# Golang 在高并发场景下的技术实践

周洋

# Go 相关的从业经历:

- 腾讯会议　　后台
- 某大厂和直播领域　　后台
- 奇虎 360　　负责长连接和 IM 相关 PAAS 服务

# 目录


- 金三银四面试讨论
- 高并发场景的知识储备
- Golang 高并发场景的设计与问题

# 金三银四面试讨论

# 金三银四面试问题讨论

极客时间 | 训练营

## 基础数据结构和标准库

基础: Slice，Interface

选修: net，http，bufio，grpc

补充阅读：50 Shades of Go

## 语言特性常识

基础： GMP模型，Channel，GC

选修： GMP模型处理系统调用环节的调度策略;

补充阅读: go mem model，effective go, go spec

## 实践

基础： context，error，go tool（pprof）

选修： 通信Client的封装与设计（连接池，服务发现，流控，单测）

补充阅读：大厂实践案例

# 高并发场景下的知识储备

# 高并发场景的知识储备

极客时间 | 训练营

## 操作系统与硬件环境

- CPU：
  - golang runtime实现与操作系统上下文切换的影响
- 内存管理：
  - 如果宏观上的定位内存泄露
  - 在 golang 场景下的 GC 与内存分配的影响
- 网络：
  - 资源的限制、TCP 优化组参、网卡中断处理在多核下的均衡性。
  - 网络瓶颈下，golang 程序的表现。
  - 通信框架基于网络的配置。
- 工具：
  - golang pprof，pmap，sar，strace、pstack，动态诊断与静态诊断技术，eBPF 技术
- 其他：
  - 容器化对于 golang 程序的影响，跨平台对于 golang 的影响

# 高并发场景的知识储备



架构能力一定程度是对不同环节和层次的精确计算、评估、整合

| 基础库能力 | **框架与基准性能(grpc，gin)** | 外网依赖服务的性能上限<br>(mysql,redis) |
| --- | --- | --- |
| • 网络库设计，连接池设计<br>• 日志库性能<br>• 基础库协程模型和数据传输的设计，核心数据结构的极限压力的内存和协程占用<br>• 容错能力，与异常处理 | • RPC框架的基础能力，基准测试，异常测试。<br>• 核心数据结构的编解能力(PB,JSON) | • 系统和资源的性能上限<br>• 线性的增长，非线性的增长。 |

