPU 核上观察所读到的全局变量值。这个用例向读者展示了多核系统令人惊奇的、反直觉的效果；作者对内存屏障的讲解，特别是内存屏障传递性的讲解，十分深入。这些深入的内容，难得一见，非大师不能为。

其次，这本书也得到 Linux 内核社区和应用软件专家的一致推荐。这些推荐者既包括 Linux 社区大名鼎鼎的 Ingo Molnar、Rusty Russel、Greg Kroah-Hartman、Maged M.Micheal，也包括国内活跃于社区的庞训磊、Shawn Guo 等开源贡献者，还包括 Linaro 开源组织的领导和资深工程师，以及在 BAT 工作多年的高级应用软件专家。

第三，这本书的内容比较全面。除了介绍常见的锁以外，还重点介绍了 RCU 的使用及其原

理，以及实现 RCU 的基础：内存屏障。本书最后还介绍了并行软件的验证，以及并行实时计算等内容。实际上，其中每一部分都是并行编程的宝藏。由于篇幅和难度的原因，作者在当前版本中，将 RCU 部分作了大幅压缩。对 RCU 感兴趣的读者可以阅读早期原版著作。即使如此，本书对 RCU 的讲解也非常深入。对于并行软件的验证，作者提出了不少独特的观点，这些观点和作者多年的编程经验息息相关，与常见的理论著作相比，有一定的新意。形式验证部分，作者以实际的例子，一步一步讲述验证过程，很明显，作者亲自动手做过这种验证。并行实时计算部分，是作者新增的内容，别具一格，值得读者细读。内存屏障部分，是本书一个难点，借助于作者在这方面的功力，需要读者反复阅读，才能真正理解。

第四，这本书讲解得很深入。有些语句，需要读者反复琢磨、推敲，甚至需要多次通读本书才能领会作者的意思。也许，经典书籍的阅读方法均是如此。刚刚开始接触 Linux 内核的读者，不太会喜欢阅读《深入理解 Linux 内核》一书，觉得这本书不易理解。但是，如果你愿意花一年时间，将这本书反复阅读三遍，则会有一种别样的心情。本书也是如此，建议读者在初次阅读时，不要轻易放弃。本书实为并行编程方面不可多得的好书。举两个例子：第一，5.2.2 节中有一句原文是“One way to provide per-thread variables is to allocate an array with one element perthread (presumably cache aligned and padded to avoid false sharing).”。译者将其翻译为“一种实现每线程变量的方法是分配一个数组，数组每个元素对应一个线程（假设已经对齐并且填充过了，这样可以防止共享出现“假共享”）”。第一次阅读本书，可能会不理解括号中那句话，有一种云里雾里的感觉。要真正理解这句话，需要读者仔细阅读本书后面关于 MESI 消息协议部分，参阅更多参考资料。要理解本句中“对齐”和“填充”两个词，也需要深厚的内核功底。第二，14.2.10.2 节，“一个 LOCK 操作充当了一个单方面屏障的角色。它确保：对于系统中其他组件的角度来说，所有锁操作后面的内存操作看起来发生在锁操作之后。LOCK 操作之前的内存操作可能发生在它完成之后。”这句话读起来也比较绕，难于理解，似乎也相互矛盾。实际上，读者需要琢磨“看起来”这个词，它表示其他核看到内存操作的顺序，并不代表内存操作的完成时机。

总之，如果你对并行编程或者操作系统内核有兴趣，或者需要对项目进行深度性能优化，我强烈推荐这本并行编程的经典好书！

中兴通讯操作系统产品部 钟卫东

译者序 1

希望这本著作能够成为经典！

20 年前，当我正式成为一名软件工程师的时候，就有一个梦想：开发一款操作系统。那时候，虽然知道 Linux 的存在，但是实在找不到一台可以正常安装使用 Linux 的 PC。因此只能阅读相关的源码分析书籍而不能动手实践。

我至今仍然清楚记得：大约 10 年前，中兴通讯操作系统团队的钟卫东部长，可能被我对操作系统的热情所感动，不顾我没有上过大学的事实，冒着风险将我招聘到中兴通讯成都研究所。面对 100 多种型号的单板，我既兴奋又惶恐。这些单板涉及 ARM、X86、PowerPC、MIPS、SH、Sparc 等不同的 CPU 架构。从此，我开始了激动人心而有趣的内核之旅。

