

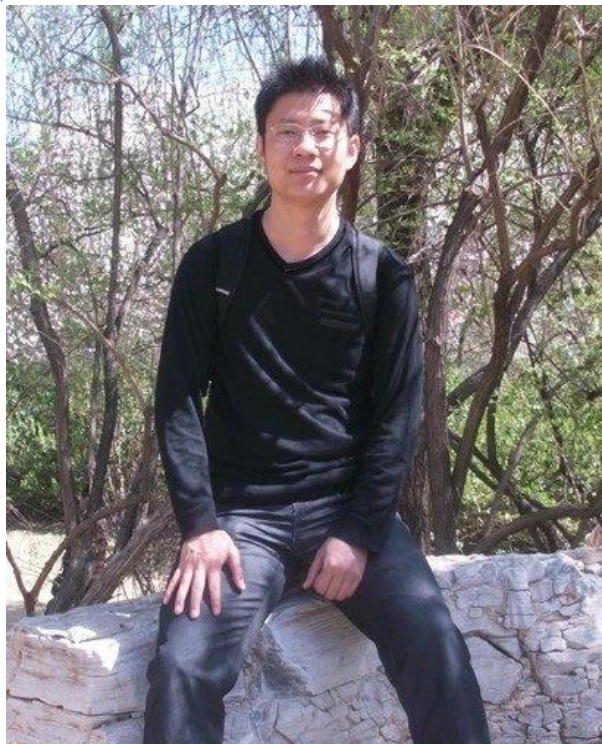


Gdevops

全球敏捷运维峰会

58速运订单调度系统架构演进

演讲人：胡显波



胡显波 58速运 后端架构负责人

8年的架构和团队管理经验，涉及CRM、互联网、同城物流等领域。

2014年初至今一直任职58到家，期间负责过58家政、58速运等业务。

在这3年当中见证了58到家一步步成长成为行业巨头，也见证了技术部从30人至500+人，从跟不上业务发展到驱动业务发展的过程。

目录

- 创业之初
- 高速发展
- 智能驱动
- 总结



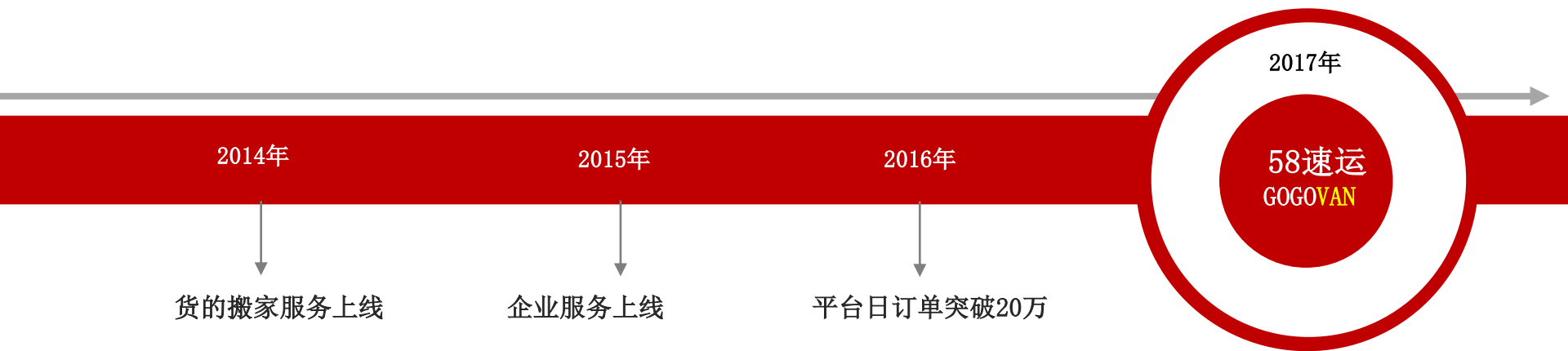
搬家

货运

58速运

覆盖中国及东南亚地区的同城货运平台

100个驻点城市，100万注册司机，累计服务货运用户过亿次





