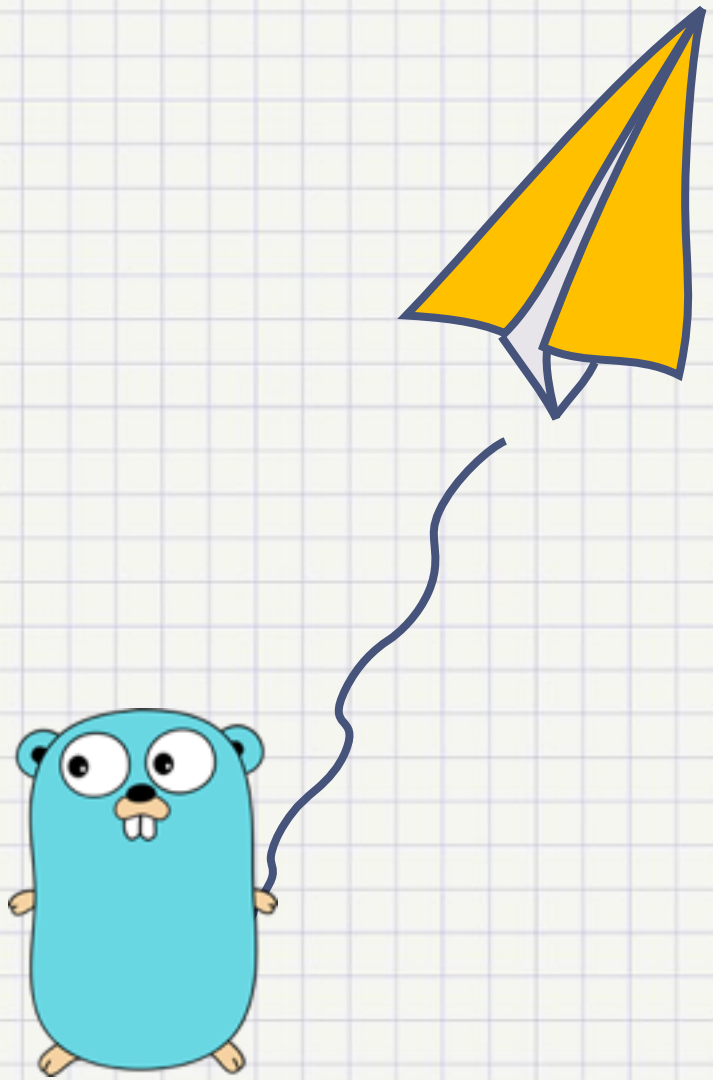


Golang 高性能实战

github.com/rfyiamcool

xiaorui.cc

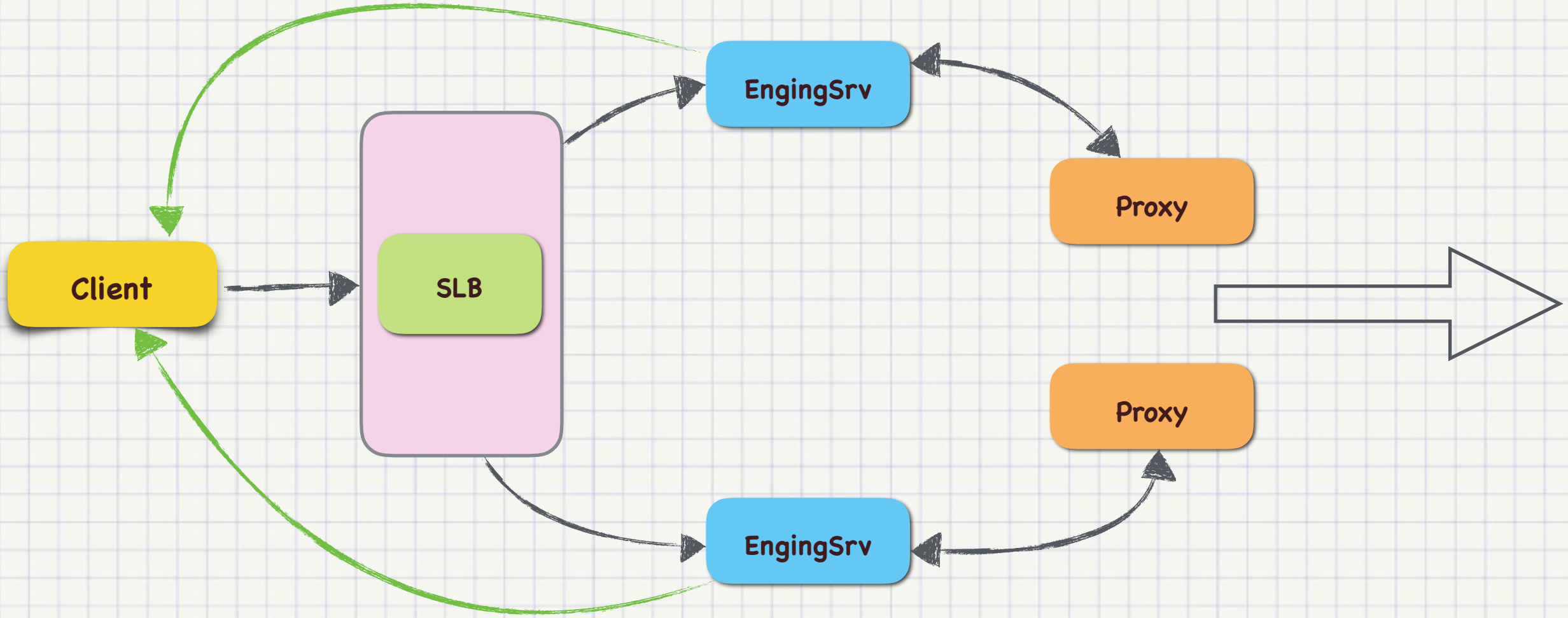
CDN 刷新系统 是一个高性能、可扩展的分布式系统.

支持全网刷新及预缓存业务.

