

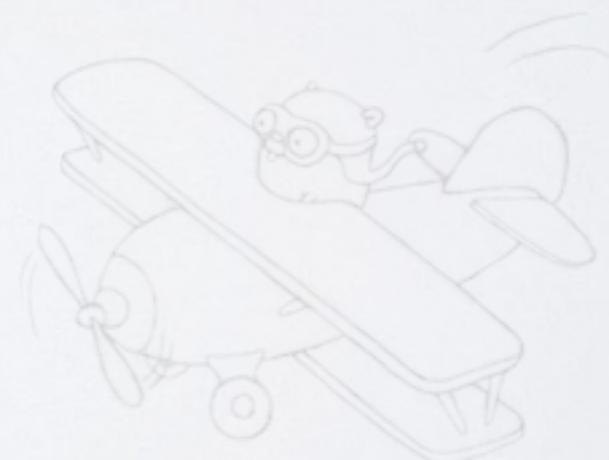


# Go 1.4 runtime

Gopher China 2015



1. Memory Allocator
2. Garbage Collector
3. Goroutine Scheduler





# 1. Memory Allocator

内存分配器





# base on tcmalloc.

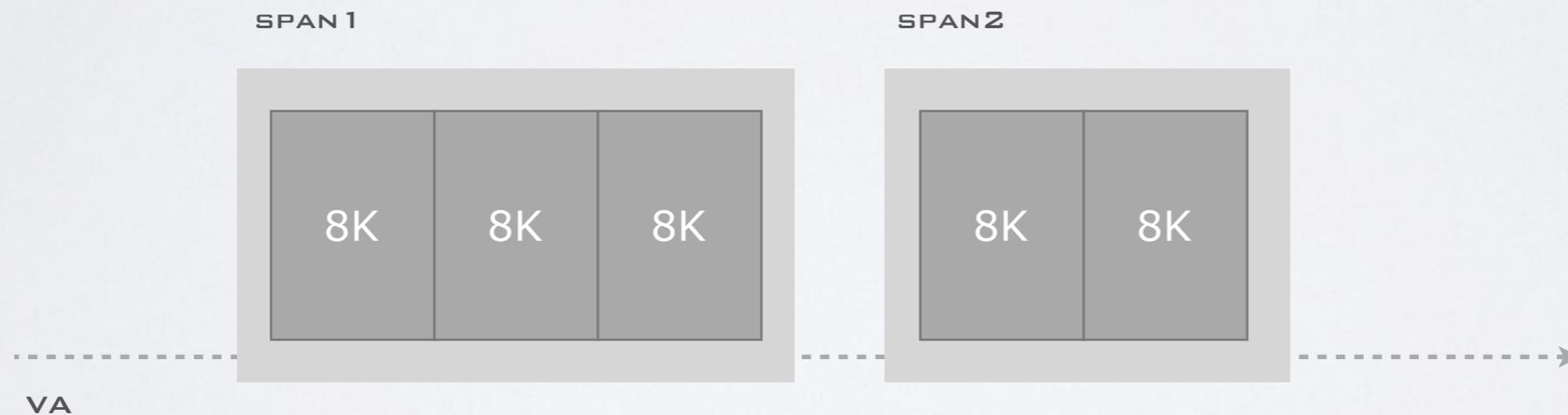
基于成熟方案，性能优秀。随着版本升级，针对性改进，以期与垃圾回收器更好协作。

核心：自主管理，缓存复用，无锁分配。



