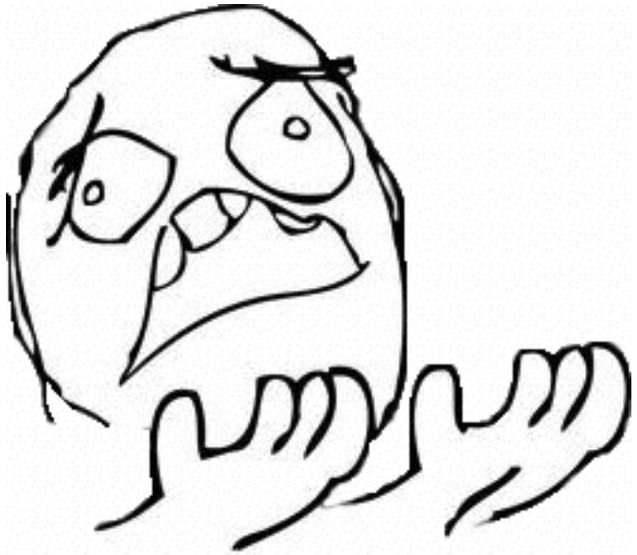
Go语言实现中的几个研究课题

毛康力
@舜飞科技

关于我

- Go发布1.0时开始关注
- 最早接触是在2012年
- 喜欢研究底层实现
- http://www.zenlife.tk/

有同事称我大神或大师(湿?)

