Broadview
www.broadview.com.cn

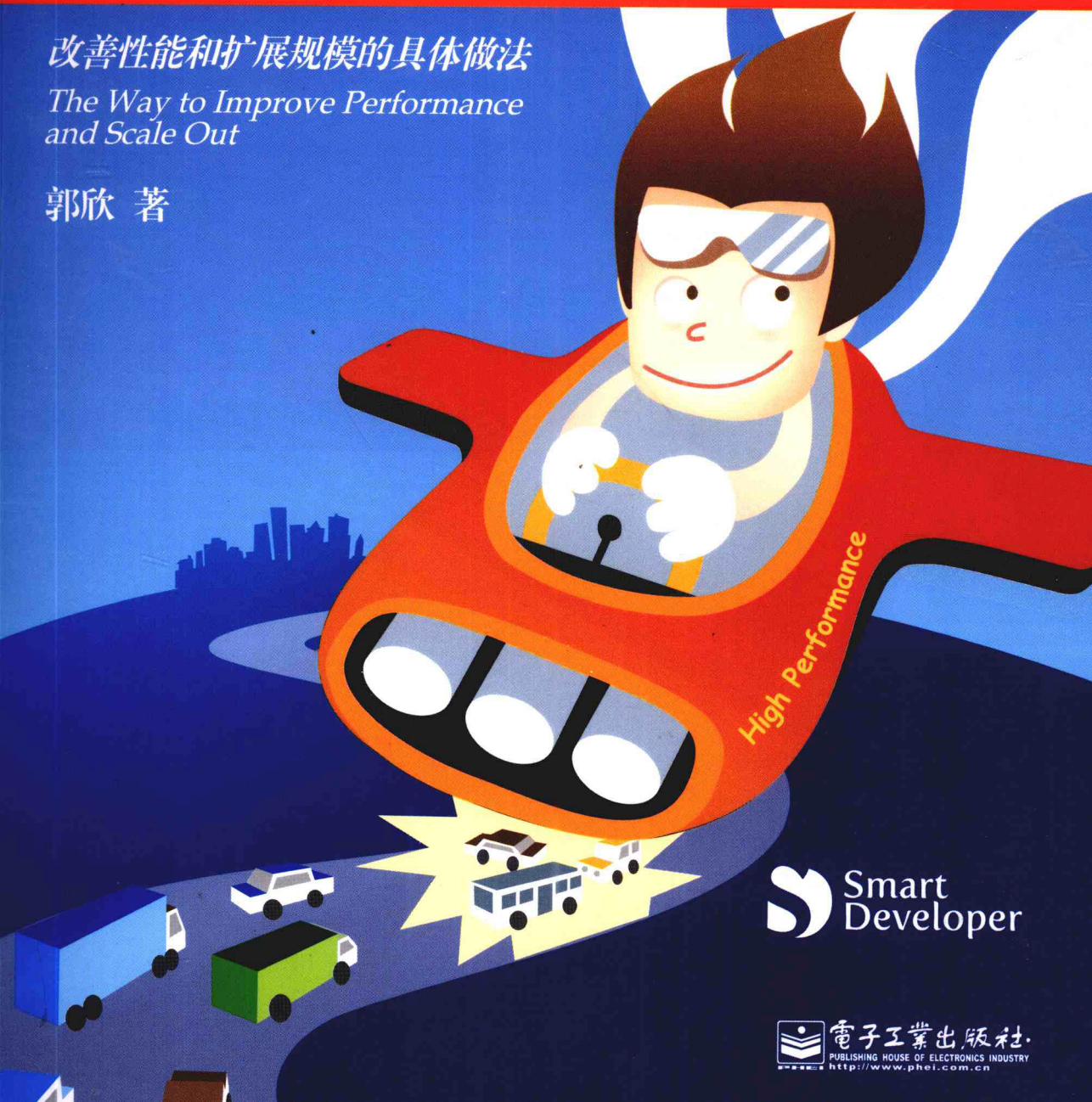
构建高性能Web站点 修订版

Building High Performance Web

改善性能和扩展规模的具体做法

The Way to Improve Performance and Scale Out

郭欣 著



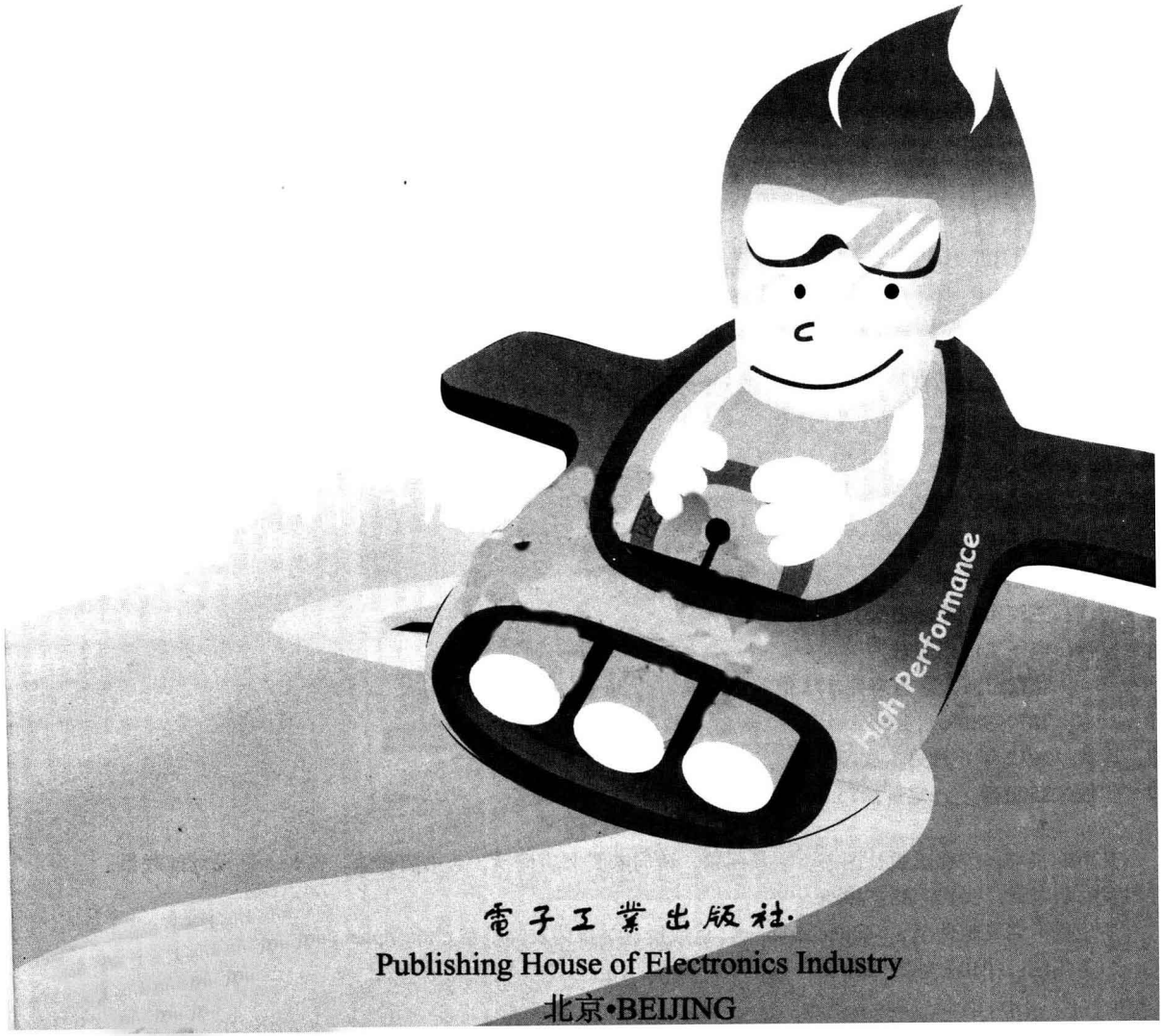
Smart Developer

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

构建高性能Web站点 修订版

Building High Performance Web

郭欣 著



電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书是畅销修订版，围绕如何构建高性能 Web 站点，从多个方面、多个角度进行了全面的阐述，几乎涵盖了 Web 站点性能优化的所有内容，包括数据的网络传输、服务器并发处理能力、动态网页缓存、动态网页静态化、应用层数据缓存、分布式缓存、Web 服务器缓存、反向代理缓存、脚本解释速度、页面组件分离、浏览器本地缓存、浏览器并发请求、文件的分发、数据库 I/O 优化、数据库访问、数据库分布式设计、负载均衡、分布式文件系统、性能监控等。在这些内容中充分抓住本质并结合实践，通过通俗易懂的文字和生动有趣的配图，让读者充分并深入理解高性能架构的真相。同时，本书充分应用跨学科知识和科学分析方法，通过宽泛的视野和独特的角度，将本书的内容展现得更加透彻和富有趣味。