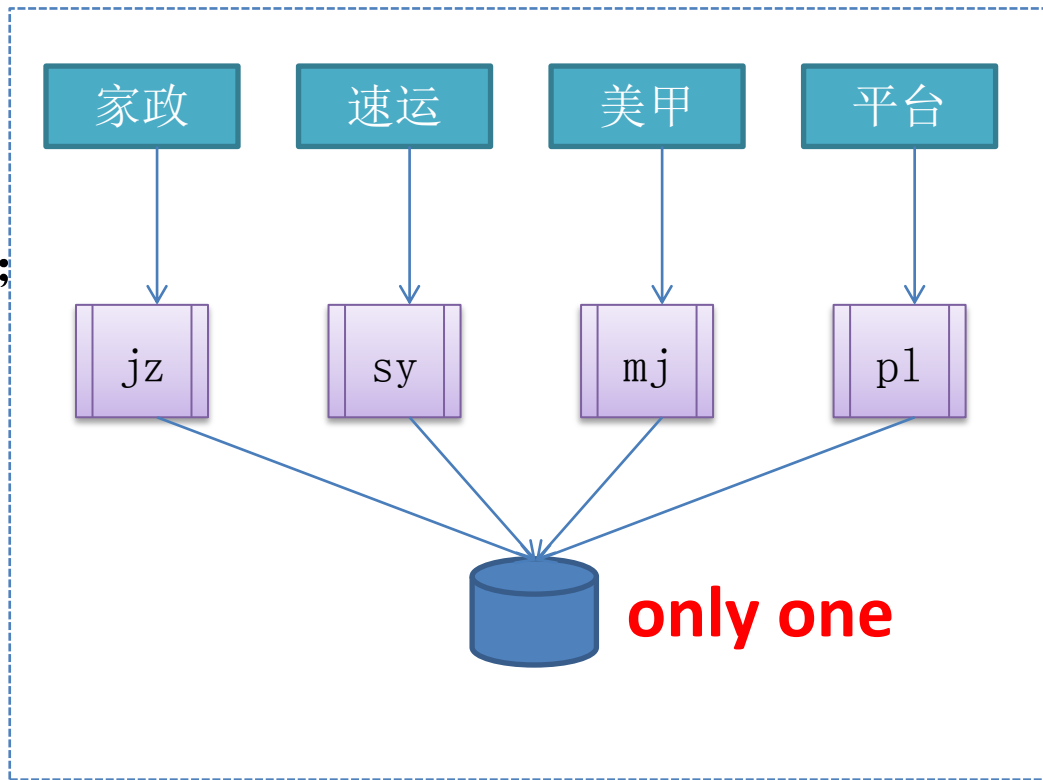
创业之初

-快速迭代试错

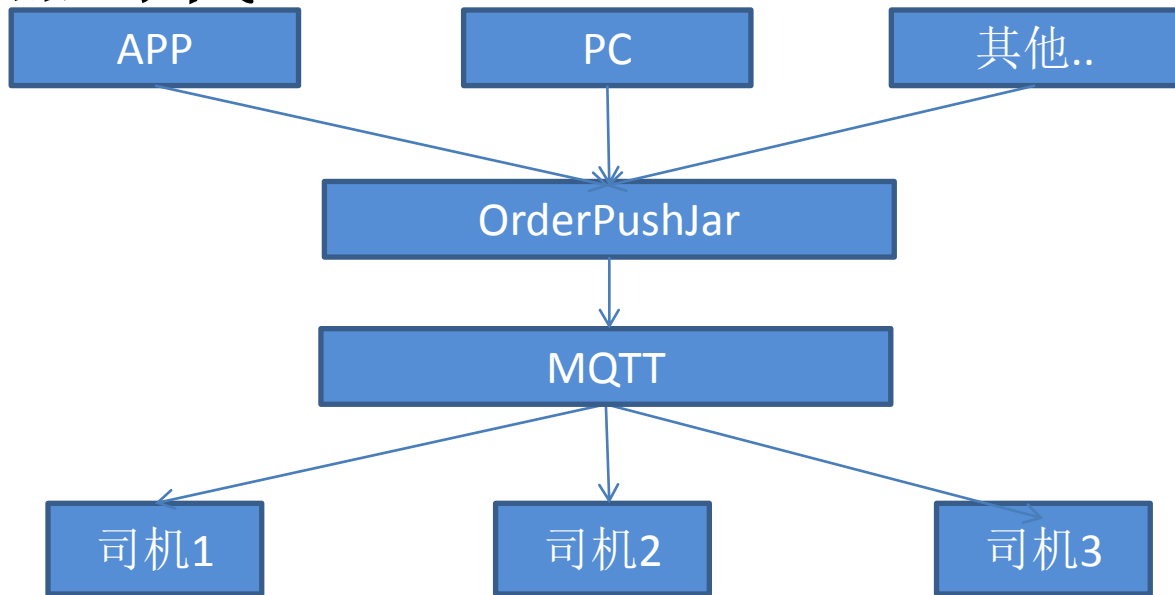


创业阶段:

- 快速孵化多个业务(20+);
- 节奏: 3周内上线, 含用户侧APP、商家APP、管理后台

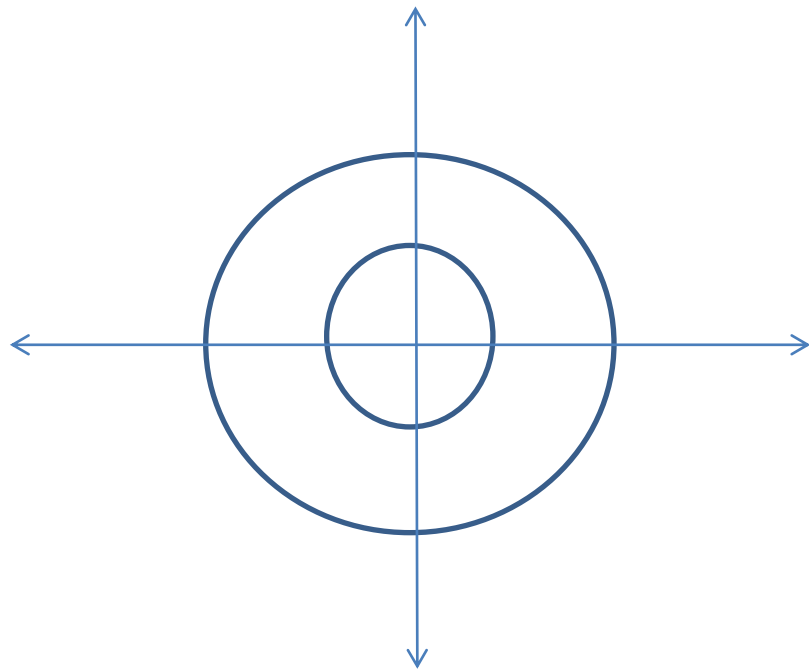


派单-石器时代



订单调度（初级）：

- 简单的距离推送
- 抢单即中单
- 每单都有补贴



痛点：

- 系统不稳定，一个慢SQL，全业务受影响。
- 多业务并存，订单表索引多，性能下降。
- 订单字段冗余，新增和修改字段非常痛苦。
- 业务增长迅猛，数据库已成为瓶颈。



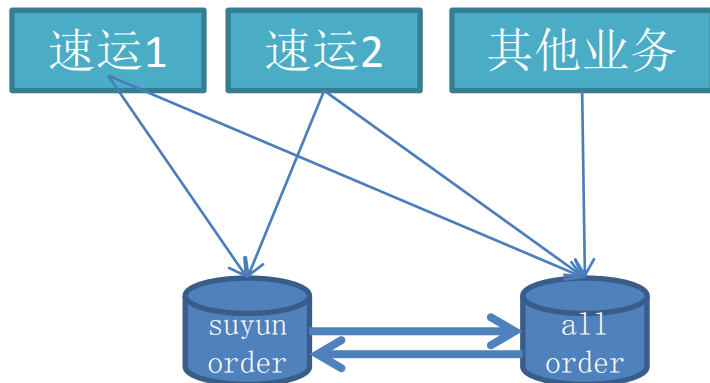
第一次技术演进 迁库、集群解耦

思考：

- 停服迁移？NO!!!

