

链家网高可用架构演进



董金勇

LIANJIA.COM



促进软件开发领域知识与创新的传播



关注InfoQ官方微信
及时获取ArchSummit
大会演讲视频信息



全球软件开发大会 [北京站]

2017年4月16-18日 北京·国家会议中心
咨询热线: 010-64738142



全球架构师峰会 2016 [深圳站]

2017年7月7-8日 深圳·华侨城洲际酒店
咨询热线: 010-89880682

自我介绍

工作经历：

网易


百度

淘宝

链家网


特点：

文艺



董金勇

北京 海淀



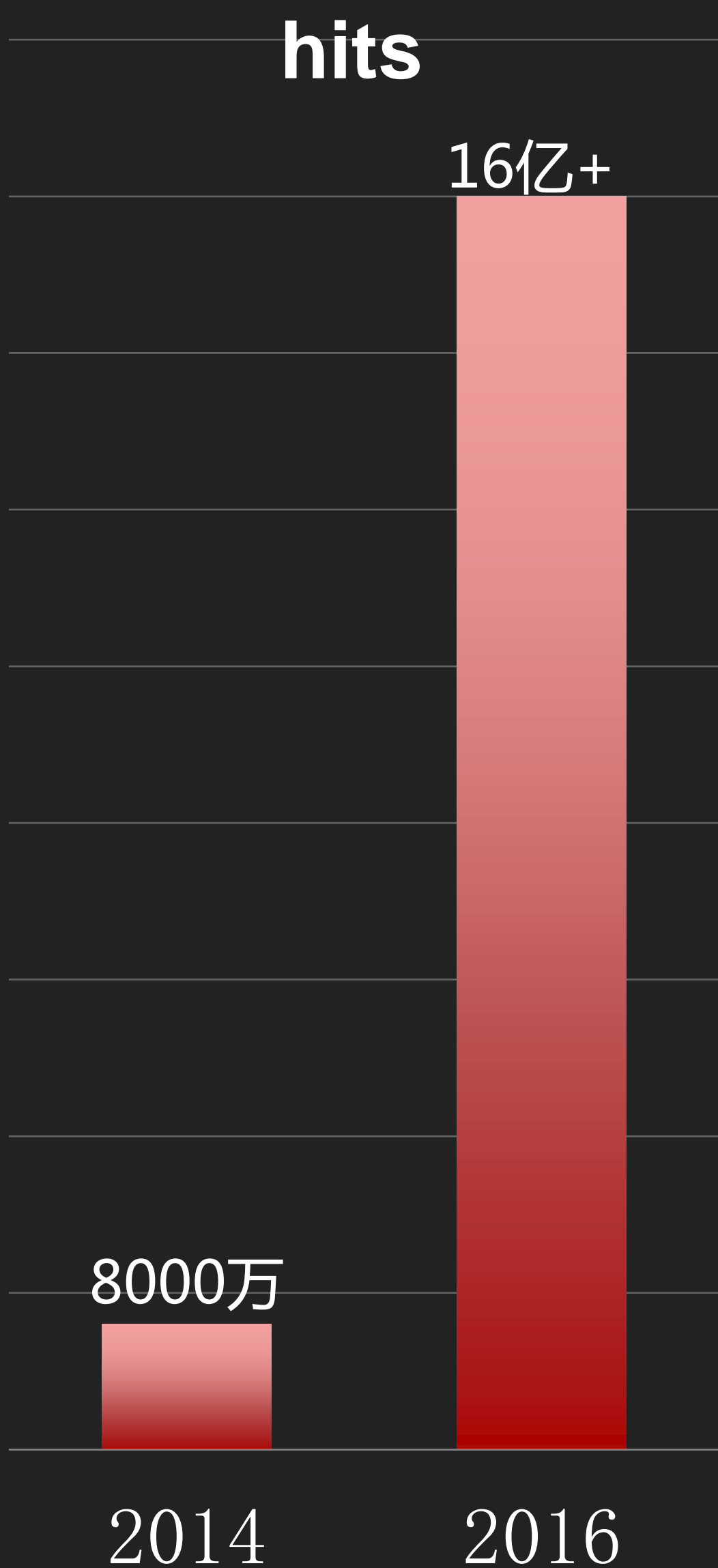
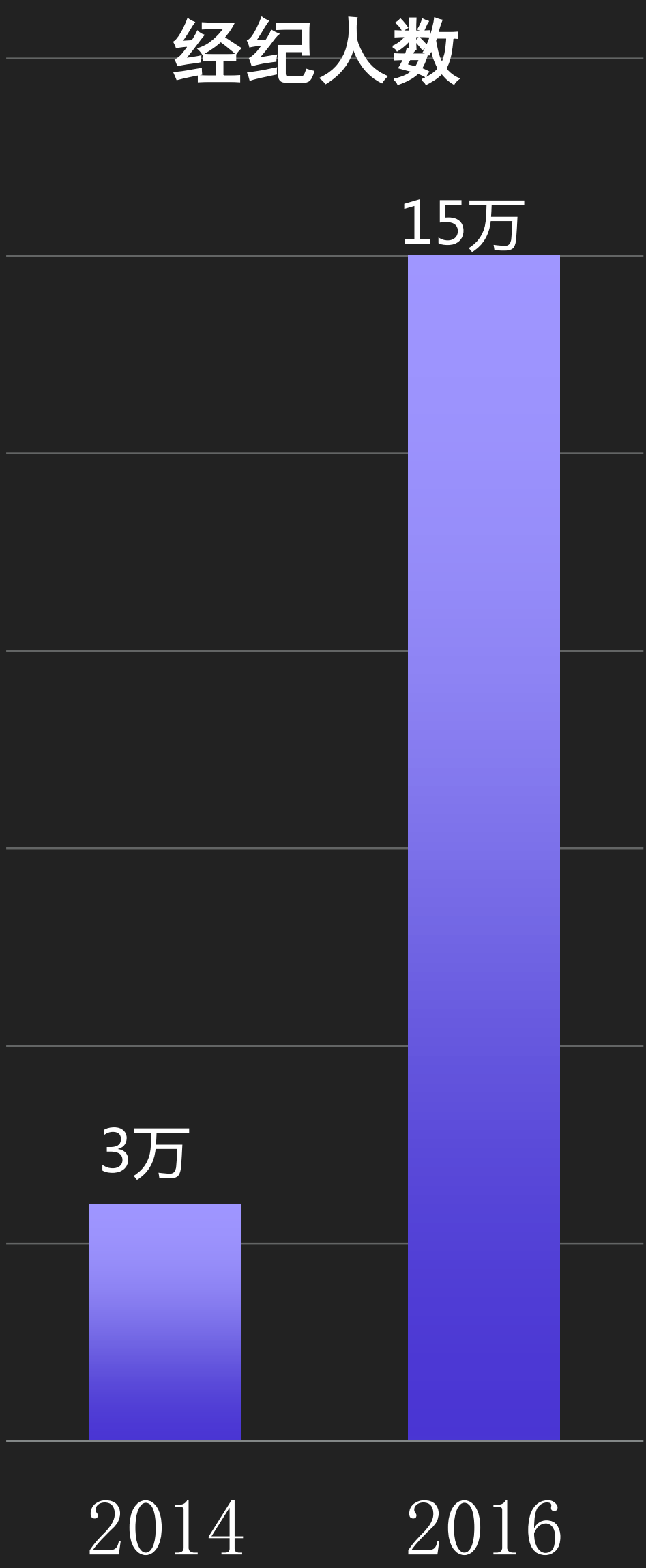
扫一扫上面的二维码图案，加我微信

