Geekbang. 极客邦科技 InfoQ

QCon 全球软件开发大会【北京站】2016

饿了么异构服务平台数据访问层的演进

饿了么 徐东焱

Intemational Software Development Conference

谁讲？

- 徐东焱
- 饿了么 数据访问层架构师
- 曾在爱立信、PayPal任开发工程师
- 专注于C++/Java 中间件开发

dongyan.xu@ele.me dongyanxu@live.cn

谁听？

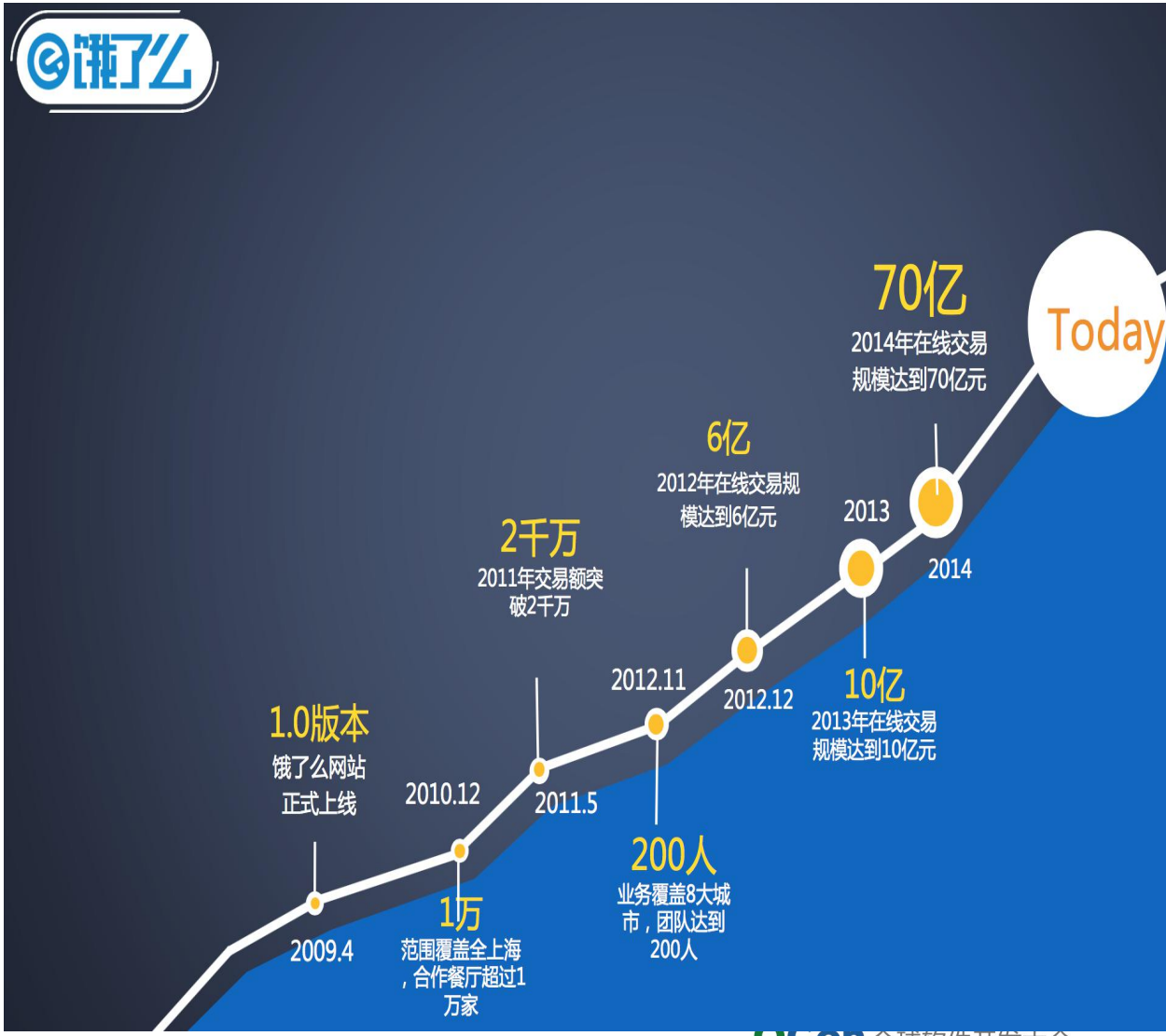
- 不是
  - 新技术推介
  - 产品功能介绍
- 是
  - 基于Java Netty.io 高性能服务的实践和踩过的坑
  - 创业公司里繁杂需求中保持KISS原则的实际案例

