探探微服务架构演进之路

彭亮

Agenda

- Why do we need microservices
- What are microservices
- How to “micro” services
- go and microservices

Why do we need microservices?

Why microservice

模块边界耦合

- 代码耦合度高
- 团队急剧扩张，职能划分不清晰

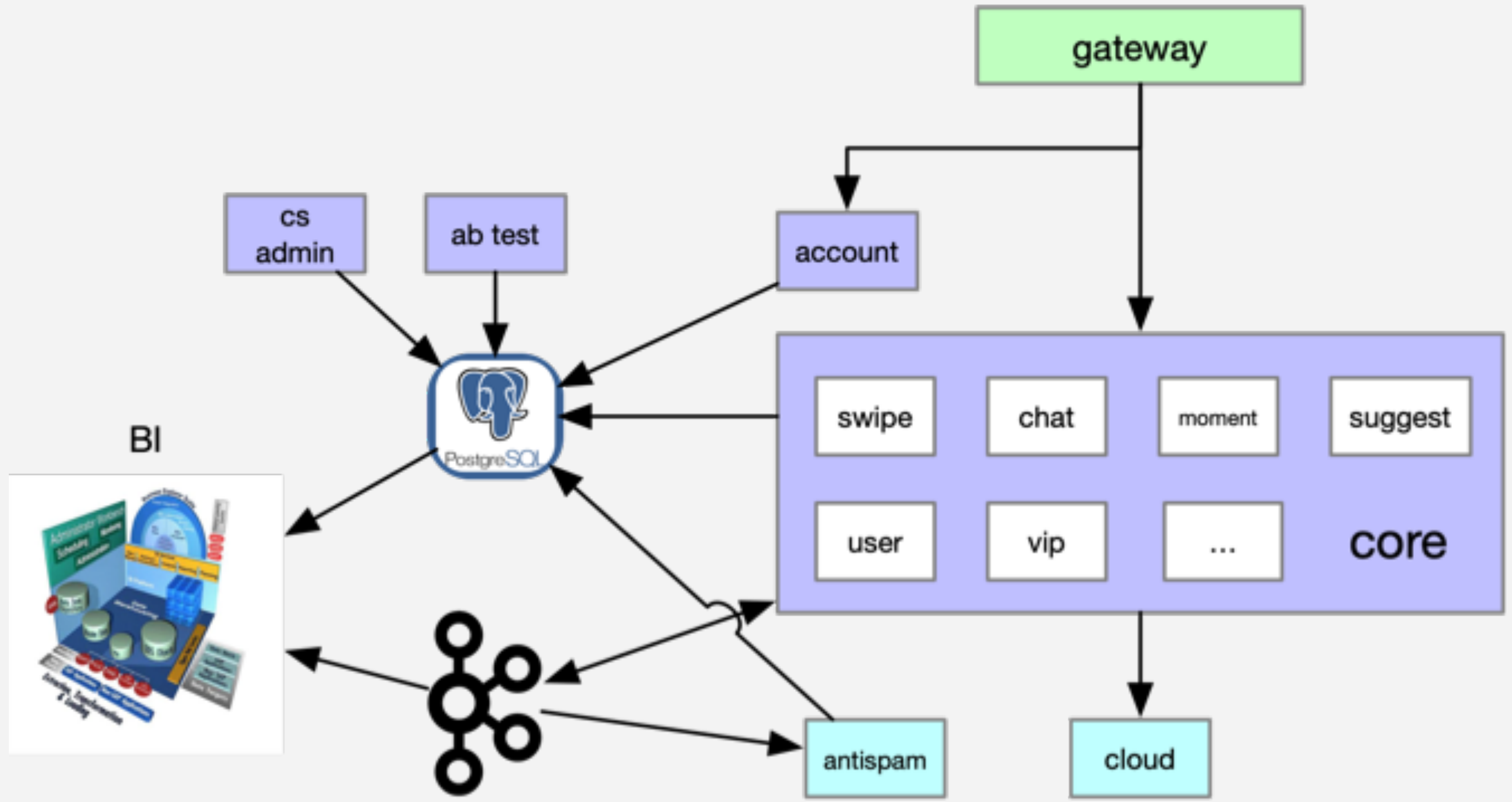
技术带宽限制

- 无法满足业务高速发展

技术创新和债务

- 引入新语言，框架，技术栈难度高
- 存储共用
- 部署时间长，影响大

后端服务架构


gateway
cs admin
ab test
account
BI
PostgreSQL
swipe
chat
moment
suggest
user
vip
...
core
antispam
cloud

What are microservices?

微服务架构有多 “微” ?

敏捷开发专家Martin Fowler 定义了微服务架构：

In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

- 单一职责
- 服务自治
- 实现敏捷开发

微服务不是银弹

开发

- 开发复杂度变高
- 接口联调更难：链路过长，接口变更
- 沟通成本高：接口定义，组间合作

测试

- 单元集成测试更复杂
- 测试环境复杂：不同的基础设施
- 测试环境脆弱

运维

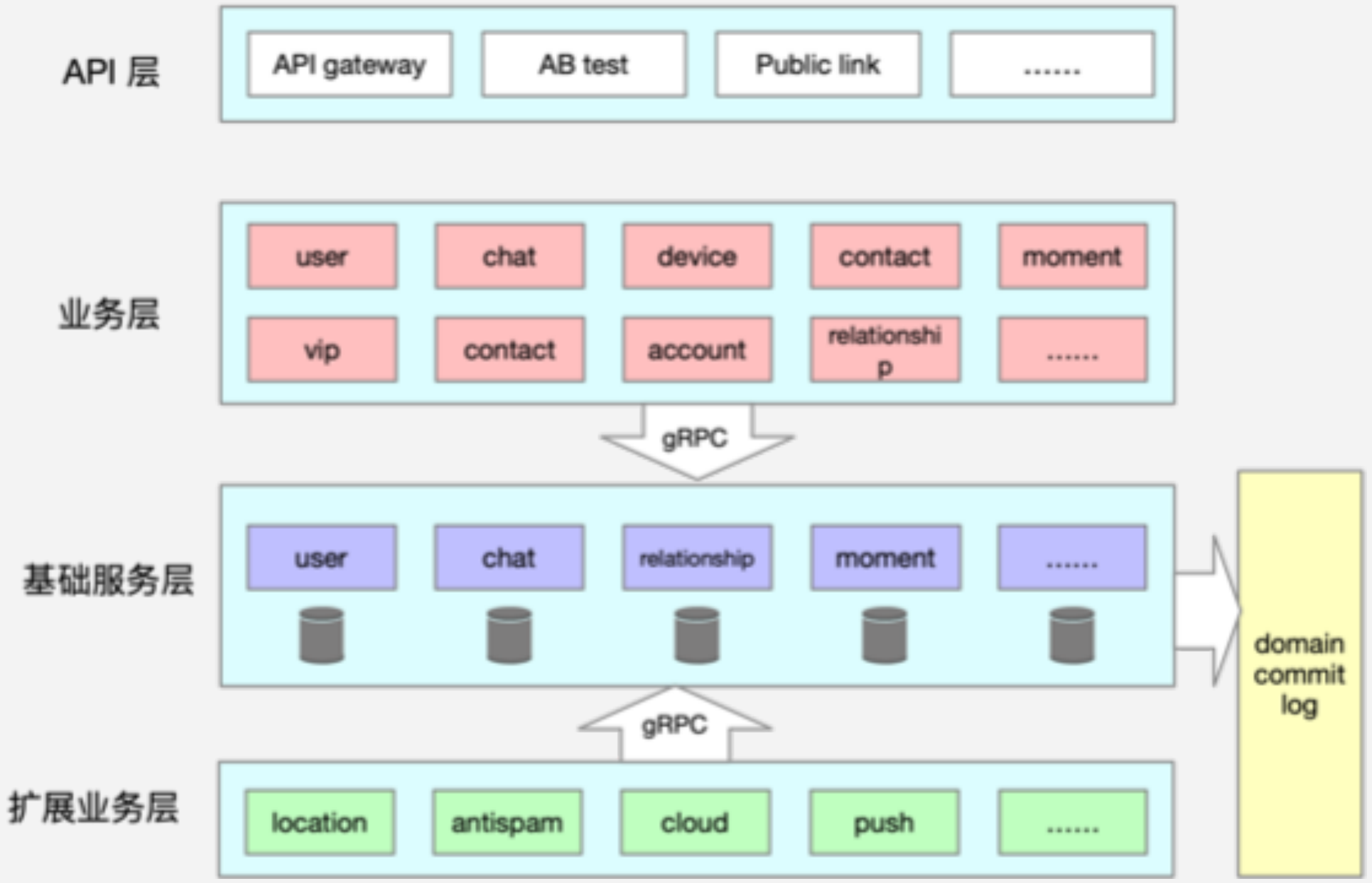
- 问题排查难
- 运维复杂
- 服务高可用更难保障

How to “micro” services ?

