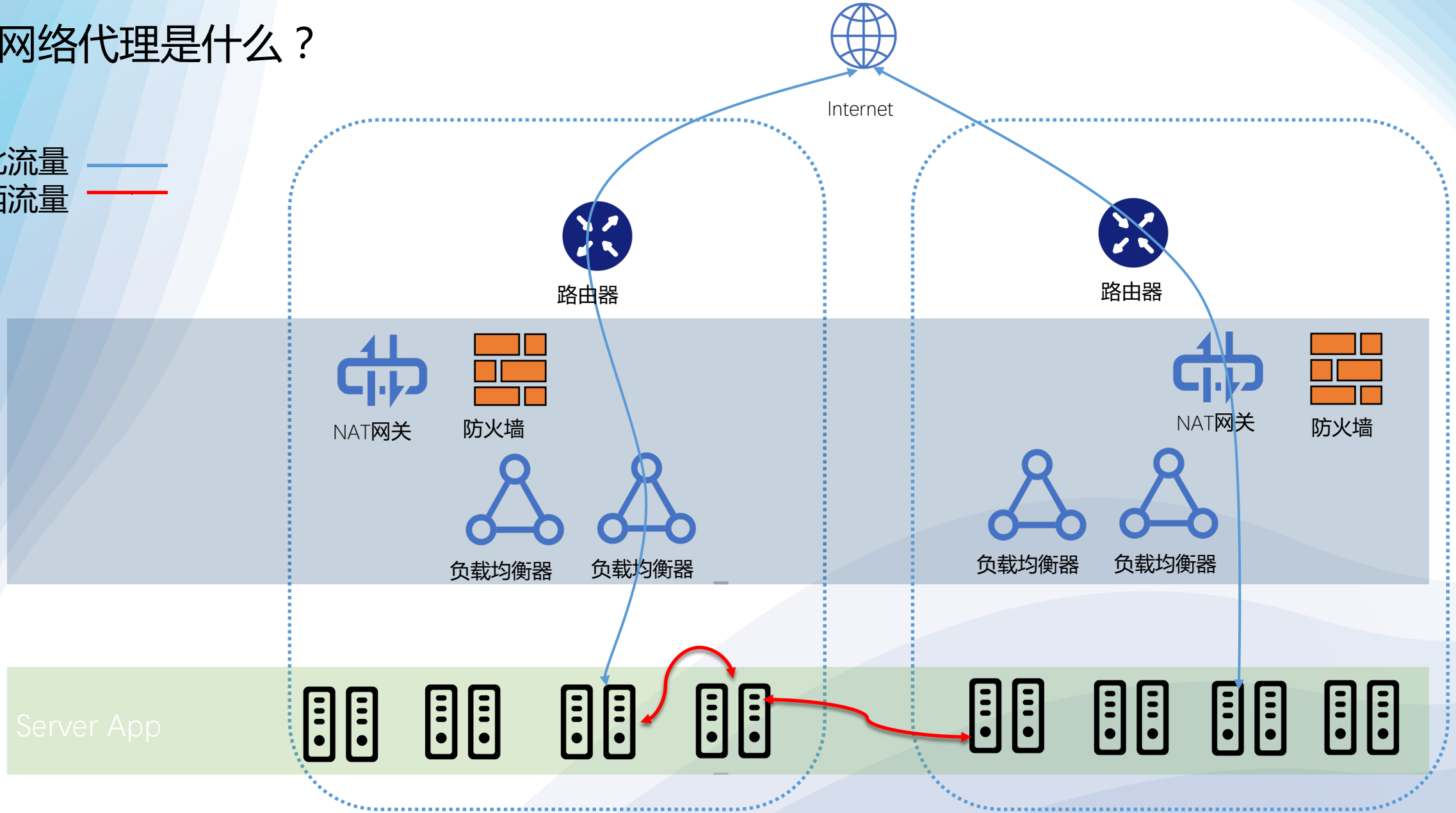
蚂蚁金服网络代理演进之路

田阳（烈元）
蚂蚁金服技术专家

2019.10.27

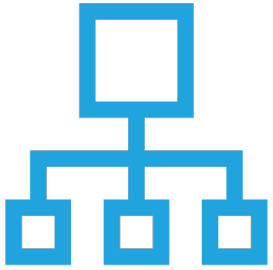
网络代理是什么？

南北流量
东西流量

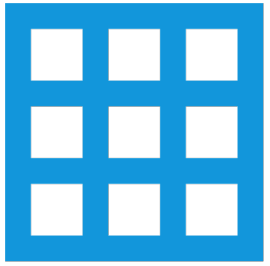


网络代理有什么？

Maglev
Ipvs
Katran



GFE
BFE
TGW
Nginx
Apache
httpd



SOFAMosn
Envoy
Linkerd

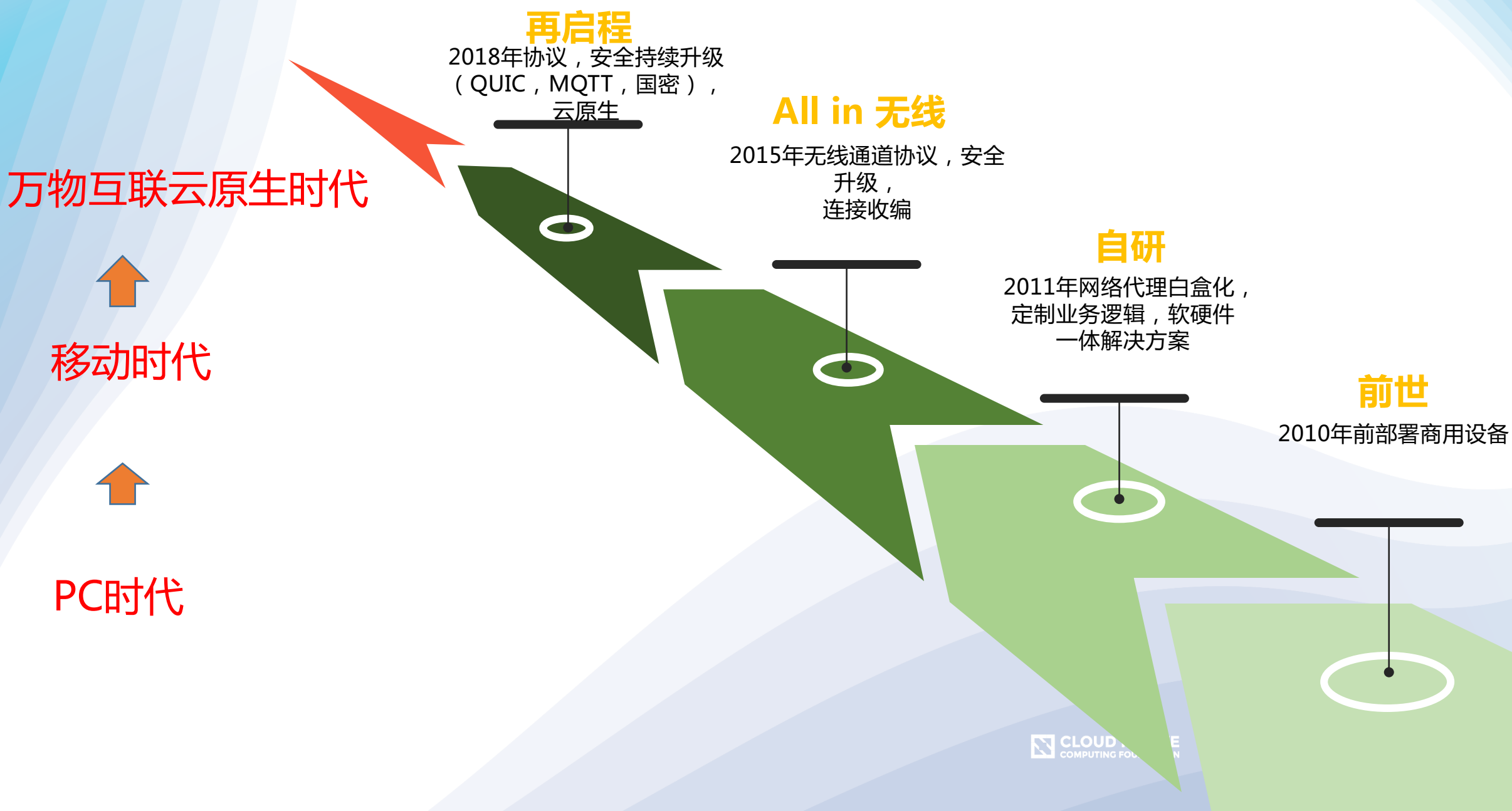
网络的挑战


11.11
天猫全球狂欢节
2017
全球支付总笔数
14.8亿
同比增长41%
支付峰值
为去年的2.1倍
25.6万笔/秒
中国自主研发数据库OceanBase处理峰值
4200万次/秒
蚂蚁金服
ANT FINANCIAL
支付宝
ALIPAY


中国年福气年
2016 LUCKY NEW YEAR
咻
今年过年，又多了一种声音
春晚期间支付宝咻一咻
21点09分峰值达到210亿次/分钟
总参与3245亿次

网络的挑战

蚂蚁金服网络接入十年变迁



前世

Netscaler

F5 BigIP

自研四层网络代理

- 全面使用DPDK技术重构

2011

- 四层负载均衡-IPVS
- NAT网关

2014

2018

- EBPF，XDP
- 可编程交换芯片（P4语言）

未来

蚂蚁七层网络代理

Google Spanner?

