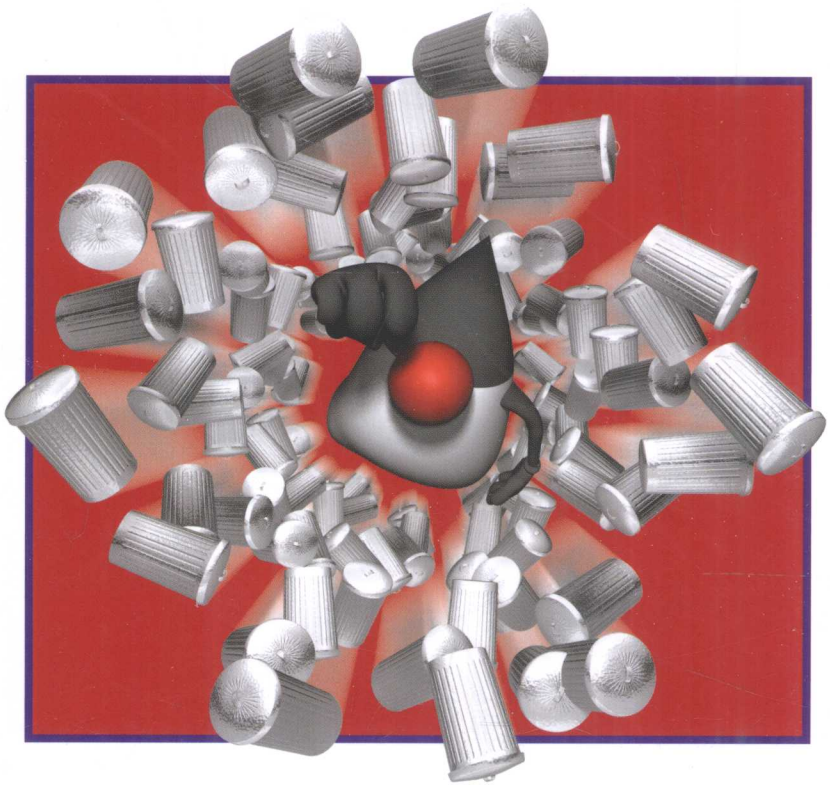
Broadview®
www.broadview.com.cn

Pearson

Java Performance Companion

# Java性能调优指南

[美] Charlie Hunt Monica Beckwith Poonam Parhar Bengt Rutisson 著
李源 季虎 译



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

# Java性能调优指南

# Java Performance Companion

[美] Charlie Hunt
Monica Beckwith
Poonam Parhar
Bengt Rutisson 著
李源 季虎 译

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

## 内 容 简 介

本书主要展示了如何在当今先进的多核硬件和复杂的操作系统环境下，系统且主动地提高 Java 性能。本书对 Charlie Hunt 和 Binu John 的经典图书 *Java™ Performance* 进行延伸，提供了两个前所未有的、强大的 Java 平台创新细节：Garbage First（G1）垃圾收集器和 HotSpot 虚拟机服务代理。

阅读本书，你就可以在任何情况下从 JDK8 或 9 中发挥 Java 的最大性能。

Authorized translation from the English language edition, entitled Java Performance Companion, 978-0133796827, by Charlie Hunt, Monica Beckwith, Poonam Parhar, Bengt Rutisson, published by Pearson Education, Inc., publishing as Addison-Wesley Professional, Copyright © 2016 Pearson Education, Inc.
All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.
CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and PUBLISHING HOUSE OF ELECTRONICS INDUSTRY Copyright © 2017.