提纲

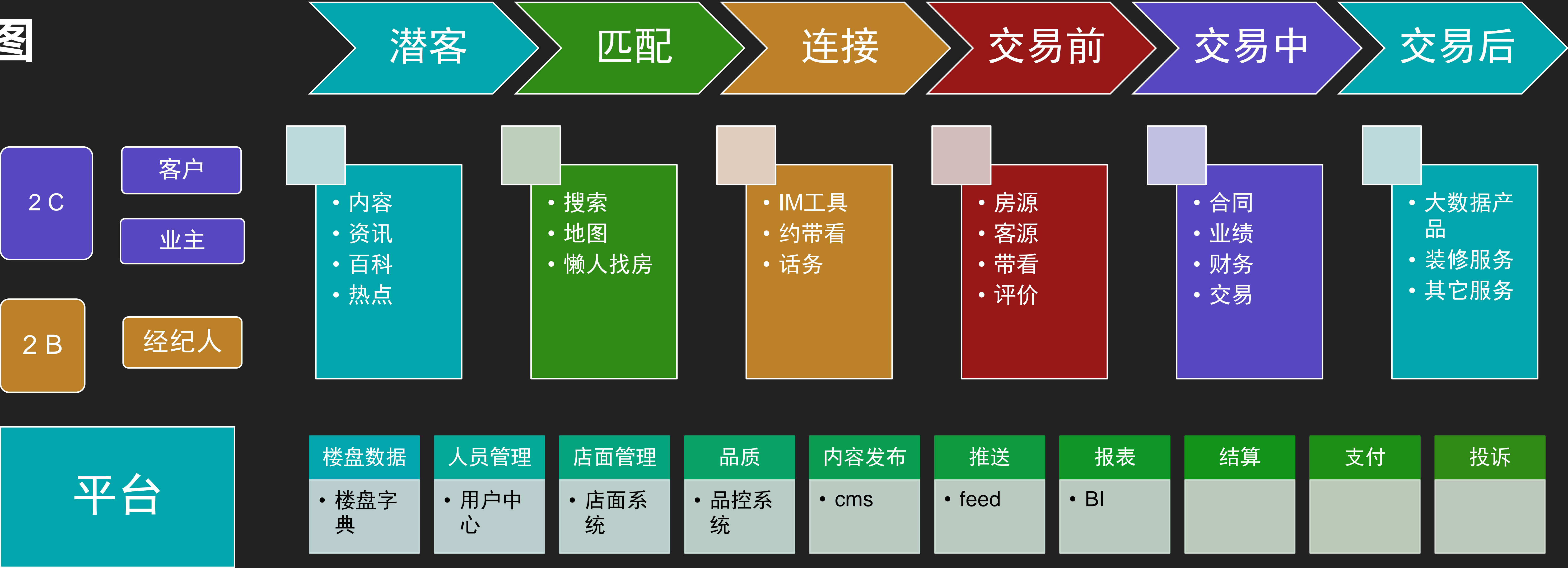
- 一、架构之痛
- 二、高可用架构演进与实践
- 三、总结



业务背景

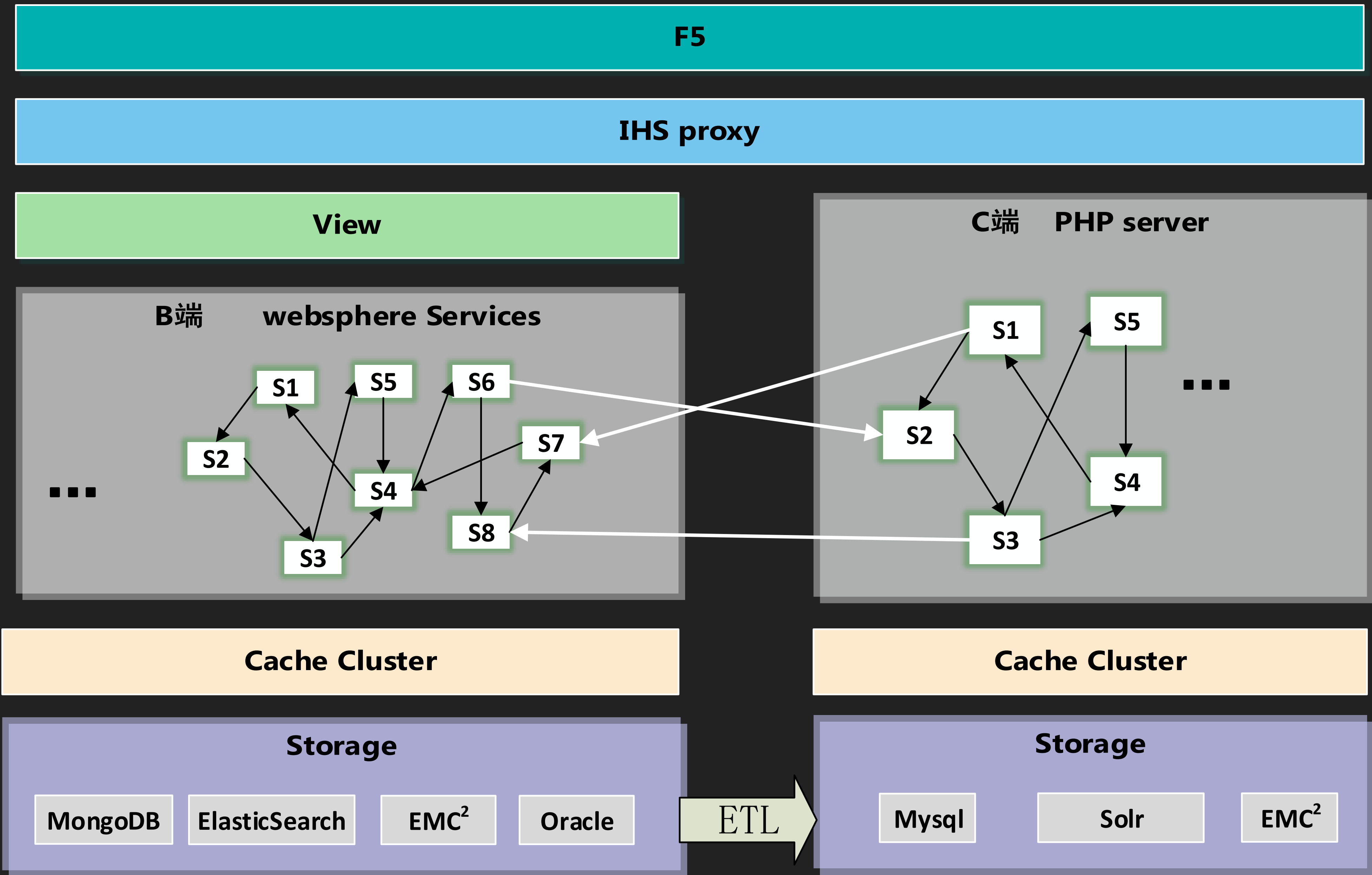


业务架构图



原系统架构

- 系统个数：
300+
- 单库表个数：
10000+
- 成本高达数亿.....



