

深入理解Java虚拟机：JVM高级特性与最佳实践（第2版）

作者：周志明

深入理解Java虚拟机

——JVM高级特性与最佳实践

周志明 著

ISBN: 978-7-111-42190-0

本书纸版由机械工业出版社于2013年出版，电子版由华章分社（北京华章图文信息有限公司）全球范围内制作与发行。

版权所有，侵权必究

客服热线：+ 86-10-68995265

客服信箱：service@bbbvip.com

官方网址：www.hzmedia.com.cn

新浪微博 @研发书局

腾讯微博 @yanfabook

目 录

[前言](#)

[第2版与第1版的区别](#)

[本书面向的读者](#)

[如何阅读本书](#)

[语言约定](#)

[内容特色](#)

[参考资料](#)

[勘误和支持](#)

[致谢](#)

[第一部分 走近Java](#)

[第1章 走近Java](#)

[1.1 概述](#)

[1.2 Java技术体系](#)

[1.3 Java发展史](#)

[1.4 Java虚拟机发展史](#)

[1.4.1 Sun Classic/Exact VM](#)

[1.4.2 Sun HotSpot VM](#)

[1.4.3 Sun Mobile-Embedded VM/Meta-Circular VM](#)

[1.4.4 BEA JRockit/IBM J9 VM](#)

[1.4.5 Azul VM/BEA Liquid VM](#)

[1.4.6 Apache Harmony/Google Android Dalvik VM](#)

[1.4.7 Microsoft JVM及其他](#)

[1.5 展望Java技术的未来](#)

[1.5.1 模块化](#)

[1.5.2 混合语言](#)

[1.5.3 多核并行](#)

[1.5.4 进一步丰富语法](#)

[1.5.5 64位虚拟机](#)

[1.6 实战：自己编译JDK](#)

[1.6.1 获取JDK源码](#)

[1.6.2 系统需求](#)

[1.6.3 构建编译环境](#)

[1.6.4 进行编译](#)

[1.6.5 在IDE工具中进行源码调试](#)

[1.7 本章小结](#)

[第二部分 自动内存管理机制](#)

[第2章 Java内存区域与内存溢出异常](#)

[2.1 概述](#)

[2.2 运行时数据区域](#)

[2.2.1 程序计数器](#)

[2.2.2 Java虚拟机栈](#)

[2.2.3 本地方法栈](#)

[2.2.4 Java堆](#)

[2.2.5 方法区](#)

[2.2.6 运行时常量池](#)

[2.2.7 直接内存](#)

[2.3 HotSpot虚拟机对象探秘](#)

[2.3.1 对象的创建](#)

[2.3.2 对象的内存布局](#)

[2.3.3 对象的访问定位](#)

[2.4 实战：OutOfMemoryError异常](#)

[2.4.1 Java堆溢出](#)

[2.4.2 虚拟机栈和本地方法栈溢出](#)

[2.4.3 方法区和运行时常量池溢出](#)

[2.4.4 本机直接内存溢出](#)

[2.5 本章小结](#)

[第3章 垃圾收集器与内存分配策略](#)

[3.1 概述](#)

[3.2 对象已死吗](#)

[3.2.1 引用计数算法](#)

[3.2.2 可达性分析算法](#)

[3.2.3 再谈引用](#)