蚂蚁七层网络接入代理

Spanner

蚂蚁七层网络接入代理

亿级用户同时在线
千万级每秒RPC请求
百万级每秒推送

AGNA
(Ant Global
Network Accelarator)

MQTT
MMTP
Mtls
HTTP1
TLS1.2
HTTP2
TLS1.3
QUIC
国密

LDC1

LVS(四层负载)
DNS
Keycenter
硬件加速
安全合规
Spanner
Spanner
Spanner
APP
APP
APP
APP

网络控制面

LDC2

LVS(四层负载)
DNS
Keycenter
硬件加速
安全合规
Spanner
Spanner
Spanner
APP
APP
APP
APP

CLOUD NATIVE COMPUTING FOUNDATION
蚂蚁金服 ANT FINANCIAL
ServiceMesher

Spanner

2010
- 自研，网络设备白盒化
- 全面实践全网https

2012
- 首次全流量支撑双十一大促

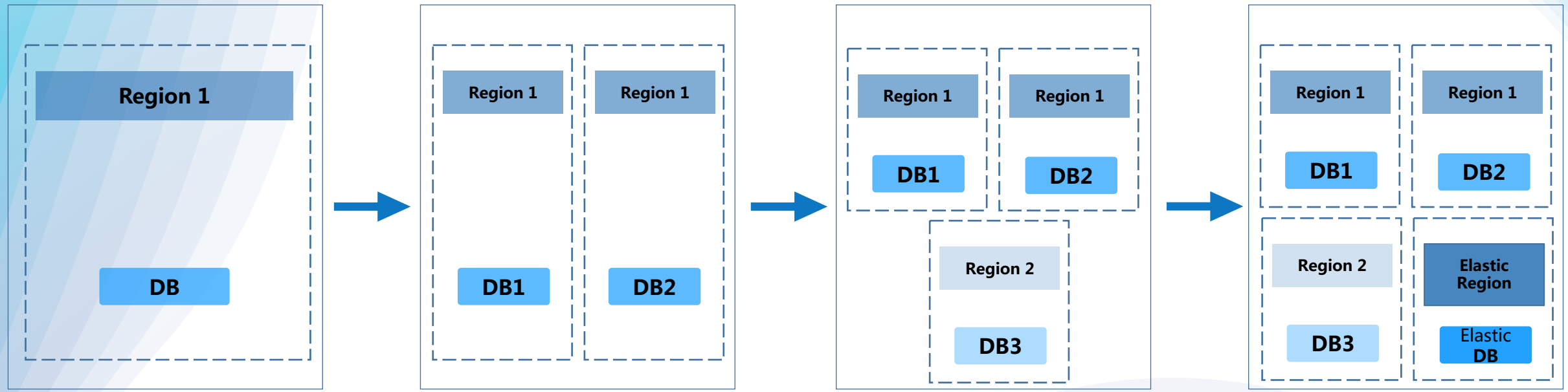
2013
- 支持蚂蚁LDC架构，三地五中心容灾架构
- 全面上线SSL加速卡，提供软硬件一体加速方案

2015
- All in 无线，通信通道全面升级（MMTP，MTLS协议）

2016
- 安全防护能力提升，WAF，流量镜像

2018至今
- 通信协议，架构，安全再次升级（物联终端接入，QUIC协议，国密，可信计算，海外加速，云原生）

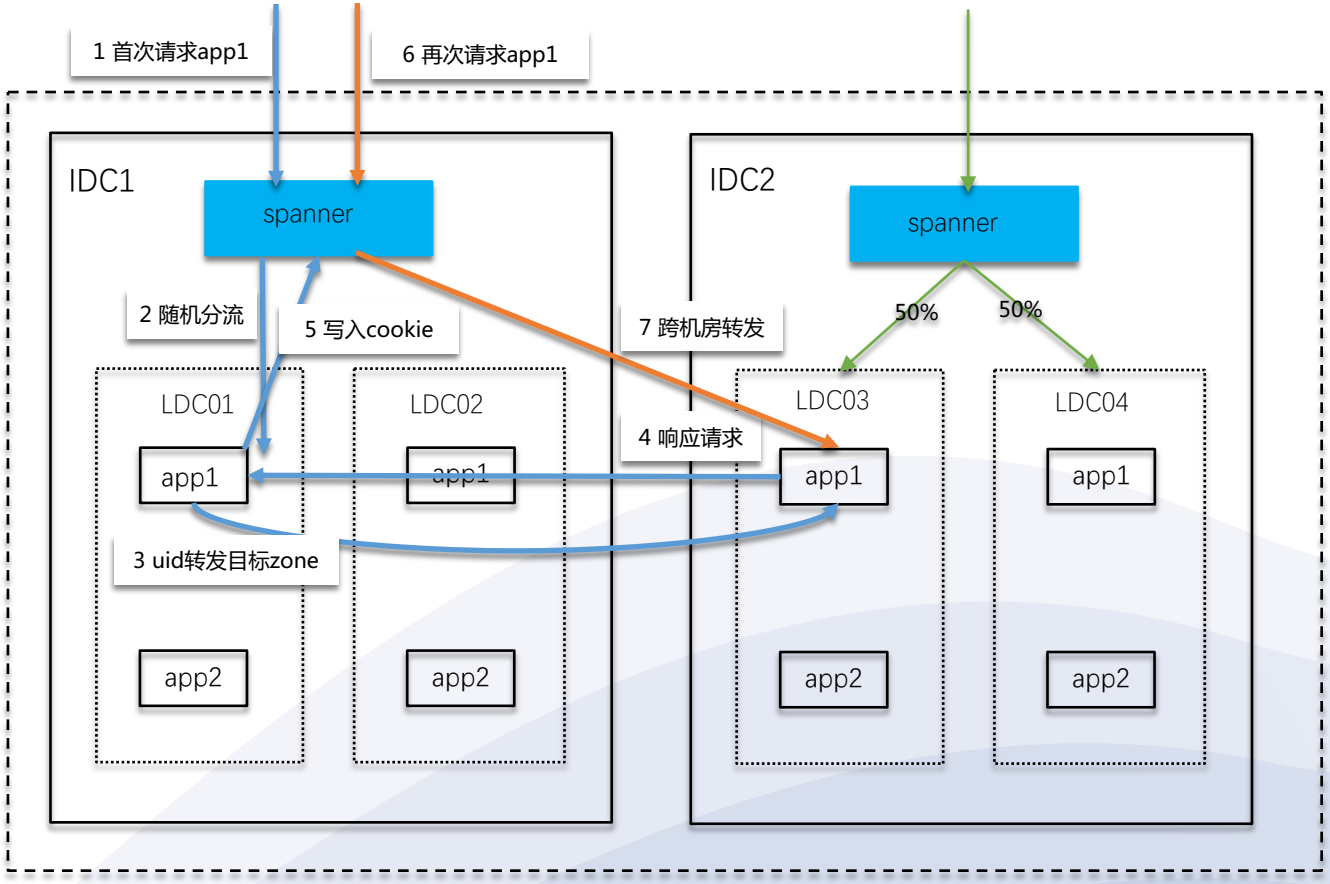
金融级三地五中心容灾架构（LDC）



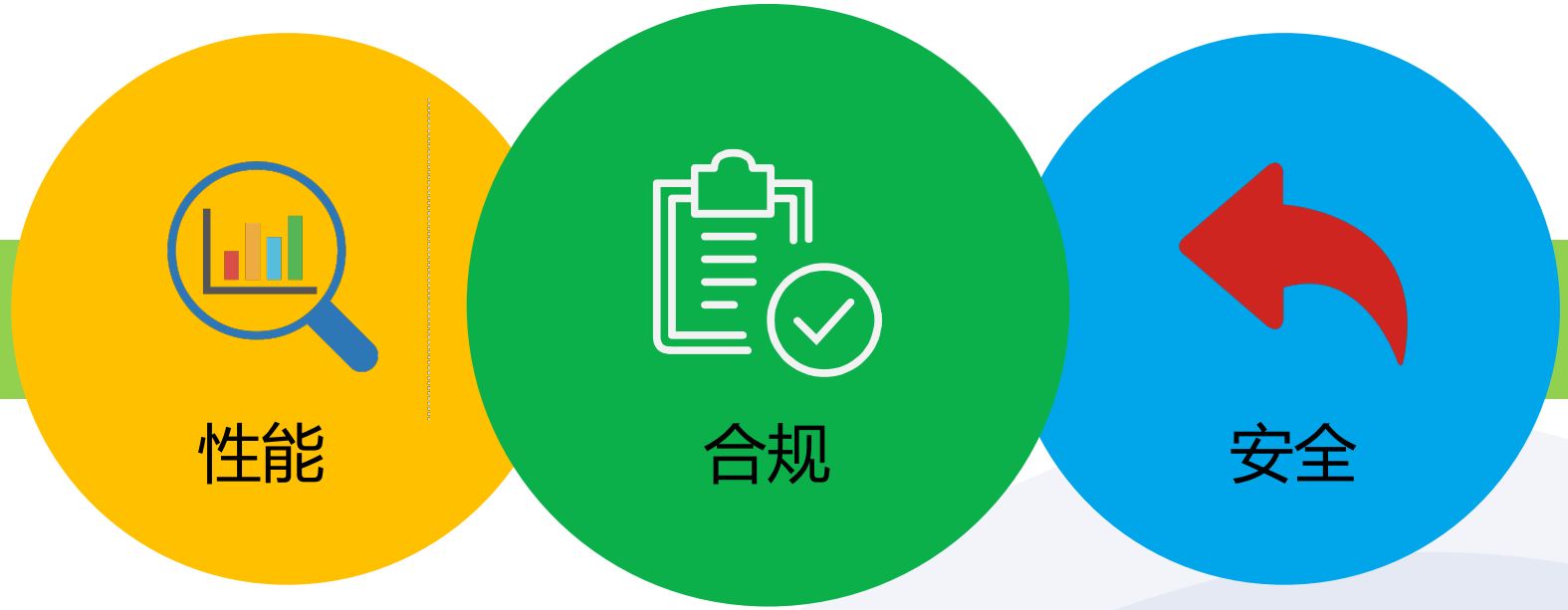
单机房 LDC → 同城双活 LDC → 异地多活 LDC → 弹性伸缩混部 LDC

金融级三地五中心架构的流量调度

- 机房内zone随机路由
- Cookie zone转发
- 蓝绿发布
- 容灾
  - ➢ Zone内容灾
  - ➢ 机房级别
  - ➢ 城市级别
- 弹性调度
- 压测
- 灰度