# Golang 高并发场景的设计与问题

# Golang 高并发场景的设计与问题-案例分享

极客时间 | 训练营

## 场景描述：

50+ 内部产品，万款开发平台 app

实时长连接数亿量级，日独数十亿量级

1 分钟内亿量级广播,日下发峰值百亿量级

400 台物理机，9 个独立集群，国内外近10个 IDC

## 性能描述：

线上单机最高160w长连接 （24核 E5-2630 @ 2.30GHz  64G内存 ）

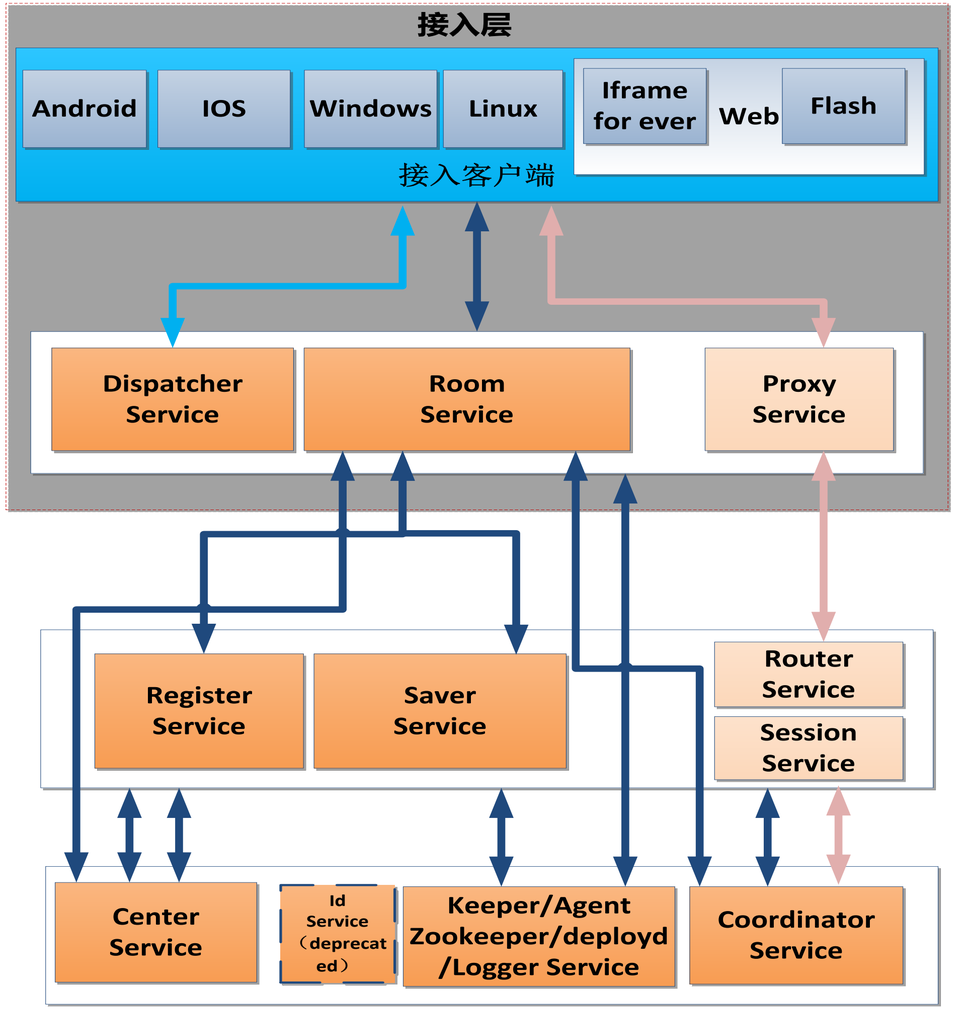
Qps 在2~5w（取决于协议版本，业务逻辑，接入端网络状况，不算心跳包）

可以通过 300w 长连接压测（网络,连接稳定,无带宽限制,实际可以更高,决定于广播时候业务内存开销的 cpu 消耗带来的心跳或者业务延时能否接受）

小包(4k)打满万兆网卡

# Golang 高并发场景的设计与问题-案例分享






架构特点:

- 实时性强: 通信和传输性能并发要求高
- 微服务: 内部模块交互多，通信复杂。
- 写扩散场景: 对存储重依赖，存储性能需要横向扩展。需要内存缓存策略做进一步缓冲。

# 问题与瓶颈

| | | |
|---|---|---|
| 42.35G | 36.27G | Aug31 |
| 43.55G | 39.67G | Aug31 |
| 42.01G | 36.57G | Aug31 |
| 42.58G | 39.62G | Aug31 |
| 42.15G | 38.72G | Aug31 |
| 42.44G | 35.80G | Aug31 |
| 47.50G | 38.14G | Aug31 |
| 43.00G | 36.04G | Aug31 |
| 44.56G | 35.46G | Aug31 |
| 43.41G | 35.37G | Aug31 |
| 47.66G | 37.53G | Aug31 |
| 47.95G | 44.10G | Aug31 |
| 49.01G | 40.49G | Aug31 |
| 49.38G | 41.34G | Aug31 |
| 48.16G | 41.19G | Aug31 |
| 51.59G | 41.02G | Aug31 |
| 48.31G | 40.60G | Aug31 |
| 54.86G | 41.43G | Aug31 |
| 50.12G | 40.86G | Aug31 |
| 47.67G | 41.41G | Aug31 |
| 47.99G | 40.53G | Aug31 |
| 47.79G | 41.51G | Aug31 |
| 48.91G | 42.67G | Aug31 |
| 48.16G | 42.64G | Aug31 |

