PROGRAMMER TO PROGRAMMER™

Professional ASP.NET Performance

ASP.NET

性能高级编程

K.Scott Allen
James Avery 等著
侯 彧 译

清华大学出版社

ASP.NET 性能高级编程

K.Scott Allen
James Avery
等著

侯 彧
译

清华大学出版社
北京

北京市版权局著作权合同登记号：01-2002-5386

内 容 简 介

ASP.NET 为创建高性能的 Web 应用程序提供了很多重要功能。

本书全面讲述了提高 ASP.NET 应用程序性能的方方面面，从设计、编码，到测试、监控。全书共包括 9 章和 1 个附录，书中首先讨论了性能的概念和性能的重要性，然后逐步讲解性能的设计，编写高性能代码的原则和实例，如何提高数据处理、数据访问、数据操作和数据表示的性能，使用 WAS 和 ACT 工具测试应用程序，使用系统的性能计数器和自定义的性能计数器监控应用程序。本书注重从整体上讲解如何改进性能，并提供了很多实用的技巧。

本书适合有 ASP.NET 和 Visual Basic .NET 的实际使用经验，又想全面了解如何创建高性能 ASP.NET 应用程序的开发人员阅读。

EISBN 1-86100-755-8
Professional ASP.NET Performance
K.Scott Allen James Avery et al

图书在版编目(CIP)数据

ASP.NET 性能高级编程 /(美)艾伦等著；侯彧译. —北京：清华大学出版社，2003

书名原文：Professional ASP.NET Performance

ISBN 7-302-06495-4

Ⅰ.A… Ⅱ.①艾… ②侯… Ⅲ.主页制作—程序设计 Ⅳ.TP393.092

中国版本图书馆 CIP 数据核字(2003)第 022920 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.com.cn
责任编辑：于平
封面设计：康博
版式设计：康博
印 刷 者：北京鑫丰华彩印有限公司
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16 **印张**：20.25 **字数**：518 千字
版　　次：2003 年 4 月第 1 版 2003 年 4 月第 1 次印刷
书　　号：ISBN 7-302-06495-4/TP・4881
印　　数：0001～4000
定　　价：45.00 元

前　言

如果想让您创建的 ASP.NET 应用程序运行得越来越快，并且能够得到扩展以处理更多的并发用户，那么就请阅读本书吧。

本书将讨论提高 ASP.NET 应用程序的性能的许多途径—— 设计、编码、测试和监控。

我们先来讨论什么是性能，以及性能的重要性。接下来了解一个重要的概念：感觉到的性能(perceived performance)——毕竟，不论您有多少统计数字证明您的应用程序运行得非常快，但假若用户觉得您的站点运转不好，那么它就是运转不好，就这么简单。然后学习如何编码取得高性能，如何提高数据性能。本书的后半部分讲述测试应用程序，并生成关于应用程序和 Web 服务器性能的可测量的结果，测试条件是模拟用户生产环境下的工作负载。

本书内容概要

第 1 章首先概述性能的概念和性能的重要性，以及如何保证开发站点的性能达到要求。本章还将讲解性能和可伸缩性的区别，扩展一个站点的方法，以及一个项目的整个周期中的开发工作如何影响最终结果。我们还将介绍“性能即过程”(performance as process)的概念——如何在项目的整个生命周期中考虑性能的提高，从最初的要求到站点投入运行到以后的运转。

第 2 章开始讲解通过 ASP.NET 页面编码节约性能开销的方法。本章先讨论页面本身，然后讨论如何通过管理状态和安全达到性能开销上的节约，接下去讲述的是不同类型的控件和它们各自的问题。最后，探讨@Page 指令属性和它们对页面性能的影响。

第 3 章涉及的是管理用户对性能的期望，从在设计阶段设计性能的期望讲起，包括静态内容和动态内容的用途比较。然后讨论设计可伸缩对象、把 UI 和业务模式分离的重要性，以及 UI 设计的基础知识。本章在.NET Remoting、Web 服务和与 COM 组件的集成方面，讨论将内容和外部系统集成并同时管理感觉到的性能的方法。最后，本章讲解在背景处理过程中感觉到的性能问题，并建议管理的方法。

第 4 章讲述在.NET 环境中编写高性能代码的原则和示例。我们经常用代码为用户完成一个任务所花的时间来衡量它的性能。有的时候可以选择一个技巧或算法既提高应用程序的性能，又提高它的可伸缩性，但也有时这两个目标彼此矛盾。本章将讨论的问题包括保存资源，用缓存提高性能，有效使用集合和值类型，使用 Trace 功能测量页面操作的时间。

