链滴

内存回收算法

作者：AlanSune

原文链接：https://ld246.com/article/1591523661938

来源网站：链滴

本文基于《深入理解Java虚拟机第二版》

在Java运行数据区时我们了解了Java运行时数据的各个区域，其中程序计数器、Java虚拟机栈和本地法栈的生命周期与线程相关，在线程创建时分配，在线程结束时释放，故该部分内存不需要进行专门内存回收。而方法区和Java堆内存的分配和回收是动态的，内存回收主要关注这部分内存。在Java虚机规范中，并没有对虚拟机应当如何进行垃圾回收进行规定，不同的虚拟机的实现方式也不同，下面要是对相关的算法进行介绍。

对象存活判定

在GC收集器回收内存之前，首先要判定哪些对象不再存活(不存在该对象的引用)，存在以下两种算法以进行对象存活的判断。

引用计数算法

最基本的算法是给每个对象添加一个引用的计数器，当增加一个该对象的引用时，对计数器加一，反则减一。在进行回收时，计数器的值为零，则表明该对象不再被使用，可以进行回收。该方法既简单又有效率，是一个非常实用的方法。然而，它存在一个严重的问题，即它无法解决对象之间的相互引。

```
public class ReferenceCounter {
    private ReferenceCounter reference;

    public static void main(String[] args) {
        ReferenceCounter r1 = new ReferenceCounter();
        ReferenceCounter r2 = new ReferenceCounter();
        r1.reference = r2;
        r2.reference = r1;
    }
}
```

针对以上代码，使用引用计数的方法时，r1和r2都存在对方的引用，造成计数器不为零而无法回收。而实际上，上述对象是应当被回收的。

可达性分析算法

该算法的主要思想是以一系列的‘GC Roots’对象为起点，不断向下搜索，搜索的路径形成引用链当一个对象没有任何引用链引用时，表明该对象不再被使用，可以进行垃圾回收。

可以作为"GC Roots"的对象如下：

- 虚拟机栈中引用的对象
- 方法区类静态属性引用的对象
- 方法区中常量引用的对象
- 本地方法栈中本地方法引用的对象

在Java虚拟机的主流实现中，都采用了该算法进行对象是否存活的判定。

垃圾收集算法

垃圾收集的方式有很多，各个平台虚拟机的实现方式也不仅相同，下面介绍几种算法以便了解。

标记-清除算法

该算法分为"标记"和"清除"两个阶段，第一阶段采用上面介绍的相关算法标记出需要进行回收的对象然后第二阶段将标记的对象进行回收。

该算法简单易懂，然而却存在以下问题：

- 标记和清除的过程效率不高
- 清除对象后，会造成内存产生大量的内存碎片，导致分配大对象时没有足够的连续内存而提前触发圾收集

复制算法

为了解决效率的问题，出现一种"复制"的算法。该算法将内存分为大小相同的两块，每次使用其中的块。当其中一块使用完时，将存活的对象复制到另外一块的一端，本块内存可以全部清理。这种算法免了内存碎片的产生，内存分配只需要顺序分配即可。缺点就是只能使用总内存的一半，代价太高。

标记-整理算法

为了避免标记-清除算法产生大量的内存碎片，该方法应运而生。该方法第一阶段也是对不再使用的象进行标记，第二阶段却不再直接进行垃圾收集，而是将所有存活的对象移向内存的一端，然后将剩的内存全部清除。这样就避免的不连续内存的产生，缺点就是需要进行大量的对象移动。

分代收集算法

该算法主要针对**Java堆**内存，将堆分为新生代和老年代。新生代中每次垃圾回收都会有大量的对象被收，而老年代中对象的存活率比较高。

老年代中的对象一般不会发生频繁的回收，所有可以使用**标记-清除算法**或者**标记-整理算法**进行垃圾收，而新生代对象变动比较频繁，可以使用**复制算法**进行垃圾回收。

在HotSpot虚拟机的实现中，对新生代的复制算法进行了优化。由于新生代中大量的对象存活时间短所以不需要将内存进行1:1划分来保证有足够的内存可以进行复制，而是将内存分为一块较大的区域Edn和两块较小的区域Survivor，默认比例是8:1:1。每次只使用Eden和一块Survivor空间，所以可用空是新生代总空间的90%。在进行垃圾回收时，将存活的对象复制到另外一块Survivor空间，然后将使过的Eden和Survivor空间进行清除。如果存活的对象较多导致Survivor空间无法容纳，那么这些对象接进入老年代。

内存分配

在多数情况下，对象在新生代的Eden空间进行分配，当Eden空间无法提供足够的内存时，虚拟机将起一次Minor GC(新生代垃圾回收，与之对应的Full/Major GC，表示发生在老年代的垃圾回收，通这种回收速度慢，伴随至少一次的Minor GC)。

虚拟机给每个对象定义了一个对象年龄计数器，如果对象在Eden空间创建，在每经过一次Minor GC没有被回收时，该计数器加一。当该计数器达到一定的值(默认15)，就会晋升到老年代。为了更好的用内存，如果Survivor空间中相同年龄的对象总和大于Survivor空间的一半，那么年龄大于等于该年的对象直接进入老年代。

当要创建需要连续内存空间的大对象时，如长字符串和大数组，为了避免Eden和Survivor空间之间发大量的内存复制，大对象可以之间分配到老年代。大对象所需内存可以通过参数 **-XX:PretenureSizehreshold** 设置。