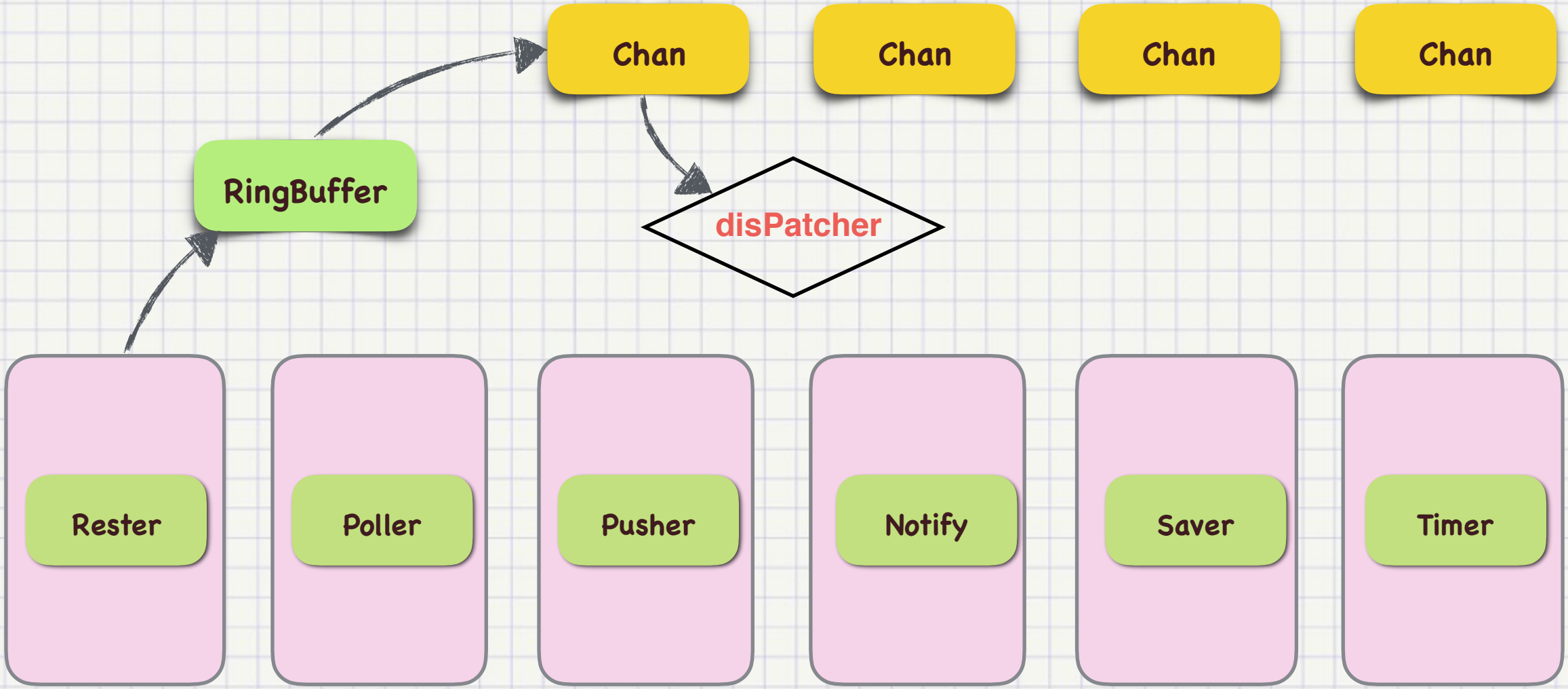
Contents
目录

架构及框架设计

架构设计



框架设计


Chan
Chan
Chan
Chan
RingBuffer
disPatcher
Rester
Poller
Pusher
Notify
Saver
Timer

技术选型

语言: Golang

框架: Gin

压缩: Snappy

序列化: Json-iterator

ORM: Gorm

客户端: imroc/req

日志: Logrus

配置: Toml

缓存: Redis

持久化: Mysql

调优

基本优化

不要滥用time.After，减少gc压力

不要滥用goroutine, 减少gc压力

不要滥用锁, 引起runtime上下文切换

使用unsafe装换类型

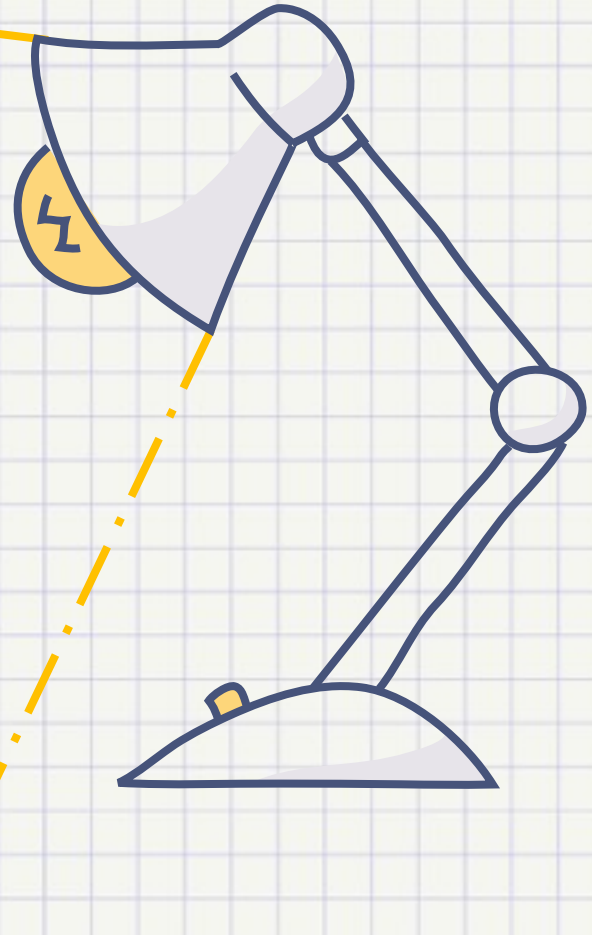
不要总是[]byte String转换.

