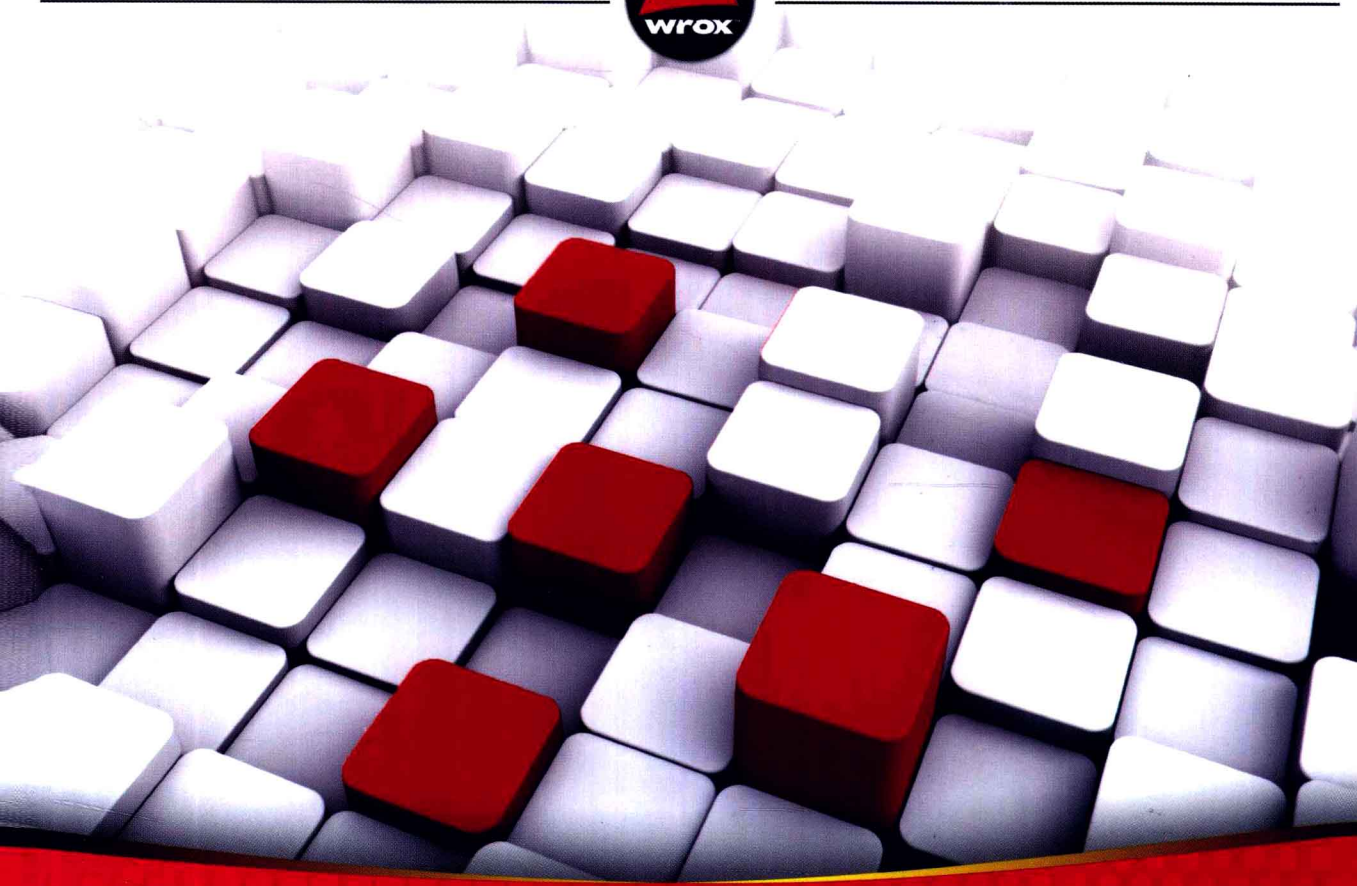

Professional Parallel Programming with C#:
Master Parallel Extensions with .NET 4

