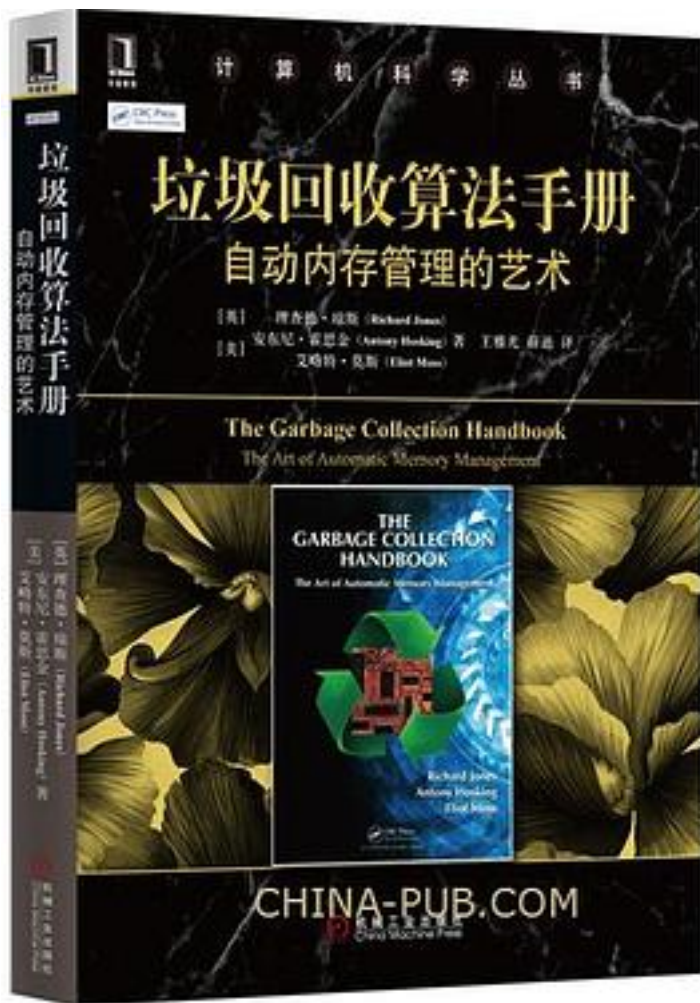


垃圾回收算法手册：自动内存管理的艺术



[垃圾回收算法手册：自动内存管理的艺术 下载链接1](#)

著者:Richard Jones

出版者:机械工业出版社

出版时间:2016-3

装帧:平装

isbn:9787111528821

在自动内存管理领域，Richard Jones于1996年出版的《Garbage Collection: Algorithms for Automatic Dynamic Memory

Management》可谓是一部里程碑式的作品。接近20年过去了，垃圾回收技术得到了非常大的发展，因此有必要将该领域当前最先进的技术呈现给读者。本书汇集了自动内存管理研究者和开发者们在过去50年间的丰富经验，在本书中，作者在一个统一的易于接受的框架内比较了当下最重要的回收策略以及最先进的回收技术。

本书从近年来硬件与软件的发展给垃圾回收所带来的新挑战出发，探讨了这些挑战给高性能垃圾回收器的设计者与实现者所带来的影响。在简单的传统回收算法之外，本书还涵盖了并行垃圾回收、增量式垃圾回收、并发垃圾回收以及实时垃圾回收。书中配备了丰富的伪代码与插图，以描述各种算法与概念。

本书特色

为1996年《Garbage Collection: Algorithms for Automatic Dynamic Memory Management》一书提供了完整的、最新的、权威的续作。

全面讲解并行垃圾回收算法、并发垃圾回收算法以及实时垃圾回收算法。

深入剖析某些垃圾回收领域的棘手问题，包括与运行时系统的接口。

提供在线数据库支持，包含超过2500条垃圾回收相关文献。

作者介绍:

理查德·琼斯 (Richard Jones)