业务背景

- 纯 Python 平台
- 18000服务实列
- 330万订单/日

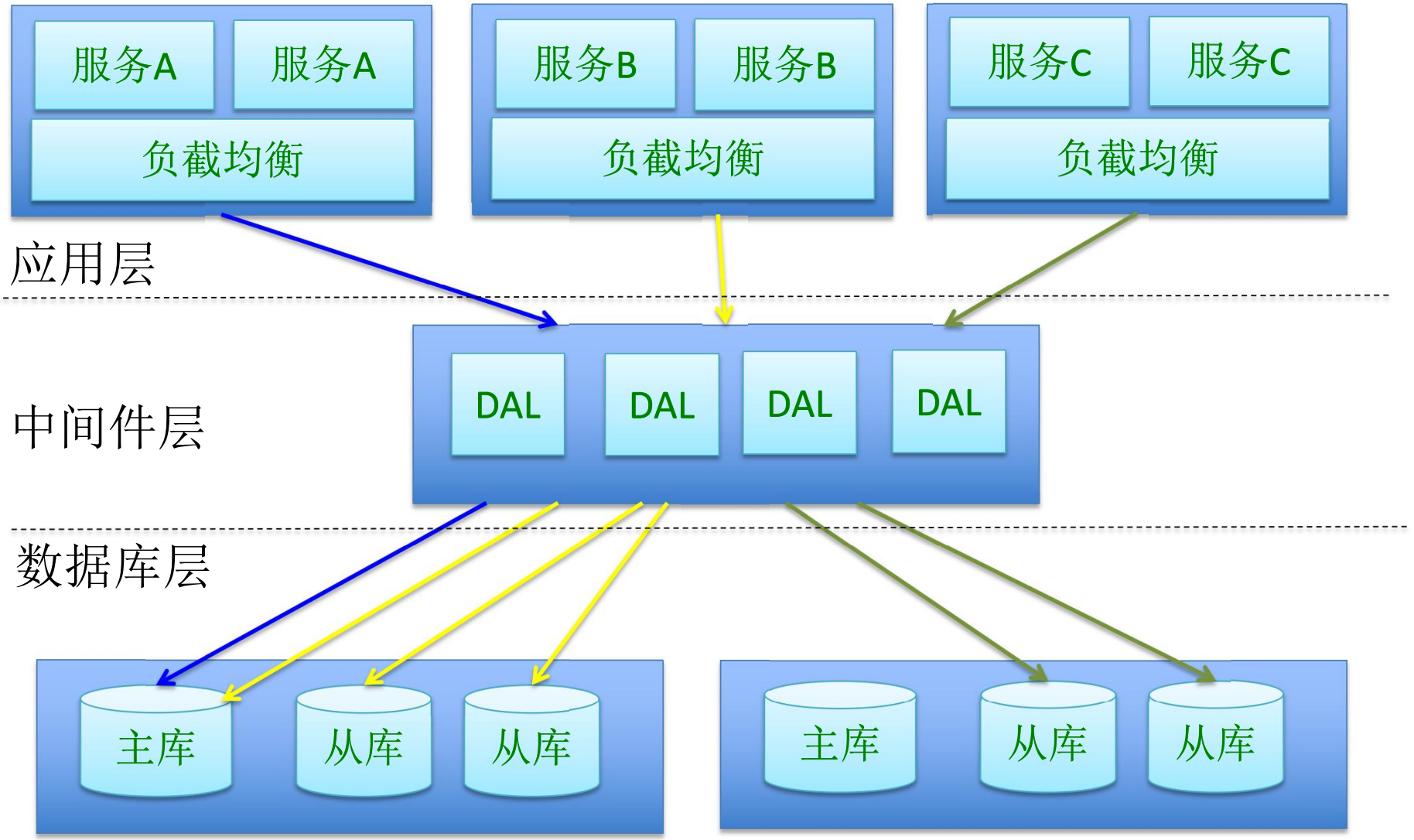
- 产研团队急剧扩张
- 引入java生态圈

- 标准化服务框架
- 引入数据访问层

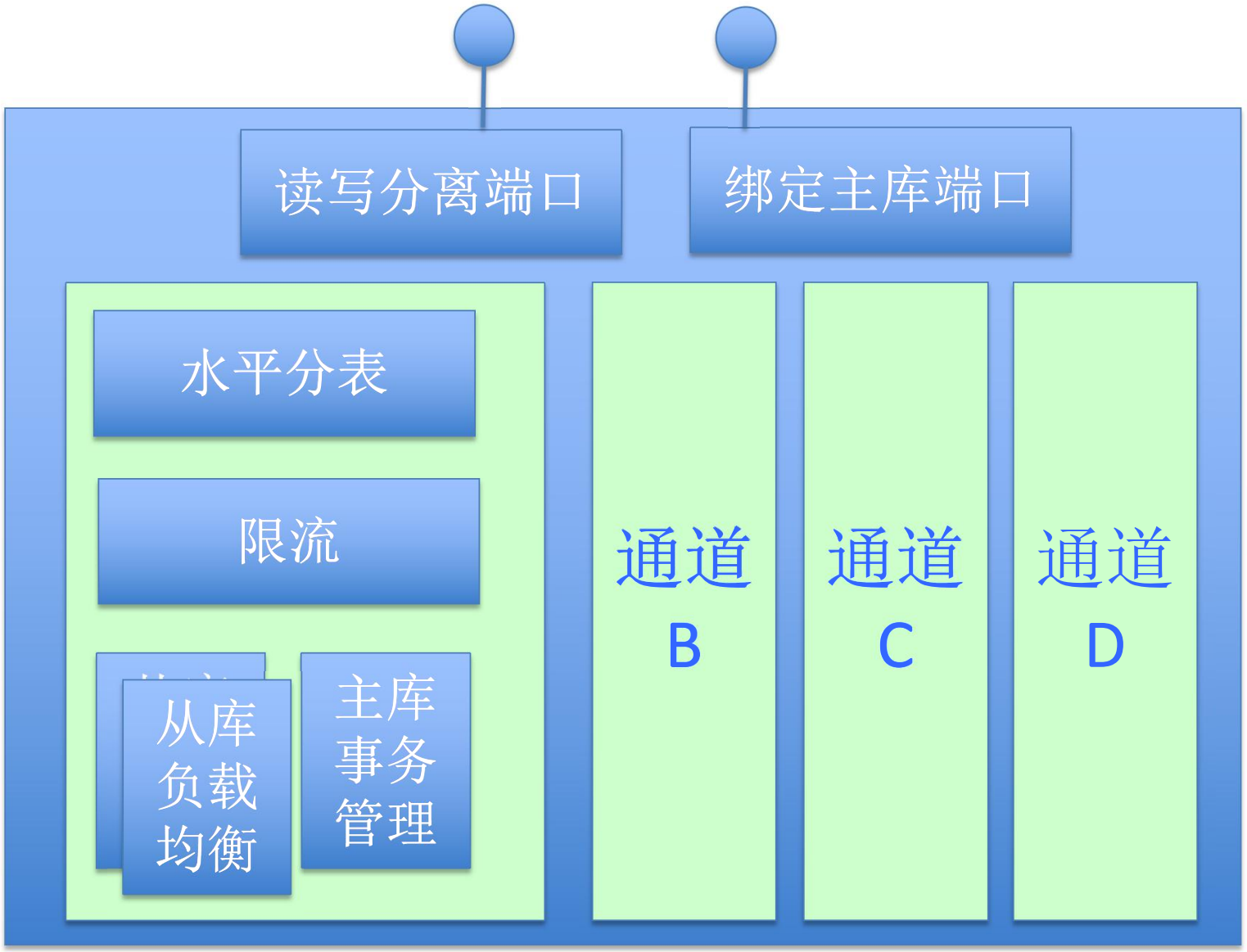


DAL特性

数据房问层(DAL)