基本优化

提前预估size, make 之

临时的mapl slice，可采用sync.pool

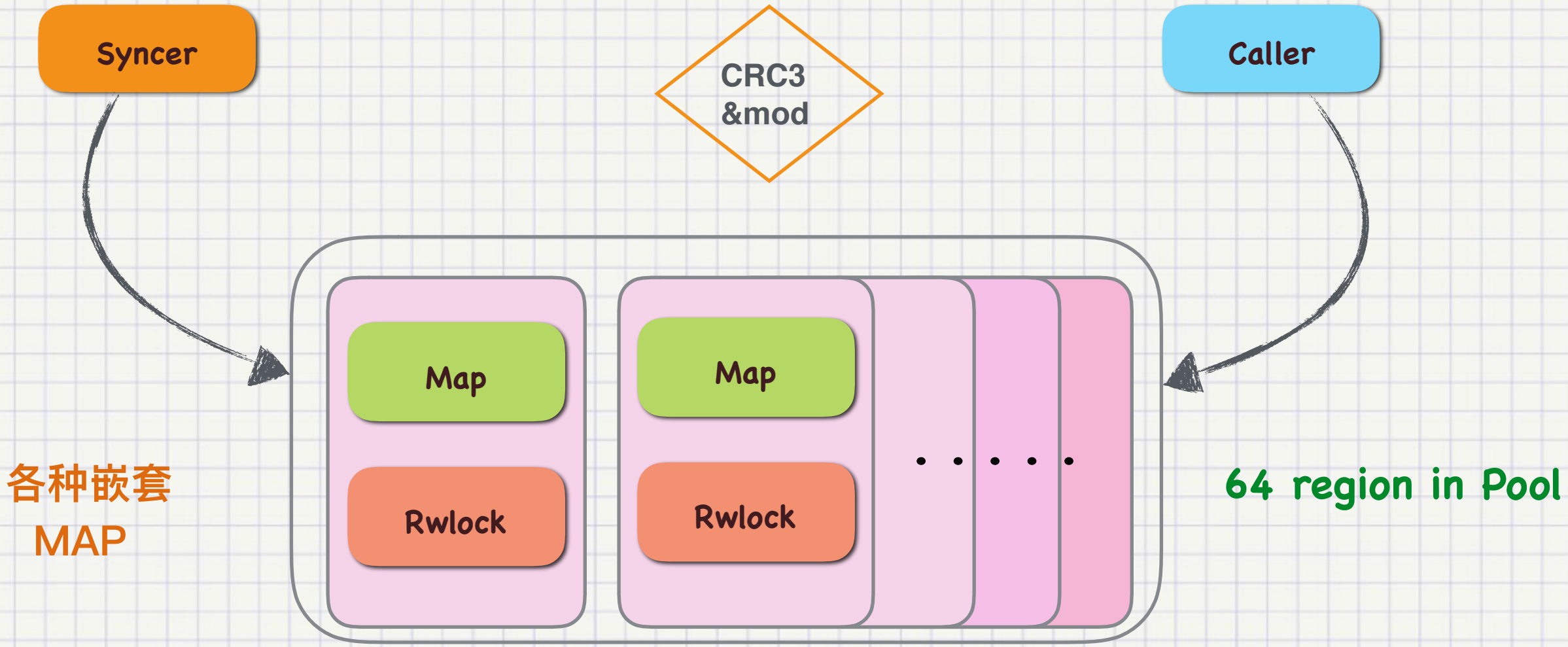
大于32kb，也可使用 sync.pool



业务调优

- 减少网络io的消耗
  - 批量接口
  - 网络调用链
  - 使用连接池
- 缓存内存化
  - 减少同步逻辑
  - 使用压缩，节省带宽

全局任务状态

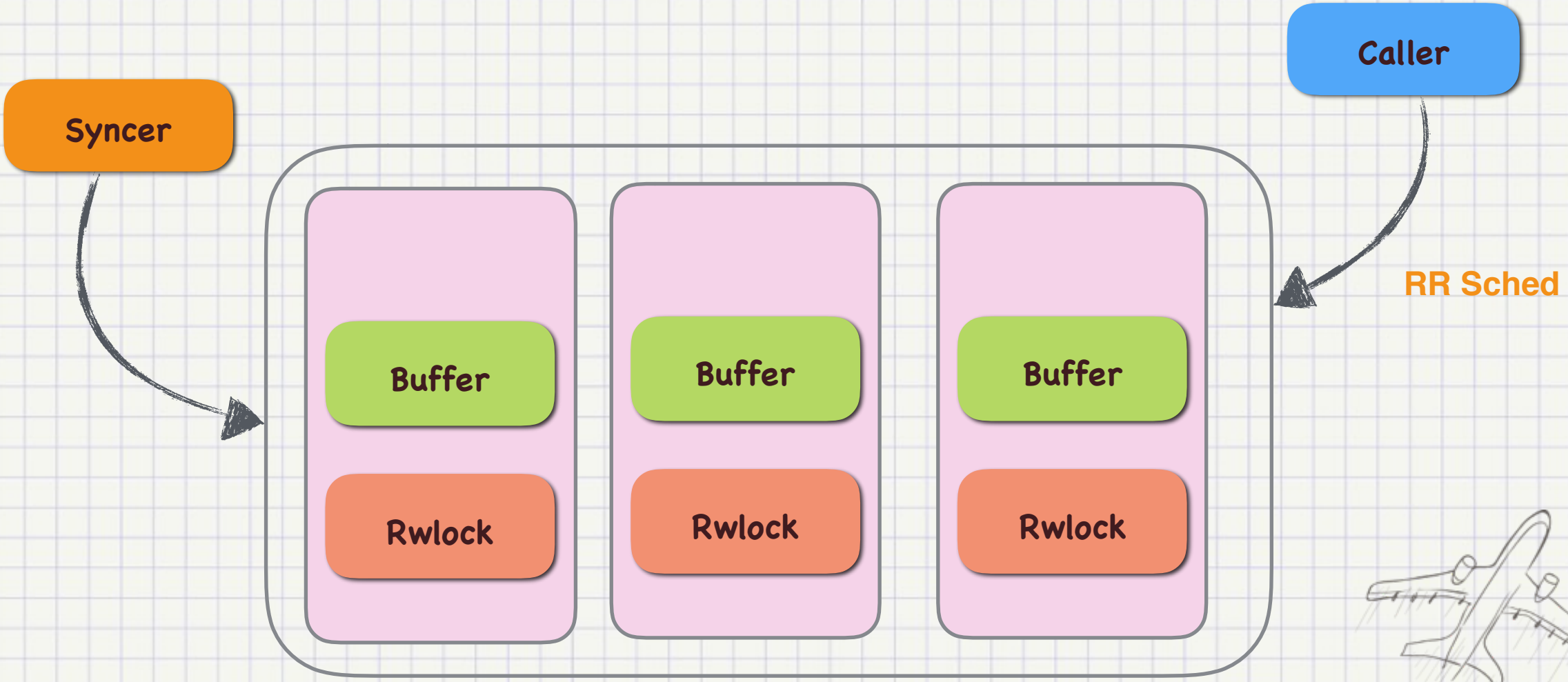

Syncer
CRC3
&mod
Caller
Map
Rwlock
Map
Rwlock
各种嵌套
MAP
64 region in Pool

全局任务状态-code

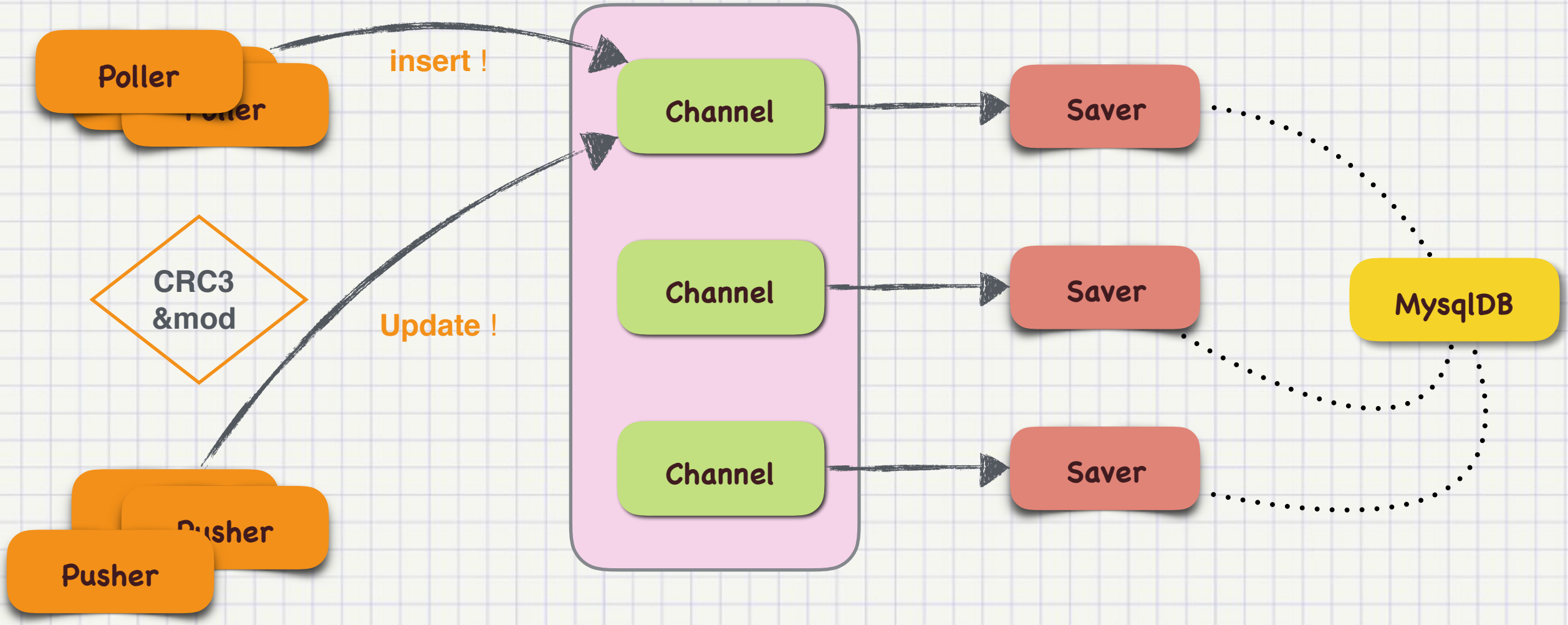

```go
var (
    TaskMapRegionPool = [PreAllocRegion]*TaskMapRegion{}
)

type TaskMapRegion struct {
    map[string]*DetailTaskStats
    *sync.RWMutex
}

type DetailTaskStats struct {
    TaskId       string
    TaskType     string
    WorkingLayer string
    Active       bool // 是否还需要执行?
    Counter      int  // 计数器
    Total        int  // 任务总数
    DueTimeStamp time.Time
    CreatedAt    time.Time
    UpdatedAt    time.Time
    Subtasks     map[string]map[string]string
}
```

```go
func GetOneMapRegion(taskId string) *TaskMapRegion {
    idx := utils.HashToInt(taskId) % PreAllocRegion
    return TaskMapRegionPool[idx]
}

func ExistInRegionPool(taskID string) bool {
    reg := GetOneMapRegion(taskID)
    reg.AcquireR()
    defer reg.ReleaseR()

    _, ok := reg.HasTaskID(taskID)
    return ok
}
```

