BOOKS FOR PROFESSIONALS BY PROFESSIONALS™

Maximizing .NET Performance

.NET性能优化

- 提供优化.NET Framework 性能的第一手资料
- 提供可靠的建议——所有建议都建立在对.NET Framework 进行全面研究和系统性基准测试的基础之上
- 通过阐述为何、何时以及如何优化.NET Framework 性能，帮助您避免程序性能上的缺陷

(澳) Nick Wienholt 著
田松茂 谢君英 译

清华大学出版社

.NET 性能优化

(澳) Nick Wienholt 著

田松茂 谢君英 译

清华大学出版社

北 京

内 容 简 介

.NET 平台大大提高了软件开发的效率，但要想获得最佳性能的应用程序，还需要程序员对该平台进行优化。本书内容基于作者的切身实践，有助于程序员开发出高性能的应用程序。全书共分为 15 章和一个附录。第 1-2 章介绍了.NET 性能方面的知识；第 3-14 章集中讲述.NET Framework 特定领域的性能问题，包括类型设计、表达式、集合、编程语言、垃圾回收、异常、安全性、线程、I/O 和序列化、远程处理、托管技术、CLR 等内容；第 15 章是一个故障排除指南；附录 A 介绍了一些基准测试工具。

本书适合于熟悉.NET Framework 的中高级程序员阅读。

EISBN: 1-59059-141-0
Maximizing .NET Performance
Nick Wienholt

北京市版权局著作权合同登记号 图字：01-2004-2949

图书在版编目(CIP)数据

.NET 性能优化/(澳)维恩霍尔特(Wienholt, N.)著；田松茂，谢君英译.—北京：清华大学出版社，2005.8
书名原文：Maximizing .NET Performance
ISBN 7-302-11046-8
Ⅰ. N… Ⅱ. ①维…②田…③谢… Ⅲ. 计算机网络—程序设计 Ⅳ. TP393
中国版本图书馆 CIP 数据核字(2005) 第 050105 号

出 版 者：清华大学出版社 **地 址**：北京清华大学学研大厦
http://www.tup.com.cn **邮 编**：100084
社 总 机：010-62770175 **客户服务**：010-62776969
组稿编辑：曹 康
文稿编辑：李 阳
封面设计：康 博
版式设计：康 博
印 装 者：北京昌平环球印刷厂
发 行 者：新华书店总店北京发行所
开 本：185×230 **印张**：16.25 **字数**：281 千字
版 次：2005 年 8 月第 1 版 2005 年 8 月第1 次印刷
书 号：ISBN 7-302-11046-8/TP·7322
印 数：1～4000
定 价：32.00 元

序　言

“托管代码的运行速度太慢了。我们必须在 C++中改善这一点。”

——佚名

在 1999 年，ACM(美国计算机协会)公布了一项调查报告[1]，对一个计算密集型问题的 40 个独立实现进行了比较，这些实现是由不同程序员用 Java 或 C/C++编写的，Java 是当时的托管运行时环境。结论是“程序员之间的个人差异比 Java 和 C/C++之间的平均差异要大得多”，并且“中等质量的程序与上等质量的程序及下等质量的程序之间在性能上经常会存在 30 倍及以上的差距”。

这应该让您明白一些道理：如果您不是顶级 C++程序员，那么托管代码实现还是很有希望与一般的 C++解决方案执行得一样好——尤其是在假设大多数.NET 语言只允许您引入很少一些与内存相关或性能相关的微妙问题时。并且记住，这项调查是在几年前完成的，即时编译(Just-In-Time Compilation，JIT)和内存管理及垃圾回收(garbage collection，GC)技术在这些年中已得到了改进。

但是这并不意味着您就不用担心会用.NET 创建出非常慢且消耗内存的应用程序。因此您应该关注调查报告另一个方面的结论，即“程序员之间的个人差异……大得多”。实际上这意味着，您应该知道如何优化应用程序，使之以预期的性能在托管环境中运行。尽管.NET 完成了许多在 C++中应该由程序员来完成的任务，但是这些任务仍然存在；这些“小木偶”只是完成了主要的任务，现在正躲在幕后的某个小角落里。如果您想要应用程序以较高的性能运行，就仍然需要找到合适的“线”来拉动这些隐藏的“小木偶”，使其发挥作用，并且不许它们对应用程序的性能造成负面影响。

但是只了解公共语言运行库的内部情况还是不够的，因为大量的性能问题实际上出现在应用程序设计阶段，而不只是出现在代码编写期间。您还要了解集合、远

1　Lutz Prechtelt, “Comparing Java vs. C/C++ Efficiency Differences to Interpersonal Differences,” *Communications of the ACM* 42, no. 10 (October 1999): 109–112。