图书在版编目（CIP）数据

构建高性能 Web 站点 / 郭欣著. —修订本. —北京：电子工业出版社，2012.6
ISBN 978-7-121-17093-5

Ⅰ. ①构… Ⅱ. ①郭… Ⅲ. ①网页制作工具－程序设计 Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字(2012)第 102508 号

责任编辑：李　冰
印　　刷：北京东光印刷厂
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：26　字数：608 千字
印　　次：2012 年 6 月第 1 次印刷
印　　数：3500 册　定价：75.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

推荐评价

本书是作者在 Web 系统领域多年工作、实践和探索的结晶。本书涉及 Web 系统优化的各个方面，从浏览器、Cache 到 Web、数据库和分布式文件系统等；穿插了大量的实际测试数据和很多流行开源软件的使用方法与案例；内容丰富，文字生动，对比形象。对网络系统架构师、运维和开发人员来说，这是很好的参考书；对想了解 Web 性能并希望动手实践的人员来说，这是由浅入深的学习书籍。

——章文嵩博士，LVS 作者，Linux 内核作者之一

本书深入分析了常见的高性能 Web 技术的方法和原理，对搭建高性能 Web 站点具备很强的可操作性。

——张松国，腾讯网技术总监

这是一个令人兴奋的领域，这一系列准则和方法在 TopN 的互联网公司中都有大规模的实践和应用，作者在书中进行了详细而量化的论述。如果你正在为日益庞大的应用而手足无措，那么你唯一要做的就是拥有这本书，并且实践它。

——朱鑫，MemcacheDB 作者，新浪网研发中心平台部高级工程师

互联网寄托着我们的梦想，它改变了人们的生活，从社交网站到网络游戏，从搜索引擎到电子商务，成功的秘诀在于如何构建高性能 Web 站点。郭欣在这本书中几乎涵盖了 Web 性能优化的所有内容，并从多个角度进行了全面的阐述，你可以通过其通俗易懂的文字深入理解高性能站点架构的真相，并开拓视野，从而对性能瓶颈对症下药。本书可谓是高性能站点的必读精作。

——沈翔，Google Developer Advocate，加州总部

你很幸运能拿到这本书，更重要的是，你的网站用户也会很幸运。郭欣在这本书里深入而系统地分享了构建高性能网站技术的方方面面。从后台到前台，从网络传输到数据存储，涉及诸多技术原理和实现细节。通俗的语言，亲切的叙述，仿佛作者在耳边轻轻细语，然而又蕴涵着一种技术思想和力量，并且融合了人文思想。

——王速瑜，腾讯 R&D 研发总监，资深技术专家

即便你是征战 Web 架构设计多年的高手，也能够从本书获得系统化分析 Web 架构性能和成本优化的完整思路，从而为工作中遇到的一些棘手的技术问题提供部分参考思路。在本书修订版出版之际，再次推荐本书，希望每一位互联网从业技术人员都能够拥有这么一本难得的精品之作，也希望国内能够出现越来越多的精品技术图书。

——林水洋，新浪微博产品总监，前新浪博客高级系统架构师

作为一名写了二十年代码的有技术情结的人，深知一本好书给人带来的好处，2009 年在得知此书出版后，我第一时间购买并阅读，并且给我的同事每人买了一本，希望他们都看看，得到的反馈都不错，两年过去了，高性能的定义、技术、产品也在改变，希望这本书的修订版能给更多做网站开发的技术人员们带来更多更好的帮助！

——高春辉，卓越网创始人，手机之家创始人，ECSHOP 软件创始人

国内具备实战指导意义的大型网站开发书籍不多，翻译版本质量又参差不齐。究其原因是我们的行业环境和国外不同，分享和开放还没有形成行业文化，多数从业者是半路出家的，他们疲于应付繁重的日常工作，难有时间和精力著书立说。很久以来一直想总结整理多年的工作经验，看了郭欣的这本书之后感觉很多内容都已经借他之笔完成了。推荐！

——李嵩波，德丰杰 VC 合伙人，前新浪 CTO

本书从前端到后台，就 Web 网站的性能优化深入浅出，娓娓道来，对于从事大容量高并发 Web 网站开发和架构设计的同行来说，是一本不可多得的好书。

——魏震，淘米网 CTO

作者在互联网领域实践多年，在搭建高性能 Web 站点方面有丰富的实战经验。书中几乎涵盖 Web 性能优化的所有内容，包括带宽、缓存、数据库、文件系统等各个方面，并从多个角度进行了全面的阐述。本书讲解通俗易懂、内容丰富、生动幽默，是 Web 站点架构师、运维和开发人员的必备佳作。