服务A
服务A
负截均衡
服务B
服务B
负截均衡
服务C
服务C
负截均衡
应用层
DAL
DAL
DAL
DAL
中间件层
数据库层
主库
从库
从库
主库
从库
从库

DAL 架构概览



DAL主要特性

- 限流、削峰
- 熔断
- 读写分离
- 多客户隔离
- DB连接复用
- 一维分表、二维分表

DAL主要性能指标

- 单机1万 QPS（极限4.8万）
- DB连接数从大于1.5万下降到小于1千
- 单条SQL经过DAL平均多耗时0.6ms（和直连DB对比）
- 控制DB实际并发度<60，有效阻挡异常流量冲击

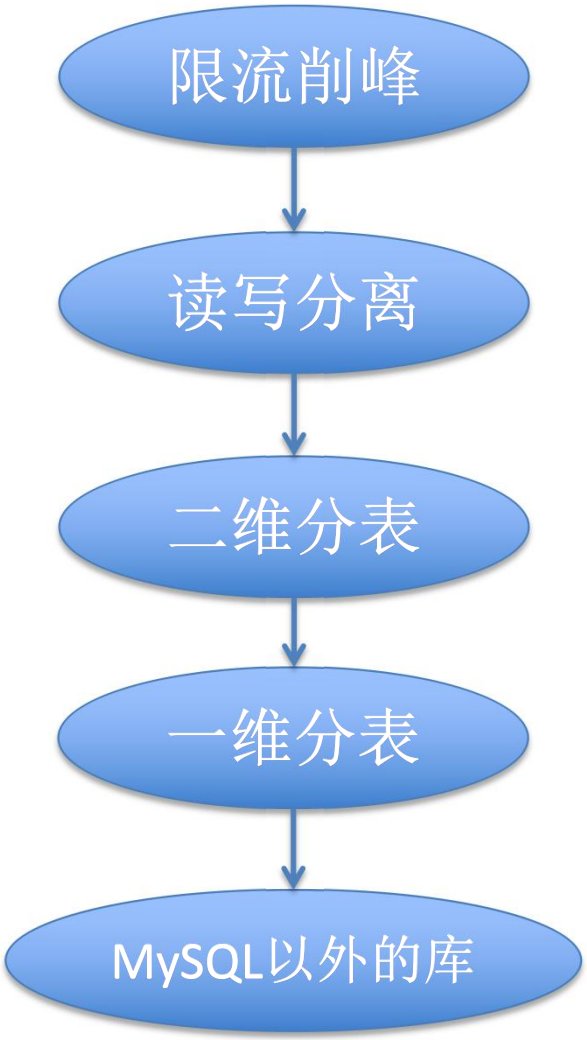
DAL功能
基于用户痛点的演进

DAL功能演进路线图

DBA：阻挡异常流量的冲击

DBA：主库流量太大，要把查询分散到多个从库

DBA：订单表太大，必须切片



开发：不会有那么严重吧？

开发：没时间改造，最好接入DAL就自动处理了

开发：基于用户和基于商户的查询频度是同等的

皮之不存毛将焉附
演进之前的架构选型

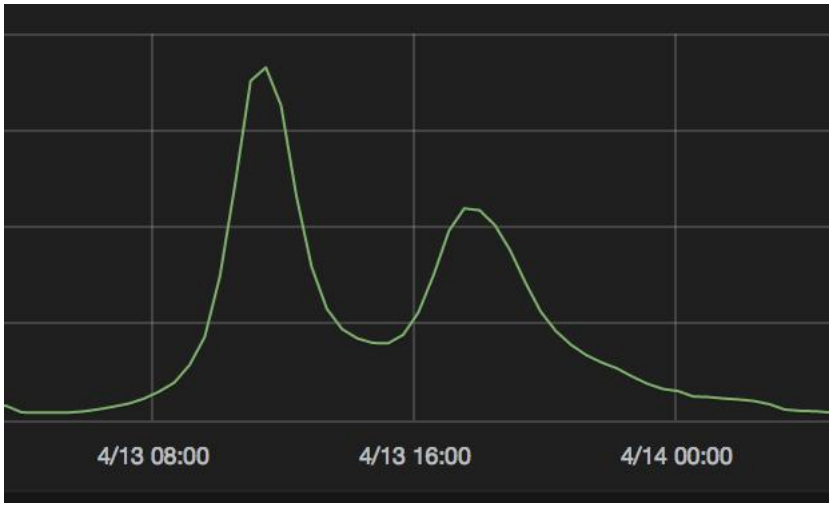
DAL架构选型(康威定律)

- 利益相关者
  - DBA : 异常流量对数据库冲击大，@#￥&.....
  - 应用方：新特性任务急，少给我们加条条框框
  - 管理层：稳定，稳定，稳定，重要的事情说三遍
- 解决关键的用户痛点

DAL架构选型(技术要素)