# page, span.

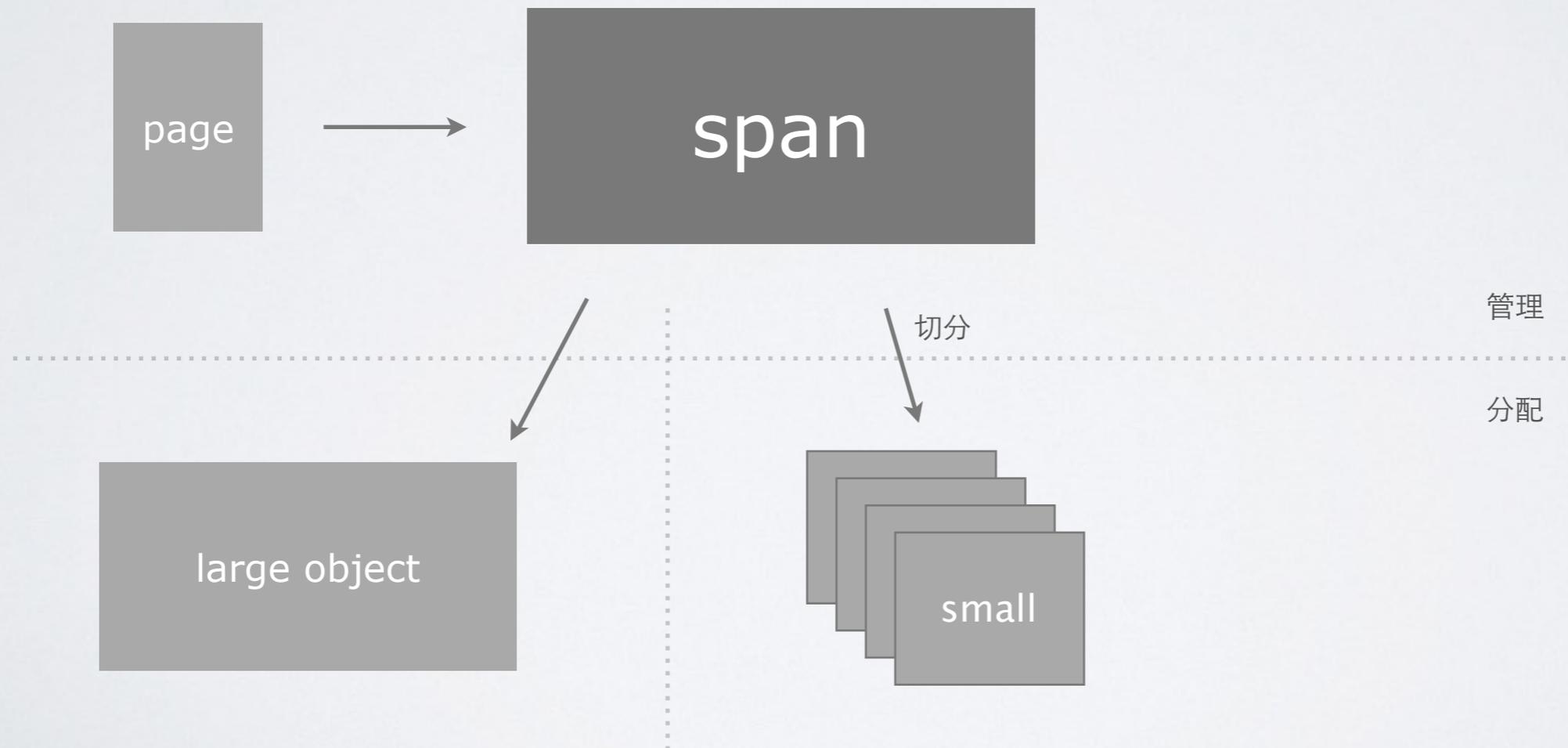
内存管理以页为基本单位，多个地址连续页构成内存块。





# small, large.

以 32KB 为界，将对象分成大小两类。

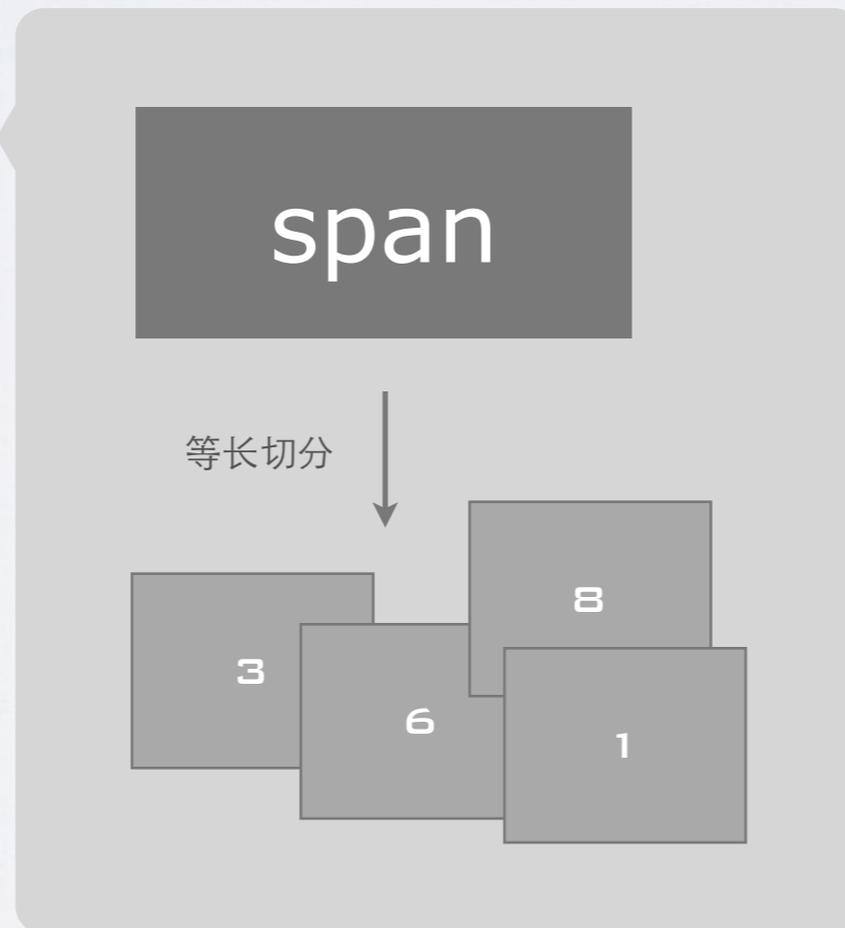




# size class.

按 8 倍数，将小对象分成多种大小规格。

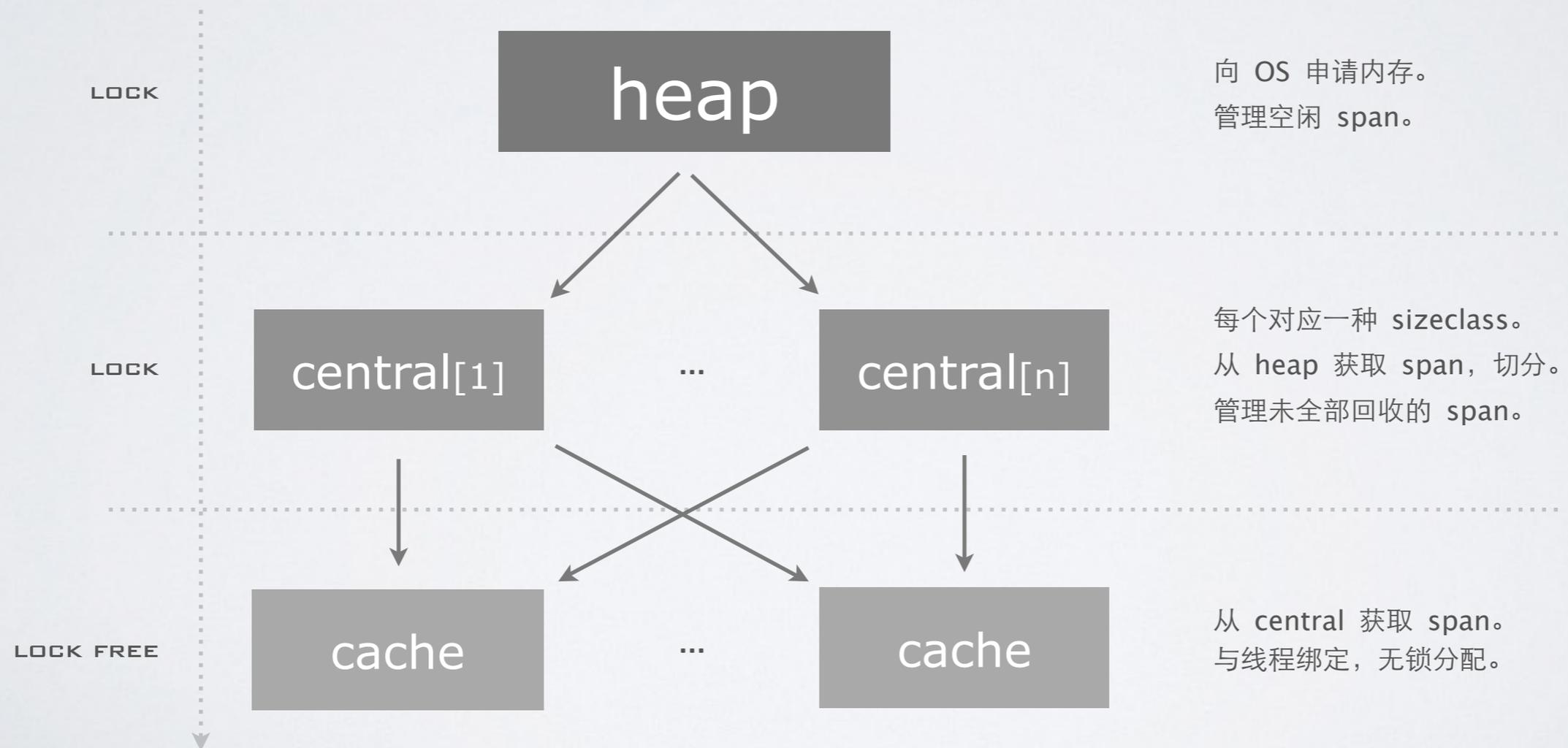
[1]	8
[2]	16
...	...
CLASS	SIZE





# heap, central, cache.

三级管理机构。





