TURING 图灵程序设计丛书 微软技术系列

Apress®

Ultra-Fast ASP.NET

Build Ultra-Fast and Ultra-Scalable web sites using ASP.NET and SQL Server

构建高性能可扩展ASP.NET网站

[美] Richard Kiessig 著

余昭辉 译

- 让ASP.NET和SQL Server网站飞
- 迅速提升网站性能，全面挖掘网站潜力
- 微软资深技术专家力作，亚马逊全五星评价

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Ultra-Fast ASP.NET

Build Ultra-Fast and Ultra-Scalable web sites using ASP.NET and SQL Server

构建高性能可扩展ASP.NET网站

[美] Richard Kiessig 著
余昭辉 译

人民邮电出版社
北京

图书在版编目（CIP）数据

构建高性能可扩展ASP.NET网站 / (美) 基斯格(Kiessig,R.) 著 ; 余昭辉译. -- 北京 : 人民邮电出版社, 2011.3
(图灵程序设计丛书)
书名原文: Ultra-Fast ASP.NET:Build Ultra-Fast and Ultra-Scalable web sites using ASP.NET and SQL Server
ISBN 978-7-115-24833-6

Ⅰ. ①构… Ⅱ. ①基… ②余… Ⅲ. ①主页制作－程序设计 Ⅳ. ①TP393.092

中国版本图书馆CIP数据核字(2011)第012631号

内 容 提 要

本书针对ASP.NET网站开发中可能遇到的问题，给出了经过实践检验的具体解决方法。涉及的内容包括：加快显示HTML的方法、缓存的最佳方式、如何使用IIS、如何处理会话状态、如何配置SQL Server以及如何优化基础设施等。

本书适用于所有Web开发和运维人员以及对优化网站感兴趣的读者。

图灵程序设计丛书

构建高性能可扩展ASP.NET网站

◆ 著 [美] Richard Kiessig
译 余昭辉
责任编辑 明永玲
执行编辑 丁晓昀

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：24
字数：597千字 2011年3月第1版
印数：1－3 000册 2011年3月北京第1次印刷

著作权合同登记号 图字：01-2010-2367号
ISBN 978-7-115-24833-6

定价：65.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

译 者 序

长期以来，在.NET Web 开发领域一直缺少一本用来讲解性能优化的书籍，大多数.NET Web 开发书籍都将大量笔墨倾注在控件使用、控件开发以及数据源操作上。当性能出现问题时，有的.NET Web 开发人员要么束手无策，因为这些控件对于他们来说就像一个黑盒；要么去求助其他平台的 Web 性能优化书籍，但这些书籍往往没有针对.NET 的特殊处理，优化效果也大打折扣。

现在，.NET Web 开发人员终于迎来了这么一本属于我们自己的 Web 性能优化手册。这本亚马逊五星级图书，本着压榨每一个字节的态度从 Web 前端优化、缓存、服务器到数据库后端各个方面给.NET Web 性能优化提供端到端的指导。书中包含许多真实的代码，大部分都可以在自己的项目中直接使用，有些优化只需要花费少量的时间，就可以收到立竿见影的回报。

本书分为 11 章，第 1 章是原则和方法，给全书定下了一个基调：我们要压榨 Web 开发中每个字节。第 2 章讨论客户端性能，包括网络连接、CSS 以及 JavaScript 等资源的优化。第 3 章是本书的重头戏——缓存，缓存在 Web 性能中可谓是一哥，这章按照缓存离用户的远近顺序做了全面的介绍。做 ASP.NET 肯定离不开 IIS，第 4 章以 IIS 7 为例，对 IIS 调优进行了讲解，原来 IIS 不是配一下网站就万事大吉了。第 5 章介绍了线程与会话，对于构建可扩展的 Web 架构尤为重要。第 6 章从 ASP.NET 特有的特性来讲解其对优化的作用，比如母版页，比如针对特定浏览器生成输出等。第 7 章介绍了 `HttpHandler` 和 `HttpModule` 等方面的内容，对于习惯拖控件构建 Web 页面的开发人员，这一章会让你看到 ASP.NET 的另一个世界。第 8 章、第 9 章介绍了 SQL Server 的优化，从缓存、索引等方面做了详细的介绍，第 8 章也是本书篇幅最大的一章。第 10 章从运维的角度来讨论优化。第 11 章是集成，将前 10 章所有的东西结合在一起使用。