程技术、与非托管代码的交互性、COM 组件等内容。

本书的目的是，让您了解设计问题和底层的 CLR(公共语言运行库)机制，以便创建的程序能够以 ACM 调查报告所说的 30 倍的性能差异运行。Nick 真的是想编写一本能够解决这些问题的书，若在托管环境中粗心地编写代码时就会出现这些问题。本书引导您深入细节，却不会被公共语言运行库的底层复杂性所左右。

阅读本书之后，您惟一需要防范的事情就是不要过度优化代码。我感到欣慰的是，Nick 在本书中一开始就讨论了识别应用程序性能的关键部分，然后才介绍了如何隔离和解决这些实际的性能瓶颈。依我的观点来看，这是最重要的任务之一，当处理大规模的应用程序时，这也是最复杂的任务之一。

现在就往下读吧，去赢得那 30 倍优越的性能。

Ingo Rammer，*Advanced .NET Remoting* 一书的作者

http://www.ingorammer.com

前　言

与非功能性软件生产的其他方面一样，性能通常是构建过程的事后想法，需要在现有代码的基础上进行追踪。对于像.NET 这样的新平台，这种生产实践的负面结果就更加显著，因为开发人员和架构师在托管环境中并不具有相同深度的经验。在互联网时代，开发具有适当性能特征的软件比以往更为重要，而在缩减 IT 预算的大气候下，人们不再允许将硬件运行在很差的执行系统上。

本书的目的是用必要的知识来武装开发人员和架构师，使他们能够在.NET 平台上设计和实现具有优良性能特性的软件解决方案。本书假定读者熟悉.NET Framework，如果不了解.NET 的话，可以通过 Apress 网站(http://www.apress.com)找到许多介绍该平台的优秀图书。本书适合那些了解.NET 并想寻找更高级的资料以构建性能优良软件的开发人员。

本书可以逐章阅读，也可以挑选独立的章节阅读。当挑选独立的章节阅读时，建议最好阅读第 1、2 章，这两章为介绍后面章节的材料作了铺垫，是后续章节的基础。第 3-14 章集中讲述特定于.NET Framework 某一领域的性能问题，这些章节中依赖于其他章节信息的材料，本书已经突出显示，这是特意为那些不准备按顺序阅读的读者准备的。

第 15 章给出了一个故障排除指南，目的是为诊断和解决.NET 应用程序中的性能问题提供一个系统化的方法。本章介绍的工具和技术，可以用于可靠地找出一个程序为什么运行缓慢或者过度消耗内存的原因，并解决问题。

附录 A 介绍了用于执行全书讨论的基准测试的基准测试工具。对于那些有兴趣扩展和添加本书所给出的基准测试的读者，本附录肯定是很有用的。

本书的源代码可通过 Apress 网站(http://www.apress.com)获得，也可以通过本书合作站点 http://www.tupwk.com.cn/loadpage 下载，鼓励读者下载并分析这些源代码。所编写的基准测试工具可以容易地再现本书中所引用的基准测试，利用这些基准测试进行实验是帮助学习的一个很好的方法。好奇心对于理解性能是一个基本的先决条件，而源代码可以为读者进行许多新的性能调查提供一个起点。

作 者 简 介

Nick Wienholt 是一位 Windows 和.NET 顾问。过去 10 年间，Nick 参与了多个 IT 项目，从海岸侵蚀的数字建模到财务和工资系统，最为成功的是乘客信息显示系统(Passenger Information Display System，PIDS)项目。他在该项目中担任 Rail Services Australia 的顾问，成为其职业生涯中的一个亮点。PIDS 项目在悉尼 2000 年奥运会之前开发并安装，是奥运会期间国家火车成功运载无数观众的一个关键项目。

Nick 是 Sydney Deep .NET 用户组的创立者之一，并为 Pinnacle Publishing, *Australian Developer Journal* 和 Microsoft Developer Network 撰写技术文档，是.NET 相关新闻组的积极参与者，同时还是技术协会的常任会员。通过站点 http://www.dotnetperformance.com 可获得 Nick 的 SDNUG 演示文稿和文章。作为对他在.NET 领域所作贡献的奖赏，2002 年他被授予“微软最有价值的专家奖(Microsoft Most Valued Professional Award)”。

读者可以通过邮箱 nick@dotnetperformance.com 与 Nick 联系。

技术审稿人简介

Simon Robinson 是一位知名的作家，擅长于.NET 编程，他还是关于 ASP 和 ASP.NET 相关技术的知名 Web 站点 ASP Today 的资深编辑。

