tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=2000&h=837&s=171428&e=png&b=ede7e6)

显示堆内存占用只有14.5MB，那么其他120MB内存去哪儿了?

通过`top`查看

```
top - 06:22:47 up  3:34,  0 user,  load average: 0.25, 0.32, 0.47
Tasks:   1 total,   0 running,   1 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.8 us,  0.9 sy,  0.0 ni, 97.2 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :  31644.3 total,  15941.1 free,   4054.4 used,  12104.9 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.  27589.8 avail Mem
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  17867 65532     20   0 1336224  77632  27100 S   0.0   0.2  22:36.45 manager
```

那是不是Goroutine, 线程泄漏？ 排查了Goroutine只有84个，线程也稳定在16个。

runtime.MemStats

`runtime.MemStats` 是Golang runtime提供的用来存储关于内存统计信息的结构体，注意这里计算的是虚拟内存与实际内存是有差异的，层级结构如下：

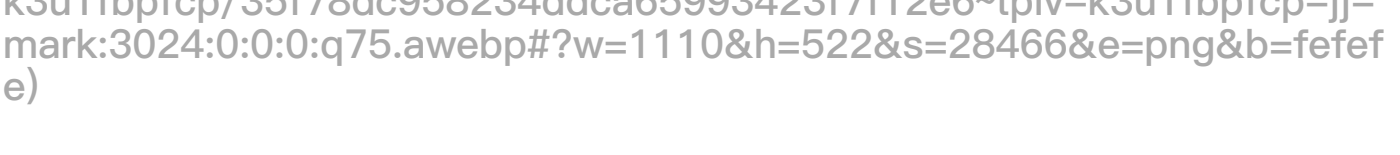
```
Sys # 总的系统内存
    HeapSys # 堆总内存
        HeapInUse # 堆中在使用的内存
        HeapIdle # 堆空闲内存
            HeapReleased # 返回给OS的内存数据
            ...
```

```
    StackSys # Stack总内存
    MSpanSys # mspan总内存
    CacheSys
    OtherSys

```

`MemStats`各个数据有如下关系：

表达式	含义
`Sys=HeapSys + StackSys + MSpanSys + ..Sys`	总内存
`Sys-HeapReleased`	Golang总的使用内存
`HeapIdle-HeapReleased`	Golang系统预留内存
`HeapInus 验证，好像走到了死胡同，唯一的推测还是这部分缺少内存还是Golang系统保留了，如果再次压测内存再与前一次测验相差不多的话，那么说明多出的内存是被Golang预留了。

压测之后，确实符合预期，两个压测后内存基本一致


Golang的内存分配非常复杂，暴露出的指标很难推测出这个问题的根因，需要根据代码深入分析，目前只能合理推测RSS与MemStats显示的内存差值是由于以下原因：

* 采集误差
* 内存碎片
* Golang系统预留
* 内核内存占用
* CGO使用的内存

总结
--

通过以上分析，我们可以定位Golang引起的内存泄漏，一般排查步骤如下：

1. 通过PProf获取Heap相关内存系统
2. 通过监控排查线程、Goroutine是否有异常
3. 检查Golang版本是否开启了`MADV_FREE`
4. 获取`runtime.MemStats`进行分析

* `RSS ~ Sys - HeapReleased`, 可能是Golang问题
* `RSS >> Sys - HeapReleased`, 可能是非Golang问题
* `HeapIdle-HeapReleased > 0`, Golang系统预留了内存

通过这些步骤，我们可以更有效地识别和解决由Golang引起的内存相关问题。

参考
--

* [go.dev/src/runtime...](http://cxyroad.com/”https://go.dev/src/runtime/mstats.go”)
* [github.com/golang/go/i...](http://cxyroad.com/”https://github.com/golang/go/issues/32284”)
* [fanlv.fun/2022/06/02/...](http://cxyroad.com/”https://fanlv.fun/2022/06/02/golang-pprof-mem”)
* [www.datadoghq.com/blog/go-mem...](http://cxyroad.com/”https://www.datadoghq.com/blog/go-memory-metrics/”)

> Explore more in [qingwave.github.io](http://cxyroad.com/”https://qingwave.github.io/”)