GOPHER CHINA 2020

中国 上海 / 2020-11.21-22

# 云原生go-zero微服务框架设计思考

万俊峰Kevin@好未来

# 关于我

万俊峰Kevin

- go-zero作者
- 好未来资深专家
- 晓黑板研发负责人
- 十多年研发团队管理经验
- 近20年开发和架构经验

# Agenda

- go-zero之前世今生
- go-zero是如何设计的
- go-zero如何高效解决问题

# go-zero之前世今生

GOPHER CHINA 2020
中国 上海 / 2020-11.21-22


# go-zero的由来

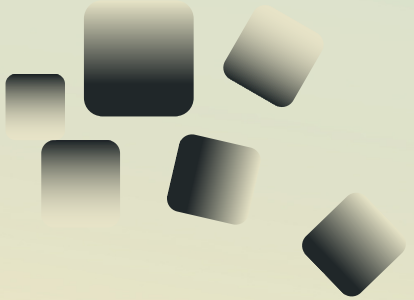
- 单体服务的困局
- 架构的选型
- 如何无痛切换


# go-zero是什么？

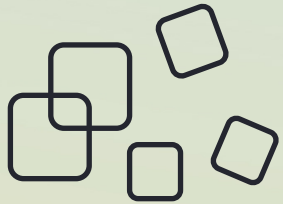
- Web & RPC微服务框架
- 微服务代码生成工具goctl
- 通用API定义规范

# go-zero的设计原则

- 保持简单，第一原则
- 弹性设计，面向故障编程
- 工具大于约定和文档
- 尽可能约束做一件事只有一种方式
- 对业务开发友好，封装复杂度



# go-zero是如何设计的



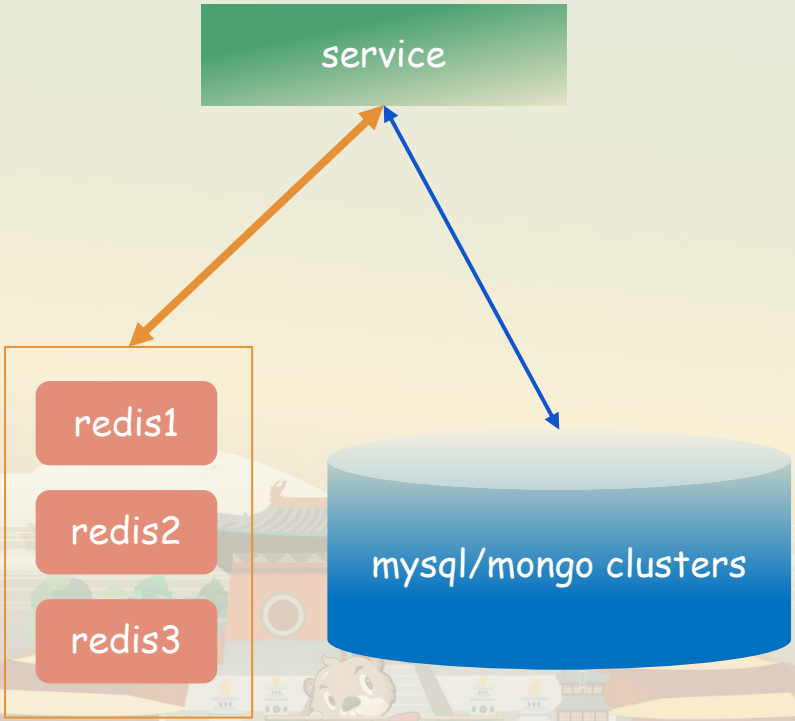
GOPHER CHINA 2020

中国 上海 / 2020-11.21-22


客户端
iOS
安卓
web
PC
API端
HTTP协议
鉴权&防重放
加解密
日志记录
异常捕获
监控报警
数据统计
并发控制
链路跟踪
超时控制
自动熔断
自动降载
服务发现
ETCD集群
Service端
gRPC协议
调用鉴权
缓存控制
日志记录
异常捕获
监控报警
数据统计
并发控制
链路跟踪
超时控制
自动熔断
自动降载
缓存层
Redis集群
Redis集群
Redis集群
数据库
MySQL集群
MongoDB集群
ClickHouse集群