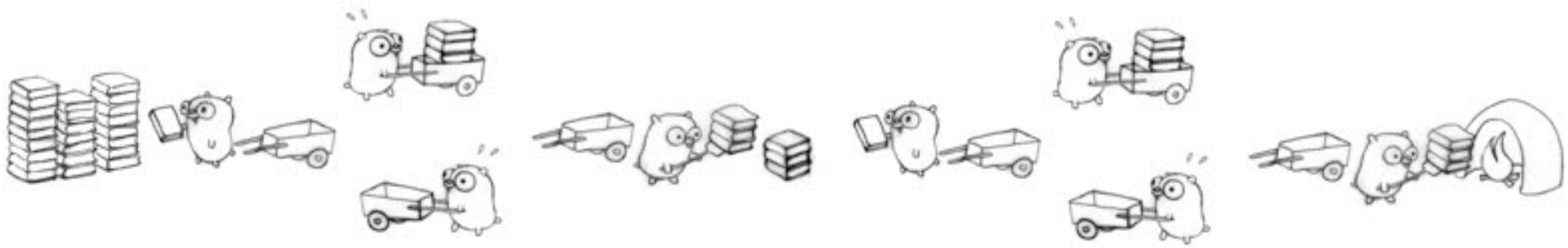


其实，我只是一个研究僧

大纲

- 并发
- 接口
- 垃圾回收
- 调度
- 死锁检测

并发



并发

- goroutine提供轻量的并发机制
- channel用于goroutine之间通信

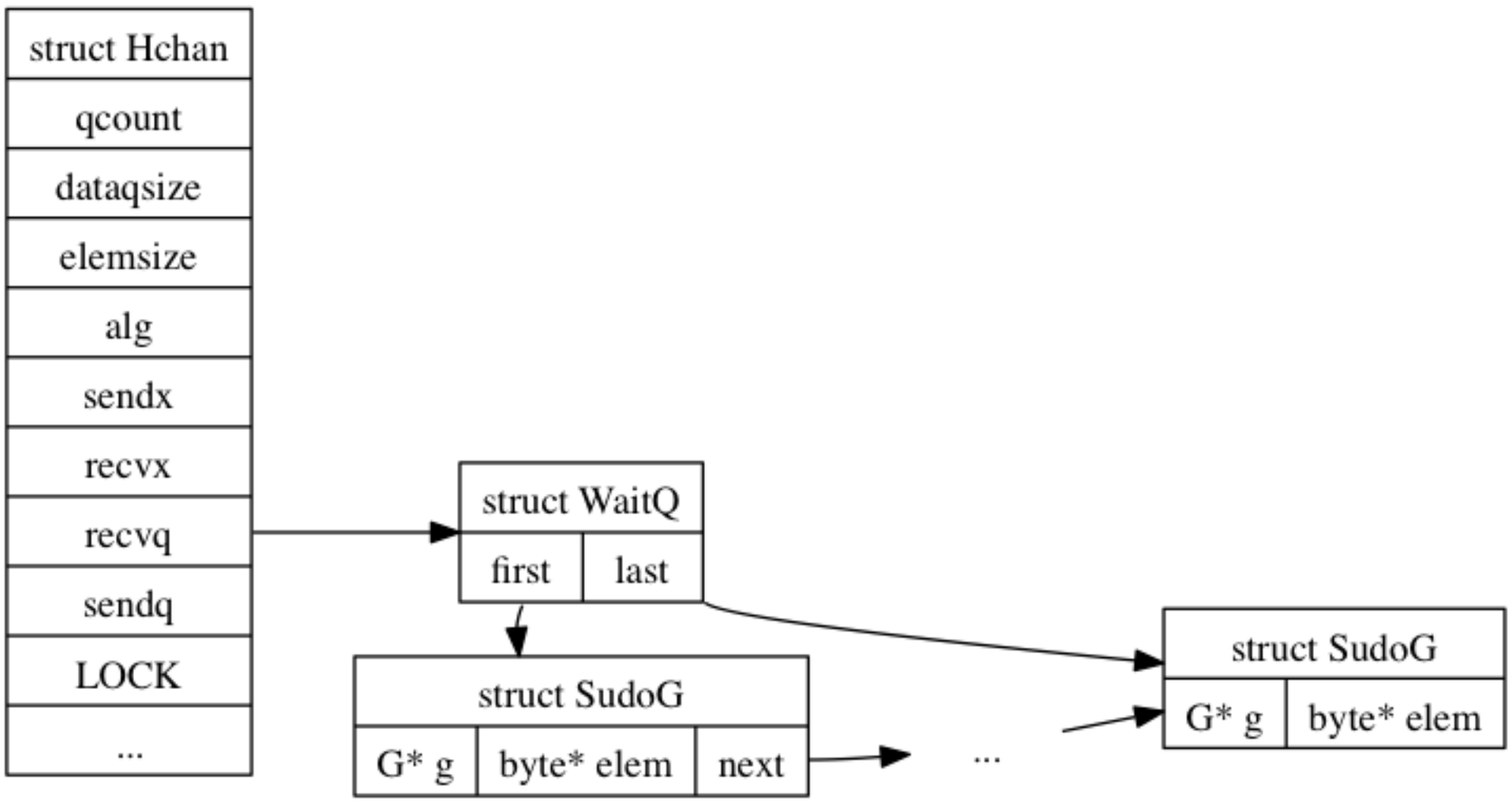
goroutine

- 协程!!!
- 每个goroutine必须有独立的栈
- 如何让goroutine非常轻量呢

goroutine

- 分段栈 -> 连续栈
- 有没有代价?
- 已经解决的问题 or 有没有更好的方法?

channel



- 如果一个goroutine执行channel操作阻塞，它会被挂在这个channel的结构上面，以便唤醒。

select

```
select {
case v := <-c1:
  fmt.Println("received from c1: ", v)
case c2 <- 1:
  fmt.Println("sent to c2")
case <-time.After(time.Second):
  fmt.Println("timed out")
default:
  fmt.Println("nothing ready at the moment")
}
```

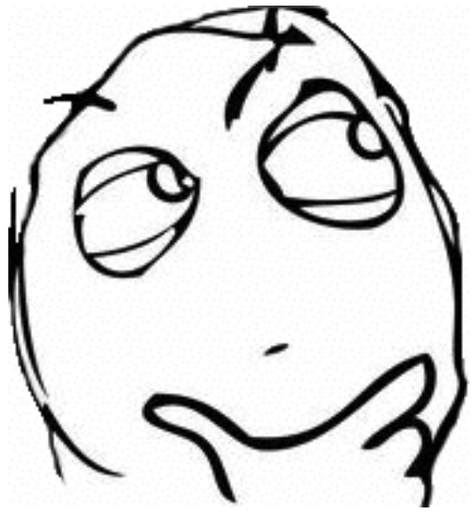
- select如何实现?

```
select {
case c1<-1:
case c2<-1:
case c3<-1:
}
```

```
select {
case <-c2:
case <-c3:
}
```