组织架构划分，团队职能划分

聊天	朋友圈	推荐
用户中心	反垃圾	商业化
基础服务	架构	平台

业务梳理，边界划分


API 层
API gateway
AB test
Public link
....
业务层
user
chat
device
contact
moment
vip
contact
account
relationshi
p
....
gRPC
基础服务层
user
chat
relationship
moment
....
gRPC
扩展业务层
location
antispam
cloud
push
....
domain
commit
log

service communication

RPC

thrift:

transport：Socket,File,Zip,HTTP

protocol：binary, compact, json

vs

gRPC:

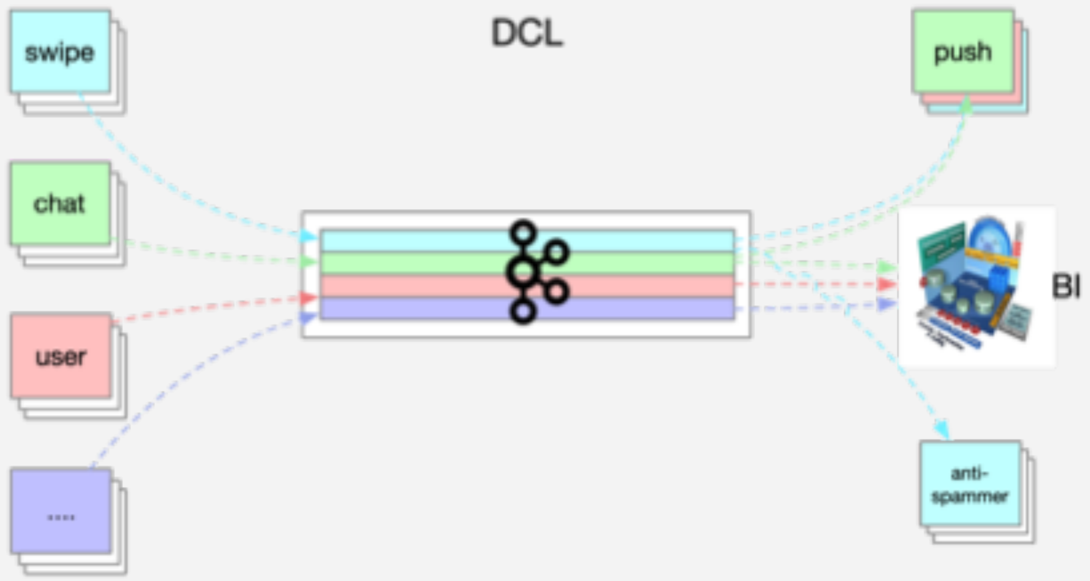
protocol：protobuf,

transport: http2

streaming

DCL(domain commit log)

- 解耦：可以通过发布订阅模式，发布DCL，让订阅者自行订阅数据变更的DCL
- 最终一致性：通过DCL来达到最终一致性，提高系统的稳定性和性能
- 事件溯源



DCL的产生

应用驱动双向写

- 一致性问题：
  分布式事务
- 延迟

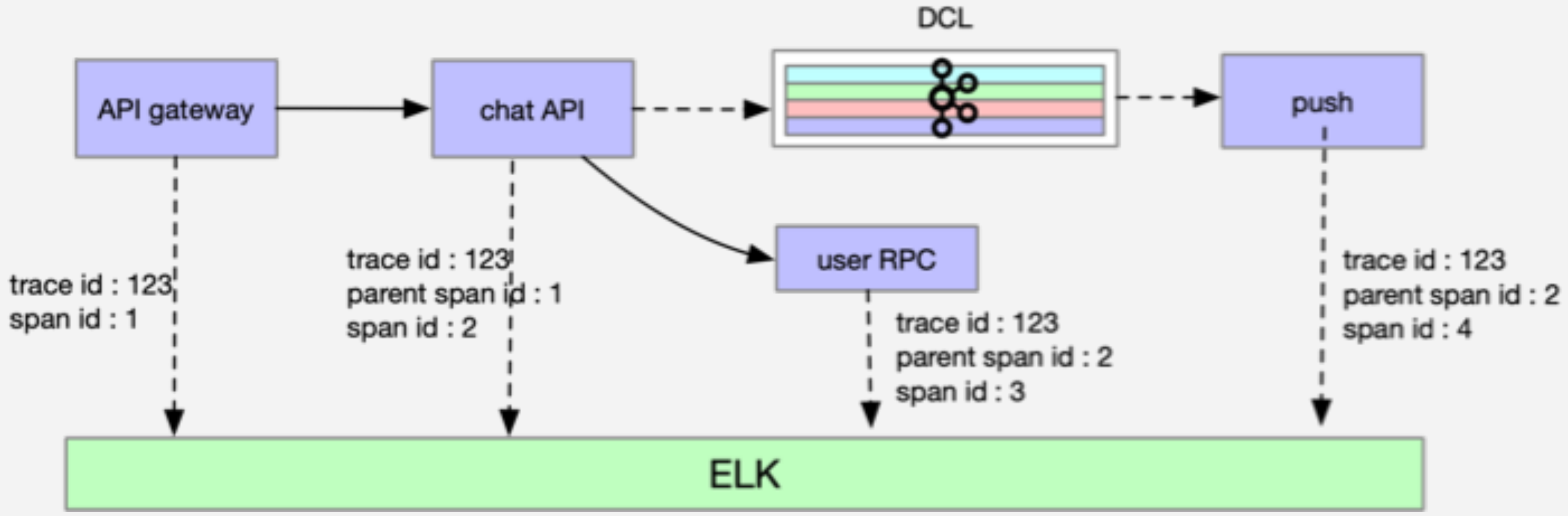
VS

数据变更捕获 change data capture

- 大部分业务db基于物理复制
- table格式变更的频繁

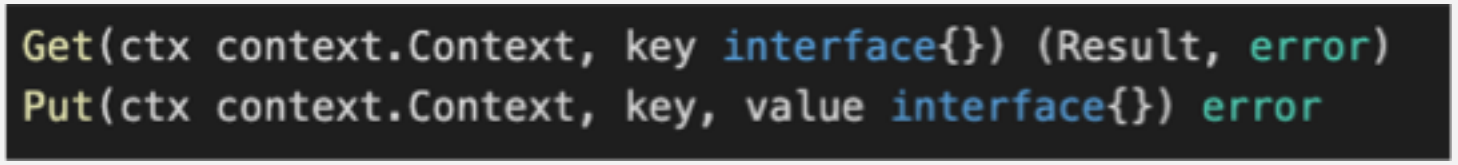
tracing

日志上下文


DCL
API gateway
chat API
push
user RPC
trace id : 123
span id : 1
trace id : 123
parent span id : 1
span id : 2
trace id : 123
parent span id : 2
span id : 3
trace id : 123
parent span id : 2
span id : 4
ELK

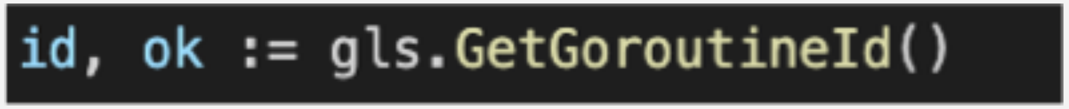
进程内上下文

更改接口



```
Get(ctx context.Context, key interface{}) (Result, error)
Put(ctx context.Context, key, value interface{}) error
```

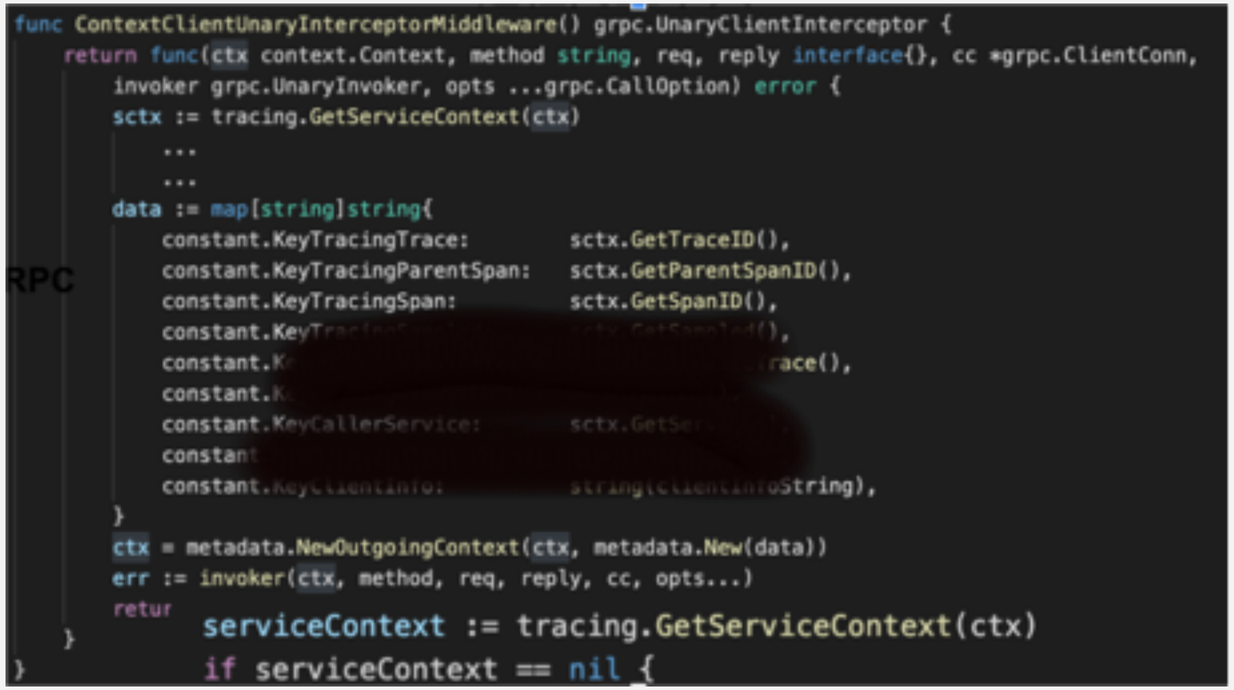
goroutine局部存储



```
id, ok := gls.GetGoroutineId()
```