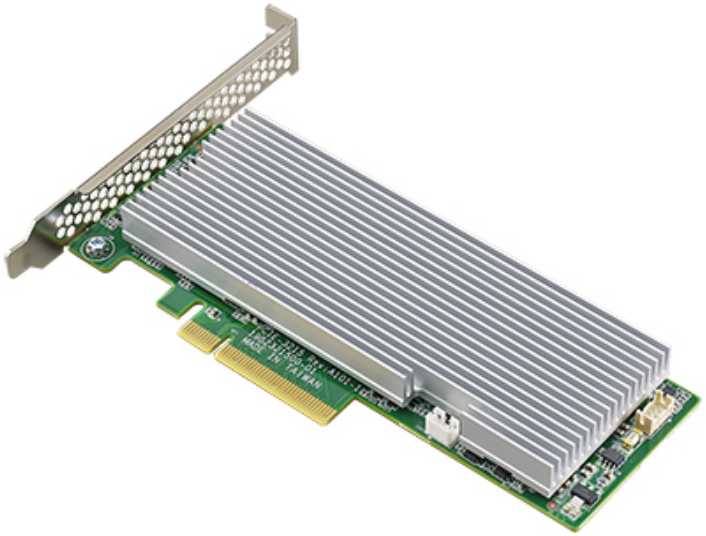


蚂蚁金服SSL/TLS实践



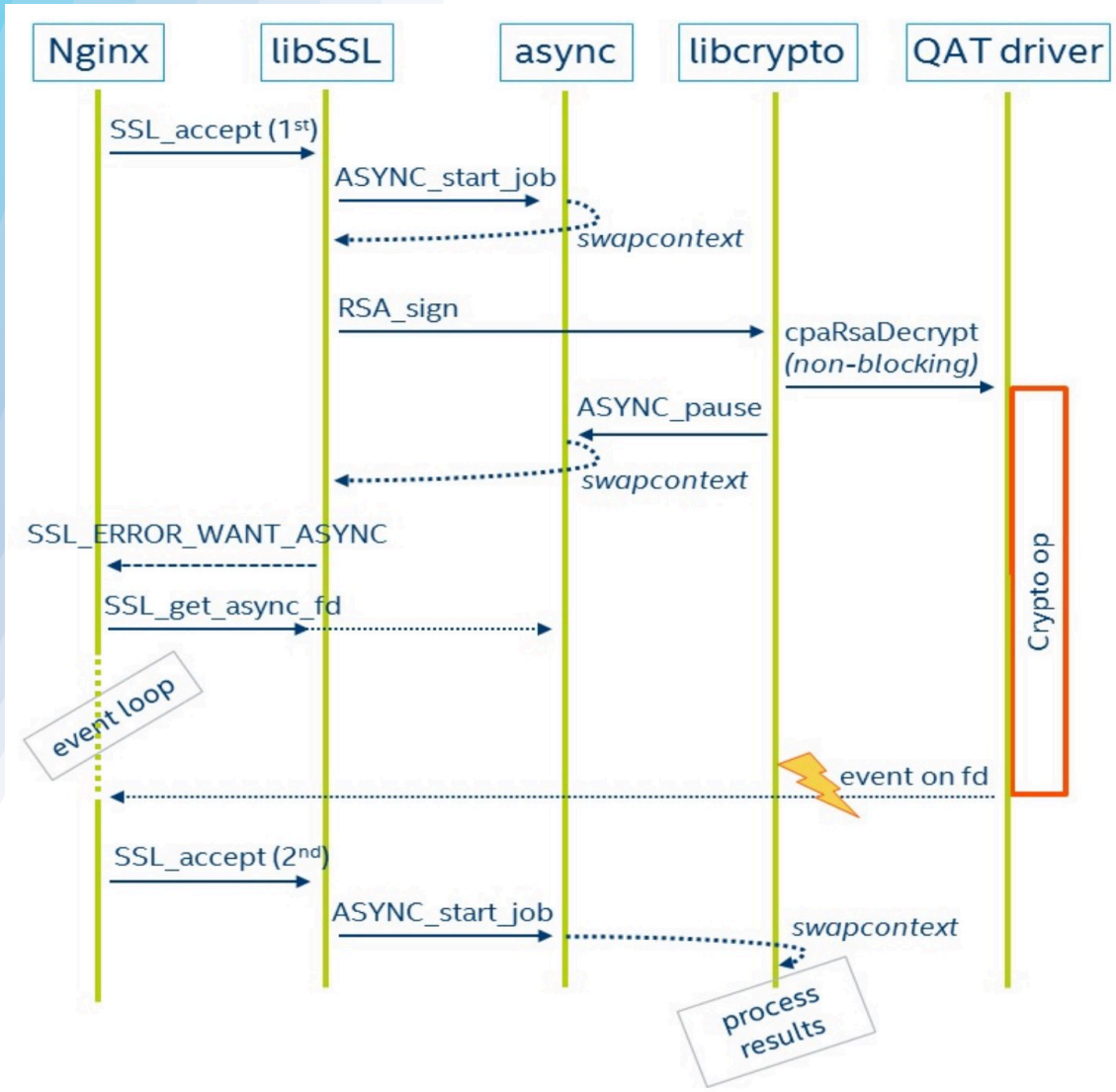
软硬件一体解决方案

Cavium Nitrox



Intel QAT

软硬件一体解决方案



- 对Spanner实现了异步化改造
- 对openssl进行了异步化引擎改造
- 实现多芯片卡的负载均衡

SSL握手性能提升3倍，
RSA2048 3万/每秒签名能力

协议实现的改造-MTLS

MTLS：1) 轻量级TLS库，小于50k；2) 优化的TLS协议

高效　优化　灵活

0-RTT

- 减少握手延迟
- 代价：握手前发送的数据不能保证防重放攻击，因此要求应用程序自己保证防重放攻击

按需握手

- 业务可根据需求灵活选择明文或密文传输，提升业务效率

TLS扩展

- Session Ticket扩展
  用于会话复用，加速握手过程
- Cached-info扩展
  缓存证书等服务端信息，避免再次握手时重复传输数据
- ECDHE-keyshare扩展
  将TLS1.3草案中的1-RTT机制通过扩展的方式提前应用
- ECC-signature扩展
  使用高效ECDSA签名算法的同时，兼容广泛使用的RSA证书

Small Ticket

- 自定义Session Ticket编码格式
- 160 byte -> 76 byte

动态Record Size

- 平衡吞吐与时延

安全合规能力持续升级

国密算法

- 拥抱监管
- 安全可控
- 金融科技

AntTLS库

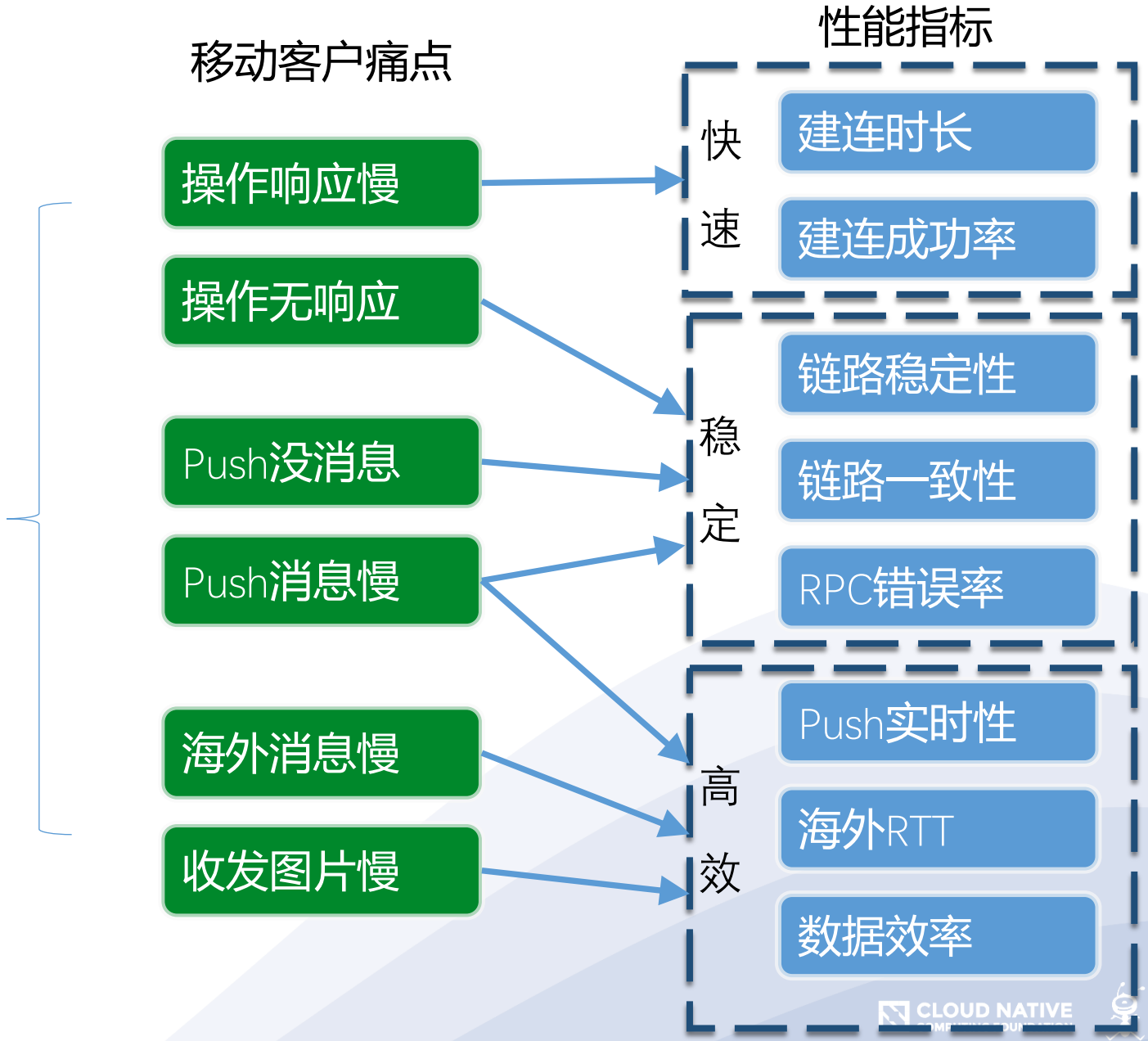
- 基于OpenSSL
- 全面拥抱TLS1.3
- 国密优化实现，国密单证书标准支撑
- 支持SGX等可信机制
- 多硬件卡Engine
- Mobile，iot设备等多终端支持
- OpenSSL Committer

