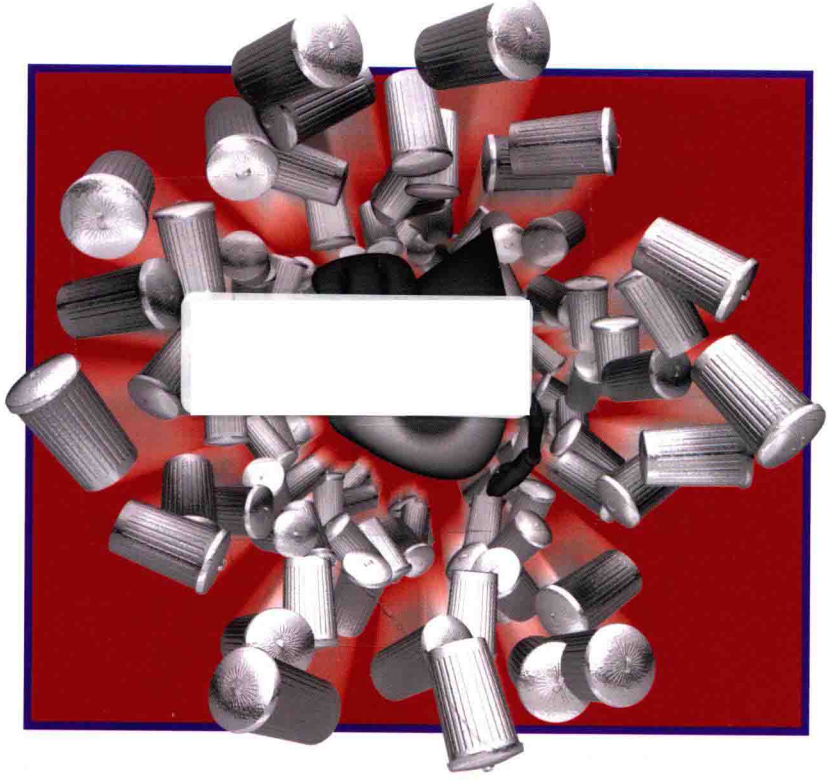
Broadview®
www.broadview.com.cn

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

Charlie Hunt
[美] Monica Beckwith 著
Poonam Parhar
Bengt Rutisson
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。

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。

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。

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。

目录

前言 XIII

致谢 XVI

作者介绍 XIX

第 1 章 Garbage First 综述 1

术语 1
并行垃圾收集器 2
串行垃圾收集器 4
并发标记清除（CMS）垃圾收集器 5
收集器的概括总结 7
Garbage First（G1）垃圾收集器 8
G1 设计 10
巨型（Humongous）对象 12
Full 垃圾收集 12
并发周期 13
堆空间调整 13

引用 14

第 2 章 深入 Garbage First 垃圾收集器 15

背景 15
G1 中的垃圾收集 16
年轻代 17
年轻代收集暂停 18
对象老化与老年代 19
巨型分区 19
混合收集 22
收集集合及其重要性 24
已记忆集合及其重要性 24
并发优化线程以及栅栏 28
G1 GC 的并发标记 30
并发标记阶段 34
初始标记 34
根分区扫描 34
并发标记 34
重新标记 36
清除 36
转移失败与 Full 收集 37
引用 38

第 3 章 Garbage First 垃圾收集器性能优化 39

年轻代收集的各阶段 39
所有并行活动的开始 41
外部根分区 42
已记忆集合和已处理缓冲区 42
已记忆集合总结 44
转移和回收 47
终止 47
GC 外部的并行活动 48

所有并行活动总结 48
所有串行活动的启动 48
其他串行活动 49
年轻代调优 50
并发标记阶段调优 52
混合垃圾收集阶段回顾 54
混合垃圾收集阶段调优 56
避免转移失败 59
引用处理 60
观察引用处理 60
引用处理调优 62
引用 65

第 4 章 The Serviceability Agent 67

SA 是什么 67
为什么要用 SA 68
SA 组件 68
JDK 中的 SA 二进制文件 69
SA 的 JDK 版本说明 69
SA 如何获得 Hotspot 虚拟机的内部数据结构 70
SA 版本对照 71
SA 调试工具 72
HSDB 72
HSDB 工具 80
命令行 Hotspot 调试器 CLHSDB 100
其他工具 103
CoreDump 和崩溃 Dump 文件 109
调试非本地生成的 Core 文件 109
SA 的共享库问题 110
消除共享库问题 110
SA 的系统属性 111