——肖彬，世纪佳缘 CTO

作者结合自身的经验和实践，从性能、前后端、编码、DB、监控等多个角度，非常系统地阐述了构建高性能 Web 站点的各个方面。难能可贵的另一个地方，是在每个不同的方案提出的同时，都附上了测试方法和数据，体现了作者非常严谨的风格。入门者参考此书，可以少走很多弯路；提高者参考此书，可以得到更多交流和思考的灵感。

——郝冲，北京快网科技有限公司 CTO

要成为一名优秀的 Web 性能架构师，需要有丰富的实战经历和持续做靠谱的总结，并最终沉淀成系统化的实战经验。如果你是一名刚入行的新人，那么恭喜你，你可以从书中获得过来人的大量实战经历，迅速开阔眼界；如果你是一名有多年经验的“老人”，同样恭喜你，你可以从书中得到更系统化的总结，以及更多的启发。

——李琦，腾讯网管平台产品经理

如果你的站点采用开源软件构建，并且你正在为改善性能或准备解决架构扩展性问题而头疼，这本《构建高性能 Web 站点（修订版）》提供了相当全面的指导和参考信息。当然，越早阅读这本书，并能对书中提倡的一些具体建议进行实践的话，收效就会越大。

——冯大辉，丁香园 CTO

作为一个大型网站的架构师，性能是最关注的核心问题。本书由浅入深地讲解了 Web 性能的各方面，对于指导实践具有重要作用。近几年，Web 的趋势朝着更实时化和社交化的方向发展，系统功能比已往更丰富，页面元素和复杂性也在持续增加，对网站的性能提出了更高的要求，希望业内同行能够更好地利用本书阐述的高性能网站构建的方法，迎接新的挑战。

——杨卫华，新浪微博技术总监

看到此书时，颇有相见恨晚的感觉。作者以他广博的知识面，深入浅出的叙述方法，生动形象的类比案例，严谨负责的实践经验，给人很好的启发和借鉴。我真想说，拥有此书的读者实在是太幸福了！

——陈桂新，盛大游戏技术保障中心总监，资深研究员

这本书可以让你从网络传输到缓存、数据库优化、分布式计算等方面去了解构建高性能 Web 站点所需要关注的每一个地方。书中引用了大量数据来说明每一个示例的实际效果，有理有据，具有很强的实践性。对于想在互联网领域大展身手的人来说，这本书值得你拥有！

——潘少宁，“LAMP 人”发起人

荣登 China-pub 计算机类图书 2009 年度畅销榜 TOP 50!

精彩书评

这是第一本今年我只花一周就读完的技术书籍，并不是因为它简单，而是因为它很吸引我，很多地方讲得非常好，例如，服务器的 I/O 模型以及缓存使用的各个环节。非常可贵的是作者的很多测试数据是非常真实的，可以更好、定量地去看问题。

——yangliang1217

很不错的一本书。由浅入深，不管你是初学还是进阶，都能有所收获。强力推荐给所有做网站的人！

——gogobu

很不错的书，讲到了运维的方方面面，很有实际价值！

——spark8103

少见的国内作者写出的好书，极力推荐！

——stillr

这本书还真帮了我大忙，昨天工作中的问题得不到解决，随手翻了一下桌子上的书，看到有关这方面的讲解，还真解决了。很不错，实用的书。

——和平的颜色

目前正在看这本书，此书内容非常不错，包括网站性能优化的方方面面，并且讲得非常细，有理有据，非常适合从事 Web 开发、测试、运维的相关人员学习！

——eweek2@*** ***

修订版前言

从我写出第一个 HTML 网页到现在，已经过去 10 年多的时间了，回顾过去的 Web 开发经历，我曾经尝试过各种不同的技术，与此同时，我和我的团队也犯了很多错误，但我们为此感到自豪。是的，成长是需要不断付出代价的，每次的挫折都会让我更加深刻地看到隐藏在深处的本质，为什么不把这些内容分享出来呢？于是便有了这本书。