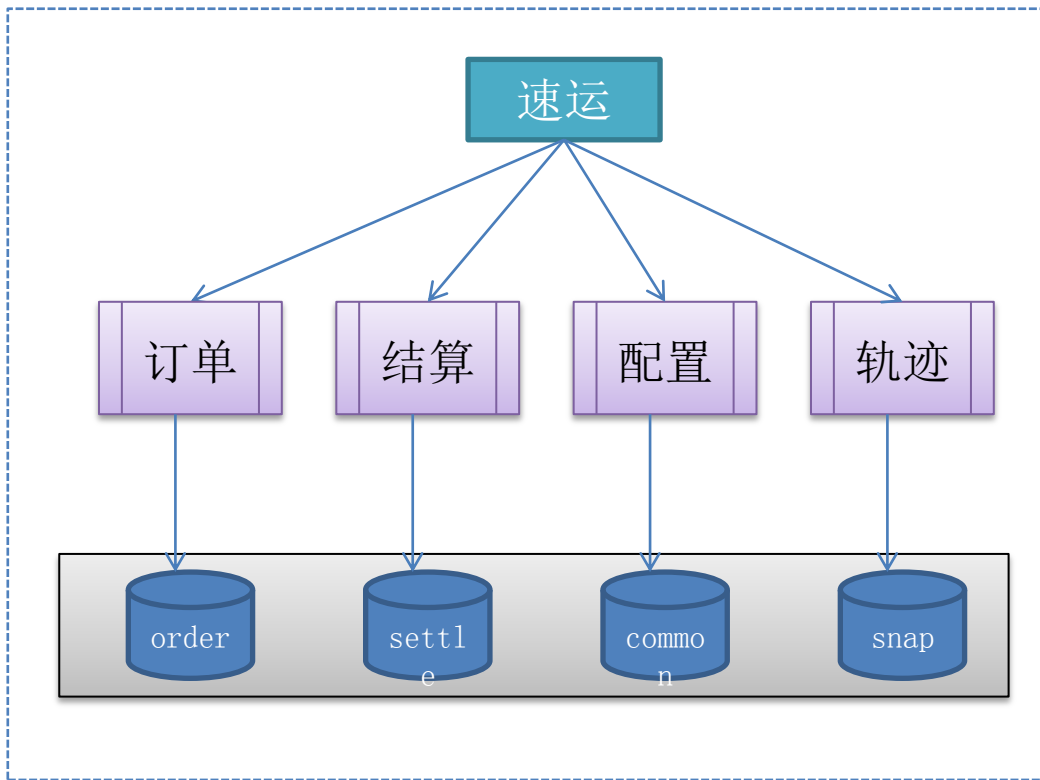
方案：

- 将速运的表单独同步到速运独立库
- 数据双向同步（特殊的订单ID保证主键不重复）
- 数据校验（数据写入日志，数据校对）。



迁移后：

- 按数据内容分库，解耦合
- 按业务量调整数据库配置，减少成本





高速发展

- 稳定高效


- 争分夺秒
- 补贴大战



同城供应链配送平台

高速发展中的问题:

- 快速迭代多人维护一套工程，效率差，BUG频发
- 业务高速发展，数据量急速增长
- 运营分析需求越来越复杂
- 补贴大战，大量无效补贴，运营成本高

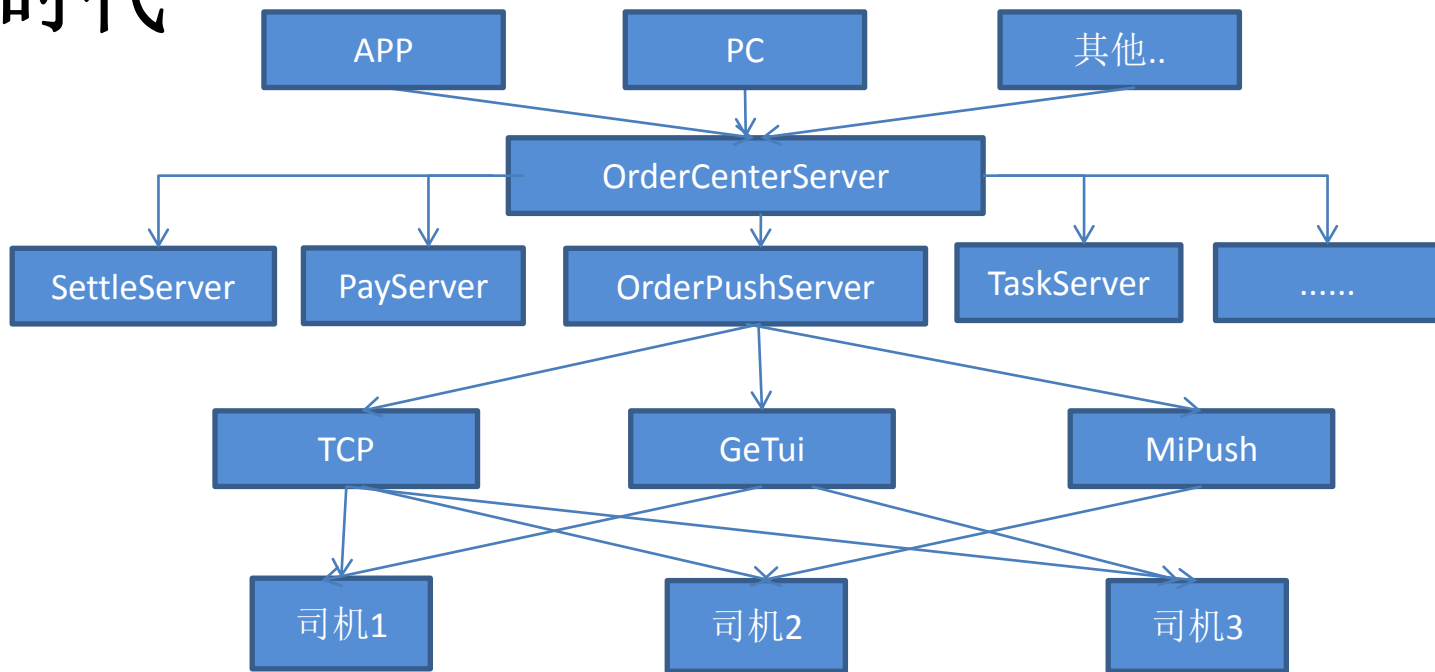


第二次技术演进 奔跑中的火车换轮子

- 服务化解耦
- 缓存&分库分表-提升系统性能
- 大数据平台-复杂需求分析

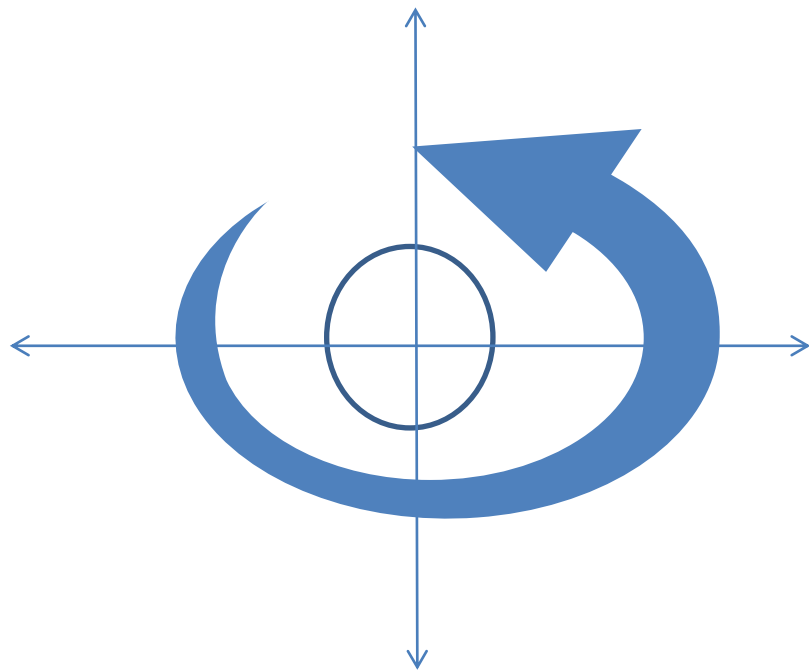
派单-铁器时代

- 模块服务化
- 推送多通道



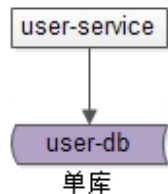
订单调度（探索）：

- 按距离、象限推送
- 多司机择优（好评、完成率）
- 多轮推送、补贴随抢单人数调整



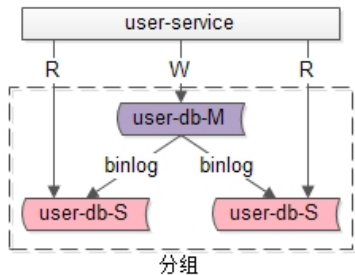
分库分表

项目之初



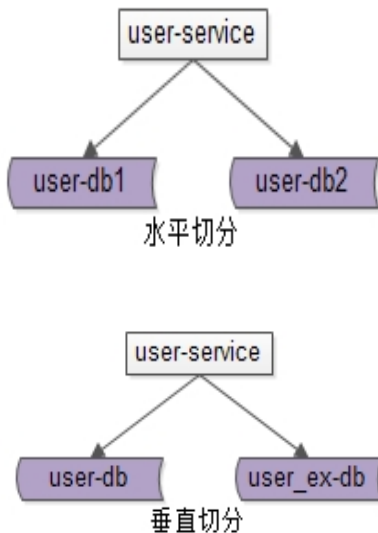
单库

读写分离

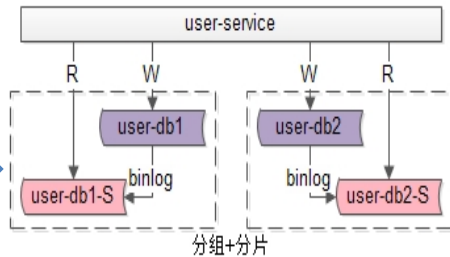


分组

水平/垂直拆分



分库+读写分离



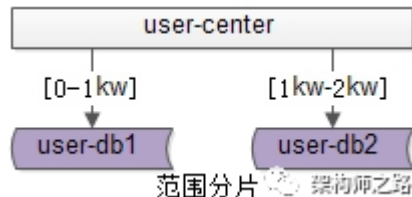
分组+分片

水平拆分

■范围法:

优点: 切分简单, 扩容简单。

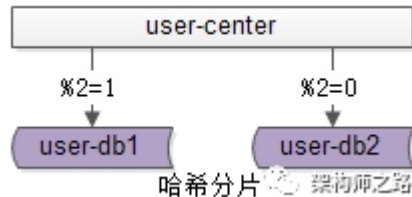
不足: Key必须增长, 负载不均匀。



■哈希法:

优点: 切分简单, 负载均衡。

不足: 扩容麻烦, 平滑迁移是需要解决的困难点



拆分后问题

- 部分查询变慢了

非partition key查询，需要遍历全部库。

- 运营需求无法实现

各种维度统计，没办法联表查询。

问题分析-任何脱离业务的架构设计都是耍流氓

■ 前端需求

uid → **99%**

login_name
phone
email
xxxxx } **1%**

■ 运营需求

按照年龄、性别、头像、登陆时间、注册时间来进行统计分析。

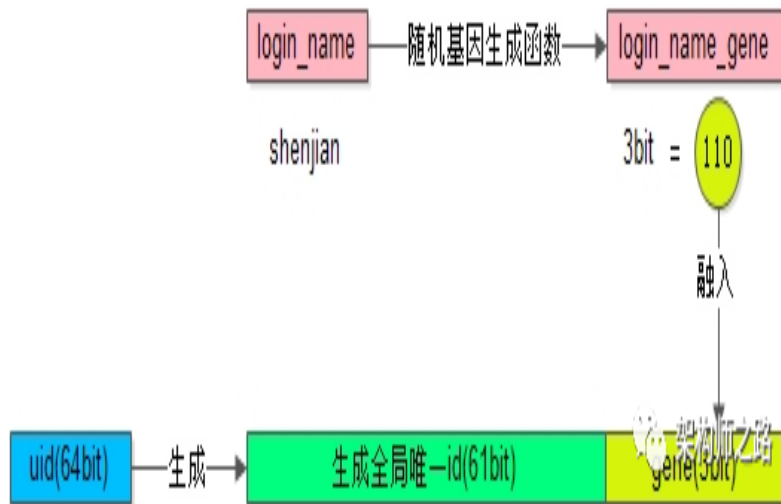
前端解决方案

- 索引表法
非partition key与uid建立索引表
- 缓存映射法
非partition key与uid映射关系放入缓存，缓存命中率高
- 非partition key生成uid
根据loginname或XXXX生成uid，需要技巧，有主键冲突风险
- 基因法
根据loginname或XXXX多个属性的部分字段生成uid

基因法

思路:

- 1) 假设分8库，采用 $uid\%8$ 路由，潜台词是，uid的最后3个bit决定这条数据落在哪个库上，这3个bit就是所谓的“基因”。
- 2) $f(login_name)$ 生成login_name_gene。
- 3) 生成全局唯一的id(61bit)
- 4) uid=全局唯一id与基因的拼装