- 百万级日订单量(2015/4)
- 主要服务类型: python
- 新增服务类型: java、node.js、go
- 主要业务数据存于MySQL

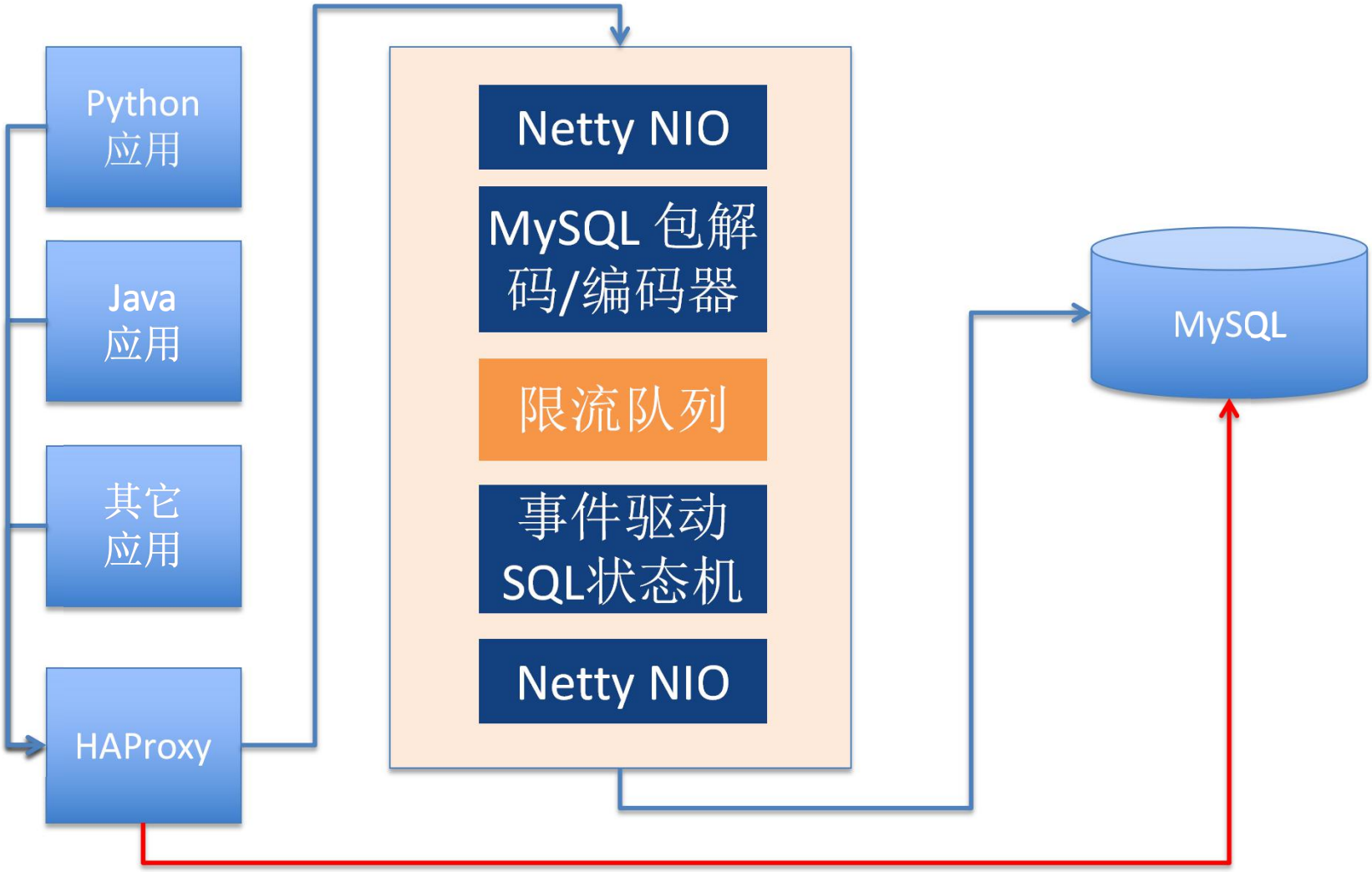
- 流量峰值显著(午高峰/晚高峰)



DAL架构选型(目标规纳)

- 独立进程服务，支持多语言平台
- 从简单功能开始，DAL团队完全掌握实现细节
- 业务可回滚到DAL接入前的状态

DAL架构选型(产出)

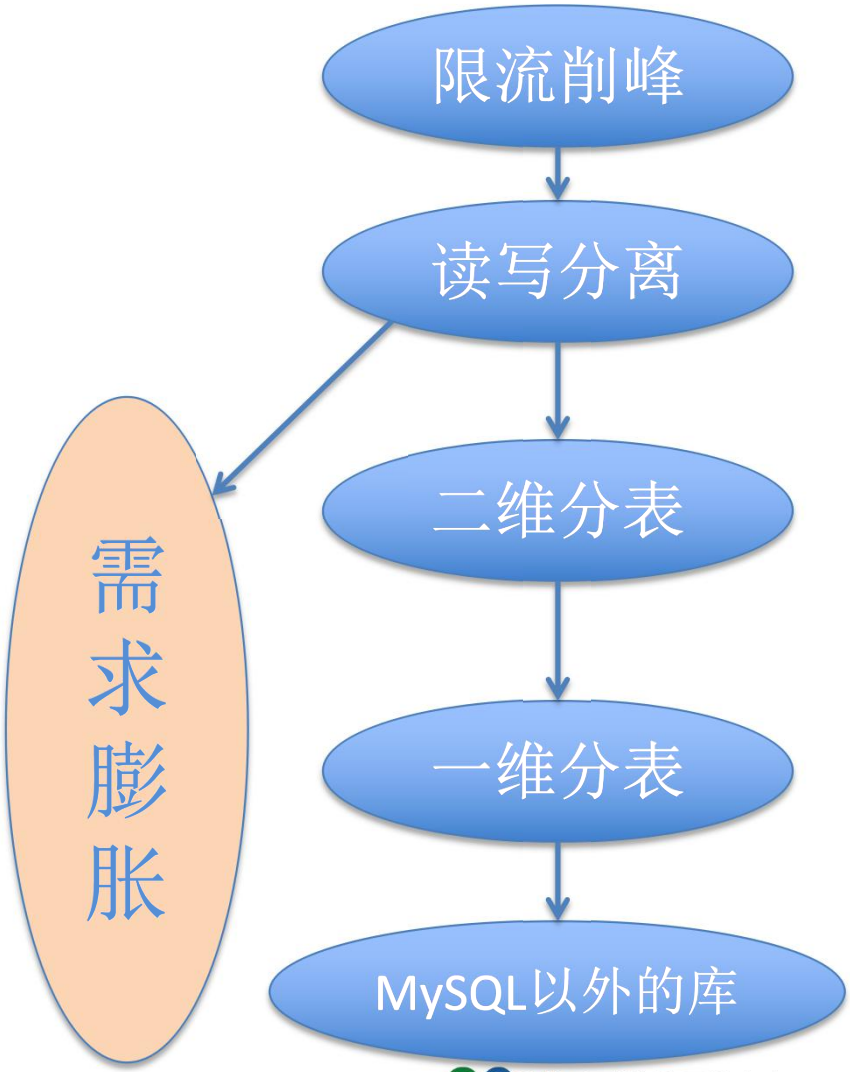

Python
应用
Java
应用
其它
应用
HAProxy
Netty NIO
MySQL 包解码/编码器
限流队列
事件驱动 SQL状态机
Netty NIO
MySQL