遇到的问题

- 可用性低：99%
- 性能差：单机QPS<50
- 强耦合
- 扩展性差、数据一致性差、容错能力差.....
- 可维护性差：上线难，迭代难

提纲

- 一、架构之痛
- 二、高可用架构演进与实践
- 三、总结



01.
会话层

02.
服务层

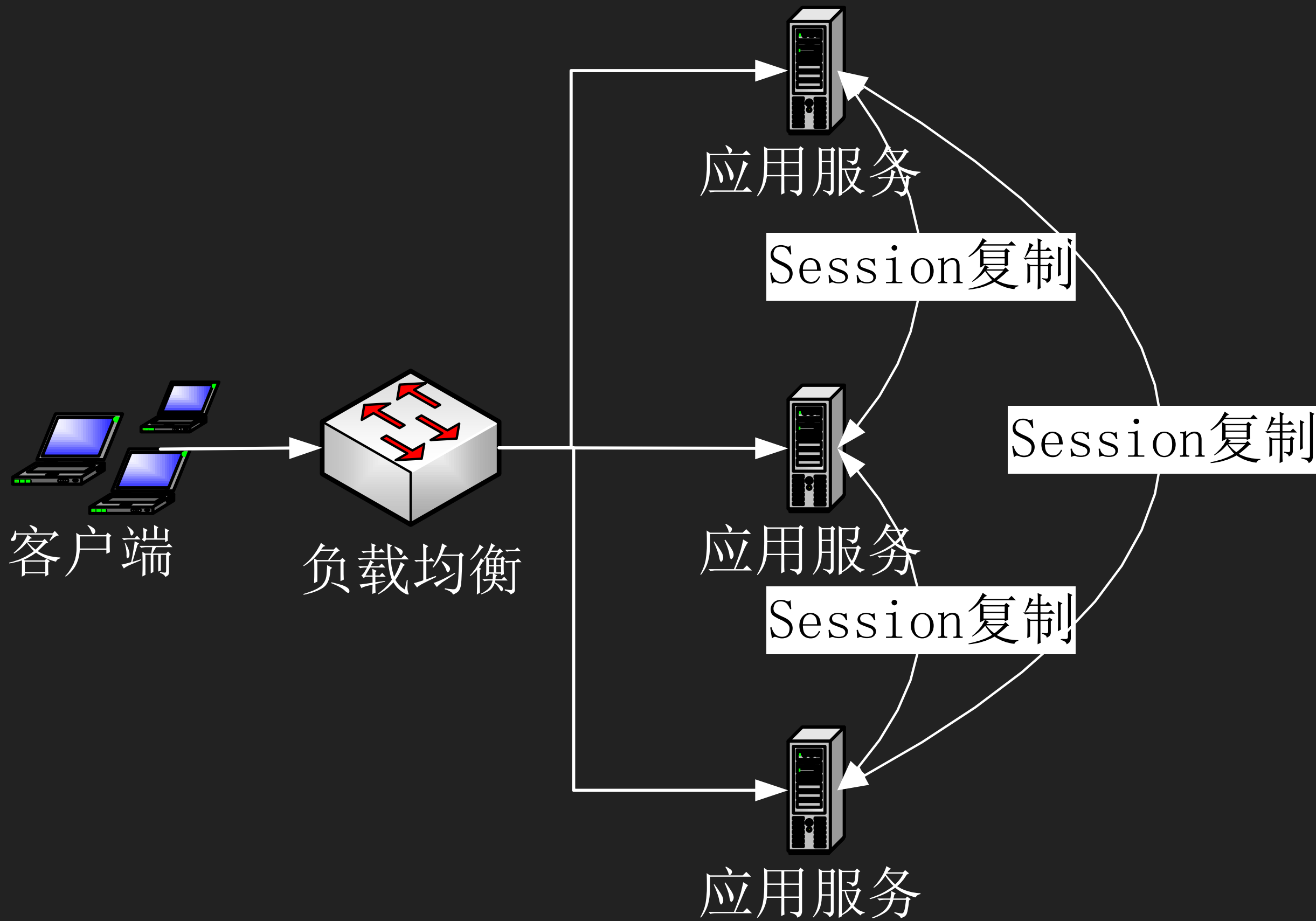
03.
数据层

04.
基础设施体系建设

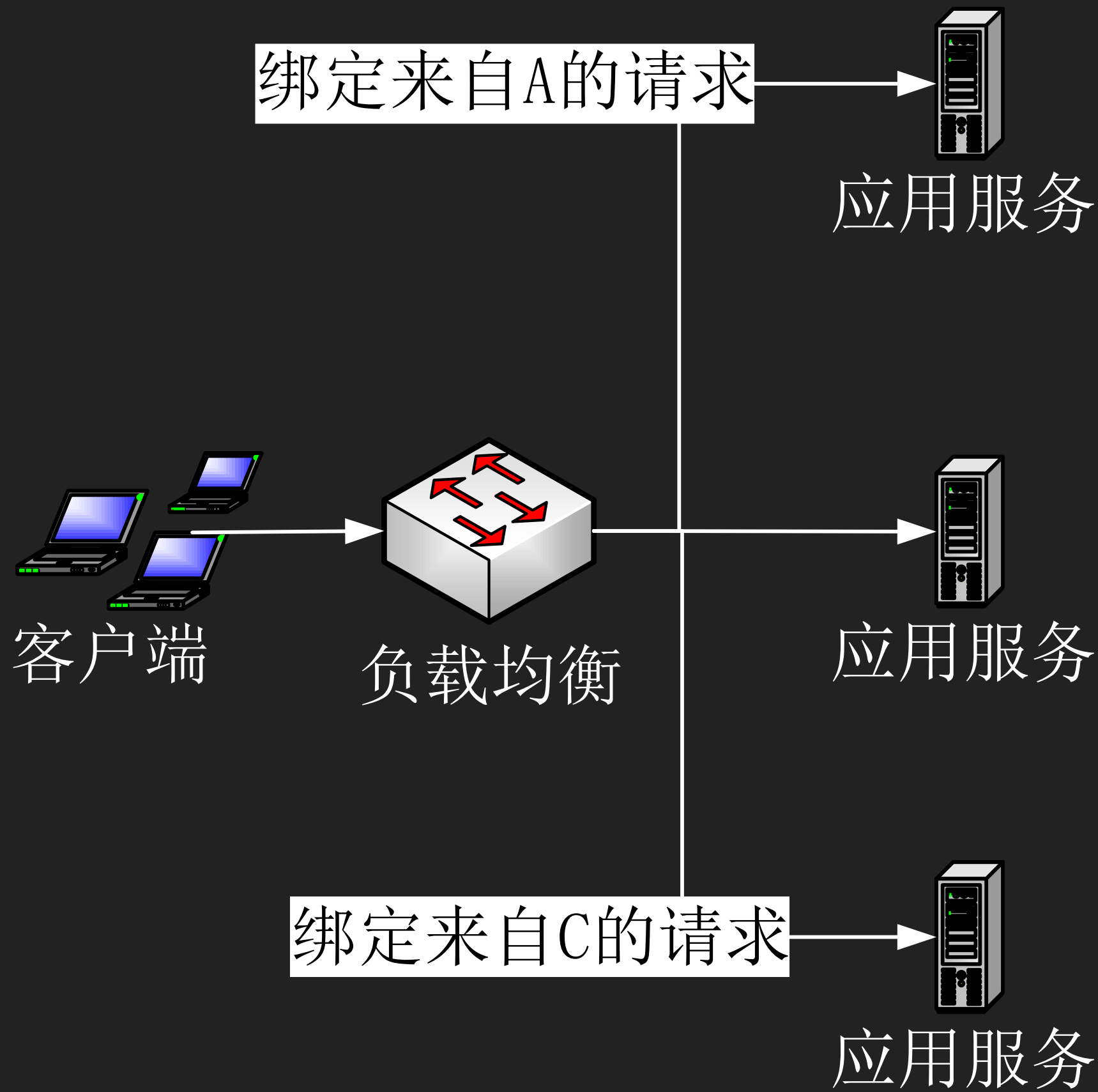
会话层

遇到问题：

- 容量不足
- 串号
- 恢复困难



原B端方案



原C 端方案

适度超前设计

D-I-D原则：

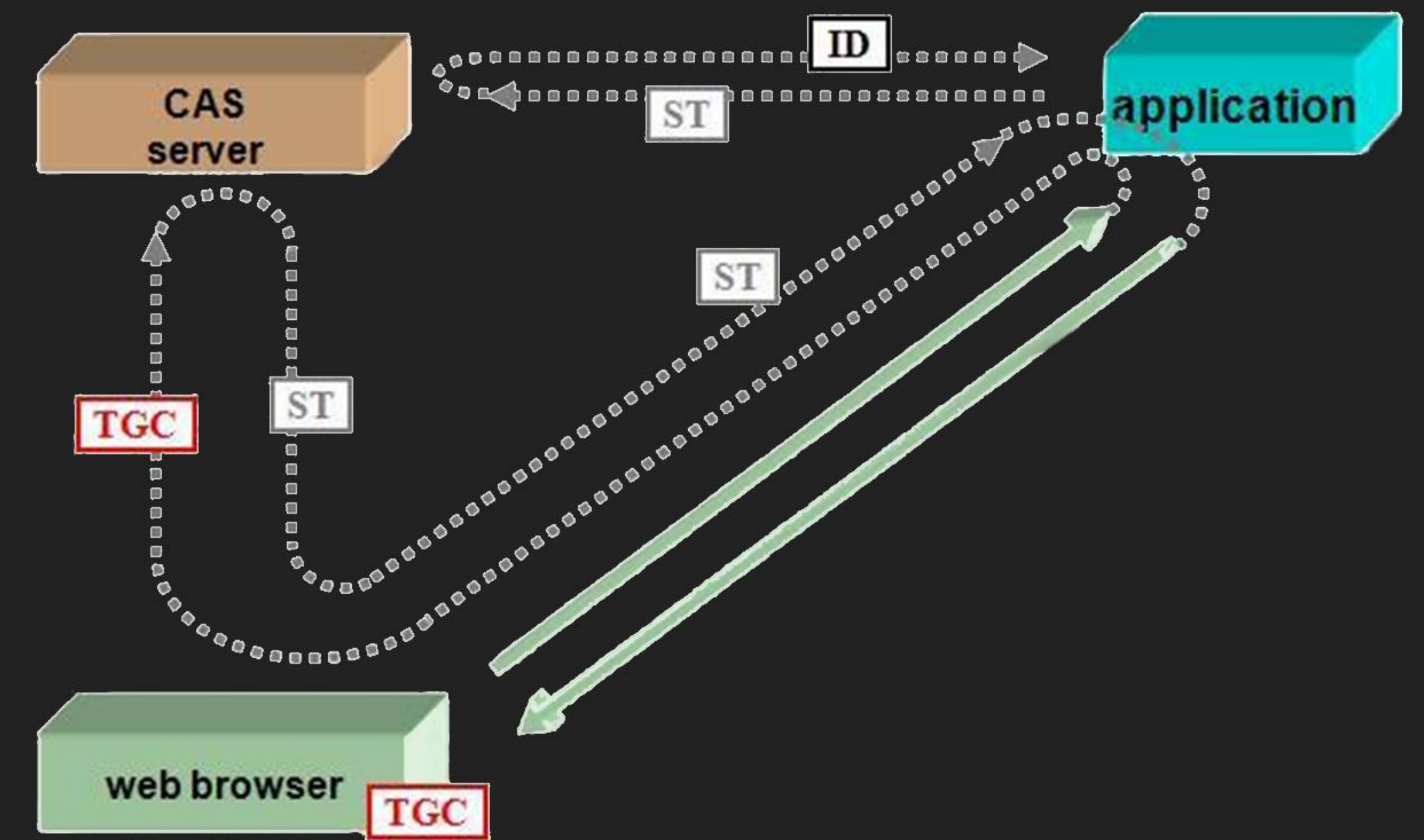
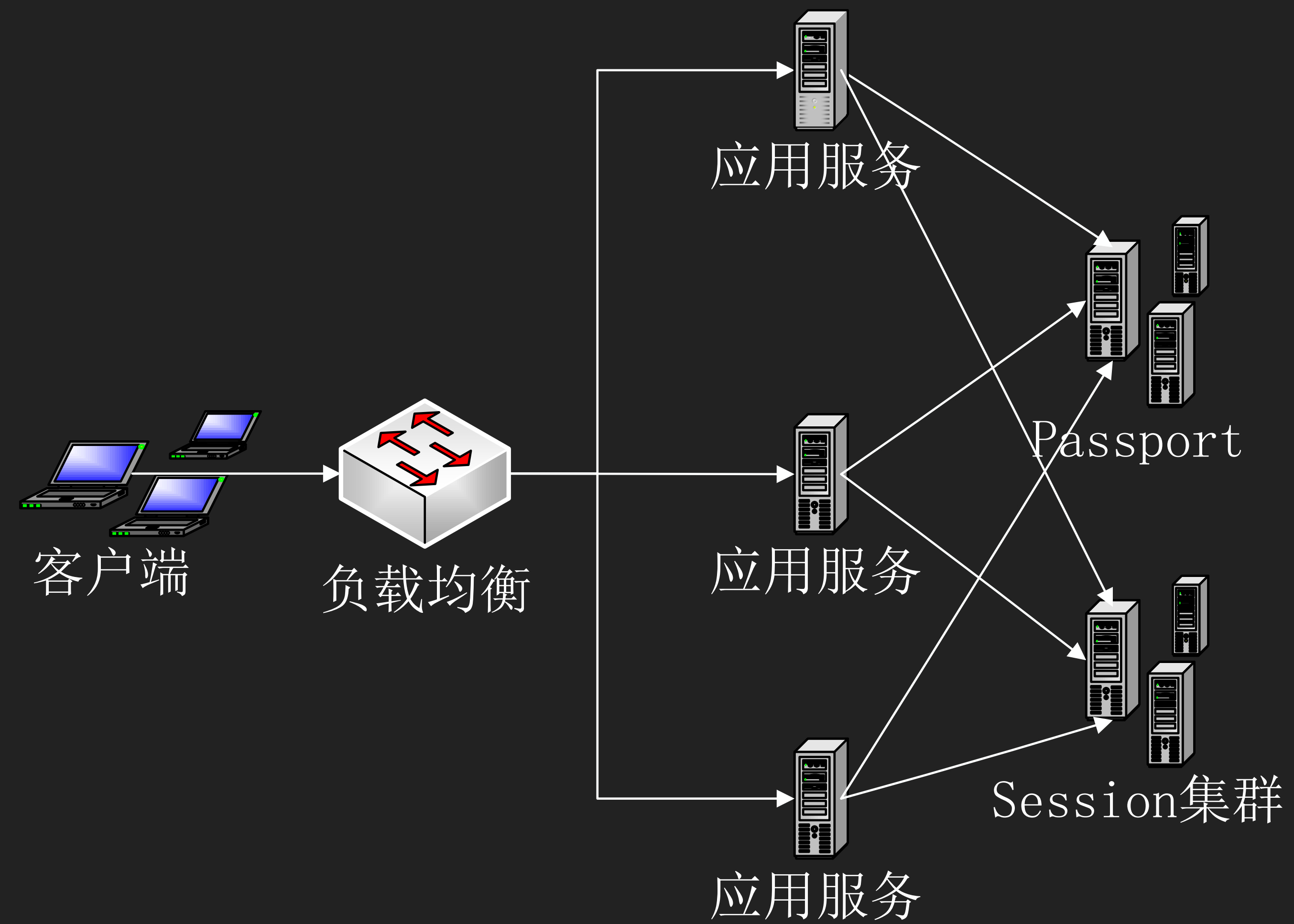
DESIGN：按20倍体量设计

IMPLEMENT：按3倍体量实现

DEPLOY：按1.5倍体量部署

——AKF Partners

DESIGN :
IS SSO NEEDED ?