# init.

算法依赖连续地址，预留较大地址空间。



按页保存 span 指针。

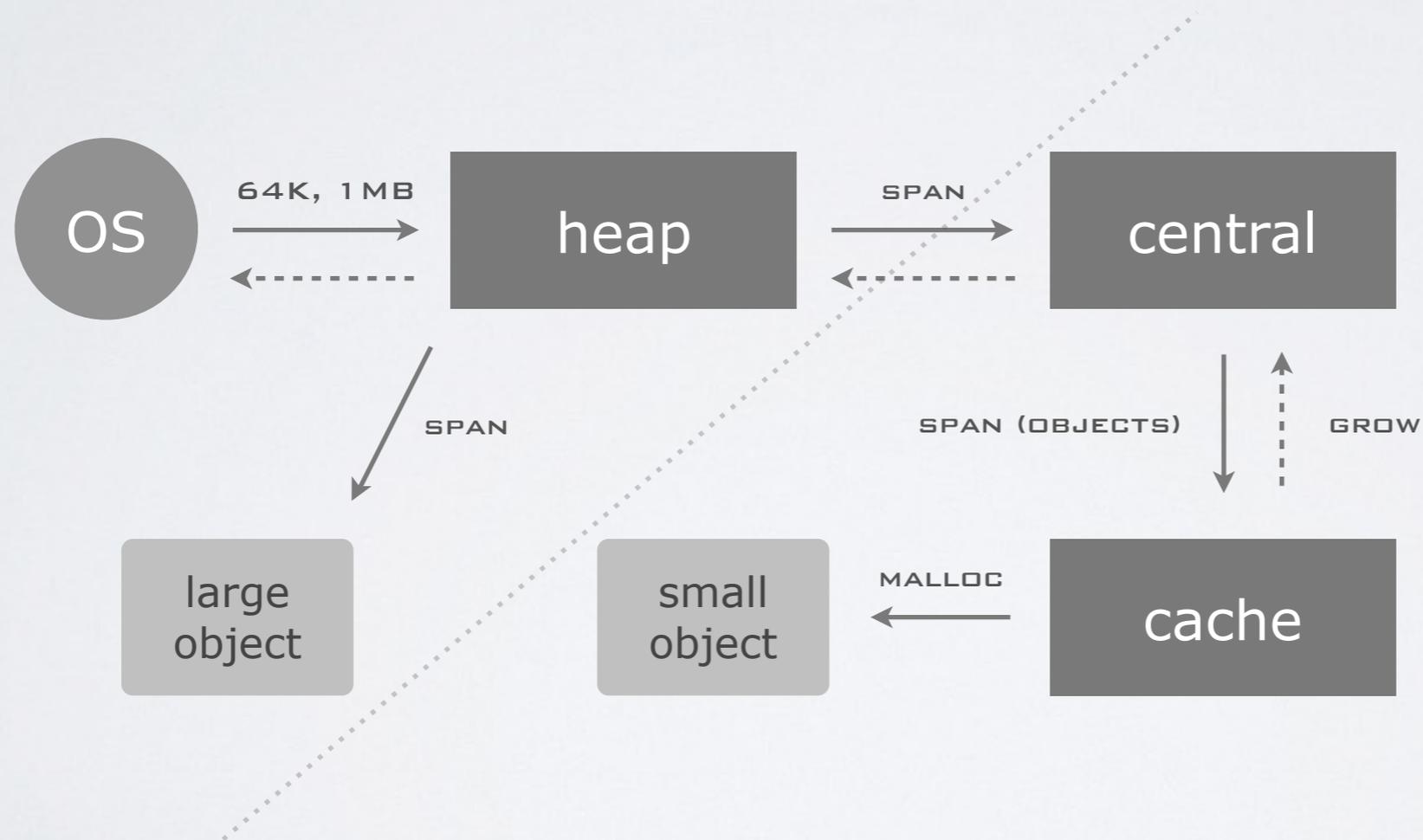
反查 object 所属 span。

检查相邻 span 是否可合并。



# malloc.

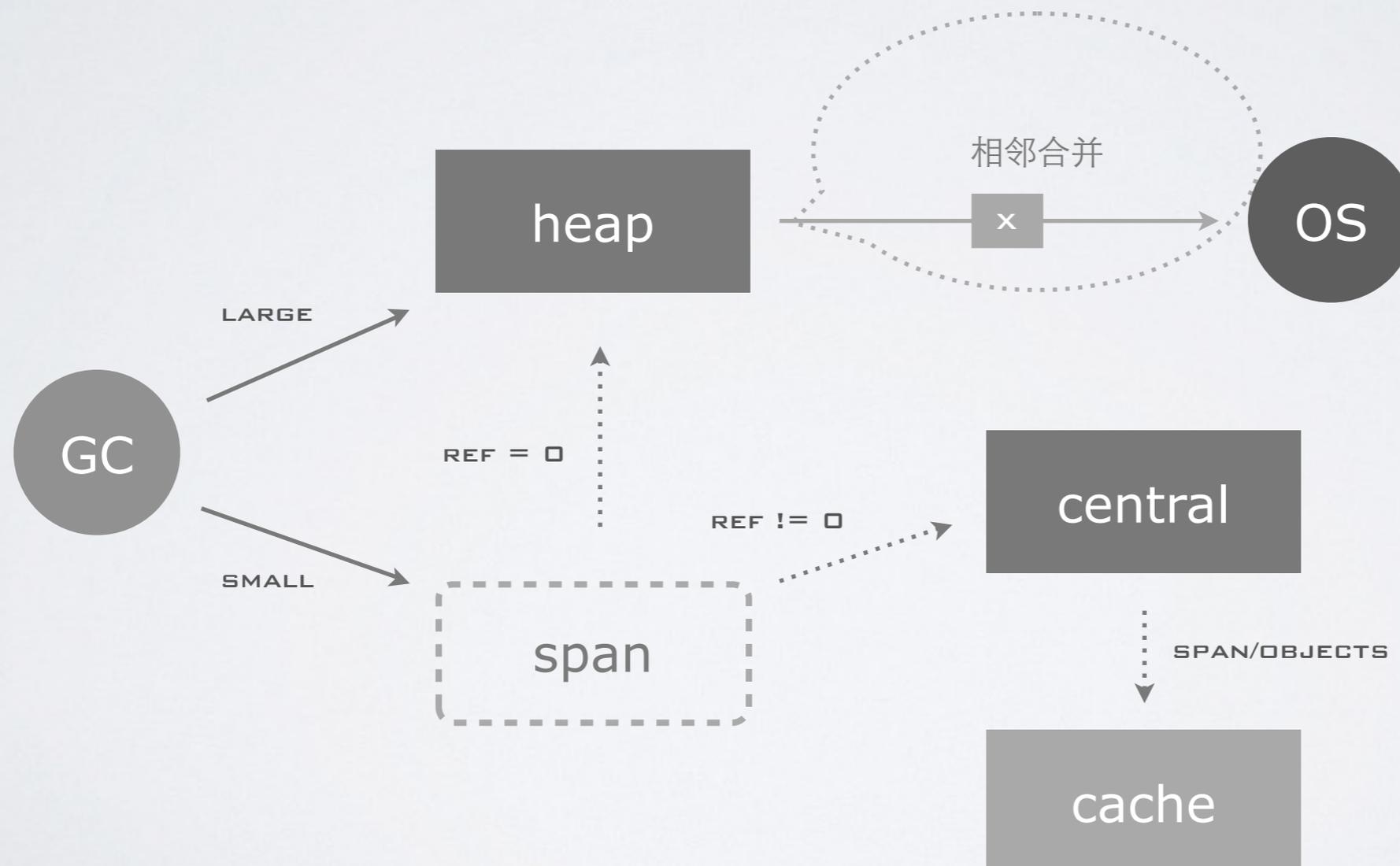
快速分配，按需扩张。





# sweep.

垃圾回收器引发回收操作。



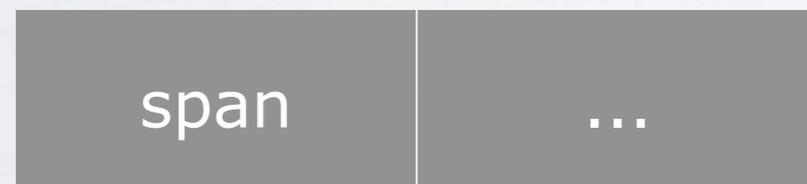
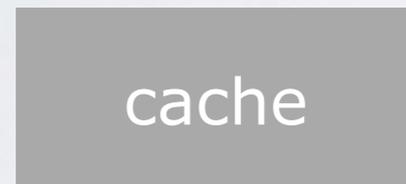