C#并行编程高级教程：
精通.NET 4 Parallel Extensions

(美) Gastón C. Hillar 著
郑思遥 房佩慈 译
金旭亮 审稿

清华大学出版社

C#并行编程高级教程：精通.NET 4 Parallel Extensions

(美) Gastón C. Hillar 著
郑思遥 房佩慈 译

清华大学出版社
北 京

Gastón C. Hillar
Professional Parallel Programming with C#: Master Parallel Extensions with .NET 4
EISBN：978-0-470-49599-5

北京市版权局著作权合同登记号　图字：01-2011-0938

图书在版编目(CIP)数据

C#并行编程高级教程：精通.NET 4 Parallel Extensions/(美) 希拉里(Hillar, G. C.) 著；郑思遥，房佩慈 译.
—北京：清华大学出版社，2012.1
书名原文：Professional Parallel Programming with C#: Master Parallel Extensions with .NET 4
ISBN 978-7-302-27356-1
Ⅰ.C…　Ⅱ.①希…②郑…③房…　Ⅲ.C 语言—程序设计—教材　Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2011)第 232978 号

责任编辑：王　军　韩宏志
装帧设计：牛艳敏
责任校对：蔡　娟
责任印制：李红英
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：32　**字　数**：779 千字
版　　次：2012 年 1 月第 1 版　　**印　　次**：2012 年 1 月第1 次印刷
印　　数：1～4000
定　　价：68.00 元

产品编号：040514-01

推荐者序

2011 年 6 月 18 日，国内著名的电子商务网站——京东商城(http://www.360buy.com/)在周年庆典之际举办大规模促销活动，在抢购狂潮席卷而来时，京东的服务器无法及时处理数量庞大的并发订单请求而瘫痪，结果京东 CEO 刘强东先生的新浪微博几乎被消费者潮水般的抱怨所淹没……。这个例子充分说明当今社会对高性能软件系统的需求日益突显，而本书所介绍的并行计算技术就可以应用于高性能的软件系统开发。

再说另外一件事。2011 年 3 月，苹果公司发布 iPad2，配备了 A5 1.0GHz 双核处理器，这标志着平板电脑追随台式机和笔记本电脑一同进入“多核时代”。同样在 2011 年，拥有双核处理器的 Android 智能手机取得了骄人的销售业绩，表现相当抢眼……

处理器的“多核化”趋势已经势不可挡，这意味着硬件的计算能力在持续增强；很明显，如果软件不能充分应用这些新增的计算能力，计算机硬件技术进步所带来的好处就无法显现出来。

所有这一切都告诉我们：并行计算的时代已经来临。

并行计算并非什么新鲜事物，历史上人们在为拥有多个处理器的计算机系统开发软件时，就已经深入研究了并行计算中的许多问题，并提出了相应的解决方案。

概括起来，可将并行计算技术领域大致分为以下三个比较活跃的子领域：

(1) 并行计算的理论研究，比如针对某类问题设计新的更高效的并行算法。

(2) 构建并行计算基础架构，比如设计新的易于开发并行程序的程序设计语言、开发各种并行开发工具、构造并行组件库等等。

(3) 基于并行计算基础架构构建真实软件系统。

本书介绍的.NET 4“并行扩展(Parallel Extensions)”，就属于上述第 2 点所涉及的技术领域。

微软于 2010 年发布的.NET 4 包容众多的技术亮点，本书作者为何单单选取.NET 4 并行扩展来介绍？

笔者本人也曾仔细研究过.NET 4 并行扩展这一技术领域，得到的结论与作者相同——.NET 4 并行扩展有其独到之处，并且具有广泛的应用前景。

得到上述结论的依据是什么？这需要介绍一下并行计算软件系统开发技术的发展现状。

当前的并行计算软件系统，多使用操作系统所直接提供的线程来实现。开发人员编写代码“直接操纵”线程完成并行计算软件系统的开发工作。更详细一点说，在开发并行计算软件系统时，开发人员要依据具体应用场景设计或选择一种并行算法，将整个数据处理工作进行切分，然后创建并启动合适数目的线程，并为每个线程分配特定的工作任务。由于一个算法通常会同时存在串行/并行的部分，因此要保证各工作线程之间满足特定的执行