需求膨胀
不忘初心，DAL只是SQL代理

保持核心模块的纯净(需求膨胀)

- SQL黑名单
- SQL白名单
- 熔断
- 基于QPS的流量控制
- 请求链追踪

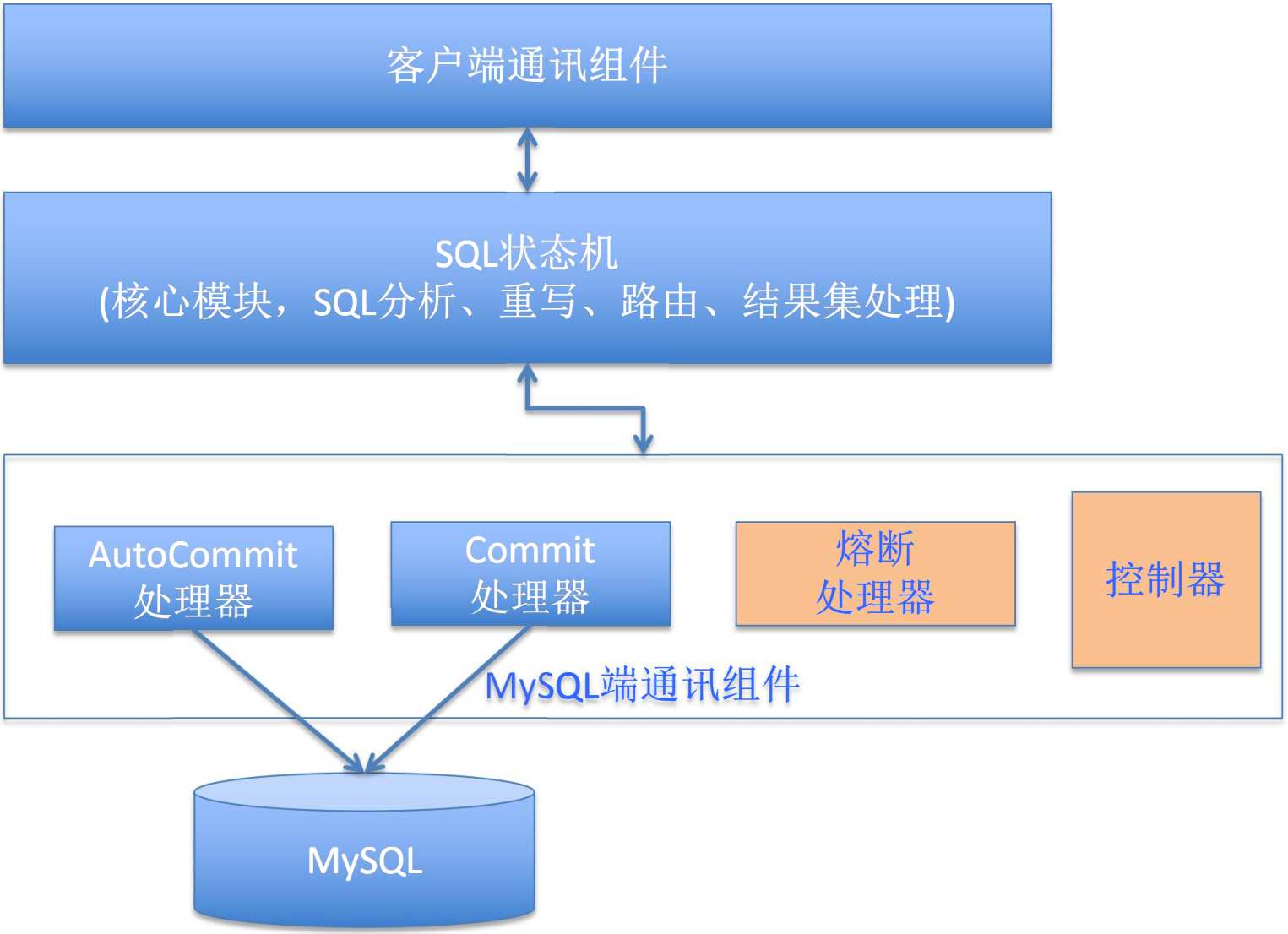
此去省略N行……



保持核心模块的纯净(解耦)