本书简体中文版专有出版权由 Pearson Education 培生教育出版亚洲有限公司授予电子工业出版社。未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

本书简体中文版贴有 Pearson Education 培生教育出版集团激光防伪标签，无标签者不得销售。

版权贸易合同登记号　图字：01-2016-6028

**图书在版编目（CIP）数据**

Java 性能调优指南 /（美）查理·亨特（Charlie Hunt）等著；李源，季虎译. —北京：电子工业出版社，2017.4
书名原文：Java Performance Companion
ISBN 978-7-121-30981-6

Ⅰ. ①J… Ⅱ. ①查… ②李… ③季… Ⅲ. ①JAVA 语言－程序设计－指南 Ⅳ. ①TP312.8-62

中国版本图书馆 CIP 数据核字(2017)第 032589 号

策划编辑：张春雨
责任编辑：徐津平
印　　刷：北京京科印刷有限公司
装　　订：北京京科印刷有限公司
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：11.75　字数：221.25 千字
版　　次：2017 年 4 月第 1 版
印　　次：2017 年 4 月第 1 次印刷
定　　价：69.00 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。

# 推荐序 1

若干年前，Charlie Hunt 出过一本 Java 之父 Gosling 欣然作序推荐的 *Java™ Performance*，我现在还记得推荐语中那些星光熠熠的名字——Azul 的 R 大莫枢，阿里花名叫毕玄的林昊，《深入理解 Java 虚拟机》的作者周志明等等。

若干年后，Charlie Hunt 又出了它的姊妹篇 *Java Performance Companion*（中文书名为《Java 性能调优指南》）。这次轮到我被春雨邀来作序，深感荣幸。

与前一本大而全的 Java 调优圣经相比，这本薄薄的姊妹篇只深入讲解了两件事情：一是很多人知道但并不太精通的 G1 垃圾收集算法，另一个是 JDK 自带的，强大但鲜有人知，面试里问谁谁死的工具——HotSpot Serviceability Agent。为什么要单挑这两点来出一本书呢，大概是作者觉得一线的 Java 性能调优工程师以及负责 Trouble Shooting 的大神，都需要掌握这些细节，而目前已有的书籍和文章，又达不到作者期望的深度吧。

与 Java 程序员相依相伴了很多年的 CMS 垃圾收集算法，随着现在 Java 应用的内存越占越大的情况，在暂停时间上有点力不从心了。但压测的经历告诉我们，不是简单地把算法改成 G1 就可以。从十年前 G1 理论的提出，到在 JDK8 里真正可用，JVM 的工程师们反复打磨了漫长的时光，可见其实现是相当复杂的。如果真的在生产环境对延时敏感严苛的应用上使用它，你需要了解更多的实现细节，更多的优化技巧，才能有足够的信心去把控未来可能出现的情况——本书正好提供了这方面的内容。

而另一个 SA 就更好玩了。大部分负责火线处理状况的工程师都会用 Thread Dump、

Heap Dump、BTrace 来获取 JVM 和应用的状态，偶然间，我也在 R 大指导下，用过 gdb 来查找线程死锁的原因。其实，JDK 还为这些苦恼的救火员们，准备了另一份强大的分析工具——SA 及其可视化的工具 HSDB。可惜搜遍互联网，也就只有 R 大写过一篇介绍，还有寒泉子和占小狼写过使用的案例。希望凭这本书，能多制造出几个在自己公司里充当定海神针的大神来。

如果我没有加入唯品会的基础架构部，也不会每天睁眼就面对各种服务调用超时、GC 暂停、并发锁或其他不可描述的奇怪状况，也不会获得如此多的锻炼和感悟。我有感于此，希望有更多的 Java 程序员，在业务代码开发之外，永远对性能优化，对优雅代码，保持自己的欲望；对各种意外情况，保持好奇心而不是恐惧感。最后做个广告（嗯，侠少答应的写序的福利），我的所有这些 Java 工作的日常与思考，都会写在个人微信号“春天的旁边”里，欢迎来访。