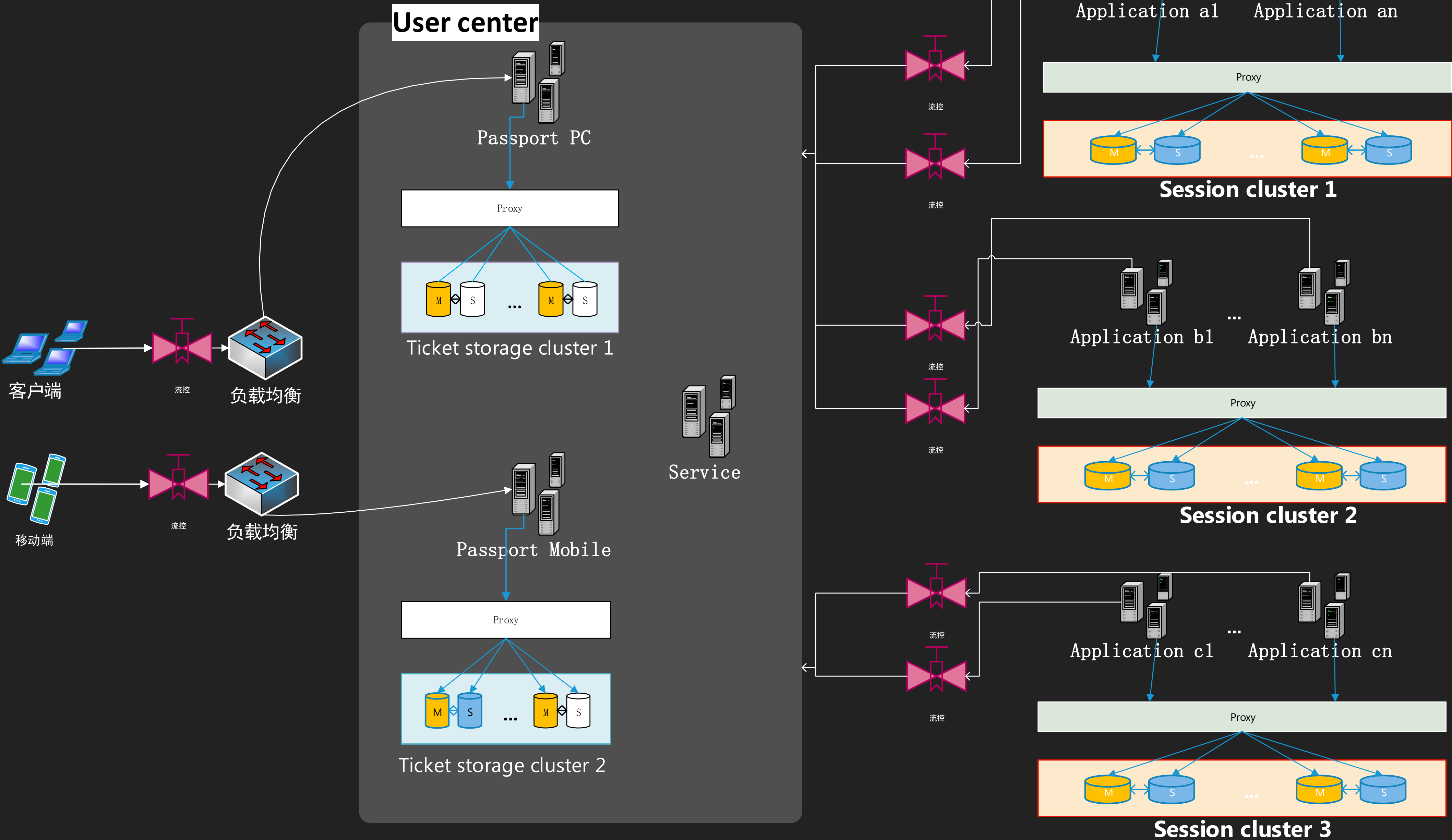


站在巨人肩上——改造CAS

IMPLEMENT:

超前实现防御措施

- 白盒监控
- 独立熔断能力
- 易扩容
- FAILOVER
- 去中心化



DEPLOY TIPS :

- BUFFER BUFFER BUFFER !
- SSD持久化
- 业务分级隔离
- SESSION、CACHE分开

01.
会话层

02.
服务层

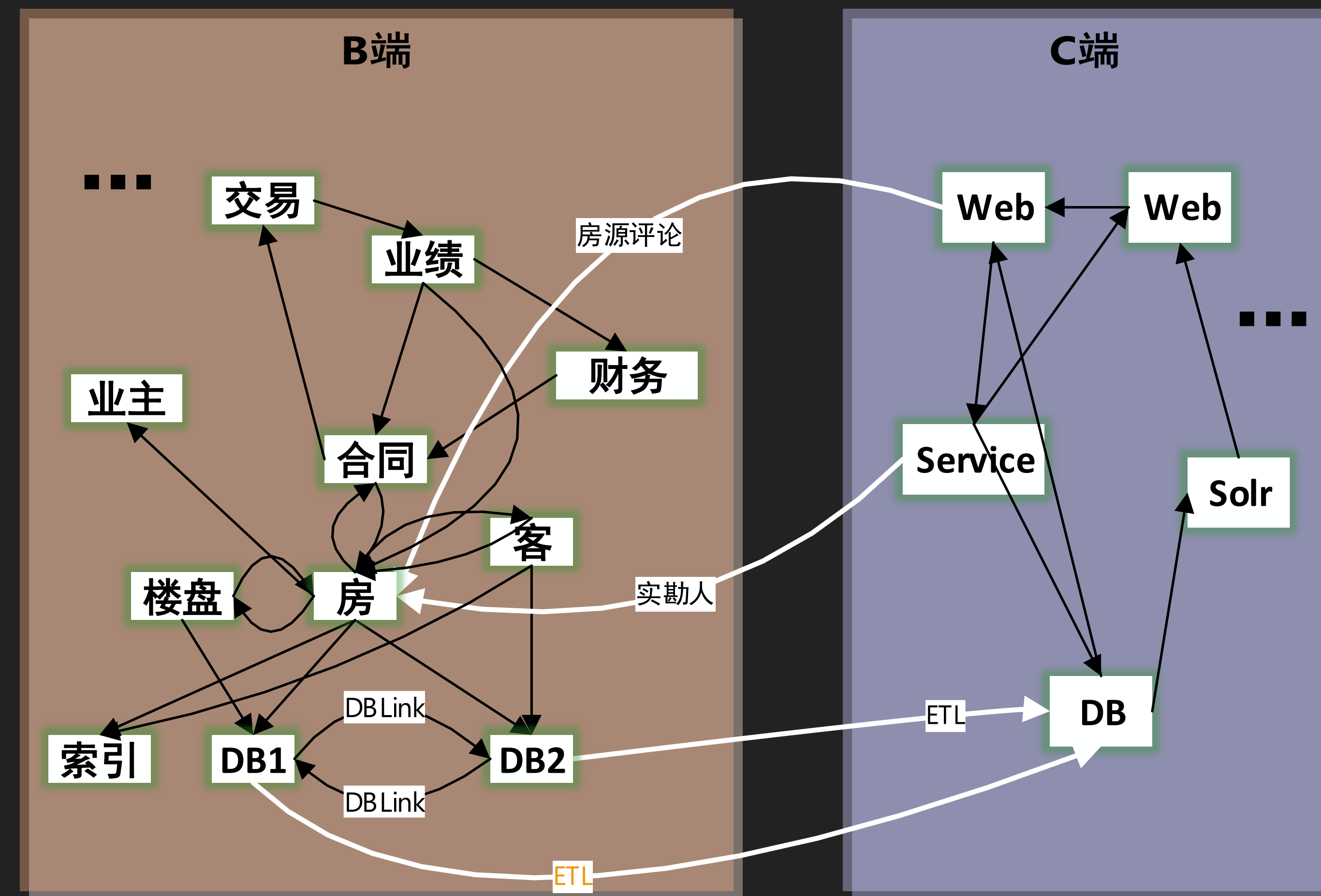
03.
数据层

04.
基础设施体系建设

服务层

What happened?

- 系统数量多达300+
- 跨系统
- 业务迅速增长10倍
- 访问时间集中
- 数据写入多入口
- 状态多
- 强耦合
- 性能差
- 数据一致性差
- 可维护性差
- 扩展性差
- 容错能力差.....



Why?

逃不出的法则

——康威定律

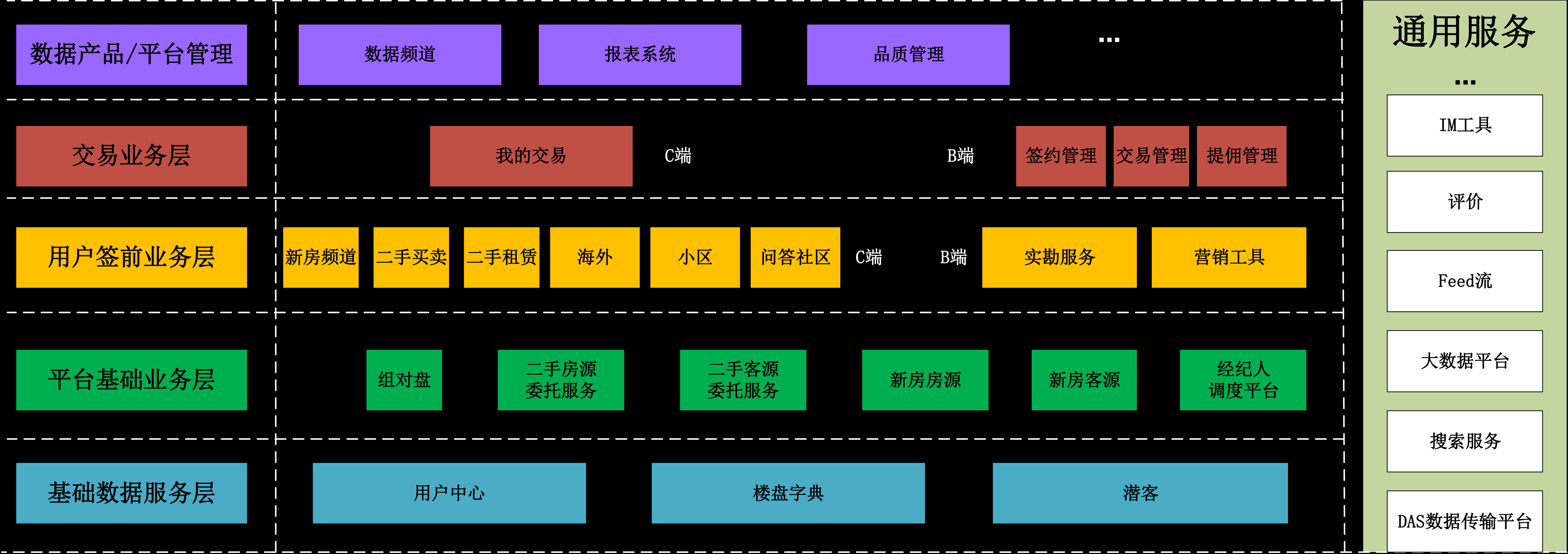
Conway's Law

How?