进程间上下文

RPC

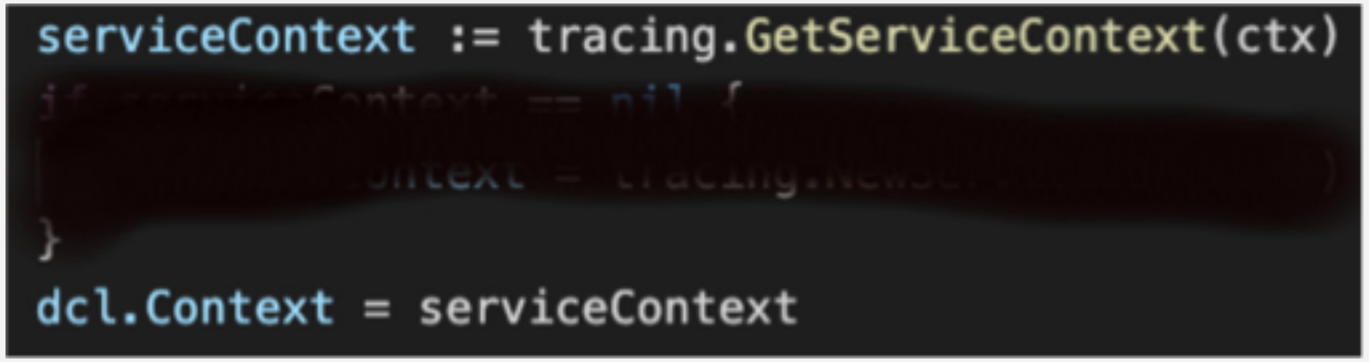


RPC

```
func ContextClientUnaryInterceptorMiddleware() grpc.UnaryClientInterceptor {
    return func(ctx context.Context, method string, req, reply interface{}, cc *grpc.ClientConn,
        invoker grpc.UnaryInvoker, opts ...grpc.CallOption) error {
        sctx := tracing.GetServiceContext(ctx)
            ...
            ...
        data := map[string]string{
            constant.KeyTracingTrace:        sctx.GetTraceID(),
            constant.KeyTracingParentSpan:   sctx.GetParentSpanID(),
            constant.KeyTracingSpan:         sctx.GetSpanID(),
            constant.Key[redacted]
            constant.                                  ace(),
            constant.
            constant.
            constant
            constant.                                  String),
        }
        ctx = metadata.NewOutgoingContext(ctx, metadata.New(data))
        err := invoker(ctx, method, req, reply, cc, opts...)
        retur
    }
}
```

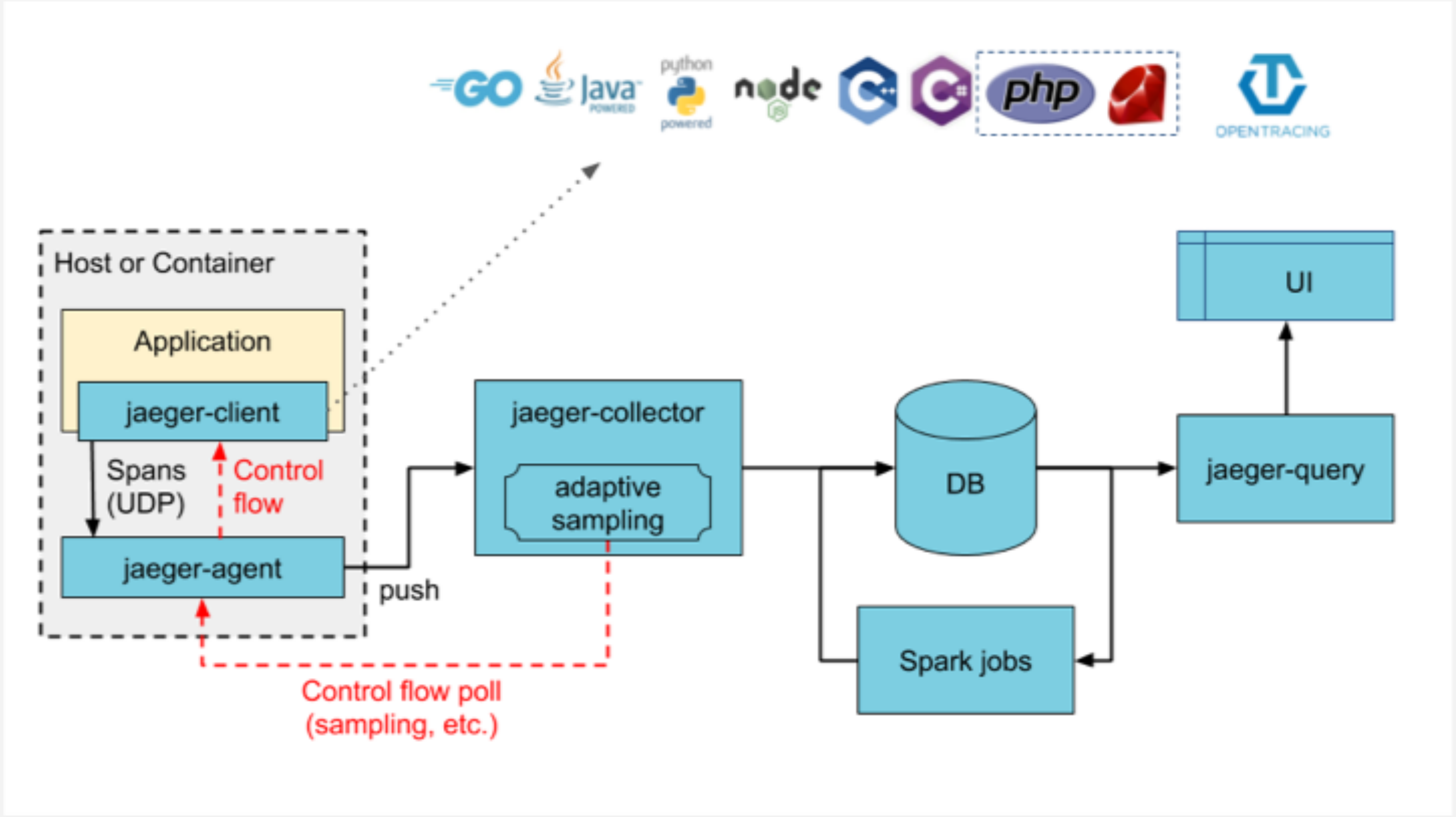
```
serviceContext := tracing.GetServiceContext(ctx)
if serviceContext == nil {
```

DCL

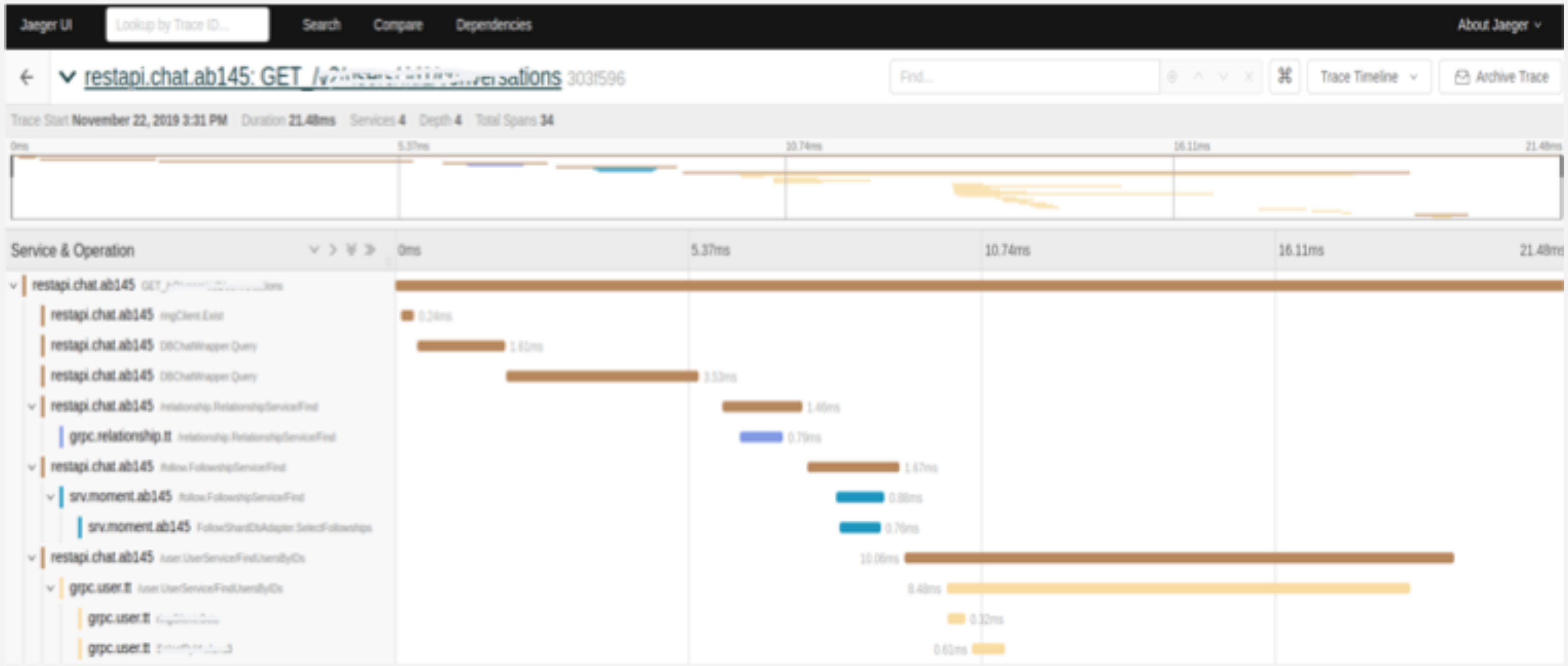


```
serviceContext := tracing.GetServiceContext(ctx)
[redacted]
}
dcl.Context = serviceContext
```