本书的翻译和编辑工作差不多贯穿了 2010 年整整一年，其中因为工作变动对交付进度造成很大的影响，在这里要特别感谢图灵公司的编辑的体谅和支持。感谢好友王雄对部分章节的审阅。

最后，要感谢我的女友照云，非常感谢你这几个月的陪伴，对我因赶进度对你的冷落毫无怨言，甚至还自告奋勇帮忙审稿。

关于作者

1979年我在加州大学圣巴巴拉分校（UC Santa Barbara）毕业，获得数学学士学位。毕业后我去了兰德公司（Rand Corporation），继续从事Unix、C以及Internet相关工作。20世纪80年代，我去了硅谷（Silicon Valley），专攻底层操作系统，性能调优以及面向网络的应用。我曾从零编写过类Unix的操作系统，包括一个高性能文件系统。我还开发了一个基于XNS的网络栈，并且帮Intel的架构师开发了第一个从Unix到x86的移植版本。我还编写过一些3D的科学动画系统，以及一个门阵列模式布局（gate array placement）包。

20世纪90年代，我编写了一个自定义实时操作系统，用在美国海军F-18飞机上。我开发的实时应用还用在航天器和相关的地面支持系统上，包括一个名为Stellar Compass的系统，使用行星的数字图像来测量飞行器状态。该软件曾登陆过一次月球，三次火星，而且与一个彗星打过照面并返回。我还曾经做过世界一流的图像卫星地面系统和各种飞行软件组件的首席架构师和设计者。

当我第一次听到Java时，我就疯狂地爱上它了。我使用Java开发的第一个大规模系统是一个音频会议系统。在这之后，我使用其开发了一个定制的高性能应用服务器。我帮助架构并构造了许多大规模基于Java的，数据密集型网站和Web应用，其中一个应用可以部署到两千万机顶盒中，为TV提供Internet支持。我最后一个基于Java的项目是构建一个面向文档管理的文件系统，我是许多相关模式的主要开发者。现在，许多金融机构还使用该系统解决风险管理问题。

在1999年末我去了Microsoft。我的第一个项目是使用C#、ASP.NET和SQL Server开发一个通过面向TV的中间件平台分发MSN内容的综合架构，比如WebTV。几年后，在开发完原始的系统后我进入了微软技术中心（Microsoft Technology Center），我的工作是建议Microsoft的大客户们关注他们系统架构中的.NET和SQL Server。

将我的职业生涯联系在一起的主线是关注性能和可靠性。软件开发过程是我的另一个长期兴趣点，因为我曾直接看到它对项目的成功或失败有多大的影响。

在2006年12月，我们一家离开了压抑的硅谷，搬到了美丽的新西兰，就是我现在居住的地方。我的业余爱好包括业余无线电（呼号是ZL2HAM）以及摄影。

前　言

我在微软的工作经历给我的人生带来了意想不到的巨大转变。到微软之前，我所在的公司都是微软的竞争对手，时不时会听到反微软的言论和宣传。但是，当我听说.NET 后，我就知道应该了解更多，而最好的方式就是到它的发源地去学习。