坎特伯雷-肯特大学计算机学院教授。1998年联合创立了国际存储管理研讨会，并担任*届会议主席。他发表了多篇关于垃圾回收技术、堆可视化技术、电子出版技术相关的论文，多次担任主要国际会议计划委员会的常务委员，同时还是《Software Practice and Experience》杂志的编辑委员会成员。因在动态存储管理领域的研究和学术成绩，他于2005年被聘任为格拉斯哥大学名誉研究员，2006年被计算机协会评为杰出科学家。

安东尼·霍思金 (Antony Hosking)

普渡大学西拉法叶分校计算机学院副教授。他的主要研究方向是编程语言的设计与实现，特别是数据库与持久化编程语言、面向对象数据库系统、动态存储管理、编译器优化以及编程语言和应用的架构支持。

艾略特·莫斯 (Eliot Moss)

马萨诸塞大学阿默斯特分校计算机科学学院教授。他的主要研究方向为编程语言及其实现，而且早在1978年就构建出垃圾回收器。除了自动存储管理领域之外，他在持久编程语言、虚拟机实现、事务性编程与事务内存方面也拥有较高的知名度。他曾与IBM研究员一起推动Jikes RVM Java虚拟机的学术研究许可，并*终促使其成为开源项目。

目录: 出版者的话

译者序

前言

作者简介

第1章 引言 1

1.1 显式内存释放 1

1.2 自动动态内存管理 3

1.3 垃圾回收算法之间的比较 5

1.3.1 安全性	5
1.3.2 吞吐量	5
1.3.3 完整性与及时性	5
1.3.4 停顿时间	6
1.3.5 空间开销	7
1.3.6 针对特定语言的优化	7
1.3.7 可扩展性与可移植性	8
1.4 性能上的劣势	8
1.5 实验方法	8
1.6 术语和符号	10
1.6.1 堆	10
1.6.2 赋值器与回收器	11
1.6.3 赋值器根	11
1.6.4 引用、域和地址	11
1.6.5 存活性、正确性以及可达性	12
1.6.6 伪代码	12
1.6.7 分配器	13
1.6.8 赋值器的读写操作	13
1.6.9 原子操作	13
1.6.10 集合、多集合、序列以及元组	14
第2章 标记-清扫回收	15
2.1 标记-清扫算法	16
2.2 三色抽象	18
2.3 改进的标记-清扫算法	18
2.4 位图标记	19
2.5 懒惰清扫	21
2.6 标记过程中的高速缓存不命中问题	24
2.7 需要考虑的问题	25
2.7.1 赋值器开销	25
2.7.2 吞吐量	26
2.7.3 空间利用率	26
2.7.4 移动，还是不移动	26
第3章 标记-整理回收	28
3.1 双指针整理算法	29
3.2 Lisp 2算法	30
3.3 引线整理算法	32
3.4 单次遍历算法	34
3.5 需要考虑的问题	36
3.5.1 整理的必要性	36
3.5.2 整理的吞吐量开销	36
3.5.3 长寿数据	36
3.5.4 局部性	37
3.5.5 标记-整理算法的局限性	37
第4章 复制式回收	38
4.1 半区复制回收	38
4.1.1 工作列表的实现	39
4.1.2 示例	40
4.2 遍历顺序与局部性	42
4.3 需要考虑的问题	46
4.3.1 分配	46
4.3.2 空间与局部性	47
4.3.3 移动对象	48
第5章 引用计数	49
5.1 引用计数算法的优缺点	50