肖桦（江南白衣）
唯品会高级架构师

此为试读，需要完整PDF请访问：www.ertongbook.com

# 推荐序 2

侠少邀请我为本书写序已经挺久了，直到 2017 年春节假期之后才开始动笔，再拖都有点过意不去了，在此还是要感谢下侠少及博文视点对我的信任和包容。

本序主要分为两部分，一部分是对本书的内容做一个大致的介绍，另外一部分是简述下本人走上 JVM 这条路的心路历程。

收到本书样稿之后满怀期待地看了起来，当我第一次看到目录时还是震惊了一下，因为本书的重点在 G1 GC 和 SA，两者实际上并没有多大关系，将它们俩合在一起写成此书，本人觉得挺新鲜的，而且本书有且仅有这两者，不过再想想这本书的名字，其实也合情合理。

G1 GC 是 HotSpot 重点发展的 GC 算法，和其他 GC 算法最大的区别是弱化了分代的概念，引入了分区的思想，从 JDK8 开始算比较稳定了，官方也推荐大家使用 G1 GC 来替代 CMS GC，是未来大家首选的 GC 算法。尽管 G1 希望能做到自适应，但是毕竟每个系统的特性不一样，最终可能仍然省不了参数调优的活，本书从实现的角度来阐述了这个算法的各个细节，另外值得一提的是，书中提到的细节完全可以映射到代码中来一起看，它将带您一步步揭开 G1 GC 的神秘面纱。另外我曾有幸和本书的作者之一——Monica 有过一面之缘，她谈及对 G1 的理解还是让我挺惊讶的。

SA 作为 JDK 自带的一款优秀的分析工具，本身就是用 Java 实现的，所以它可以让 JVM 里的数据结构直接以 Java 的形式暴露在您面前，比如说我们方法调用会创建一个栈帧，那

这个栈帧在 JVM 里是怎样的结构？又比如我们想要获取运行时加载的类的字节码，都可以直接通过 API 获取到。另外本书还介绍了一个重要的图形化工具 HSDB，它是基于 SA 实现的，可以让您更直观地看到对象在 VM 里的数据结构以及对象依赖和被依赖的关系等，这个工具在我们在平时查问题的时候提供了极大的便利，可以这么说吧，如果您熟悉 SA 的数据结构，那您对 JVM 的数据结构也算清晰了解了，它为我们 Java 开发者打开了学习 JVM 的另外一扇大门。

接下来说说我的经历吧，其实我算不上 JVM 领域的前辈，更谈不上 JVM 领域的精英，说是 JVM 领域的一个新手一点不为过。阿里是我 2010 年大学毕业后的第一份工作（目前我还一直在支付宝），它是一个 Java 王国，对 Java 重度依赖，然而我在进阿里之前对 Java 其实并没有进行太深入的学习，可以说了解的非常少，主要技能集中在 Flex/Flash/ActionScript 领域，现在没有进入前端界其实还是有点小遗憾的，因为一直都喜欢所见即所得的这种感觉。进入支付宝之后我从事的是 Java 框架研发的偏基础性质的工作，一干就是三四年，非常感谢这段宝贵的经历，让我从前端过渡到了后端，同时让我有足够多的机会碰到各种中间件周边技术的疑难杂症，并不断尝试去解决他们。后面我们框架做大版本升级改造，迁移到标准的 OSGI 容器 equniox 上来，于是碰到了大量的类加载问题，譬如 ClassNotFoundException、LinkageError 等这些问题，如果对类加载机制不是很熟悉还是比较难解决的，我们当时仅仅对 JDK 层面的类加载机制熟悉，深入到 JVM 里的实现就完全是个黑盒了，类似 LinkageError 这种问题在您看了 JVM 里的实现之后才可能有更深的理解，就这样一个偶然的机会开始下载 OpenJDK 代码，并研究起 JVM 来，不过说来也奇怪，在此之后公司碰到的 JVM 的问题感觉也越来越多了，比如内存溢出、频繁 GC、系统卡死等。我在 2014 年年底的时候正式加入到了阿里 JVM 团队，开始了真正的 JVM 探索之旅。

