Microsoft

设计模式
——.NET并行编程

（美）Colin Campbell　Ralph Johnson
Ade Miller　Stephen Toub　著

曹泽文　邹雪梅　李 岸　译

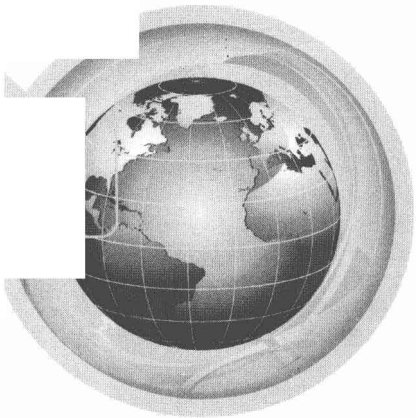
清华大学出版社

Microsoft

设计模式
——.NET并行编程

（美）Colin Campbell　Ralph Johnson
Ade Miller　Stephen Toub　著

曹泽文　邹雪梅　李 岸　译



清华大学出版社
北 京

内 容 简 介

本书结合大量的项目实践，介绍了与并行编程相关的概念、方法和应用。本书共 7 章：第 1 章主要介绍并行编程的基本概念与并行计算的基础理论，第 2 章主要介绍并行循环的知识，第 3 章介绍并行任务处理，第 4 章阐述并行合并计算的机理，第 5 章介绍 future 模式，第 6 章在前文的基础上深入探讨动态并行任务机制，第 7 章介绍并行编程的流水线机制。

本书适用于在.NET Framework 上编写托管代码的程序员，包括在 Visual C#、Visual Basic 以及 Visual F#上编写代码的程序员。本书不假定读者具有并行编程技术的预备知识。不过，读者需要熟悉 C#的特征，如委托、lambda 表达式、泛型以及语言集成查询(LINQ)表达式等。读者还至少应该对进程和线程的概念有基本的了解。

北京市版权局著作权合同登记号　图字：01-2011-0438

图书在版编目(CIP)数据

设计模式——.NET 并行编程/(美)坎贝尔(Campbell, C.)等著；曹泽文，邹雪梅，李岸译. --北京：清华大学出版社，2012

书名原文：Parallel Programming with Microsoft .NET: Design Patterns for Decomposition and Coordination on Multicore Architectures

ISBN 978-7-302-27997-6

Ⅰ. ①设… Ⅱ. ①坎… ②曹… ③邹… ④李… Ⅲ. ①计算机网络—并行编译程序—程序设计 Ⅳ. ①TP393

中国版本图书馆 CIP 数据核字(2012)第 018680 号

责任编辑：文开琪　汤涌涛
封面设计：杨玉兰
责任校对：周剑云
责任印制：王静怡
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**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
课 件 下 载：http://www.tup.com.cn,010-62791865
印 装 者：北京嘉实印刷有限公司
经　　销：全国新华书店
开　　本：185mm×230mm　　**印　　张**：14.75　　**字　　数**：327 千字
版　　次：2012 年 4 月第 1 版　　**印　　次**：2012 年 4 月第 1 次印刷
印　　数：1～4000
定　　价：39.00 元

产品编号：040287-01

译 者 序

随着计算机技术和计算方法的飞速发展，世界各国的高性能并行计算机的研制已取得长足进步。目前计算速度高达每秒百亿次、千亿次乃至数万亿次的高端并行计算机也已相继研制成功。正是由于并行计算的发展，许多以前无法求解和研究的问题，现在要解决它们已变得可能甚至轻而易举。因此，当我初读这本书时，就迫切希望能与别人分享建立在并行计算基础之上的并行编程技术，它也正是并行计算的精髓所在。

什么是“并行编程”呢？初次接触这一概念时我们可能会对这一描述感到惶恐甚至望而却步，但是仔细探索本书后，你会发现，它远没有我们想象的困难。所谓并行编程，简单地讲就是在并行计算机或分布式系统计算机等高性能系统上进行超级计算，以便合理调度资源。实际上，读者现在拥有的 PC 很可能就是一台双核甚至是四核的计算机，而其高效的计算处理能力正是源于本书将要阐述的并行编程技术。

本书内容丰富，几乎涵盖了并行编程的各个方面。一方面，本书既有对并行计算理论的基础原理及架构的阐述，也有对动态任务并行机制以及流水线技术的深入探讨，更重要的是本书每一章都有丰富的实例及示例代码，这有助于读者深入了解其原理与应用。另一方面，本书并非只是一本纯理论的书籍，因此真正善于学习的读者，应该适当地完成一些“课后练习”。对本书示例进行自我学习和探索，我相信您能够从本书获得宝贵经验。