# 代码未动，数据先行

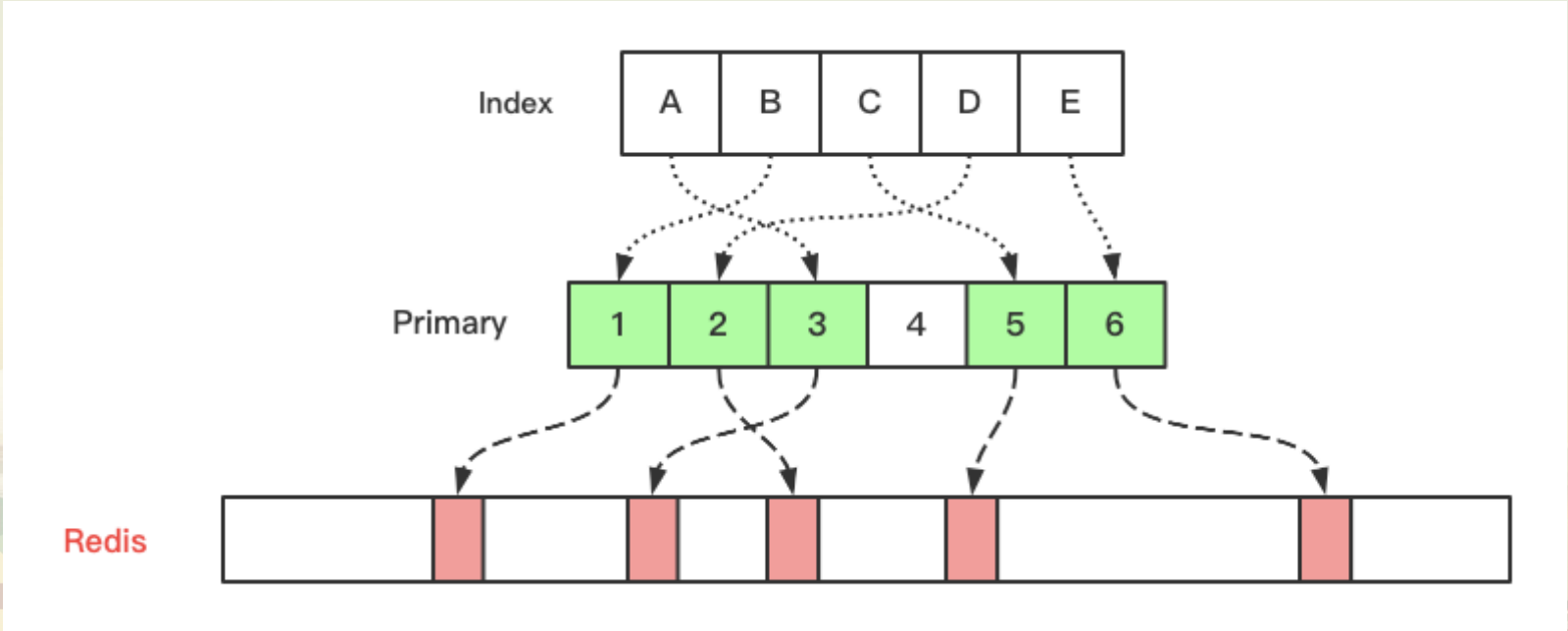
- 定义数据边界
- 数据库互相隔离，通过RPC访问
- No join, no pain!




# 如何设计缓存

- 缓存穿透，不存在的数据
  - 缓存一分钟
- 缓存击穿，热点key过期
  - 只拿一次数据，共享结果
- 缓存雪崩，大量缓存同时过期
  - 过期时间设置随机偏差