运营侧需求解决方案

■ 冗余后台库

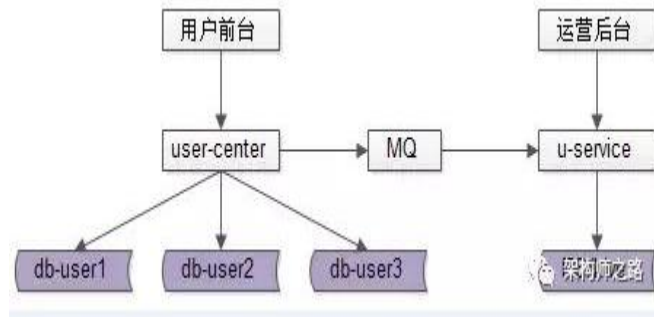
通过MQ\Canal实时同步到后台库

■ 外置搜索引擎

ES/Solr/XXXX

■ 大数据平台

实时拉取数据库数据进行统计和回写。





第三次技术演进 战斧项目

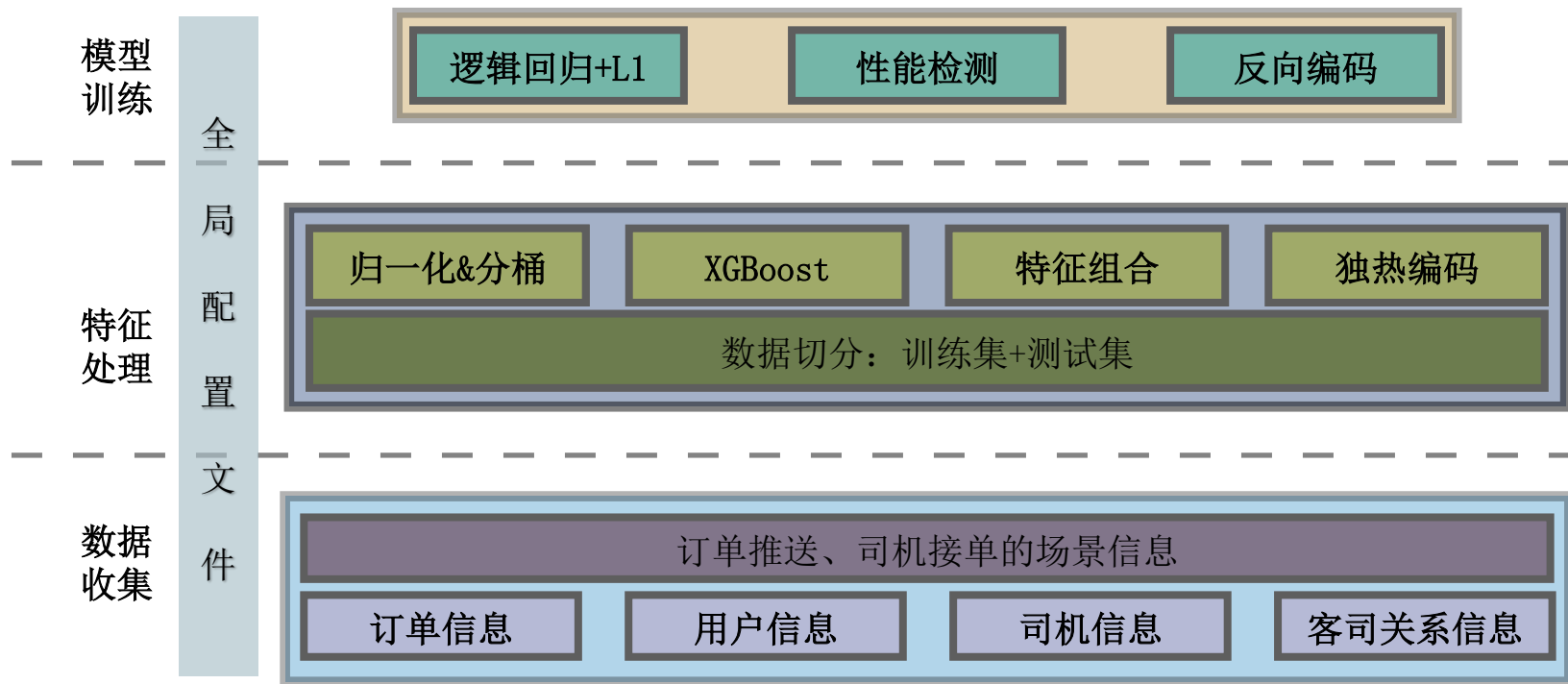
- 策略服务细化
- 智能模型接入
- 智能分流框架



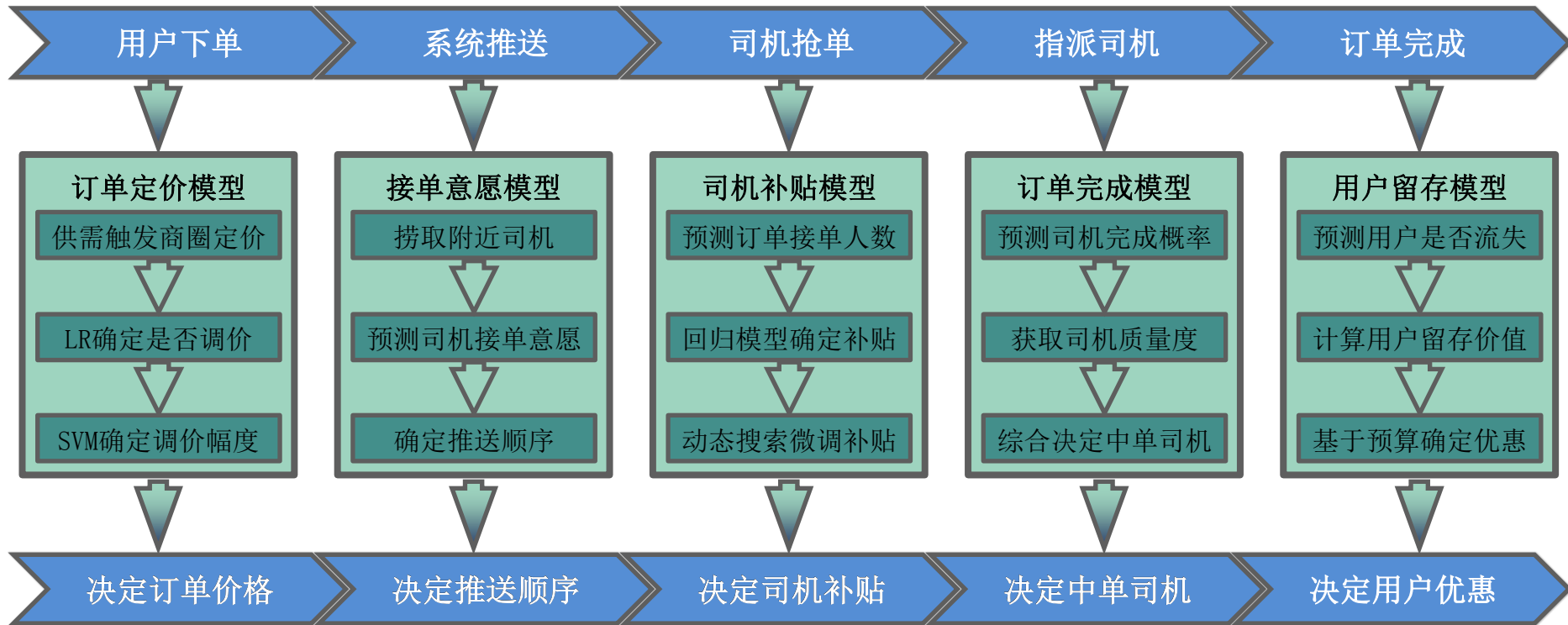
智能时代

-效率、精准

智能模型训练

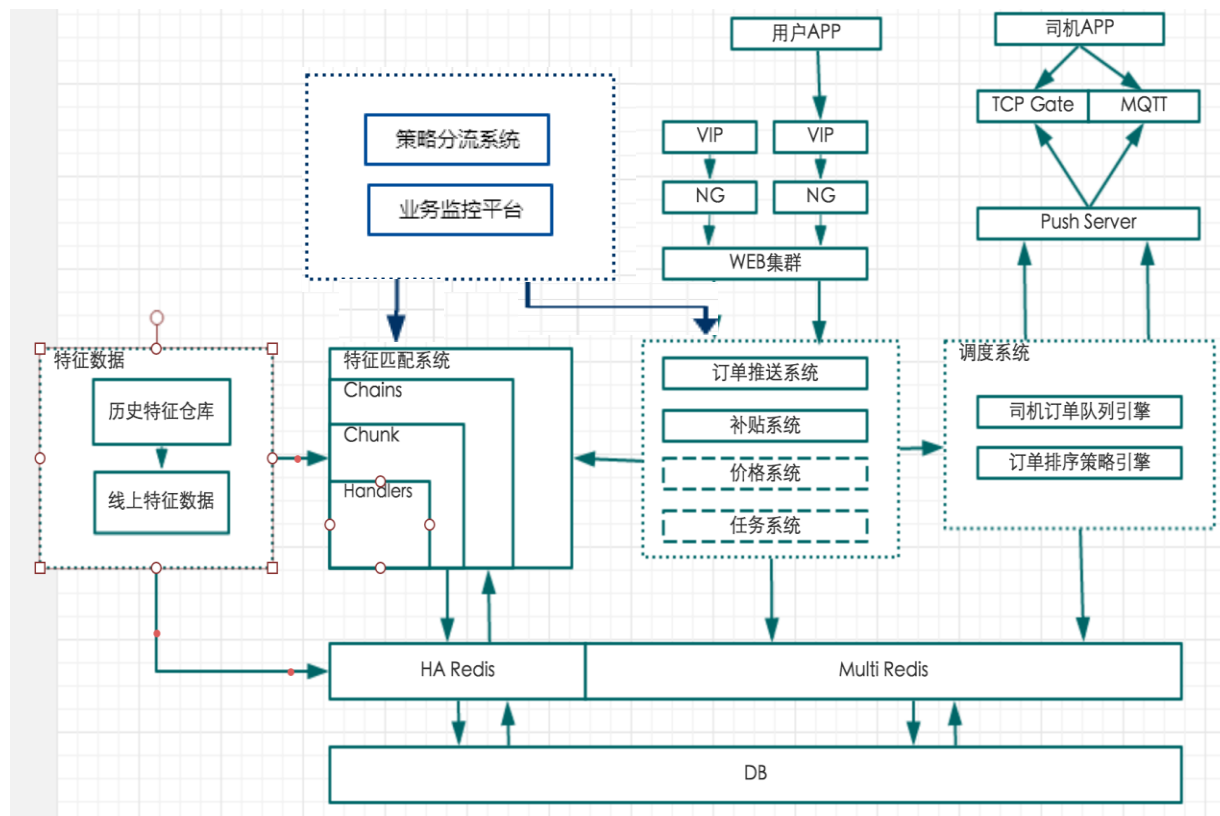


订单-模型运用



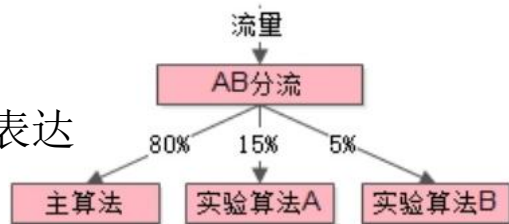