- [3.2.4 生存还是死亡](#)
- [3.2.5 回收方法区](#)
- [3.3 垃圾收集算法](#)
 - [3.3.1 标记-清除算法](#)
 - [3.3.2 复制算法](#)
 - [3.3.3 标记-整理算法](#)
 - [3.3.4 分代收集算法](#)
- [3.4 HotSpot的算法实现](#)
 - [3.4.1 枚举根节点](#)
 - [3.4.2 安全点](#)
 - [3.4.3 安全区域](#)
- [3.5 垃圾收集器](#)
 - [3.5.1 Serial收集器](#)
 - [3.5.2 ParNew收集器](#)
 - [3.5.3 Parallel Scavenge收集器](#)
 - [3.5.4 Serial Old收集器](#)
 - [3.5.5 Parallel Old收集器](#)
 - [3.5.6 CMS收集器](#)
 - [3.5.7 G1收集器](#)
 - [3.5.8 理解GC日志](#)
 - [3.5.9 垃圾收集器参数总结](#)
- [3.6 内存分配与回收策略](#)
 - [3.6.1 对象优先在Eden分配](#)
 - [3.6.2 大对象直接进入老年代](#)
 - [3.6.3 长期存活的对象将进入老年代](#)
 - [3.6.4 动态对象年龄判定](#)
 - [3.6.5 空间分配担保](#)
- [3.7 本章小结](#)
- [第4章 虚拟机性能监控与故障处理工具](#)
 - [4.1 概述](#)
 - [4.2 JDK的命令行工具](#)
 - [4.2.1 jps: 虚拟机进程状况工具](#)
 - [4.2.2 jstat: 虚拟机统计信息监视工具](#)
 - [4.2.3 jinfo: Java配置信息工具](#)
 - [4.2.4 jmap: Java内存映像工具](#)
 - [4.2.5 jhat: 虚拟机堆转储快照分析工具](#)
 - [4.2.6 jstack: Java堆栈跟踪工具](#)
 - [4.2.7 HSDIS: JIT生成代码反汇编](#)
 - [4.3 JDK的可视化工具](#)
 - [4.3.1 JConsole: Java监视与管理控制台](#)
 - [4.3.2 VisualVM: 多合一故障处理工具](#)
- [4.4 本章小结](#)
- [第5章 调优案例分析与实战](#)
 - [5.1 概述](#)
 - [5.2 案例分析](#)
 - [5.2.1 高性能硬件上的程序部署策略](#)
 - [5.2.2 集群间同步导致的内存溢出](#)
 - [5.2.3 堆外内存导致的溢出错误](#)
 - [5.2.4 外部命令导致系统缓慢](#)
 - [5.2.5 服务器JVM进程崩溃](#)
 - [5.2.6 不恰当数据结构导致内存占用过大](#)
 - [5.2.7 由Windows虚拟内存导致的长时间停顿](#)
 - [5.3 实战: Eclipse运行速度调优](#)
 - [5.3.1 调优前的程序运行状态](#)
 - [5.3.2 升级JDK 1.6的性能变化及兼容问题](#)
 - [5.3.3 编译时间和类加载时间的优化](#)
 - [5.3.4 调整内存设置控制垃圾收集频率](#)
 - [5.3.5 选择收集器降低延迟](#)
- [5.4 本章小结](#)
- [第三部分 虚拟机执行子系统](#)
- [第6章 类文件结构](#)
 - [6.1 概述](#)
 - [6.2 无关性的基石](#)

- 6.3 [Class类文件的结构](#)
 - 6.3.1 [魔数与Class文件的版本](#)
 - 6.3.2 [常量池](#)
 - 6.3.3 [访问标志](#)
 - 6.3.4 [类索引、父类索引与接口索引集合](#)
 - 6.3.5 [字段表集合](#)
 - 6.3.6 [方法表集合](#)
 - 6.3.7 [属性表集合](#)
- 6.4 [字节码指令简介](#)
 - 6.4.1 [字节码与数据类型](#)
 - 6.4.2 [加载和存储指令](#)
 - 6.4.3 [运算指令](#)
 - 6.4.4 [类型转换指令](#)
 - 6.4.5 [对象创建与访问指令](#)
 - 6.4.6 [操作数栈管理指令](#)
 - 6.4.7 [控制转移指令](#)
 - 6.4.8 [方法调用和返回指令](#)
 - 6.4.9 [异常处理指令](#)
 - 6.4.10 [同步指令](#)
- 6.5 [公有设计和私有实现](#)
- 6.6 [Class文件结构的发展](#)
- 6.7 [本章小结](#)

第7章 [虚拟机类加载机制](#)

- 7.1 [概述](#)
- 7.2 [类加载的时机](#)
- 7.3 [类加载的过程](#)
 - 7.3.1 [加载](#)
 - 7.3.2 [验证](#)
 - 7.3.3 [准备](#)
 - 7.3.4 [解析](#)
 - 7.3.5 [初始化](#)
- 7.4 [类加载器](#)
 - 7.4.1 [类与类加载器](#)
 - 7.4.2 [双亲委派模型](#)
 - 7.4.3 [破坏双亲委派模型](#)
- 7.5 [本章小结](#)

第8章 [虚拟机字节码执行引擎](#)