# fixalloc.

为管理对象分配内存，不占用预留地址。



span: 管理内存块的元数据。



allspans: 垃圾回收遍历。



## 2. Garbage Collector

垃圾回收器





# gc.

阈值触发，并行标记，并发清理。

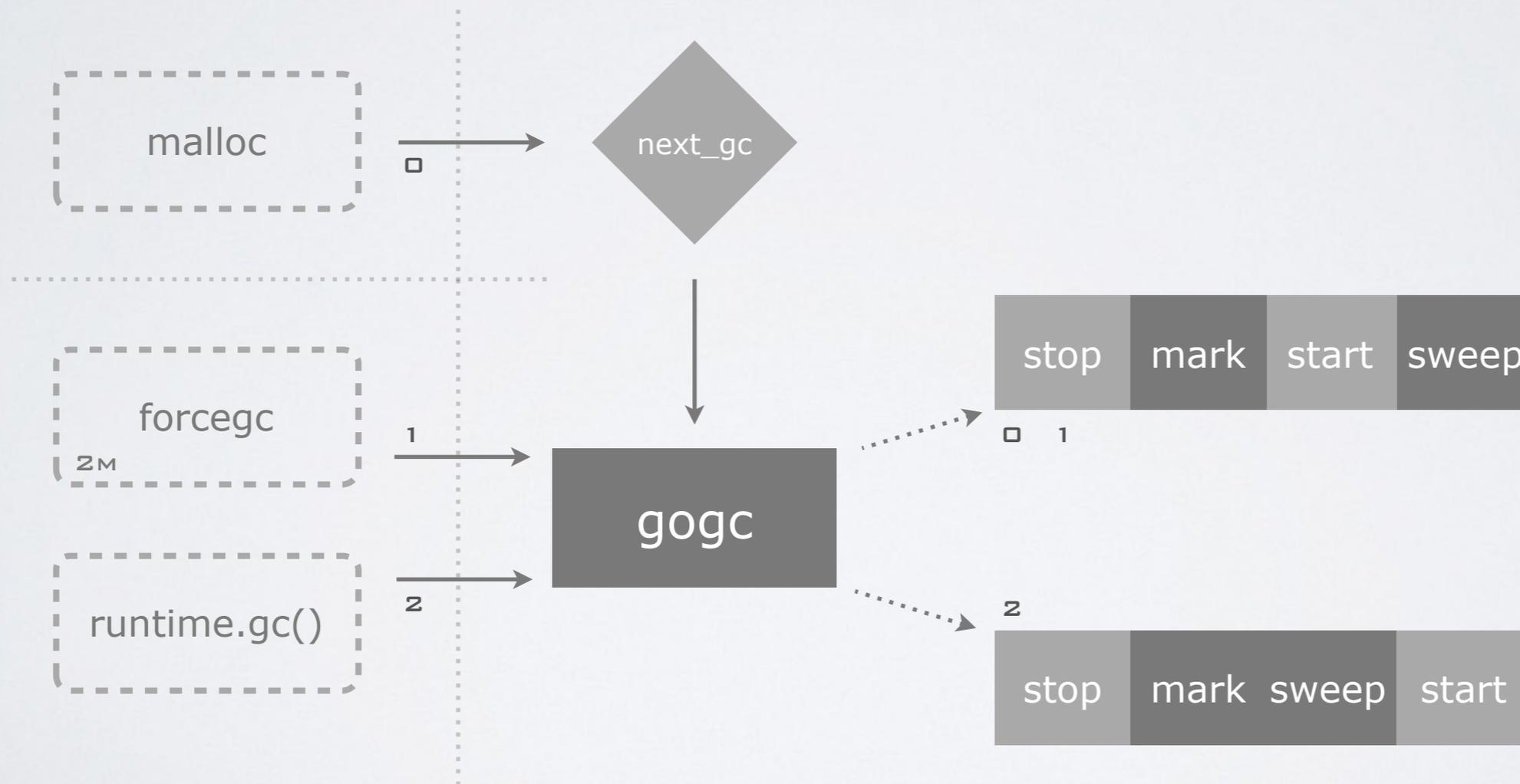
定期强制回收，释放物理内存。

版本升级，垃圾回收效率总是核心问题。



# gogc.

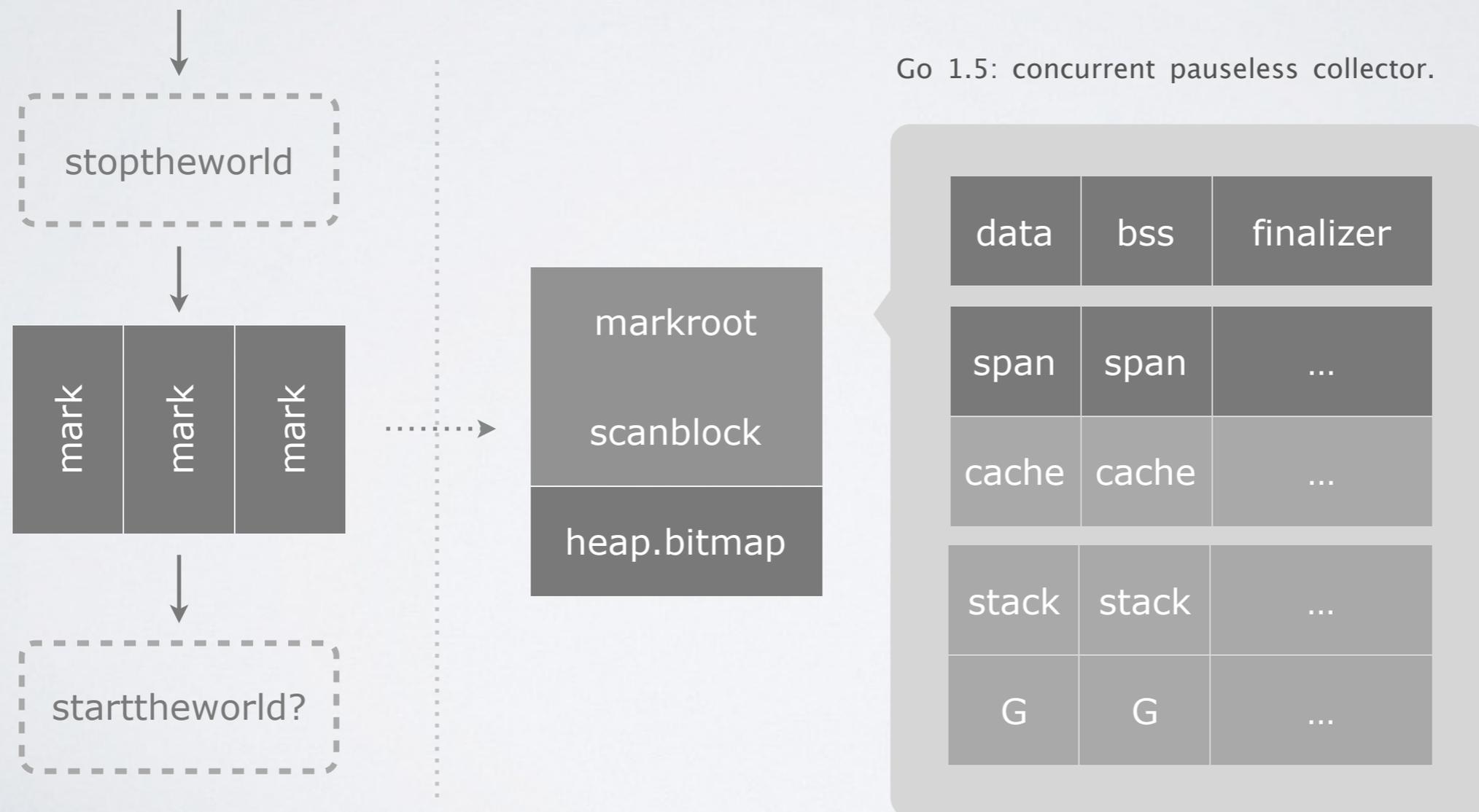