| | | |
|---|---|---|
| 29.46G | 15.86G | Aug31 |
| 55.62G | 51.63G | Aug31 |
| 50.59G | 33.09G | Aug31 |
| 51.33G | 35.53G | Aug31 |

| memVirt | memRes | StartTime |
|---|---|---|
| 67.99G | 56.39G | Aug31 |
| 67.40G | 57.09G | Aug31 |
| 69.50G | 54.26G | Aug31 |
| 69.52G | 54.24G | Aug31 |
| 38.44G | 35.83G | Sep01 |
| 41.54G | 36.16G | Sep01 |
| 43.61G | 36.06G | Sep01 |
| 43.65G | 36.41G | Sep01 |
| 40.73G | 35.98G | Sep01 |
| 45.21G | 35.98G | Sep01 |

单机内存占用
高达69G
GC 3~6s

- 核心数据结构设计和协程占用较大，golang早期的GC瓶颈

内存暴涨

2~3s 的 GC

奔放的协程使用

散列在协程里面的 io

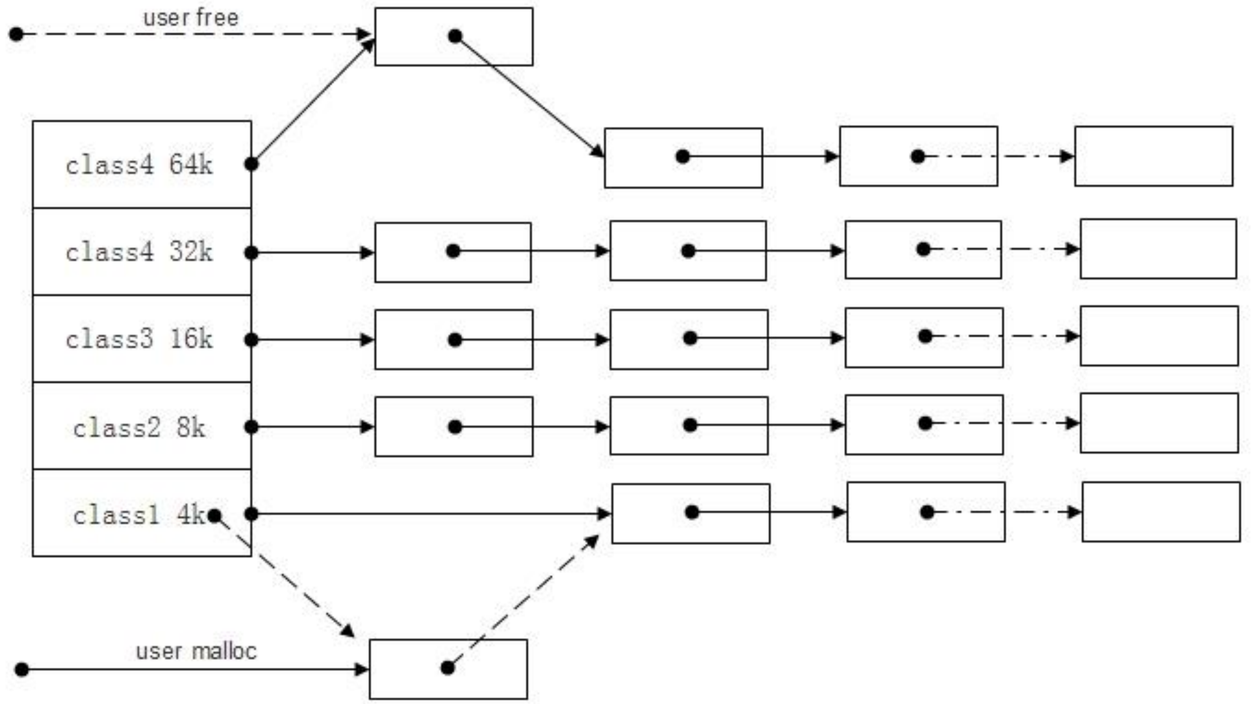
# 问题与瓶颈



性能瓶颈分析

- 对象复用
- 资源复用(timer)
- bufio，内存复用
- 连接复用与多路复用
- 集中化处理

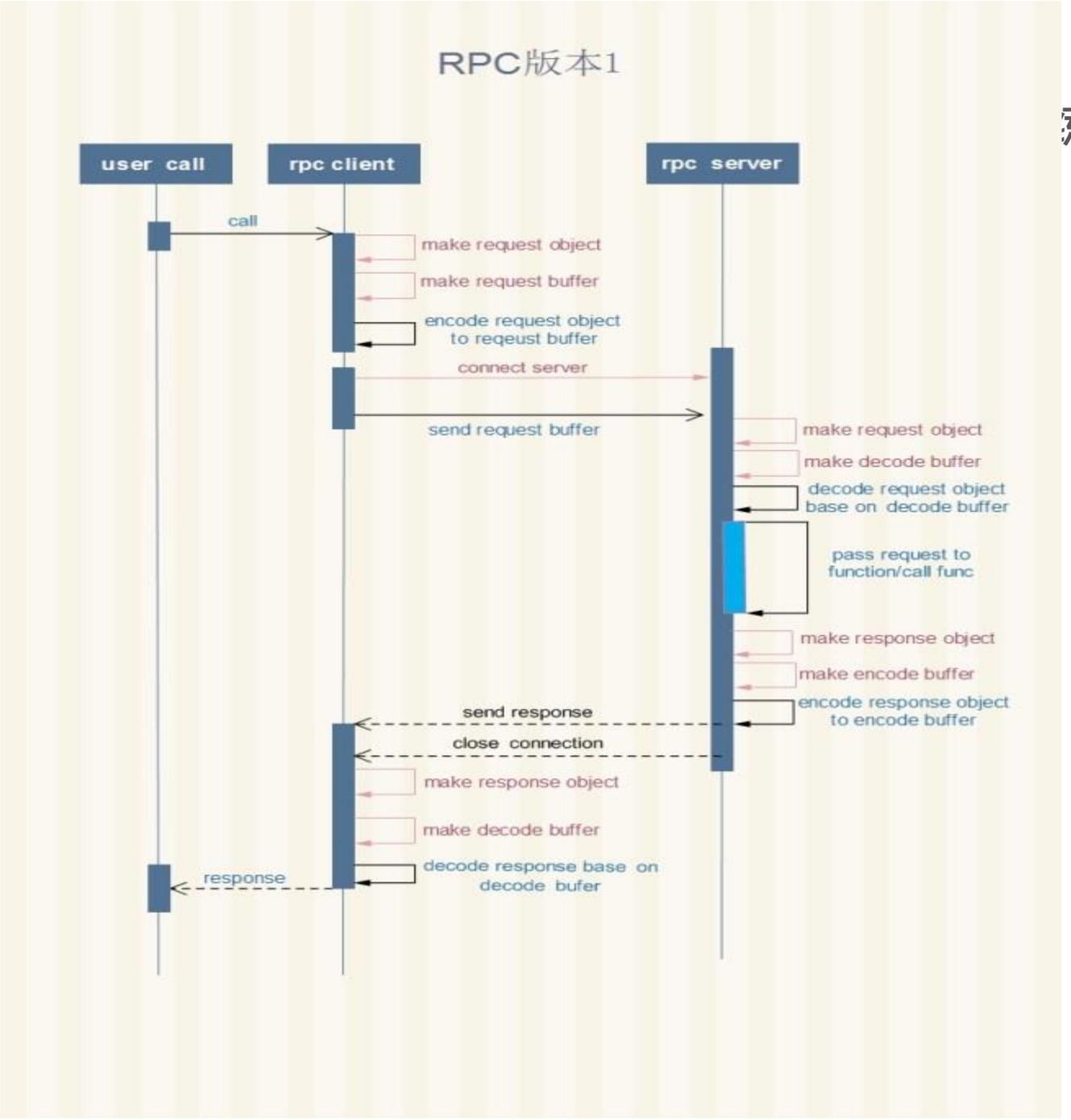
```
atomic.CompareAndSwapPointer(bufList, buf.next, unsafe.Pointer(buf))
```




```
atomic.CompareAndSwapPointer(bufList, ptr, ((*ElasticBuf)(ptr)).next)
```