- 其实DAL是一个SQL代理
- 坚决杜绝在核心模块中直接处理各项杂务
- 在独立的模块中实现各项功能，通过简单的接口和核心模块整合

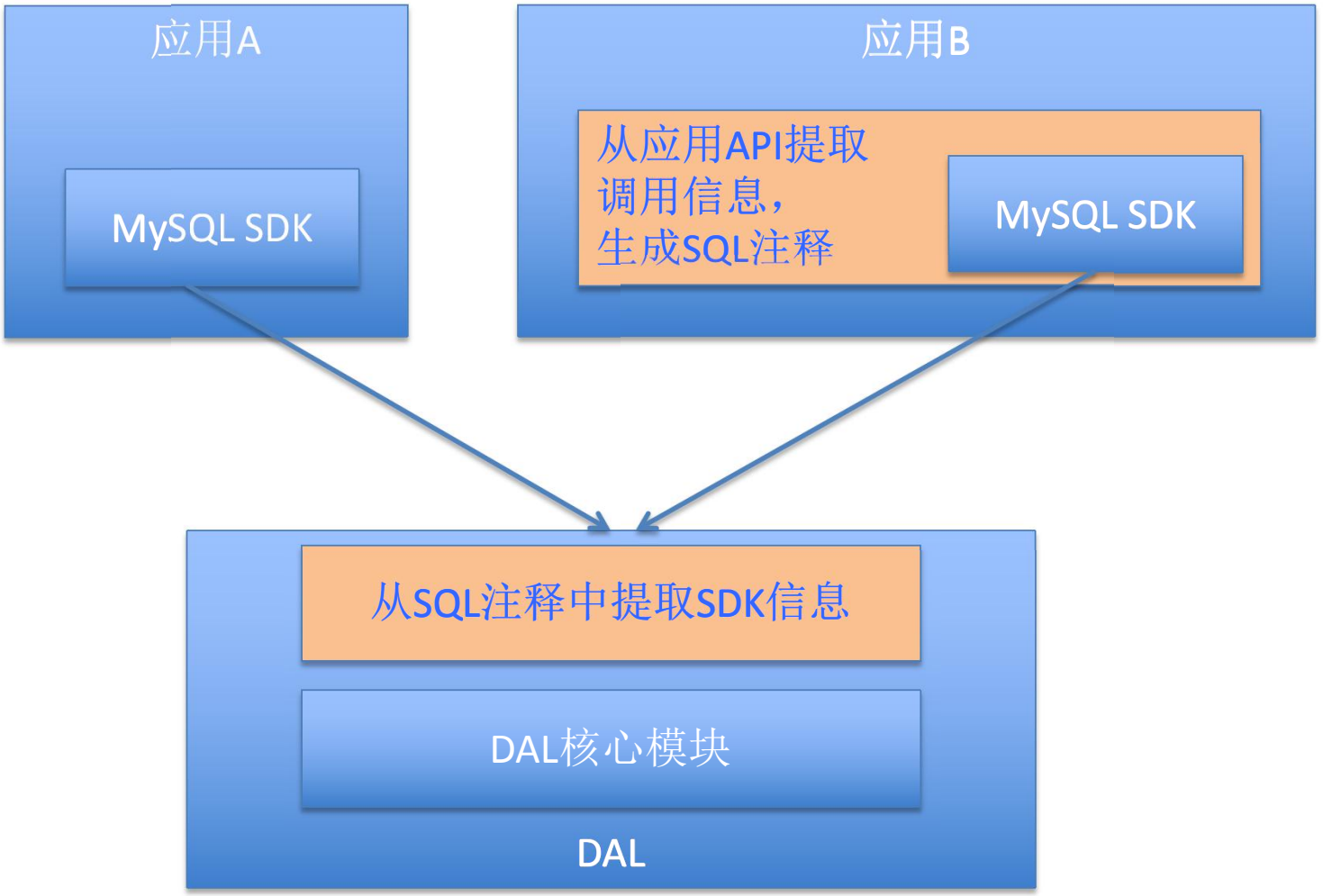
保持核心模块的纯净（熔断解耦）



保持核心模块的纯净(调用链追踪痛点)

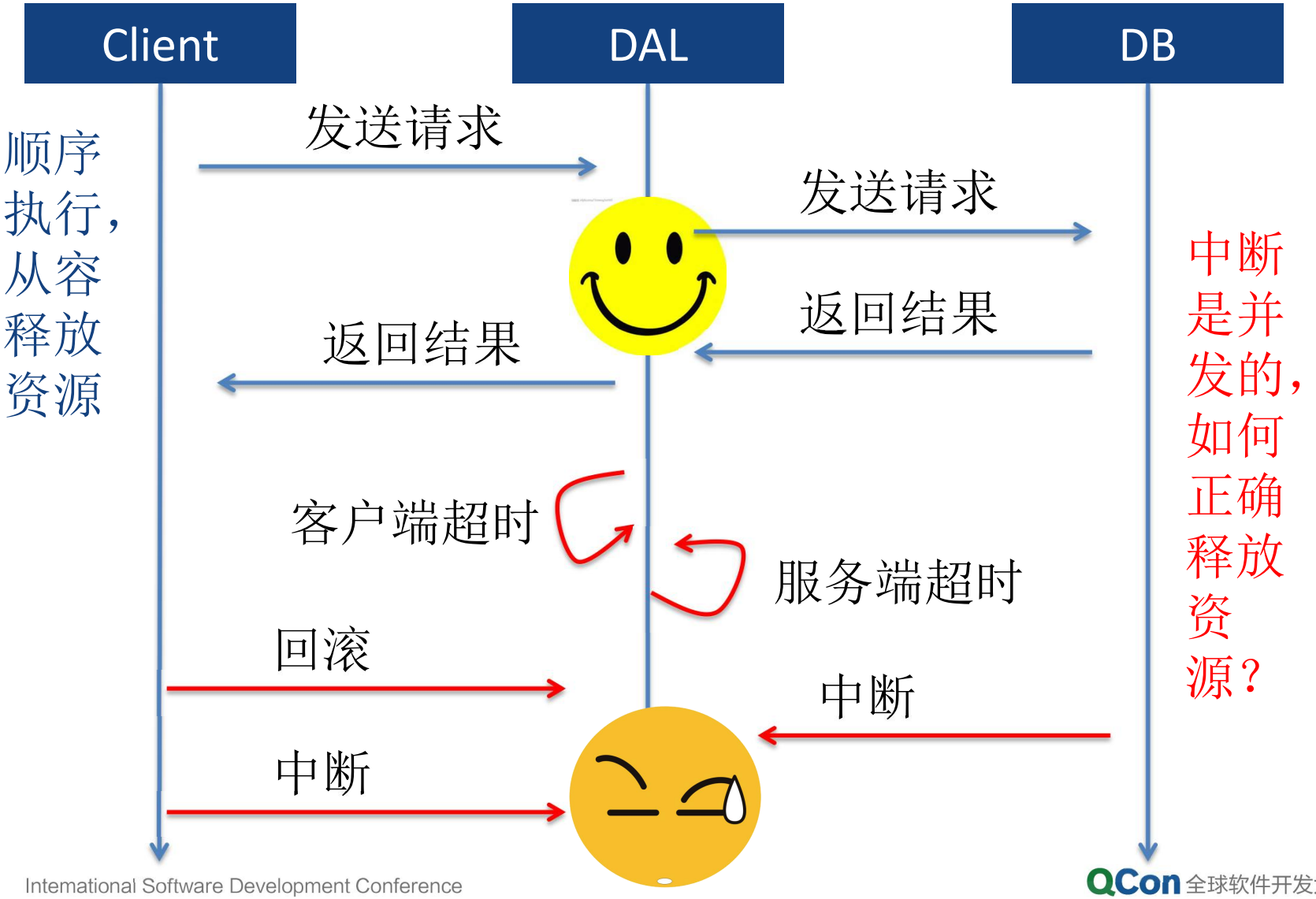
- 很难追踪异常SQL从那里来
- 当多个SQL被锁阻塞时，不易发现最开始持有锁的事务执行的SQL