10 年来，我们见证了互联网有史以来最快速的发展，商业应用层出不穷，业务逻辑不断变复杂，对用户体验的要求也不断提升，随之而来的是应用技术和开发语言的日新月异，开发者永不停息地学习新技术。同样，在 Web 站点性能方面，我们一直在跟时间赛跑，社交网站和微博客成为大众的主流应用，带来了更加快速、实时的信息传递，更多的站点意识到开放的重要性，数据访问和计算无处不在，每秒数以万次的数据传递和读写正在我们身边进行。

但是，构建 Web 站点的基础技术几乎多年来从未改变，比如诞生于 20 世纪 80 年代的 TCP，如今依旧是网络数据传输的主宰者，而 HTTP 则更与我们息息相关，可是你真的认真学习过它们吗？人们始终在做的事情就是在这些基础技术之上一层一层地封装概念，不断地诞生新的技术。加上商业化产品的市场竞争和炒作，.NET 和 Java 阵营中的概念让我眼花缭乱却又无可奈何。它们已经成为营销用语，有时候过度会让事情变得更加复杂，让开发者迷失方向。

不论你是一名从事 Web 开发的工程师，还是一名关心 Web 性能的架构师，都应该更多地关注各种技术和架构的本质。

从哲学意义上讲，对本质的研究属于形而上学的范畴，但是在自然科学中，我们从来不缺乏对本质的探索，因为只有认识事物的本质才能做出正确的决策，并且真正驾驭它们，这是毫无争议的。

也许你曾经被商家的促销活动所打动。是的，我们往往只看到事物的表面现象，而经济学家却看到了事物的本质，这正是他们的高明之处。技术和架构同样如此，你要明白任何收获都是有代价的，天下没有免费的午餐，很多时候，你完全可以用成本经济学的知识来思考技术的合理性，你甚至可以像经济学家一样思考技术问题。

当然，仅仅理解本质是远远不够的，因为在庞大的架构体系中，涉及太多的部件，而影响

整体性能的因素究竟有哪些呢？你也许会感到扑朔迷离，但你必须知道瓶颈所在，并且能够意识到何时需要优化性能或者扩展规模。与此同时，系统化的分析方法至关重要，中医理论对人体的系统思辨能力体现了先哲们的智慧，在站点性能不尽如人意的时候，我们能否“对症下药”？这与你对整个系统能否全面把握有着密切的关系。

另一方面，绝对与相对、变化与平衡，是永恒的大道，在很多时候，你实际上需要考虑的是如何做出权衡，同时，我们也要铭记变化的道理，系统瓶颈不是一成不变的，久经考验的架构师深知这一点。

道可道，非常道。要将所有的架构之道讲出来实属不易，架构就像艺术品一样，往往无法完全复制，但是独立的技术以及分析的思路是可以学习的，作为优秀的开发者或者架构师，心中的架构才是最有价值的。

如果你希望寻找心中的架构，那么，从本书的第 1 章“绪论”开始吧！

读者群

如果你希望学习如何创建一个 Web 站点，那么这本书可能并不适合你，但是当你对站点的性能开始担忧时，欢迎你的归来。

这本书适合以下读者：

- 编写 Web 应用程序、关心站点性能，并且希望自己做得更加出色的开发人员
- 关心性能和可用性的 Web 架构师
- 希望构建高性能 Web 站点的技术负责人
- 实施 Web 站点性能优化或者规模扩展的运维人员
- 与 Web 性能有关的测试人员

的确，整个技术团队的所有成员都适合阅读这本书。另外，高校学生以及个人网站站长也可以阅读，笔者希望本书可以帮助他们开拓视野。

如何阅读

本书涉及大量的软件和工具，由于篇幅有限，书中并没有对它们的具体使用方法展开详细介绍，你可以通过 Google 查找相关的在线手册进行自学，同时，在本书的参考文献列表中，也会提供一些相关链接。

本书不想在阐释体系上束缚读者的天分，所有内容的安排更像是引导你身处 Web 站点的各个角落，与影响性能的各种因素自由碰撞，并且一步步地思考和分析问题。所以，你也

许不会直接找到一个高性能 Web 站点的完整解决方案，但是，当你认真地依次阅读完本书的所有章节后，你也许会找到你更想要的东西，并从中获益，我衷心希望如此。

如果没有什么特殊的原因，还是建议你能够按照章节顺序，心平气和地通读一遍，因为在内容组织上，本书有太多的连贯性设计，我担心随机阅读会让你的收益大打折扣。

创作过程

说到这次写书的过程，我同样感到很有意义，各种全新的尝试让创作过程充满乐趣，它们同样值得分享。

不同于传统使用 Word 编写内容，我使用了快捷的 Google 在线文档，并使用在线表格保存测试数据，它们支持出色的版本管理，并且提供快速的分享和协作功能。当然，最激动人心的莫过于我可以在任何地点通过浏览器继续我的写作过程，甚至当灵感突如其来时打开 Google G1 手机便可以写上两句。