- select其实是一个整体
- c1 c2并不能独立对待
- 要么全部成功，要么失败，否则可能死锁

- 单纯对select结构加锁行不通!!!
- 共享一个全局锁?



select

- 使用前获取select中的所有channel的锁
- 按channel结构体的地址顺序加锁

- 研究课题：能否实现lock-free channels?

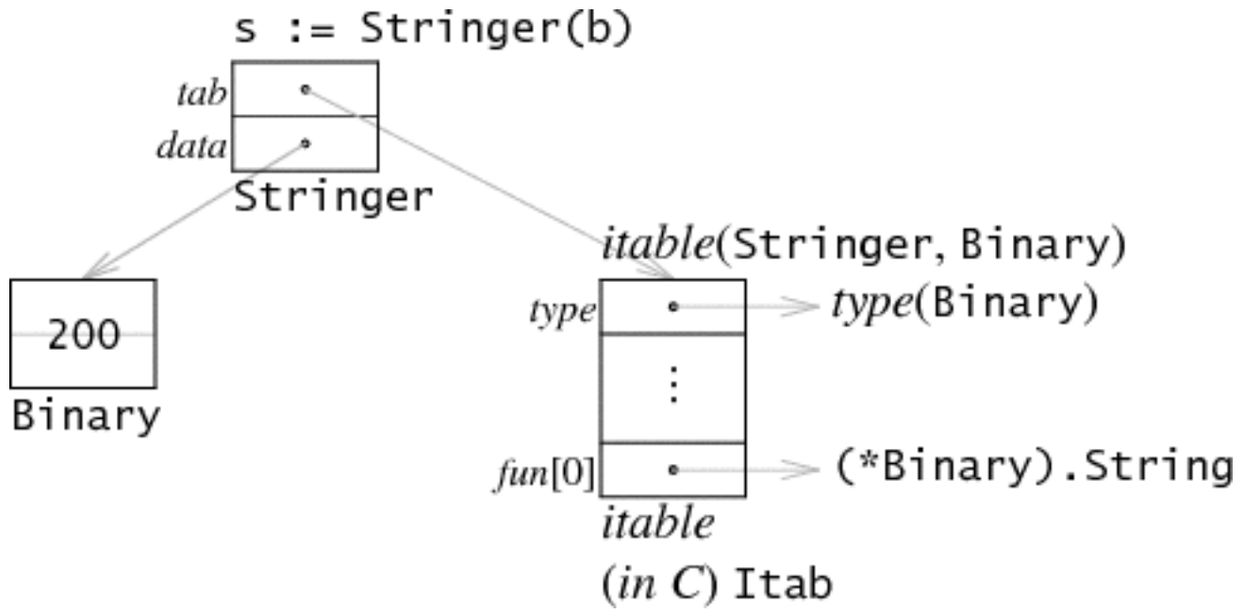
大纲

- 并发
- 接口
- 垃圾回收
- 调度
- 死锁检测

接口

- 接口定义了一系列方法
- 如果类型实现了这些方法，它就实现了这个接口
- 接口不依赖实现
- 避免了过度设计，比如OO里面基于继承的方式

接口



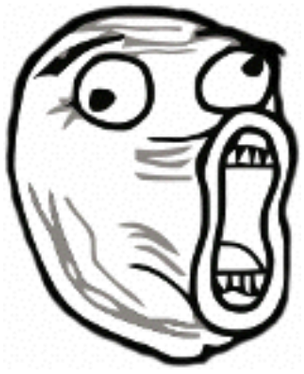
- 如何高效实现方法派发方法?