閾值检查，或強制回收。





# mark.

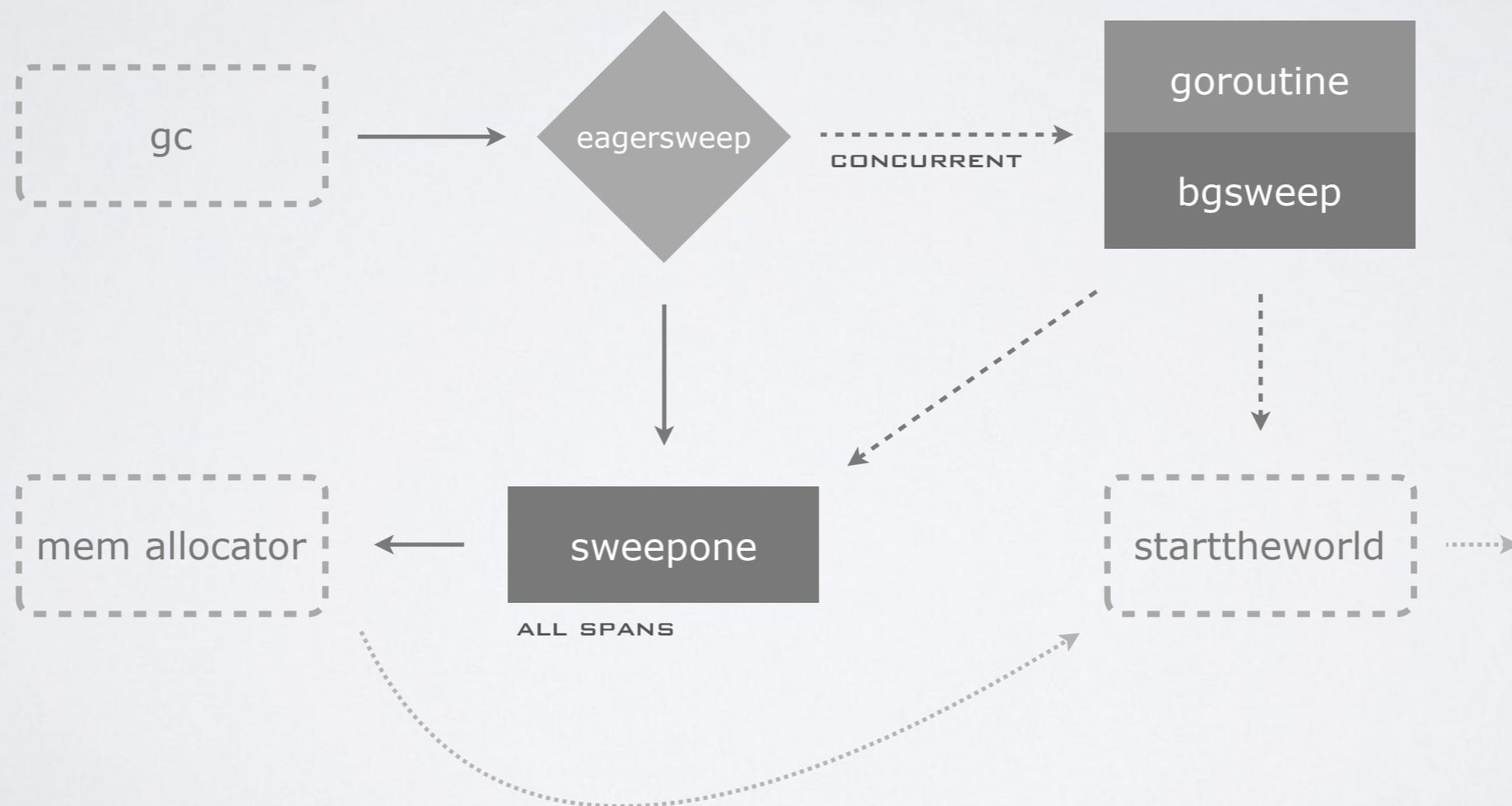
暂停用户逻辑，并行标记。





# sweep.

串行，或与用户逻辑并发执行。





# sysmon.

如阈值过大，可能会导致长时间无法触发垃圾回收。因此，每 2 分钟强制检查回收是非常必要的。

每 5 分钟，释放堆中长时间闲置块的物理内存。

在系统初始化时，使用专门的线程在后台运行监控循环。