第 5 章学习提高数据处理、数据访问、数据操作和数据表示的性能的方法。本章将探讨如何使用.NET Framework 提高数据处理性能，如何改善底层数据库的性能。将涉及在数据读取器和 DataSet 之间进行选择，数据绑定优化和处理字符串与 XML 这样的话题。我们还要讨论如何创建高效的查询，如何利用存储过程，以及如何通过仔细选择索引等改善数据库本身的性能等问题。

在部署 Web 应用程序之前，需要对它进行适当地测试——在应用程序完全部署后预期的工作负载下进行测试。第 6 章介绍 Microsoft 提供的一个免费实用工具 Web Application Stress Tool，可以用这个工具模拟大量用户访问站点的情况，然后检验应用程序和 Web 服务器的性能，但本章着重于检测 Web 服务器的性能。我们将深入探讨这个 WAS 工具，然后举几个例子来说明如何使用该工具测试和改善应用程序。

第 7 章介绍 Microsoft 的另一个工具 Application Center Test (ACT)——该工具模拟多个 Web 客户端同时向 Web 服务器发出 HTTP 请求的情形，它通过再现应用程序在实时数据中心的预期负载下的运行情况来优化应用程序的性能。该章将比较 ACT 和 WAS 之间的区别，用 ACT 测试 ASP.NET 应用程序和 Web 服务，讨论 cookie、身份验证和其他有用的 ACT 提示。

本书的最后一章介绍如何监控 Web 应用程序的性能。虽然在部署应用程序之前已经对它们进行了测试，在应用程序和服务器处于生产环境中以后仍需要继续监控它们的性能。本章讨论两种跟踪应用程序性能的方法——Performance Monitor(性能监控器)和 Performance Counter(性能计数器)。我们将学习如何精确地监控和诊断 ASP.NET 应用程序和整体的系统——如何使用默认的 ASP.NET 性能计数器来监控和提高 ASP.NET 应用程序的性能，如何创建自定义的性能计数器，以及如何直接从 ASP.NET 应用程序访问和使用性能计数器。

本书读者对象

本书适合已经掌握 ASP.NET 和 Visual Basic .NET 的应用知识，需要了解创建高性能 ASP.NET 应用程序的所有方面的 ASP.NET 开发人员阅读。

本书针对的是学过.NET，尤其是学过 ASP.NET 的开发人员。建议事先阅读清华大学出版社引进并出版的《ASP.NET 1.0 高级编程》一书。

说明：

因为本书中的语法和代码示例都是用 Visual Basic .NET 编写的，我们假设您了解这门编程语言。

使用本书的条件

使用本书的前提是您的计算机上安装了.NET Framework。因此您需要运行安装 IIS 的 Windows 2000 Professional (或更高版本)，或 Windows XP Professional。

还建议使用 Visual Studio .NET Professional (或更高版本)。

第 7 章要求使用 Visual Studio .NET Enterprise Developer 版和 Visual Studio .NET Enterprise Architect 版打包的 ACT 工具，因此需要有其中一个版本，以充分地学习本章的内容。

第 5 章，“数据性能的开发”，要求使用一个数据库服务器——虽然这个服务器可以和您的.NET Framework 安装在同一台物理机器上。本书使用的数据库服务器是 Microsoft SQL Server 2000。

示例文件可从 http://www.wrox.com/下载。

客户支持

我们一贯重视读者的意见，并想知道每位读者对本书的看法，包括读者喜欢和不喜欢的内容，以及读者希望我们下一次完善的地方。您可以通过发送电子邮件(地址为feedback@wrox.com)向我们反馈意见。请确保反馈信息提到本书的书名。

本书示例代码的下载

当您访问 Wrox 公司站点(地址为 http://www.wrox.com/)时，通过 Search 工具或书名列表，可以方便地找到需要的书目。然后，单击 Code 列中的 Download 超链接，或者单击本书详细页面中的 Download Code 超链接，就可以下载相应的示例代码。

当您单击下载本书中的代码时，将会看到带有以下三个选项的 Web 页面：

- 如果您是 Wrox Developer Community 的成员(即如果您已经在 ASPToday、C#Today 或者 Wroxbase 上注册)，就可以使用一贯的用户名和密码进行登录以下载代码。
- 如果不是它们的成员，则会向您询问是否愿意注册为会员，以便可以免费下载代码。此外，也可以从 Wrox Press 下载免费的文章。注册为会员后，可以得到本书升级版本和新版本的有关信息。
- 第三个选项是完全绕过注册过程，直接下载代码。