# 类似DB的缓存索引方式



# 缓存的最佳实践

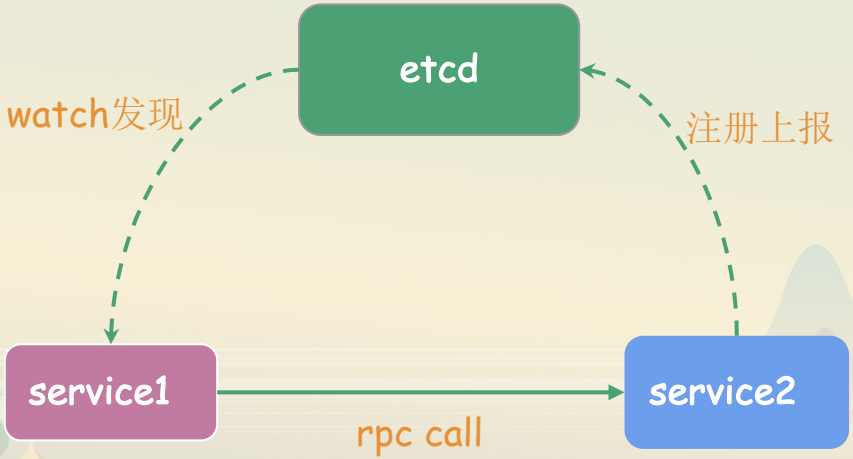
- 不允许不过期的缓存
- 分布式缓存，易伸缩
- 自动生成，自带统计

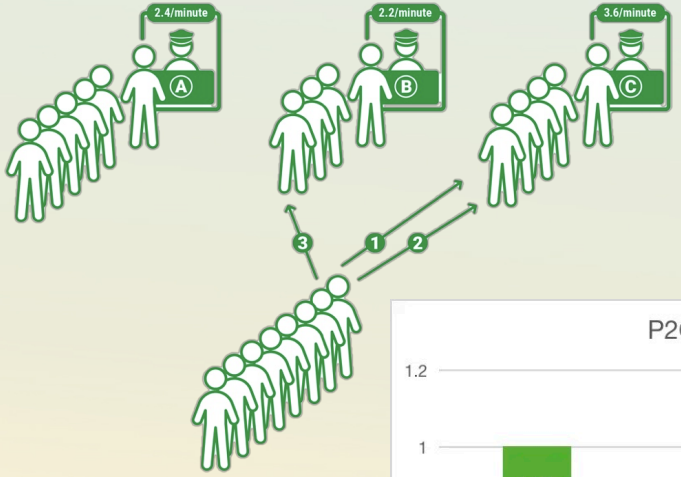
```go
func (m *BookModel) FindOne(book string) (*Book, error) {
    bookKey := fmt.Sprintf( format: "%s%v", bookPrefix, book)
    var resp Book
    err := m.QueryRow(&resp, bookKey, func(conn sqlx.SqlConn, v interface{}) error {
        query := `select ` + bookRows + ` from ` + m.table + ` where book = ? limit 1`
        return conn.QueryRow(v, query, book)
    })
    switch err {
    case nil:
        return &resp, nil
    case sqlc.ErrNotFound:
        return nil, ErrNotFound
    default:
        return nil, err
    }
}
```

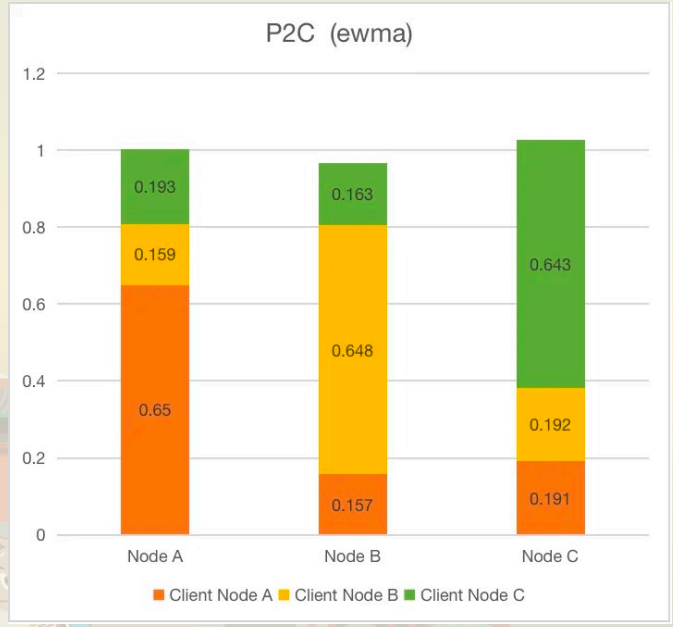
| | | |
|---|---|---|
| t | _index | k8s_pro-2020.11.19 |
| # | _score | - |
| t | _type | doc |
| t | content | dbcache(sqlc) - qpm: 5057, hit_ratio: 99.7%, hit: 5044, miss: 13, db_fails: 0 |
| ? | k8s_cluster | pro4 |