在进入微软公司深入了解这些技术之后，我被眼前的一切震撼了。.NET Framework、C# 语言、ASP.NET 以及 SQL Server 都非常先进，都有非常漂亮的技术架构。在使用 Java（无疑也是非常优雅的语言）数年后，我可以使用一个集成良好的平台，将（几乎）所有东西无缝地集成在一起工作，确实令人精神振奋，干劲倍增。从技术层面上讲，我很认同平台开发人员的决策和权衡，而且最终的系统也能相当大地提高开发人员的效率。微软的工程团队也人才济济，极具创新意识，而给我这个曾经圈外人带来的最大惊奇是，他们打心眼儿里想要解决客户的问题。

我对技术的热衷让我成了直面客户的人——担任位于硅谷的微软技术中心的解决方案架构师。直面深层次的客户问题是另一项令人大开眼界的经历。首先，我可以在许多人或公司那里体会到微软技术产生的积极影响。其次，我还能感受到有些人面对很差劲的结果时的强烈挫败感。这本书，在某种程度上就是为减少挫败感而写的。

我的观点是 ASP.NET 和 SQL Server 很有潜力。但是，这些技术的关键点还未被很多开发人员认识到。我曾经与许多想要挖掘这种潜力，但却苦于不得其门的开发人员和经理交谈过。可是，认识到该技术全部的潜力必须投入更多努力；其中的特性丰富多彩，但要完全领略其风采必须有正确的视角。本书的一个目的就是赶走遮住技术美景的层层迷雾，让你亲眼目睹 ASP.NET 和 SQL Server 的美丽。

写这本书的另一个原因是我常被一些很慢的网站所折磨，希望本书介绍的内容能帮你改变这个现状。在 Web 的世界里一切皆有可能，未来的 Web 一定会超过现在的水平——但良好的性能是实现它们的基础。到那时谁也不会再遇到缓慢的网站。

我现在使用的是高于 3 Mb/s 的 DSL 连接到 Internet，我的台式机 4 个 CPU 内核都工作在 3GHz，与过去几年相比是出奇得快。即使这样的连网速度和处理能力，加载许多网站仍然要花很长时间，有时甚至加载一个页面都要好几分钟，而此时我的本地网络和 CPU 几乎是空闲的。作为软件专业人员，我们应该关注这种问题，因为出现这种问题实在太丢人了。我不仅要自己做得更好，更想让我的职业为人称道。为此，我们开发的网站就不仅要快，还要超快！

读者对象

本书的前两章和最后两章提供的信息几乎对所有 Web 开发人员都有用，且与使用的底层技术无关。中间 7 章是为使用 ASP.NET 和 SQL Server 设计、构建或维护网站的中高级的架构师和开发人员写的。刚刚从 Java 或 PHP 转到.NET 的有经验的 Web 开发人员也会从本书找到很多有价值的信息。

本书也非常适合那些不是开发人员，但对优化网站技术感兴趣的读者。尤其是对参与网站运维、测试或管理的相关人员，这本书给出了许多开发团队应该重视的原则和问题，并提供了示例帮助理解。

联系作者

可以通过 rick@12titans.net 联系到我，也欢迎访问我的网站 http://www.12titans.net/。

我非常想听到你关于高性能方法的经验。

提高性能和可扩展性的技术会随着底层技术的变迁不断地发展。我非常想听到你发现很有效，但本书却没有涉及的技术。

欢迎指出正文或代码示例中的错误，也欢迎各种改进本书的建议。

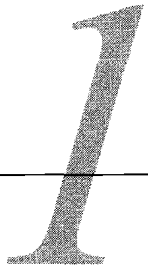
致谢

我要感谢 Apress 优秀的团队：Ewan Buckingham 早期给了我大力支持和鼓励；Matthew Moodie 对全书的结构和流程提供了帮助；Simon Taylor 负责了技术审校，还复查了代码示例；Anita Castro 负责项目管理，Kim Wimpsett 和 Tiffany Taylor 帮我润色了本书的文字。