对于本书而言，不注册也能下载代码。但是，如果您愿意注册后下载代码，您的注册信息不会泄漏给第三方。关于这方面的详细条款和条件，可以通过单击下载页面上的相关链接来查看。

从我们的站点上下载的文件都是使用 WinZip 压缩过的文档。保存文件到本地硬盘上的文件夹中后，需要使用解压缩程序(例如 WinZip 或 PKUnzip)来解压缩文件。在解压缩文件时，通常将代码解压缩到每一章所在的文件夹中。在解压缩的过程中，应确保解压缩程序(WinZip、PKUnzip, 其他)被设置为使用原有文件夹名。

勘误表

我们已经尽最大努力确保本书中的文本和代码没有错误，但是错误仍然在所难免。如果您发现本书中存在错误，例如拼写错误或不正确的代码段，请反馈信息给我们，我们将不胜感激。勘误表的发送可以节省其他读者学习本书的时间，而且能够帮助我们提供更高质量的信息。您的反馈信息将被检查，如果正确，将被粘贴到本书的勘误页面上，或者在本书的后续版本中使用。

要在我们的站点上找到勘误表，请访问 http://www.wrox.com/，并通过 Advanced Search 或者书名列表轻松找到本书页面。然后，单击 Book Errata 超链接即可，该链接位于本书详细页面中的封面图解的下面。

E-mail 支持

如果您希望直接向详细了解本书的专家咨询本书中的问题，可以发送电子邮件到 support@wrox.com，要求在邮件的主题栏中带上本书的书名和 ISBN(国际标准图书编号)的后 4 位数字。一个典型的电子邮件应包括下面的内容：

- 在主题栏中必须有本书的书名、ISBN 的后 4 位数字和问题所在的页码。
- 邮件正文中应包括读者的名字、联系信息和问题。

我们将不给您发送无用邮件，因为我们仅仅需要有用的详细资料，以便节约您和我们的时间。当您发送一个电子邮件信息时，它将经过下面一系列支持：

- 用户支持：首先，您的信息将被递送到我们的用户支持人员手中，并由他们阅读。对于一些频繁提到的问题将被归档，并将立即回答有关本书或者 Web 站点的任何常见问题。
- 编辑支持：接着，一些深层次的问题将送到对本书负责的技术编辑手中，他们在程序设计语言或者特定的产品上有着丰富的经验，能够回答相关主题的详细技术问题。
- 作者支持：最后，如果编辑不能回答您的问题(这种情况很少发生)，他们将请求本书的作者回答。我们将尽量保护作者免受干扰，以便不影响其写作。然而，我们也非常高兴转寄给他们一些特殊的问题。所有 Wrox 公司的作者都为他们的书提供技术支持。作为回应，他们将发送电子邮件给用户和编辑，进而使所有的读者受益。

Wrox 公司的支持过程仅仅对那些与我们出版的书目内容直接相关的问题提供支持，对于超出常规书目支持的问题，您可以从 http://p2p.wrox.com/论坛中的公共列表中获得支持信息。

p2p.wrox.com 站点

为了便于作者和其他人讨论，特将讨论内容加入到 p2p 站点的邮件列表中，而且我们独特的系统将 programmer to programmer™(由程序员为程序员而著)的编程理念与邮件列表、论坛、新闻组以及所有其他服务内容(一对一的邮件支持系统除外)相联系。如果您向 p2p 发送一个问题，应该相信它一定会被登录邮件列表的 Wrox 公司作者和其他相关专家检查到。无论您是在阅读本书，还是在开发自己的应用程序，都可以在 p2p.wrox.com 站点中找到许多对自己有所帮助的邮件列表。

按照下面的步骤可以预订一个邮件列表：

(1) 登录 http://p2p.wrox.com/站点。
(2) 从左边的菜单栏选择一个适当的类别。
(3) 单击希望加入的邮件列表。
(4) 按照说明订阅并填写自己的邮件地址和密码。
(5) 回复您收到的确认邮件。
(6) 使用预订管理程序加入更多的邮件列表并设置自己的邮件首选项。

本系统能提供最佳支持的原因

您可以加入整个邮件列表，也可以只接收每周的邮件摘要。如果您没有时间和工具来接收邮件列表，可以直接查找我们的在线文档。独特的 Lyris 系统可以将一些没有用的垃圾邮件删除，并保护您的电子邮件地址不被侵扰。当存在加入和离开列表、以及任何有关列表的其他常见问题时，请发送邮件到 listsupport@p2p.wrox.com。