# madvise.

在类 *UNIX* 系统，通过建议操作系统内核解除内存映射的方式释放物理内存，但不回收虚拟内存。

再次使用时，因缺页异常，由内核重新分配物理内存。

Microsoft Windows 系统不支持 `madvise`。



# 3. Goroutine Scheduler

并发调度器





# goroutine.

轻量级实现，支持创建成千上万并发任务。

线程多路复用。

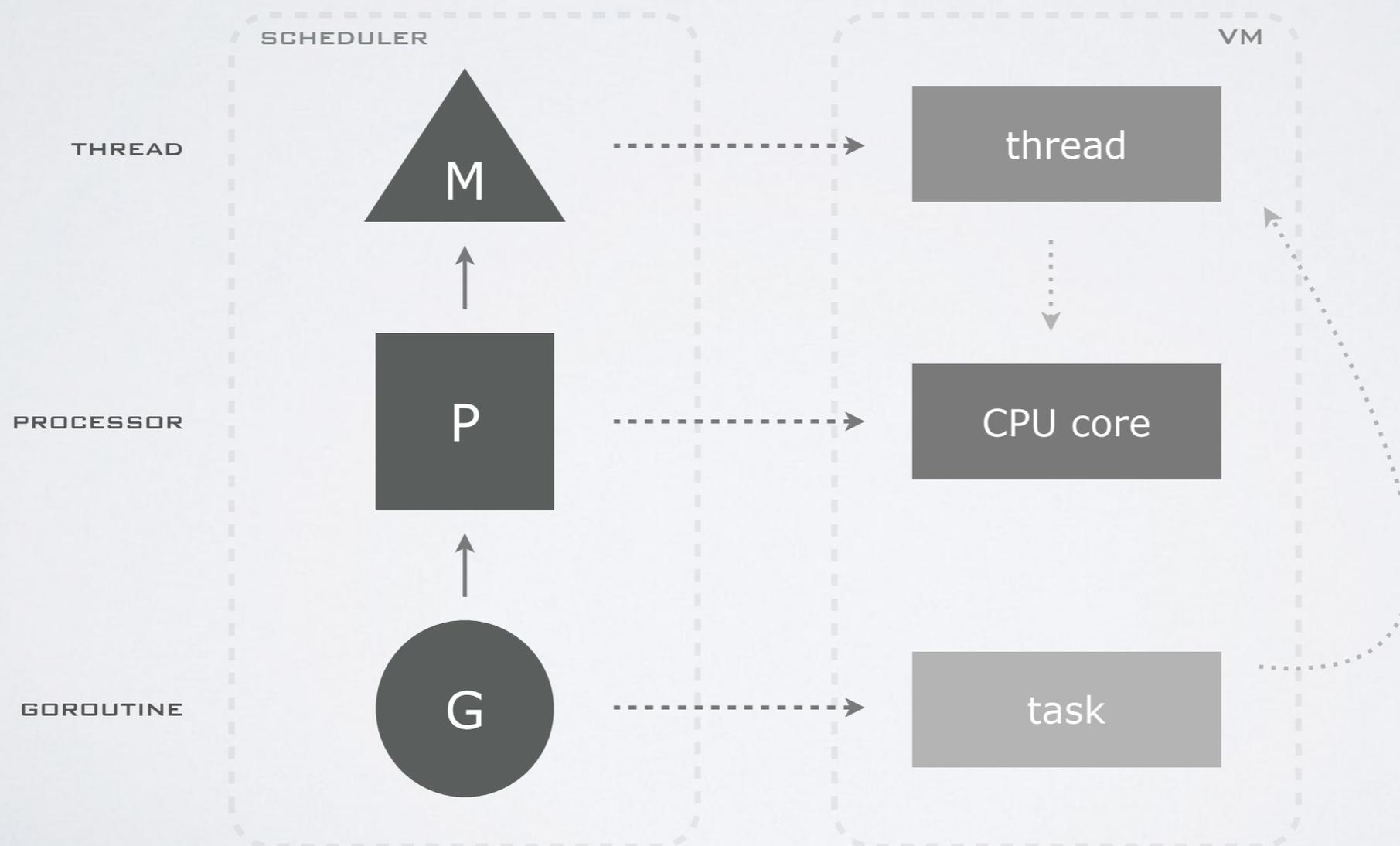
极小自定义初始栈。

任务在多个线程间切换。



