WILEY

垃圾收集

Garbage Collection

Algorithms for Automatic Dynamic Memory Management

Richard Jones
Rafael Lins 著

谢之易 译

人民邮电出版社
POSTS & TELECOM PRESS

垃圾收集

Richard Jones
Rafael Lins
著

谢之易 译

人民邮电出版社

图书在版编目（CIP）数据

垃圾收集 /（美）琼斯（Jones, R. ），（美）林斯（Lins, R.）著；谢之易译.
—北京：人民邮电出版社，2004．4
ISBN 7-115-12070-6

Ⅰ．垃… Ⅱ．①琼…②林…③谢… Ⅲ．电子邮件—基本知识 Ⅳ．TP393.098

中国版本图书馆 CIP 数据核字（2004）第 007011 号

版 权 声 明

垃圾收集

◆ 著 Richard Jones Rafael Lins
译 谢之易
责任编辑 陈冀康

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www. ptpress.com.cn
读者热线 010-67132705
北京汉魂图文设计有限公司制作
北京隆昌伟业印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092 1/16
印张：22. 5
字数：541 千字 2004 年 4 月第 1 版
印数：1-4 000 册 2004 年 4 月北京第 1 次印刷

著作权合同登记 图字：01 - 2003 - 0668 号
ISBN 7-115-12070-6/TP • 3831

定价：45.00 元

本书如有印装质量问题，请与本社联系 电话：（010）67129223

内 容 提 要

本书围绕着动态内存自动回收的话题，介绍了垃圾收集机制，详细分析了各种算法和相关技术。

本书共 12 章。第 1 章首先介绍计算机存储器管理的演化和自动内存回收的需求，并引入了本书所使用的术语和记法。第 2 章介绍了 3 种“经典”的垃圾收集技术：引用计数（reference counting）、标记一清扫（mark-sweep）和节点复制（copying）。 随后的 4 章更详细地讨论了上述这些垃圾收集方式和标记一缩并（mark-compact）收集。第 7 章和第 8 章分别介绍了在现代垃圾收集实现中具有重要地位的分代式（generational）垃圾收集和渐进式（incremental）垃圾收集。第 9 章和第 10 章扩展了垃圾收集的领域，讨论了如何让垃圾收集能够在无法得到来自语言编译器的支持的环境（分别是 C 和 C++）中运行。第 11 章讨论了一个相对较新的研究领域 —— 垃圾收集和硬件数据 cache 的相互作用。第 12 章简要地考察了用于分布式系统的垃圾收集。

本书适合对动态内存管理感兴趣的读者阅读，可供专业的研究人员参考。

译 者 序

时至今日，垃圾收集（garbage collection）已经成为了一种在主流的软件开发领域中得到普遍应用的技术，它带来的好处是显而易见的。多年以来，内存管理错误一直是人们在程序开发中最大的困扰之一，若能彻底摆脱这个麻烦，无疑是每个开发人员的福音。然而，围绕着垃圾收集存在许多争议，特别是关于它对程序执行效率的影响，历来是争论的热点。这从侧面反映了这样一个事实：是否采用垃圾收集是一个涉及全局的决策，它的影响决不仅仅是“不再需要手工释放内存”那么简单，因此，深入了解垃圾收集技术是十分重要的。要想开始这种了解，本书就是一个很好的出发点。

与许多人的印象不同，垃圾收集是一种十分“古老”的技术，它与 LISP 语言一起出现在 20 世纪 60 年代。1995 年，当 James Gosling 和他的同事们决定将垃圾收集作为 Java 的一部分时，它早已是一种相当成熟的技术了。在 IT 业高度发达的美国，这是一种比较常见的模式：一项新技术，首先在学术界、在大学里经过学者们充分的酝酿和尝试，然后再由工程师们带入业界，作为“新”产品推出。此时，这项技术已经基本成熟，有了坚实的理论和实践基础。显然，这种“大学栽树，企业结果”的模式，正是美国大学可以保持较高的学术水平、美国 IT 企业能够不断推陈出新的重要原因之一，也是“科教与产业相结合”最好的诠释。而那种“学校做着企业该做的事，而企业不得不来做学校该做的事”的做法，则是应该摒弃的。