对于几十万字的篇幅，一气呵成绝对是不可能的，多次迭代必然贯穿整个写作过程，从有灵感到写提纲，再从框架到最终的文字，虽然没有完善的过程管理，但是我时刻都能感觉到敏捷的火花。

为了尽早地获得读者的反馈，我考虑尽早“部署”，于是选择了讨论组和邮件列表的方式，在 Google Group 上创建了读者讨论组，上传了一些试读章节，收集到了大量的修改意见和想法，这些都是我所需要的，同时也给我带来了鼓励和支持。

整个过程还有很多花絮，这里就不一一介绍了。创作的过程是艰辛的，需要作者的坚持和毅力，虽然创作本身没有捷径，但是我们可以让创作过程更加充满乐趣，让作者和读者更加近距离地接触。

关于修订版

本书第一版于 2009 年 8 月出版，当年即获得 China-pub 2009 年度畅销榜 TOP50，在第一版发行期间经历 6 次印刷。时至 2012 年，令人欣喜的是，读者朋友们对本书的热情依旧不减，他们一直以来的支持和反馈让这本书的生命力得以延续，也让这本书有幸获得更多的新读者。为了服务更多的读者朋友们，我将读者们的反馈和评价进行整理，并与一些热心读者进行反复探讨，最终对书中内容进行了勘误修订。于是就有了你手中的这本修订版图书。已经购买过本书第 1 版的读者朋友们，此次不建议再次购买。

虽然没有增加太多新的内容或许令人遗憾，但另一方面，我始终认为书中的内容并不过时，

至少大部分内容在 2012 年甚至未来 10 年仍有价值，因为这些内容多为设计高性能 Web 架构的本质，而不是应用层的各种流行技术。希望这本书可以帮助更多的读者朋友们找到心中的架构，我对此充满信心！

联系作者

在本书出版后的任何时间，我都欢迎你提出自己的意见或思想，你可以通过微博联系到我，我的微博地址是：http://weibo.com/guoxin

致谢

感谢我的父母，他们在我读初中的时候送给我第一台电脑（Cyrix 1.66G 的兼容机），让我走进了计算机的世界，并且给予我非常多的支持和鼓励，让我毫无顾虑地追逐梦想。

感谢我的妻子 Cris，她不仅为我创作了本书的封面，而且在整个写作过程中毫无抱怨地陪伴我，给予我无尽的支持和灵感。

感谢电子工业出版社的策划编辑李冰，她严谨认真的工作态度以及作为出版方的开放态度让我深感敬佩。

感谢为本书撰写推荐和评价的朋友们，他们是王速瑜、章文嵩、张松国、沈翔、朱鑫、肖彬、郝冲、李琦、冯大辉、杨卫华、李嵩波、陈桂新、高春辉、林水洋、魏震、潘少宁。他们在繁忙的工作中抽出时间，阅读了本书的样稿并写下了推荐和评价。特别值得一提的是，章文嵩博士对书中一些内容建议让我受益匪浅。

感谢对本书提出宝贵修改意见的朋友们，他们是蒋琦、丁吉亮、汤文亮、刘健、朱李、周伟强。

感谢在读者讨论组中所有积极阅读、并提出意见的成员。

CONTENTS

目 录

第 1 章 绪论 …… 1

1.1 等待的真相 …… 1
1.2 瓶颈在哪里 …… 2
1.3 增加带宽 …… 3
1.4 减少网页中的 HTTP 请求 …… 4
1.5 加快服务器脚本计算速度 …… 4
1.6 使用动态内容缓存 …… 5
1.7 使用数据缓存 …… 5
1.8 将动态内容静态化 …… 6
1.9 更换 Web 服务器软件 …… 6
1.10 页面组件分离 …… 7
1.11 合理部署服务器 …… 7
1.12 使用负载均衡 …… 8
1.13 优化数据库 …… 8
1.14 考虑可扩展性 …… 9
1.15 减少视觉等待 …… 10