- 8.1 [概述](#)
- 8.2 [运行时栈帧结构](#)
 - 8.2.1 [局部变量表](#)
 - 8.2.2 [操作数栈](#)
 - 8.2.3 [动态连接](#)
 - 8.2.4 [方法返回地址](#)
 - 8.2.5 [附加信息](#)
- 8.3 [方法调用](#)
 - 8.3.1 [解析](#)
 - 8.3.2 [分派](#)
 - 8.3.3 [动态类型语言支持](#)
- 8.4 [基于栈的字节码解释执行引擎](#)
 - 8.4.1 [解释执行](#)
 - 8.4.2 [基于栈的指令集与基于寄存器的指令集](#)
 - 8.4.3 [基于栈的解释器执行过程](#)
- 8.5 [本章小结](#)

第9章 [类加载及执行子系统的案例与实战](#)

- 9.1 [概述](#)
- 9.2 [案例分析](#)
 - 9.2.1 [Tomcat: 正统的类加载器架构](#)
 - 9.2.2 [OSGi: 灵活的类加载器架构](#)
 - 9.2.3 [字节码生成技术与动态代理的实现](#)
 - 9.2.4 [Retrotranslator: 跨越JDK版本](#)
- 9.3 [实战: 自己动手实现远程执行功能](#)
 - 9.3.1 [目标](#)
 - 9.3.2 [思路](#)

- [9.3.3 实现](#)
- [9.3.4 验证](#)
- [9.4 本章小结](#)
- [第四部分 程序编译与代码优化](#)
- [第10章 早期（编译期）优化](#)
- [10.1 概述](#)
- [10.2 Javac编译器](#)
 - [10.2.1 Javac的源码与调试](#)
 - [10.2.2 解析与填充符号表](#)
 - [10.2.3 注解处理器](#)
 - [10.2.4 语义分析与字节码生成](#)
- [10.3 Java语法糖的味道](#)
 - [10.3.1 泛型与类型擦除](#)
 - [10.3.2 自动装箱、拆箱与遍历循环](#)
 - [10.3.3 条件编译](#)
- [10.4 实战：插入式注解处理器](#)
 - [10.4.1 实战目标](#)
 - [10.4.2 代码实现](#)
 - [10.4.3 运行与测试](#)
 - [10.4.4 其他应用案例](#)
- [10.5 本章小结](#)
- [第11章 晚期（运行期）优化](#)
- [11.1 概述](#)
- [11.2 HotSpot虚拟机内的即时编译器](#)
 - [11.2.1 解释器与编译器](#)
 - [11.2.2 编译对象与触发条件](#)
 - [11.2.3 编译过程](#)
 - [11.2.4 查看及分析即时编译结果](#)
- [11.3 编译优化技术](#)
 - [11.3.1 优化技术概览](#)
 - [11.3.2 公共子表达式消除](#)
 - [11.3.3 数组边界检查消除](#)
 - [11.3.4 方法内联](#)
 - [11.3.5 逃逸分析](#)
- [11.4 Java与C/C++的编译器对比](#)
- [11.5 本章小结](#)
- [第五部分 高效并发](#)
- [第12章 Java内存模型与线程](#)
- [12.1 概述](#)
- [12.2 硬件的效率与一致性](#)
- [12.3 Java内存模型](#)
 - [12.3.1 主内存与工作内存](#)
 - [12.3.2 内存间交互操作](#)
 - [12.3.3 对于volatile型变量的特殊规则](#)
 - [12.3.4 对于long和double型变量的特殊规则](#)
 - [12.3.5 原子性、可见性与有序性](#)
 - [12.3.6 先行发生原则](#)
- [12.4 Java与线程](#)
 - [12.4.1 线程的实现](#)
 - [12.4.2 Java线程调度](#)
 - [12.4.3 状态转换](#)
- [12.5 本章小结](#)
- [第13章 线程安全与锁优化](#)
- [13.1 概述](#)
- [13.2 线程安全](#)
 - [13.2.1 Java语言中的线程安全](#)
 - [13.2.2 线程安全的实现方法](#)
- [13.3 锁优化](#)
 - [13.3.1 自旋锁与自适应自旋](#)
 - [13.3.2 锁消除](#)
 - [13.3.3 锁粗化](#)
 - [13.3.4 轻量级锁](#)
 - [13.3.5 偏向锁](#)