目　　录

第1章　性能的概念

无论您的 ASP.NET 应用程序有多么好，您在 Web 站点中添加了多少很酷的新特性，如果您的站点在处理用户请求时哪怕只是让用户等待了几秒钟，用户也不会对您的站点留下好印象。事实上，如果用户等得太久，他们就会放弃而去访问另一个站点。本书将讲述提高 ASP.NET 性能的许多不同的方面——如何保证应用程序尽快地运行，应用程序如何能够扩展以处理更多的并发用户，如何让应用程序不仅运行得快，还要让用户感觉到它快。我们还将学习分析性能、测试应用程序可伸缩性的工具和方法，以使我们能够识别(并因此减少)系统中的瓶颈。

在这介绍性的第 1 章中，我们将概述什么是性能，为什么应该重视性能，如何确保开发的站点的性能达到要求。还将了解性能和可伸缩性之间的区别(站点在不严重降低性能的前提下同时处理多个请求的能力)，扩展站点的方法，以及项目的整个生命周期中开发工作对最终结果的影响。最后，介绍“性能即过程”的概念——如何在项目的整个生命周期中考虑性能的提高，从最初的要求到站点投入运行，以及运行以后的运转。

1.1　性能如此重要的原因

在详细讨论性能的概念并简要介绍几种测试性能的方法之前，先花一点儿时间来考虑为什么性能如此重要——很可能其重要性仅次于一个站点如何运转。一个性能很差的站点是没有用的，如果您的站点太慢，不久您就会发现潜在的用户全都跑光了。

1.1.1　从最终用户的角度看待性能

访问一个响应不能如您所愿的站点是很令人无奈的。您有过多少次填好了一个表单，单击了提交按钮，站点却迟迟没有响应？很有可能您就离开此站点，而去访问另一个站点。可用性专家(参考 www.useit.com 上的 Jacob Nielson 的访谈) 告诉我们站点响应时间是保证站点可用性的最重要因素之一。一般的原则是用户等待页面下载的时间上限为 10 秒钟。超过 10 秒钟，用户就会感到不耐烦，转而访问另一个站点。显然，站点速度越快，用户越不会在页面下载之前就放弃。如果您运行的是一个电子商务站点，性能对您的业务量会有直接的影响。

从用户的角度出发，一个 Web 站点不管提供什么其他的好处——漂亮的设计，华丽的视觉效果，降价且转天送货的很棒的产品，或者是高质量的信息——但是，如果站点的性能不能让人接受，所有其他的艰苦工作都会白费。但这里有一个权衡的问题——如果花费了一百万美元保证良好的伸缩性和性能，而站点只能多产出 50 万美元，就没有意义了。

1.1.2　经济状况

一个高性能的 Web 站点还能降低成本，增加价值。尽管硬件成本在下降，性能在提高，但

确保高性能软件最好、最有效地利用硬件总是合情合理的。除了最初购买硬件的成本之外，还应考虑总体拥有成本。这包括维护和监控硬件的人员成本、托管成本、许可证成本、连接性成本和电力消耗成本。在这些成本中，许可证成本经常比硬件成本更重要。

我们把这些成本加起来，看看较差性能的成本是多少。考虑有两个 Web 站点 A 和 B。站点 A 在整个设计过程中都采用最佳做法。站点是高度优化的，适当地使用缓存，伸缩性高。站点 B 是临时开发的，没有质量控制或很少质量控制。站点 A 的一台服务器可以处理 1200 个并发用户，伸缩性很好。而站点 B 的一台服务器只能处理 300 个并发用户，伸缩性很差。

假设站点在使用高峰期需要处理 3000 个并发用户。再假设每台服务器的软件许可证成本是 5000 美元，服务器本身的成本是 3000 美元，每单位的机柜空间的电力、托管和连接性成本是每年 250 美元。

- 站点 A 要求使用 3 台服务器，所以第一年的成本是 24750 美元。
- 站点 B 因为伸缩性不好而要求 15 台服务器，第一年的成本为 123750 美元。设计差、性能低的站点的成本要贵 4 倍。这个数字还不包括安装、配置或维护多出来的这几台服务器的成本(或存放它们需要的额外空间)。

用 3 台服务器，站点 A 就可以轻松地处理高峰期的使用。但站点 B 已经到了伸缩能力的极限，如果使用超过了 3000 个并发用户，它的性能就会开始降低。这时候添加服务器会带来更糟的结果，因为数据库开始成为瓶颈。但对于站点 A，处理更多的通信量只需要另外一台或几台服务器。站点 B 由于其性能差和伸缩性差，而成本高昂，还限制了业务扩展的能力。