缓存多副本


Caller
Syncer
RR Sched
Buffer
Rwlock
Buffer
Rwlock
Buffer
Rwlock

操作顺序排队

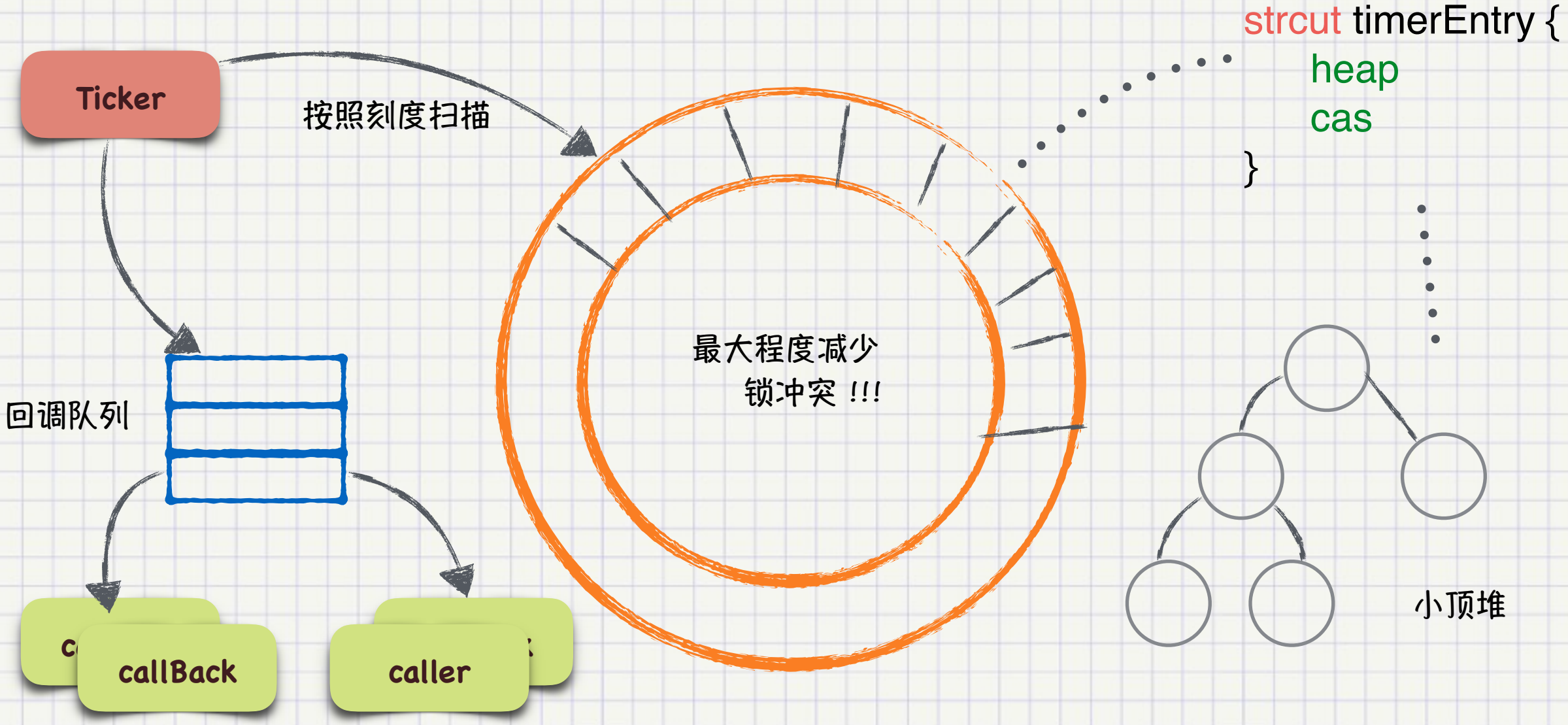

Poller
insert !
Channel
Saver
CRC3
&mod
Update !
Channel
Saver
MysqlDB
Channel
Saver
Pusher

操作顺序排队-CODE

```go
var (
    preSaveNum    int
    saveQueuePool []chan *saveAction
)

func getOneSaveQueue(taskId string) chan *saveAction {
    idx := utils.HashToInt(taskId) % preSaveNum // range 0 <-> (preSaveNum - 1)
    return saveQueuePool[idx]
}

func pushOneSaveQueue(taskId string, data *saveAction) bool {
    queue := getOneSaveQueue(taskId)
    select {
    case queue <- data:
        return true
    default:
        // log
        time.Sleep(1 * time.Second)
        return false
    }
}

func GetSaverLength() int {
    c := 0
    for _, queue := range saveQueuePool {
        c += len(queue)
    }
    return c
}
```

```go
// 消费队列
func (s *saveHandler) popQueue() (*saveAction, bool) {
    var outDto *saveAction
    var ok bool

    select {
    case outDto = <-saveQueuePool[s.slot]:
        ok = true
    case <-EngineWaitGroup.ExitQ:
        return outDto, false
    }

    return outDto, ok
}

// 分段的槽位需要跟协程数对应
func initSaveQueuePool(size int) {
    var defOneSize = 100

    preSaveNum = conf.Gconf.Engine.SaverWorkerNum
    saveQueuePool = make([]chan *saveAction, preSaveNum, preSaveNum)

    oneSize := size / preSaveNum
    if oneSize < defOneSize {
        oneSize = defOneSize
    }
```

时间轮设计


Ticker
按照刻度扫描
strcut timerEntry {
heap
cas
}
回调队列
最大程度减少
锁冲突 !!!
callBack
caller
小顶堆

stats 全局状态

```
{
    "alert_queue": 30,
    "input_task_queue": 400,
    "notify_queue": 1092,
    "pushing_queue": 311,
    "reporting_queue": 612,
    "saver_queue": 3197,
    "tasks_in_cache": 13189,
    "timer_tasks": 13189,
    "timestamp": 1521682815
    "qps_proxy": {
        "15m": 9031,
        "1m": 10221,
        "5m": 11005,
        "avg": 91110,
        "count": 897811
    },
    "qps_boss": {
        "15m": 5113,
        "1m": 5200,
        "5m": 5300,
        "avg": 5001,
        "count": 297811
    },
}
```

排查死锁技巧

Caller

Acquie()

方便调试, 抽象锁方法.

加入 代码行 及 随机id

acquie加入, release删除.

Caller

Release()

单一的就是死锁了

排查OOM

追踪内存泄露点 OR 增长点

fatal error: runtime: out of memory

runtime stack:

runtime.throw(0x665297, 0x16)

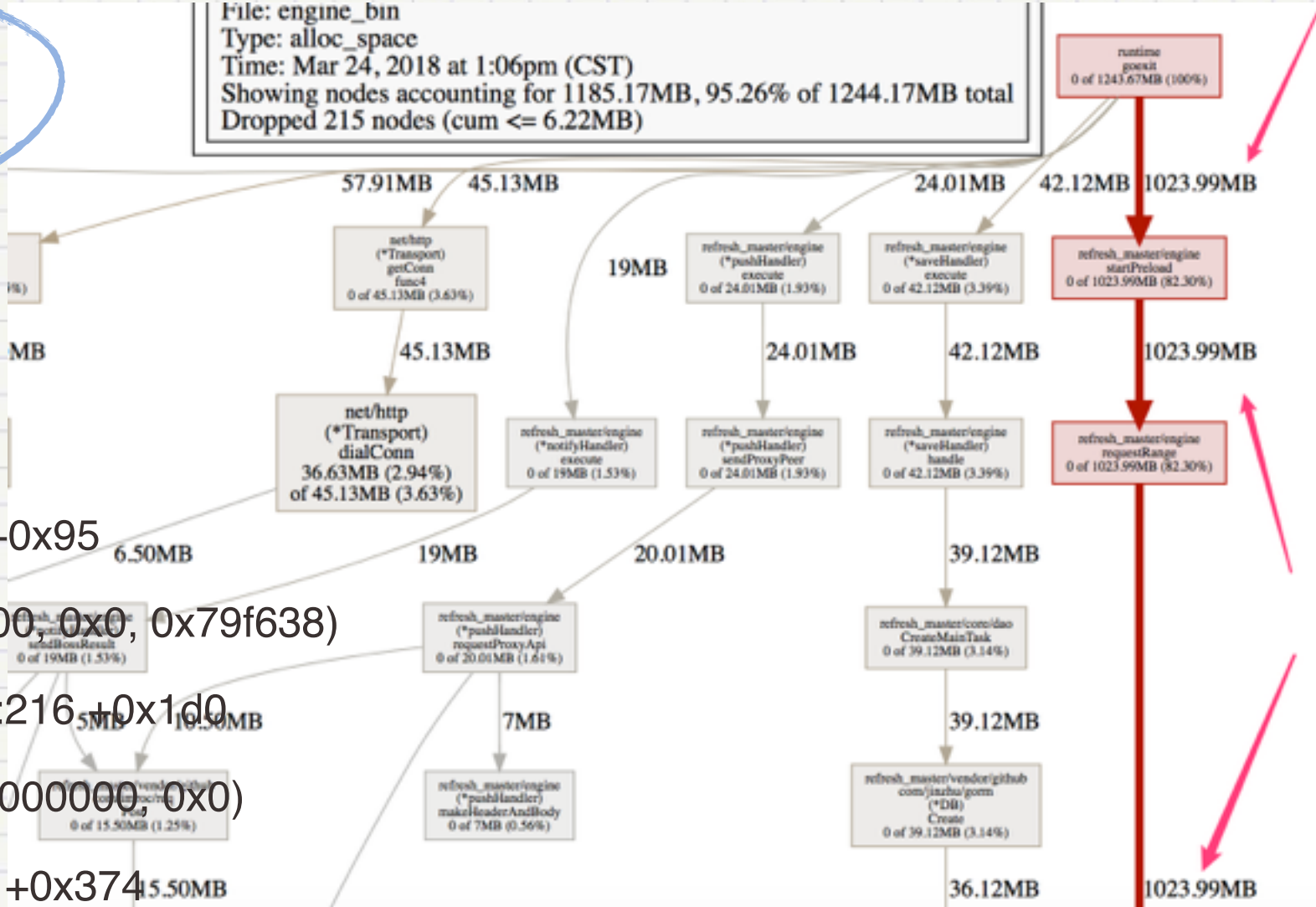
/usr/local/go/src/runtime/panic.go:596 +0x95

runtime.sysMap(0xc4a0200000, 0x80000000, 0x0, 0x79f638)