泛型

- func Sort(data []T, less func(x, y *T) bool)

- func Sort(data []interface{}, less func(x, y interface{}) bool)

泛型



- C说：我不管。(使用者累)
- C++把同一个函数(的不同类型)实现了很多遍。(编译器累)
- Java把所有东西都打包了，只有一个函数。(运行时累)
- 研究课题：怎么样实现才合理?

大纲

- 并发
- 接口
- 垃圾回收
- 调度
- 死锁检测

垃圾回收

- 王尼玛：内存管理太重要！交给机器我不放心
- 曹尼玛：内存管理太重要！给王尼玛管我不放心

Go还是很体贴大伙的~

Go vs Java

跟Java相比，Go内存方面有一些不同：

- 指针更少
- 大量的栈上分配
- 大量的goroutine
- 没有对象头

垃圾回收

- 并发->精确
- 是否分代？压缩？增量？

Go1.5 GC

- Go's new garbage collector is a concurrent, tri-color, mark-sweep collector

Off			GC disabled Pointer writes are just memory writes: *slot = ptr
Stack scan			Collect pointers from globals and goroutine stacks Stacks scanned at preemption points
Mark	WB on		Mark objects and follow pointers until pointer queue is empty Write barrier tracks pointer changes by mutator
Mark termination	WB on	STW	Rescan globals/changed stacks, finish marking, shrink stacks, … Literature contains non-STW algorithms: keeping it simple for now
Sweep			Reclaim unmarked objects as needed Adjust GC pacing for next cycle
Off			Rinse and repeat

