[德] Marcel Weiher 著
李俊阳 马超 程伟 孙莹 译

Pearson
Broadview
www.broadview.com.cn

iOS和macOS性能优化

Cocoa、Cocoa Touch、Objective-C和Swift

iOS and macOS Performance Tuning

Cocoa, Cocoa Touch, Objective-C, and Swift

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

[德] Marcel Weiher 著
李俊阳 马超 程伟 孙莹 译

iOS和macOS性能优化

Cocoa、Cocoa Touch、Objective-C和Swift

iOS and macOS Performance Tuning
Cocoa, Cocoa Touch, Objective-C, and Swift

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

内 容 简 介

随着应用功能的日益增加，性能问题也逐渐浮出水面，进入我们的视野之中。本书作者 Marcel Weiher 在性能调优领域有着丰富的经验，在他的带领下，你将会了解如何提升 CPU、内存、I/O、图像、Swift 等方面的性能，如何在开发过程中定位到性能的瓶颈和问题，他同时还介绍了性能优化的编程技巧和最佳实践，从而帮助你写出更高效、更快速的代码。此外，你还将学习到定位性能问题的工具使用方法以及最佳实践，并跟随示例来学习性能优化。

本书适合寻求进阶及所有对性能优化感兴趣的 iOS 开发人员阅读。

版权贸易合同登记号　图字：01-2017-3551

图书在版编目（CIP）数据

iOS 和 macOS 性能优化：Cocoa、Cocoa Touch、Objective-C 和 Swift /（德）马塞尔 •韦伊尔（Marcel Weiher）著；李俊阳等译. —北京：电子工业出版社，2018.7

书名原文：iOS and macOS Performance Tuning: Cocoa, Cocoa Touch, Objective-C, and Swift

ISBN 978-7-121-33814-4

Ⅰ. ①i… Ⅱ. ①马… ②李… Ⅲ. ①移动终端－应用程序－程序设计 Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字（2018）第 042956 号

策划编辑：刘恩惠
责任编辑：张春雨
印　　刷：北京天宇星印刷厂
装　　订：北京天宇星印刷厂
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：23　字数：442 千字
版　　次：2018 年 7 月第 1 版
印　　次：2018 年 7 月第 1 次印刷
定　　价：99.00 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）88254888，88258888。

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。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

译者序

我们在开发应用的时候，最关心的都是些什么呢？首要的自然就是功能需求了，此外还有那些炫酷的用户界面和转场动画。当我们进入书店，或者在网上搜索时，映入眼帘的普遍都是开发入门、功能需求、动画特效等诸如此类的书籍和文章。那么，我们何时才会关注那些所谓的“非功能性需求”呢？

在我们翻译的上一本书《iOS 应用安全权威指南》中，我们很高兴地看到了 iOS 开发中关于“安全性”需求的话题。在本书中，我们将继续关注“性能和可靠性”这一非功能性需求。

提到性能，我们通常想到的是后台的服务器性能和网络带宽。在出现性能问题的时候，我们会想方设法地提升后台服务器的性能，增加 CPU 内核、增加内存、增加网络带宽……但是有没有想过，有时候应用本身的性能也会阻碍用户数量的增长，甚至严重影响用户体验。试想，如果用户在使用应用的时候，出现了严重的卡顿，这种情况对于用户而言往往是无法忍受的。这对很多公司而言，致命性不言而喻。更严重的是，应用一旦出现了性能问题，我们是没办法通过增加 CPU 内核、内存等方法来进行提升的。因此，我们需要对应用进行性能调优。

但是，目前市面上关于 iOS 性能优化的文章和书籍实在是少之又少，我们通常会看到人们用 Instruments 的 Allocations、Leaks 等工具来检查性能问题，但是它们的具体用法是怎么样的呢？我们要怎样做才能找到应用真实的性能瓶颈呢？有哪些问题是通过编写代码本身就可以规避的呢？这些，本书都能够告诉你。

我们相信，每位软件开发者都有一种精益求精、追求卓越的心态和想法，在我们完成了庞杂的功能需求之后，自然就会开始关注如何提升用户体验，其中就包括性能优化。无论你的应用规模如何，我们都强烈建议大家阅读本书，掌握一些基本的技巧，只要在开发过程中规避这些陷阱和漏洞，那么基本上我们的应用就能够满足性能的要求，在绝大多数时间，使你完全没必要担忧性能的相关问题。

我们都知道，Xcode 拥有一个简单的性能检测窗口 Debug Navigator，其中分别展示了 CPU、内存、硬盘、网络的使用量等。类似地，本书包含如下四部分内容：CPU、内

存、I/O，以及图像处理和响应速度。每部分内容都同时包括了基本的理论知识、测量的工具和度量标准、常见的问题和处理方法，以及真实的案例演示。内容循序渐进，由浅至深，无论你是急于寻找性能问题的解决方案，还是想要系统化地学习性能调优的相关知识，都能够依据这个完整的架构体系寻找到想要的内容。

在本书翻译过程中，由于书中的内容对于我们而言也比较新颖，在百忙之余，我们也一一根据书中的案例和步骤先行学习、试验和体会，以期能够用更为准确的语言和文字，将书中的思想完整地分享给各位读者朋友。在此也特别感谢参与校对的 SwiftGG 翻译组的小伙伴们，对我们在翻译过程中出现的错误进行了勘正。同时也感谢电子工业出版社的编辑、审稿专家，他们认真负责、高效细心地进行了编辑和校对，也给我们提出了不少好的建议。当然，由于本书用到了大量的专业术语，在中文世界中找不到相应的描述，因此我们也斗胆“创造”了很多新词，如果出现了缺漏、不准确、不到位的情况，还请各位读者批评指正。