图 1-1 显示了一般的模式——虽然站点 B 开发的成本低，但总体拥有成本会迅速上升，而较差的伸缩性不久也会限制其进一步的开发。对比之下，站点 A 的成本一直相对比较平稳，所以它能较快较好地增值。

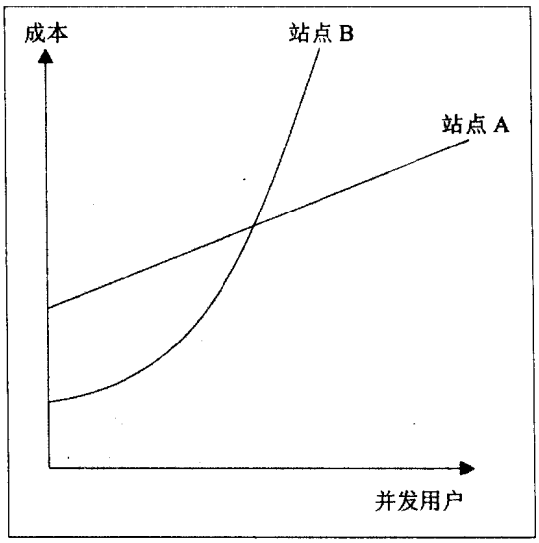

图 1-1

一个开发时注意充分利用硬件，了解硬件的性能数据和伸缩能力的站点是极具价值的，并可以降低成本。

1.1.3　增长的能力

站点随业务增长的能力是很关键的。如果站点不能进行扩展以满足业务不断增长的需求，不论是内部需求还是外部需求，它都会限制业务的发展。这说明了认真对待可伸缩性和性能问题是多么重要。

1.2　性能的含义

性能表示系统执行任务的速度和效率。从许多不同的角度来看待 Web 站点的性能。最重要的因素是从用户的角度看性能是怎样的。Web 性能很有意思，因为对 Web 上的性能会有很多不同的解释。有 500K ADSL 宽带连接的用户可以认为一个 Web 站点响应性好、性能高，而一个使用标准的 56K 调制解调器连接的用户会认为同一个站点几乎无法使用——性能要看访问的内容类型和连接的设备是什么。

先花一点儿时间探讨到底什么是高性能的 Web 站点。最重要的测量数据之一是站点能够服务的每秒页面数(有时称为页面吞吐量)。根据页面的特征(例如，需要访问什么资源)，不同的页面有不同的吞吐量。每秒服务的页面越多，您的站点能够同时服务的用户就越多。每秒页面数经常和其他两个数据一起衡量：

- TTFB ——从请求到收到第一个字节的时间
- TTLB ——从请求到收到最后一个字节的时间

就它自己来说，站点的每秒页面数只能反映一半的情况。还需要了解站点在服务这些页面时的表现如何。TTFB 和 TTLB 的值越小，两个值越接近，用户感觉站点响应性越好。随着站点用户的增加，TTFB 和 TTLB 将开始增加(用户会感觉站点很慢)。TTFB 和 TTLB 会增长到一个令人无法接受的程度(可不可以接受要看您开发的应用程序的本质和预期的响应时间)。此时，您的站点已经达到了最大的容量。它还可以继续提供页面，但从可用性的角度看，性能将下降很多，以致站点无法再使用。

一般来说，CPU 占用率到了一定的比例，站点会达到峰值容量，所以在进行测试或监控站点运行时要注意 CPU 的占用率。如果 CPU 占用率为 100%时站点达到峰值容量，响应时间会变慢，因为请求的处理必须等待 CPU 周期才能完成处理。作者发现 CPU 占用率为 80～90%时达到峰值容量比较好，因为如果 CPU 占用率经常保持在 90%的话，它有时可能会达到 100%。不管达到 100%的时候有多么少，达到了就会引起性能的下降，这样的事情一经发生就会成为恶性循环，因为用户会感到厌烦并不断刷新页面，把问题变得更糟糕。

最大容量是如何测量的

站点的最大容量是在维护可接受的响应时间的条件下能够处理的并发连接的总数。最大容量总是一个指导原则，而不是一个精确的数字，因为根据对站点使用的特性，这个值会发生变化(本章后面将讨论这个话题)。

因为 HTTP 是一个无状态的机制，没有用户被“连接”到一个 Web 站点的概念——我们所知道的就是某个用户请求了一个或几个页面。我们不了解他们是否将请求另一个页面，还是已经离开了。因此，站点使用估算并发用户的近似值的方法。例如，Windows 2000 的