GC垃圾回收（计算机科学） 动态语言 修改

计算机动态语言如何合理选择垃圾收集系统? 修改

引用计数? 标记清扫? 标记压缩? 标记拷贝? 其他? ...显示全部

关注问题 写回答 邀请回答 添加评论 分享 邀请回答 举报 ...

关注者 234 被浏览 5,088

他们也关注了该问题

查看全部 6 个回答

Felis sapiens
函数式编程、编程语言、编程 话题的优秀回答者

1 人赞同了该回答

@连城的答案说得很详细了。
稍微补充一点，个人做玩具语言的话可以偷懒，想办法从语义层面避免循环引用的出现，然后直接用引用计数。比如Lisp，如果不允许对cons cell赋值，那就不会有循环引用。

UPDATE:
我答错了，如果有递归的函数那就有循环引用（不过可以用lazy thunk不经济地规避）。老老实实mark and sweep吧。。

编辑于 2014-12-31

▲ 赞同 1 ▼ 7 条评论 分享 收藏 感谢 ...

更多回答

连城
Apache Spark PMC member

86 人赞同了该回答

推荐一篇论文：A unified theory of garbage collection。这篇论文指出，tracing和reference counting（以下简称为ref count）是GC机制的两个极端，其他各种GC机制都是这二者之间的变种，同时给出了量化分析模型。Tracing的优势在于回收完备性，劣势在于回收及时性；ref count的优劣势则相反。各种mark-*类GC算法都可以视作tracing类GC；为了减少GC过程中的停顿时间而引入的分代机制（generational GC），则可以视作是在向ref count靠拢。

对于个人实现的玩具语言，选择具体GC机制时可以参考以下两个因素：

1. **实现复杂度。**各类GC中，实现最简单的两种分别是ref count和mark-copy。但缺点也很明显。Ref count有无法回收循环引用对象的问题，同时在日常使用时很容易遗忘增减引用计数（C++中可以借助智能指针缓解）。Mark-copy GC内存会导致占用量翻倍。对于面向小规模应用、运行时长较短的语言实现，可以考虑采用。早期的ActionScript采用的就是ref count GC。Python则仅对内部不含有指向其他对象的引用的对象（如整数、字符串）采用ref count，保证回收及时性；对可能含有指向其他对象的引用的复合对象（如列表、map）采用mark-sweep，保证回收完备性。
2. **GC发生时是否移动对象。**Mark-copy、mark-compact都会移动对象，ref count和mark-sweep则不会。当用C/C++等语言通过FFI（Foreign Function Interface）与动态语言运行时环境直接交互时，前者会导致麻烦。C/C++环境中往往会以指针形式引用动态语言GC堆上的对象，如果GC过程中移动了对象的位置，就会导致C/C++环境中的对象引用失效。为了避免这种情况，必须引入pin机制，也就是将指定对象的位置固定住，避免GC时被移动。V8采用的是generational mark-compact GC，相应地，V8通过C++接口中的Handle类来pin住V8运行时GC堆中的对象。Lua、mRuby则都采用经过优化的mark-sweep GC。Guile和Mono（早期版本）使用的Boehm GC也属于mark-sweep类GC。

关于作者

Felis sapiens

函数式编程、编程语言、编程 话题的优秀回答者

电影旅行敲代码、Antokha Yuuki、暮无井见铃也关注了她

回答 624 文章 40 关注者 14,871

已关注 发私信

相关问题

如何系统的学习动态语言的类型推导，类型系统等知识? 7 个回答

Objective-C是动态语言吗？为什么? 5 个回答

为什么java的垃圾收集中，复制收集器不会访问非活跃对象而标记-清除收集器则会呢? 2 个回答

为何大量设计模式在动态语言中不适用? 23 个回答

python这类动态语言怎么看文档? 7 个回答

相关推荐

至于对GC停顿时间和GC触发时机的优化，没有深入了解，就不掺和了。

编辑于 2012-10-26

▲ 赞同 86 ▼ 添加评论 分享 收藏 感谢 ···

冯东

编程、Linux 话题的优秀回答者

10 人赞同了该回答

垃圾回收方式分为两类：reference counting 和 tracing 。各种『标记xx』一般都属于后者。

Reference counting 实现简单，而且适合多线程实现。对它最大的批评是无法处理 circular reference （但是配合 weak reference counting 可以基本解决）。第二大缺点是不能有效的利用系统的空闲时间，在 cascading deletion 的时候系统性能会大幅下降（但是也有基于统计的算法可以降低 cascading deletion 的性能损失）。

展开阅读全文

▲ 赞同 10 ▼ 添加评论 分享 收藏 感谢 ···

查看全部 6 个回答