前面提到，垃圾收集技术经历了多年的发展和演化。因此，今天人们对垃圾收集的种种诘问和质疑，绝大多数并不是什么新话题。它们早已被当作颇具挑战性的课题研究了许多年，而且成果丰硕。但令人遗憾的是，由于垃圾收集技术长期以来处于“阳春白雪”的状态，因此大量精妙的构想散见于数以千计的学术论文中，除了专攻这个领域的学者，它们并不为人们所熟知。

这本书的出现，恰好弥补了这个遗憾。本书的两位作者都是这一领域内的专家，他们把大量学术文献中有关垃圾收集的知识和技术抽取出来，理清脉络，有机地组织在一起，为读者们提供了一幅关于垃圾收集技术的全景图。就我所知，在这一方面，本书可以说是“独此一家，别无分号”。本书从垃圾收集技术的基本概念和基本技术出发，深入浅出地分析了各种技术背后的运行机理，指出它们的优势和缺点，进而介绍人们为克服这些缺点所设想的种种改进方案，并着重介绍了能够有效提高垃圾收集运行效率的分代式（generational）技术和可以大大扩展垃圾收集使用范围的渐进式（incremental）技术。与此同时，对于一些常见的误解，本书也用事实进行了澄清。特别地，两位作者一再强调，对于是否选择垃圾收集与选择何种垃圾收集方案，并不存在一个简单而普适的原则，而是必须在充分理解系统的需求和特征的基础上决定。这种反对简单化、反对“非此即彼”的态度，在某种意义上可以说是这本书的基调。总的来说，如果你希望深入了解垃圾收集技术，无论是打算在这一领域内展开研究，还是需要构造自己的垃圾收集实现，甚至仅仅是为了更好地使用垃圾收集，这本书都是

非常有益的。

可能会让部分读者失望的是，由于成书较早，本书中并没有提到一些比较“时髦的”技术（例如 Java 或是.Net），所举的例子绝大多数来自函数式语言。而且，两位作者都是大学教授，因此书中的论述比较偏重于技术背后的算法和构想，而非具体的实现。此外，由于作者的背景，大量关于程序语言方面的知识（函数式语言、延迟求值、图规约等等），在书中是不加说明而引入的，不太熟悉这些知识的读者在阅读时可能会略感困难。对此，我的建议是首先阅读《程序设计语言 —— 概念和结构》（Ravi Sethi 著、裘宗燕等译）一书，对于理解相关内容会有很大帮助。

作为译者，我已经尽可能地避免错误和疏漏出现在译本中，但鉴于墨菲定律的强大力量，可能仍然有一些角落避过了我的视线。为此，我将在 http://childchen.51.net/errata/gc.htm 维护本书的勘误表。如果你发现任何错误，请发送电子邮件给我，地址是：wizard@sh163.net。

最后（也是最俗气的一部分），在这里我要向许多人致以深深的谢意。首先，我要感谢人民邮电出版社的陈冀康编辑和我的朋友王昕（cber），没有他们，我不会有机会翻译这样一本有趣的书。其次，我要感谢本书的作者 Richard Jones 和北大的裘宗燕教授，在翻译过程中他们给了我许多帮助。最后，我要感谢我的家人和朋友，在这几个月里，是他们给了我宝贵的支持，让我能够从疲倦和挫折感带来的沮丧情绪中振奋起来，最终完成这本书的翻译。谢谢你们！