顺序需求。除此之外，开发人员还需要确保各线程都能安全地存取共享资源，并妥善处理系统运行期间可能出现的各种异常，……，到了最后，在所有功能开发完毕并通过测试之后，还需要依据系统运行的真实软硬件环境进行性能调优。

与单线程程序相比，多线程程序的开发需要考虑的事项更多，另外，由于同时存在多个同时运行的线程，多线程程序的调试也变得复杂起来。有那么一句虽然不太精确却十分形象的评论："多线程程序的开发难度是完成同样功能的单线程程序的 10 倍。"

必须设法提升并行程序的开发效率。为此，Java 平台和.NET 平台都针对多线程开发提供了功能完备的类库，这使得多线程程序的开发难度降低了不少，但仍然相当麻烦。

在大量的软件开发实践中，人们发现：在更高的抽象层次上开发是提升开发效率的有效途径。.NET 4 的并行扩展(Parallel Extensions)走的正是这条路。

.NET 4 并行扩展的核心是"任务并行库(TPL：Task Parallel Library)"，它将并行程序的"构造单元"从"线程(Thread)"提升到"任务(Task)"，将任务分派给线程的工作由 TPL 负责完成，有效降低了.NET 平台上并行计算程序的开发难度，提升了开发效率。

本书主要介绍基于.NET 4 中的并行扩展开发高性能软件系统所需的知识与技能，从并行计算理论到实际代码再到辅助开发工具，娓娓道来，内容紧贴实际，详尽实用。

以下是本人的阅读建议。

不熟悉 TPL 的读者需要通读本书的第 2 章~第 6 章，这部分内容系统介绍了 TPL 的基本使用方法，为实际开发所必需。另外，在阅读之前，不妨先看看附录 A 以"统观全局"。

以下是本书各章节中的一些值得重点关注的亮点：

第 1 章中有关并行计算的经验之谈很有价值，需要反复品味。

第 7 章对 Visual Studio 2010 并行程序调试器进行了详尽介绍，各种技巧一览无余，是本人所看过的相关书中介绍得最清楚的。

第 8 章对线程池的介绍非常重要，它是理解 TPL 工作原理的关键。

第 10 章有关并行测试和调优的介绍很精彩。

第 11 章介绍了如何集成 Intel 所提供的非托管函数库提升软件系统性能，很有参考价值。

附录 B 展示了 UML 并发模型，从事"面向对象系统分析与设计(OOAD：Object Orient Analysis & Design)"的系统架构师们可能会对它感兴趣。

附录 C 介绍了微软所提供的一组 TPL 附加示例，建议读者好好地阅读一下这些示例代码，从中可以学到比较规范的并行编程技巧，很有参考价值。

下面向读者介绍一下.NET 4 并行扩展的主要应用领域。

作为.NET 基类库的有机组成部分，.NET 4 并行扩展可以用于传统的桌面和服务器端应用程序。

需要特别指出的是：在微软云计算平台(Azure)中，同样可以使用.NET 4 并行扩展以充分利用"云"所提供的强大计算和数据处理能力，而且本人已经在互联网上看到了先例，使用 TPL 开发能并行执行 SQL 命令和启动多个并行数据处理任务的组件库。

.NET 4 并行扩展的应用领域同样延伸到了 Windows Phone 和 XBox，比如在 Codeplex 网站上就有一个 PortableTPL 项目(http://portabletpl.codeplex.com/)，使用它就可以在 Windows Phone 和 XBox 的游戏及应用开发中使用 TPL。

阅读完本书之后，我相信读者也会同意我们的观点，.NET 4 并行扩展有着广泛的应用前景，是一项值得认真学习的技术，而本书能为读者掌握这一技术提供切实的帮助。

最后感谢两位译者的辛勤工作，将这本有很大实用价值的书奉献给中国的读者。在此也希望所有读者都能读有所获。

金旭亮

2011 年 11 月作于北京理工大学

作者简介

Gastón C. Hillar 从 8 岁起就开始使用计算机了。在 20 世纪 80 年代初，他开始在传奇的 Texas TI-99/4A 和 Commodore 64 家用计算机上编写程序。他作为一名优秀毕业生在 UADE 大学获得了学士学位，然后又在 UCEMA 大学凭借出色的毕业论文获得了工商管理硕士学位。