# scheduler.

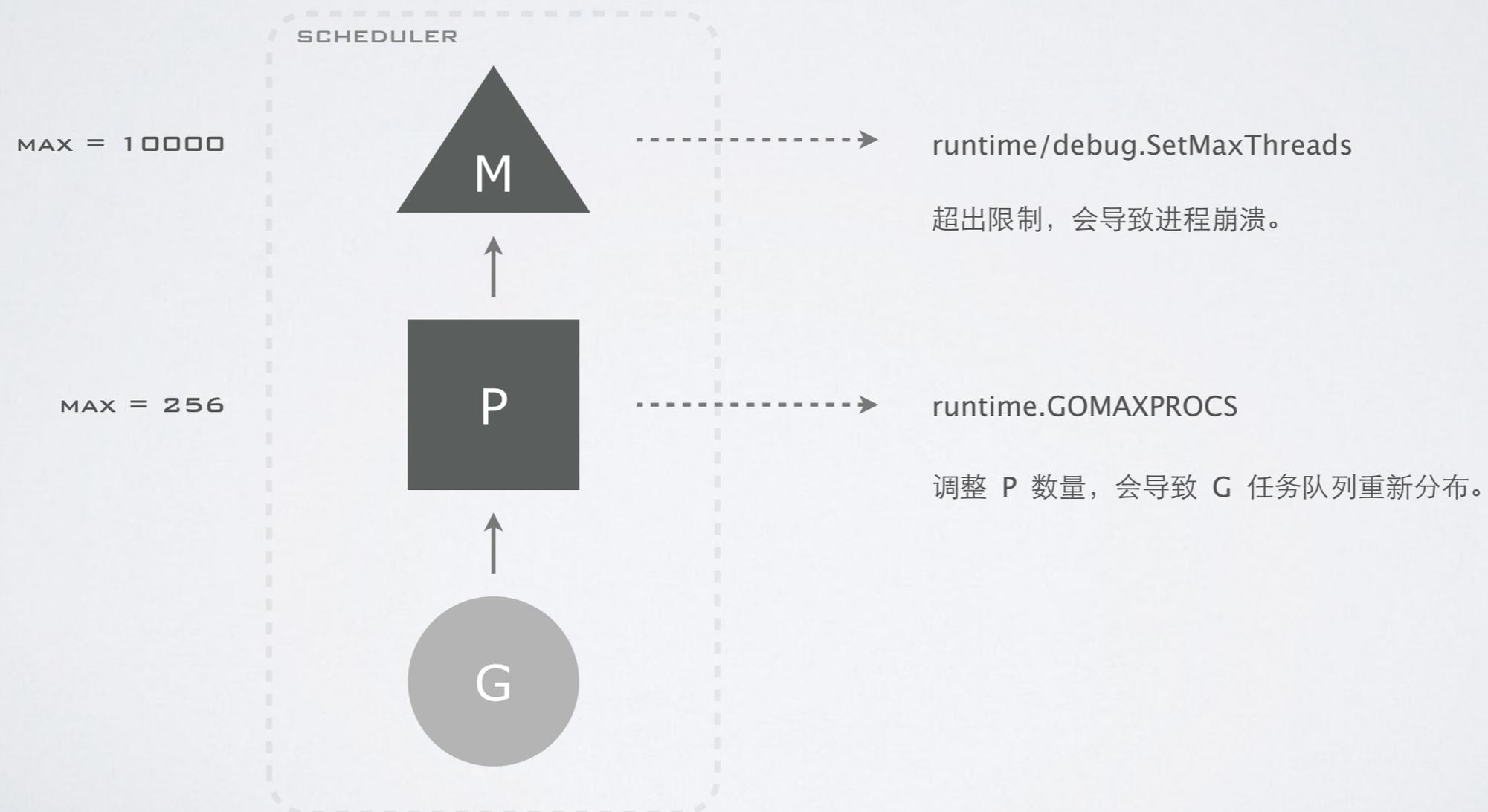
三种抽象模型协作。





# max.

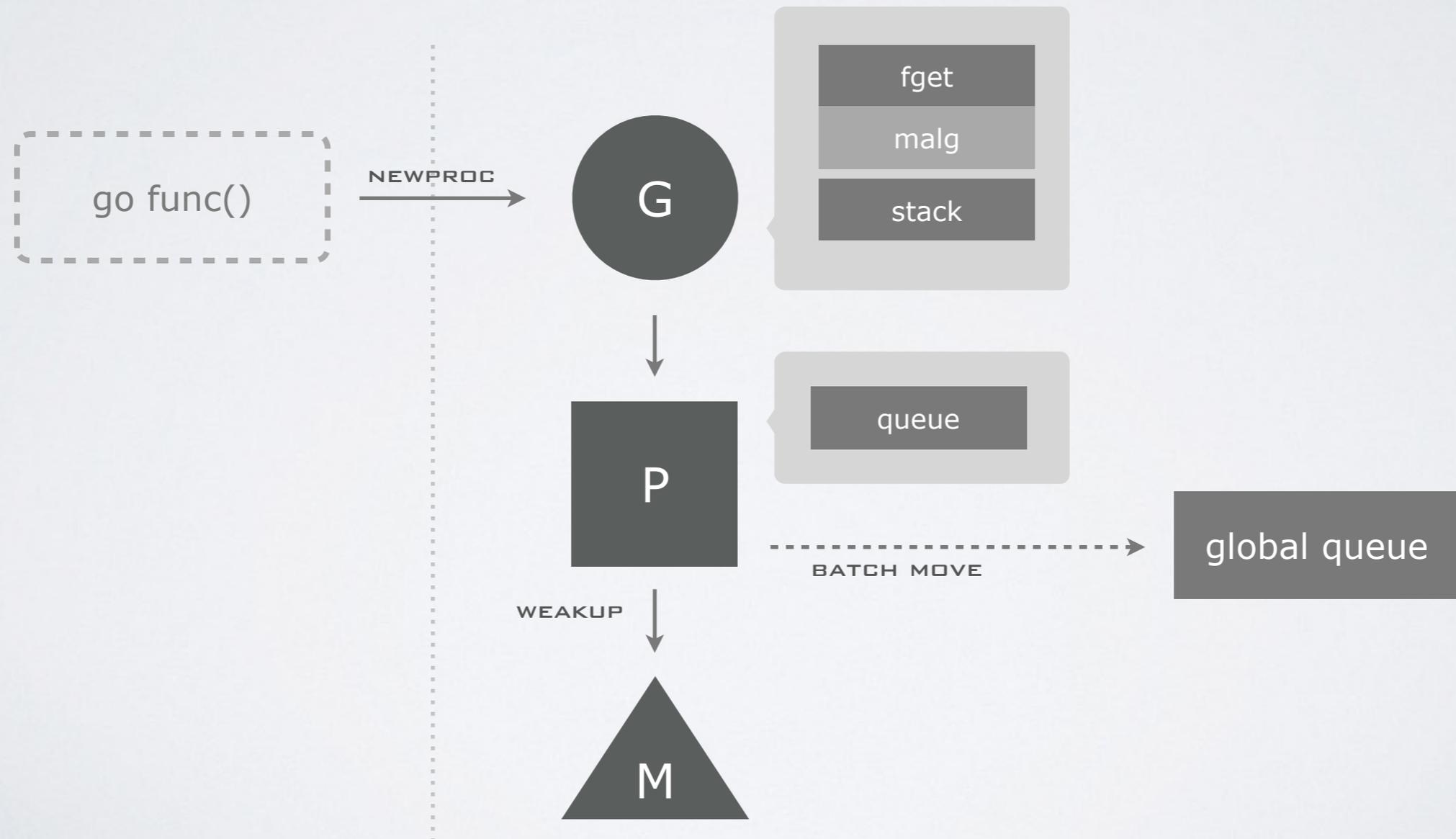
系统限制，允许调整。





# newproc.

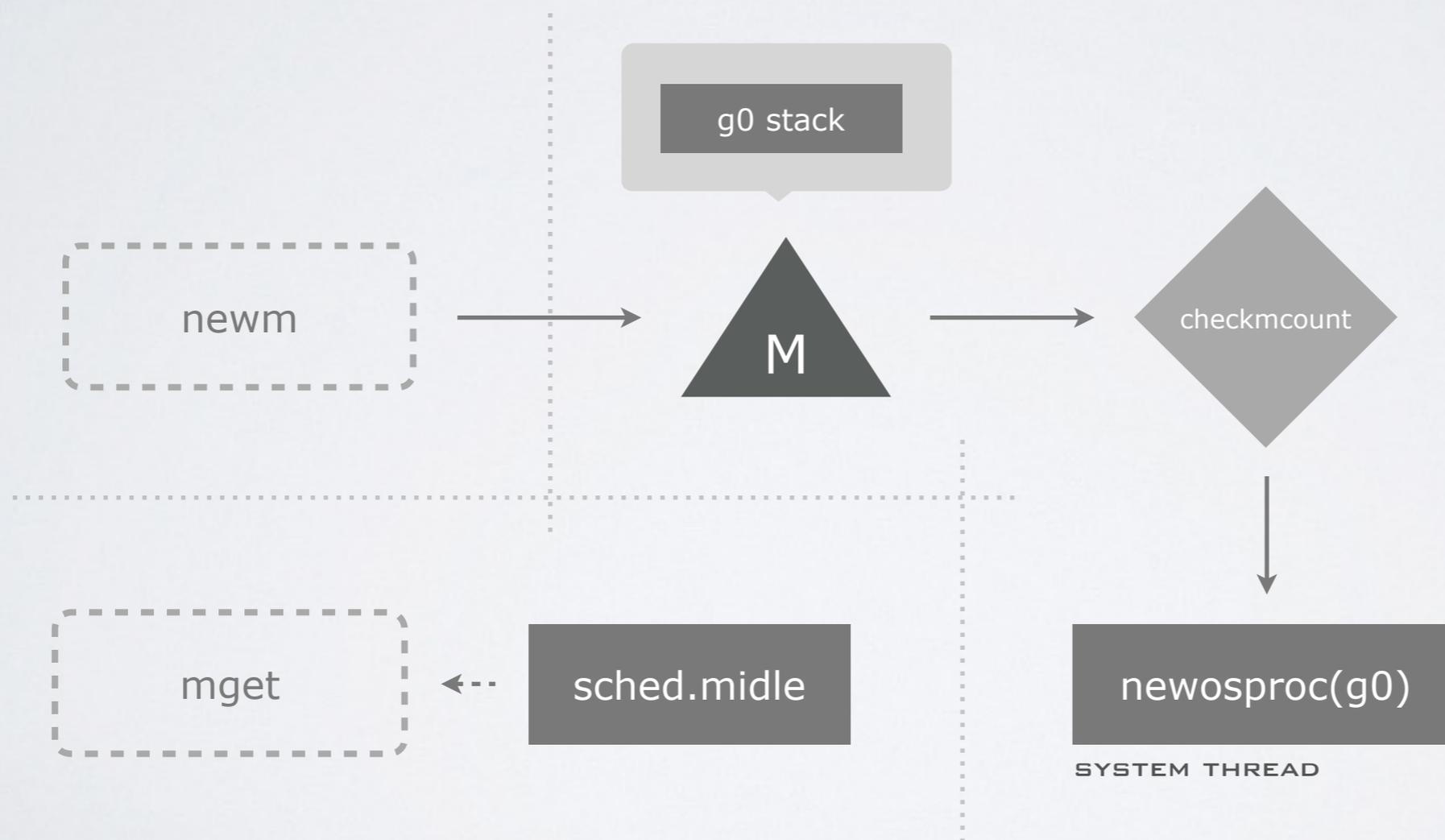
创建新并发任务。





# newm.

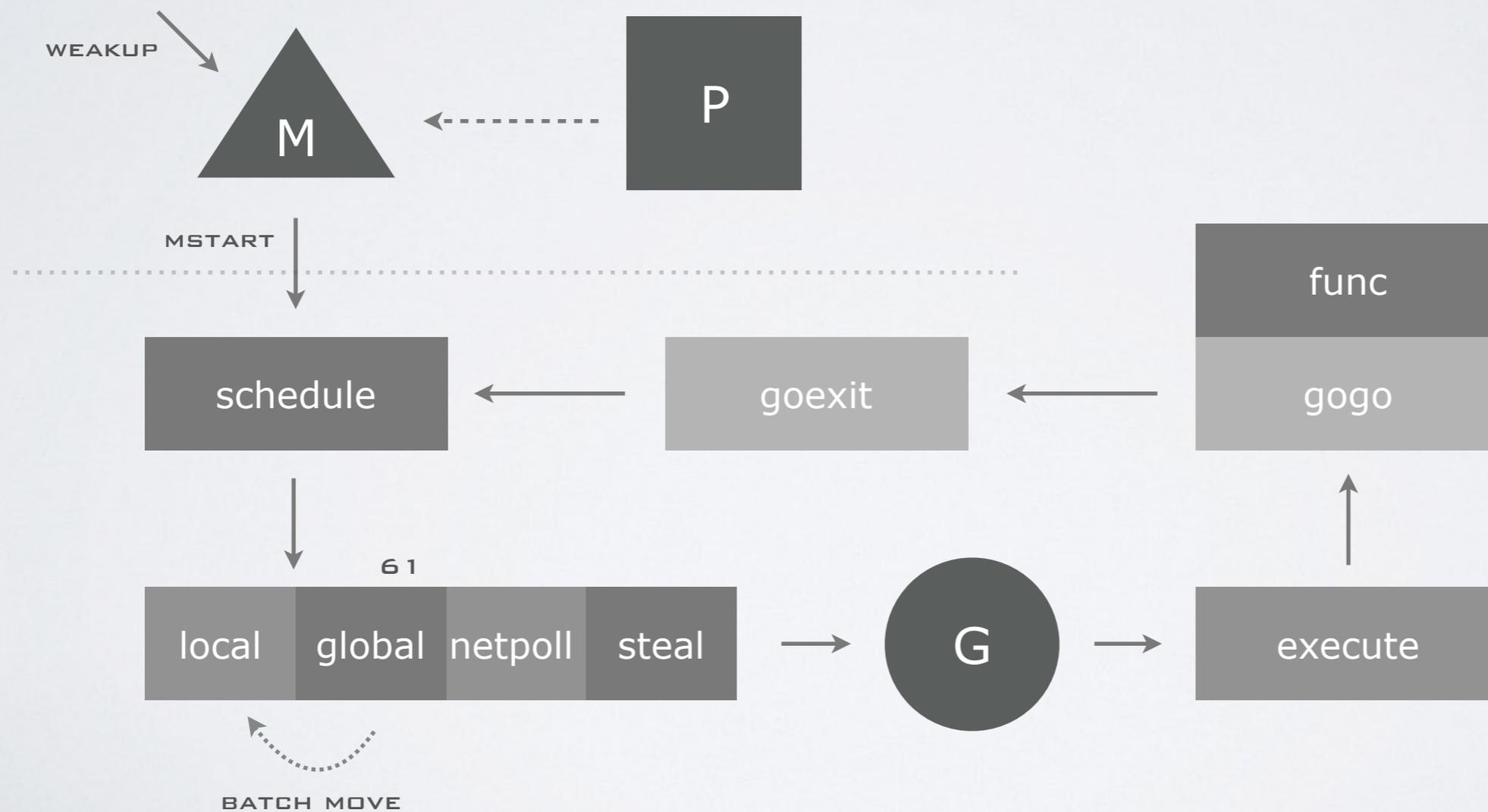
创建系统线程执行任务。