译 者

2003 年 12 月于上海

前　言

这是一本关于垃圾收集（Garbage Collection，GC）的图书，而垃圾收集是指在程序最后一次使用堆分配的内存之后将其自动回收的技术。存储器始终是而且总是一种稀缺而宝贵的资源。在计算机发展的早期阶段，在超大规模集成电路发明之前，存储器是相当昂贵的，甚至像 Unix 这样的分时操作系统都被期望能够在 64kB 的段内运行。今天，尽管我们可以相对廉价地买到 SIMM 存储器模块，而且安装也很容易，但是各种程序在消耗资源方面也越来越挥霍无度。Microsoft Windows95，一个用于单用户个人计算机的操作系统，它要达到最佳的运行状态需要超过 12MB 的 RAM。因此，在主存储器上的花费可能占了整台 PC 成本的一半。像所有宝贵的资源一样，我们需要小心地管理存储器，并在不再需要使用的时候回收它们。

许多计算机程序的存储器需求是简单且可以预测的。在这类程序中，存储器的分配和释放可以由程序员或编译器来处理，而且此时这种做法确实是最好的。而其他一些程序则在规模和复杂性方面有了巨大的增长。类似 Lisp 和 Prolog 这样的语言通常需要操纵巨大的数据结构，而且这些数据结构之间有着复杂的相互依赖关系。函数式（functional）和逻辑式（logic）程序设计语言有着复杂的执行模式。最终的结果就是，许多数据结构的生命周期不再能够在程序执行之前（无论是由程序员还是编译器）确定了。自动内存回收是必要的。

计算机科学界对垃圾收集这个话题的争论的热烈程度，反映出垃圾收集技术变得越来越重要。除了于某些期刊和会议上发表的个别论文之外，1990、1991 和 1993 年的“Object-Oriented Systems, Languages and Applications”（OOPSLA）会议中已经有了关于垃圾收集的专题研讨会，而且在 1992、1995 和 1998 年，已经有了专门讨论这一主题的国际性研究会议。垃圾收集还是 Usenet 新闻组上一个反复出现的流行话题，它引起了广泛的争论。

今天，在软件分析、设计和编程方面，面向对象是人们兴趣增长得最强烈的领域。实现良好的软件工程的关键是控制复杂性。面向对象设计实现这一目标的方式之一，是将抽象封装入对象，让对象之间通过定义良好的接口互相通信。但是，程序员控制的内存管理限制了这种模块化。由于这个原因，绝大多数现代的面向对象语言，例如 Smalltalk、Eiffel、Java 和 Dylan，都拥有垃圾收集的支持。今天，甚至像 Modula-3 和 Oberon 这样部分用于系统编程的语言，都因为这些合理而且实际的原因提供了垃圾收集。对于像 C 和 C++这样“不合作的”语言，也已经出现了一些可用的垃圾收集库了。

致读者

关于垃圾收集的文献数量巨大。针对这个主题，存在着成千份的各种期刊文章、书中的章节、会议上的讲座、技术报告和毕业论文。尽管如此，关于垃圾收集的各种荒谬说法依然相当

盛行。“它们仅仅对于 Lisp 和函数式语言来说是必要的；它们只能用于解释器而不是编译代码；它们给程序运行带来无法忍受的开销” —— 毫无疑问，它们还有着分岔的蹄子和像叉子一样的尾巴！这种情况带来了两种后果。第一，在那些使用垃圾收集的解决方案能够带来收益的地方，它常常会被忽略。第二，当问题涉及的数据结构的复杂性要求人们使用垃圾收集时，人们往往对各种文献所提供的经验并不熟悉，从而就会出现“重新发明轮子”的情况。

这本书的目标就是，把这些宝贵的经验集中到一个容易理解的、统一的框架内。我们会描述技术上的最新进展，我们会比较它们在声明式（declarative）和命令式（imperative）程序设计风格中的应用，比较它们在顺序执行的、并发的和分布的体系结构中的应用。每一种最重要的算法都会得到详细的讨论，并且常常伴随着对它的独有特征的说明和实际使用情况的演示。我们还会讨论它的复杂度、性能、适用性、以及它与其他相关算法的关系。