无线移动战役

线下支付

大促

国际支付



咻一咻与敬业福

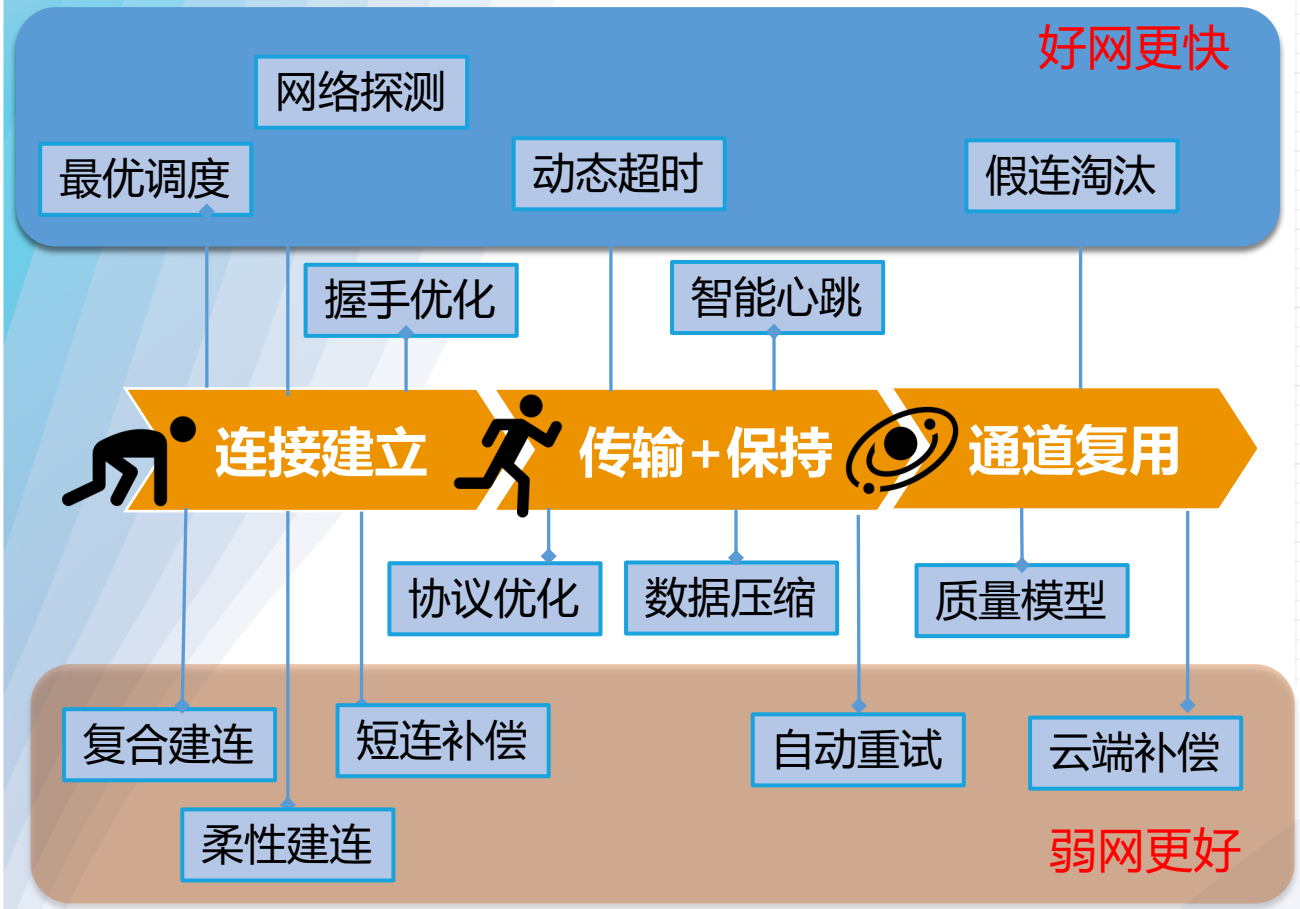
咻一咻的挑战

剩余红包实时显示

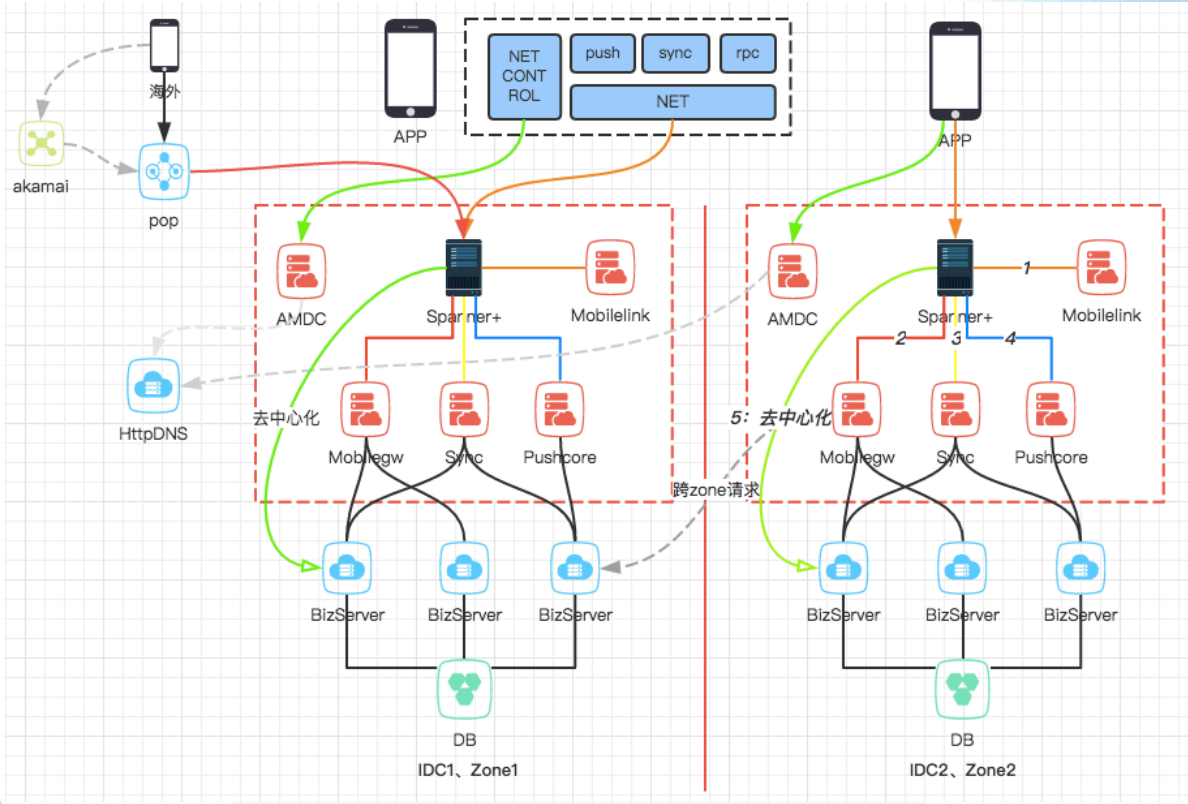
亿级用户快速进入

亿级用户同时点击

无线移动网络优化



- 终端策略覆盖移动网络难点
- 优化对业务透明
- ROI考虑



支付宝网络接入层架构示意

- **统一通道**：主长连接 + 短连接
- **统一协议**：MTLS+MMTP
- **统一调度**：MobileDC
- **关键词**：动态Hpack + PB + 动态字典 + Zstd

通信协议&架构持续升级

多终端&协议接入

- MQTT协议的IOT设备接入
- QUIC/HTTP3

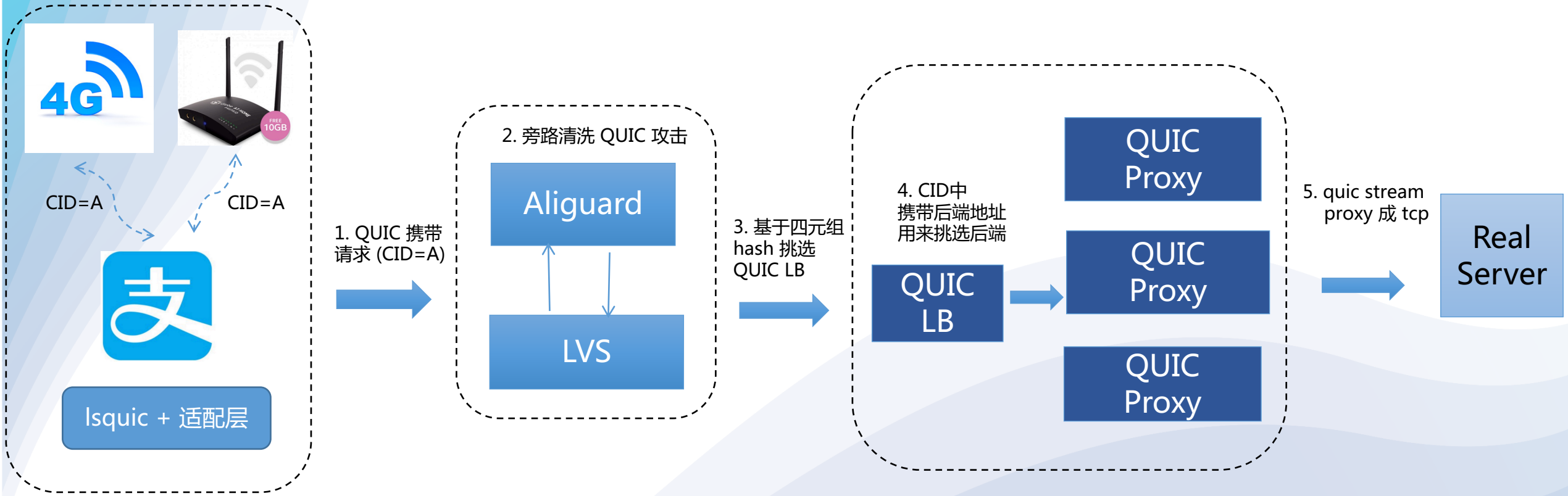
架构升级

- 就近就优海外接入，智能调度
- 支持QUIC协议的LB建设
- 蚂蚁全球加速节点，全协议支持

云原生生态融合

- 支持UDPA
- 接入层容器化，混部
- Web Assembly模块扩展

QUIC方案介绍



QUIC较优实践

UDP实践

- NGINX_QUIC_MODULE
  - 多进程模型
  - 全异步框架
  - mlisten 机制避免 so_reuseport 带来的内核查找
  - disable GRO
  - skip IPID calculation
  - recv buf 调整
  - 无损发布升级
  - 进程 Crash 快速 reset
  - 服务端不可达，快速 reset
  - QUIC 连接可监控、可观察