- KISS原则
- 分层依赖原则
- 解耦——异步化
- CAP原理：CP → AP
- SMART原则



- 分层依赖
- 服务分级
- 功能分级



(SMART - Specific)

SLA度量体系

(SERVICE LEVEL AGREEMENT)

平台基础业务层 SLO (SERVICE LEVEL OBJECTIVES) :

- 99%的简单业务接口响应时间<30MS
- 99%的复杂业务接口响应时间<100MS
- 单机QPS>500
- 可用性：99.99%
- 集群最大处理能力：*****每秒

(SMART - Measurable)

服务化 ——服务治理

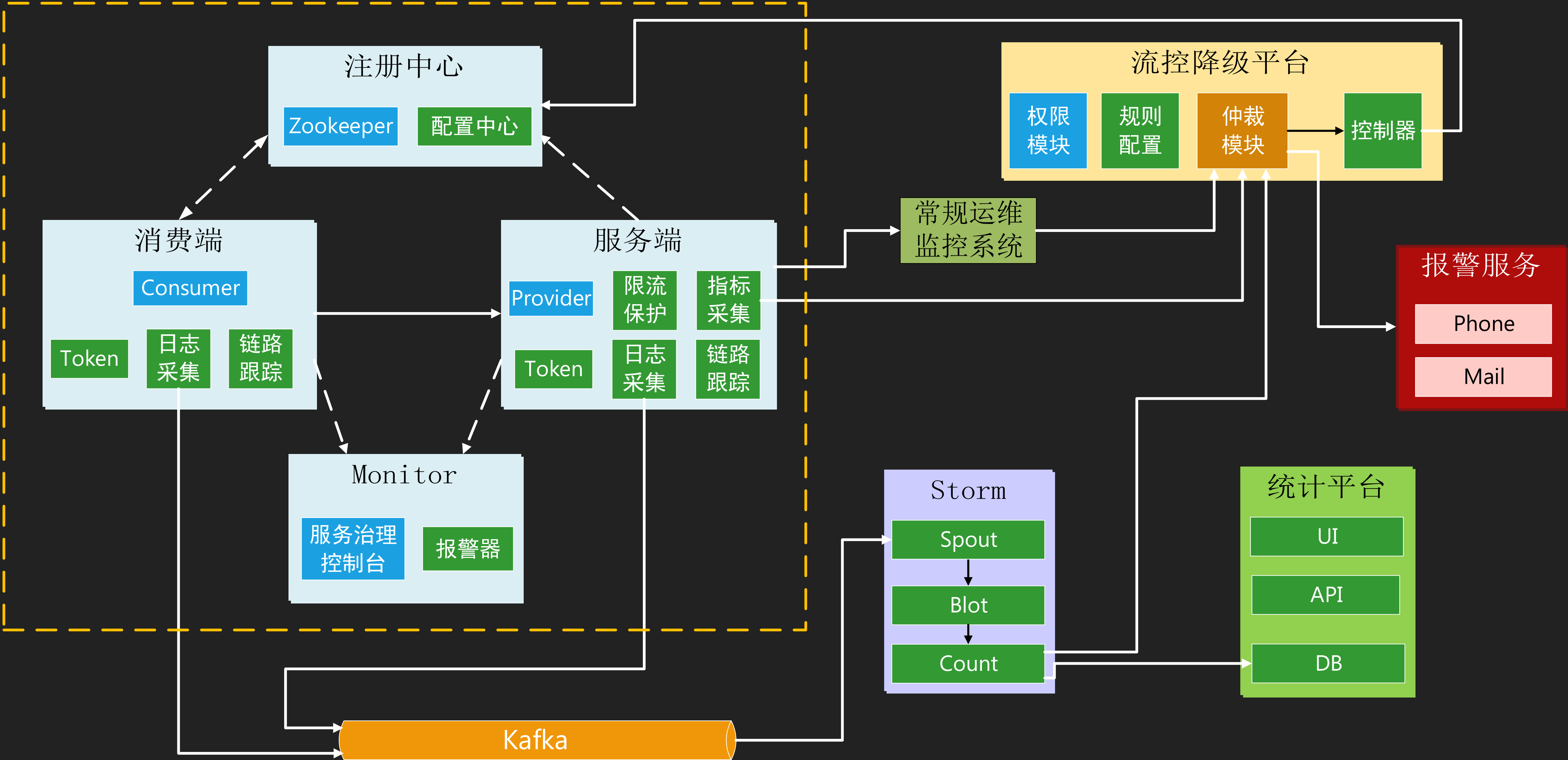
- 发现问题
- 解决问题

(SMART - Attainable)



服务框架

- 服务发现
- 故障转移
- 去中心化
- 监控
- 流控
- 降级
- 日志平台

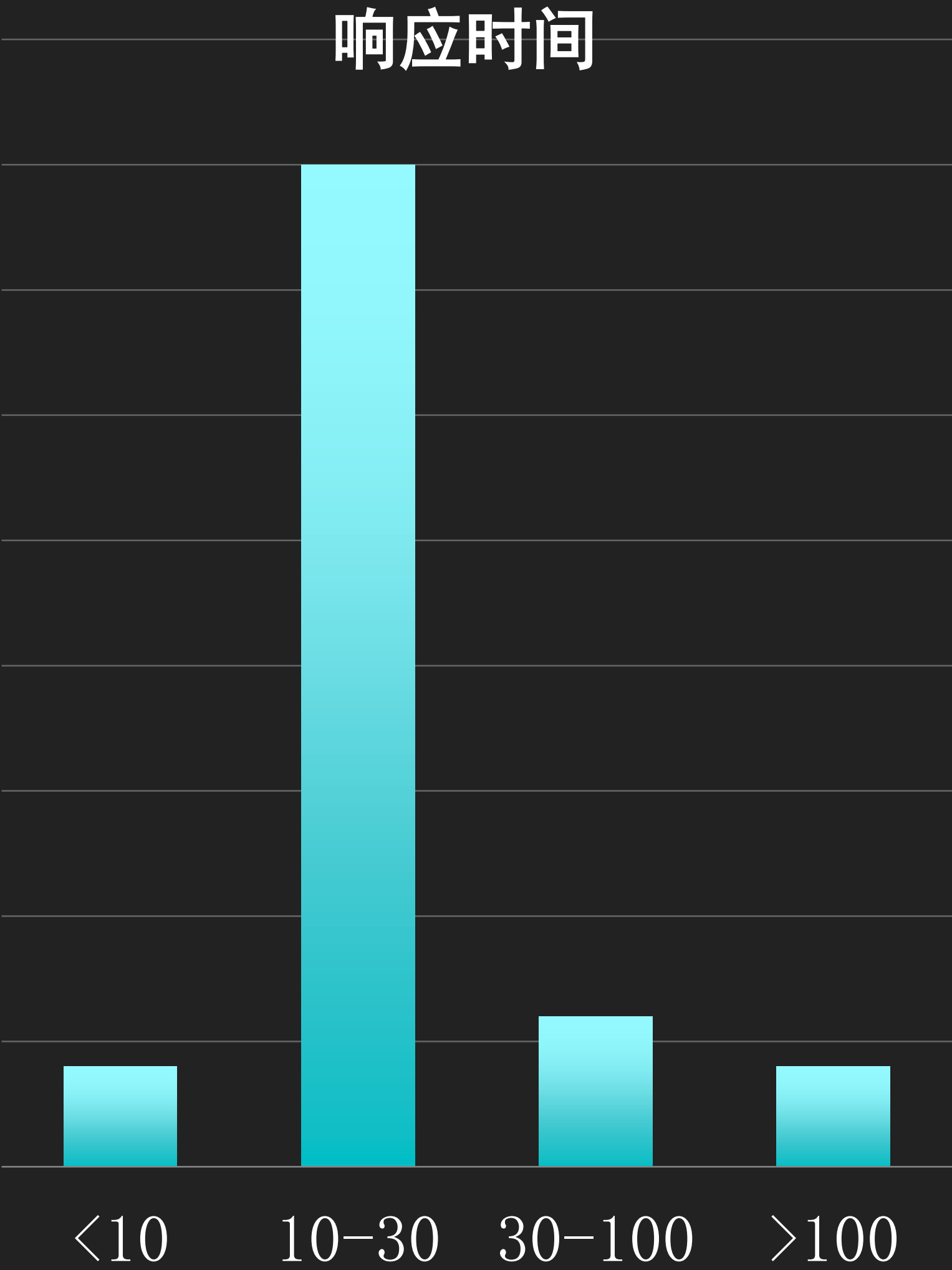
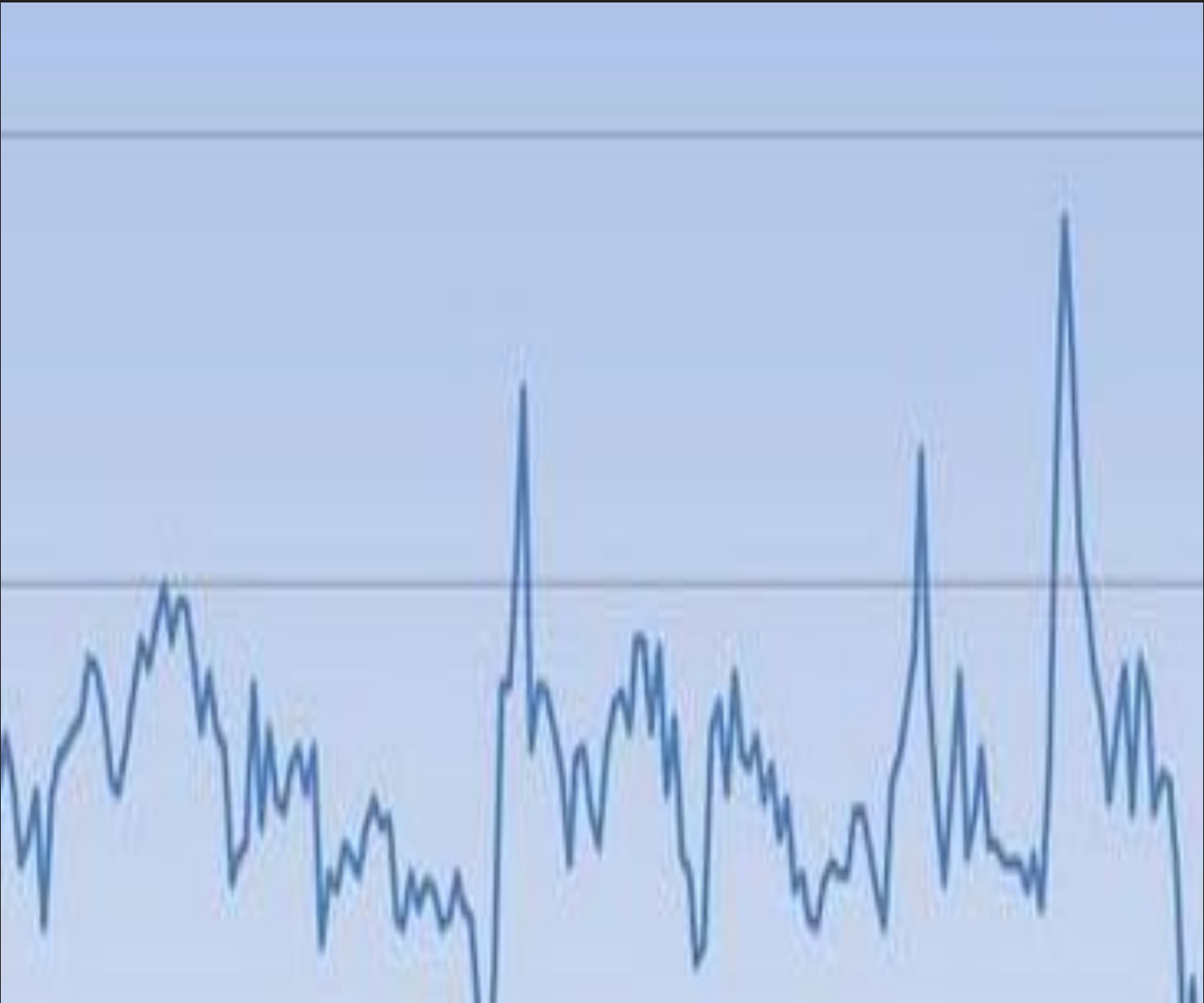