在探索 JVM 的过程中，我基本是问题驱动型的，因为我们在平时工作过程中也很容易碰到很多 JVM 相关的问题，通过一个个攻破这些问题从而积累了不少实践经验，对实现原理也慢慢地熟悉起来，针对一些比较有代表性的文章，我常常会将解决思路记录成文保存下来，并通过个人博客（http://lovestblog.cn），个人微信公众号（“你假笨”），或者阿里内部的 ATA 技术论坛等进行传播，希望这些经验能帮助到更多的人，让他们少走弯路，更多的精力能花在他们自己的业务开发上面。

你假笨

阿里 JVM 专家寒泉子

# 译者序

早些年用 C/C++做开始时，内存管理错误一直是程序员们（包括我）在开发中面对的最大的困扰之一，内存方面的各种怪异问题经常使人抓狂，当然我也不否认 Debug 过程极其有效地帮助我更深入地学习了内存管理机制。在最初转向 Java 开发时，我觉得笼罩在头上几十年的雾霾一下都不见了，要对象是吗？来吧，new 一个。再也不用管什么时候 free 或 delete。但随着对 Java 的不断了解，我们发现天底下真的没有免费的午餐，有所得也就必有所失。在享受垃圾收集器的便利的同时，你要忍受因它造成的低效（原因有很多，比如配置不当）。而且这些年垃圾收集的技术一直在持续发展，是否采用垃圾收集？采用何种垃圾收集？如何调优？这一系列的问题在架构设计时就要考虑到，不再仅仅是“自动回收内存”那么简单。于是乎，深入了解垃圾收集技术，就十分必要了。

垃圾收集技术本身不是一个新技术，各种效率方面的质疑与探讨也不是什么新话题，而垃圾收集器本身也发展了一代一代又一代。但很多开发人员因为工作原因，停留在知其然而不知其所以然的层面上。而本书，针对 Java G1 垃圾收集器以及调试诊断方面，做了全面而深入的总结。目前专注于 G1 的书籍目前几乎是个空白，本书的出现恰好弥补了这方面的内容。本书的发起者及作者之一的 Charlie Hunt，也是 *Java™ Performance* 的作者，他曾任 Oracle 公司首席 JVM 性能工程师，负责 HotSpot Java 虚拟机和 Java SE 类库性能的改进。而其他几位作者同样在 G1 垃圾收集器方面有非常深厚的功底和丰富的调优经验。本书作为 *Java™ Performance* 的姊妹篇，内容可能不如其那么丰富，但在 G1 的细节方面会更加

深入与广泛。

作为译者，我已经尽可能地避免错误和疏漏出现在译本中，不过由于译者本人能力有限，书中难免有些不妥之处，希望广大读者与同行批评指正。

最后要感谢电子工业出版社的张春雨老师与倩雪编辑，感谢他们在我翻译过程中给予的帮助和包容。同时还要感谢家人和朋友的宝贵支持，这段时间基本处于失联状态，现在我又回来啦。谢谢你们。

李源
途牛旅游网供应链研发中心总经理

# 前言

欢迎来到 Java 性能调优指南。本书为 *Java*™ *Performance*[1] 提供了配套教程，其首次发行于 2011 年 9 月。尽管本书涉及的附加主题的内容不如 *Java*™ *Performance* 那么丰富，但会更加广泛地深入细节。本书涉及到的主题有 G1 垃圾收集器，也称作“Garbage First 垃圾收集器”，以及 Java HotSpot VM Serviceability Agent。本书附录中还包含了一些额外的 HotSpot VM 命令行选项，它们没有被包含进 *Java*™ *Performance* 关于 HotSpot VM Serviceability Agent 命令行选项的附录中。

如果你正在使用 Java8，有兴趣迁移到 Java8，或者计划使用 Java9，有很大可能是要么在评估 G1 垃圾收集器，要么已经在使用它。因此，本书中的信息将对你有很大帮助。如果你有兴趣诊断意料之外的 HotSpot VM 失败原因，或者学习更多有关现代 Java 虚拟机的细节，本书中关于 HotSpot VM Serviceability Agent 的内容也将对你很有用。不仅是 HotSpot VM 开发者，对于那些日常工作包含定位以及排除 HotSpot VM 故障行为的 Oracle 支持工程师来说，HotSpot VM Serviceability Agent 都是他们的可选工具。