# rpc服务层 - zRPC

- 协议选择 - gRPC
- 服务发现方式 - etcd
- 负载均衡 - p2c ewma
- 支持自定义中间件






Power of Two Choices

- 默认算法
- 当前请求数
- 处理时长
- 指数加权移动平均

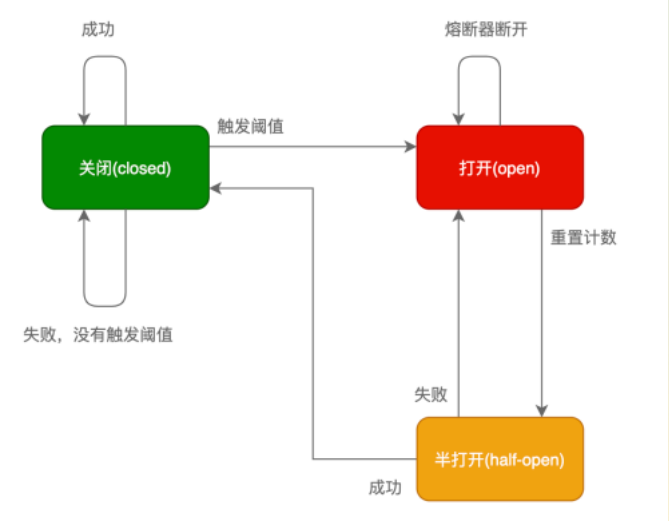
参考自Nginx & Envoy & Finagle & Linkerd：

- https://www.nginx.com/blog/nginx-power-of-two-choices-load-balancing-algorithm/
- https://linkerd.io/2016/03/16/beyond-round-robin-load-balancing-for-latency/

# api gateway层

- 流控
- 请求鉴权
- 请求参数校验
- 业务聚合
- 支持自定义中间件

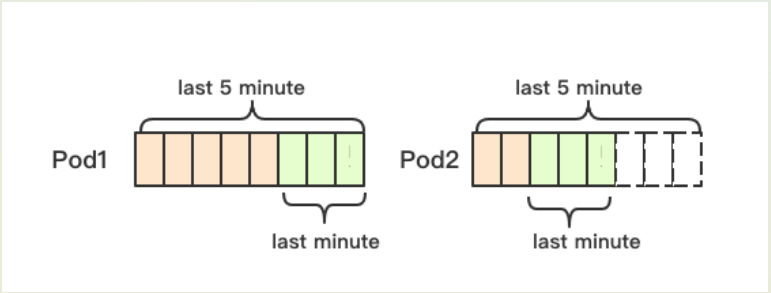
```go
func MaxConns(n int) func(http.Handler) http.Handler {
    if n <= 0 {
        return func(next http.Handler) http.Handler {
            return next
        }
    }

    return func(next http.Handler) http.Handler {
        latch := syncx.NewLimit(n)

        return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
            if latch.TryBorrow() {
                defer func() {
                    if err := latch.Return(); err != nil {
                        logx.Error(err)
                    }
                }()

                next.ServeHTTP(w, r)
            } else {
                internal.Errorf(r, format: "concurrent connections over %d, rejected with code %d",
                    n, http.StatusServiceUnavailable)
                w.WriteHeader(http.StatusServiceUnavailable)
            }
        })
    }
}
```

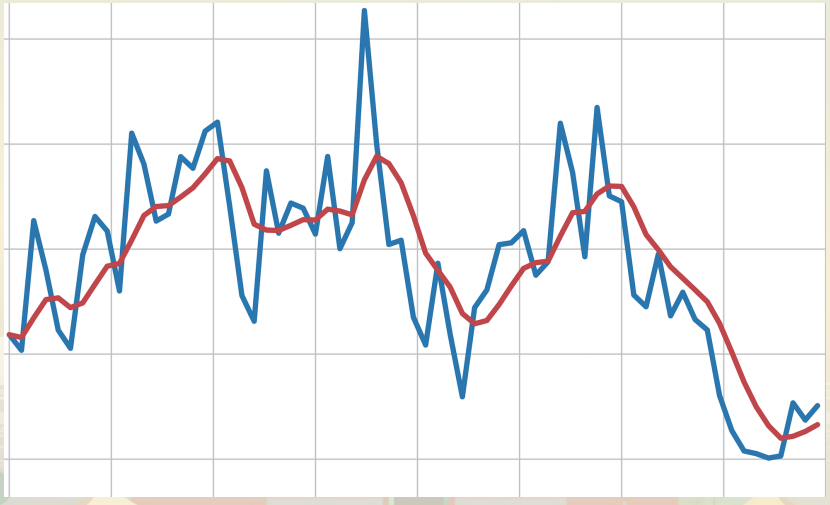



$$dropRatio = max(0, \frac{(requests - protection) - K \times accepts}{requests + 1})$$