决策精准化

——模糊控制策略

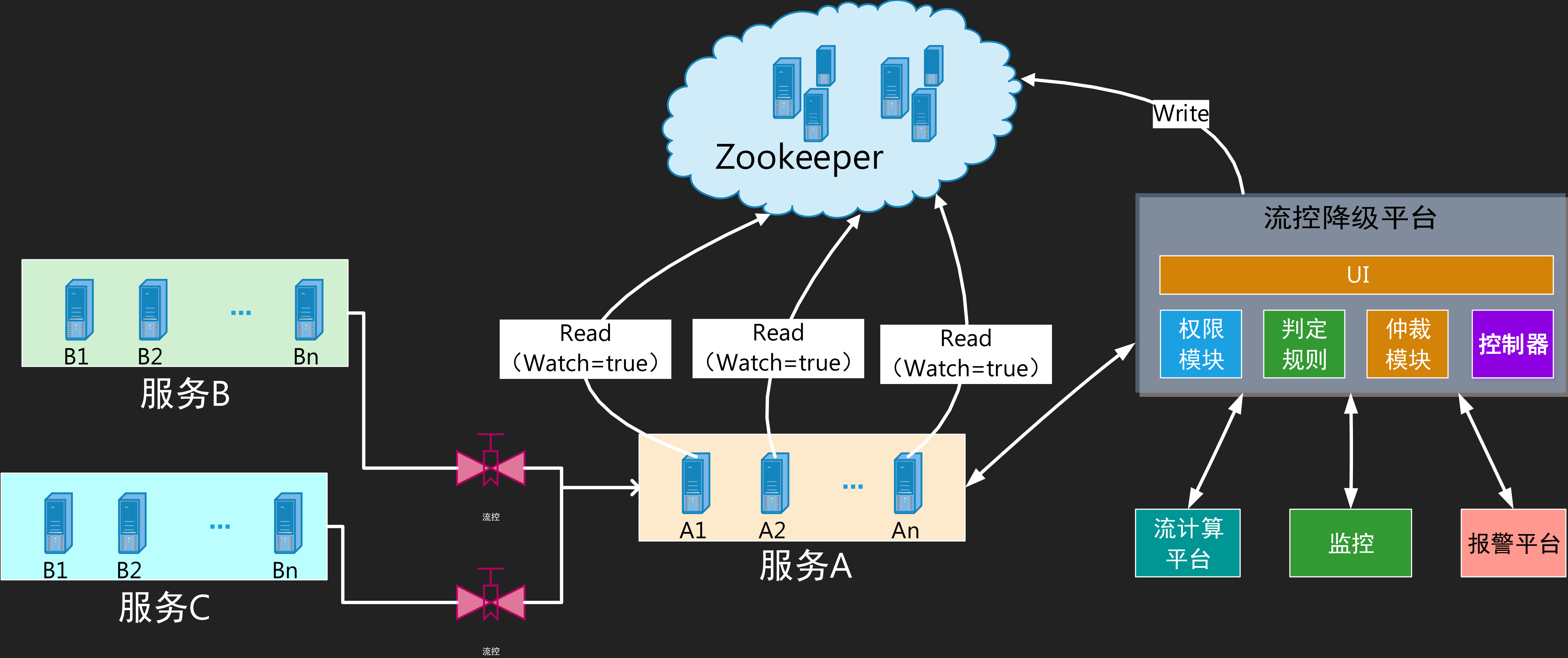
4个黄金指标：

- 延迟
- 流量/并发数
- 错误
- 饱和度



高度关注毛刺分布，定期巡检

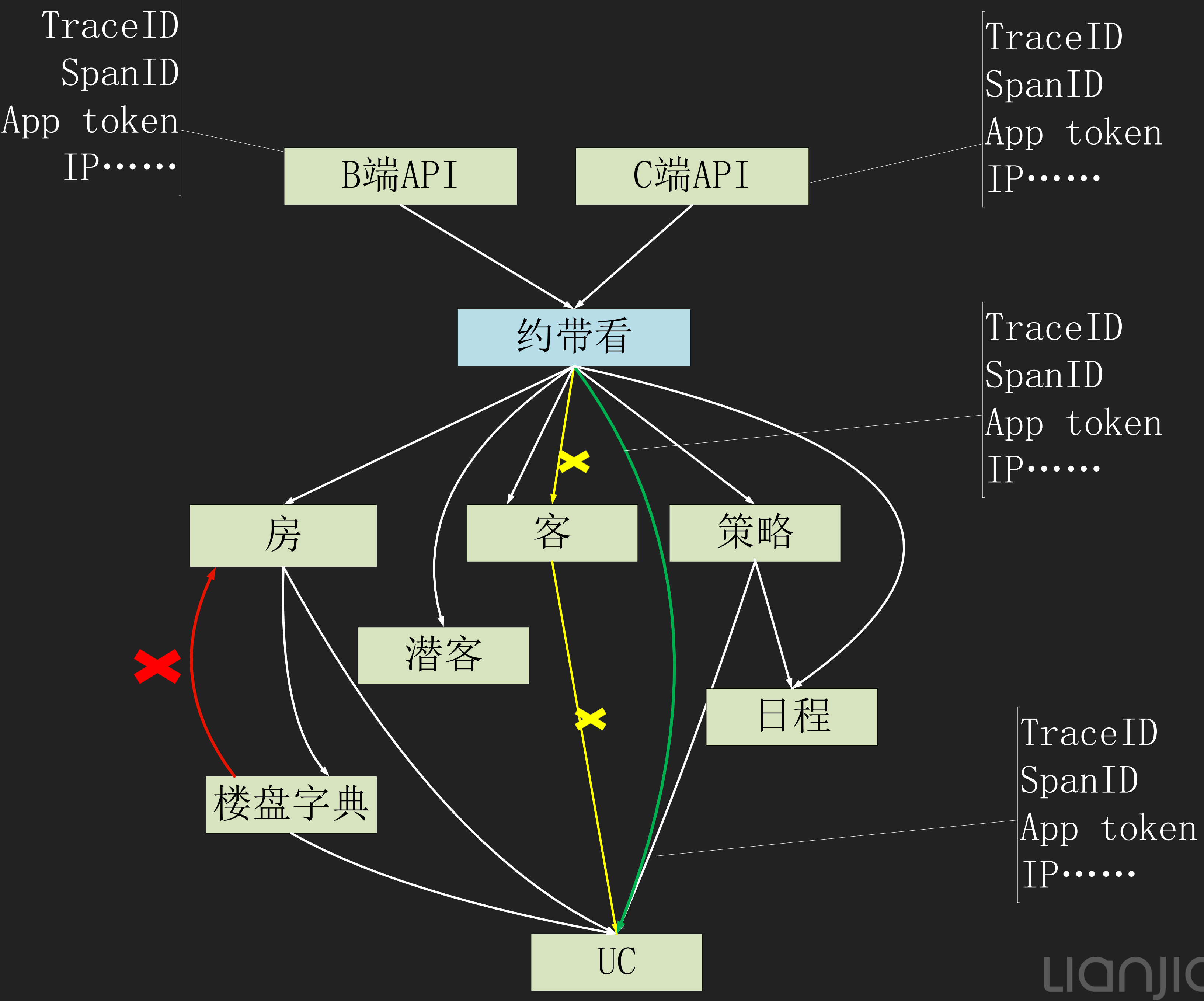
细粒度流控
&迅速降级



链路跟踪平台

——跨团队之利器

- 快速排查问题
- 优化路径



OP ? RD ? QA ?

