优秀的Java程序员必须了解GC的工作原理Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/569/2021_2022__E4_BC_98_E7_A7_80_E7_9A_84J_c104_569032.htm 一个优秀的Java程序员必须了解GC的工作原理、如何优化GC的性能、如何与GC进行有限的交互，因为有一些应用程序对性能要求较高，例如嵌入式系统、实时系统等，只有全面提升内存的管理效率，才能提高整个应用程序的性能。本篇文章首先简单介绍GC的工作原理之后，然后再对GC的几个关键问题进行深入探讨，最后提出一些Java程序设计建议，从GC角度提高Java程序的性能。 GC的基本原理 Java的内存管理实际上就是对象的管理，其中包括对象的分配和释放。 对于程序员来说，分配对象使用new关键字；释放对象时，只要将对象所有引用赋值为null，让程序不能够再访问到这个对象，我们称该对象为\"不可达的\".GC将负责回收所有\"不可达\"对象的内存空间。 对于GC来说，当程序员创建对象时，GC就开始监控这个对象的地址、大小以及使用情况。通常，GC采用有向图的方式记录和管理堆（heap）中的所有对象（详见 参考资料1）。通过这种方式确定哪些对象是\"可达的\"，哪些对象是\"不可达的\".当GC确定一些对象为\"不可达\"时，GC就有责任回收这些内存空间。但是，为了保证GC能够在不同平台实现的问题，Java规范对GC的很多行为都没有进行严格的规定。例如，对于采用什么类型的回收算法、什么时候进行回收等重要问题都没有明确的规定。因此，不同的JVM的实现者往往有不同的实现算法。这也给Java程序员的开发带来行多不确定性。本文研究了几个与GC工作相关的问题，努力减少这种不确定

性给Java程序带来的负面影响。 增量式GC（ Incremental GC ） GC在JVM中通常是由一个或一组进程来实现的，它本身也和用户程序一样占用heap空间，运行时也占用CPU.当GC进程运行时，应用程序停止运行。因此，当GC运行时间较长时，用户能够感到Java程序的停顿，另外一方面，如果GC运行时间太短，则可能对象回收率太低，这意味着还有很多应该回收的对象没有被回收，仍然占用大量内存。因此，在设计GC的时候，就必须在停顿时间和回收率之间进行权衡。一个好的GC实现允许用户定义自己所需要的设置，例如有些内存有限有设备，对内存的使用量非常敏感，希望GC能够准确的回收内存，它并不在意程序速度的放慢。另外一些实时网络游戏，就不能够允许程序有长时间的中断。增量式GC就是通过一定的回收算法，把一个长时间的中断，划分为很多个小的中断，通过这种方式减少GC对用户程序的影响。虽然，增量式GC在整体性能上可能不如普通GC的效率高，但是它能够减少程序的最长停顿时间。 Sun JDK提供的HotSpot JVM就能支持增量式GC.HotSpot JVM缺省GC方式为不使用增量GC，为了启动增量GC，我们必须在运行Java程序时增加-Xincgc的参数。HotSpot JVM增量式GC的实现是采用Train GC算法。它的基本想法就是，将堆中的所有对象按照创建和使用情况进行分组（分层），将使用频繁高和具有相关性的对象放在一队中，随着程序的运行，不断对组进行调整。当GC运行时，它总是先回收最老的（最近很少访问的）的对象，如果整组都为可回收对象，GC将整组回收。这样，每次GC运行只回收一定比例的不可达对象，保证程序的顺畅运行。 详解finalize函数 finalize是位于Object类的一个方法，该方法的访

问修饰符为protected，由于所有类为Object的子类，因此用户类很容易访问到这个方法。由于，finalize函数没有自动实现链式调用，我们必须手动的实现，因此finalize函数的最后一个语句通常是super.finalize（）。通过这种方式，我们可以实现从下到上实现finalize的调用，即先释放自己的资源，然后再释放父类的资源。 根据Java语言规范，JVM保证调用finalize函数之前，这个对象是不可达的，但是JVM不保证这个函数一定会被调用。另外，规范还保证finalize函数最多运行一次。 很多Java初学者会认为这个方法类似与C 中的析构函数，将很多对象、资源的释放都放在这一函数里面。其实，这不是一种很好的方式。原因有三，其一，GC为了能够支持finalize函数，要对覆盖这个函数的对象作很多附加的工作。其二，在finalize运行完成之后，该对象可能变成可达的，GC还要再检查一次该对象是否是可达的。因此，使用finalize会降低GC的运行性能。其三，由于GC调用finalize的时间是不确定的，因此通过这种方式释放资源也是不确定的。 通常，finalize用于一些不容易控制、并且非常重要资源的释放，例如一些I/O的操作，数据的连接。这些资源的释放对整个应用程序是非常关键的。在这种情况下，程序员应该以通过程序本身管理（包括释放）这些资源为主，以finalize函数释放资源方式为辅，形成一种双保险的管理机制，而不应该仅仅依靠finalize来释放资源。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com