# 自适应熔断

- Google SRE算法
- 放弃了Netflix Hystrix算法
- 基于滑动窗口（10秒/40窗口）
- 支持自定义触发条件
- 支持自定义fallback
- http/rpc框架内建
- 自动触发，自动恢复



$$min(InFlight, MovingAvg(InFlight)) > MaxPass \times AvgRT$$

# 自适应降载

- K8S的HPA 80%触发
- CPU>90%开始拒绝低优先级请求
- CPU>95%开始拒绝高优先级请求
- 基于滑动窗口，防止毛刺
- 有冷却时间，防止抖动
- 实践检验，配合K8S弹性伸缩
- http/rpc框架内建

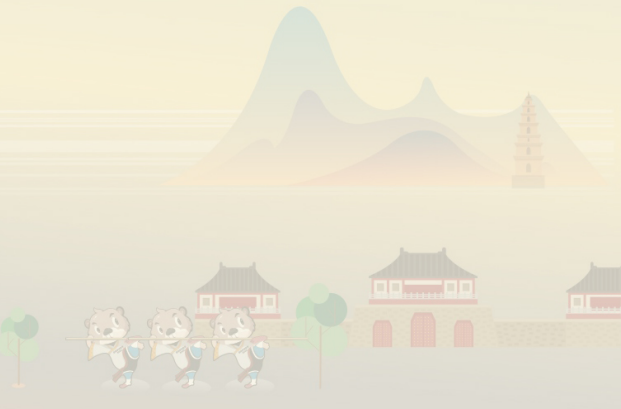
# 更多组件

- 超时
  - 级联调用
  - 跟客户端超时配合
- 重试
  - 指数退避
  - 流量quota
  - 超时相关性

# 多重防护，保障高可用

# 可观测性

- 链路跟踪
- Logging
- Metrics
- 监控报警



dashboard/cluster service easy

Customer Service

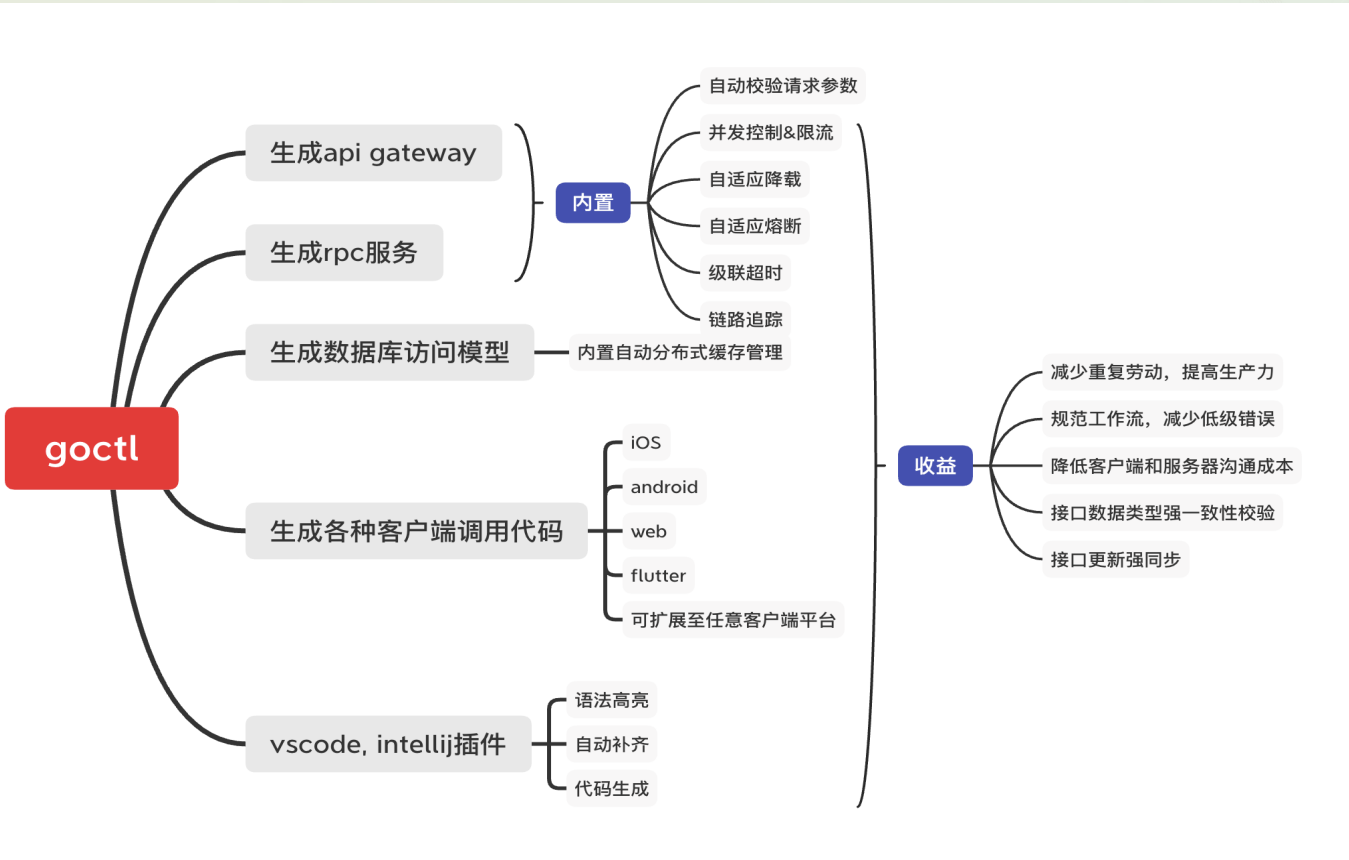
service: easystash-pro, qps: 283009.2/s, drops: 0, avg: 0.9ms, med: 0.0ms, tp90: 0.0ms, tp99: 0.1ms, tp999: 1.1ms


没有度量，就没有优化！

- 数据上报到控制台服务
- 数据上报到prometheus

# go-zero如何高效解决问题

GOPHER CHINA 2020
中国 上海 / 2020-11.21-22


goctl
生成api gateway
生成rpc服务
内置
自动校验请求参数
并发控制&限流
自适应降载
自适应熔断
级联超时
链路追踪
生成数据库访问模型
内置自动分布式缓存管理
生成各种客户端调用代码
iOS
android
web
flutter
可扩展至任意客户端平台
vscode, intellij插件
语法高亮
自动补齐
代码生成
收益
减少重复劳动，提高生产力
规范工作流，减少低级错误
降低客户端和服务器沟通成本
接口数据类型强一致性校验
接口更新强同步

```
type Request {
    Name string `path:"name"`
}

type Response {
    Message string `json:"message"`
}

service greet-api {
    @handler NoResponseHandler
    get /greet/message

    @handler PutUserHandler
    put /greet/users/:name(Request)

    @handler NoRequestHandler
    delete /greet/talk returns (Response)

    @handler NormalHandler
    post /greet/from/:name(Request) returns (Response)
}
```

# http请求自动解析校验

```
type Request {
    Name string `path:"name"`
    Age int `form:"age,range=[18:)"`
    Address string `json:"address,optional"`
    Role string `json:"role,default=guest"`
    Gender string `json:"gender,options=male|female"`
}
```