自 1997 年以来，Gastón 在并行编程、多处理器和多核处理器领域进行了深入研究。在设计和开发各种类型复杂的利用多核处理能力的并行解决方案方面，他有着 14 年的丰富经验，后来，他开始通过 C#和.NET Framework 编写并行解决方案。从第一个 Community Technology Preview 开始，他就一直在使用 Parallel Extensions。他的大部分时间都花在了与.NET 平台相关的咨询、培训和开发工作上，同时，他还非常关注如何在现代硬件上创建高效软件。此外，他还经常在世界各地的软件开发会议上做演讲。在 2009 年，他获得了 Intel® Black Belt Software Developer 奖。

Gastón 用英语撰写了 4 本著作，为其他两本书编写了一些章节，还用西班牙语撰写了 40 多本其他书籍。他为 Dr Dobb's 网站(www.drdobbs.com)和 Dr. Dobb's Go Parallel 编程门户(www.ddj.com/go-parallel/)贡献了大量文章，他还是 Intel Software Network(http://software. intel. com)的客座博客作者。

他曾在阿根廷 Buenos Aires 市做过开发人员、架构师和项目经理。目前，他是一名独立 IT 顾问，为美国、德国、西班牙和拉美洲的很多公司提供服务，同时，他还是一名自由作家。

他与妻子 Vanesa 及儿子 Kevin 生活在一起。当不在计算机前工作的时候，他喜欢和父亲、儿子及外甥 Nico 在一起玩无线虚拟现实设备和电子玩具。

您可以通过电子邮箱 gastonhillar@hotmail.com 找到他，也可以在 Twitter 上关注他(http: //twitter.com/gastonhillar)。Gastón 的博客地址为 http://csharpmulticore.blogspot.com。

技术编辑简介

Doug Parsons 是一名软件架构师，同时也是 NJI New Media Ohio Operations 的总监。他的专业特长在于 Web 开发。最值得一提的是，他参与了 2008 年 John McCain 总统竞选网站的工作。最近，他在 Mitt Romney 的新书宣传官方网站工作过。在工作之余，他喜欢与可爱的未婚妻 Marisa 以及他们的小狗们一起度过美好时光。

致　谢

并行编程是最难编写的话题之一。通常很难将需要讨论的主题分离开来，而不去引用其他很多密切相关的主题。然而，在编写这本非常具有挑战性的高质量图书的时候，我在所有的必要环节都得到了很多帮助。

我要特别感谢 Paul Reese、Edward Connor、Ginny Munroe 和 Rosemarie Graham——他们非常有耐心，允许我对章节进行所有必要的修改，从而包含最准确最恰当的信息。编写这本书需要做很多工作，然而他们很好地理解到编写一本关于并行编程的高级图书与编写其他编程相关的书籍稍有不同。他们通过不懈的努力使这本书的完成成为可能。此外，我必须感谢 Doug Parsons 和 Kathryn Duggan。您所读到的每句话都包含他们所做的改进。他们极富价值的反馈让我能够将草稿转变为一本正式的书籍。

我要感谢 Microsoft Parallel Computing Platform 团队的 Principal Program Manager Stephen Toub。他对所有章节都提出了有价值的反馈意见。在 Stephen 富有见地的评注下，我得以对书中的示例和内容进行了改进。没有 Stephen 的帮助，这本书会很难完成。他的团队的博客地址为 http:// blogs.msdn.com/b/pfxteam/。这个博客提供了 Parallel Extensions 改进和使用的最新信息。

我还必须要感谢 Daniel Moth，他是 Microsoft Technical Computing 小组的一员。Daniel 帮我改进了有关 Visual Studio 2010 中的调试功能的章节。正是由于他的反馈使我能够在本书中包含这样精彩的一章。

