<<Windows并发编程指南>>

图书基本信息

书名：<<Windows并发编程指南>>

13位ISBN编号：9787111288206

10位ISBN编号：7111288203

出版时间：2010-1

出版时间：机械工业出版社

作者：杜飞

页数：604

译者：聂雪军

前言

计算机业界再次走到了一个十字路口。
以多核处理器为代表的硬件并发性的出现，以及不断增长的软件复杂性，都要求业界的人们重新思考现代计算机的架构以及软件开发模式。
在过去几十年中，计算机的性能和容量都发生了指数级的增长，但底层的计算模型却没有根本性的变革。
硬件的发展遵循摩尔定律，时钟频率不断增加，而软件在发展过程中也充分利用了这种不断增长的硬件性能，并且通常要领先于硬件的发展。
硬件与软件的这种共生关系一直延续到最近。
尽管摩尔定律仍然有效，但时钟频率却无法再继续线性地增长。
加州大学伯克利分校的David Patterson教授通过一个简单的等式总结了在硬件上发生这种变化的原因：能耗障碍+内存障碍+ILP障碍=限制串行性能的障碍CPU的能耗随着时钟频率的增加而增加，因此时钟频率存在着一个上限。
目前，CPU的散热能力已经达到了一个物理临界点。
因此，如果没有强大的（以及昂贵的）冷却功能（或者在材料技术上取得突破），那么将不可能在时钟速度上获得重大提升。
这就是等式中的“能耗障碍”。
内存性能的提升远远落后于处理器性能的提升，这使得在访问主存时需要的CPU时钟周期数量将持续增加。
这就是等式中的“内存障碍”。
硬件工程师们已经采取了猜测执行指令（Speculatively Executing Instruction）技术来提升串行软件的性能，这种技术也称为指令级并行（Instruction I,evel Parallelism，ILP）。
ILP对程序性能的提升程度是很难确定的，并且ILP的复杂性还将提高能耗。
因此，ILP带来的提升同样存在着局限性，这就是等式中的“ILP障碍”。
因此，我们目前到达了一个转折点。
软件系统必须通过进一步的演变来更好地支持多核架构，而这种演变是需要时间的。
为了充分利用提升的计算机硬件性能，并继续保持“只需编写一次，就可以在新硬件上运行得更快”这种模式，开发人员必须学习如何构造并发程序。
与并发的广泛使用相伴随的，还有异步性和松耦合、客户端并行以及服务器端云计算等，这些技术使得“软件+服务”的模式成为可能。
Windows和.NET框架平台对并发提供了丰富的支持。
这种支持最初在Windows NT对多核处理器的支持中引入的，并且在过去十年中得到了不断的发展。
Windows在线程调度性能、同步API以及内存层次结构等方面的持续改进，尤其是在Windows Vista中增加的一些新功能，使Windows操作系统能够最大限度地发挥硬件并发的优势。
本书对所有这些领域都进行了介绍。
当你在应用程序中开始大规模地使用多线程方式时，简单明了的架构设计将变得非常重要，因为这种架构可以有效地降低软件的复杂性并提高代码的可维护性。
这就要求开发人员不仅要理解平台上的各项功能，而且还要关注新近出现的各种最优方法。
本书中，Joe不仅详尽地介绍各种并发机制，同时还提出了大量的最优方法。
毫无疑问，多核提升了现有应用程序的性能。
不仅如此，它同样使我们可以从一个完全不同的角度来重新思考计算机为人们提供的各种功能。
计算能力的不断增加，不仅将提升应用程序的质量从而使其变得更加有用，而且还可以实现一些在过去无法做到的事情。
在这种演变过程中，软件将变得更加个性化和人性化。
好好享受本书吧，它为你在Windows平台上迈出编写并发软件的第一步提供了完备的信息。

<<Windows并发编程指南>>

内容概要

本书详细讲述Windows并发编程的相关知识。
内容分为四个部分：第一部分“概念”从高层视角介绍并发的基本概念，为读者理解并发打下基础：第二部分“机制”重点介绍了一些基础的功能、内部工作机制以及各种APl等；第三部分“技术”介绍了一些常见的编程模式、最优方法、算法，以及在编写并发软件时需要使用的各种数据结构；第四部分“系统”介绍了一些在系统架构和流程中经常出现的问题。
本书内容翔实，实例丰富，适合Windows开发人员、Windows测试人员；和Windows技术支持人员等参考。

作者简介

Joe Duffy是微软公司．NET框架团队中负责并行扩展库(Parallel Extensions]的开发管、架构师和奠基人。
除了编写代码和管理开发团队之外，他还致力于一些长期性和前瞻性的工作，例如支持并发安全性的语言和类型系统。
他曾在通用语言运行时(CommonI Language Runtime)团队中