QUIC 特性支持

- 支持连接迁移
  - 四元组变化
  - CID 更新
- 支持 0-RTT
  - token 校验升级
  - 防止 0-rtt 重放攻击
- 更加安全，稳定
  - reset token 机制升级
  - chromium quic 库
  - MTU 探测功能

标准参与

- QUIC TLS 标准参与
- QUIC LB 标准参与
- 输出解决方案专利

Nginx epoll性能优化

海量VIP的性能退化

- Accept_Mutex off 惊群，cpu sys 消耗过大
- Accept Mutex on 性能瓶颈

	弊端
Reuseport	Reload连接reset
EPOLLEXCLUSIVE	内核4.5，nginx1.11.3

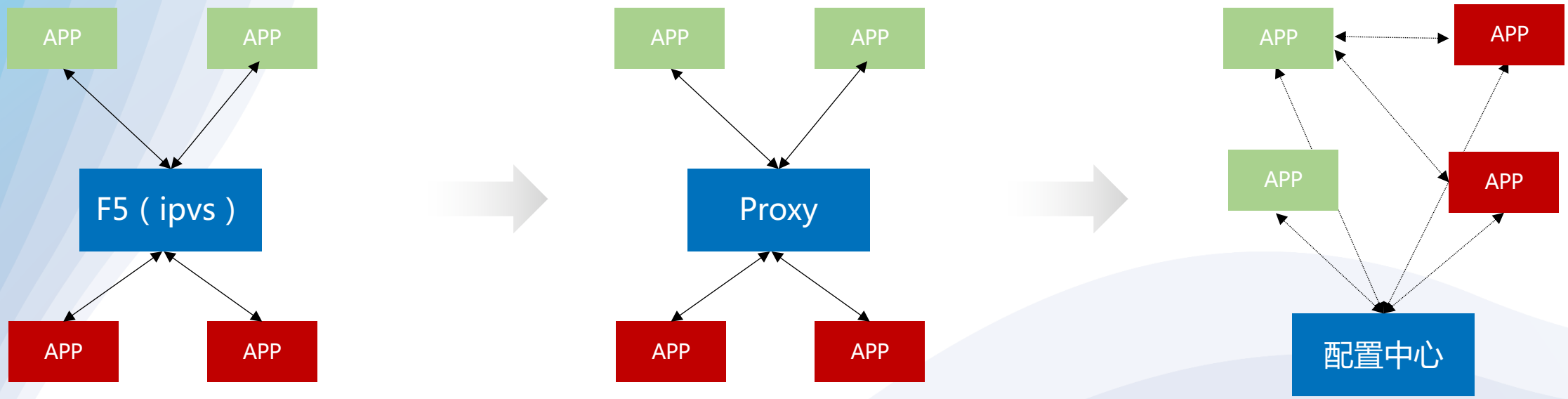
性能优化

```
if (ngx_use_accept_mutex) {
    if (ngx_accept_disabled > 0) {
        ngx_accept_disabled--;

    } else {
        if (ngx_trylock_accept_mutex(cycle) == NGX_ERROR) {
            return;
        }

        if (ngx_accept_mutex_held) {
            flags |= NGX_POST_EVENTS;

        } else {
            if (timer == NGX_TIMER_INFINITE
                || timer > ngx_accept_mutex_delay)
            {
                timer = ngx_accept_mutex_delay;
            }
        }
    }
}
```

```
377  ngx_int_t
378  ngx_enable_accept_events(ngx_cycle_t *cycle)
379  {
380      ngx_uint_t         i;
381      ngx_listening_t   *ls;
382      ngx_connection_t  *c;
383
384      ls = cycle->listening.elts;
385      for (i = 0; i < cycle->listening.nelts; i++) {
386
387          c = ls[i].connection;
388
389          if (c == NULL || c->read->active) {
390              continue;
391          }
392
393          if (ngx_add_event(c->read, NGX_READ_EVENT, 0) == NGX_ERROR) {
394              return NGX_ERROR;
395          }
396      }
397
398      return NGX_OK;
399  }
```