我要特别感谢 Aaron Tersteeg 和 Kathy Farrel，他们是 Intel Software Network 的并行编程社区的经理。我很高兴有机会通过这个伟大的社区扩充了我在并行计算方面的知识。如果没有收听和观看 Parallel Programming Talk 节目(www.intel.com/software/parallelprogrammingtalk)，我就无法写出这本书，这个节目让我更好地把握并行计算的最新趋势。

本书中有一些内容是我参加 Intel Black Belt Annual Meetups 时经过深入细致讨论的结果——我要感谢 James Reinders、Dr. Clay Breshears、Jim Dempsey 和 Doug Holland 能够分享他们的智慧。Doug 还和我分享了对.NET 的热情，通过他的博客我也学到了很多与 Parallel Extensions 相关的第一手经验，他的博客为 An Architect's Perspective(http://blogs.msdn.com/b/dohollan/)。

我必须感谢 Dr. Dobb's 网站(www.drdobbs.com)的编辑 Jon Erickson。Jon 使我有机会能够为 Dr. Dobb's 和 Dr. Dobb's Go Parallel(www.ddj.com/go-parallel/)贡献文章，从而与其他开发人员和架构师分享我的经验。这本书也结合了很多我所收到的反馈意见。

我要感谢 Hector A. Algarra 不断地帮助我提高写作技能。

序

行并程编难很。呃，再试一下，并行编程很难。

尽管这个例子有些滑稽，但是我输入的第一个句子确实能够反映我们开发人员在编写多线程代码的时候需要面对的一些典型问题。当我在写这篇序言的时候，我的两只正在笔记本电脑上打字的手实际上就是两个完全分开的物理进程。如果您更进一步，将我的每一根手指都考虑为一个独立的实体，那么就相当于有 10 个完全独立的进程。我是一个公认的打字快手，为了在每分钟内能输入 100 个以上的单词，我的大脑要设法协调我所有的手指，让它们能够并发地转移到下一个目标，然而还要确保手指的输出能够根据我脑中想要输入的内容的拼写方式进行串行化。在输入第一个句子时，我故意没有使用上述协调机制，这样我的手指就再也不能正确地同步了。这样做的结果就是我的想法很难以可读的方式表现出来。幸运的是，这种错误很容易调试。

并行编程实在是很难，至少从历史上来说一直是这样。利用一些可用的工具，只有少部分软件开发人员能够熟练掌握成功开发和调试多线程应用程序的技能。可是，随着现代计算机的出现，那些需要开发能够响应的用户界面、构建可扩展服务以及为了获得性能要利用多处理内核的开发人员们被迫要处理并发问题，被迫要在线程、互斥量和信号量的层次上开发软件。开发这些软件的困难在于：申请超额(oversubscription)、竞争条件(race condition)、死锁(deadlocks)、活锁(live lock)、二步舞(two-step dance)、优先级翻转(priority inversion)、锁封护(lock convoy)、伪共享(false sharing)等。

因为有这些复杂性的存在，再加上最近业界向多核和众核(manycore)转移的趋势，使得并行编程又受到了很多公司的重视，所以很多公司建立了开发平台，而 Microsoft 是这些公司中的领军者。好几年前，Microsoft 的 Parallel Computing Platform 团队就以清晰的愿景和目标出现在了人们的面前：为了让并行软件的构建更加简单。应该让开发人员能够非常简单地表达存在于应用程序中的并行性，并且要让底层的框架、运行时和操作系统来为开发人员实现并行化，能够将开发人员表达出来的并行化映射到底层硬件，使其能够正确且高效地执行。Parallel Computing Platform 团队的第一波并行支持组件是在 2010 年 4 月份作为 Visual Studio 2010 的一部分发布的。不论您使用的是原生代码还是托管代码，这一次发布的内容都提供了简化并行应用程序开发的基础组件。对于使用托管代码的开发人员来说，这一次发布的内容包括：Task Parallel Library、Parallel LINQ、新的 Parallel Stacks 调试窗口和 Parallel Tasks 调试窗口、能够让您深入查看多线程应用程序执行的 Concurrency Visualizer，当然还有更多的精彩内容。