5.2 提升效率	51
5.3 延迟引用计数	52
5.4 合并引用计数	54
5.5 环状引用计数	57
5.6 受限域引用计数	61
5.7 需要考虑的问题	62
5.7.1 应用场景	62
5.7.2 高级的解决方案	62
第6章 垃圾回收器的比较	64
6.1 吞吐量	64
6.2 停顿时间	65
6.3 内存空间	65
6.4 回收器的实现	66
6.5 自适应系统	66
6.6 统一垃圾回收理论	67
6.6.1 垃圾回收的抽象	67
6.6.2 追踪式垃圾回收	67
6.6.3 引用计数垃圾回收	69
第7章 内存分配	72
7.1 顺序分配	72
7.2 空闲链表分配	73
7.2.1 首次适应分配	73
7.2.2 循环首次适应分配	75
7.2.3 最佳适应分配	75
7.2.4 空闲链表分配的加速	76
7.3 内存碎片化	77
7.4 分区适应分配	78
7.4.1 内存碎片	79
7.4.2 空间大小分级的填充	79
7.5 分区适应分配与简单空闲链表分配的结合	81
7.6 其他需要考虑的问题	81
7.6.1 字节对齐	81
7.6.2 空间大小限制	82
7.6.3 边界标签	82
7.6.4 堆可解析性	82
7.6.5 局部性	84
7.6.6 拓展块保护	84
7.6.7 跨越映射	85
7.7 并发系统中的内存分配	85
7.8 需要考虑的问题	86
第8章 堆内存的划分	87
8.1 术语	87
8.2 为何要进行分区	87
8.2.1 根据移动性进行分区	87
8.2.2 根据对象大小进行分区	88
8.2.3 为空间进行分区	88
8.2.4 根据类别进行分区	89
8.2.5 为效益进行分区	89
8.2.6 为缩短停顿时间进行分区	90
8.2.7 为局部性进行分区	90
8.2.8 根据线程进行分区	90
8.2.9 根据可用性进行分区	91
8.2.10 根据易变性进行分区	91
8.3 如何进行分区	92

8.4 何时进行分区	93
第9章 分代垃圾回收	95
9.1 示例	95
9.2 时间测量	96
9.3 分代假说	97
9.4 分代与堆布局	97
9.5 多分代	98
9.6 年龄记录	99
9.6.1 集体提升	99
9.6.2 衰老半区	100
9.6.3 存活对象空间与柔性提升	101
9.7 对程序行为的适应	103
9.7.1 Appel式垃圾回收	103
9.7.2 基于反馈的对象提升	104
9.8 分代间指针	105
9.8.1 记忆集	106
9.8.2 指针方向	106
9.9 空间管理	107
9.10 中年优先回收	108
9.11 带式回收框架	110
9.12 启发式方法在分代垃圾回收中的应用	112
9.13 需要考虑的问题	113
9.14 抽象分代垃圾回收	115
第10章 其他分区策略	117
10.1 大对象空间	117
10.1.1 转轮回收器	118
10.1.2 在操作系统支持下的对象移动	119
10.1.3 不包含指针的对象	119
10.2 基于对象拓扑结构的回收器	119
10.2.1 成熟对象空间的回收	120
10.2.2 基于对象相关性的回收	122
10.2.3 线程本地回收	123
10.2.4 栈上分配	126
10.2.5 区域推断	127
10.3 混合标记-清扫、复制式回收器	128
10.3.1 Garbage-First回收	129
10.3.2 Immix回收以及其他回收	130
10.3.3 受限内存空间中的复制式回收	133
10.4 书签回收器	134
10.5 超引用计数回收器	135
10.6 需要考虑的问题	136
第11章 运行时接口	138
11.1 对象分配接口	138
11.1.1 分配过程的加速	141
11.1.2 清零	141
11.2 指针查找	142
11.2.1 保守式指针查找	143
11.2.2 使用带标签值进行精确指针查找	144
11.2.3 对象中的精确指针查找	145
11.2.4 全局根中的精确指针查找	147
11.2.5 栈与寄存器中的精确指针查找	147
11.2.6 代码中的精确指针查找	157
11.2.7 内部指针的处理	158
11.2.8 派生指针的处理	159