派单-智能时代

- 特征灵活变更
- 策略分流验证
- 业务实时监控



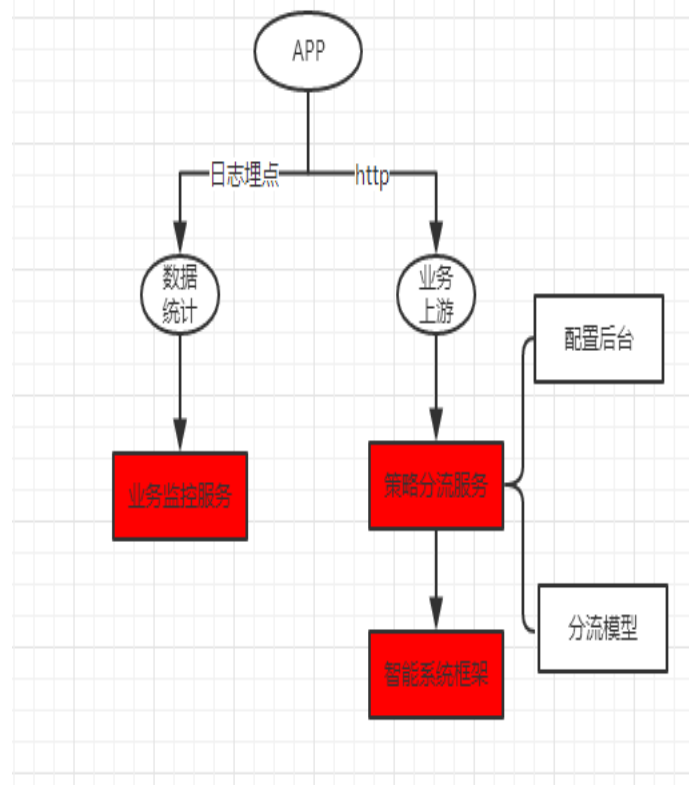
策略分流+监测

- 手机号&设备码&用户属性
&.....
- random、取模、与非正则表达式、集合
- 配置热加载
- 实时日志上报
- 业务监控预警（转化率）



支持实验的系统

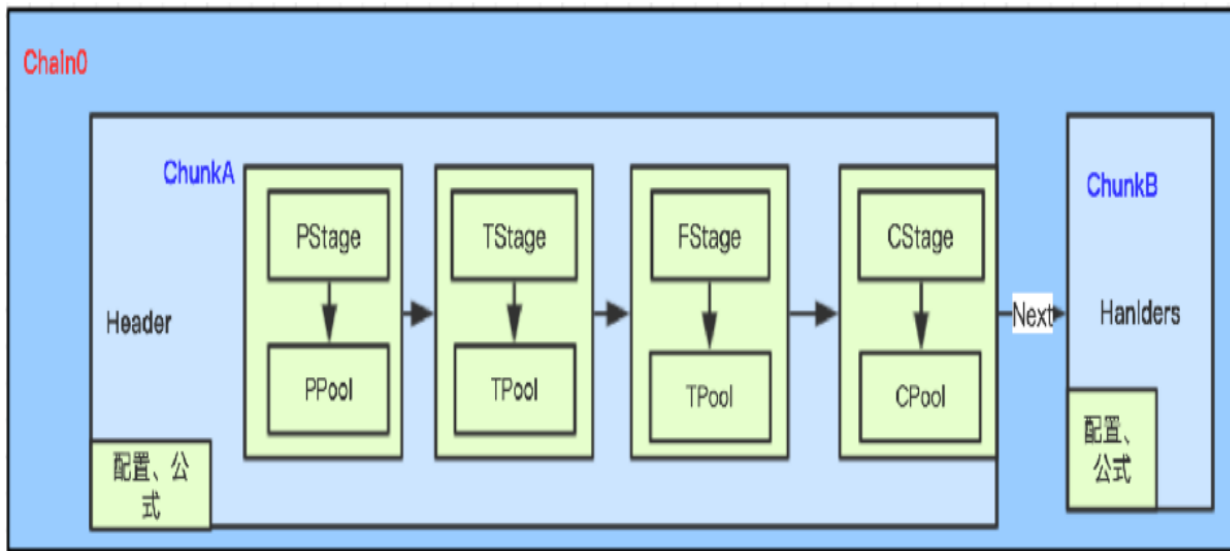
一、算法流量分配



二、算法分流监控

特征计算

- SEDA (Staged Event-Driven Architecture) , 阶段事件驱动
- 并行计算能力最大化
- 可插拔 (Chain、Chunk)
- 核心关闭, 个性开放