尽管有了这些工具的支持，并行编程仍然需要深入的知识。在这个以 140 个字符短语作为主要交流手段的年代里(特指微博)，我个人认为一本高质量的书籍仍然是传播那些深奥知识的最佳载体。幸运的是，您现在就在阅读这样一本书。Gastón Hillar 向大家呈现的是一本内容全面的书籍，涵盖了通过 Visual Studio 2010 和.NET Framework 4 开发并行应用程序所涉及的方方面面。从基于任务的编程，到数据并行化、共享状态的管理以及并行程序的调试；从 Task Parallel Library 到 Parallel LINQ、ThreadPool 以及新的协调数据结构和同步原语。Gastón 以深入浅出的方式为大家讲解了.NET Framework 4 和 Visual Studio 2010 中对并行编程的广泛支持。

本书包含的内容可以帮助您获得开发并行应用程序所需的坚实基础知识。恭喜您已经已经跨入多核新世界的第一步了。

——Stephen Toub
微软并行编程平台首席项目经理
2010 年 9 月

前　言

在 2007 年，Microsoft 发布了适用于.NET Framework 的 Parallel Extensions 的第一个 Community Technology Preview(CTP)。老.NET Framework 中的多线程编程模型太复杂，属于重量级模型，已经不适合于正在流行的多核和众核 CPU。从 1997 年开始，我就一直在研究并行编程、多处理器和多核处理器，所以我情不自禁地安装了第一个版本的 CTP 进行试用。很明显，这个 CTP 充分展现了未来 C#版本中激动人心的并行编程方式。

Visual Studio 2010 随着.NET Framework 第 4 版一起发布，这是第一个包含了 Parallel Extensions(并行扩展)的发行版。通过 C# 4 和.NET Framework 4，您可以通过一个全新的基于任务的编程模型来描述并行。编写充分利用多核处理器能力的代码变得更加容易。现在，您可以编写能够随着可用内核增长而扩展的代码，而不用费力地去处理复杂的托管线程。您可以编写运行任务的代码，而 CLR(Common Language Runtime，公共语言运行时)会自动注入必要的线程。现在还能很容易地运行充分利用多核处理能力的数据并行算法。

在撰写本书时，多核微处理器已经无处不在了。服务器、桌面计算机、笔记本电脑、上网本、移动因特网设备(MID)、平板电脑甚至连智能手机都在使用多核微处理器了。每个微处理器的平均内核数在未来会不断增长。您肯定不想错过将这些多核处理能力转换为应用程序性能的机会吧？

并行编程必然会成为您在现代硬件上高效开发 C#应用程序的技能的一部分。在 Visual Studio 2010 正式发布之前，我花了 3 年多时间使用各种不同版本的 Parallel Extensions。我很享受用 C#语言开发并行应用程序的乐趣，并且我尽可能地在这本书中讲解了最常见的情形。

Visual Studio 2010 带有一个专为并行开发人员设计的 IDE，而 C#也非常适合这个新的基于任务的编程模型。

本书读者对象

本书旨在帮助有经验的 C#开发人员能够利用.NET Framework 4 中引入的 Parallel Extensions，将现代微处理器中的多核处理能力转换为应用程序的性能。无论您是刚刚开始从老的多线程模型开始转换，还是已经有过一些编写并发和并行代码的经验，而且想要获得更深入的理解，这本书都能够为您提供您所需要的最常用的并行编程技能和概念。

尽管这本书提供了大量并行编程概念，但 C#和.NET Framework 4 提供的并行编程能力太多太复杂，因此不可能在一本书中涵盖所有内容。这本书的目的是为那些需要充分利用

多核和众核体系结构处理能力的C#开发人员提供与重要技术相关的知识。本书为有经验的C#开发人员提供了充裕的内容，从而让他们能够在很多高层次的并行领域中游刃有余。本书涵盖了新的基于任务的编程模型。对分布式并行化和底层并发编程技术感兴趣的开发人员也可以选择本书作为这些领域专著的补充，从而完善自己的知识。

