链滴

go 垃圾回收学习的总结

作者：xhaoxiong
原文链接：https://ld246.com/article/1533804620735
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

按照官方的说法，GO GC的基本特征是"非分代、非紧缩、写屏障、并发标记清理"

什么时候启动垃圾回收?

启动过早则浪费cpu,过晚则导致堆内存恶性膨胀。

所有问题的核心：抑制堆增长，充分利用CPU资源。

解决措施：

基于go1.5的垃圾回收

三色标记和写屏障

这是标记和用户代码并发的基本保障，基本原理：

1. 起初所有对象都是白色
2. 扫描所有可达对象，标记为灰色，放入待处理队列
3. 从队列中取出灰色对象，将其引用对象标记为灰色放入队列，自身标记为黑色
4. 写屏障监视对象内存修改，重新标记色或放回队列

什么是屏障???

控制器

控制器全程参与并发回收任务，记录相关状态数据，动态调整运行策略，影响并发标记单元的工作模和数量，平衡CPU资源占用，当回收结束的时候，参与next_gc回收阈值设置，调整垃圾回收出发频率

辅助回收

某些时候，对象分配速度可能快于后台标记，

这时候会引发一系列恶果，堆恶心扩张，垃圾

回收永远无法完成。此时用户代码线程参与后台回收标记就很有必要。在为对象分配堆内存的时候通相关策略去执行一定限度的回收操作，平衡分配和回收操作，让进程处于良性状态。

初始化->启动->标记->清理->监控

初始化

设置gcpercent

原文链接：go 垃圾回收学习的总结

(GOGC --- 新分配内存和上次GC回收后剩下的实时数据比，默认值为100,通过runtime/debug 这个内的SetGCPercent方法设置)

和

next_gc的阈值

```
func gcinit() {
// 并发执行器
work.markfor=parforalloc(_MaxGcproc)

// 设置GOGC
_ =setGCPercent(readgogc())

// 初始启动阈值（4MB）
memstats.next_gc=heapminimum
}

func readgogc()int32{
p:=gogetenv("GOGC")
if p== "" {
    return 100
}
if p== "off" {
    return-1
}
return int32(atoi(p))
}

func setGCPercent(in int32) (out int32) {
out=gcpercent
if in<0{
    in= -1
}
gcpercent=in
heapminimum=defaultHeapMinimum*uint64(gcpercent) /100
return out
}
```

引用

启动

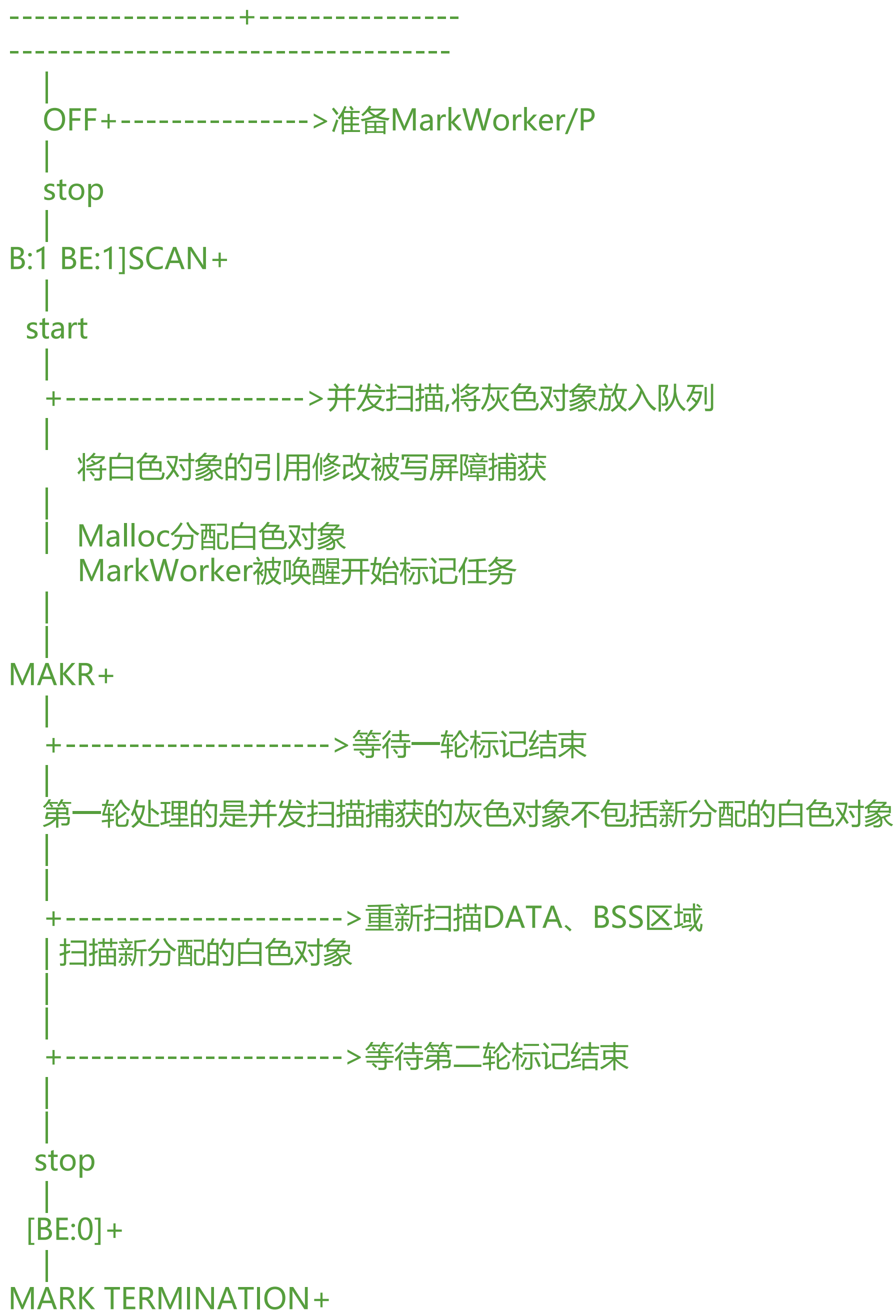
在为对象分配堆内存后，mallocgc函数会检查
垃圾回收出发条件，并依照相关状态启动或参与辅助回收。