解决方案：嵌套epoll listen fd

东西流量的服务发现与路由

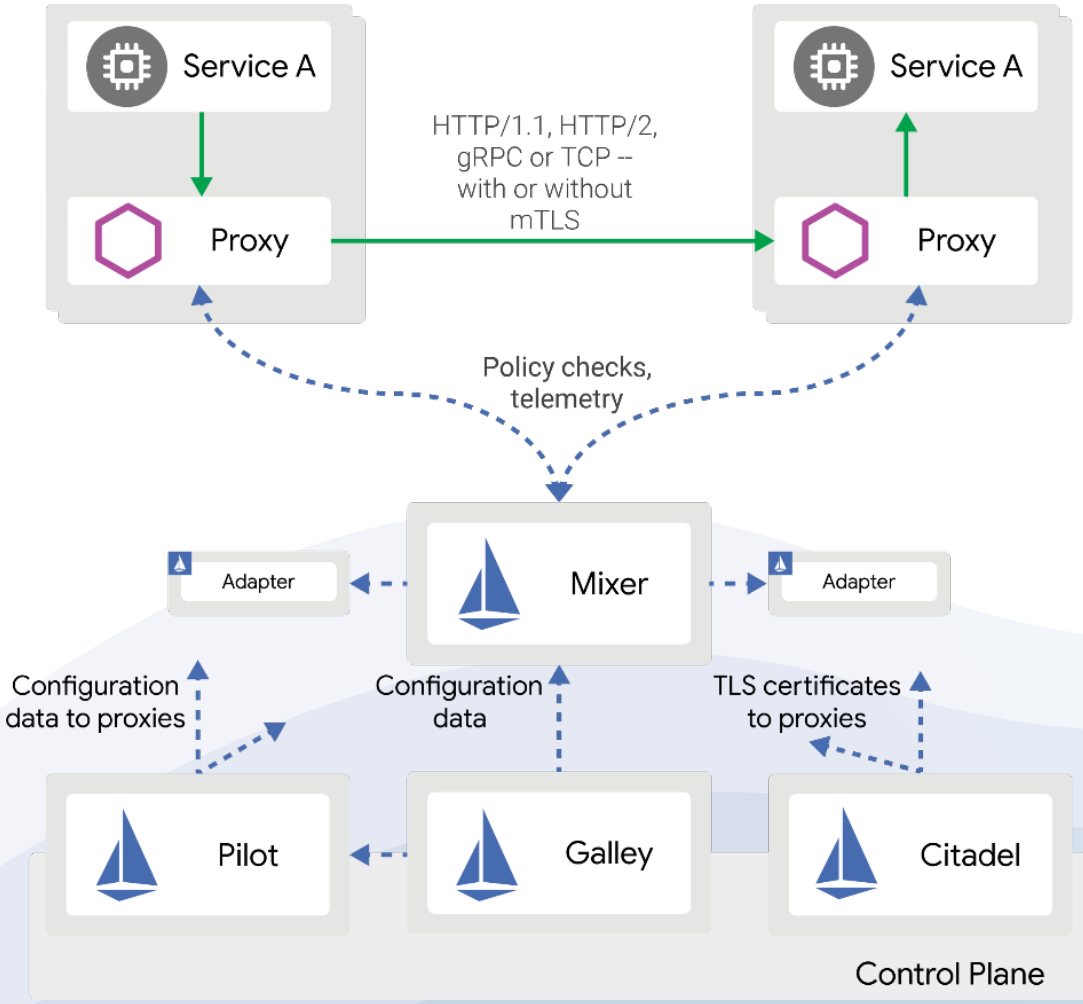


Service Mesh

混合在一个进程内，
应用既有业务逻辑，
也有各种功能

将SDK客户端
的功能剥离
Sidecar专注服务间通讯

业务进程专注于业务逻辑

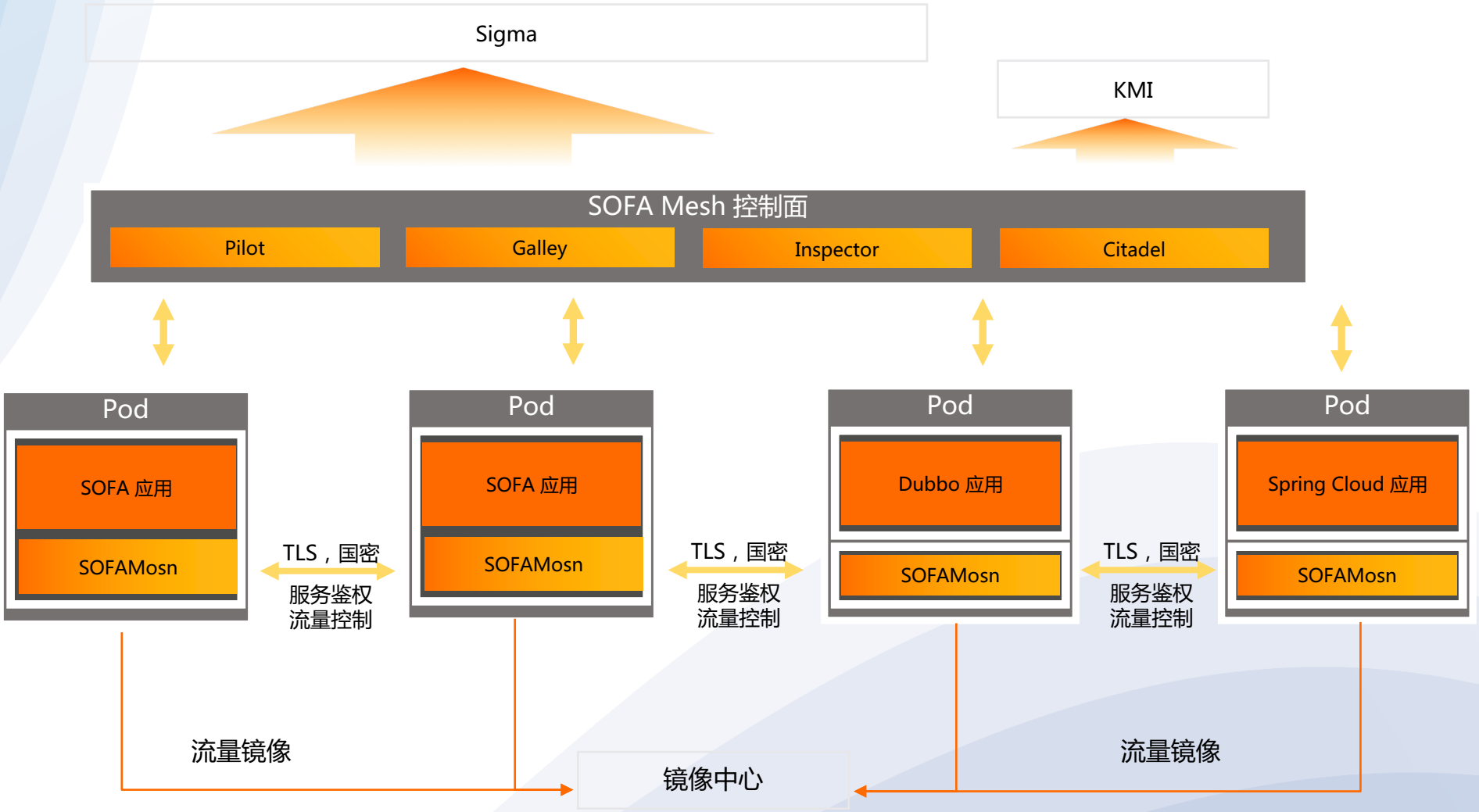


CLOUD NATIVE COMPUTING FOUNDATION
蚂蚁金服 ANT FINANCIAL
ServiceMesher

为什么蚂蚁需要Service Mesh

- 拥抱微服务，云原生
- 异构语言体系融合
- 统一服务治理
- 运维体系有利支撑
- 全局流量管理，打通南北，东西
- 金融级网络安全

为金融业务而生的SOFAMesh



SOFAMosn API 网关

中心化网关

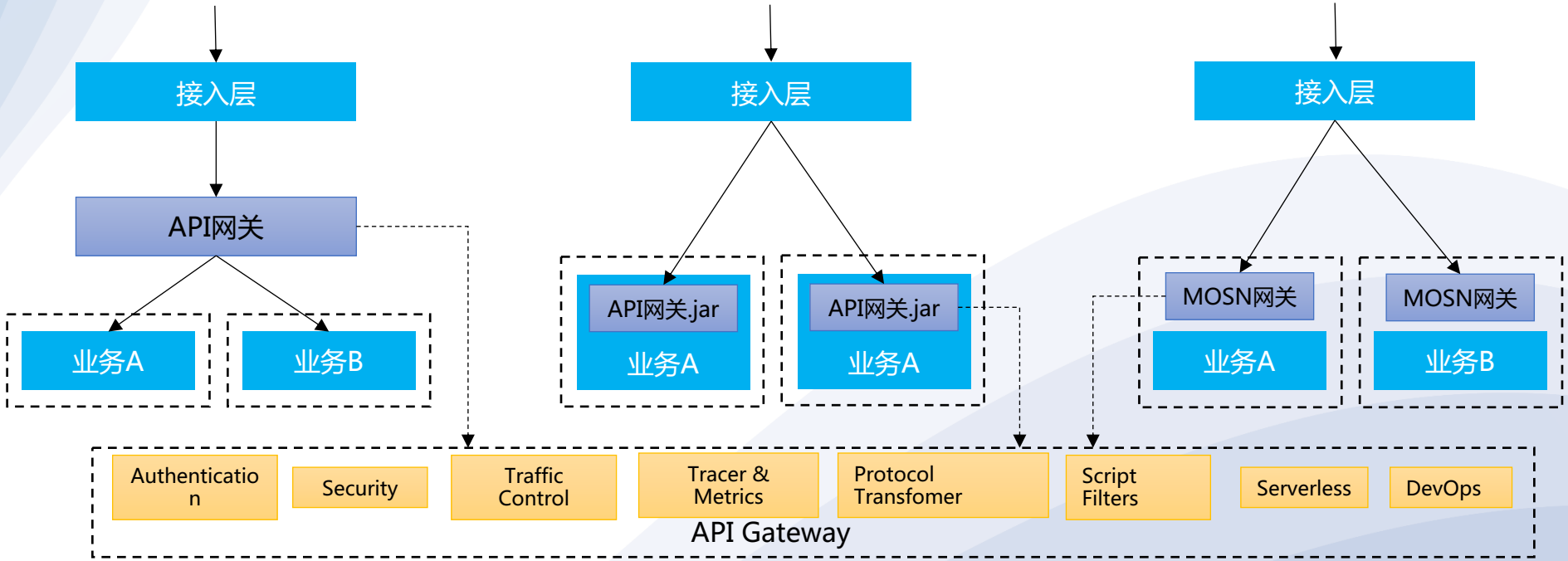
- 流量单点
- 性能瓶颈
- 流量难于评估

去中心化网关

- 流量分散
- 性能提升
- 升级困难
- 无法支持异构系统（Chair、C++）

Mesh化网关

- 基于MOSN，复用能力
- 易于管控升级
- LUA支持
- 计划开源



蚂蚁金服率先大规模落地SOFAMesh

蚂蚁金服100+应用，10w+容器已经mesh化，部分业务链路通过下沉，
RT降低了7%，平稳支撑了618大促。

多协议

SOFARPC
Dubbo
HTTP1.1/2

UDPA

统一数据
平面API

平滑升级

存量连接无损迁移
提升5倍发布效率

安全

TLS双向加密
支持国密算法
WAF
流量镜像

性能

单跳CPU增加5%消耗
0.2ms RT

SOFAMosn

SOFAMosn

https://github.com/sofastack/sofa-mosn

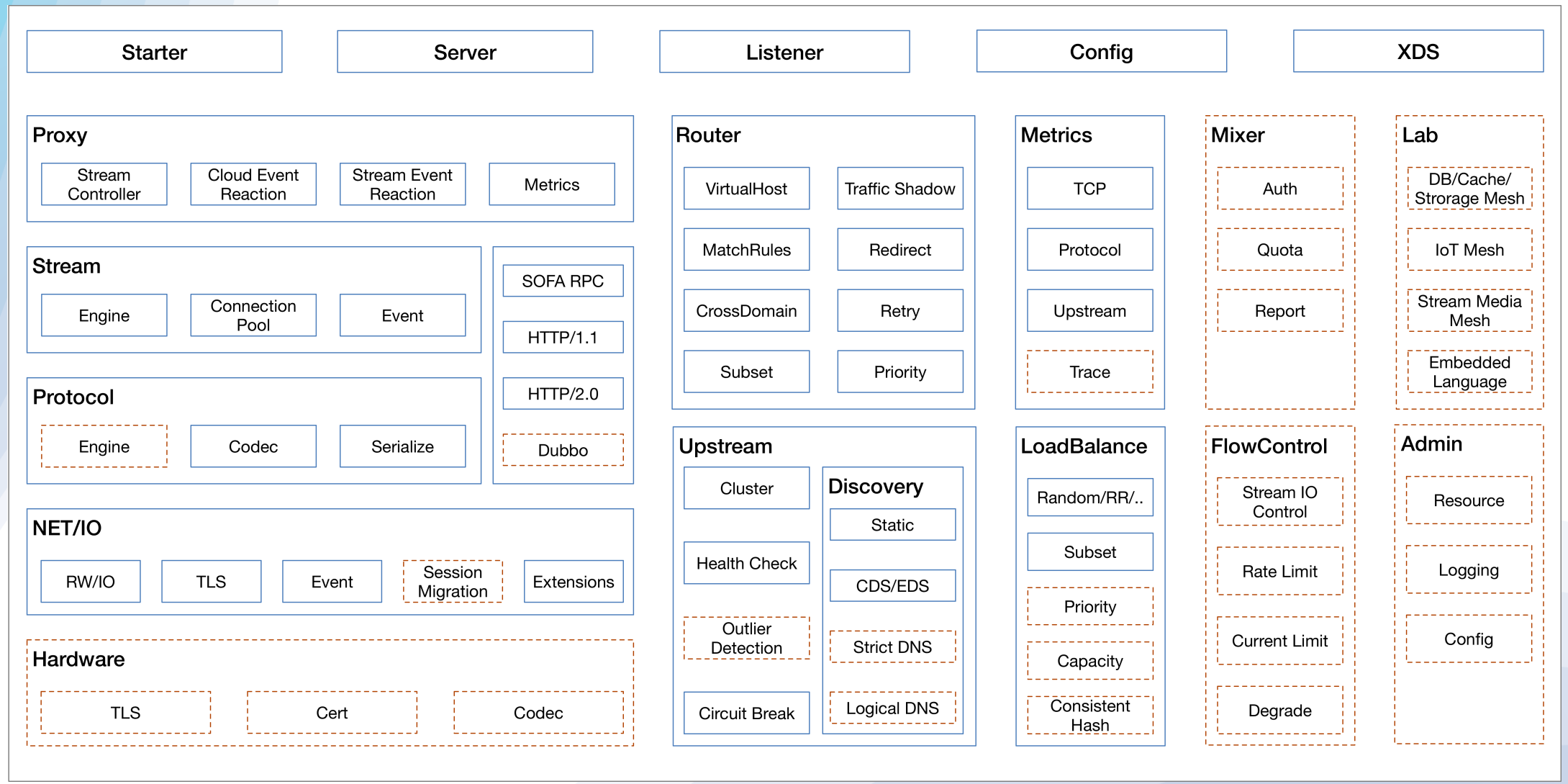
Written in go

SOFAMosn是一个云原生安全网络代理

为什么自研golang版本？

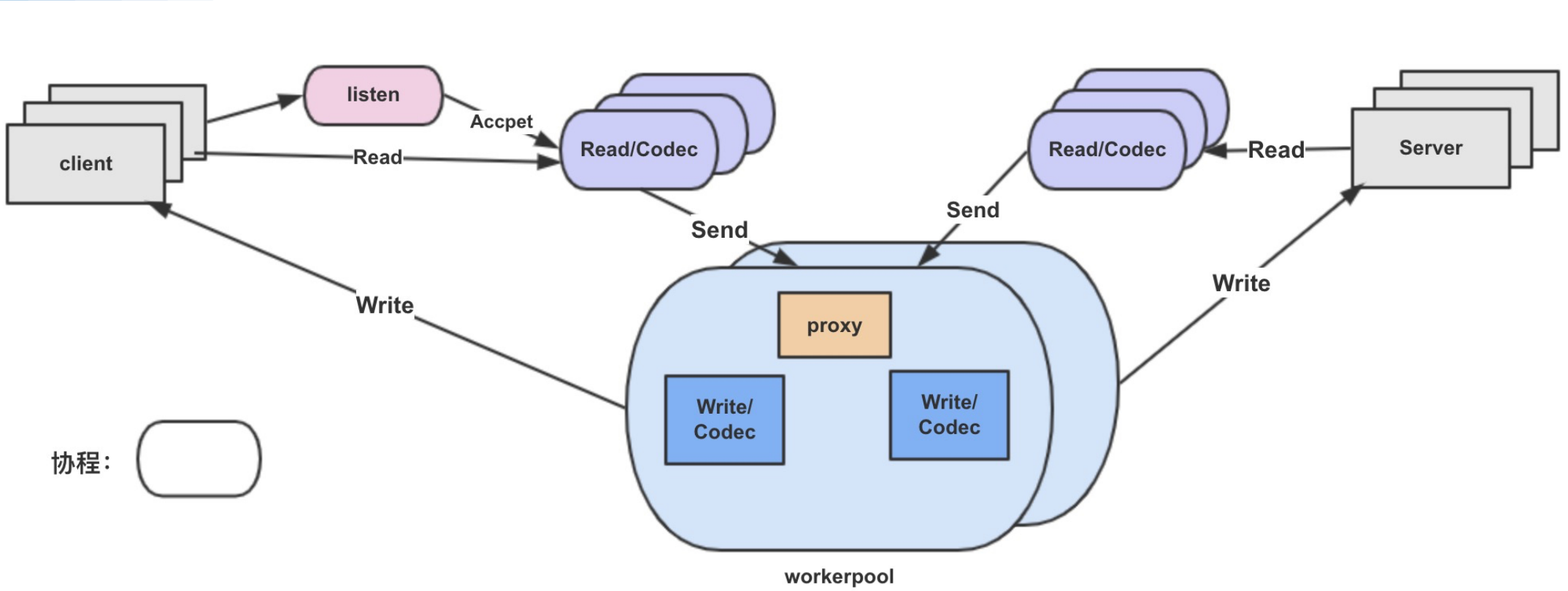
跨团队合作需要考虑技术栈落地成本
Golang性能，成本符合蚂蚁实际需求
近十年的网络代理研发，运维经验

SOFAMosn模块与能力划分



SOFAMosn协程模型

- ✓ 一条TCP连接对应一个Read协程，执行收包，协议解析
- ✓ 一个请求对应一个worker协程，执行业务处理，proxy和Write逻辑



SOFAMosn能力扩展

✓ 协议扩展

MOSN 通过使用同一的编解码引擎以及编/解码器核心接口，提供协议的 plugin 机制，包括支持

- SofaRPC
- HTTP1.x,/HTTP2.0
- Dubbo

✓ NetworkFilter 扩展

MOSN 通过提供 network filter 注册机制以及统一的 packet read/write filter 接口，实现了 Network filter 扩展机制，当前支持：

- TCP proxy
- Fault injection

✓ StreamFilter 扩展

MOSN 通过提供 stream filter 注册机制以及统一的 stream send/receive filter 接口，实现了 Stream filter 扩展机制，包括支持：

- 流量镜像，RBAC鉴权

SOFAMosn能力扩展

心跳检查

```
func (f *healthCheckFilter) OnReceive(ctx context.Context, headers types.HeaderMap, buf types.IoBuffer, trailers types.HeaderMap) types.StreamFilterStatus {
        if cmd, ok := headers.(sofarpc.SofaRpcCmd); ok {
                if cmd.CommandCode() == sofarpc.HEARTBEAT {
                        f.protocol = cmd.ProtocolCode()
                        f.requestID = cmd.RequestID()
                        f.healthCheckReq = true
                        f.handler.RequestInfo().SetHealthCheck(true)

                        if !f.passThrough {
                                f.handleIntercept()
                                return types.StreamFilterStop
                        }
                }
        }
        return types.StreamFilterContinue
}

func (f *healthCheckFilter) handleIntercept() {
        // todo: cal status based on cluster healthy host stats and f.clusterMinHealthyPercentages
        hbAck := sofarpc.NewHeartbeatAck(f.protocol)
        if hbAck == nil {