本书提供的一些背景信息对于避免常见的并行编程陷阱非常重要。因此，在阅读本书的时候最好不要跳过章节。此外，在没有读完某一章的时候，不要将这一章中所展示的代码当作是最佳实践。随着每一章对Parallel Extensions介绍的深入，会通过更简单更高效的编程技术对这些示例进行改进。

本书假设您是一位富有经验的C#和.NET Framework 4开发人员，并且正在关注并行编程相关的主题。如果您缺少以下方面的经验：高级面向对象编程技术、列表、数组、闭包、委托、lambda 表达式、LINQ、类型转换以及基本的调试技术，那么您可能需要一些额外的学习才能完全理解本书中列举的示例。

本书涵盖的内容

本书涵盖了以下关键技术和概念：

- 现代多核和众核共享内存体系结构
- 通过Task Parallel Library(TPL)、C#和.NET Framework进行高层的基于任务的程序设计
- Parallel Language Integrated Query(PLINQ)
- 用于基于任务的编程的大部分常见的协调数据结构和同步原语
- Visual Studio 2010中与并行编程相关的调试和性能剖析能力
- 在真实世界的应用程序中能够让您主宰多核编程的额外的库、工具和其他库

本书没有包含老的多线程编程模型和分布式并行的内容。

本书结构

某些主题的掌握非常关键。如果您之前没有使用过.NET Framework 4引入的新的基于任务的编程模型，那么您应该逐章阅读本书。每一章在编写的时候都假定您已经阅读了前一章。然而，如果您之前使用过TPL和PLINQ，那么您应该可以阅读并理解与并行调试和调优相关的章节。

本书分为以下11章和3个附录：

第1章“基于任务的程序设计”——探讨新的基于任务的编程模型，通过这个新的模型可以在.NET Framework 4应用程序中引入并行性。并行对于发挥现代多核和众核体系结构的威力来说非常关键。这一章讲解了新的轻量级并发模型以及和并发与并行相关的重要概念。这一章的内容非常重要，因为这一章包含了后面10章和附录所需要的必备背景知识。

第2章“命令式数据并行”——开始学习C# 4和.NET Framework 4中引入的新的编程模型，并且将这些模型应用于纯数据并行问题。这一章讲解了一些新的类、结构体和枚举，

通过这些结构可以处理数据并行的情形。运行这些示例，可以更好地理解性能提升。

第 3 章“命令式任务并行”——开始使用新的 Task 实例，以简单的代码解决命令式任务并行的问题和复杂的算法。这一章讲解了一些新的类、结构体和枚举，通过这些结构可以处理任务并行的情形。通过新的基于任务的模型所提供的基本功能和复杂功能来并行地实现现有算法。使用任务而不是线程来创建并行代码。

第 4 章“并发集合”——基于任务的程序设计、命令式数据并行和任务并行都要求使用能够支持并发更新的数组、列表和集合。通过使用新的并发集合，可以简化代码，并且获得最佳性能。这一章讲解了一些新的类和新的接口，通过这些类和接口可以在多个任务中使用共享的并发集合。这一章讲解了如何创建不同的顺序模式和结构的并行代码在列表中添加、删除以及更新不同类型的值。

第 5 章“协调数据结构”——同步由不同的并发任务所执行的工作。这一章讲解了一些经典的同步原语以及.NET Framework 4 引入的轻量级协调数据结构。学习不同的替换方案是非常重要的，这样就可以为每种需要在多个任务之间进行通信和同步的并发情形选择最合适的方案。这是本书中最复杂的一章，也是最重要的章节之一。在编写复杂的并行算法之前，一定要首先阅读这一章。

第 6 章“PLINQ：声明式数据并行”——通过 Parallel Language Integrated Query(PLINQ)及其聚合函数实现声明式数据并行。通过 PLINQ，可以简化混合运行任务和数据分解的代码。您还可以执行经典的并行 Map Reduce 算法。这一章结合了之前章节所介绍的很多主题，并且讲解了如何将一个 LINQ 查询转换为 PLINQ 查询。此外，这一章还讲解了如何根据各种不同的情形对 PLINQ 的并行执行进行调优。

