CMS

Concurrent Mark Sweep

马士兵

cms 简介

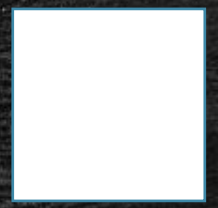
- 1.4版本后期引入
- 1.5 1.6开始流行
- 1.7引入G1
- 1.8如果内存比较大推荐G1

并发标记算法

难点：在标记对象过程中，对象引用关系正在发生改变

你妈妈在标记屋里面垃圾的同时
你跟你弟正在产生新垃圾
或者从你妈妈手里把垃圾夺回来

三色标记法



白色：未被标记的对象

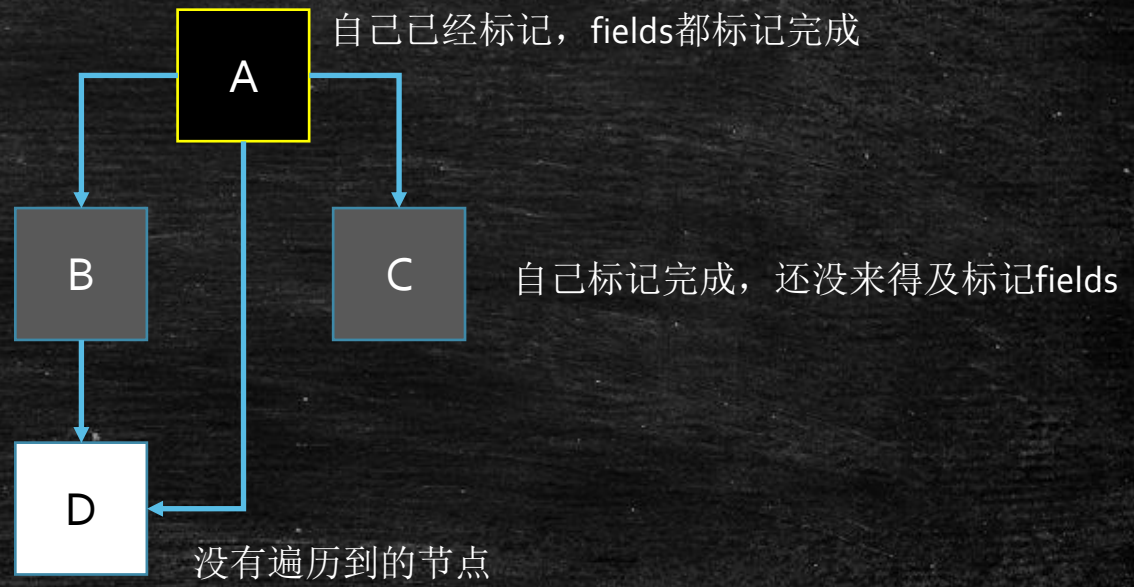
灰色：自身被标记，成员变量未被标记

黑色：自身和成员变量均已标记完成

漏标

在remark过程中，黑色指向了白色，
如果不对黑色重新扫描，则会漏标
会把白色D对象当做没有新引用指向从而回收掉

并发标记过程中，Mutator删除了所有从灰色到白色的引用，会产生漏标
此时白色对象应该被回收



漏标是指，本来是live object，但是由于没有遍历到，被当成garbage回收掉了

产生漏标：

1. 标记进行时增加了一个黑到白的引用，如果不重新对黑色进行处理，则会漏标

2. 标记进行时删除了灰对象到白对象的引用，那么这个白对象有可能被漏标

打破上述两个条件之一即可

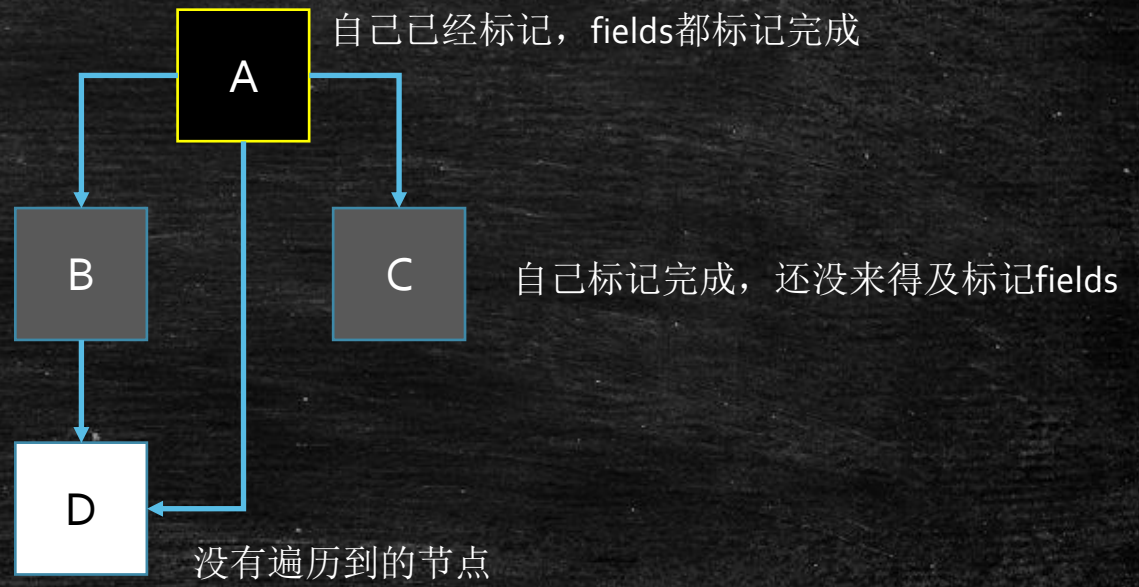
1. incremental update -- 增量更新，关注引用的增加，
   把黑色重新标记为灰色，下次重新扫描属性，CMS使用
2. SATB snapshot at the beginning – 关注引用的删除
   当B->D消失时，要把这个引用推到GC的堆栈，保证D还能被GC扫描到
   G1使用

为什么G1用SATB?

灰色 → 白色 引用消失时，如果没有黑色指向白色
引用会被push到堆栈
下次扫描时拿到这个引用，由于有RSet的存在，不需要扫描整个堆去查找指向白色的引用，效率比较高
SATB 配合 RSet，浑然天成

并发标记的算法

- 三色扫描算法
  - 白 灰 黑
- 在并发按标记时，引用可能产生变化，白色对象有可能被错误回收或者漏标
- 解决方案
  - SATB
    - snapshot at the beginning
    - 在起始的时候做一个快照
    - 当B->D消失时，要把这个引用推到GC的堆栈，保证D还能被GC扫描到
  - Incremental Update
    - 当一个白色对象被一个黑色对象引用
    - 将黑色对象重新标记为灰色，让collector重新扫描
  - CMS采用Incremental Update
  - G1采用SATB（为了配合G1的RSet, 效率更高）



参考资料

https://blogs.oracle.com/jonthecollector/our-collectors