# execute.

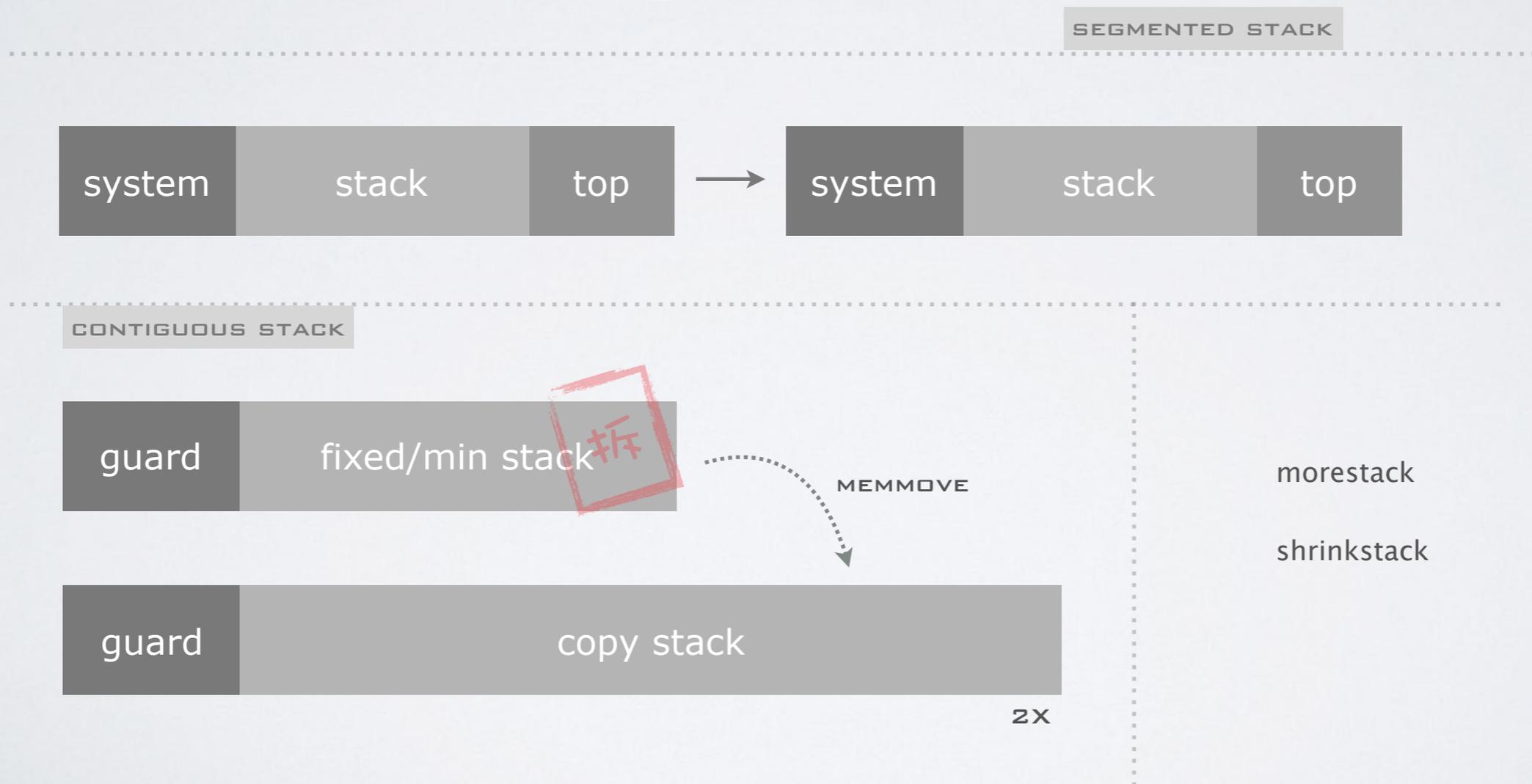
并发任务调度执行。





# copystack.

连续栈替代分段栈。





## 4. channel, defer...

请参考拙作《学习笔记》

<https://github.com/qyuhēn/book>

谢谢！

[qyuhenslack.com](https://qyuhenslack.com)

[qyuhen@hotmail.com](mailto:qyuhen@hotmail.com)