[13.4 本章小结](#)

[附录](#)

[附录A 编译Windows版的OpenJDK](#)

[A.1 获取JDK源码](#)

[A.2 系统需求](#)

[A.3 构建编译环境](#)

[A.4 准备依赖项](#)

[A.5 进行编译](#)

[附录B 虚拟机字节码指令表](#)

[附录C HotSpot虚拟机主要参数表](#)

[C.1 内存管理参数](#)

[C.2 即时编译参数](#)

[C.3 类型加载参数](#)

[C.4 多线程相关参数](#)

[C.5 性能参数](#)

[C.6 调试参数](#)

[附录D 对象查询语言（OQL）简介](#)

[D.1 SELECT子句](#)

[D.2 FROM子句](#)

[D.3 WHERE子句](#)

[D.4 属性访问器](#)

[D.5 OQL语言的BNF范式](#)

[附录E JDK历史版本轨迹](#)

前言

Java是目前用户最多、使用范围最广的软件开发技术之一。Java的技术体系主要由支撑Java程序运行的虚拟机、提供各开发领域接口支持的Java API、Java编程语言及许多第三方Java框架（如Spring、Struts等）构成。在国内，有关Java API、Java语言语法及第三方框架的技术资料和书籍非常丰富，相比之下，有关Java虚拟机的资料却显得异常贫乏。

这种状况在很大程度上是由Java开发技术本身的一个重要优点导致的：在虚拟机层面隐藏了底层技术的复杂性以及机器与操作系统的差异性。运行程序的物理机器的情况千差万别，而Java虚拟机则在千差万别的物理机上建立了统一的运行平台，实现了在任意一台虚拟机上编译的程序都能在任何一台虚拟机上正常运行。这一极大优势使得Java应用的开发比传统C/C++应用的开发更高效和快捷，程序员可以把主要精力集中在具体业务逻辑上，而不是物理硬件的兼容性上。在一般情况下，一个程序员只要了解了必要的Java API、Java语法，以及学习适当的第三方开发框架，就已经基本能满足日常开发的需要了，虚拟机会在用户不知不觉中完成对硬件平台的兼容及对内存等资源的管理工作。因此，了解虚拟机的运作并不是一般开发人员必须掌握的知识。

然而，凡事都具备两面性。随着Java技术的不断发展，它被应用于越来越多的领域之中。其中一些领域，如电力、金融、通信等，对程序的性能、稳定性和可扩展性方面都有极高的要求。程序很可能在10个人同时使用时完全正常，但是在10000个人同时使用时就会缓慢、死锁，甚至崩溃。毫无疑问，要满足10000个人同时使用需要更高性能的物理硬件，但是在绝大多数情况下，提升硬件效能无法等比例地提升程序的运作性能和并发能力，甚至可能对程序运作状况完全没有任何改善。这里面有Java虚拟机的原因：为了达到给所有硬件提供一致的虚拟平台的目的，牺牲了一些与硬件相关的性能特性。更重要的是人为原因：如果开发人员不了解虚拟机一些技术特性的运行原理，就无法写出最适合虚拟机运行和自优化的代码。

其实，目前商用的高性能Java虚拟机都提供了相当多的优化特性和调节手段，用于满足应用程序在实际生产环境中对性能和稳定性的要求。如果只是为了入门学习，让程序在自己的机器上正常运行，那么这些特性可以说是可有可无的；如果用于生产开发，尤其是企业级生产开发，就迫切需要开发人员中至少有一部分人对虚拟机的特性及调节方法具有很清晰的认识，所以在Java开发体系中，对架构师、系统调优师、高级程序员等角色的需求一直都非常大。学习虚拟机中各种自动运作特性的原理也成为了Java程序员成长道路上必然会接触到的一课。本书可以使读者以一种相对轻松的方式学习虚拟机的运作原理，对Java程序员的成长也有较大的帮助。

第2版与第1版的区别

JDK 1.7在2011年7月28日正式发布，相对于2006年发布的JDK 1.6，新版的JDK有了许多新的特性和改进。本书的第2版也相应地进行了修改和升级，把讲解的技术平台从JDK 1.6提升至JDK 1.7。例如，增加了对JDK 1.7中最新的G1收集器，以及JDK 1.7中JSR-292 InvokeDynamic（对非Java语言的调用支持）的分析讲解等内容。