保持核心模块的纯净(调用链追踪解耦)



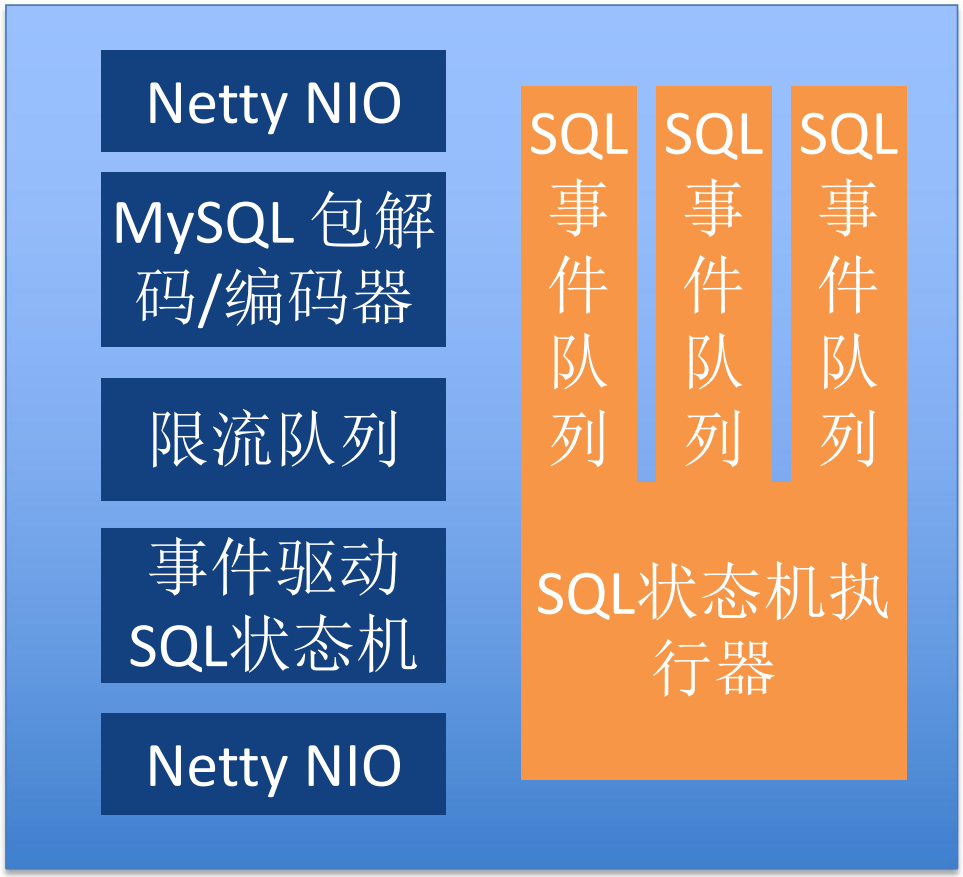
要并行，不要并发

走出并发的沥清坑1(加锁.加锁!加锁？)



走出并发的沥清坑2（去掉并发）

- Netty 线程只负责传送字节流
- 所有SQL状态都在SQL状态机执行器中执行
- 一个SQL会话上的任务路由到相同SQL执行器线程，排队依次执行，从容检查当前状态



要容灾切换，
就把切换变成常规操作

DAL成了最大的单点(问题)

- 多节点集群可靠吗？
- 线上交换机故障
- 备用交换机故障

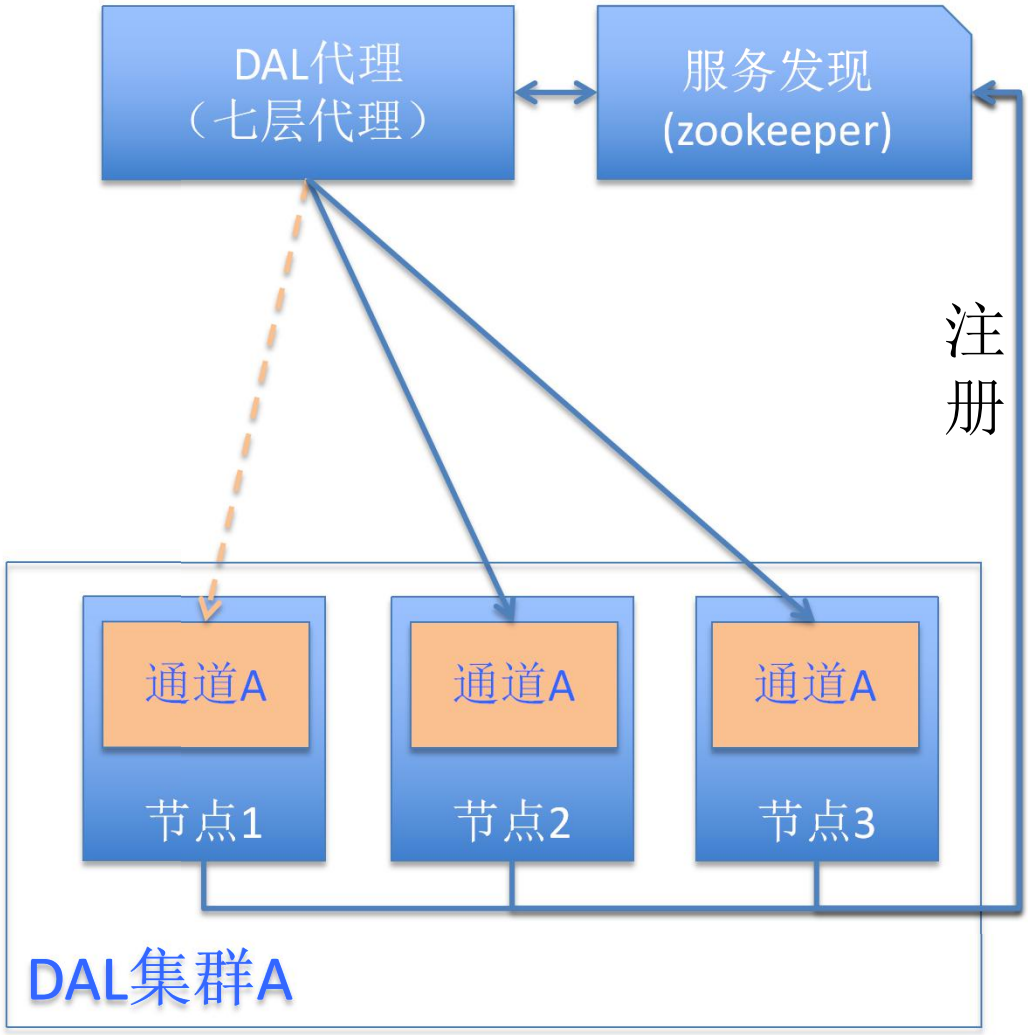
DAL成了最大的单点(方案)