- 11.3 对象表 159
- 11.4 来自外部代码的引用 160
- 11.5 栈屏障 162
- 11.6 安全回收点以及赋值器的挂起 163
- 11.7 针对代码的回收 165
- 11.8 读写屏障 166
 - 11.8.1 读写屏障的设计工程学 167
 - 11.8.2 写屏障的精度 167
 - 11.8.3 哈希表 169
 - 11.8.4 顺序存储缓冲区 170
 - 11.8.5 溢出处理 172
 - 11.8.6 卡表 172
 - 11.8.7 跨越映射 174
 - 11.8.8 汇总卡 176
 - 11.8.9 硬件与虚拟内存技术 176
 - 11.8.10 写屏障相关技术小结 177
 - 11.8.11 内存块链表 178
- 11.9 地址空间管理 179
- 11.10 虚拟内存页保护策略的应用 180
 - 11.10.1 二次映射 180
 - 11.10.2 禁止访问页的应用 181
- 11.11 堆大小的选择 183
- 11.12 需要考虑的问题 185
- 第12章 特定语言相关内容 188
 - 12.1 终结 188
 - 12.1.1 何时调用终结方法 189
 - 12.1.2 终结方法应由哪个线程调用 190
 - 12.1.3 是否允许终结方法彼此之间的并发 190
 - 12.1.4 是否允许终结方法访问不可达对象 190
 - 12.1.5 何时回收已终结对象 191
 - 12.1.6 终结方法执行出错时应当如何处理 191
 - 12.1.7 终结操作是否需要遵从某种顺序 191
 - 12.1.8 终结过程中的竞争问题 192
 - 12.1.9 终结方法与锁 193
 - 12.1.10 特定语言的终结机制 193
 - 12.1.11 进一步的研究 195
 - 12.2 弱引用 195
 - 12.2.1 其他动因 196
 - 12.2.2 对不同强度指针的支持 196
 - 12.2.3 使用虚对象控制终结顺序 199
 - 12.2.4 弱指针置空过程的竞争问题 199
 - 12.2.5 弱指针置空时的通知 199
 - 12.2.6 其他语言中的弱指针 200
 - 12.3 需要考虑的问题 201
- 第13章 并发算法预备知识 202
 - 13.1 硬件 202
 - 13.1.1 处理器与线程 202
 - 13.1.2 处理器与内存之间的互联 203
 - 13.1.3 内存 203
 - 13.1.4 高速缓存 204
 - 13.1.5 高速缓存一致性 204
 - 13.1.6 高速缓存一致性对性能的影响示例：自旋锁 205
 - 13.2 硬件内存一致性 207
 - 13.2.1 内存屏障与先于关系 208

- 13.2.2 内存一致性模型 209
- 13.3 硬件原语 209
 - 13.3.1 比较并交换 210
 - 13.3.2 加载链接/条件存储 211
 - 13.3.3 原子算术原语 212
 - 13.3.4 检测-检测并设置 213
 - 13.3.5 更加强大的原语 213
 - 13.3.6 原子操作原语的开销 214
- 13.4 前进保障 215
- 13.5 并发算法的符号记法 217
- 13.6 互斥 218
- 13.7 工作共享与结束检测 219
- 13.8 并发数据结构 224
 - 13.8.1 并发栈 226
 - 13.8.2 基于单链表的并发队列 228
 - 13.8.3 基于数组的并发队列 230
 - 13.8.4 支持工作窃取的并发双端队列 235
- 13.9 事务内存 237
 - 13.9.1 何谓事务内存 237
 - 13.9.2 使用事务内存助力垃圾回收器的实现 239
 - 13.9.3 垃圾回收机制对事务内存的支持 240
- 13.10 需要考虑的问题 241
- 第14章 并行垃圾回收 242
 - 14.1 是否有足够多的工作可以并行 243
 - 14.2 负载均衡 243
 - 14.3 同步 245
 - 14.4 并行回收的分类 245
 - 14.5 并行标记 246
 - 14.6 并行复制 254
 - 14.6.1 以处理器为中心的并行复制 254
 - 14.6.2 以内存为中心的并行复制技术 258
 - 14.7 并行清扫 263
 - 14.8 并行整理 264
 - 14.9 需要考虑的问题 267
 - 14.9.1 术语 267
 - 14.9.2 并行回收是否值得 267
 - 14.9.3 负载均衡策略 267
 - 14.9.4 并行追踪 268
 - 14.9.5 低级同步 269
 - 14.9.6 并行清扫与并行整理 270
 - 14.9.7 结束检测 270
- 第15章 并发垃圾回收 271
 - 15.1 并发回收的正确性 272
 - 15.1.1 三色抽象回顾 273
 - 15.1.2 对象丢失问题 274
 - 15.1.3 强三色不变式与弱三色不变式 275
 - 15.1.4 回收精度 276
 - 15.1.5 赋值器颜色 276
 - 15.1.6 新分配对象的颜色 276
 - 15.1.7 基于增量更新的解决方案 277
 - 15.1.8 基于起始快照的解决方案 277
 - 15.2 并发回收的相关屏障技术 277
 - 15.2.1 灰色赋值器屏障技术 278
 - 15.2.2 黑色赋值器屏障技术 279