翻译这本书使我受益良多，原作者在并行计算领域的丰富经验使得本书的探讨更加全面和细致，书中展示的许多实例都来源于真实项目的总结，实用性很强。作为译者，我十分高兴能有机会将作者的经验分享给大

家，希望它能帮助到更多的公司或者团队打造出更加高效而强大 的产品，同时也希望为计算机工作者提供一份可贵的参考。

参与本书翻译工作的有曹泽文、李岸、邹雪梅、邓振国、夏韵、徐连君、刘青、邓朝阳和邹剑波。在此感谢团队的每一个成员，没有你们的辛勤工作，这本书就不可能那么快地展现在大家的面前。另外，非常感谢陆昌辉老师的悉心指导。尽管团队的成员都尽心尽力，但由于经验所限，翻译工作尚有很多不足之处，恳请大家提出宝贵意见。

译 者

序

在并行计算出现的四十多年里，各国的并行计算专家将其应用到各领域中，如高能物理、工程应用及计算流体动力学等。自早期的应用以来，我们在并行计算方面已经取得了巨大的进步。

并行计算领域的变化是由硬件快速发展驱动的，持续不断地提高处理器的时钟速度的时代已经结束了。取而代之的是摩尔定律所预言的，通过增加芯片密度来生产多核处理器或者具有多处理器内核的单芯片。目前，四核处理器非常普遍，而且依照这种趋势，在不久的将来，十核处理器也将面世。

在过去五年中，微软利用这项技术优势转化并且创造各种并行应用。其中包括为消息传递接口程序(MPI)而实现的 Windows 高性能集群(HPC)技术，为并行数据处理提供大聚簇数据处理(Map-Reduce)方式的 Dryad，按需提供计算内核的 Windows Azure 平台，为机器码提供的并行模式库(PPL)以及在.NET Framework 4 上加入的并行扩展。

多核计算影响着各层次和各方面的应用，从复杂的科学和设计问题到简单的消费类应用程序和新的人机接口。过去，我们常常开玩笑说“并行计算是未来的事，并且将永远都是”，但这种悲观主义现在被证明是错误的。并行计算从一项生僻的技术最终转变为应用程序开发者和整个 IT 产业的“明星”。

不过，这隐含着一个陷阱。为了加快应用程序的开发，程序员现在不得不将计算工作分成几块，以便充分利用多核处理器的性能，而这仍然需要专业技能。并行程序设计给绝大多数开发人员带来一个巨大的挑战，他们中的许多人还是第一次面对并行开发。因此，目前迫切需要通过实用的

方式获得这方面的知识，以便他们能将并行技术融入应用之中。

有两种可能的方案在从事计算机科学的同僚中非常受欢迎：设计一个新的并行编程语言，或者开发一个“健壮的”并行编译器。从学术上讲，这两种方法都相当有意思，但两者都不能成功地帮助非专业人员简化和普及并行工作。相比之下，另一种更实用的方法是为程序员提供一个库，隐藏并行程序设计的大部分复杂操作，然后再教程序员如何使用它。

为此，Microsoft .NET Framework 并行扩展提供了一个比先前 API 更高级的编程模式。例如，程序员可以从任务的角度来思考，而不是从线程的角度，因而能够避开管理线程所带来的复杂性。Microsoft .NET 的并行程序设计通过让程序员置身于设计模式的背景，来告诉程序员怎样使用这些库。这样一来，应用程序开发者就能快速学习编写并行程序，从而即时获得性能优势。

本书强调了并行设计模式和最新的编程模型，我相信它将代表并行程序设计走向主流的重要的第一步。

Tony Hey

微软研究院，公司副总裁

前　言

本书描述了并行程序设计中需要用到的模式，它们使用Microsoft .NET Framework 4中新加入的并行编程功能来进行并行设计，这种功能通常被称为并行扩展。你可以使用本书中的模式提高应用程序在多核计算机上的性能。在代码中使用这些模式能够使它们在现有机器上运行得更快，并且有助于适应将来的硬件环境——据预计，它们将融入越来越多的并行计算架构。

本书适合的读者

本书针对的读者对象是在Windows操作系统上利用.NET Framework编写托管代码的程序员。这包括在Visual C#、Visual Basic以及Visual F#上编写代码的程序员。本书不假定读者事先已了解并行编程技术。不过，读者需要熟悉C#的特征，如委托、lambda表达式、泛型以及语言集成查询(LINQ)表达式等。读者还应该大致了解进程和线程的概念。

请注意：本书中的示例是用C#写的，并且使用了.NET Framework 4中的特性，包括任务并行库(TPL)和并行LINQ(PLINQ)。不过，也可以将本书中呈现的概念应用到其他框架和库以及其他语言中。