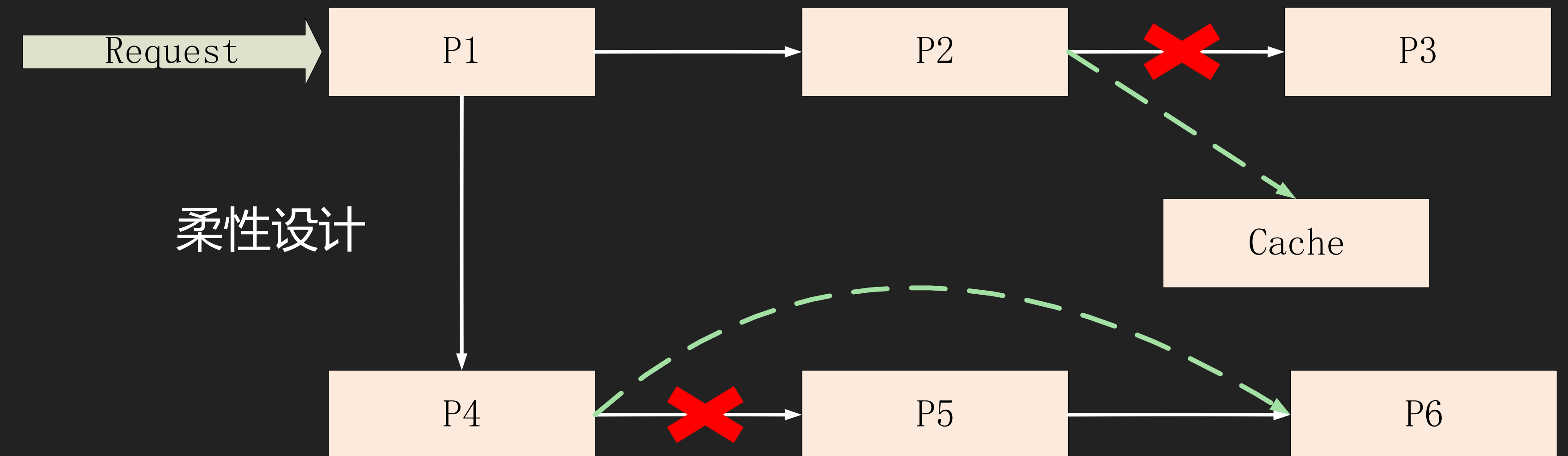
——运维融入设计，人人都是SRE (SITE RELIABILITY ENGINEER)

(SMART - Relevant)

好代码是高可用的基石

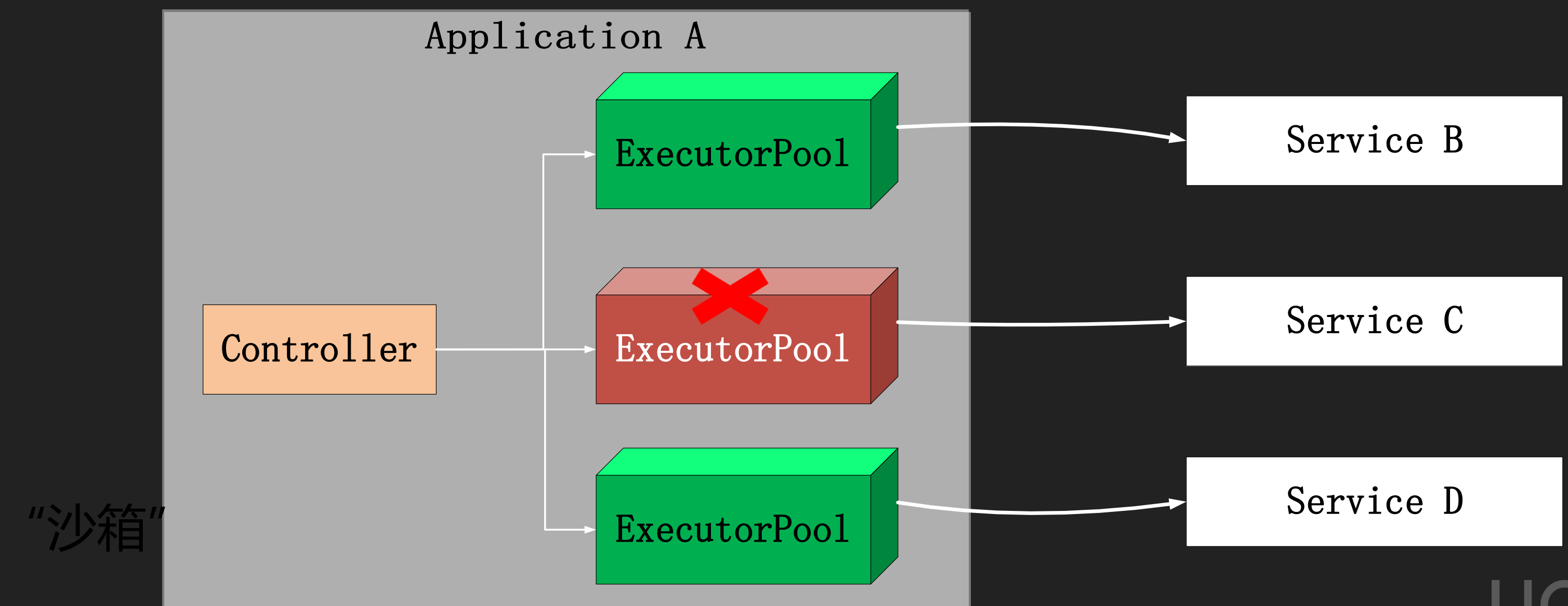
容错能力：

- 柔性接口设计，减少串联
- 对下游可控制
- 对上游轻依赖，CACHE关键数据
- 幂等/数据自修复能力



隔离能力：

- 营造“沙箱”环境，线程资源独立
- 事务中避免调用第三方



01.
会话层

02.
服务层

03.
数据层

04.
基础设施体系建设

遇到的问题

- 大量单表过亿
 - 找不到路由ID
 - 时间成本
- (SMART – Time-based)



技术选型

	ShardingDAO	COBAR	MYCAT
读写分离	Yes	No	Yes
主库绑定	Yes	No	No->Yes
跨库查询	Yes	No	Yes
跨库事务	No->Yes	No	No
独立部署	No	Yes	Yes
辅助索引	Yes	No	No