- 15.2.3 屏障技术的完整性 280
- 15.2.4 并发写屏障的实现机制 281
- 15.2.5 单级卡表 282
- 15.2.6 两级卡表 282
- 15.2.7 减少回收工作量的相关策略 282
- 15.3 需要考虑的问题 283
- 第16章 并发标记-清扫算法 285
 - 16.1 初始化 285
 - 16.2 结束 287
 - 16.3 分配 287
 - 16.4 标记过程与清扫过程的并发 288
 - 16.5 即时标记 289
 - 16.5.1 即时回收的写屏障 290
 - 16.5.2 Doligez-Leroy-Gonthier回收器 290
 - 16.5.3 Doligez-Leroy-Gonthier回收器在Java中的应用 292
 - 16.5.4 滑动视图 292
 - 16.6 抽象并发回收框架 293
 - 16.6.1 回收波面 294
 - 16.6.2 增加追踪源头 295
 - 16.6.3 赋值器屏障 295
 - 16.6.4 精度 295
 - 16.6.5 抽象并发回收器的实例化 296
 - 16.7 需要考虑的问题 296
- 第17章 并发复制、并发整理算法 298
 - 17.1 主体并发复制：Baker算法 298
 - 17.2 Brooks间接屏障 301
 - 17.3 自删除读屏障 301
 - 17.4 副本复制 302
 - 17.5 多版本复制 303
 - 17.6 Sapphire回收器 306
 - 17.6.1 回收的各个阶段 306
 - 17.6.2 相邻阶段的合并 311
 - 17.6.3 Volatile域 312
 - 17.7 并发整理算法 312
 - 17.7.1 Compressor回收器 312
 - 17.7.2 Pauseless回收器 315
 - 17.8 需要考虑的问题 321
- 第18章 并发引用计数算法 322
 - 18.1 简单引用计数算法回顾 322
 - 18.2 缓冲引用计数 324
 - 18.3 并发环境下的环状引用计数处理 326
 - 18.4 堆快照的获取 326
 - 18.5 滑动视图引用计数 328
 - 18.5.1 面向年龄的回收 328
 - 18.5.2 算法实现 328
 - 18.5.3 基于滑动视图的环状垃圾回收 331
 - 18.5.4 内存一致性 331
 - 18.6 需要考虑的问题 332
- 第19章 实时垃圾回收 333
 - 19.1 实时系统 333
 - 19.2 实时回收的调度 334
 - 19.3 基于工作的实时回收 335
 - 19.3.1 并行、并发副本回收 335
 - 19.3.2 非均匀工作负载的影响 341

19.4 基于间隙的实时回收	342
19.4.1 回收工作的调度	346
19.4.2 执行开销	346
19.4.3 开发者需要提供的信息	347
19.5 基于时间的实时回收：Metronome回收器	347
19.5.1 赋值器使用率	348
19.5.2 对可预测性的支持	349
19.5.3 Metronome回收器的分析	351
19.5.4 鲁棒性	355
19.6 多种调度策略的结合：“税收与开支”	355
19.6.1 “税收与开支”调度策略	356
19.6.2 “税收与开支”调度策略的实现基础	357
19.7 内存碎片控制	359
19.7.1 Metronome回收器中的增量整理	360
19.7.2 单处理器上的增量副本复制	361
19.7.3 Stopless回收器：无锁垃圾回收	361
19.7.4 Staccato回收器：在赋值器无等待前进保障条件下的尽力整理	363
19.7.5 Chicken回收器：在赋值器无等待前进保障条件下的尽力整理（x86平台）	365
19.7.6 Clover回收器：赋值器乐观无锁前进保障下的可靠整理	366
19.7.7 Stopless回收器、Chicken回收器、Clover回收器之间的比较	367
19.7.8 离散分配	368
19.8 需要考虑的问题	370
术语表	372
参考文献	383
索引	413
• • • • •	(收起)