我还要感谢 Phil de Joux 的反馈意见。

目　录

第1章 原则和方法



建设现代的大型网站是一项令人惊讶的复杂工程。许多网站随着规模的扩大，都会遇到严重的性能和可扩展性问题。我经常见到一些大型网站在某一时刻，为了应付不断增长的访问量而不得不将设计推倒重来。所幸的是，如果能在网站规模还不大的时候就坚持遵循一些基本原则，不仅可以让当前网站的速度更快，而且还能避免将来网站成长过程中的一些问题。

本书会探讨这些原则，并帮助你理解为何及如何应用这些原则。

这些原则是我在近30年互联网软件开发实践的基础上总结出来的。从1974年开始我就从事因特网方面的工作，1979年开始使用Unix和C，后来转到C++，接着又加入到Java和C#阵营。在微软工作的日子里，我深入地学习了ASP.NET和SQL Server。在那里，我为MSN TV设计并开发了一个大型网站。后来，在位于硅谷的MTC（Microsoft Technology Center，微软技术中心）担任架构师的几年里，我对建设大型网站有了更深刻的理解。当时，我每周都要给微软的大客户们上一两堂架构设计课，每次两到三天。我和其他MTC的架构师会首先了解客户的烦恼和问题，然后从架构的角度提出解决方案。

没过多久我就发现，许多人都碰到了相同的问题，而其中大部分都是围绕性能和可扩展性的。以下是我们常听到的问题。

- 如何才能让HTML显示得更快？（第2章）
- 缓存的最佳方式是什么？（第3章）
- 如何使用IIS让网站更快？（第4章）
- 如何处理会话状态？（第5章）
- 如何改进ASP.NET代码？（第5～7章）
- 我的数据库为什么这么慢？（第8章和第9章）
- 如何优化基础设施和运维？（第10章）
- 从哪儿开始改进性能？（第11章）

本书会针对这些问题给出有效的解决方案。

我的做法是，不要将网站仅仅看成运行在远程服务器的应用程序，而要将其看成分布式的组件集合，这个集合应该像一个完整系统那样很好地协作。

本章首先从性能和可扩展性的差异谈起，接着谈一谈我对高性能和高可扩展性的理解。然后我从一个较高的层次来讲解网页生成的完整过程，并且描述一些核心原则，这些原则是提升网站

性能的关键原则。最后介绍一下本书中示例所采用的环境和工具。

1.1　性能和可扩展性的差异

当有人对我说他希望系统运行得更快的时候，我就会问："你所说的快是什么意思？"典型的回答大概是："应该能支持成千上万的用户。"即使是一个很慢的网站，也可以支持成千上万的用户。实际上，一些大型网站都很慢。

可扩展性和性能是完全不同的概念。在本书中，当我谈到提升网站性能时，我的意思是减少特定页面加载的时间或完成对特定用户可见的操作的时间。用户坐在电脑前所看到的就是"性能"。

另一方面，可扩展性与网站可以支持的用户数有关。可扩展性高的网站很容易通过添加更多的硬件和网络带宽（不对软件做重大变更）支持更多的用户，而且对性能的影响很小或没有影响。如果用户的增加导致网站严重变慢，而且添加硬件或增大带宽还不能解决问题，那么就说明网站已到了扩展的临界值了。可扩展性设计的一个目标就是提高这个极值，这是一个永恒的话题。

1.2　为什么需要高性能和高可扩展性

速度和可扩展性不仅仅是 Web 服务器端的话题，Web 开发的许多方面都能也都应该提升速度和可扩展性。所有代码都应该快速运行，不管是运行在客户端、Web 层还是数据层。所有页面（而不是其中一少部分）都应该快速显示，而且变更、修正以及部署也应该够快。

随着性能和可扩展性目标在项目中日益深入人心，一个良性互动的局面就会形成。不仅客户和用户会更满意，工程师也会更高兴并觉得更具挑战性。也许有人会不相信，其实做到这一点通常不需多少硬件，质保及运维团队也不用很多人。这就是我所说的高性能和易扩展的含义（简称高性能，易扩展则隐含其中）。