链路分析


GO
Java
python powered
node
php
OPENTRACING
Host or Container
Application
jaeger-client
Spans (UDP)
Control flow
jaeger-agent
push
jaeger-collector
adaptive sampling
Control flow poll (sampling, etc.)
DB
Spark jobs
jaeger-query
UI

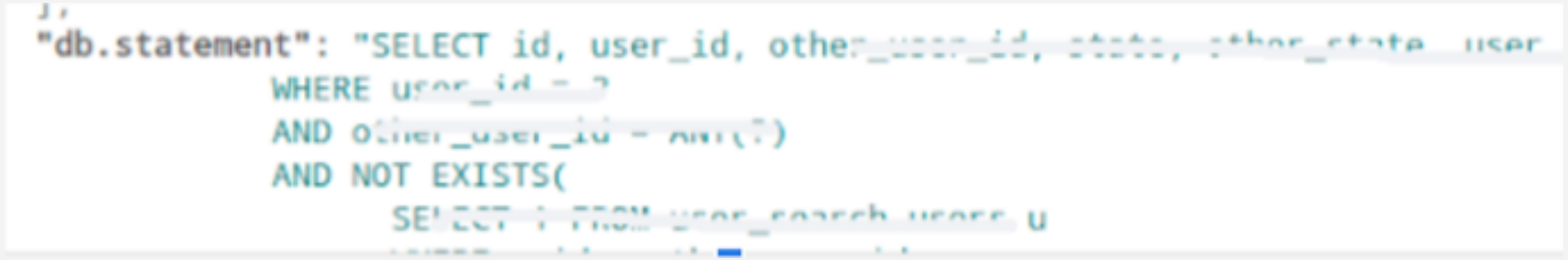
服务链路



restapi.chat.ab145: GET_/v2/[illegible]/conversations 303f596

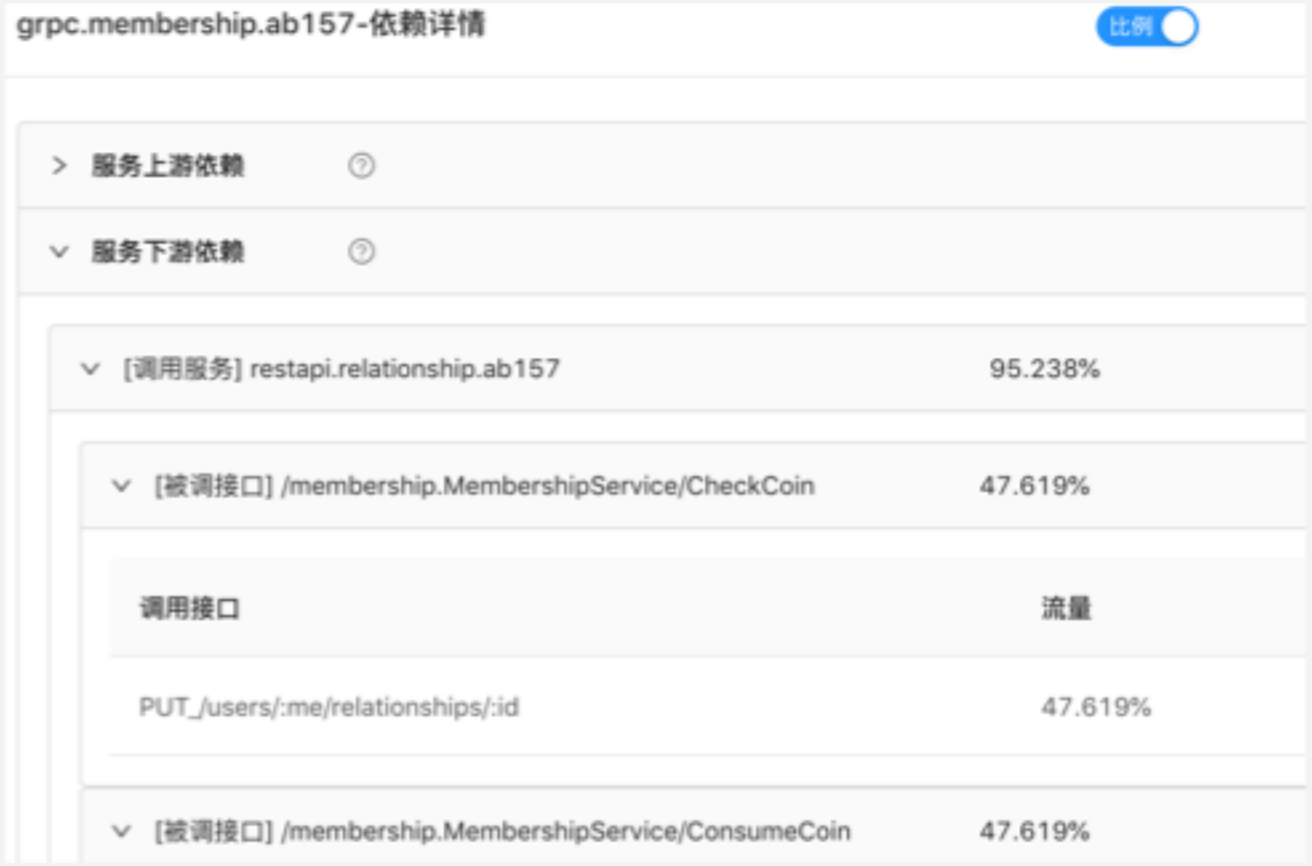
Trace Start November 22, 2019 3:31 PM Duration 21.48ms Services 4 Depth 4 Total Spans 34

```
Service & Operation                                         0ms    5.37ms    10.74ms    16.11ms    21.48ms
restapi.chat.ab145  GET_/v2/[illegible]ions
  restapi.chat.ab145  ringClient.Exist                      0.24ms
  restapi.chat.ab145  DBChatWrapper.Query                   1.61ms
  restapi.chat.ab145  DBChatWrapper.Query                   3.53ms
  restapi.chat.ab145  /relationship.RelationshipService/Find   1.46ms
    grpc.relationship.tt  /relationship.RelationshipService/Find   0.79ms
  restapi.chat.ab145  /follow.FollowshipService/Find        1.67ms
    srv.moment.ab145  /follow.FollowshipService/Find        0.88ms
      srv.moment.ab145  FollowShardDbAdapter.SelectFollowships   0.76ms
  restapi.chat.ab145  /user.UserService/FindUsersByIDs      10.06ms
    grpc.user.tt  /user.UserService/FindUsersByIDs          8.48ms
      grpc.user.tt  [illegible]                             0.32ms
      grpc.user.tt  [illegible]                             0.61ms
```



```
],
"db.statement": "SELECT id, user_id, other_[illegible]
            WHERE u[illegible]
            AND o[illegible] ANY(?)
            AND NOT EXISTS(
                SE[illegible] u
```