/usr/local/go/src/runtime/mem_linux.go:216 +0x1d0

runtime.(*mheap).sysAlloc(0x786b20, 0x80000000, 0x0)

/usr/local/go/src/runtime/malloc.go:440 +0x374



深度排雷

SystemTool

- perf top
- lsof、netstat、sar
- go tool pprof
- strace
- iostat、vmstat、top
- wrk

Established 暴增问题

```
task_disp 14138 root  130u  IPv4 739219083      0t0  TCP 10.3[redacted]:17170->[redacted] (ESTABLISHED)
task_disp 14138 root  132u  IPv4 739219051      0t0  TCP 10.3[redacted]:17106->[redacted] (ESTABLISHED)
task_disp 14138 root  136u  IPv4 739219055      0t0  TCP 10.3[redacted]:17114->[redacted] (ESTABLISHED)
task_disp 14138 root  137u  IPv4 739219056      0t0  TCP 10.3[redacted]:17116->[redacted] (ESTABLISHED)
task_disp 14138 root  138u  IPv4 739219057      0t0  TCP 10.3[redacted]:17118->[redacted] (ESTABLISHED)
task_disp 14138 root  139u  IPv4 739219058      0t0  TCP 10.3[redacted]:17120->[redacted] (ESTABLISHED)
task_disp 14138 root  140u  IPv4 739219059      0t0  TCP 10.3[redacted]:17122->[redacted] (ESTABLISHED)
task_disp 14138 root  141u  IPv4 739219060      0t0  TCP 10.3[redacted]:17124->[redacted] (ESTABLISHED)
```

```
netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'
```

```
TIME_WAIT 52
CLOSE_WAIT 5011
FIN_WAIT2 12
ESTABLISHED 19303
LAST_ACK 25
```

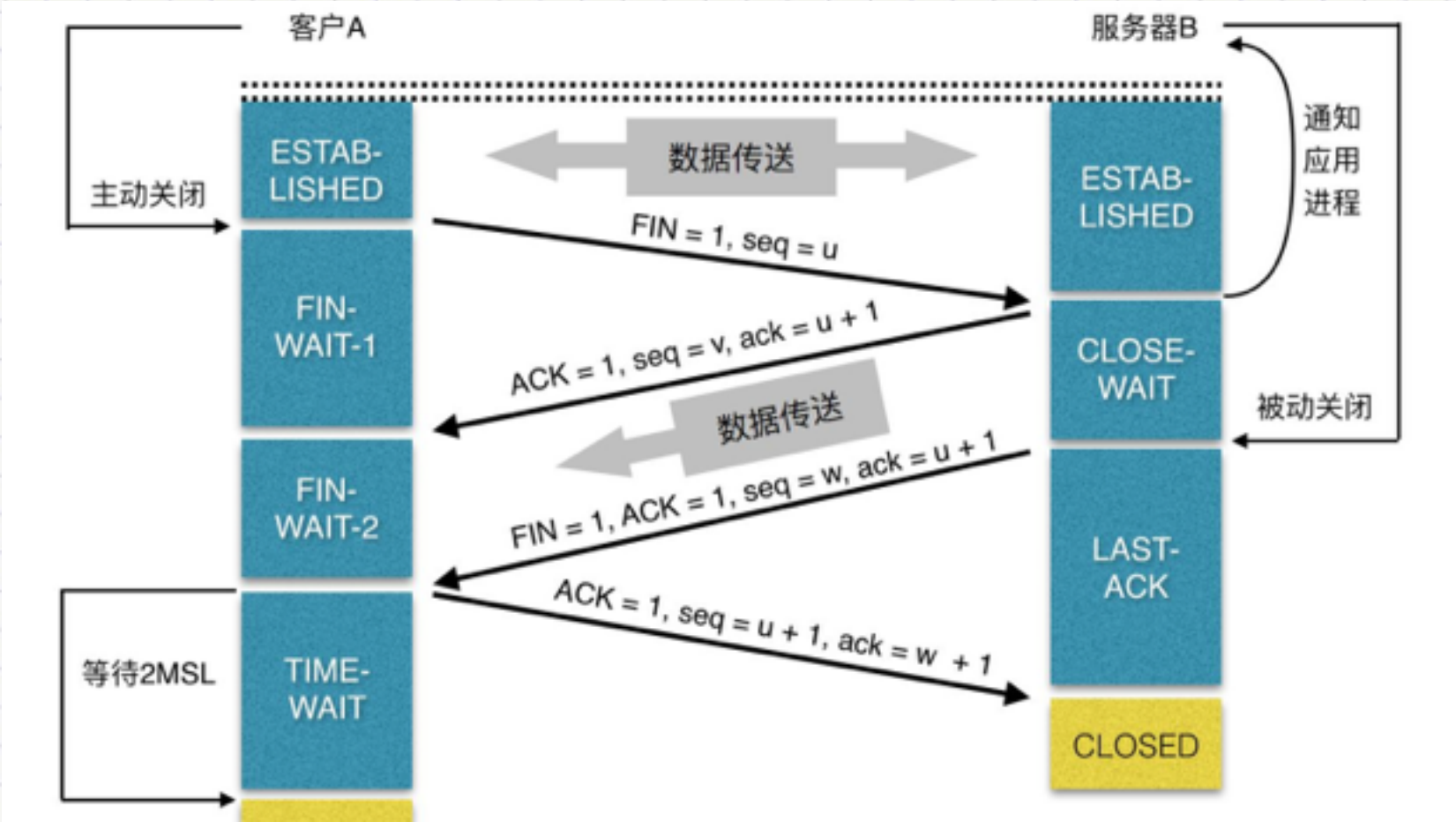
TimeWait 连接暴增问题

TimeWait 暴增到 3.5W ...

TimeWait 暴增到 5W ...

TimeWait 暴增到 3W ...

TimeWait 暴增到 4W ...



TimeWait 连接暴增问题

TimeWait 暴增到 3.5W ...

TimeWait 暴增到 5W ...

TimeWait 暴增到 3W ...

TimeWait 暴增到 4W ...

```
http.DefaultTransport.(*http.Transport).MaxIdleConnsPerHost = 200

const DefaultMaxIdleConnsPerHost = 2
t := &http.Transport{
    ...
    DialContext: (&net.Dialer{
        ....
    }).DialContext,
    MaxIdleConnsPerHost:   200,
    MaxIdleConns:          1000,
    ...
```

系统线程暴涨

```
  wsec/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
113328.00   809.49   145.55  602.14    0.00  602.14   7.14 100.00
    0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
    0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00

  wsec/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
111480.00   995.36   132.68 1184.48    0.00 1184.48   8.93 100.00
    0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
    0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
```

profiles:

0 block

398 goroutine

760 heap

0 mutex

1009 threadcreate

full goroutine stack dump

系统线程暴涨

```
strace -fp pid -s 512
```

```
write(8, ……
mmap(NULL, 10489856, PROT_READ|PROT_...
mprotect(0x7f04d1e6d000, 4096,...
clone(child_stack=0x7f04d286cff0,
...
clone(child_stack=0x7f04d286cff0,
```

- 线程不会被回收
- 每个线程8M内存
- 增加调度消耗
- cpu cache miss
- more

```
lsof -P -p pid -n
```

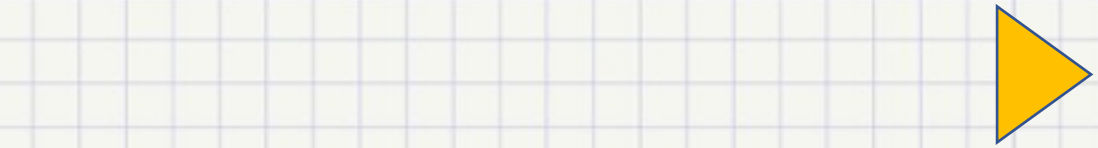
```
refresh_master 2184 root    8u   REG    ... refresh_master/logs/http.log
```