本书从 G1 垃圾收集器的概述开始，通过提供为什么 G1 作为一个垃圾收集器被开发出来并被包含进 HotSpot VM 中的背景，接着是关于 G1 垃圾收集器是如何工作的概况。紧随其后的是 2 个关于 G1 的附加章节。第 1 部分是对 G1 内部的深入描述。如果你已经很好地理解了 G1 垃圾收集器是如何工作的，并且有进一步调优 G1 的需求，或者想要知道更多它的内部工作，那么这一章将是一个很好的着入点。第 3 章讲述了如何针对你的应用程序进

行 G1 调优。G1 的主要设计点之一就是能简化实现良好性能所需的调优工作。举例来说，主要输入 G1 的是可用的初始且最大的 Java 堆尺寸，以及你可以容忍 GC 暂停的最长时间。通过这些输入，当执行你的应用程序时 G1 将尝试自适应地调整来满足它们。如果你想要获得更好的性能，或者想要在 G1 上做一些额外的调优，就可以在这一章节找到你想要的信息。

最后一章完全专注于 HotSpot VM Serviceability Agent。这一章节会提供一个如何使用 SA 的深入描述和说明。如果你有兴趣了解更多关于 HotSpot VM 的内部原理或者如何诊断和排除 HotSpot VM 的意外问题，那么这一章节正是为你准备的。在本章节中你将会通过案例及说明学习如何利用 HotSpot VM Serviceability Agent 用不同方式观察和分析 HotSpot VM 行为。

最后，附录中包含了一些 HotSpot VM 命令行选项，它们没有被包含在 *Java*™ *Performance* 中关于 HotSpot VM 命令行选项的附录里。在附录中的许多 HotSpot VM 命令行选项都与 G1 相关。我们建议在合适的时间尝试使用这些选项，而不是仅仅把它们列在清单上。

## 引用

1 Charlie Hunt and Binu John. *Java*™ Performance. Addison-Wesley, Upper Saddle River, NJ, 2012. ISBN 978-0-13-714252-1.

在 informit.com 上登记你的 Java 性能调优指南的副本，当它们有可用的下载、更新以及修改时，你可以更便捷地获取相关内容。到 informit.com/注册并登陆或者创建一个账户来开始这个注册过程。输入产品 ISBN（9780133796827）并单击提交，一旦完成这个过程，你会发现在“已注册产品”下有一些可用的奖励内容。

此为试读，需要完整PDF请访问：www.ertongbook.com

轻松注册成为博文视点社区用户（www.broadview.com.cn），您即可享受以下服务：

- **提交勘误**：您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **与作者交流**：在页面下方【读者评论】处留下您的疑问或观点，与作者和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/30981

二维码：

# 致谢

## Charlie Hunt

对于那些曾经考虑过写一本书，或者对需要付出的努力好奇的人，我要告诉他们，写书绝对是一件繁重的任务！对我而言，这一切离不开许多人的帮助，在这里我很难将他们全部提及。

我尝试着从文章草稿开始回想对本书产生过深远影响的人。首先要感谢我的合著者：Monica Beckwith、Bengt Rutisson 和 Poonam Parhar。第一次浮现写一本 *Java™ Performance* 姊妹书的念头时，我认为可以将机会提供给那些有才华的 HotSpot VM 工程师，以此来展示他们的专业知识，这一定会非常棒。我确信我从他们每一个人身上学到的远比他们从我身上学到的要多，他们对本书的贡献使我感到无比的骄傲。

