

垃圾回收算法

什么样的对象是垃圾？

GC Root Tracing算法-从GC Root出发，所有可达的对象都是存活的对象，而所有不可达的对象都是垃圾

如何判断垃圾？

引用计数法，及其存在的循环引用问题。

GC Root Tracing算法

可达

强引用

弱引用

软引用

虚引用

可复活

不可触及

如何回收垃圾

标记清除法

复制算法

标记压缩算法

垃圾回收思想

分代思想

新生代：存活对象少，采用复制算法。使用复制算法并不需要1:1划分内存空间，实际采用8:1:1。一块大的Eden空间和两块小的Survivor。当回收时，将Eden和Survivor中还存活的对象一次性复制到另外一块Survivor空间上，最后清理掉Eden和刚用过的Eden空间。

老年代：存活对象多，标记压缩或标记清除。

分区思想

将整个堆空间划分成连续不同的小区间，每一个小区间都独立使用，独立回收，控制一个回收多少空间，控制GC的时间。

垃圾回收器

串行回收器：使用单线程进行垃圾回收

并行回收器：使用多线程进行垃圾回收

需根据机器的并行能力进行选择

CMS回收器：关注系统停顿时间，使用标记清除算法，多线程并行回收

G1回收器：使用分区算法，使得Eden、From、Survivor区和老年代等各块内存不必连续。

新生代GC

并发标记周期

混合收集

如果需要，可能进行Full GC

垃圾回收的类型

当JVM无法为一个新的对象分配空间时会触发Minor GC。

Minor GC：从年轻代空间回收内存

当年轻代中的Eden区分配满的时候，年轻代中的部分对象会晋升到老年代。

对于大部分应用程序，停顿导致的延迟都是可以忽略不计的，因为大部分Eden区中的对象都能被认为是垃圾，永远不会被复制。

Major GC：从老年代空间回收内存

许多Minor GC是由于Minor GC触发的，分配对象内存时发现内存不够，触发Minor GC。Minor会将对象移到老年代中，此时老年代空间不够，触发Major GC。

Full GC：清理整个堆空间-包括年轻代、老年代和永久代

当准备要触发一次Minor GC时，发现年轻代的剩余空间比以往晋升的空间小，则不会触发Minor GC而使转为触发Full GC。因为JVM此时认为：之前这么大空间的时候已经发生对象晋升了，那现在剩余空间更小了，那么很大概率上也会发生对象晋升。

永久代分配空间DNA已经没有足够空间时，也会触发Full GC。

补充 Stop-The-World：是指进行垃圾回收时因为标记或清理的需要，必须让所有执行任务的线程停止执行任务，从而让垃圾回收线程回收垃圾。