忙轮询

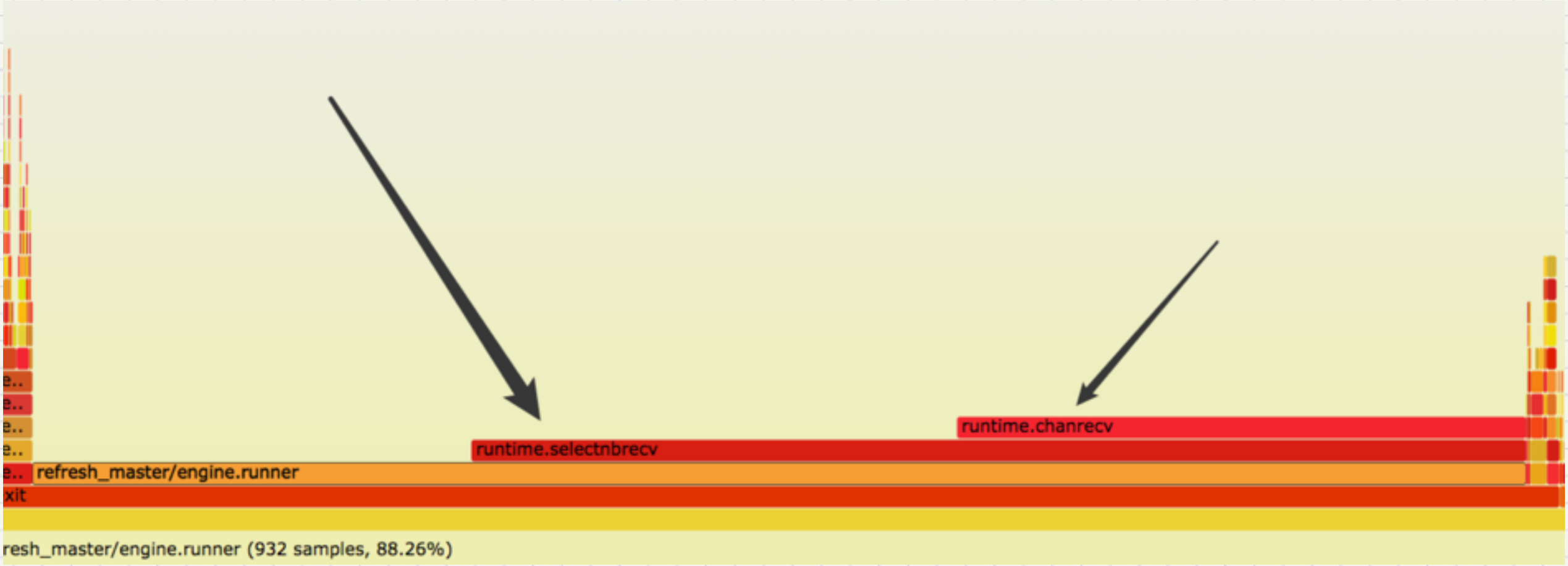
```
...
for {
    select {
        case res = <-dataQueue:
            retrun res
        case <-done:
            return res
        default:
            // do something
    }
    return res
}
...
```

```
Samples: 78K of event 'cpu-clock', Event count (approx.): 121561151512
Overhead  Shared Object       Symbol
  49.09%  k                   [.] runtime.chanrecv
  25.58%  k                   [.] runtime.selectnbrecv
  17.98%  k                   [.] main.runner
   1.01%  [kernel]            [k] finish_task_switch
```

- perf top 可以看出 select chan奇高
- strace 几乎无调用, 单纯cpu指令
- top 直接 100%



忙轮询


runtime.chanrecv
runtime.selectnbrecv
refresh_master/engine.runner
xit
resh_master/engine.runner (932 samples, 88.26%)

Json性能问题


enc..
enc..
encod..
encod..
encodi..
encoding..
encoding/json..
encoding/json..
encoding/json...
en..
i..
encoding/json.Unma..
i..
refresh_master/core/..
ref..
refresh_master/rest.handlePushTask
fm..
refresh_master/vendor/gopkg.in/gin-gonic/gin%2ev1.(*Co..
refresh_master/vendor/gopkg.in/gin-gonic/gin%2ev1.LoggerWithWri..
scall.Sy..
refresh_master/vendor/gopkg.in/gin-gonic/gin%2ev1.(*Context).Next
s..
s..
s..
o..
o..
ref..
refres..
refresh..
r..
e..
e..
e..
e..
enco..
refre..
refre..

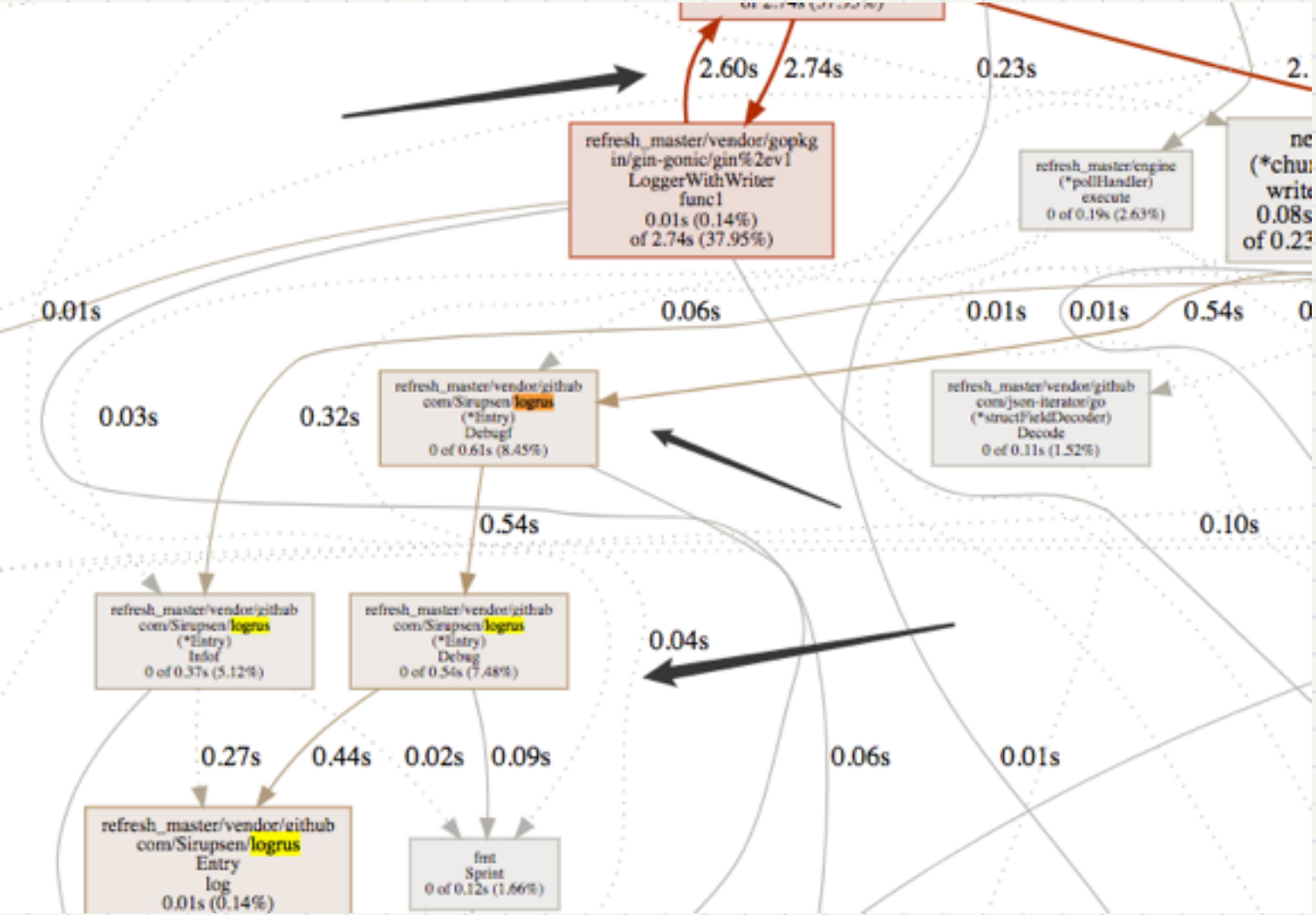
Json 性能问题

- 固定数据结构
  - json-iterator/go
- 不固定数据结构
  - buger/jsonparser

减少反射，减少对象创建 !!!