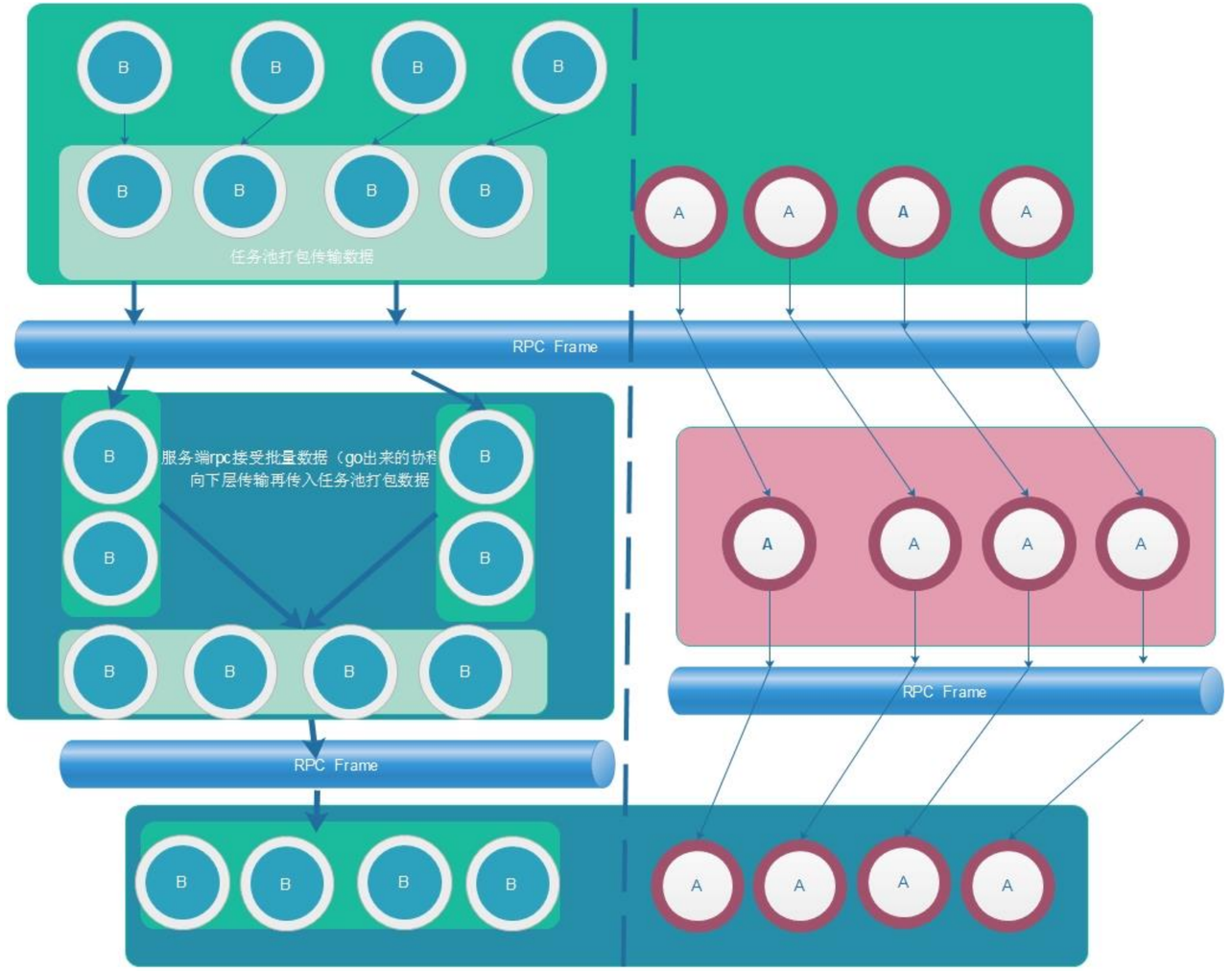
性能瓶颈分析

- 对象复用
- 资源复用(timer)
- bufio，内存复用
- 连接复用与多路复用
- 集中化处理



## 性能瓶颈分析

- 对象复用
- 资源复用(timer)
- bufio，内存复用
- 连接复用与多路复用
- 集中化处理