第 2 章 数据的网络传输 …… 11

2.1 分层网络模型 …… 11
2.2 带宽 …… 22
2.3 响应时间 …… 28
2.4 互联互通 …… 33

第 3 章 服务器并发处理能力 …… 36

3.1 吞吐率 …… 36
3.2 CPU 并发计算 …… 50
3.3 系统调用 …… 61
3.4 内存分配 …… 64

3.5 持久连接 …… 66
3.6 I/O 模型 …… 69
3.7 服务器并发策略 …… 82

第 4 章 动态内容缓存 …… 97

4.1 重复的开销 …… 97
4.2 缓存与速度 …… 99
4.3 页面缓存 …… 99
4.4 局部无缓存 …… 113
4.5 静态化内容 …… 113

第 5 章 动态脚本加速 …… 122

5.1 opcode 缓存 …… 122
5.2 解释器扩展模块 …… 133
5.3 脚本跟踪与分析 …… 134

第 6 章 浏览器缓存 …… 144

6.1 别忘了浏览器 …… 144
6.2 缓存协商 …… 148
6.3 彻底消灭请求 …… 161

第 7 章 Web 服务器缓存 …… 168

7.1 URL 映射 …… 168
7.2 缓存响应内容 …… 169
7.3 缓存文件描述符 …… 176

第 8 章 反向代理缓存 …… 179

8.1 传统代理 …… 179
8.2 何为反向 …… 180
8.3 在反向代理上创建缓存 …… 181
8.4 小心穿过代理 …… 202
8.5 流量分配 …… 204

第 9 章 Web 组件分离 …… 206

9.1 备受争议的分离 …… 206
9.2 因材施教 …… 207

9.3 拥有不同的域名 ····· 208
9.4 浏览器并发数 ····· 211
9.5 发挥各自的潜力 ····· 213

第 10 章 分布式缓存 ····· 221

10.1 数据库的前端缓存区 ····· 221
10.2 使用 memcached ····· 222
10.3 读操作缓存 ····· 226
10.4 写操作缓存 ····· 230
10.5 监控状态 ····· 233
10.6 缓存扩展 ····· 235

第 11 章 数据库性能优化 ····· 239

11.1 友好的状态报告 ····· 240
11.2 正确使用索引 ····· 242
11.3 锁定与等待 ····· 256
11.4 事务性表的性能 ····· 264
11.5 使用查询缓存 ····· 265
11.6 临时表 ····· 267
11.7 线程池 ····· 267
11.8 反范式化设计 ····· 268
11.9 放弃关系型数据库 ····· 270

第 12 章 Web 负载均衡 ····· 272

12.1 一些思考 ····· 272
12.2 HTTP 重定向 ····· 275
12.3 DNS 负载均衡 ····· 284
12.4 反向代理负载均衡 ····· 292
12.5 IP 负载均衡 ····· 305
12.6 直接路由 ····· 317
12.7 IP 隧道 ····· 325
12.8 考虑可用性 ····· 325

第 13 章 共享文件系统 ····· 328

13.1 网络共享 ····· 328
13.2 NFS ····· 330

13.3 局限性 ····· 335

第 14 章 内容分发和同步 ····· 337

14.1 复制 ····· 337
14.2 SSH ····· 338
14.3 WebDAV ····· 342
14.4 rsync ····· 342
14.5 Hash tree ····· 344
14.6 分发还是同步 ····· 345
14.7 反向代理 ····· 346

第 15 章 分布式文件系统 ····· 348

15.1 文件系统 ····· 348
15.2 存储节点和追踪器 ····· 350
15.3 MogileFS ····· 352

第 16 章 数据库扩展 ····· 361

16.1 复制和分离 ····· 361
16.2 垂直分区 ····· 364
16.3 水平分区 ····· 366

第 17 章 分布式计算 ····· 372

17.1 异步计算 ····· 372
17.2 并行计算 ····· 377

第 18 章 性能监控 ····· 382

18.1 实时监控 ····· 382
18.2 监控代理 ····· 384
18.3 系统监控 ····· 386
18.4 服务监控 ····· 389

参考文献 ····· 392

索引 ····· 394

第 1 章
绪　论

一般而言，人们评估一个 Web 站点的性能如何，通常先置身于用户的角度，来访问该站点的一系列页面，体验等待时间。

当用户输入页面地址后，浏览器获得了用户希望访问该地址的意图，便向站点服务器发起一系列的请求。注意，这些请求不仅包括对页面的请求，还包括对页面中许许多多组件的请求，比如图片、层叠样式表（CSS）、脚本（JavaScript）、内嵌页面（iframe）等。接下来的一段时间，浏览器等待服务器的响应以及返回的数据。待浏览器获得所有的返回数据后，经过本地的计算和渲染，最终一幅完整的页面才呈现于用户的眼前。