在第1版出版后，笔者收到了许多热心读者的反馈意见，部分读者提出OpenJDK开源已久，第1版却很少有直接分析OpenJDK源码的内容，有点“视宝山而不见”的感觉。因此，在本书第2版中，笔者特别加强了对这部分内容的讲解，其中在第1章中就介绍了如何分析、调试OpenJDK源码等。在本书后续章节中，不少关于功能点的讲解都直接使用OpenJDK中的HotSpot源码或者JIT编译器生成的本地代码作为论据。

如何把Java虚拟机原理中许多理论性很强的知识、特性应用于实践开发，是本书贯穿始终的主旨。由于笔者希望在本书第2版中进一步加强知识的实践性，因此增加了许多对处理JVM常见问题技能的讲解，包括如何分析GC日志、如何分析JIT编译器代码优化过程和生成代码等。并且，在第1版的基础上，第2版中进一步增加了若干处理JVM问题的实践案例供读者参考。

另外，本书第2版还修正了第1版中多处错误的、有歧义的和不完整的描述。有关勘误信息，可以参考第1版的勘误页面（<http://icyfenix.iteye.com/blog/1119214>）。

本书面向的读者

（1）使用Java技术体系的中、高级开发人员

Java虚拟机作为中、高级开发人员必须修炼的知识，有着较高的学习门槛，本书可作为学习虚拟机的优秀教材。

（2）系统调优师

系统调优师是近几年才兴起的职业，本书中的大量案例、代码和调优实战将会对系统调优师的日常工作有直接的帮助。

（3）系统架构师

保障系统的性能、并发和伸缩等能力是系统架构师的主要职责之一，而这部分与虚拟机的运作密不可分，本书可以作为他们制定应用系统底层框架的参考资料。

如何阅读本书

本书一共分为五个部分：走近Java、自动内存管理机制、虚拟机执行子系统、程序编译与代码优化、高效并发。各部分基本上是互相独立的，没有必然的前后依赖关系，读者可以从任何一个感兴趣的专题开始阅读，但是每个部分中的各个章节间有先后顺序。

本书并没有假设读者在Java领域具备很专业的技术水平，因此在保证逻辑准确的前提下，尽量用通俗的语言和案例讲述虚拟机中与开发的关系最为密切的内容。当然，学习虚拟机技术本身就需要读者有一定的基础，且本书的读者定位是中、高级程序员，因此本书假设读者自己了解一些常用的开发框架、Java API和Java语法等基础知识。

笔者希望读者在阅读本书的同时，把本书中的实践内容亲自验证一遍，其中用到的代码清单可以从华章网站（<http://www.hzbook.com>）下载。

语言约定

本书在语言和技术上有如下约定：

本书中提到HotSpot、JRockit虚拟机、WebLogic服务器等产品的所有者时，仍然使用Sun和BEA公司的名称，实际上，BEA和Sun分别于2008年和2009年被Oracle公司收购，现在已经不存在这两个商标了，但毫无疑问的是，它们都是在Java领域中做出过卓越贡献的、值得程序员纪念的公司。

JDK从1.5版本开始，在官方的正式文档与宣传资料中已经不再使用类似“JDK 1.5”的名称，只有程序员内部使用的开发版本号（Developer Version，例如java-version的输出）才继续沿用1.5、1.6和1.7的版本号，而公开版本号（Product Version）则改为JDK 5、JDK 6和JDK 7的命名方式，为了行文一致，本书所有场合统一采用开发版本号的命名方式。

由于版面关系，本书中的许多示例代码都没有遵循最优的代码编写风格，如使用的流没有关闭流等，请读者在阅读时注意这一点。

如果没有特殊说明，本书中所有讨论都是以Sun JDK 1.7为技术平台的。不过如果有某个特性在各个版本间的变化较大，一般都会说明它在各个版本间的差异。

欢迎访问：电子书学习和下载网站 (<https://www.shgis.cn>)

文档名称：深入理解Java虚拟机：JVM高级特性与最佳实践（第2版） - 周志明.epub

请登录 <https://shgis.cn/post/1869.html> 下载完整文档。

手机端请扫码查看：