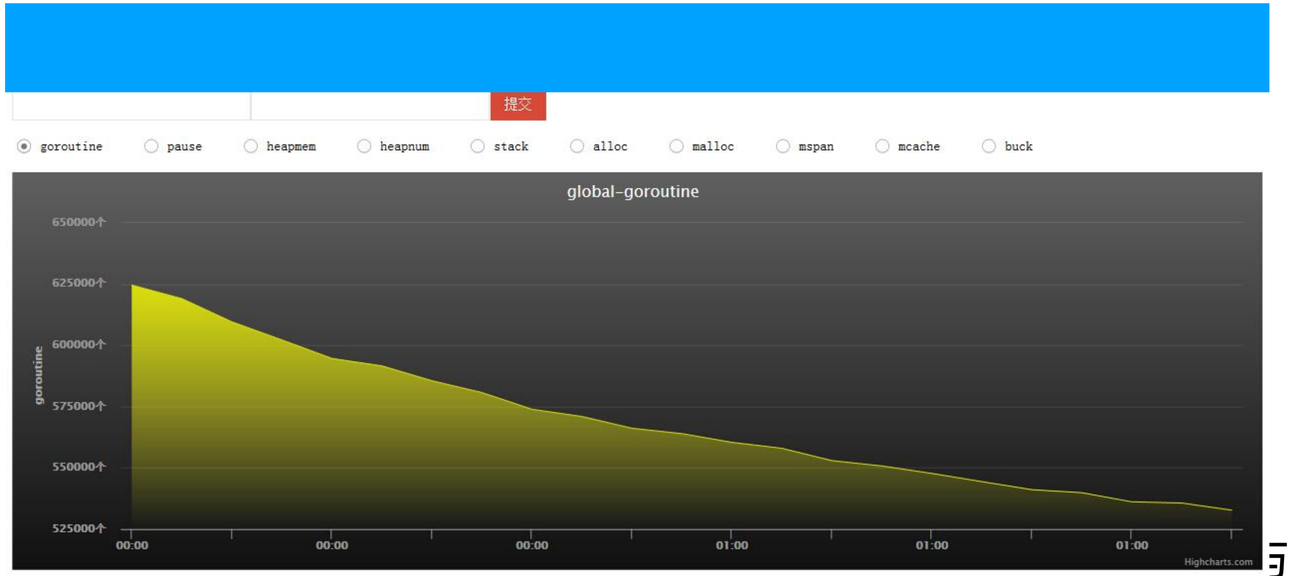


# Go 语言运维管理方面的独特魅力



- Go 语言原生提供的各组工具，构建分布式系统配套设施方面，提供了便利
- 配套设施= 测试 + 调优 + 监控 + 运维
- 便利 = 原生 profiling 工具 + 开协程模拟测试终端+协程协作模拟业务