<<Windows并发编程指南>>

书籍目录

章节摘录

插图：总的来说，许多问题都有内在的并发特性。
例如，在构建一个服务器程序时，需要考虑到有许多请求会并发地到来，而服务器程序必须同时处理这些请求。
如果正在编写处理Web请求的模块并且需要访问一些共享状态，那么也会发现需要处理与并发相关的问题。
有时候，我们可以通过并发来将一些问题以更自然的方式表达出来，但在实际情况中却很少会使用并发。
人们似乎正在经历一个需要面对大量异步性的困难时期，而这些异步性是随着各种交互操作的激增而带来的。
于是，人们开始越来越多地使用并发，甚至在一些并不适合的情况中也使用并发。
出现这种现象的原因是由于多处理器架构发生了重大的变化，各种主流计算机都在广泛地使用并行处理器。
多核处理器已经占据了PC市场和移动市场，而高度并行的图形处理器（Graphics ProcessingUnit，GPU）也随处可见，有时候甚至用来执行普通的图形计算。
为了最大限度发挥这些新型处理器的强大功能，我们必须以一种可伸缩的方式来编写程序。
这就意味着在程序中必须包含足够的潜在并发（Latent Concurrency），这样当采用更新的机器时，这些潜在的并发将变成真实的并发，而程序的性能也将自动提升。
事实上，尽管我们大多数人都习惯以串行方式来编写程序，但在代码中经常会包含大量的潜在并发，这些并发是由于我们所选择的编程语言而带来的。
在许多情况下，循环、if分支和内存移动中包含的数据依赖性和控制依赖性将限制这种并发，这些限制都是在一些不符合C编程风格的代码上施加的人为限制。
相对于过去的编程方式来说，并发是一种重大的变化，对于客户端程序来说尤其如此。
并行是指通过并发将一个操作分解为一组粒度更细的子操作，并且这些子操作可以在不同的处理器上独立运行。
这种思想并非最近才出现的。
早在数十年前，在科学计算和超级计算机中已经通过并行在数十个、数百个，甚至数千个处理器上获得可伸缩性。
然而，一些主流的商业软件和Web软件通常是采用串行方式来编写的，因为这些软件的开发商认为CPU的时钟速度每年都会提升40%～50%，而因此带来的程序性能提升足以满足将来的需求。
1.2 程序架构与并发我们首先从程序架构开始来讨论并发。
当我们可以在现有的程序中增加并发性时，如果预先仔细地进行设计，那么将极大降低一些常见缺陷的出现概率。
下面这些术语对于构思并发程序的结构来说是非常有用的，它们在项目的初始设计阶段将带来极大的帮助：·代理（Agent）。
大多数程序都可以分解为一些独立的代理。
在这里，代理是一个非常抽象的术语，它包含一些关键的属性：1）代理的状态自始至终都包含在代理的内部；2）代理与外部环境的交互过程是异步的；3）在同级代理之间的关系通常是松散耦合的。
在真实的系统中，代理的表现形式有多种，如Web请求、Windows基础通信层（Windows Communication Foundation，WCF）的服务请求、COM组件调用，以及线程中的异步行为等。
而且，在一些程序中只存在一个代理，即程序的入口点。

媒体关注与评论

“ 在程序中使用多线程时，简单明了的架构和设计是常重要的……这要求我们不仅要全面地理解平台的各项功能，而且还要随时关注一些新近出现的最优方法。
Joe在本书中完美地阐述了并发的基本理论以及各种最优方法。
”　　——Craig Mundie，微软公司首席研究与战略官

编辑推荐

《Windows并发编程指南》：深入揭示Windows并发编程理论及实现微软首席研究与战略官Craig Mundie作序推荐《Windows并发编程指南》详尽地介绍了如何在开发软件时高效地使用并发和硬件并行，以及如何设计、实现和维护大型的，主要以Windows平台上的C#和C++等语言编写的并发程序。《Windows并发编程指南》为应用程序、系统以及库等的开发人员在多核处理器上编写高效、安全的代码提供必要的工具和技术指导。
这不仅对于那些存在内在并发性的领域(如服务器应用、计算密集的图像处理、金融分析、数值模拟以及AI算法等)来说是很重要的，而且对于一些可以通过并行来获得加速的领域(如数学库、排序算法、报告生成、XML处理以及流式处理算法等)来说同样是重要的。
《Windows并发编程指南》主要包括四部分：第一部分从高层视角介绍并发的一些基本概念；第兰部分重点介绍一些重要的平台功能、内部工作机制以及API细节等；第三部分介绍在编写并发软件时一些常见的编程模式、最优方法、算法和数据结构等；第四部分介绍在编写并发程序时需要注意的一些系统性的架构和流程方面的问题。
要想学习在windows和.NET上进行并发编程的最优方法和通用模式，那么《Windows并发编程指南》将是你唯一需要的一《Windows并发编程指南》。

版权说明