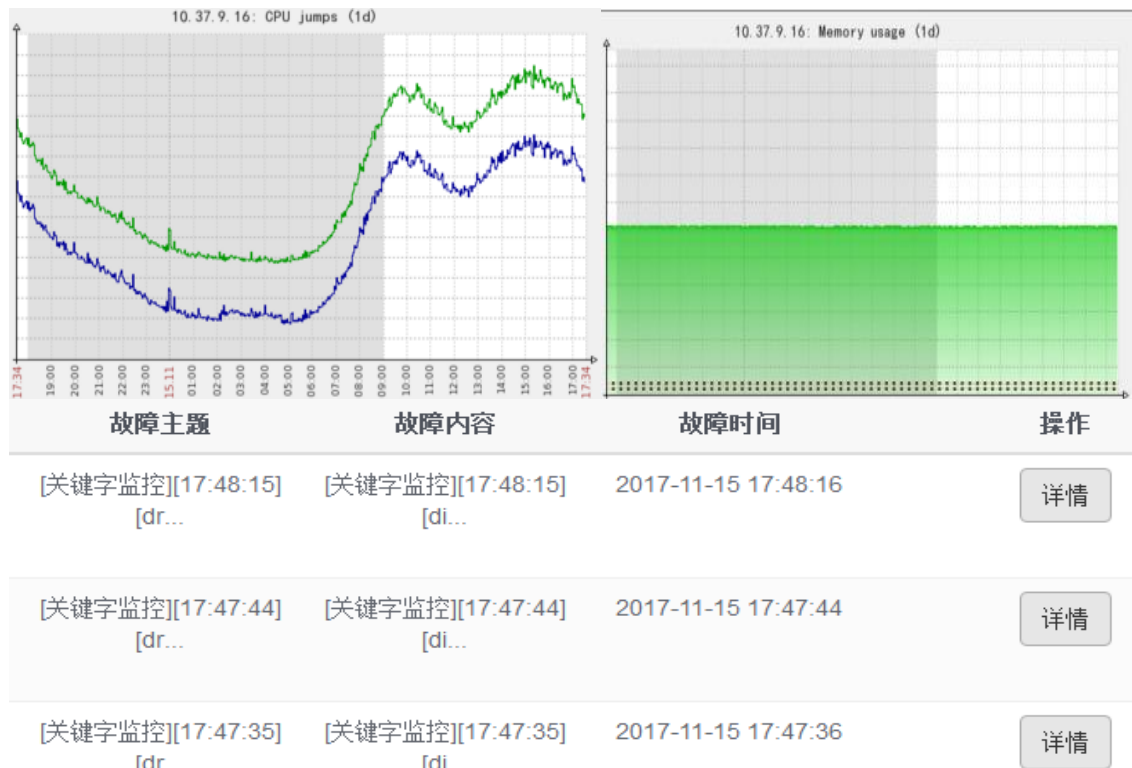


利器-监控平台

- 牛逼的算法需要稳定的系统支撑
- 业务的波动要第一时间知悉
- 提高问题排查效率就是在挽救损失

立体化监控

- 关键字监控
- 接口监控
- 流量
- PORT
- JVM
- CPU
- 线程
- CACHE
- DB
-

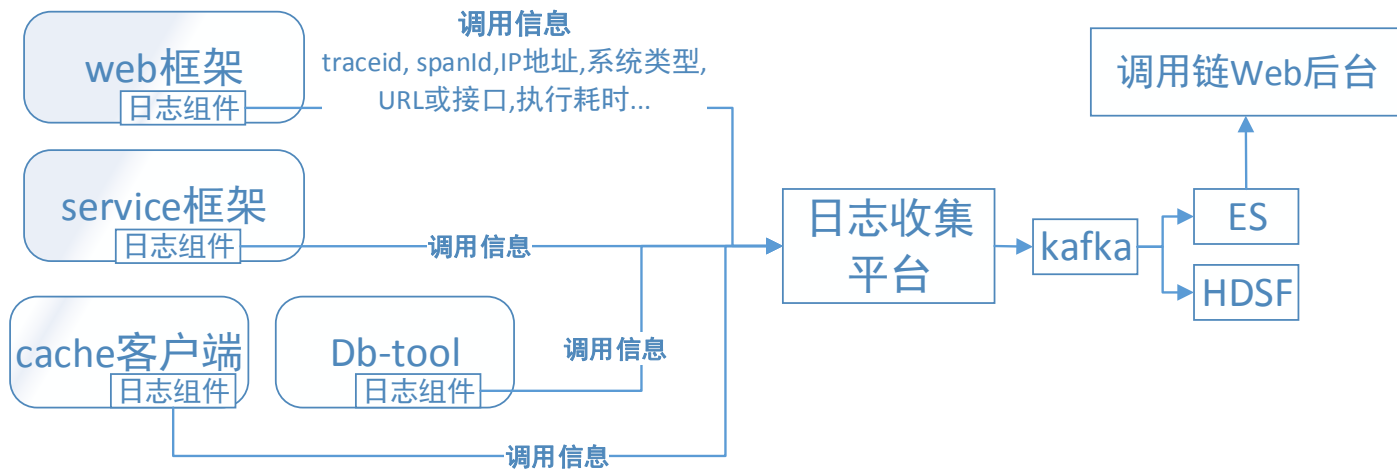




业务指标监控

- 渠道转化率
- 渠道取消率
- 渠道推送数量
- 异常订单数量
-

调用跟踪系统



调用跟踪系统

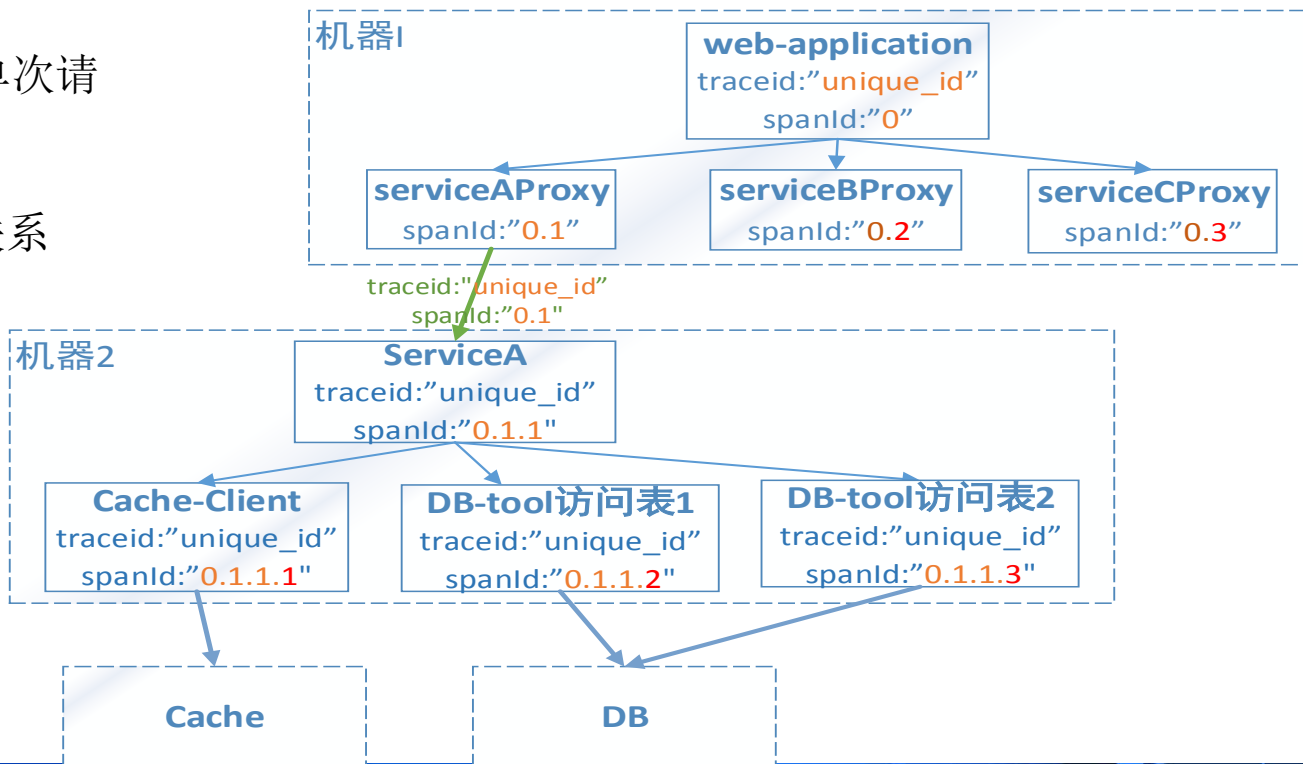
- 全局跟踪
- 异常报警
- 清晰的调用结构
- 整体效果监测

开始执行时间	耗时	起始IP	请求url	结构图
2017-10-13 18:06:59.843	48ms	10.37	ht[redacted].daojia.com/ap...	

集群名称	类型	状态	IP	接口	参数	sql相关耗时	时间轴	
[redacted]	yun_daojia	dwf	OK	10.37	http://suyun-guest.daojia.com/a...	undefined		48ms
[redacted]	yun_daojia	dsf(c)	OK	10.37	sy_common.AppVipAccountSer...	[707838118367207424]		1ms
[redacted]	yun_daojia	dsf(c)	OK	10.37	dus.AppUsersimpl.getIdByMob...	[15920134665]		1ms
[redacted]		dsf(s)	OK	10.37	AppUsersimpl.getIdByMobile(l...	[15920134665]	{dbConsum...	0ms
[redacted]	yun_daojia	dsf(c)	OK	10.37	sy_daojiacommon.AppAccountS...	["clazzlu003d1 and uid\u003d...		1ms