- 上层代码
  - 直接替换
- 第三方库修改方法
  - 修改Vendor相关库的引用 .
  - 使用Monkey的替换标准库 .

pprof profile



追踪耗时的相关逻辑

由于日志级别过低
导致调用过于频繁 .

压测

- 压测工具
  - wrk > ab > golang
- 不要本地环路测试
- 多源压力
- 注意带宽问题
- 注意内核的种种error
- more…

```
Mar 15 16:56:55 xxx kernel: nf_conntrack: table full, dropping packet.
Mar 15 16:56:55 xxx kernel: nf_conntrack: table full, dropping packet.
Mar 15 16:56:55 xxx kernel: nf_conntrack: table full, dropping packet.

Mar 10 13:45:11 xxxx kernel: TCP: time wait bucket table overflow
Mar 10 13:45:11 xxxx kernel: TCP: time wait bucket table overflow
Mar 10 13:45:11 xxxx kernel: TCP: time wait bucket table overflow

more ….
```

Wrk

```
wrk -t8 -c800 -d100s -T10s --script=js --latency http://xxxxx:9181/v1/engine/task
```

```
thread addr: xxxxx
...
  8 threads and 800 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    19.60ms   11.97ms 144.14ms   73.42%
    Req/Sec     23.73k   2784.29     29.85k    79.29%
  Latency Distribution
     50%   17.90ms
     75%   24.92ms
     90%   34.63ms
...
Requests/sec:   23189.11
Transfer/sec:    5.06MB
```

4 上线部署

打包方法

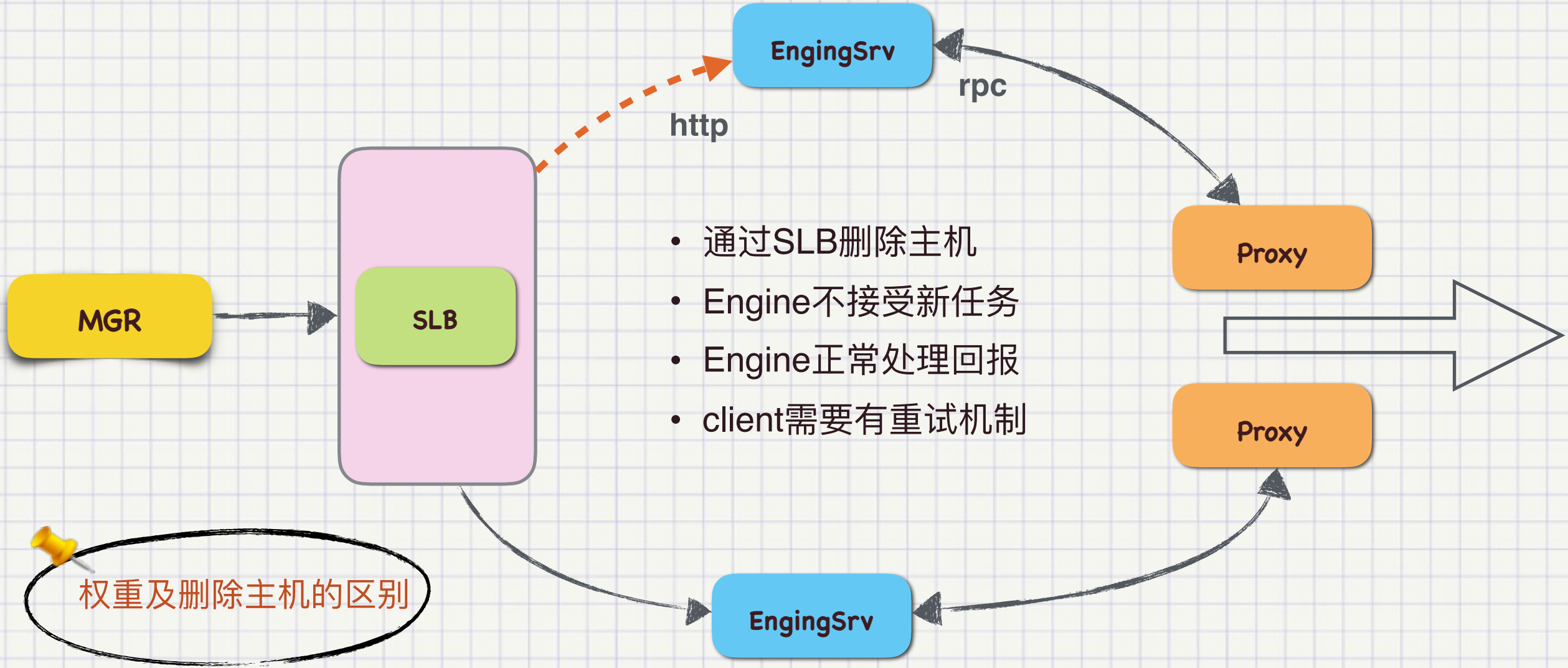
Enging端

- build
- pack
- repo
- salt
- unpack
- reload

Proxy端，封装RPM包上线 .

```
go build -ldflags "-X main.Version=V0.1.10" main.go
```

灰度平滑迁移-SLB层



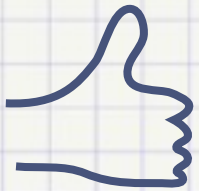
- 通过SLB删除主机
- Engine不接受新任务
- Engine正常处理回报
- client需要有重试机制

权重及删除主机的区别

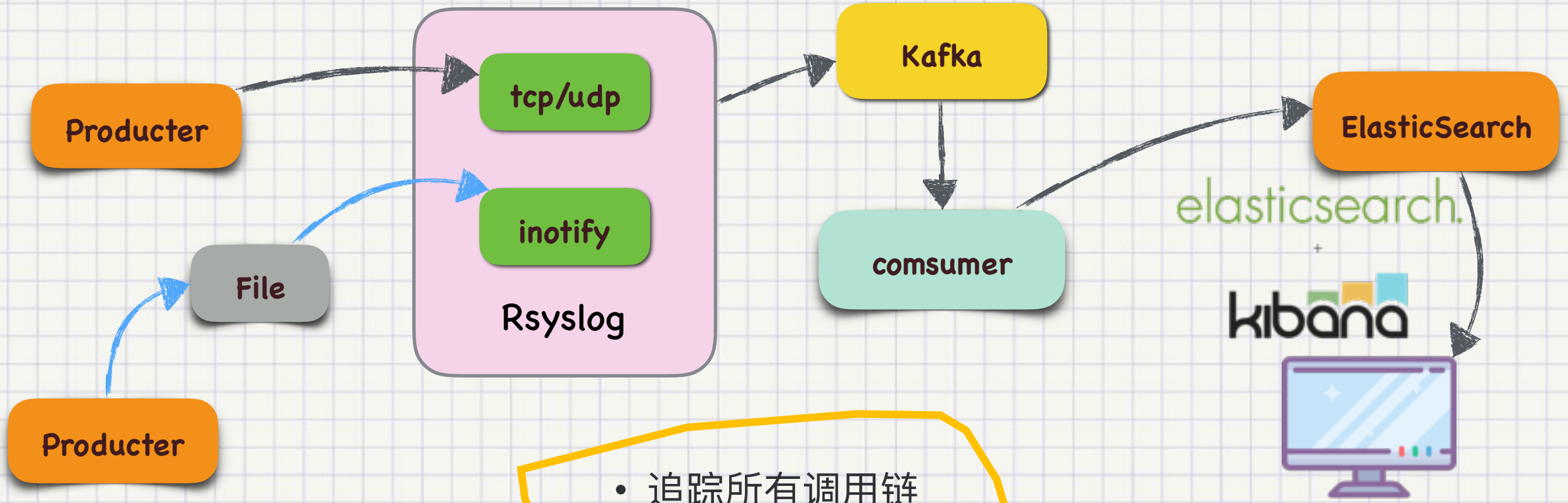
灰度平滑迁移-框架层

- 从调用链最前端开始顺序关闭.
- 持久化的数据要优先保证
- join住所有的业务协程.

```
# 伪代码
recvAndSetExit(signal)
publishSigWorerks()

if globalExited && len(queue) == 0 && pushExit {
        return
}
wait(saveQueueExit)
wait(eventBus)
```