- 单节点拉入/拉出集群
- 不间断升级
- 集群迁移
- 以上三者其实是一个功能

DAL成了最大的单点（不间断升级）

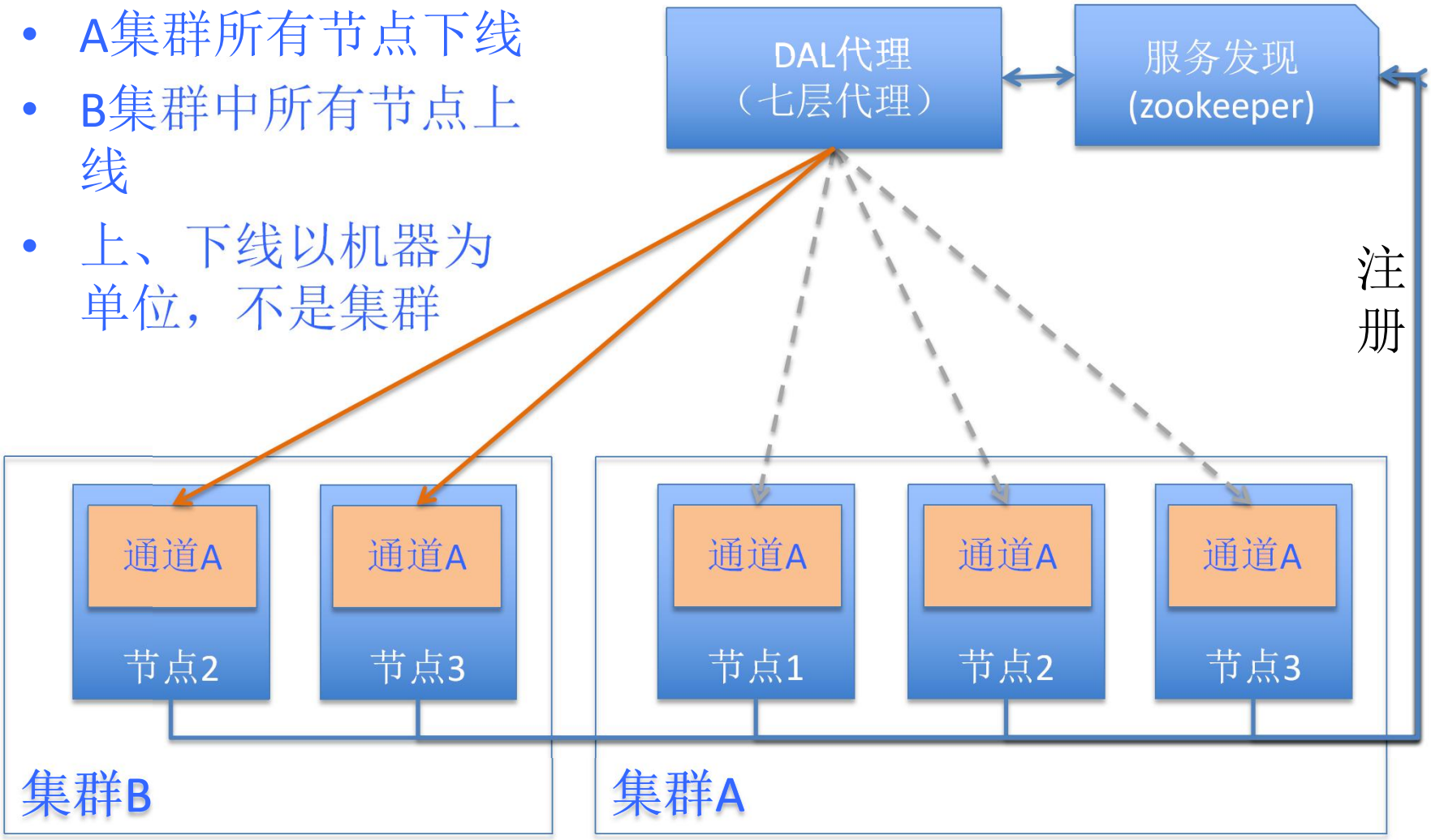
- 节点1下线指令
  1. 代理转发新连接到其它节点
  2. 节点1等待当前SQL完成
  3. 节点1等待当前事务完成
  4. 代理将现有连接迁移到其它节点

- 节点1 升级重启
- 节点1 上线



DAL成了最大的单点（容灾迁移）

- A集群所有节点下线
- B集群中所有节点上线
- 上、下线以机器为单位，不是集群



总结

- 方案：以客户痛点为中心
- 实现：坚守KISS原则
- 容灾切换：随时切换

Geekbang 极客邦科技 InfoQ

THANKS!