最后，再次感谢你关注这本讲解性能优化的书籍，我们相信你一定能从这本书中学习到各种有用的知识，向进阶的 iOS 开发者更进一步！

李俊阳

2018 年 3 月 3 日

前言

性能是软件极其重要的特性之一。若没有世界一流的性能，软件也就称不上是世界级的。长期以来，硬件的改善意味着担心软件的性能似乎是浪费时间的，但随着摩尔定律不再自然而然地提供显著的自动性能改进，性能优化也逐步回到了计算机科学和工程的前沿。

此外，尽管底层硬件的性能已经提升了很多倍，但终端用户对性能提升的感知似乎并不明显。比尔·盖茨认为“软件的速度每 18 个月就会变慢一半”，同样在 *A Plea for Lean Software*（《为软件瘦身请命》）一文中提出的维尔特定律还认为，“软件变慢的速度永远快过硬件变快的速度”[1]。

iPad 面世之初，行业资深人士被其流体式的布局界面所惊艳，但同时不得不接受一个只配备了 1GHz 的 CPU，这是让人感到遗憾的一点。不过，那时的 iPad 比我的 Apple II 快了 1000 多倍，比大屏的 NeXT Cube 也要快 40 倍。如果真的有什么值得诧异的，那就是在使用 GPU 处理屏幕渲染的时候，它居然没怎么变快。

本书将尝试在 Objective-C、Cocoa 和 Cocoa Touch 的背景下深入了解这些发展的根本原因，并尝试提供技术，帮助我们充分利用计算机惊人的原始力量——那些易于肆意挥霍的力量。我会试图告知何时挥霍计算机的性能是恰当的，以及何时需要对性能引起高度重视。程序员的注意力也是一种稀缺资源，但却经常浪费在尝试优化无关紧要的部分程序上。

主题将涵盖延迟与带宽，处理事项成本损耗（开销）与实际完成工作的对比，其具有普遍性，且硬件和软件栈的表现形式随级别不同而不同。

你可能注意到任何单一的操作时间取决于机器的速度，而机器总是足够快的，因此得出关键方程项目数×损耗。大多数优化是减少公式的一部分或两部分，通常我们要先尝

---

1 Niklaus Wirth, A Plea for Lean Software(Los Alamitos,CA:IEEE Computer Society Press,1995),pp.64-68. http://dx.doi.org/10.1109/2.348001。

试将其分解。

降低成本损耗的一种常用方法：认识到损耗实际上是由损耗 1 和损耗 2 两个独立的成本组成的，并且两者之中有且仅有一个需要应用于所有项目——项目数×（损耗 1+损耗 2）→损耗 1+项目数×损耗 2。我称之为基本优化方程，大部分优化技术均属于这一类，它也是构成我们每天处理的大多数硬件/软件栈的基础。

本书有一个非常规则的目录结构，将依次讨论性能的 4 个基本主题领域：

1. CPU 的性能。
2. 内存。
3. I/O。
4. 图像和响应速度。

尽管已经努力保证每个主题领域的独立性，但是逻辑之间千丝万缕，因此对基础的主题有一定了解有助于对后续主题的理解。

上述 4 个主题分别又被划分成 4 个特定的兴趣领域，如下。

1. 原理。
2. 测量和工具。
3. 陷阱和优化技巧。
4. 实战演示应用技术。

再次强调，我们将遵循这样一个逻辑结构：在进行实际的性能优化技术之前，你需要了解一些理论知识并知道如何测量，同样地，如果基本熟悉前面几个话题，应该也能够深入感兴趣的特定领域。

本书采用这种结构划分成了 4×4 =16 章，加上内存和 I/O 之间穿插的特别章节 Swift，总计 17 章。Swift 在整本书中被广泛使用，其独特的性能特点值得我们新开一章来讨论。

对我而言，软件性能是一种激情和呼唤，贯穿了我的整个职业生涯。关于性能，我深有体会，性能无法自动优化，我们也无法在最后时刻弃它而去。另一方面，不要过分担心性能，才能集中精力在真正需要的性能工作上。这并非自相矛盾，设定一个合理的基础性能水平，通常情况下软件都是能够达到的，这样就免去了大部分时间都在对性能担忧的困扰。

简而言之，本书是关于如何出色完美地提升软件执行效率的一本书。

关于作者

Marcel Weiher 是一名软件工程师和研究人员，拥有超过 25 年的 Cocoa 相关技术经验。Marcel 致力于性能优化相关的工作，曾在英国广播公司优化过世界上极繁忙的网络的性能，解决了当下机器上难以忍受的积压问题，而其在 Apple 的 Mac OS X 性能团队任职时，也帮助过其他工程师提高代码性能。

除了帮助知名公司和初创企业开发屡获殊荣的软件以及组建开发团队，Marcel 还教授课程，维护博客，出席会议演讲，为开源项目做贡献，并发明了如高阶消息传递等新技术。自 1987 年开始，Marcel 着手 Objective-C 的实现，开始从事与编程语言相关的工作，最终实现了 Objective-Smalltalk 架构研究语言。Marcel 目前在柏林的微软公司担任首席软件工程师，并拥有自己的软件和咨询公司 metaobject ltd。

目录