JVM相关问题

-----GC

1. 什么是垃圾

没有任何引用指向的一个对象或者多个对象（指循环引用）

1. 定位垃圾

引用计数 ---循环引用存在问题

根可达算法 --根包括 虚拟机栈中的变量，本地方法栈中的变量，静态变量，常量池，加载的类信息

1. 垃圾回收算法

标记清除 –容易产生碎片

拷贝算法 –没有碎片，效率高，但是空间利用率低（一半浪费）

标记压缩 –没有碎片，需要移动内存，效率低

1. 分代垃圾回收算法

模型：新生代+老年代+永久代（1.7）/元数据(1.8)

元数据区主要放Class

永久代在堆内，必须指定大小，元数据在堆外，可不指定最大大小，受限于物理内存显示

字符串常量1.7放在永久代，1.8放在堆

新生代年龄进入老年代 G1 15 CMS 6

老年代满🡪FULLGC

５.　常见垃圾回收器

１.Serial年轻代 Serial老年代

2. PS年轻代 ParNew年轻代（配合CMS老年代）

3.ParallelOld老年代

4． CMS老年代 （低延迟）

5.G1 ZGC

1.8默认 PS+ParallelOld(高吞吐量)

6. JVM参数

-开头 所有HotSpot都支持

-X开头 特定版本支持

-XX 开头 下个版本可能取消

-XX:+PrintCommandLineFlags 打印设置的JVM参数

-XX:+PrintFlagsFinal 打印最终参数值

-------CMS回收器详解

1. 主要阶段

**初始标记**

标记GC Roots可达的老年代对象

遍历新生代对象，标记可达的老年代对象

STW 但是速度快

**并发标记**

继续递归标记这些对象可达的对象

应用正常执行

如果应用执行过程中，老年代新产生对象等，标记为dirty，待重新标记时处理

**重新标记**

老年代的新对象被GC Roots引用

老年代的未标记对象被新生代对象引用

老年代已标记的对象增加新引用指向老年代其它对象

新生代对象指向老年代引用被删除

STW

**并发整理**

回收老年代，标记清除算法

1. 优点

速度快，STW时间少

缺点

CPU敏感（多线程）

不压缩，会有碎片

整理阶段不STW，这段时间如果产生新的垃圾，下一次才会被回收。如果此时OLD区不够用，触发Serial olggc，会长时间STW。 所以不能等到old区100%才gc，需要设立阈值。Jdk1.8默认92%，推荐70%左右。

-------G1回收器详解

Jdk1.9开始默认

模型

将堆分为等大的区域（Eden,Suvivor,Old,Humongous）

默认分成2048个区，也可以手动指定大小 –XX:G1HeapRegionSize

如果一个对象大小超过一个区的一般，则分配到H区



重要机制

Card Table 和Remember Set

每个区分成多个card,一般512byte

当某个Card里面的对象引用了其他区的对象(内存空间发生改变)，则这个Card标记为dirty，被引用对象的RS里面记录下这个card的信息，方便向上查找

RS的更新不是实时的

在赋值操作时，会有一个write barrier，将更新rs的任务放到一个队列里面，  
队列有白绿黄红四个状态，表示积压情况



Gc种类

YGC

MixedGC

FullGC

Young GC 阶段：

STW

阶段1：根扫描

静态和本地对象被扫描

阶段2：更新RS

处理dirty card队列更新RS

阶段3：处理RS

检测从年轻代指向年老代的对象

阶段4：对象拷贝

拷贝存活的对象到survivor/old区域

阶段5：处理引用队列（对象地址变化了，引用值也要跟着变）

软引用，弱引用，虚引用处理

拷贝算法

G1会记录每个阶段的耗时情况，由于调优

 -XX:MaxGCPauseMillis 设置每次gc的期望耗时

如果超过期望，会减少每次gc的region数量（增加gc频率，需要合理设置，否则影响吞吐量）

Mixed GC

当老年代使用量达到一定比例（-XX:InitiatingHeapOccupancyPercent=N，默认45%）

Mixed GC 选定所有年轻代里的 Region，外加根据 global concurrent marking 统计得出收集收益高的若干老年代 Region 进行回收。（gabage first）

STW,进行YoungGC (同时进行初始标记)

恢复应用线程，并发标记（三色标记算法）

STW，重新标记

立即清空全部是垃圾的区域

部分为空的，拷贝的新的region，回收原来的(STW)

恢复应用线程

Full GC：

收集整个堆。

触发时机：对象内存分配速度过快，Mixed GC 来不及回收，老年代被填满。

这时候会切换到 G1 之外的 Serial Old GC 来收集整个 GC heap（注意，包括 young、old、perm），这才是真正的 Full GC。

作者：helly

链接：https://hacpai.com/article/1499506308788

来源：黑客派

协议：CC BY-SA 4.0 https://creativecommons.org/licenses/by-sa/4.0/