Simon 在 Windows 编程方面具有十分丰富的经验，但核心专长是.NET 编程，他对用 C++、C#、VB 和 IL 编写代码也相当熟练；Simon 技能广泛，涉足的领域从图形和 Windows Forms 到 ASP.NET，到目录和数据访问，到 Windows 服务和本机 Windows API 等。

您可以访问 Simon 的个人网站 http://www.SimonRobinson.com 或 ASP Today 的网站 http://www.asptoday.com 来了解与本书相关的一些内容。

目　录

第 1 章　简　介

软件性能是一个充满矛盾、混乱和谜团的主题。先看下面的一些例子：

- 处理器速度每 18 个月就提高一倍，但是对性能的关注从来没有消失过。
- 两个互相竞争的软件平台的速度在各自供应商制作的基准测试结果中，都显示出超过对手数量级的速度优势。
- 能够避免使用缺乏性能优势的语言和技术，但是很少能精确地测量缓慢的程度。另外，即使性能不是项目需要优先考虑的问题，仍然会出现对速度缓慢的批评。

本书试图揭开.NET Framework 平台上代码的一些性能谜团。本书有两个主要目标，一是详细讨论各种.NET Framework 技术的性能，二是演示在面对本书没有介绍的技术和方法论时如何进行可靠的性能评估。为了达到这两个目标，本书所讨论的每一个测试的源代码都可以从 Apress 的网站(http://www.apress.com)下载得到，每个测试都被清楚地编号，这样便于在示例代码中快速定位。读者可以轻松地在自己的系统上重新运行所有测试示例，并分析构成测试实例的代码。第 2 章包含实施性能评估的详尽讨论，附录 A 则论述了用于实施获得书中测试结果的测试工具。

鼓励读者能批判地分析本书所给出的测试结果。由于服务包、.NET Framework 的新版本和操作系统等原因，本书的结果和读者所使用系统得出的结果很可能是不同的。在某些情况下，测试案例中微小的改动能显著地改变方法的性能，而且测试某项技术的测试案例和该项技术的其他使用方式可能也是毫无关系的。由于微妙的变化能够带来显著的性能变化，因此对细节的敏锐观察是很关键的。

1.1　本书主要内容

本书重点在于讨论.NET Framework 的性能。本书专注于从底层开始的 Framework 性能，而没有论及诸如 Windows Forms、ASP.NET 和 ADO.NET 等高级

技术。对于这些重要技术，将会在未来相应的著作中论述；但是要记住，所有.NET代码都建立在相同的基础上，而这正是本书的重点。系统级开发人员最适合使用本书，而对于那些使用高级技术的应用程序开发人员，我也努力使本书的内容更便于他们接受，并与这个级别的开发相关联。通过深入地理解.NET Framework 的性能，有利于应用程序开发人员更好地识别和避免自己编写的代码和高级应用类库中出现性能方面的错误。

本书讨论了以下几个关键领域：

- 使用黑盒和白盒评估技术来分析软件系统的性能
- 设计出包含最佳性能特性，并能高效地与 Framework 设计模式进行交互的类型(类和结构)
- 使用远程技术构建高性能的分布式系统
- 使用 COM 交互操作、P/Invoke 和 C++等技术与非托管代码高效地交互
- 理解性能与语言选择之间的相互影响
- 利用.NET 垃圾回收器达到高性能的对象分配和回收
- 选择正确的集合类达到最优性能
- 定位和修补具体性能问题

本书没有涉及.NET 和 J2EE 等其他竞争平台的比较。尽管某些读者对此类信息感兴趣，但跨平台技术的性能对比倾向于导致争论以及冲突的结果，最近 Java 提供商和 Microsoft 之间关于 Pet Shop 的性能论战就表明了这一点。其实大多数跨技术比较都是出于商业目的，对开发人员在任何平台上编写更快、更高效的代码毫无帮助。本书侧重于向开发人员提供编写高性能.NET Framework 平台代码所需的信息和工具，跨平台的讨论只会偏离这个目标。

如果读者对已独立验证的基准测试结果对比感兴趣，则可参考 Transaction Processing Performance Council 的结果,网址为 http://www.tpc.org。

1.2 解决具体的性能问题

如果购买本书的目的是为了优化那些花费大量时间才能完成任务的代码，那么第 15 章是最佳的起点。这一章介绍了在确定.NET 应用程序性能低下的原因方面可用的工具和技术，即使具体问题的答案不在本书中，这些工具和技术也可以帮助读