服务依赖



grpc.membership.ab157-依赖详情

比例

> 服务上游依赖

v 服务下游依赖

v [调用服务] restapi.relationship.ab157 95.238%

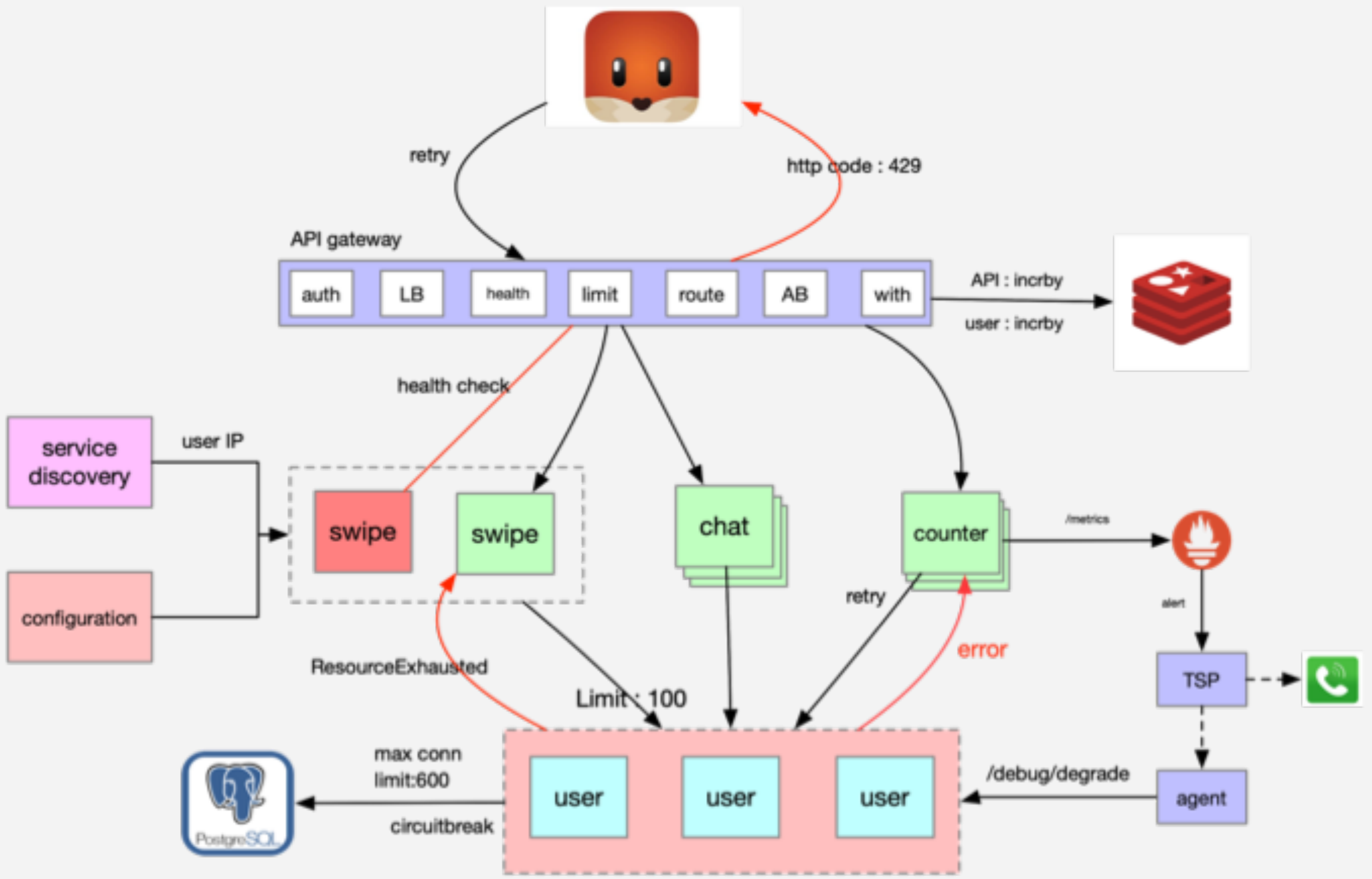
v [被调接口] /membership.MembershipService/CheckCoin 47.619%

调用接口	流量
PUT_/users/:me/relationships/:id	47.619%

v [被调接口] /membership.MembershipService/ConsumeCoin 47.619%

high availability

高可用


retry
http code : 429
API gateway
auth
LB
health
limit
route
AB
with
API : incrby
user : incrby
health check
service discovery
user IP
configuration
swipe
swipe
chat
counter
/metrics
alert
retry
error
TSP
ResourceExhausted
Limit : 100
max conn
limit:600
circuitbreak
user
user
user
/debug/degrade
agent
PostgreSQL