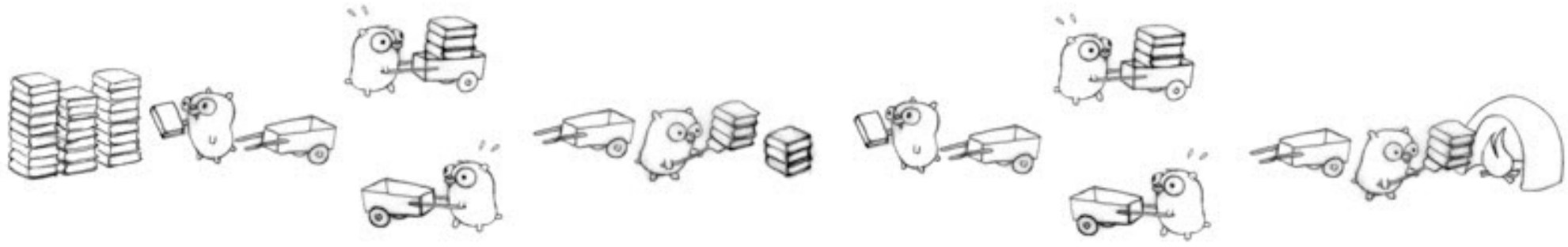
垃圾回收

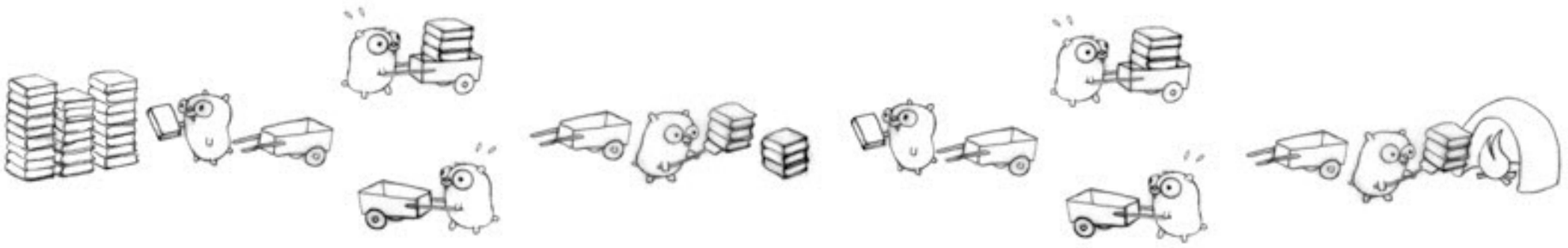
- 研究问题：最适合Go的垃圾回收算法是什么？

大纲

- 并发
- 接口
- 垃圾回收
- 调度
- 死锁检测

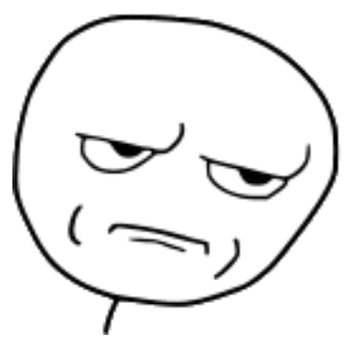
调度





行为难预测

- 用户代码具有不确定性
- 计算型的，IO型的，或者混合的
- 忙、闲、或者突发
- 没有提示!