收集分布式日志



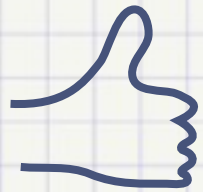
- 追踪所有调用链
- 计算耗时
- 排查问题

如何报警通知

服务本身主动去上报异常！

异步去上报！

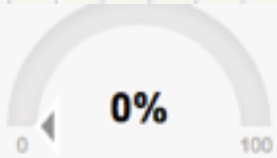
异常抽样报警！

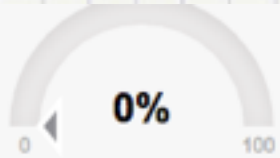


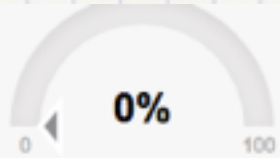
状态监控页面

模块心跳

Module	Status	Diff	LastTime
poller	✔	2	2018-03-23 14:29:13
rester	✔	2	2018-03-23 14:29:13
scaner	✔	1	2018-03-23 14:29:14
[redacted]	[redacted]	[redacted]	[redacted]







节点心跳

Module	Status	Diff	LastTi
[redacted]	✔	0	2018-0
[redacted]	✔	0	2018-0
[redacted]	✔	0	2018-0
[redacted]	✔	0	2018-0
[redacted]	✔	2	2018-0
[redacted]	⛔	597[redacted]	2018-0
[redacted]	⛔	597[redacted]	2018-0
[redacted]	⛔	597[redacted]	2018-0
[redacted]	⛔	597[redacted]	2018-0

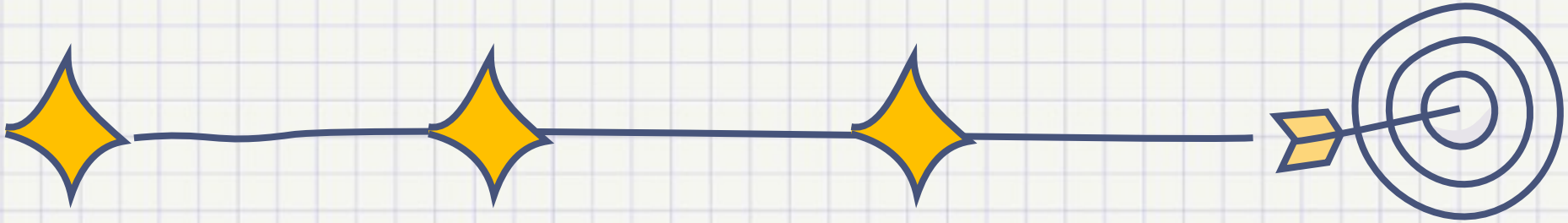
API_ID计数器

API_ID	INIT	PENDING	RUNNIN
[redacted]	0	0	0
[redacted]	0	0	0
[redacted]	0	0	0
[redacted]	0	0	0
[redacted]	0	0	0
[redacted]	0	0	0
[redacted]	0	0	0

5

总结

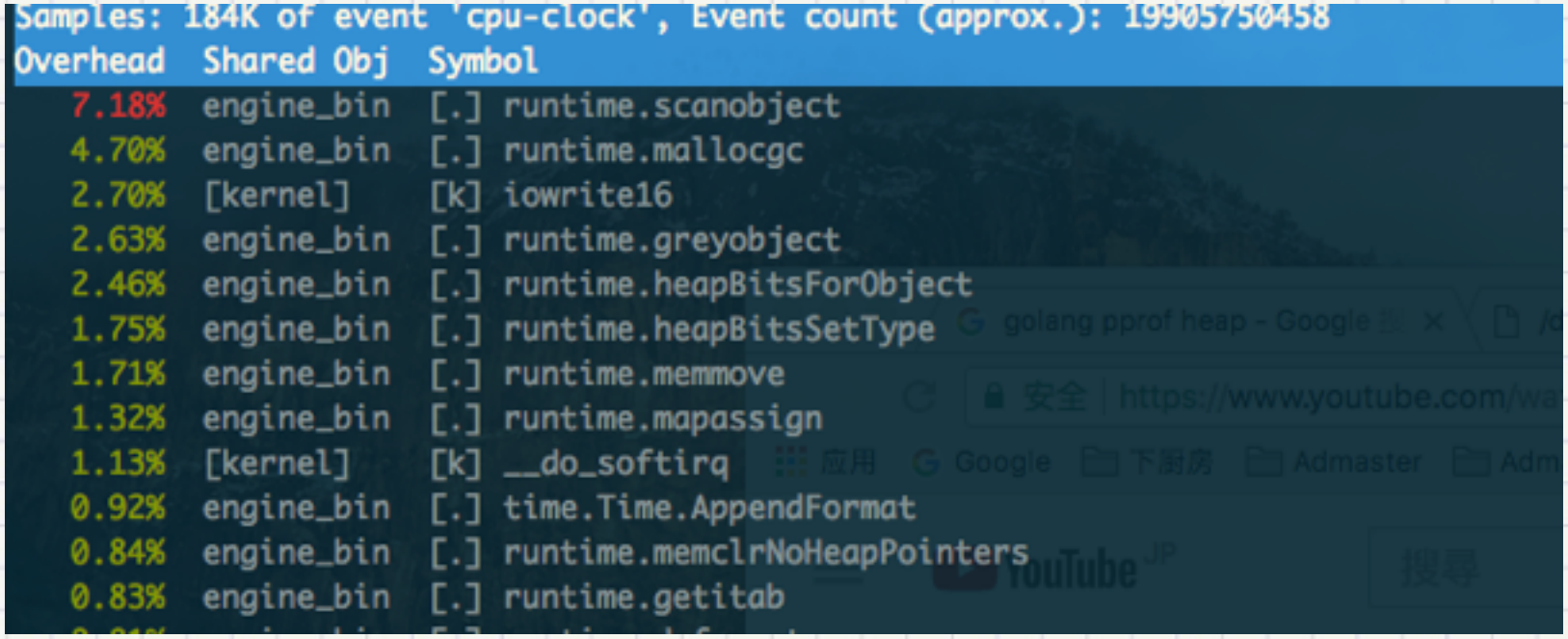
性能演进

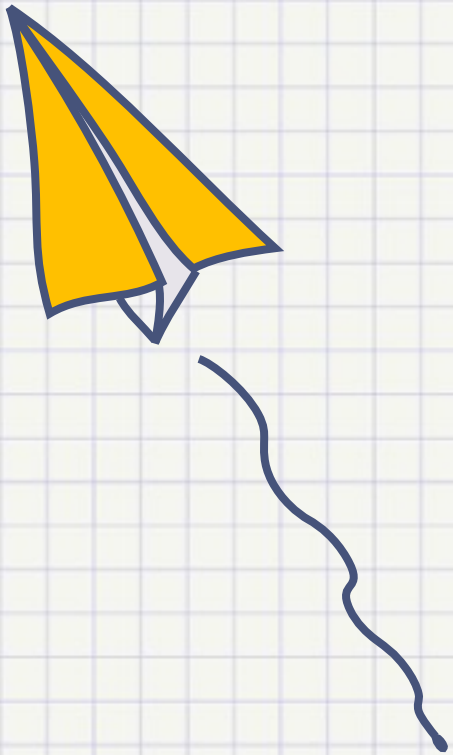


1w 1.5w 2w 3w

QPS

GC 还需优化


Samples: 184K of event 'cpu-clock', Event count (approx.): 199005750458
Overhead Shared Obj Symbol
7.18% engine_bin [.] runtime.scanobject
4.70% engine_bin [.] runtime.mallocgc
2.70% [kernel] [k] iowrite16
2.63% engine_bin [.] runtime.greyobject
2.46% engine_bin [.] runtime.heapBitsForObject
1.75% engine_bin [.] runtime.heapBitsSetType
1.71% engine_bin [.] runtime.memmove
1.32% engine_bin [.] runtime.mapassign
1.13% [kernel] [k] __do_softirq
0.92% engine_bin [.] time.Time.AppendFormat
0.84% engine_bin [.] runtime.memclrNoHeapPointers
0.83% engine_bin [.] runtime.getitab



Q & A