1.1　等待的真相

整个过程听起来好像并不复杂，也许你从来都没有考虑过在这段等待的时间里世界都发生了什么变化，也许你早已习惯了利用这段时间东张西望或者品尝零食，或者你根本没有来得及意识到这点，新的网页就已经闪亮登场。恭喜你，你很幸运！但是在这个世界上，幸运儿永远只占少数，大多数人的大脑处理速度已经让他们明显感觉到这段等待时间漫长无比，久经考验的他们可以随时身手敏捷地打开多个浏览器窗口与时间赛跑，并为此筋疲力尽。

另一方面，对于站点经营者来说，让用户等待的时间过长，也许会造成毁灭性的后果。笔者见过很多人为了享用某家特色小吃而在餐馆门口乐此不疲地排着长队，但没有听说有多少用户执著地等待着一个速度缓慢的站点而不去尝试别的站点。

在这段等待的时间里，到底发生了什么？事实上这并不简单，大概经历了以下几部分时间：

- 数据在网络上传输的时间。
- 站点服务器处理请求并生成回应数据的时间。
- 浏览器本地计算和渲染的时间。

数据在网络上传输的时间总的来说包括两部分，即浏览器端主机发出的请求数据经过网络到达服务器的时间，以及服务器的回应数据经过网络回到浏览器端主机的时间。这两部分时间都可以视为某一大小的数据从某主机开始发送，直到另一端主机全部接收所消耗的总

时间，我们称它为响应时间，它的决定因素主要包括发送的数据量和网络带宽。数据量容易计算，但是究竟什么是带宽呢？我们将在后续章节中详细介绍带宽的本质。

站点服务器处理请求并生成回应数据的时间主要消耗在服务器端，包括非常多的环节，我们一般用另一个指标来衡量这部分时间，即每秒处理请求数（也称吞吐率）。注意这里的吞吐率不是指单位时间内处理的数据量，而是请求数。影响服务器吞吐率的因素非常多，比如服务器的并发策略、I/O 模型、I/O 性能、CPU 核数等，当然也包括应用程序本身的逻辑复杂度等。这些将在后续章节中详细介绍。

浏览器本地计算和渲染的时间自然消耗在浏览器端，它依赖的因素包括浏览器采用的并发策略、样式渲染方式、脚本解释器的性能、页面大小、页面组件的数量、页面组件缓存状况、页面组件域名分布以及域名 DNS 解析等，并且其中一些因素随着各厂商浏览器版本的不同而略有变化。这部分内容在后续章节中也会适当提到。

可见，一个页面包含了若干个请求，每个请求都或多或少地涉及以上这些过程，假如有一处关键环节稍加拖延，整体的速度便可想而知。

现在，如果有用户向你抱怨在打开站点首页的时候等待了很久，你知道究竟慢在哪里了吗？

1.2 瓶颈在哪里

相信你一定知道赤壁之战，这是中国历史上一场著名的以少胜多的战役，东吴的任务是击退曹操的进攻，要完成这项任务，可谓“万事俱备，只欠东风”，这时东风便是决胜的瓶颈，所以很多系统论研究专家将其称为“东风效应”，也就是社会心理学里讲的“瓶颈效应”。

之所以称它为瓶颈，是因为尽管东吴做了很多战前准备，包括蒋干中计导致曹操错杀蔡瑁和张允、诸葛亮草船借箭、东吴苦练水军等，但是仅靠这些仍无法获得最终胜利，还需要最后的东南风才能一锤定音，完成火烧曹军战船的计划。不过之前的准备工作都是胜利的子因素，而东南风这个关键因素最终和其他子因素一起相互作用，将整个战斗的杀伤力无限放大。

曹操运气不好，遇上东南风，倒了大霉，曹军战船一片火海，这时候东吴需要派出勇猛的陆军部队登岸攻下曹营，可是东吴向来精通水战，几乎没有强大的陆战部队，只有老将黄盖，这如何与曹操的精英骑兵抗衡呢？这个时候决胜的关键因素变成了刘备的盟军支援，五虎上将各个威猛无比，身怀必杀绝技，此时正是上岸一显身手的好机会，他们不费吹灰之力就将曹军打得落花流水，试想如果没有刘备的支援，赤壁之战胜败可能就扑朔迷离了。