高性能不是靠"想到哪做到哪"的编程方式实现的。如果不以一种系统化的方式开发，架构问题就会出现，通过捷径获得的短期效益也会被抵消掉。大规模软件开发项目都是马拉松式的，不是短跑式的，预先的计划和准备可以获得巨大的长期效益。

表 1-1 是我归纳的高性能和易扩展方法的目标。

表 1-1　高性能和易扩展方法的目标

组　件	高性能和易扩展的目标
页面	在负载下，每个页面都是可扩展的，并且响应速度快
层	在负载下，每个层都是可扩展的，并且运行速度快
敏捷性	可以快速地响应业务需求的变化，而且在变化时能够保持性能和可扩展性
可维护性	快速地发现和修正性能相关的bug
运维	快速地部署和扩大网站，容量规划直观且可靠
硬件	服务器在负载下被充分利用，并且尽可能少使用硬件设备

从总体上看，构建运行速度快、可扩展的网站与制造赛车有诸多的相似性。为了达到预期目标，一开始就要考虑和设计与核心性能相关的各个方面因素。对赛车而言，需要确定参与哪种

比赛或哪个俱乐部。是参加F1（Formula One，一级方程式）、普通型汽车赛、重型拉力赛、直线加速赛，或者仅仅参加小型车赛？如果只是为小型车赛设计汽车，那么你不仅不能参加F1，而且，如果想参加新等级的比赛，还要完全抛开原来的设计，然后重新开始。对于网站来说，为你自己和朋友构建网站当然与构建eBay或Yahoo那样的大型站点完全不同。适合一个网站的设计可能完全不适合另外一个网站。

高级的赛车不仅要跑得快、换胎要快、加油要快，而且更换引擎也要快。换句话说，赛车的快是多种维度的。网站也应该在多个维度里表现得都够快。

与设计赛车只考虑让它跑得快但没有考虑安全性一样，如果设计网站时只考虑高性能而忽略安全性的话，那后果也是不堪设想的。因此，在接下来的几章中我会时不时地讨论有关安全性的问题，这些问题与性能都是密切相关的，比如第3章中对Cookie的讨论。

1.2.1 优化

许多业界的专家已经指出，优化有时候就是死亡陷阱，只会浪费时间。构建高性能网站的关键是通过缜密的设计，让网站完全不需要优化。不过，就像赛车一样，如果想获得最好的成绩，就需要在研制过程中反复测量、调整、调校、修正，并有所创新。只要你有时间、资金和想法，就一定能在某些地方做得更好。

在实际工作中，关键是要知道哪里会有性能和可扩展性问题，哪种变更能带来最大的性能提升。比较几个车轮螺栓的重量简直就是在浪费时间，但是使用合适的引擎混合比则可能赢得比赛。同理，改进调用不怎么频繁的函数不会提高网站的可扩展性，而使用异步页面则可以做到。

我的意思并不是说这些小事情不重要。实际上，许多小问题也会迅速累积成大问题。但是，当给任务设定优先级，为其分配时间时，就要先关注能产生显著效果的任务。对赛车做抛光处理，也许可以让车跑得更快点，但如果变速器不怎么样，你就得先处理这个问题。给内部的API换个好记的名字也许会让你自己觉得很满足，但更重要的还是减少客户端与服务器间的往返次数。

1.2.2 过程

高性能是一个追求，一个过程。从架构和设计开始，涉及系统的所有方面，从开发到测试、部署、维护、升级以及优化。但是，像制造赛车或其他任何复杂的项目一样，在这个过程中，你常常会有紧迫感，期望尽快完成，只要“够好”就行了。在确保商业目标的同时，还要有效地提升网站性能，关键就在于抓住那些对性能影响最大的核心问题。本书关注的是应该做的事情，而不是探索所有可以做的事情，目标是帮助你关注高影响的领域，避免迷失在杂乱的过程中。