完整的代码解决方案已经发布在CodePlex上，请参见http://parallelpatterns.codeplex.com/。每个示例都有一个C#版本。除此之外，还有Visual Basic版本和F#版本。

为何在此时写作本书

Visual Studio 2010 开发系统实现了先进的并行编程功能，这使得并行程序设计的入门比以往任何时候都更加容易。

任务并行库(TPL)是为想要编写并行程序的.NET 程序员而开发的。它简化了在应用程序中应用并行和并发的过程。TPL 动态地调整并行度以最高效地利用所有可用的处理器。此外，TPL 还能协助您在.NET 线程池中划分和调度任务。该库提供了取消支持、状态管理以及其他服务。

并行 LINQ(PLINQ)是 LINQ to Object 的并行实现。PLINQ 实现了所有的 LINQ 标准查询操作符，将它们作为 System.Linq 命名空间的扩展方法，并额外加入了用于并行操作的操作符。PLINQ 是声明式的高层次接口，能够用于查询筛选、投影及聚合等操作。

Visual Studio 2010 包含了调试并行应用程序的工具。并行堆栈(Parallel Stack)窗口展示了应用程序中所有线程的堆栈调用信息。它能够让你自由地驰骋于各个线程与线程上的堆栈帧之间。并行堆栈窗口类似于线程窗口，只不过它显示的是每个任务的信息，而不是每个线程。通过 Visual Studio 分析器中的 Concurrency Visualizer 视图，可以看到应用程序如何与硬件、操作系统以及计算机上的其他进程交互。可以使用并发观测仪来定位性能瓶颈、处理器利用不足、线程争用、跨内核的线程迁移、同步延迟、I/O 冲突区域以及其他信息。

想要了解 Microsoft 可利用的并行技术的概况，请阅读附录 C。

使用代码的系统要求

本书使用的代码可以在 http://parallelpatterns.codeplex.com/找到。以下是对系统的要求：

- Microsoft Windows Vista SP1、Windows 7、Microsoft Windows Server 2008 或者 Windows XP SP3 (32 位或 64 位)操作系统
- Microsoft Visual Studio 2010 (使用 Concurrency Visualizer 需要 Ultimate 或 Premium 版本。可以通过 Concurrency Visualizer 来分析应用程序的性能)，其中包含了运行示例代码所需的.NET Framework 4

如何使用本书

本书从具体的模式出发讲述了并行编程技术。图 0.1 展示了各个不同的模式以及它们之间的关系。其中，数字代表该模式在本书哪一章中介绍。

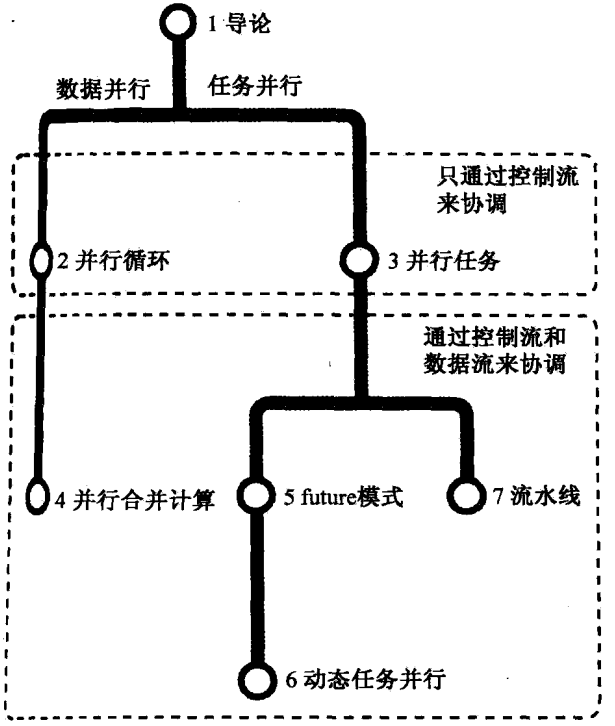


图 0.1 并行编程模式

在导论之后，本书从数据并行和任务并行两个方面讨论了并行编程技术。

并行循环和并行任务均运用程序的控制流作为协调和排列任务的方式。其他的模式则结合控制流和数据流两者来协调。控制流是指算法的步骤。数据流是指输入和输出的可用性。

导论

第 1 章阐述了意图使用并行技术来加快应用程序的运行速度的开发者们所面临的共同问题。本章解释了基本的概念，为后文做好铺垫。在 1.2.4 节中有一个表格可以帮助你选择适合你的应用的正确模式。