我要给予 Monica Beckwith 真诚的感谢，为了她在分享自身深入的 G1 GC 知识时的毅力和激情。在 G1 的早期时候，我有幸每天和 Monic 针对 G1 性能方面开展工作，最终全权让她去做。她在推动 G1 的性能以及分享她自身 G1 的知识等方面做出了杰出的工作成绩。

我同样要特别提到 Poonam Parhar 并感谢她的耐心。Poonam 非常耐心地等待其他贡献者完成他们的初稿——多年来一直是如此耐心。让我们所有人都能够及时完成初稿，令这

本书可能至少提前了 2 年上架。

我还要对无论是过去还是现在的整个 HotSpot VM 团队、HotSpot GC 工程师团队，尤其是 G1 GC 工程师们表达我的感谢。

同时还要感谢本书审稿人 Paul Hohensee 对细节的精益求精，他为提升可读性提出了绝妙的建议，还有 Tony Printezis 对 G1 GC 每个细节以及 G1 优化建议的彻底又全面的检查。

同样要感谢 John Cuthbertson 分享他的关于 G1 的知识，John 同样是曾和我一起工作过的最有才能的排除并发性故障的工程师之一。我不认为我可以用有关 G1 的离奇问题将他难住，如果那些问题明显属于某种并发错误，他总是能够追踪到。

同样感谢 Bernard Traversat 和 Georges Saab 在后续收集 *Java*™ *Performance* 材料时的支持和鼓励。

尤其感谢我们的编辑 Greg Doench，感谢他在我们多次延迟交稿、完成审稿以及将底稿逐渐成形并交到他手上这些过程中的耐心。

最后，感谢我的妻子 Barb 和儿子 Boyd，忍受了我又一轮的写书经历！

## Monica Beckwith

当我的导师 Charlie Hunt 让我在本书中写几个章节时，我感到非常荣幸。但我丝毫不知道竟然会花费这么长时间，所以首先需要感谢我的合作者们自始至终的耐心以及 Charlie 自始至终的坚持和鼓励。说到鼓励，我想谢谢我的丈夫 Ben Beckwith，他在我沮丧的时候总是全心全意地对我说很多鼓励的话，还为草稿做了最初的审校。谢谢你，Ben。接下来当然要谢谢我的 2 个孩子，Annika 和 Bodin，还有我的妈妈 Usha，他们全心全意地支持我和这本书。

我在 G1 上的技术实力来源于 John Cuthbertson，非常感谢他支持我疯狂的问题并能耐心地聆听，以及和我一起为“使 G1 自适应”和“制服混合收集”而工作。过去我们常讨论自适应标记阈值，我厌倦了拼写 `InitiatingHeapOccupancyPercent` 和它的全称，所以我将其缩写成 IHOP，John 非常喜欢这一缩写。非常难得能遇见像 John 和 Charlie 这样能鼓舞人心的同事。

接下来是 Paul Hoheness 和 Tony Printezis，他们都是我的导师，我可以向你保证他们审阅我的章节时花费的耐心至少帮助我提升了文章 75%的可读性和内容！:）

谢谢你们所有人信任我，鼓励我。我永远感激不尽！

## Poonam Parhar

当 Charlie 建议我针对服务性代理写一章节的时候，我感到非常荣幸和激动。因为这个绝妙的工具在全球还鲜为人知，讨论它的实用性和功能将会非常有益，所以我认为这是个很不错的主意。但是我之前从未写过书，为此我感到很紧张。特别感谢 Charlie 对我的信任和鼓励以及，自始至终地辅导我完成 SA 章节的编写。

我要感谢我的主管 Mattis Castegren，她一直支持和鼓励我在这本书上所做的工作，同时她也是关于 SA 章节的第一个审稿人。非常感谢 Kevin Walls 评审我的章节并帮助我提高文章内容质量。

尤其要感谢我的丈夫，同时也是我最好的朋友，Onkar，感谢他的支持以及无论什么时候我需要帮助，他一直都在那里。当然我要感谢我的两个小天使 Amanvir 和 Karanvir，他们是我的动力和幸福的源泉。