我曾经工作过的许多团队的管理层都难以接受在性能上花费精力。通常这些团队都会碰到性能危机，而且这些危机有时还导致站点的推倒重来。管理层必然更关注功能，而只要性能“足够好”就可以了。问题是在真正出现问题之前性能都会足够好。根据我的经验，要想让管理层重视性能问题，就不能把它列为一项功能。它不是功能，安全和质量问题都不是功能。性能和其他与速度相关的方面是应用程序不可分割的一部分，它们会影响每个功能。如果要制造一辆赛车，让其跑得快不是一个可以在最后添加的额外功能，它是总体设计的一部分，应该在每个组件和每个过程之中都考虑到这个问题。

这里没有捷径。下面是完成这项工作的关键点。

- 全面深入理解系统的方方面面。
- 构建稳固的架构。
- 关注最主要的问题，知道哪些可以不用考虑或延期考虑。
- 懂得“凡事预则立，不预则废”的道理。
- 使用合适的软件开发过程和工具。

你也许听到过一些类似针对 Web 性能的“8 秒钟原则”。这是一种人为因素指标，意思是如果一个页面的加载时间超过 8 秒，用户就会感到不耐烦，从而去点击其他同类网站。本书会采取完全不同的方式而不是关注类似这样的原则。我们不关注主观的性能指标，而是首先立足于架构。就像把赛车放到合适的“赛道上”。然后，遵照一系列有效的指导原则来构建网站。有了这些基础，就无需花费很多精力做优化了。在一开始就应用一些高端的设计技术可以开阔你的眼界。不要等到竞争对手都晋级到 F1 了，你才想起抛弃自己的普通赛车。

1.2.3 体验

性能应该完全围绕用户体验展开。比如，加载整个页面的时间只是全部用户体验的一方面，感知性能甚至更重要。如果有用的内容“立即”显示，而一些广告在 10 秒钟之后显示，大部分用户不会抱怨，甚至许多用户都不会发觉。但是，如果按照相反的顺序显示页面，即本来应该后出现的广告先显示了出来，而有用的内容出现在广告之后，那你就会面临丢失大部分用户的风险，即使整个页面的加载时间是一样的。

一个人构建和维护的小型 Web 站点也可以像构建更大的 Web 站点那样从这种方式中获益（想象普通赛车安装了 F1 赛车配件后的情景）。速度快的站点会比速度慢的站点吸引更多的流量和更多回头客。你也许只需要一个更小的服务器或更便宜的主机。你的用户也许会访问更多的页面。

当你关注架构和性能时，这里有一个使用 ASP.NET 和 SQL Server 的示例：一个软件开发人员独立构建的站点 plentyoffish.com，该站点现在是加拿大最高流量的站点之一。该站点每个月有超过 4 500 万次访问，12 亿页面浏览，每秒 500～600 次页面访问。然而这个网站只使用了 3 台负载均衡 Web 服务器，这些服务器使用的是双四核 CPU，8G 内存，还有一些数据库服务器，以及一个内容分发网络（CDN）。Web 服务器的 CPU 占用率一般在 30%左右。我不知道该站点的其他细节，但是查看它生成的 HTML 后，我有把握你可以使用本书提供的技术构建一个可与之媲美甚至更快的站点。

不幸的是，天下没有免费的午餐：构建高性能的站点比粗制滥造地制作网站需要更多思考和计划，而且需要更多的开发工作，当然大部分的工作量一般都集中在项目开始。经过长时间的运行，维护和开发成本实际会很小，而且还可以避免任何重写的成本。最后，我希望你认可为这些收益所付出的努力都是值得的。

1.3 完整的页面处理过程

人们一般都认为 Web 就是一个浏览器在网络连接的一端，Web 服务器和数据库在连接的另一端，如图 1-1 所示。