[垃圾回收算法手册：自动内存管理的艺术_下载链接1](#)

标签

垃圾回收

GC

计算机

JVM

编译原理

编程

评论

96年那本是在图书馆环境工程分类看到的（吊打图书管理员），之后反复借阅。今天终于买到了新版。这两本书确实在GC领域地位崇高。就是价格有点傲娇……

还真是手册啊，只扫了一遍目录，我只是想单纯了解一下GC，但是本书的内容不仅仅如此，比较适合做PL或者VM的人，对我来说没多大作用，或者说没到时候。

在杭州图书馆读完这本书。垃圾回收的基本原理很简单，无非是引用计数，标记清理整理复制之类，但考虑细节，又纷繁复杂。为了停顿时间和吞吐量，需要考虑cache，局部性，顺序性，算法复杂度不再是唯一的衡量，再加上并发并行GC，又得考虑并发正确性。不过此书还多是纸上谈兵，还得研究具体实现。

超5星推荐，看至一半也没有半分失彩，堪称字字珠玑，俨然一本论文概述集。论述配得上 the art of, 信息量也足够 handbook。看技术书就这样，总是机心徘徊，苦乐参半。闲暇之余，业余爱好，业余时间，已婚，育有三个孩子……看来，擅长内存管理的生活上也有迹可循。

去年年底研究G1的时候仔细看过，看了大部分，还没看完，但是看过的部分给了我不少启示。后面再解决相关问题的时候再继续

扫读。阅读需要较强知识背景，确实当成手册比较好。

大部分还不错，中文阅读速度比读英文快速。但是偶尔一个单词只差的翻译错误，能让

读者郁闷半天，翻开原文才豁然大悟；
还发现一个问题，对于已经刊登的勘误，翻译竟然没有修正过来。而是将错误的原文翻译过来，实在是不走心啊

1.书籍覆盖面很广，是对GC这个领域到发书日的一个比较完善的文献综述，适合当手册和了解gc的各个领域看

2.翻译不够好，比如flag还有翻译为旗标这种，建议有条件还是去看原版

读一半翻一半，总算是看完了，大概是知识储备不够，囫圇吞枣，以后再回来精读。

一来书中本身逻辑不太清晰，二来翻译有些生硬，三来伪代码比较难懂，写的不清不楚的。

[垃圾回收算法手册：自动内存管理的艺术_下载链接1](#)

书评

更新

我看有人评论这本书太理论没有代码示例（好像是在什么电商评论的），这个实际是极大的误解，这本书其实是给Jikes RVM的垃圾回收器做的超详细注释好吗。你看代码时候有啥地方抓秃了脑袋也理解不了时候找找这本书相关章节，总会有惊喜的。有些地方MMTk就是按照书里实现的。...

以前在学校图书馆翻到这本书，当时就感觉捡到了一个宝物。虽然翻译很糟糕，但是里面的内容还是吸引我反复看了好几回。
毕业后回忆以前图书馆见过的经典，发现这本书老早就买不到了，算是一个遗憾。难得重新翻译了出版了，遂入了一本。虽然要100多块，但毕竟比较是受众比较...

英文版本是垃圾算法界的“龙书”。。原著太贵下载的pdf，看了1/3，没想到出了中文译本，果断入了，虽然100块。很值得。
垃圾回收系统的各个模块都有涉及，分配器，回收器，堆管理。现在高级语言逐渐的都离不开垃圾回收，我想这也是未来语言的趋势，随着垃圾回收算法，自动内存...

[垃圾回收算法手册：自动内存管理的艺术_下载链接1](#)