在之后的 6 年中，我对照 Linux 内核源码，根据《深入理解 Linux 内核》、《深入理解 Linux 网络内幕》、《深入理解 Linux 虚拟内存管理》，以及其他一些 Linux 内核文件系统、网络协议栈方面的书籍，做了 2200 页、87 万字的内核学习笔记，同时将相应的源码注释公布到网络中供自由下载。

然而，这 6 年在看 Linux 内核代码的过程中，以及在工程实践中，总有一个幽灵般的阴影出现在我的脑海中：什么是内存屏障？2011 年，我看过内核源码目录中的文档以后，终于解决了标准内核和工具链一个关于内存屏障的故障。要复现这个故障，项目同事需要在整整一个房间里面摆满单板和服务器，才能搭建一套复现环境，并且需要用多套这样的环境，花费 2 个月时间才能复现一次相关故障。即使在解决了相关故障以后，我仍然觉得自己对内存屏障理解得不深刻。因为内核源码目录下的文档，对内存屏障描述得仍然语焉不详。那个幽灵仍然在脑海中盘旋：到底什么才是内存屏障？

直到有一天，我在办公室晃悠的时候，突然在鲁阳的办公桌上发现一本特别的书——《Is Parallel Programming Hard, And, If So, What Can You Do About It?》。说它特别，是因为它是打印出来的。当我翻看了目录看到里面包含内存屏障和 RCU 的时候，立即明白这就是我几年来苦苦追寻而未得的书！并且，这本书里面花了浓重的笔墨来讲述这两个主题。还有比这更令人高兴的事情吗？

聪明的读者一定知道接下来发生了什么事情。你猜得没错，我鼓动鲁阳一起翻译这本书，当时的目的纯粹是为了学习，并且分享到网络中。在此，我不得不向鲁阳道歉：为了让你坚持下去，我说过一些不留情面的话。这些话不过是云门禅宗棒喝之法而已。如果你早就忘记这些话，那就谢天谢地了！

为了翻译好这本书，我特意去补了一下英语方面的课。最终，这本书能够与读者见面，大概有以下几个原因。

1. 鲁阳和我都有一点 Linux 内核和并行编程基础。
2. 我们的热情和有点自大的自信。
3. 英语不算太难学。
4. 家人的容忍。也许是我常常念叨：翻译好这本书，工资可能涨一大截。
5. 有一些值得感恩的网友，他们督促我们做好翻译工作。

在终于完整地翻译完本书之际，我整理出几条理由，这些理由使得本书有成为经典的潜质。

1．深刻是本书的特点。本书从霍金提出的两个理论物理限制为起点，站在硬件而不是软件的角度，阐述并行编程的难题。让人有一种“知其然并且知其所以然”的感觉。书中不少观点，如内存屏障的传递性，是资深工程师也不容易理解的。但是，看过本书以后，读者会有一种豁然开朗的感觉。

2．在并行编程方面，本书比较全面，并且包含丰富的示例。包括但不仅仅限于硬件基础、缓存、并行设计、锁、RCU、内存屏障、验证及形式验证、实时并行计算。

3．原著作者 Paul 是真正的大师。看过作者简介，并且真正知道 RCU 在 Linux 内核中份量的朋友，不知道你们是否会在心里嘀咕：在 Linux 开源社区，有没有人愿意试着去挖 Paul 的墙角，代替 Paul 把 RCU 维护起来？

4．Paul 在 40 年的职业生涯中，大部分时间都在编写并行编程的代码。当他扛着与奔驰车价值相当的双核计算机往实验室走的时候，我这个有 20 年编程经验的程序员，还没有上小学。

5．国内对并行计算和 Linux 内核的研究逐步深入，读者期待一本深入讲解并行编程的书籍。本书有并行编程的入门知识，更有值得细细回味、反复琢磨的金玉良言。

6．译者多次核对校正，尽量做到符合原著本意，两位译者有多年 Linux 内核经验，尽量做到不错译。

当然，由于译者水平的原因，书中错漏之处在所难免，热忱欢迎读者提出批评建议。

最后，译者诚心感谢原著作者 Paul 给大家奉献了一本好书；也感谢电子工业出版社符隆美编辑的辛勤工作；以及资深互联网软件工程师刘海平，感谢你热心地向出版社推荐这本书；最真心的感谢留给同桌夫人巫绪萍及我们的儿子谢文韬，牺牲了大量陪伴你们的时间，谢谢你们的宽容！