第 7 章“Visual Studio 2010 的任务调试能力”——充分利用 Visual Studio 2010 中引入的任务调试功能。这一章讲解了新窗口如何显示一些重要的信息，这些信息包括任务、任务和源代码之间的关系以及分配给任务的支持任务的执行的线程。在利用.NET Framework 4 编写并行代码的时候，通过这些新的窗口可以检测并解决潜在的故障。

第 8 章“线程池”——理解使用任务以及直接请求工作项在线程池的线程中运行之间的区别。如果使用了一个线程池，那么就可以充分利用新的增强功能，将代码转移到基于任务的编程模型。这一章介绍了.NET Framework 4 中 CLR 线程池引擎的变化，并提供了一个自定义任务调度器的示例。

第 9 章“异步编程模型”——将现有异步编程模型和任务结合起来。这一章提供了真实的案例，讲解了如何充分利用任务和延续的简洁性来执行与现有异步编程模型有关的异步任务。此外，这一章还讲解了一个与并发编程有关的最复杂的问题：从不同的任务和线程中对用户界面(UI)进行更新。这一章介绍了在 Windows Form 应用程序以及 WPF 应用程序中更新 UI 的模式。

第 10 章“并行测试和调优”——介绍了 Visual Studio 2010 Premium 版和 Ultimate 版中引入的并发剖析功能。学习与.NET Framework 4 并行代码有关的常见且容易判断的问题是非常重要的。这一章讲解了一些不同的技术，通过这些技术可以创建并运行并行测试和基准评分。这一章还讲解了如何根据每一个性能剖析会话的结果对现有的应用程序进行重构。

第 11 章“向量化、SIMD 指令以及其他并行库”——充分利用现代硬件所提供的其他与并行化相关的能力。.NET Framework 4 不支持直接使用 SIMD 和向量化。但是，大部分

现代微处理器都提供了这些强大的附加指令。因此，您可以使用经过优化的函数库，这些函数库充分利用了这些指令带来的性能提升。这一章讲解了如何将 Intel Math Kernel Library 整合进 C#编写的基于任务的代码中。此外，这一章还讲解了如何使用 Intel Integrated Performance Primitives 对临界区进行优化。

附录 A“.NET 4 中与并行相关的类图”——这个附录包含了一些类、接口、结构体、委托、枚举和异常的示意图，这些结构通过新的轻量级并发模型和底层线程模型对并行化提供了支持。该附录中还包含了对本书章节的引用，通过引用可以找到详细解释这些图的正文内容。

附录 B“并发 UML 模型”——这个附录列举了一些示例，这些示例讲解了如何使用 UML 模型来展示并发和并行的设计和代码。通过添加一些简单的标准化的视觉元素，您就可以对经典的模型进行扩充。

附录 C“Parallel Extensions Extras”——Parallel Extensions Extras 是一个补充性质的项目，并不是.NET Framework 4 类库的一部分。但是这个项目是由 Microsoft 开发的，并且是.NET Framework 4 并行编程范例的一部分。这个附录包含了 Parallel Extensions Extras 中的类和结构体的图示以及简单介绍。

阅读本书的先决条件

为了能够最大限度地发挥这本书的作用，您需要 Visual Studio 2010 Premium 版或 Ultimate 版，这两个版本的 Visual Studio 包含了.NET Framework 4 以及并发性能剖析的功能。您也可以使用 Visual Studio 2010 Standard 版，但是这个版本并没有包含并发性能剖析的功能。您不应该使用 Visual C# 2010 Express 版，因为这个版本并没有提供对基于任务的程序设计进行调试所需要的调试窗口。

此外，您的开发计算机至少需要具有 2 个物理内核，这样您才能理解本书中所展示的示例。然而，如果您还需要测试可扩展性，那么最好要有 3 个以上的物理内核。

Windows 7 和 Windows 2008 R2 在调度器上做了显著的增强，提升了多核和众核系统的可扩展性。本书描述的应用程序应该运行在这些版本的 Windows 系统上。如果您使用的是更早的 Windows 版本，那么测试结果就可能会有所不同。

目　录