阶段一

Sharding DAO

路由

Diverter

DB Router

执行

Index
Query

DB single
Query

DB concurrent
Query

Merger

基础组件

Aspect
Manager

Connection
Manager

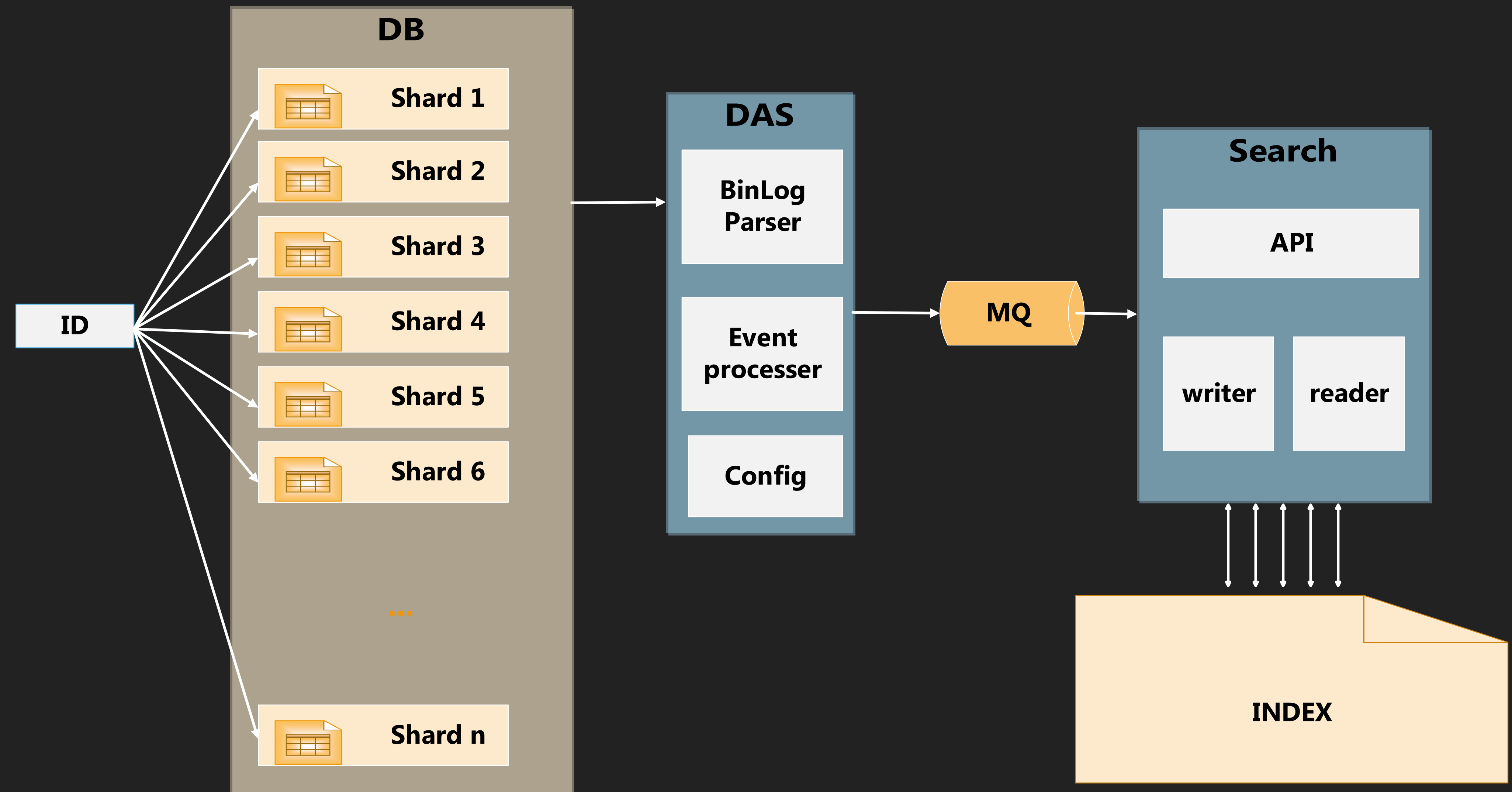
Transaction
Manager

Exception
Manager

Bypass
Manager

辅助索引

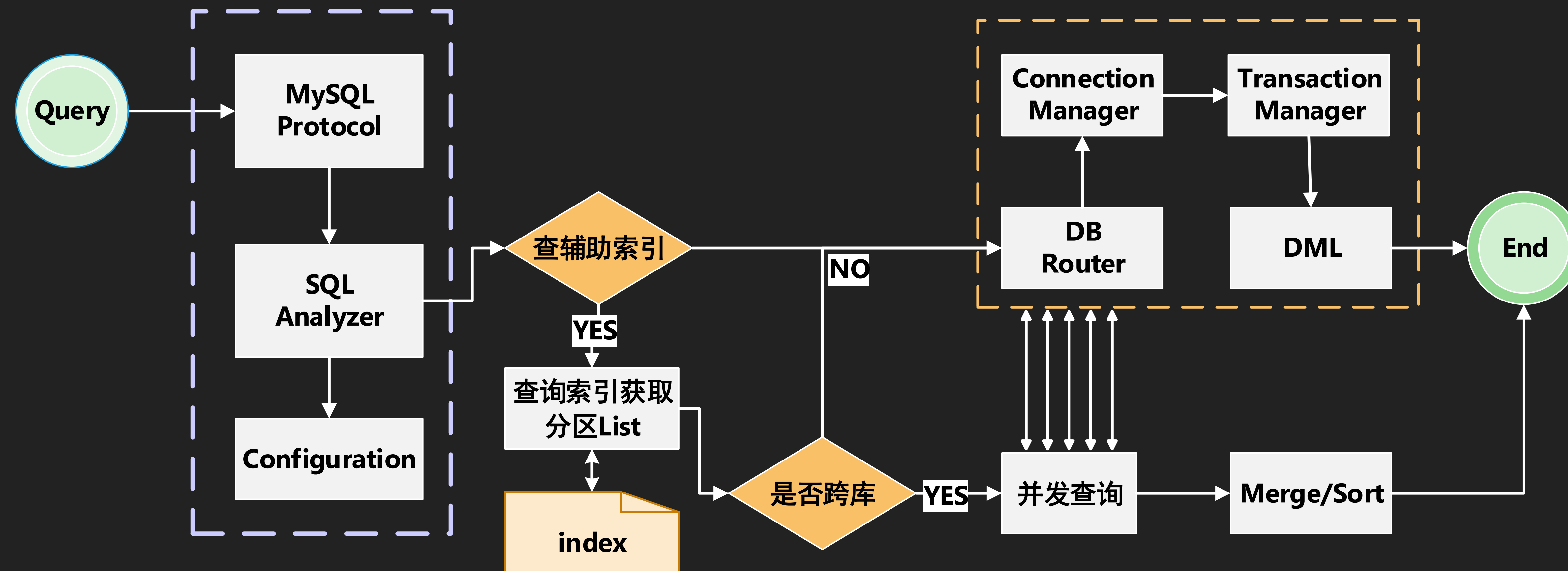
- 异步化、高性能
- 支持复杂条件
- 成本复用



阶段二

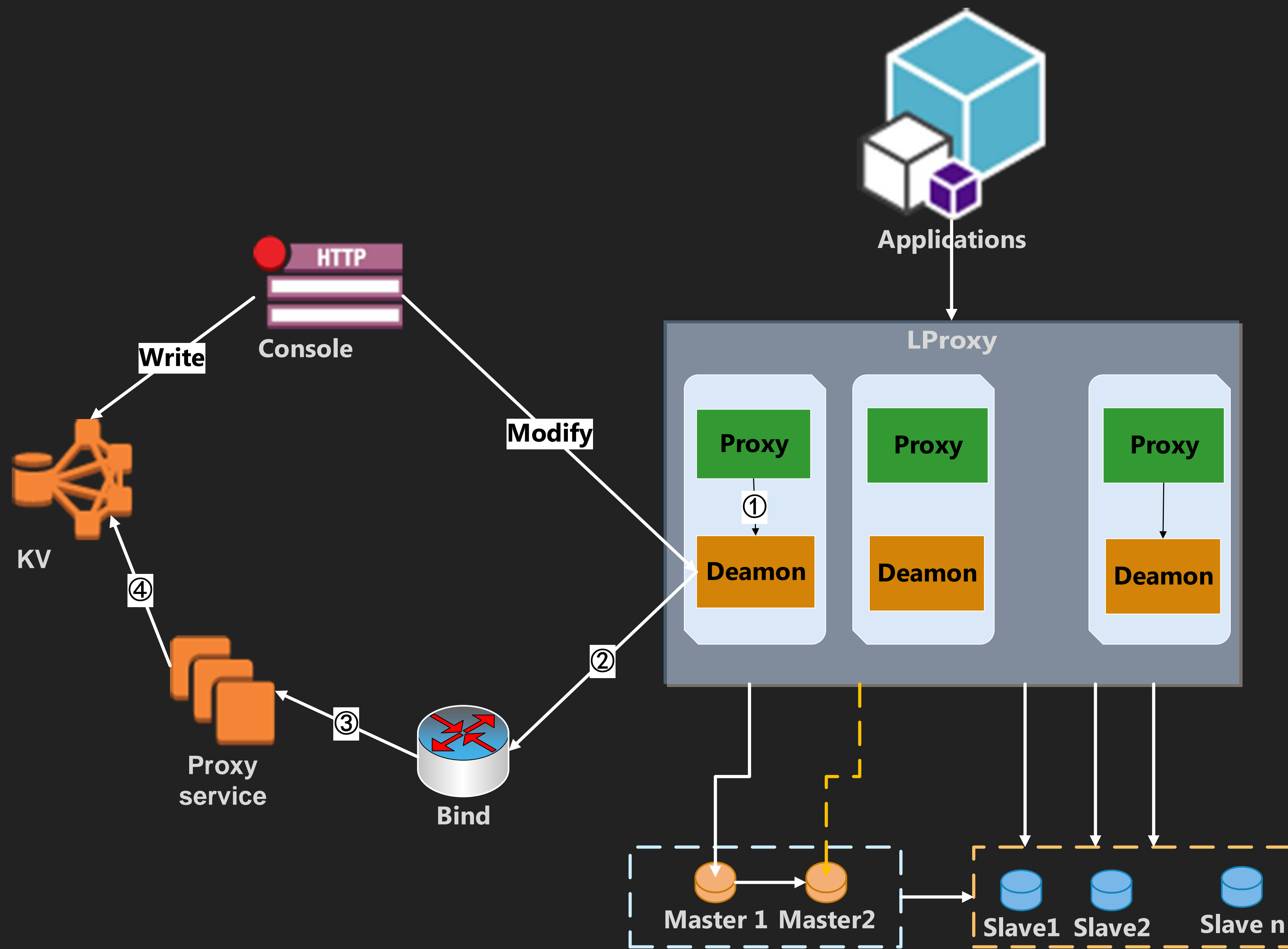
L-Proxy

- 成本复用
- 对程序透明



阶段三： Naming service

逃不出的法则——避免单点



01.
会话层

02.
服务层

03.
数据层

04.
基础设施体系建设

基础设施体系建设

运维服务平台

IDC管理

云主机管理

交换机管理

LVS管理

F5管理

DNS管理

监控治理平台

白盒/黑盒监控

流控平台

链路跟踪平台

降级管理平台

报警平台

日志平台

研发服务平台

命名服务

高可用队列/高性能消息队列

高可用分布式缓存

文件云存储

Hadoop、Hbase、Hive集群

Strom集群

持续交付平台

持续构建

持续测试

全链路压测

灰度发布

持续发布

一键扩容

流程质量管理

问题管理

事件工单管理

事故管理

变更管理

SLA协议管理

权限管理

提纲

- 一、架构之痛
- 二、高可用架构演进与实践
- 三、总结

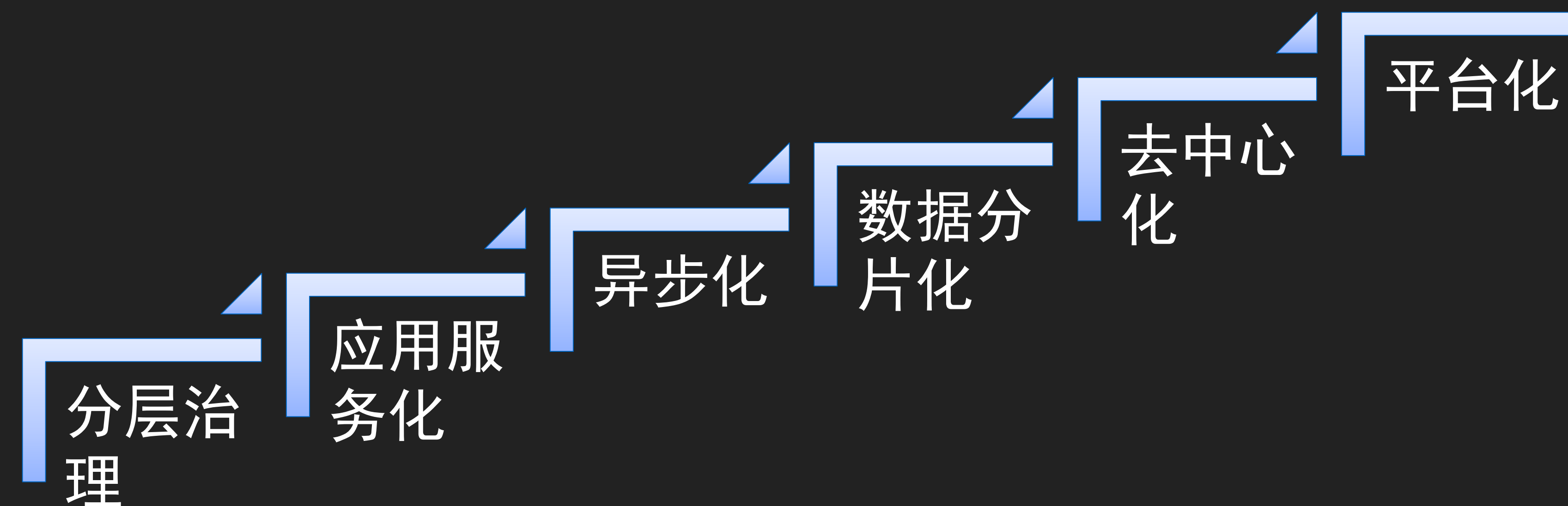


总结

- D-I-D原则：DESIGN 20；IMPLEMENT 3；DEPLOY 1.5
- 运维融入设计，人人都是SRE，积极防御，可监控，可治理，可转移
- 柔性设计，轻依赖，可扩展
- 遵守法则，没有捷径

演进之路

- 可用性：99% -> 99.99%
- 性能提升：12倍+
- 迭代速度：10倍+
- 低成本可复用





Q&A



THANK YOU



LIANJIA.COM