仅依赖控制的并行

第 2 章和第 3 章介绍了异步操作的排序仅受控制流限制的情况。

- 第 2 章介绍并行循环。想要在一个索引范围内或者一个集合的每个成员上执行相同的计算，并且成员之间没有依赖时，就使用并行循环。对于有依赖的循环，参见第 4 章。
- 第 3 章介绍并行任务。要执行几个不同的异步操作时，就使用并行任务。本章解释了为什么任务和线程适用于不同的目的。

依赖控制和数据的并行

第 4 章和第 5 章展示了既受控制流限制又受数据流限制的并发操作模式。

- 第 4 章介绍并行合并计算。并行合并计算模式适用于并行循环体中包含数据依赖的情况，例如计算总和或者在集合中查找最大值。
- 第 5 章介绍 future 模式。当操作产生的输出需要作为其他操作的

输入时，就要用到 future 模式。操作的顺序受限于数据依赖的有向图。有些操作是并行执行的，也有些是串行的，取决于输入的可用性。

动态任务并行和流水线

第 6 章和第 7 章讨论了一些更高级的应用场景。

- 第 6 章介绍动态任务并行。在某些情况下，操作是在计算过程中动态地添加到待办任务中的。这种模式可应用于某些领域中，包括图形算法和排序。
- 第 7 章介绍流水线。使用流水线将一个组件的连续输出提供给另一个组件的输入队列。当流水线被充满，或者超过一个组件同时处于活跃状态时，则将结果并行化。

支撑材料

除了对这些模式的描述，本书还提供了几个附录。

- 附录 A 针对一些常用面向对象模式(例如外观模式、装饰模式和数据仓库)在多核体系中的适应性给出了一些提示。
- 附录 B 简要介绍了如何在 Visual Studio 2010 中调试和分析并行应用。
- 附录 C 描述了并行编程方面的各种微软技术和框架。
- 术语表。术语表包含了本书中用到的术语。
- 参考文献。参考文献列出了文中提到的著作。

读者应当首先阅读第 1、2、3 章，大致了解基本的原理。尽管后面的章节是以逻辑顺序安排的，但从第 4 章起，每章都可以独立阅读。

本书中的标注是以特定的方式展示的，提示你应当特别小心的地方。

如此处边栏所示。

不要盲目应用本书介绍的模式。

试图运用一种新的工具或技术来解决面临的所有问题，而不管其适用性，这种想法是很诱人的。正如俗语所说，“当你有一把锤子的时候，任何东西看起来都像是图钉。”这种“什么都是图钉”的心态可能会导致非常不幸的结果。显然，大家都希望图 0.2 中的小兔子能够避免这种结果。

你当然也想在并行编程中避免不幸的结果。在应用中增加并行性需要花费时间，且增加了复杂性。为了取得好的效果，应当只在效益大于投入的部分应用并行化。

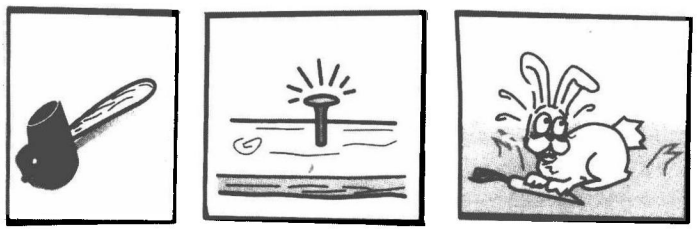


图 0.2 当你有一把锤子的时候，任何东西看起来都像是图钉

本书中没有包含的内容

比起 I/O 密集型的工作负载，本书更加关注的是计算密集型的负载。其目标是通过更好地利用计算机的可用内核，加快计算密集型的应用的运行速度。因此，本书没有花费太多的精力在 I/O 延迟上。不过，本书中也有关于负载均衡的讨论，且既包括计算密集型又包括 I/O 密集型(见第 7 章)。在第 5 章中还有一个关于用户界面的重要例子，该示例说明了带有 I/O 操作的任务的并发。

本书讨论了具有共享内存的单个多核节点中的并行性，而非集群中的并行，集群使用的是高性能计算(HPC)服务器的方法，在连通的节点上运用分布式存储。但是，想要在一个节点上利用并行优越性的集群程序员

也可以在本书中找到有用的例子，因为集群的每个节点可以有多个处理单元。

目标

读完本书，你应当能够完成以下任务。

- 回答每章后面的问题。
- 寻找书中是否有你的应用程序所适合的模式，如果有，要知道是否有望完成一个简单的并行实现。
- 理解你的应用什么时候适合使用其中的一个模式。在这个时候，你需要进行更多的阅读和研究，或者寻求专家的帮助。
- 如果你的并行实现不能工作，能够知道可能的原因，例如任务之间的依赖冲突或者错误的共享数据。
- 利用“扩展阅读”一节找到更多的资料。

目　录