高可用

重试

条件
- 错误码
- deadline
- 快速失败

等待时间
- 指数抖动
- 线性

重试次数
- 固定
- 动态

熔断

条件
- 异常比例
- 异常数
- 响应时间

粒度
- 服务
- 方法

回退
- 快速失败
- 自定义
- 故障沉默

限流

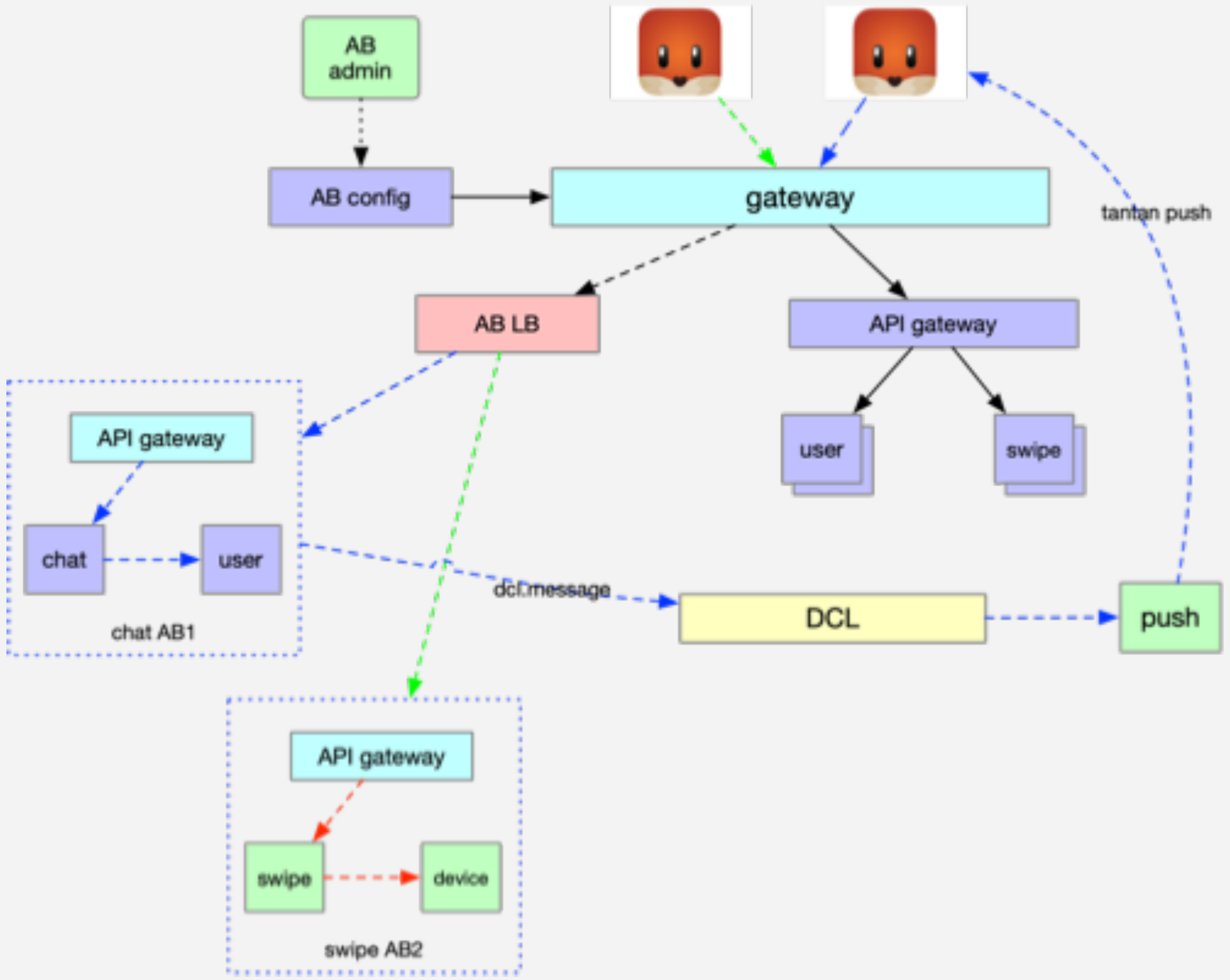
算法
- 令牌桶
- 分布式
- 漏桶

配额
- 固定
- 动态

时间窗口
- 固定
- 滑动

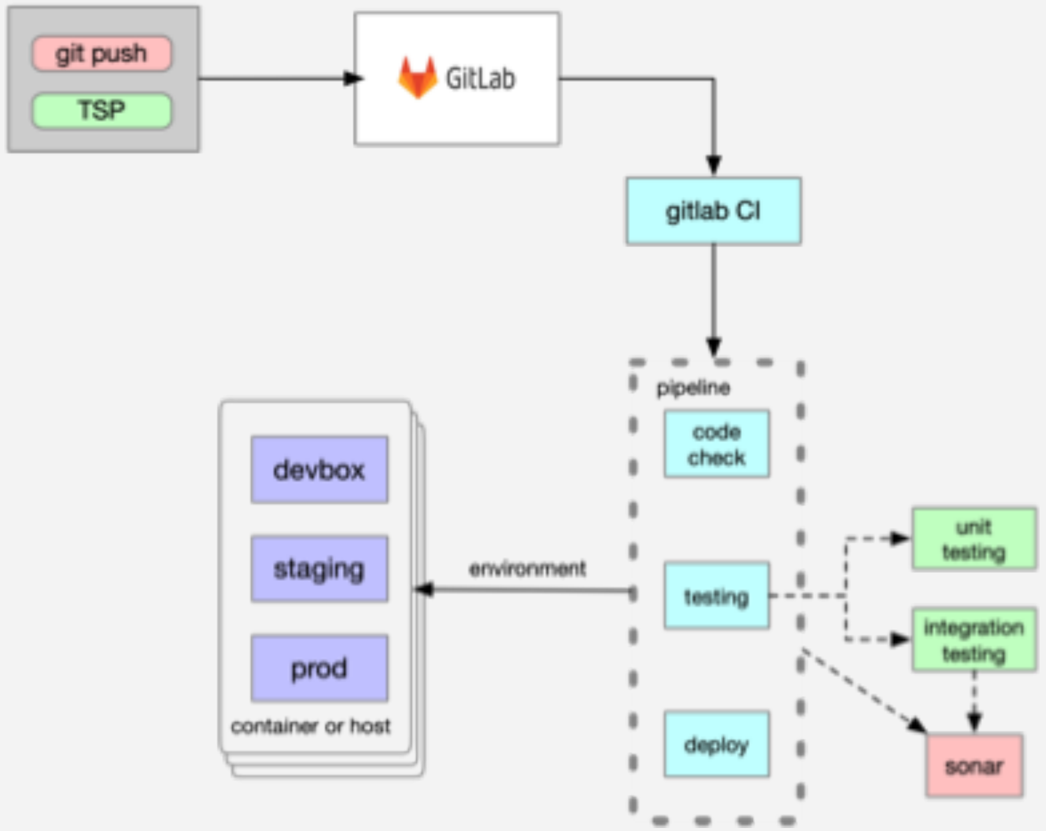
AB testing

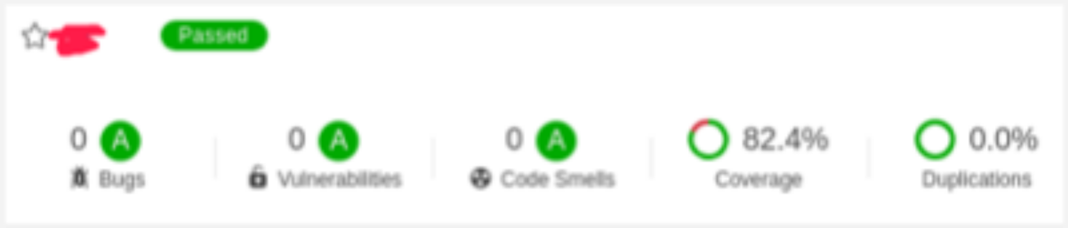
AB testing


AB admin
AB config
gateway
tantan push
AB LB
API gateway
user
swipe
API gateway
chat
user
chat AB1
dcl message
DCL
push
API gateway
swipe
device
swipe AB2

CI & CD

CI&CD




☆ [redacted] Passed

0 A Bugs | 0 A Vulnerabilities | 0 A Code Smells | 82.4% Coverage | 0.0% Duplications

generated code

重复性工作

初始化

- 配置
- 日志
- 交互接口
- 资源

server/client

- rpc
- http
- middleware

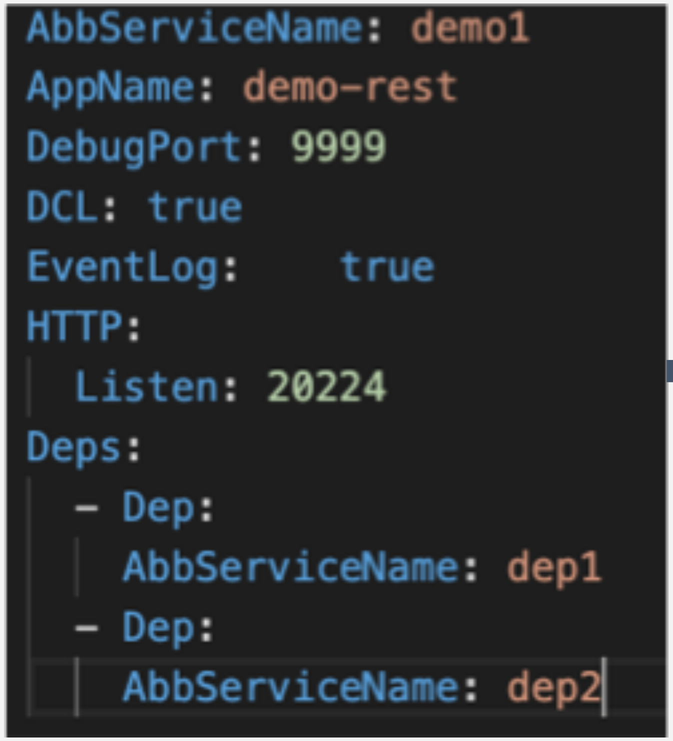
接口

- rpc
- http
- DCL
- db/cache
- mq
- ...

CI&CD

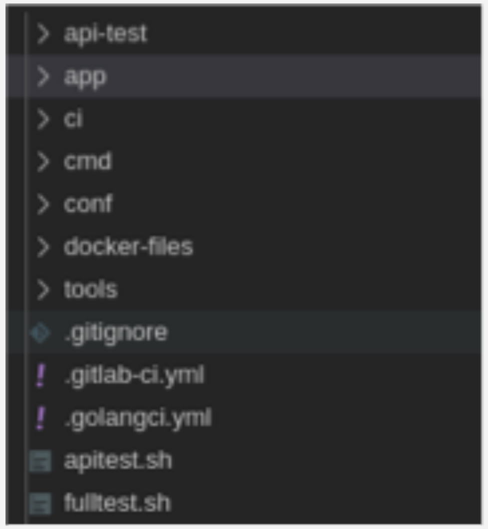
- 测试
- 编译
- 部署

代码生成



```yaml
AbbServiceName: demo1
AppName: demo-rest
DebugPort: 9999
DCL: true
EventLog:    true
HTTP:
  Listen: 20224
Deps:
  - Dep:
    AbbServiceName: dep1
  - Dep:
    AbbServiceName: dep2
```

go template

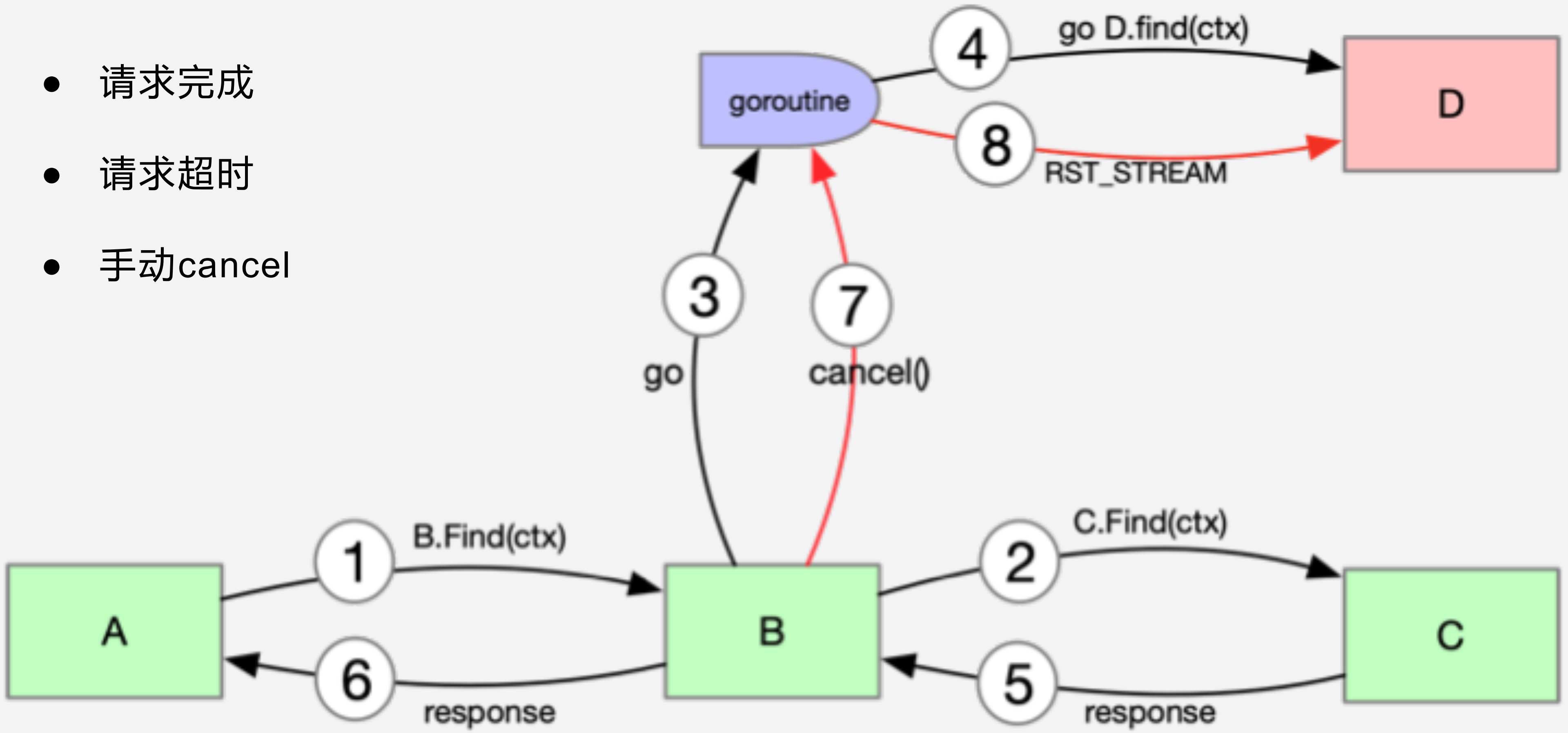


```
> api-test
> app
> ci
> cmd
> conf
> docker-files
> tools
.gitignore
.gitlab-ci.yml
.golangci.yml
apitest.sh
fulltest.sh
```