谢宝友

2017 年 4 月 20 日

于深圳

译者序 2

大概在 6 年前，那时我在中兴通讯的操作系统部门工作。当时中兴通讯希望加入 Linux 基金会，由于英语还不算差，操作系统部的钟卫东部长让我负责一些部门与开源社区的合作推进工作，所以除了编程工作以外，我时常密切关注 Linux 开源社区的最新动态。有一天我在 LWN.net 上浏览新闻，一条消息突然跳入了我的视线：Paul McKenney 出了一本关于并行编程的书（原文链接在此，https://lwn.net/Articles/421425）。那时候我还不知道 Paul 是谁，但是书的内容很吸引人，于是我开始把书介绍给周围的同事。过了一阵子，谢宝友找到我，问我有没有兴趣一起把这本书翻译成中文，我们俩一拍即合，开始利用业余时间合力翻译这本大部头。

虽然当时本书的内容尚未完成，有一些章节只列了提纲，但是关于并行程序的设计思想、相关基础知识、RCU 和内存屏障方面的内容已经非常丰富，对于从事内核开发工作的我们来说，简直就是宝库，我们认为即使只翻译这些内容，对其他中国读者来说也很有帮助。

翻译的过程远远比最初想象得困难，原著作者 Paul 是一位有近 40 年从业经验的资深大牛，在书中大量使用了 Linux 内核的术语、并行编程的研究文献，以及对复杂算法和示例代码的解释，在翻译过程中需要字斟句酌，力求准确地传达作者的原意，所以进度很慢。不过当时我还未婚，尚算自由，下班以后的空余时间几乎全部用来做这件事，大概花了 4 个月的时间完成了第一个中文版本。

我们把中文版发布到了网上，也开始和 Paul 联系，请他帮我们宣传，希望能让更多的中国开发者知道这件事。同年在南京举行的 CLK 2011（中国 Linux 内核开发者大会）上，我和宝友见到了 Paul 本人。这里有一个有趣的小插曲，由于会后找 Paul 提问和咨询的人太多，Paul 干脆在会场外席地而坐，逐一侃侃而谈。不摆架子的大牛，是 Paul 给我留下的最深刻印象。

后来因为家庭的原因，我去了美国。生活充满了新的挑战，继续更新内容的事情因此搁置。期间，宝友继续翻译了答案部分的内容，也有出版社联系过我们，想出版本书的中文版，但因为种种原因，后来无疾而终。直到去年夏天，电子工业出版社正式从原著作者处获得版权授权，邀请我和宝友将本书最新版翻译成中文。于是我们再度合作，花了大半年时间，重新审阅翻译稿，将内容更新到最新版本，完整地翻译了所有附录和小问题答案，同时修订了很多错误。相较之前的翻译稿，作者对这一版的内容有大幅改变：

第 6 章新增了对迷宫问题并行化解法的讨论，

完全重写了第 7 章锁和第 8 章数据所有权的内容。

第 9 章新增了关于危险指针和顺序锁，更新了一些例子。

新增了第 10 章，如何将哈希表并行化。

新增了第 11 章，如何验证并行算法的正确性。

新增了第 13 章综合应用。

新增了第 15 章并行实时计算。

从这个列表可以看出，原著作者 Paul 对并行编程技术的思考一直没有停止，还在不停地将这个经典领域的新问题和新进展加入到本书中。比如一些学者对常见数据结构哈希表的最新研究，使得本书不仅成为一部案头必备的工具书，还是一个开拓新知识的出发点。

作者在书中数次强调设计的重要性。工作以来，我曾参与的项目有操作系统内核、浏览器内核，目前开发的是内存数据库，都属于系统软件的范畴。这些项目都大量使用了多线程来充分利用硬件。在这些实践中，我有一点体会，关于分割数据、分割时间、减少跨线程访问、并行快速路径、锁的使用纪律这些设计思想，一旦应用到项目中，代码就变得清晰易懂，很少有 BUG。一旦违反，就带来难以维护的代码，随之而来的是层出不穷的 BUG。

书中还散落着许多宝贵的经验法则，比如 10.6.3 节中如何通过重新排列数据结构顺序来避免缓存行“颠簸”。这些前辈摸索出来的经验能让新一代的开发者少走很多弯路。

作者的行文十分幽默，我在翻译过程中，经常忍俊不禁。希望能在中文版中，尽量将作者的幽默感保留下来。

去年我的儿子小鱼儿降生，作为一个新爸爸，白天工作，晚上在照顾孩子之余还要加班翻译文稿，真是分秒必争。最后借这个机会，将最真心的感谢留给我的夫人卢静，感谢你对家庭的付出，没有你的帮助和理解，就没有本书中文版的面世。

鲁阳

2017 年 6 月 16 日于 Woburn，Massachusetts

目　录