SA 的环境变量 113
JDI 实现 114
扩展 SA 工具 115
VisualVM 的 SA 插件 118
VisualVM 中怎样安装 SA 插件 119
SA 插件使用 119
SA 插件功能 120
用 SA 做故障分析 124
内存溢出错误分析 124
诊断语言层死锁 132
事后分析 Hotspot 虚拟机崩溃 137
附录　虚拟机命令行附加参数探秘 145
索引 155

1

Garbage First 综述

本章结合 JAVA HotSpot 虚拟机（下文简称 HotSpot）中垃圾收集器的发展史介绍 Garbage First（G1）垃圾收集器（GC）以及其被引入 HotSpot 背后的过程。我们假设读者已经熟悉基本的垃圾收集概念，诸如年轻代、老年代以及压缩。*Java*™ *Performance* 的第 3 章“JVM Overview”是学习这方面更多概念的良好途径。

1999 年，串行 GC 第一个被引入 HotSpot 的垃圾收集器，并作为 Java Development Kit（JDK）1.3.1 的组成部分。2002 年，并行 GC 和并发标记清除（CMS）GC 被引入 JDK 1.4.2。这 3 种垃圾收集器基本覆盖了 GC 最重要的 3 种使用场景：“内存占用空间以及并发开销最小化”、“应用吞吐量最大化”和“GC 相关中断时间最小化”。也许有人要问，“为什么我们需要一个新的垃圾收集器，比如 G1？”。在回答这个问题之前，让我们先澄清几个经常被用来对比垃圾收集器的术语，然后再对 4 种垃圾收集器做个简单的概览，包括 G1，同时识别出 G1 与其他收集器的具体不同。

术语

在本章中，我们要明确 3 个术语：*并行*、stop-the-world、*并发*。

*并行*的意思是，这是个多线程的垃圾收集运算。当一个垃圾收集事件活动被描述为并

行，就意味着它正用多线程来执行。当一个垃圾收集器被描述为并行，就意味着它是用多线程来执行垃圾收集。就 HotSpot 的垃圾收集器而言，几乎所有多线程 GC 操作都由 Java VM（JVM）的内部线程处理。与此相比，一个很重要的例外就是 G1 垃圾收集器，在 G1 中某些后台的垃圾收集工作能够由应用线程来承担。更多细节可参见第 2 章“深入 Garbage First 垃圾收集器”和第 3 章“Garbage First 垃圾收集器性能优化”。

术语 *stop-the-world* 的意思是在一个垃圾收集事件中，所有的 JAVA 应用线程全部被暂停。一个 stop-the-world 垃圾收集器就意味着当它执行垃圾收集操作时会停掉所有 JAVA 应用线程。当一个垃圾收集阶段或事件被描述为 stop-the-world，也就意味着在这个特定的垃圾收集阶段或事件中，所有 JAVA 应用线程都会被暂停。

*并发*的意思是在 JAVA 应用执行过程中垃圾收集活动也同时在进行。一个并发垃圾收集阶段或事件意味着这个垃圾收集阶段或事件可以和应用在同一时间执行。

一个垃圾收集器可以用以上 3 个术语中的任意一个或多个组合来描述。举个例子，一个并行并发收集器是多线程的（并行部分），同时还可以与应用同一时间执行（并发部分）。

并行垃圾收集器

并行垃圾收集器是一种并行 stop-the-world 的收集器，也就是说每发生一次垃圾收集，它会停掉所有应用的线程并用多个线程执行垃圾回收工作。因此垃圾回收工作可以不受任何中断非常高效地完成。对相关的应用来说，这通常也是最小化垃圾收集工作开销时间的最好方式。然而在个别情况下，因垃圾回收而导致的应用中断也可能会非常长。

在并行垃圾收集器中，年轻代和老年代的回收都是并行的，而且会 stop-the-world。老年代的回收还会同时进行压缩动作。压缩可以将邻近的对象移动到一起，以消除它们之间被浪费的空间，形成一个最理想的堆布局。然而压缩可能会花费相当长的时间，这通常跟 JAVA 堆的大小以及老年代中存活对象的数量和大小有关。

在并行垃圾收集器被引入 HotSpot 的时候，只有年轻代会使用并行 stop-the-world 收集器。老年代的回收是使用一个单线程的 stop-the-world 收集器。我们回到最初并行垃圾收集器被引入的时候，激活并发垃圾收集器的 HotSpot 命令行选项是`-XX:+UseParallelGC`。

在并行垃圾收集器被引入时，主要是为了应对服务端要求吞吐量最优化的使用场景，因此并行垃圾收集器成为了 HotSpot 服务端虚拟机的缺省收集器。同时，绝大多数 JAVA 堆的尺寸已经趋向于 512MB~2GB，这使得并行垃圾收集器的中断时间保持一个相对低的水平，哪怕是单线程的 stop-the-world 收集器。当时对延迟方面的需求比现今更加宽松。对