## 支持的特性

- tag支持：path, form, json
- default, optional, options, range

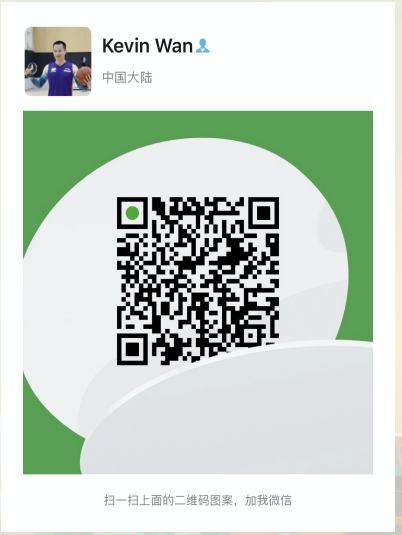
## 解析校验

- httpx.Parse(...)
- 参数错误自动返回400
- 支持自定义错误返回方式

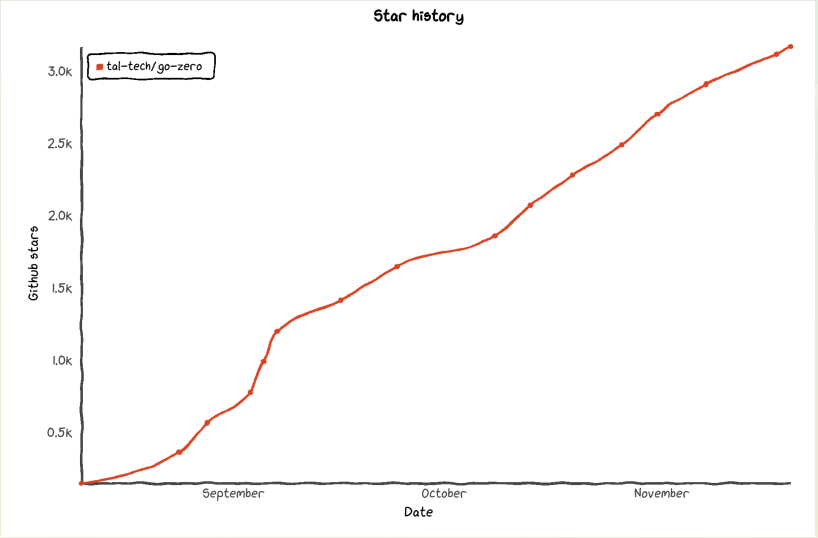
```
[dev] ➜  gopherchina goctl api new hello
Done.
[dev] ➜  gopherchina cd hello
[dev] ➜  hello tree
.
├── etc
│   └── hello-api.yaml
├── go.mod
├── hello.api
├── hello.go
└── internal
    ├── config
    │   └── config.go
    ├── handler
    │   ├── hellohandler.go
    │   └── routes.go
    ├── logic
    │   └── hellologic.go
    ├── svc
    │   └── servicecontext.go
    └── types
        └── types.go

7 directories, 10 files
[dev] ➜  hello go run hello.go -f etc/hello-api.yaml
go: finding module for package github.com/tal-tech/go-zero/core/conf
go: finding module for package github.com/tal-tech/go-zero/rest
go: finding module for package github.com/tal-tech/go-zero/rest/httpx
go: finding module for package github.com/tal-tech/go-zero/core/logx
go: found github.com/tal-tech/go-zero/core/conf in github.com/tal-tech/go-zero v1.0.27
go: found github.com/tal-tech/go-zero/rest in github.com/tal-tech/go-zero v1.0.27
go: found github.com/tal-tech/go-zero/rest/httpx in github.com/tal-tech/go-zero v1.0.27
go: found github.com/tal-tech/go-zero/core/logx in github.com/tal-tech/go-zero v1.0.27
Starting server at 0.0.0.0:8888...
```

..p/gopherchina (-zsh)

```
[dev] ➜  gopherchina wrk -t10 -c1000 -d10s --latency "http://localhost:8888/greet/from/me"
Running 10s test @ http://localhost:8888/greet/from/me
  10 threads and 1000 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     3.74ms  622.36us   8.61ms   74.17%
    Req/Sec     6.44k     2.75k   11.79k    57.20%
  Latency Distribution
     50%    3.73ms
     75%    4.06ms
     90%    4.45ms
     99%    5.57ms
  641238 requests in 10.01s, 107.63MB read
  Socket errors: connect 759, read 99, write 0, timeout 0
  Non-2xx or 3xx responses: 641238
Requests/sec:  64078.90
Transfer/sec:     10.76MB
[dev] ➜  gopherchina
```








https://github.com/tal-tech/go-zero

https://zero.gocn.vip

欢迎 star, fork, issue, PR! 👏

# GOPHER CHINA 2020

中国 上海 / 2020-11.21-22

## 谢谢！

*简单，是终极的复杂！*