# 具有 go 语言特色的运维




最近24小时性能图列表

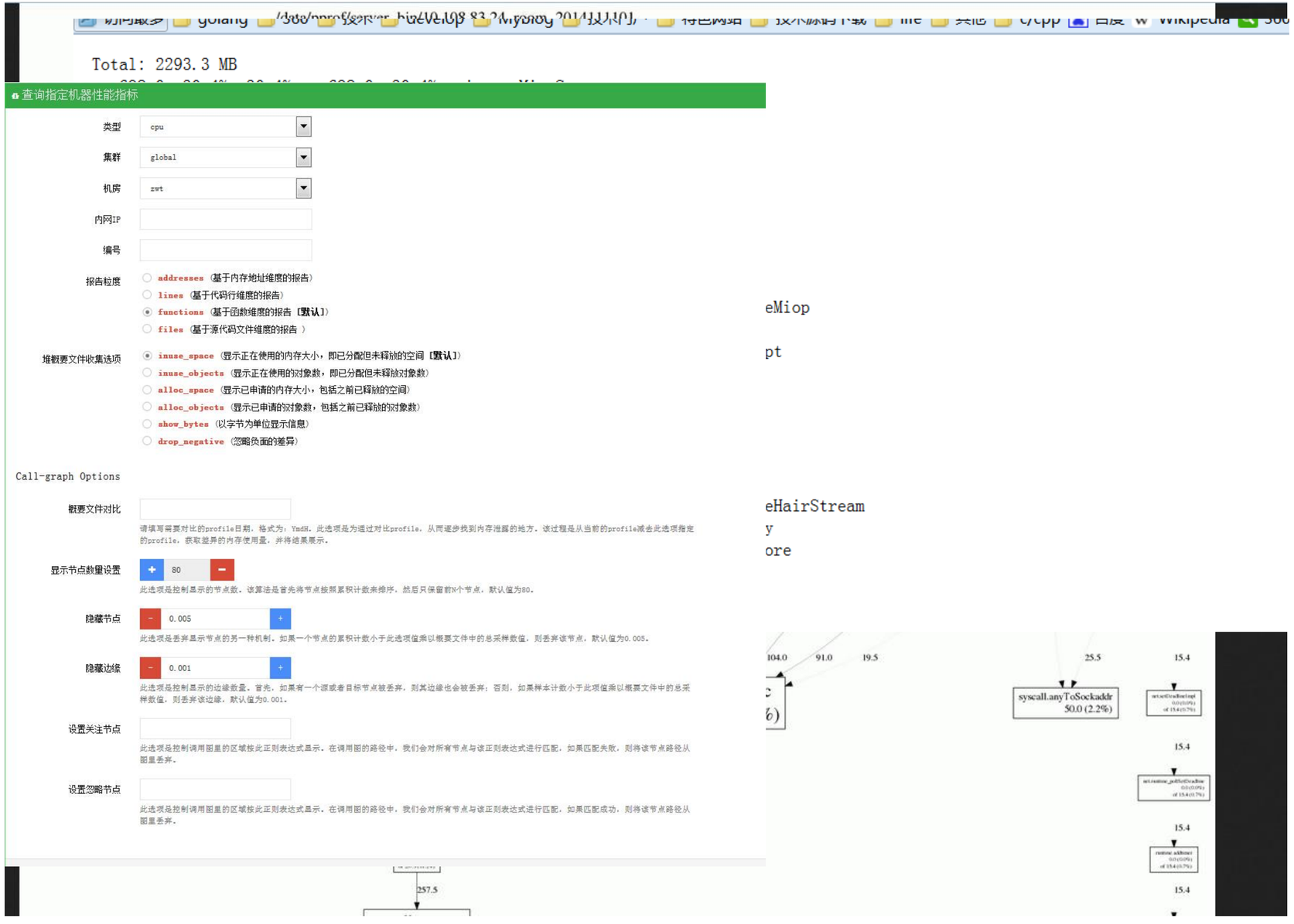
| 时间 | pprof | pprof | Func | Func | Goroutine | Thread | Block |
|---|---|---|---|---|---|---|---|
| 03-17 00:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 23:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 22:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 21:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 20:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 19:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 18:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |

最近24小时性能图列表

| 时间 | pprof | pprof | Func | Func | Goroutine | Thread | Block |
|---|---|---|---|---|---|---|---|
| 03-16 12:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 11:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 10:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 09:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 08:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 07:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |
| 03-16 06:00 | Cpu | Heap | CpuFunc | HeapFunc | Goroutine | Thread | Block |