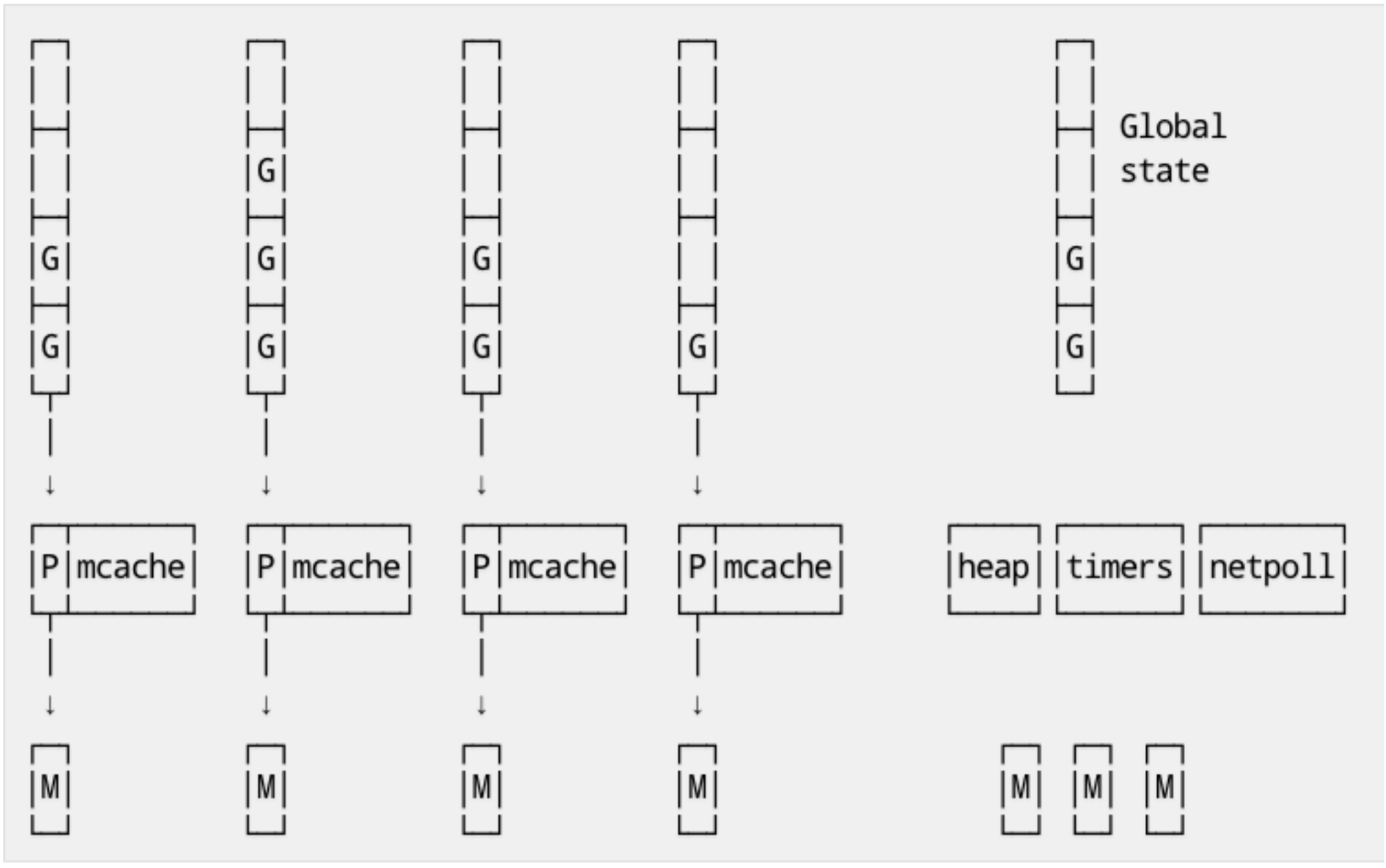
硬件很复杂

- 每核缓存
- 核间共享缓存
- cores shared between hyper threads (HT)
- multiple processors with non-uniform memory access (NUMA)

调度

- 起点：调度状态全局锁
- 工作流窃取调度器
- 网络poller整合进调度器
- 无锁工作队列

当前调度器



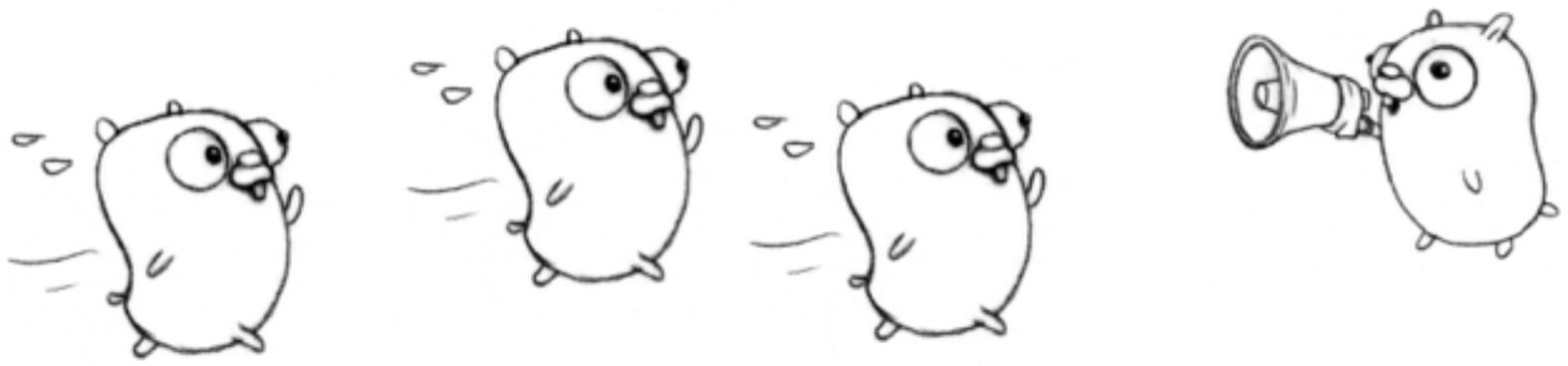
G - goroutine; P - logical processor; M - OS thread (machine)

- 缓存友好
- NUMA友好
- 调度的公平性
- 通信/非通信goroutines的分布
- 计时器和network poller的分布
- 让poll network跟上一次读在相同核上

大纲

- 并发
- 接口
- 垃圾回收
- 调度
- 死锁检测

死锁检测



死锁检测

- 构造一个图(Graph)
- L1锁住后再加锁L2，则构造边L1->L2
- 如果图中出现环，可能存在死锁
- 当图上新加一条边时，判断是否存在环

channel死锁

- Go不仅有锁，还有channel
- channel其实是信号量
- 关于信号量的死锁检测，目前还没有相关的理论

大纲

- 并发
- 接口
- 调度
- 垃圾回收
- 死锁检测

谢谢观赏

广州舜飞信息科技有限公司

舜飞科技
SunFaith

欢迎技术交流，更欢迎优秀人才加入