我们相信，事实将会证明，对于在某些领域内工作的研究生和研究者而言，这样一份详细的调查是有用的。这些领域包括：编译器构造，函数式、逻辑式和面向对象的编程与设计，软件工程和操作系统。那些进修上述领域的高级课程的学生也应该会对这本书感兴趣本书也涉及到这些领域中研究生们感兴趣的高级话题。我们希望开发软件（从最简单的软件工具到复杂的实时系统）的职业程序员将会发现这本书很有价值。特别地，在最近的几年中，面向对象系统的普及程度提高很快。因此，对于工作在这个领域的每一位程序员来说，彻底地理解垃圾收集方法是必要的。

本书的组织方式

第 1 章首先介绍计算机存储器管理的演化和自动内存回收的需求。接着，我们描述了对象在堆中的表示，并讨论了度量不同的垃圾收集策略时的标准和尺度。在这一章的末尾，还描述了我们所使用的伪码记法。

第 2 章介绍了 3 种“经典”的垃圾收集技术：引用计数（reference counting）、标记-清扫（mark-sweep）和节点复制（copying）。对这些技术有一定经验的读者可能会希望跳过这一章。

随后的 4 章更详细地讨论了上述这些垃圾收集方式和标记-缩并（mark-compact）收集。第 7 章介绍了分代式（generational）垃圾收集，人们已经证明，在大范围的各种应用之中，这种收集方式能够有效地减小垃圾收集造成的中断和总体开销。而第 8 章描述了如何将垃圾收集与计算（computation）的其余部分细粒度地交织在一起。第 9 章和第 10 章扩展了垃圾收集的领域，讨论了如何让垃圾收集能够在无法得到来自语言编译器的支持的环境（分别是 C 和 C++）中运行。在第 11 章里，我们讨论了一个相对较新的研究领域 —— 垃圾收集和硬件数据 cache 的相互作用。最后，第 12 章简要地考察了用于分布式系统的垃圾收集。

在每一章的末尾，我们都加入了一个关于需要考虑的问题的小结。我们希望，当读者在选择收集器之前，需要回答一些关于收集器、客户程序、操作系统和体系结构的问题时（设计这些问题是为了给读者一些提示），这些小结能够提供一些指引。这些小结无疑不能替代对相关章节的阅读，而且，我们也并不试图给出“完美的”解决方案。此外，在较早章节中介绍的一

些垃圾收集策略（例如引用计数、标记-清扫或是节点复制），会在较晚的章节中重新出现。而且，我们不能错误地把某种垃圾收集策略的简单实现的性能和特征，套用到它的最先进的实现上。不过，我们希望这些小结能够提供一个进一步分析的焦点（而不是一本“烹调说明书”）。

我们还应该说明这本书中缺少了什么。存储器管理开销最小的形式就是在运行时什么也不必做。对于已经在编译时确定何时能够回收或重用对象的技术，人们已经投入了数量可观的研究工作。但是，这些工作大部分仅仅是理论上的，而且到目前为止，我们相信没有多少迹象表明这一技术能够带来实质性的性能提高。我们省略了这些材料。某些技术和窍门是语言相关的。因为C++语言正在变得越来越流行，而且越来越多的C++语言的使用者意识到他们迫切地需要垃圾收集，所以我们选择了在书中覆盖与之相关的内容，但是一般而言，我们总是把精力主要集中于能够广泛应用的方法。那些针对某种特定的程序设计风格（例如纯函数式和逻辑式程序设计）的技术，我们只是简要地提及。

最后，那些充满精力的、对在线参考文献数据库进行搜索的研究者们，将会发现其他一些关于垃圾收集技术和议题的论文。我们确实对把垃圾埋入垃圾填埋地、焚化垃圾并把它倒入大海之类的问题有兴趣。不过我们还是选择了忽略它们。人们还常常提出与公众健康和垃圾收集有关的问题 —— 不过我们这里并不打算玩文字上的游戏☺！