                log.DefaultLogger.Alertf(types.ErrorKeyHeartBeat, "unknown protocol code: [%x] while intercept healthcheck.", f.protocol)
                //TODO: set hijack reply - codec error, actually this would happen at codec stage which is before this
        }
        f.handler.AppendHeaders(hbAck, true)
}
```

xDS（UDPA） （openresty）

```
"servers":[
   {
     "default_log_path":"stdout",
     "listeners":[
        {
          "name":"serverListener",
          "address": "127.0.0.1:2046",
          "bind_port": true,
          "log_path": "stdout",
          "filter_chains": [{
             "tls_context":{},
             "filters": [
                       {
                  "type": "proxy",
                  "config": {
                     "downstream_protocol": "Http2",
                     "upstream_protocol": "Http1",
                        "router_config_name":"server_router"

                  }
                       },
                       {
                       "type":"connection_manager",
                       "config":{
                         "router_config_name":"server_router",
                         "virtual_hosts":[{
                               "name":"serverHost",
                               "domains": ["*"],
                               "routers": [
                                 {
                                       "match":{"prefix":"/"},
                                       "route":{"cluster_name":"serverCluster"}
                                 }
                               ]
                         }]

                       }
```

xDS全动态配置

```
"cluster_manager":{
   "clusters":[
      {
        "Name":"serverCluster",
        "type": "SIMPLE",
        "lb_type": "LB_RANDOM",
        "max_request_per_conn": 1024,
        "conn_buffer_limit_bytes":32768,
        "hosts":[
           {"address":"127.0.0.1:8080"}
        ]
      },
      {
        "Name": "clientCluster",
        "type": "SIMPLE",
        "lb_type": "LB_RANDOM",
        "max_request_per_conn": 1024,
        "conn_buffer_limit_bytes":32768,
        "hosts":[
           {"address":"127.0.0.1:2046"}
        ]
      }
   ]
},
```

在Service Mesh场景下网络代理有着不同于接入层的挑战

- 无论应用是否接入Mesh，接入了多少Mesh，都需要保证可正常处理请求，做到可灰度、可回滚的**兼容，平滑迁移**
- 通用的框架能力（SOFAMosn/Envoy）无法直接满足复杂的、定制的业务能力，需要进行针对性的**扩展**实现
- 需要**融合**主站已有的应用体系，如注册中心、配置中心等，这些也需要扩展实现
- 大规模场景下需要面对的**资源占用，自动化问题、性能问题，稳定性**问题

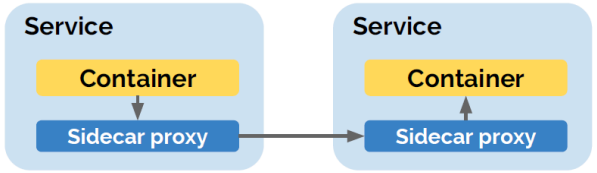
兼容问题

- 不同的应用，部分Mesh化
- 同一个应用，部分Mesh化
- 蚂蚁基础设施适配
- TLS加密链路

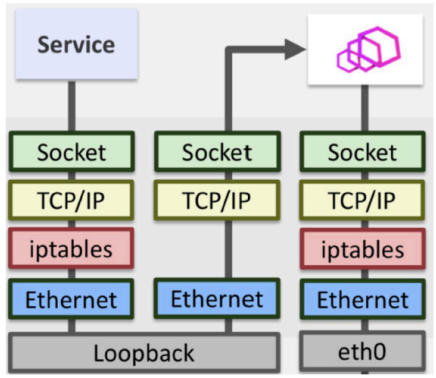
平滑迁移

BPF-based servicemesh
Acceleration

cilium



How it really looks:



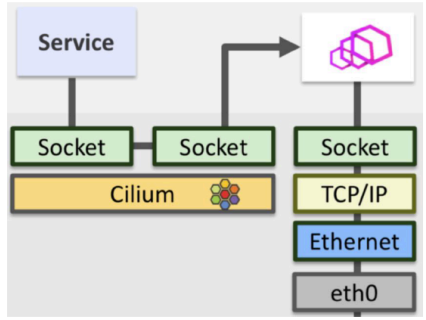
24

Localhost or Iptables

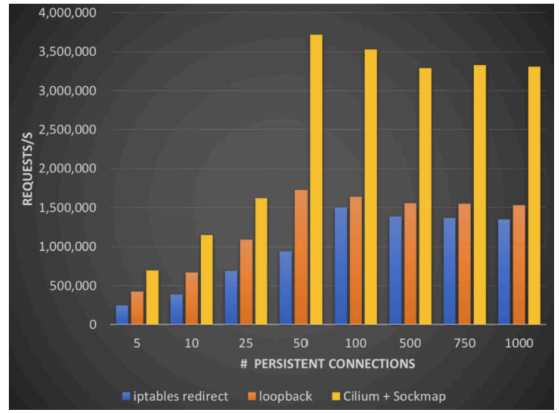
BPF-based servicemesh
Acceleration

cilium

Accelerate the service to sidecar communication



~3.5x performance improvement



25

透明劫持和加速

大规模问题

10万+实例

- 对控制平面性能，稳定性带来巨大挑战
- 单实例数万路由节点，数千路由规则，不仅占用内存，对路由匹配性能也有较大影响

动态服务发现

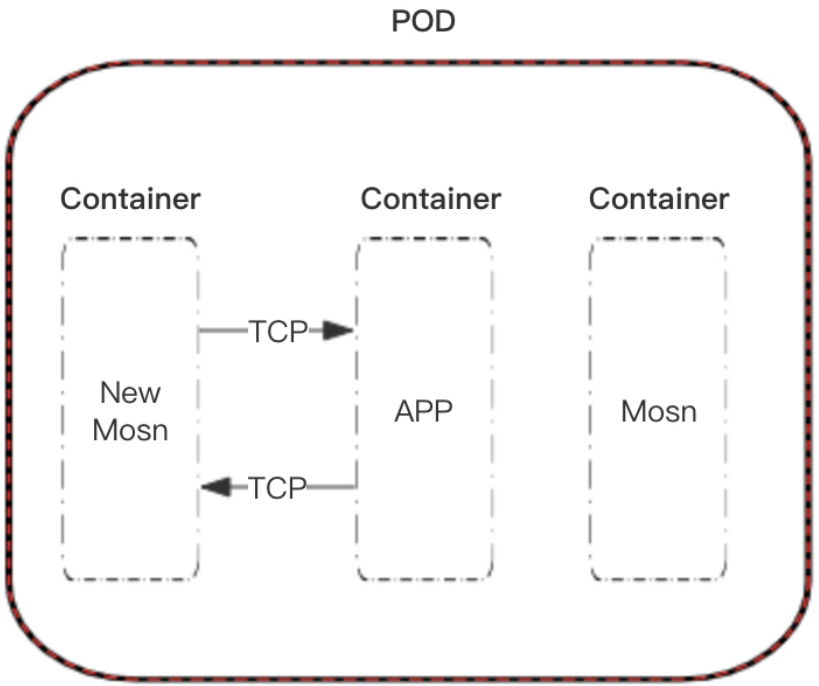
- 海量，高频的发布订阅动作

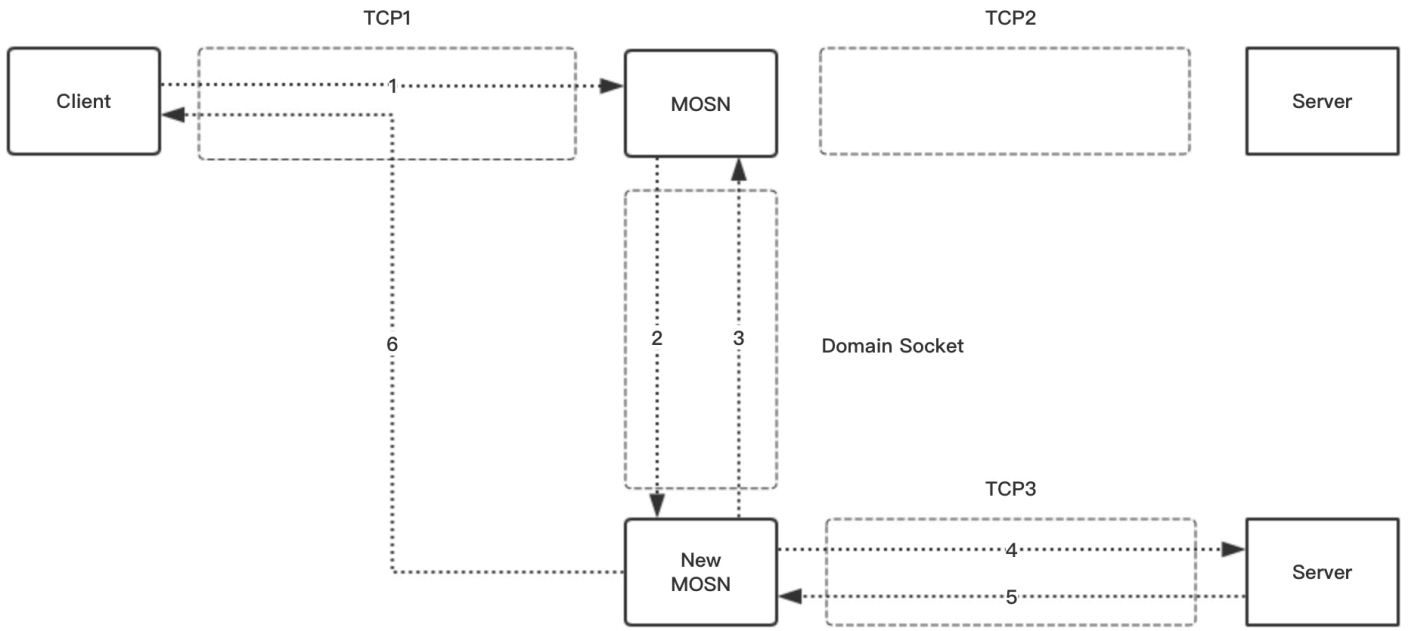
运维

- 发布分组策略，间隔策略
- SOFAMosn发布业务无感知，平滑升级

平滑升级

- 支持新老 mosn 容器间链接、metrics 无损迁移，升级全程无重新建联
- 支持对服务中 mosn 容器进行无损平滑升级，即“给奔跑的汽车换轮胎”
- 支持 SOFARPC、HTTP/1.x、消息、TLS 等多种协议无损迁移





资源问题

CPU

- Cpuset模式，与业务App共享独占核
- Cpushare模式，与应用容器共享物理机Cpu资源
- 根据具体业务情况设置Cpu资源

内存

- ~~使用应用内存的1/16~~
- 与业务Share内存，最大限制使用1G，存在超卖问题，触发Pod 级别的direct reclaim问题

磁盘

- Sidecar与业务容器共享磁盘，并且不受容器启动顺序对磁盘分配的影响，单独mount配置文件

- 不同业务不同资源占用的精细化调配

性能问题

GOMAXPROCS ：Cpu消耗与RT的tradeoff

优化GC策略升级1.12版本，MADV_FREE，MADV_DONTNEED带来的影响

Chan的吞吐极限，减少主业务数据的传递

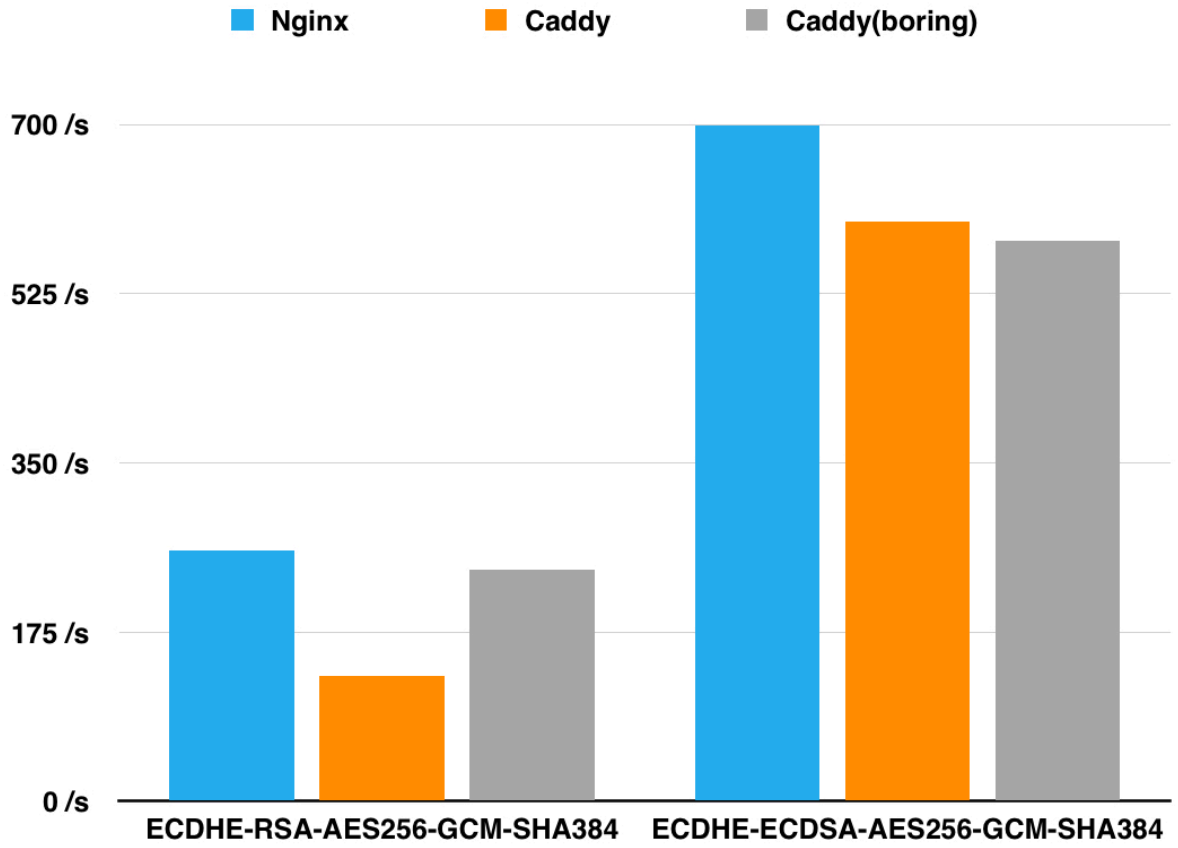
CGO对于TLS 签名计算有83%的性能衰退，AES对称加密

Unix Domain socket较TCP socket 提升8%性能

使用tmpfs 或者 mmap MAP_LOCKED 优化IO负载较高对共享内存刷page cache的影响

TLS性能

- ✓ go在RSA上没有太多优化，go-boring（CGO）的能力是go的1倍。
- ✓ p256在go上有汇编优化，ECDSA优于go-boring
- ✓ 在AES-GCM对称加密上，go的能力是go-boring的20倍
- ✓ 在SHA，MD等HASH算法也有对应的汇编优化
- ✓ 对Go-GMSSL汇编优化



HTTP性能数据

Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
linux kernel 3.10.0-327
Cpu cores 4
wrk --latency -d 1m -c 100 -t 4 http://10.210.176.123:12220/1k.html

指标	SOFAMosn	Envoy
QPS	33674	52516
CPU	390%	370%
MEM	30M	22M
RT(Avg)	3.08ms	2.16ms
RT(50%)	2.48ms	1.45ms
RT(75%)	3.42ms	1.81ms
RT(90%)	4.66ms	2.44ms
RT(99%)	6.78ms	5.67ms

总结

应用网络SDN

- 南北，东西应用层流量的全局管控，调度和安全能力

通信能力持续升级

- 随通信基础设施，通信场景的变化而演进

安全与合规

- 全方位的安全防护，全链路加密，应用层的零信任网络
- 金融级的通信安全基础设施

关于未来

- 云原生，多云混合云时代，南北，东西流量的边界逐渐模糊
- 应用网络代理层部分能力固化，下沉至系统网络栈或者智能硬件设备
- Sidecar -> Proxyless -> Networkless
- 物理通信基础设施的升级势必带来应用网络层的变革

谢谢！

CLOUD NATIVE COMPUTING FOUNDATION

蚂蚁金服 ANT FINANCIAL

ServiceMesher