[redacted]	common	dsf(s)	OK	10.37	AppAccountService.load(String...	["clazzlu003d1 and uid\u003d...		1ms
[redacted]	yun_daojia	dsf(c)	OK	10.37	sy_daojiacommon.AppDiscount...	[" uid\u003d707838118367207...		0ms
[redacted]	daojiacommon	dsf(s)	OK	10.37	AppDiscountService.load(String...	[" uid\u003d707838118367207...		0ms

调用跟踪系统

- 全局唯一的traceid将单次请求调用链串联
- spanId描述调用层级关系
- traceid、spanId的透传
- 数据采集-flume



总结-Tips

- 1、不同的阶段采用不同的架构，技术的重点跟随业务转变。
- 2、订单的推送通道，建议使用双通道，保证推送的到达率。
- 3、数据库的水平拆分，在资源允许的情况下，强烈建议分库。
- 4、算法线上分流验证必须要有实时的监控和自动流量切换。
- 5、监控很重要，第一时间发现问题，减少影响

The background is a solid blue color. In the corners, there are decorative elements consisting of dark blue spheres connected by thin white lines, forming a network or molecular structure. Additionally, there are white geometric shapes: a large inverted triangle at the top and a large triangle at the bottom, both formed by thin white lines.

Gdevops

全球敏捷运维峰会

THANK YOU!