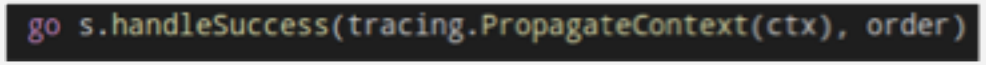
go and microservices

context

- 请求完成
- 请求超时
- 手动cancel

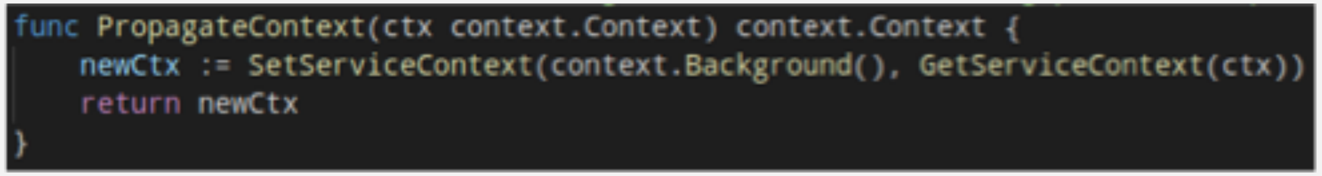


context



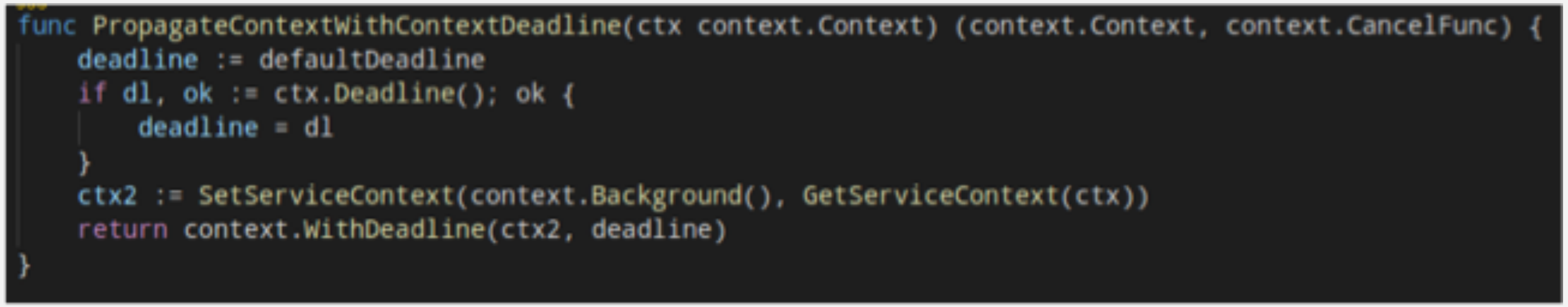
```go
go s.handleSuccess(tracing.PropagateContext(ctx), order)
```

移除cancel和deadline



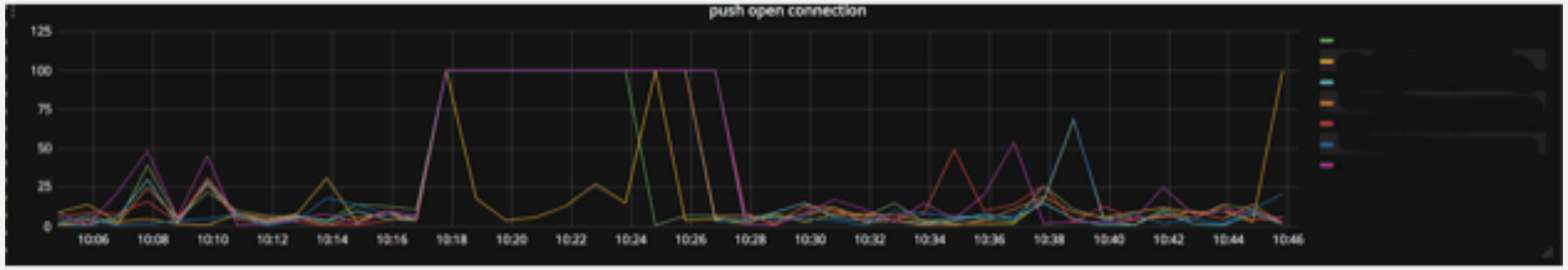
```go
func PropagateContext(ctx context.Context) context.Context {
    newCtx := SetServiceContext(context.Background(), GetServiceContext(ctx))
    return newCtx
}
```

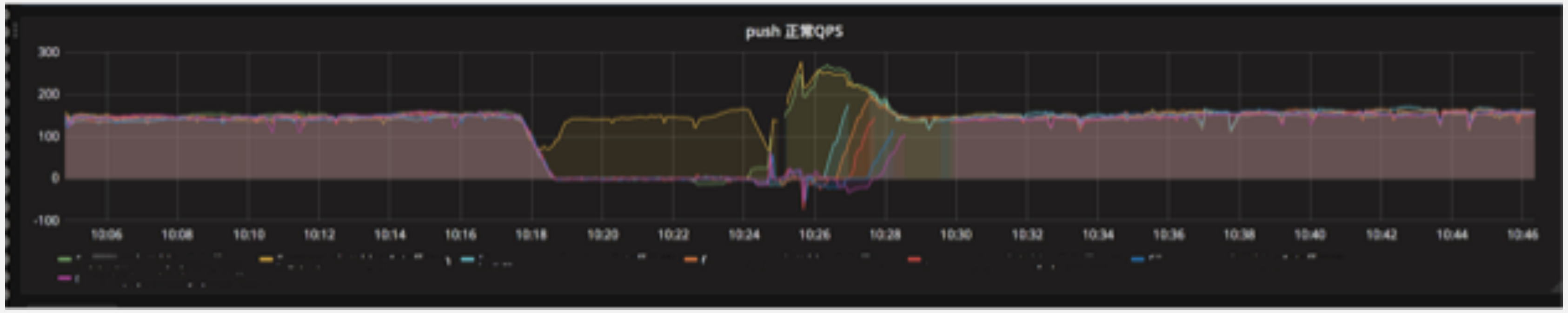
传递deadline



```go
func PropagateContextWithContextDeadline(ctx context.Context) (context.Context, context.CancelFunc) {
    deadline := defaultDeadline
    if dl, ok := ctx.Deadline(); ok {
        deadline = dl
    }
    ctx2 := SetServiceContext(context.Background(), GetServiceContext(ctx))
    return context.WithDeadline(ctx2, deadline)
}
```

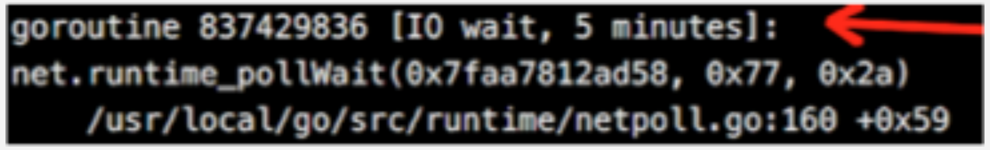
pprof


push open connection
125
100
75
50
25
0
10:06
10:08
10:10
10:12
10:14
10:16
10:18
10:20
10:22
10:24
10:26
10:28
10:30
10:32
10:34
10:36
10:38
10:40
10:42
10:44
10:46

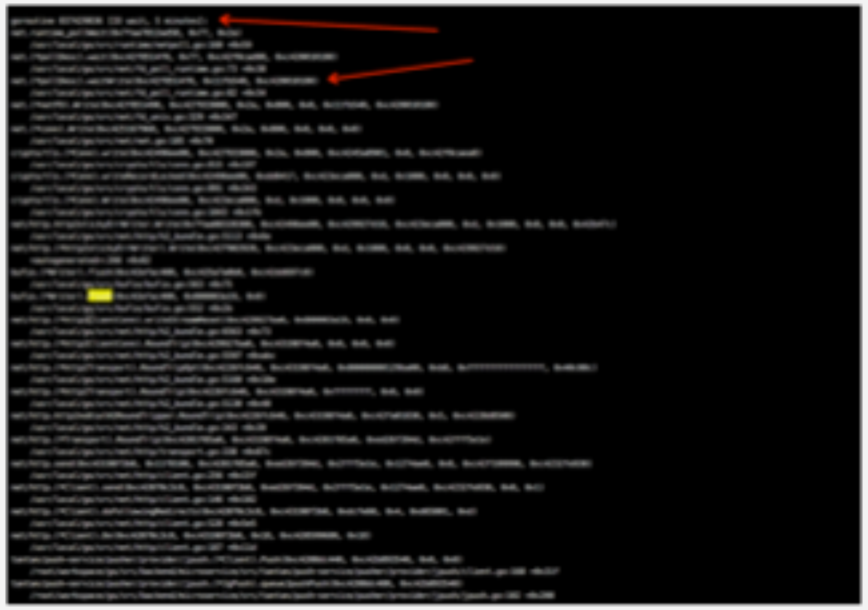

push 正常QPS
300
200
100
0
-100
10:06
10:08
10:10
10:12
10:14
10:16
10:18
10:20
10:22
10:24
10:26
10:28
10:30
10:32
10:34
10:36
10:38
10:40
10:42
10:44
10:46

pprof

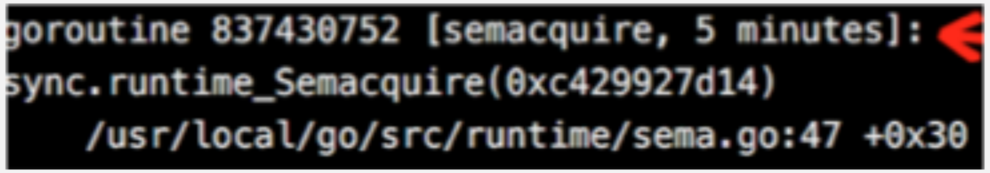
获得锁goroutine



```
goroutine 837429836 [IO wait, 5 minutes]:
net.runtime_pollWait(0x7faa7812ad58, 0x77, 0x2a)
    /usr/local/go/src/runtime/netpoll.go:160 +0x59
```