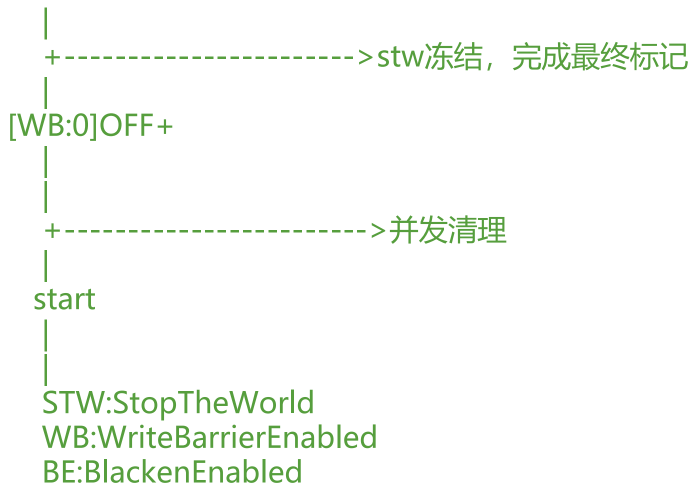
```
分配黑色对象(gcmarknewobject)
    |
    |
```

```
检查垃圾回收触发条件(shouldhelpgc&&gcTrigger)
    |
    |
启动并发垃圾回收(gcStart)
    |
    |
辅助参与回收任务(gcAssistAlloc)
```

垃圾回收默认以全并发模式运行，但可以用环境变量或参数禁用并发标记和并发清理。GC groutine 直循环，直到出发符合条件时被唤醒

并发模式垃圾回收过程示意图





标记

并发标记分为

1. 扫描：遍历相关内存区域，依照指针标记找出灰色可达对象，加入队列
2. 标记：将灰色对象从队列中取出，将其引用对象标记为灰色，自身标记为黑色

扫描函数gcscan_m启动时，用户代码和MarkWorker都在运行。扫描函数仅使用了当前线程，并未用并发方式执行,扫描目标包括多个ROOT区域，还有全部goroutine栈

并发标记由多个MarkWorker goroutine共同完成，它们在回收任务开始前被绑定到P，然后进入休状态，直到被调度器唤醒

MarkWorker有3种工作模式。

•gcMarkWorkerDedicatedMode：全力运行，直到并发标记任务结束。

•gcMarkWorkerFractionalMode：参与标记任务，但可被抢占和调度。

•gcMarkWorkerIdleMode：仅在空闲时参与标记任务。

清理

与复杂的标记过程不同，清理操作要简单得多。此时，所有未被标记的白色对象都不在被引用，可简的将其内存回收。

并发清理本质上就是一个死循环，被唤醒后开始执行清理任务。通过遍历所有span对象，触发内存分器的回收操作。任务完成后，再次休眠，等待下次任务

监控

垃圾回收器最后的一道保险措施。监控服务sysmon每隔2分钟就会检查一次垃圾回收状态，如超出2钟未曾触发，那就强制执行。

以上大体一些东西总结自《go语言学习笔记·雨痕》

Go语言中文网

yiz96

原文链接：go 垃圾回收学习的总结