参考文献和本书的网站

之前我们曾经提到过，关于这个主题已经有了超过一千篇已发表的论文。这本书末尾的参考文献列表要短得多。不过，读者可以通过下面的网站访问一个全面的电子形式的数据库：

http://www.cs.ukc.ac.uk/people/staff/rej/gc.html

该参考文献列表还包含了一些可获得电子版的论文的摘要和URL。Richard Jones将会尽力更新这份参考文献的列表，而且非常希望能够收到更多的论文（最好是BibTeX格式的）和已有论文的URL（以及任何对列表中现有URL的修正）。

我们已经尽力清除本书中代码片断的任何错误。虽然我们没有勇气重复Donald Knuth的做法，为所报告的错误提供奖金，但我们会在这个Web站点上维护一个列表，记录任何已发现的错误。错误报告可以通过Email发送给R.E.Jones@ukc.ac.uk，或是邮寄给Richard Jones，地址是Computing Laboratory，University of Kent at Canterbury，Canterbury，Kent，CT27NF，UK。

致谢

如果没有一些人的鼓励、帮助和谅解，这本书将永远无法完成。我要感谢Kent大学的Theoretical Computer Science研究组，他们在一旁为我加油。特别地，我十分感激Simon Thompson，他耐心地阅读了这本书的草稿并提出了许多意见，还给予了我很多鼓励。我还要感谢Hans Boehm、Jacques Cohen、Keith Dimond和David Turner，感谢他们的评论和建议。

我还要感谢 Martin Broom、Tim Hopkins 和 Simon Thompson，感谢他们提出了关于如何掌握 LaTeX 排版的建议。Kent 大学留给我的两个用于研究的学期，以及由 British Council 与 CNPq-Brazil 资助的到巴西 Federal 大学的访问，也是非常宝贵的。

我还非常感激 Rafael Lins，是他最早产生了写这本书的想法，是他编写了关于分布式垃圾收集的章节，同时对其他一些章节也做出了贡献。我还应该向这 36 年来在垃圾收集领域内工作的所有人（人数太多，无法一一提及）致以感谢。

最后，而且也是最重要的，我必须感谢 Robbie、Helen、Kate 和 William。没有他们的支持和容忍，这一切都是不可能的。在两年多的时间里，你们容忍了我以研究工作为名占据餐厅；你们原谅了我的坏脾气；你们通情达理地接受了我不能陪你们出去玩耍的要求。我发自内心的感谢你们。

Richard Jones
Canterbury
1996 年 2 月

我非常感激那些以许多不同的方式为这本书的出版做出贡献的人们，特别是 David Turner、Simon Thompson、Jon Salkid、Rosita Wachenchauzer、Alejanfro Martinez 和 Marcia Correia。我还要感谢 Universidade Federal de Pernambuco，Recife Brazil，准予我周期性地离开（由 CAPES-Brazil 资助），感谢 CNPq-Brazil 和 British Council 资助了到 Kent 大学的数次访问。在 Kent 大学的许多朋友使得这些访问如此的令人愉快！我还要感谢所有来自 Carmo、Gilka、Maria Teresa、Rilane 和 Silvia 的支持与爱。

Rafael Lins
Recife
1996 年 2 月

修订

这次重印给了我一个机会，让我可以对这本书做一些小改进。我扩展了索引，让读者能够更容易地从算法作者的名字找到算法。1996 和 1997 两次印刷中的某些错误已经被修正了。那些既细致（足以找出错误）又好心（为我指出它们）的人包括：Andrew Appel、Nick Barnes、Stephen Bevan、Matthieu Blondeau、Hans Boehm、Thomas Burri、Morris Chang、Sid Chatterjee、Peter Dickman、Alex Garthwaite、Tim Geisler 和 Pekka Pirinen。我还要感谢 Gaynor Redvers-Mutton，John Wiley & Sons 公司的编辑，感谢她对我不断的鼓励，特别是感谢她为我提供了做这些改进的机会。

Richard Jones
Canterbury
1998 年 11 月

目　录