我最真诚地感谢我的父亲 Subhash C. Bajaj，他那具有感染力的快乐，一直是光明之源，感谢他一直鼓励我永不放弃。

## Bengt Rutisson

当 Charlie 让我给这本书写一个章节的时候，我感到非常光荣和荣幸。我之前从没有写过书，显然不知道这有多少工作量——哪怕只是一章！我非常感谢来自 Charlie 以及所有审稿人的支持。没有他们的支持，我将无法完成这一章节。

非常感谢我的妻子 Sara Fritzell，自始至终地鼓励我，帮助我在截止日期前完成了这一章节。当然还非常地感谢我们的孩子 Max、Elsa、Teo、Emil 和 Lina，在我写作期间一直忍受着我。

我还要感谢所有无论是过去还是现在的 HotSpot GC 工程师团队的成员们，他们是我迄今为止一起工作过的最有才能的一帮工程师。我从每个人身上都学到了很多，他们都在很多方面都启发了我。

# 作者介绍

**Charlie Hunt**（芝加哥，伊利诺伊州）目前是一名在 Oracle 主导各种 Java SE 和 HotSpot VM 项目的 JVM 工程师，他的首要关注点在维持吞吐量和延迟的同时减少内存占用量。他也是 *Java*™ *Performance* 一书的第一作者。他是 JavaOne 大会的常任主持，并被公认为是 Java 超级明星。他同样是很多会议的发言人，包括 QCon、Velocity、GoTo 和 Dreamforce。Charlie 之前为 Oracle 主导过各种 Java SE 和 HotSpot VM 项目，经历过多个不同性能的岗位，包括在 Salesforce.com 担任性能工程架构师，以及在 Oracle 和 Sun Microsystems 担任 HotSpot VM 性能架构师。他在 1998 年写下了他的第一个 Java 应用程序，在 1999 年作为 Java 高级架构师加入 Sun Microsystems，从那以后一直对 Java 和 JVM 的性能抱有热情。

**Monica. Beckwith** 是一位独立的性能顾问，主要从事优化基于 Java 虚拟机的服务级系统的客户应用程序。她过去的工作经历包括 Oracle、Sun Microsystems 和 AMD。Monica 曾经从事用 Java HotSpot VM 优化 JIT 编译器、生成代码、JVM 启发式算法，以及垃圾收集和垃圾收集器方面的工作。她是许多会议上的固定发言人并多次发表主题为垃圾收集、Java 内存模型等的文章。Monica 领导过 Oracle 的 G1 垃圾收集器性能团队，并被人称为 JavaOne 摇滚明星。

**Poonam Parhar**（圣克拉拉，加利福尼亚州）现在是一名在 Oracle 的 JVM 支持工程师，她的主要工作职责是解决针对 JRockit 和 HotSpot VM 的客户升级问题。她喜欢调试和排除故障，并且一直关注着 HotSpot VM 适用性和可维护性的提升。她明确了 HotSpot VM 里很多复杂的垃圾收集问题，并且为了能更方便进行故障排除和修复垃圾收集器相关的问题，她一直致力于提升调试工具和产品可维护性。她为可适用性代理调试器做出很多贡献，并为它开发了一个 VisualVM 插件。她在 2011 年的 JavaOne 会议上分享了“适用于 SA 的 VisualVM 插件”。为了帮助客户和 Java 社区，她通过在 https://blogs.oracle.com/poonam/上维护博客来分享自己的工作经验和知识。

**Bengt Rutisson**（斯德哥尔摩，瑞典）是一名 Oracle 的 JVM 工程师，他在 HotSpot 工程团队工作。过去十年一直从事关于 JVM 里的垃圾收集器的工作，他最初接触的是 JRockit VM，随后六年使用 HotSpot VM。Bengt 是 OpenJDK 项目中的积极参与者，在特性、稳定性修复以及性能增强方面做出了许多贡献。

此为试读，需要完整PDF请访问：www.ertongbook.com