# 具有 go 语言特色的运维：以项目为例





# 具有 go 语言特色的运维：以项目为例



saver　　zwt

| 机器编号 | 内网IP | 端口 | 空闲 | 活跃 | 新建 | 新建失败 | 取超时 | 调用出错 | 空闲超时 | qps | 统计时间 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 888 | 10.108.102.227 | 6300 | 555 | 1 | 24599 | 0 | 13317 | 0 | 16045 | 3660 | 2015-04-16 18:54:00 |
| 889 | 10.108.102.228 | 6300 | 383 | 0 | 24117 | 15 | 5235 | 0 | 15285 | 3695 | 2015-04-16 18:54:00 |
| 890 | 10.108.102.229 | 6300 | 549 | 0 | 8822 | 12 | 13117 | 0 | 2315 | 3765 | 2015-04-16 18:54:00 |
| 891 | 10.108.102.230 | 6300 | 556 | 0 | 11771 | 0 | 13680 | 0 | 2599 | 3737 | 2015-04-16 18:54:00 |
| 892 | 10.108.102.231 | 6300 | 510 | 1 | 9672 | 0 | 13092 | 0 | 2178 | 3832 | 2015-04-16 18:54:00 |
| 893 | 10.108.102.232 | 6300 | 530 | 0 | 14662 | 0 | 13599 | 18 | 2397 | 3784 | 2015-04-16 18:54:00 |
| 1385 | 10.108.102.234 | 6300 | 366 | 0 | 10769 | 0 | 4915 | 0 | 2540 | 3742 | 2015-04-16 18:54:00 |

| 机器编号 | 内网IP | 端口 | idle | working | creating | read_amount | 统计时间 |
|---|---|---|---|---|---|---|---|
| 1192 | 10.108.102.112 | 6020 | 214 | 0 | 0 | 28894941 | 2015-04-16 20:26:00 |
| 1186 | 10.108.102.91 | 6020 | 206 | 0 | 0 | 28688755 | 2015-04-16 20:26:00 |
| 1187 | 10.108.102.92 | 6020 | 207 | 0 | 0 | 28598266 | 2015-04-16 20:26:00 |
| 1188 | 10.108.102.93 | 6020 | 208 | 0 | 0 | 28810773 | 2015-04-16 20:26:00 |
| 1189 | 10.108.102.94 | 6020 | 207 | 0 | 0 | 28774618 | 2015-04-16 20:26:00 |
| 1190 | 10.108.102.95 | 6020 | 201 | 0 | 0 | 28537627 | 2015-04-16 20:26:00 |

# 总结回顾



Go 语言在基于并发协作的，重业务逻辑的基础服务方向非常适用

适用 = 开发体验好 + 服务稳定 + 性能满足需要

Go 语言程序开发需要找到一种平衡，既利用协程带来的便利性又做适当集中化处理

策略 = 按请求和业务逻辑并行+任务池集中数据合并请求 + 连接池集中收发

Go 语言开发追求开销优化的极限，谨慎引入其他语言领域高性能服务的通用方案

内存池+对象池使用 与 代码可读性与整体效率的权衡

Go 语言原生提供的各组工具，构建分布式系统配套设施方面，提供了便利

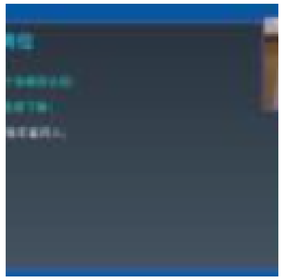
生态圈 = 测试 + 调优 + 监控 + 运维

便利 = 原生profiling工具 + 通信库集成监控+协程协作模拟业务压测

# Q&A




Go面试题new.pdf



毛剑面试公开课.m...


- 本次直播 PPT
- 大厂面试真题
- 毛剑技术面试公开课
- 算法刷题冲刺班

扫码领取 👉👉👉

# THANKS