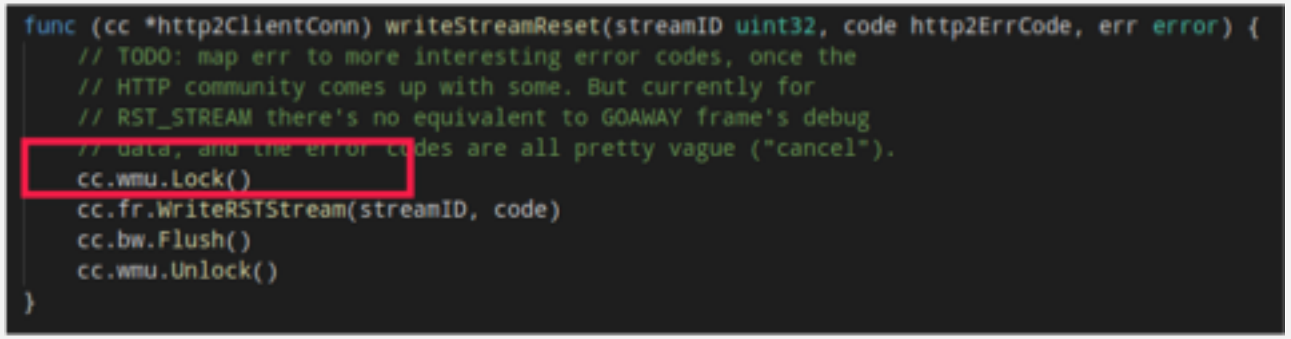


等待锁goroutine



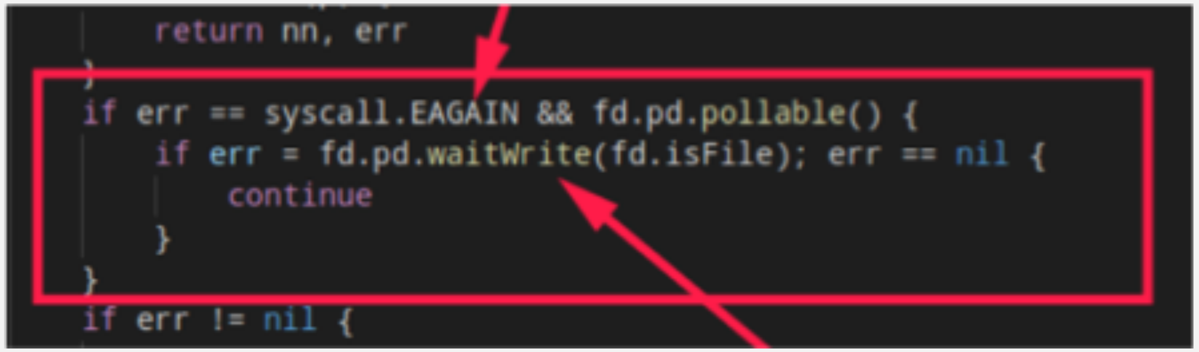
```
goroutine 837430752 [semacquire, 5 minutes]:
sync.runtime_Semacquire(0xc429927d14)
    /usr/local/go/src/runtime/sema.go:47 +0x30
```

writeStreamReset 加锁



```go
func (cc *http2ClientConn) writeStreamReset(streamID uint32, code http2ErrCode, err error) {
    // TODO: map err to more interesting error codes, once the
    // HTTP community comes up with some. But currently for
    // RST_STREAM there's no equivalent to GOAWAY frame's debug
    // data, and the error codes are all pretty vague ("cancel").
    cc.wmu.Lock()
    cc.fr.WriteRSTStream(streamID, code)
    cc.bw.Flush()
    cc.wmu.Unlock()
}
```

write返回EAGAIN，等待epoll通知



```go
        return nn, err
    }
    if err == syscall.EAGAIN && fd.pd.pollable() {
        if err = fd.pd.waitWrite(fd.isFile); err == nil {
            continue
        }
    }
    if err != nil {
```

pprof

活锁过程如下：

1. 整个链路网络不稳定，拥塞或者链路中的交换机路由服务器断网、断电等等导致此连接网络不可达可能因素，对于服务器是无法感知的

1. 网络出现问题导致http 2.0 client 开始cancel掉超时的请求，给connection加锁，发送Reset帧

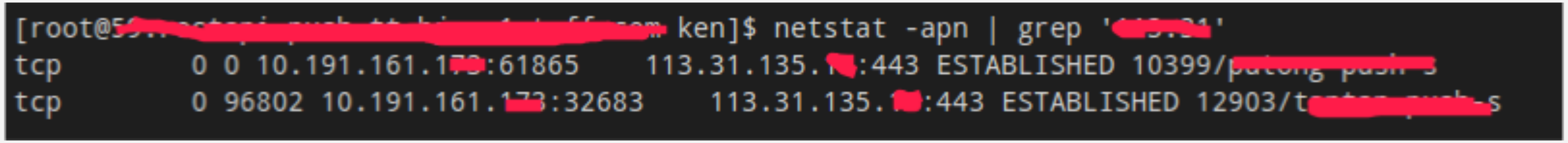
1. 系统调用write返回EAGAIN，client开始等待 epoll “可写”通知

1. go的http客户端的transport维护了一个连接池，发送请求时候会遍历连接池中连接是否可用，判断是否可用要加锁，而刚好步骤1中的连接一直在等待epoll通知，无法释放锁，http client一直拿不到锁

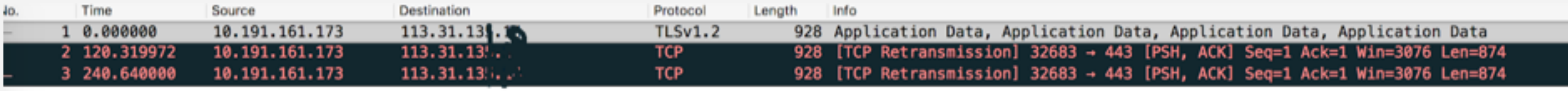
1. http超时时间并没有应用到io层，导致步骤3中的连接 开始不断的重传（可能重试15次，约15分钟），直到连接断开；释放锁

1. 其他http client获取到锁

pprof



```
[root@5[redacted] ken]$ netstat -apn | grep '[redacted]'
tcp        0 0 10.191.161.1[redacted]:61865     113.31.135.[redacted]:443 ESTABLISHED 10399/[redacted]
tcp        0 96802 10.191.161.[redacted]:32683     113.31.135.[redacted]:443 ESTABLISHED 12903/t[redacted]s
```

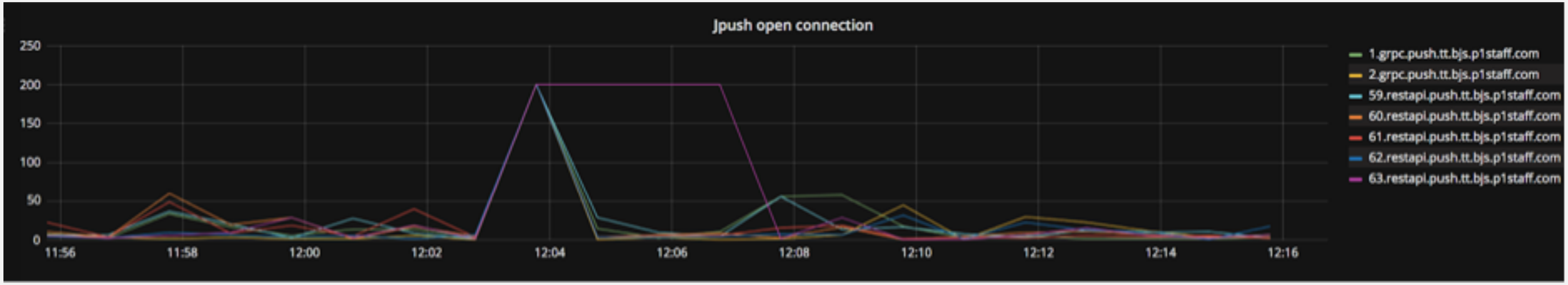


```
lo.     Time         Source          Destination            Protocol   Length  Info
   1  0.000000     10.191.161.173  113.31.13[redacted]     TLSv1.2       928  Application Data, Application Data, Application Data, Application Data
   2  120.319972   10.191.161.173  113.31.13[redacted]     TCP           928  [TCP Retransmission] 32683 → 443 [PSH, ACK] Seq=1 Ack=1 Win=3076 Len=874
   3  240.640000   10.191.161.173  113.31.13[redacted]     TCP           928  [TCP Retransmission] 32683 → 443 [PSH, ACK] Seq=1 Ack=1 Win=3076 Len=874
```

pprof

将内核RTO参数修改为：
net.ipv4.tcp_retries1 = 3 // 更新路由缓存
net.ipv4.tcp_retries2 = 5 // 断连接

一般情况下只重传5次就reset连接（约7秒），也就是说如果发生阻塞的情况下，持续7秒后就重新建立连接了。



https://github.com/golang/go/issues/23559

Thanks