China · Beijing

# 滴滴的高可用建设实践

许令波（君山）

2016/12

ArchSummit
全球架构师峰会 2016

[北京站]

Geekbang. InfoQ

# 大纲

**一．关于我**

**二．难点与挑战**

**三． 高可用体系化建设思路**

- 压测体系、管控体系、监控体系、恢复体系和度量体系建设

**四．全链路压测、服务治理等在滴滴场景下技术方案设计实践介绍**

**五．踩过的坑以及经验教训和总结**

# 关于我

http://xulingbo.net
Java、Velocity、sketch、Cassandra
性能优化、静态化架构、稳定性
《深入Java Web技术内幕》

- 2009加入阿里
- 商品详情优化
- 双11交易链路优化
- 共享All in 无线化改造
- 商品平台稳定性
- 负责过Detail、店铺、图片空间、Vsearch实时搜索业务系统
- 2016加入滴滴参与基础平台的建设，负责服务框架、小对象存储以及一些横向的高可用建设项目

# 大纲

一．关于我

二．难点与挑战

三． 高可用体系化建设思路

- 压测体系、管控体系、监控体系、恢复体系和度量体系建设

四．全链路压测、服务治理等在滴滴场景下技术方案设计实践介绍

五．踩过的坑以及经验教训和总结

# 故障的影响有多大

- 影响成千上万的用户
- 会带来公司及用户的资产损失
- 公司声誉受损

# 网站的可用性指标

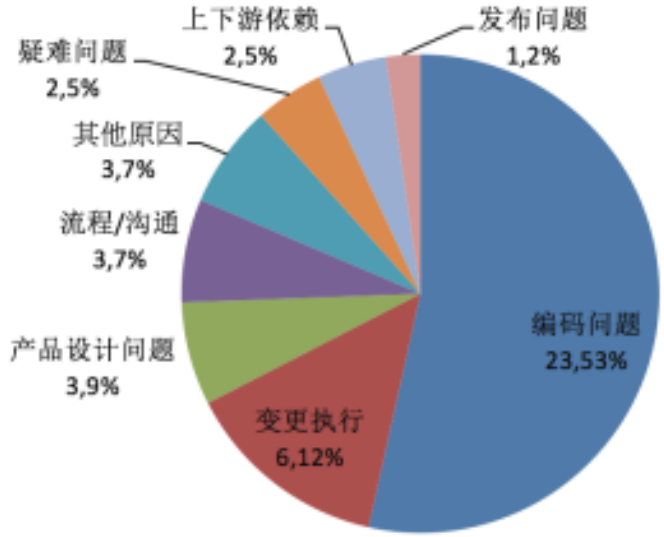
- 业界用N个9来量化可用性

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}} \times 100 = 99.XXXX\%$$

| Level of Availability | Percent of Uptime | Downtime per Year | Downtime per Day |
|---|---|---|---|
| 1 Nine | 90% | 36.5 days | 2.4 hrs. |
| 2 Nines | 99% | 3.65 days | 14 min. |
| 3 Nines | 99.9% | 8.76 hrs. | 86 sec. |
| 4 Nines | 99.99% | 52.6 min. | 8.6 sec. |
| 5 Nines | 99.999% | 5.25 min. | .86 sec. |
| 6 Nines | 99.9999% | 31.5 sec. | 8.6 msec |

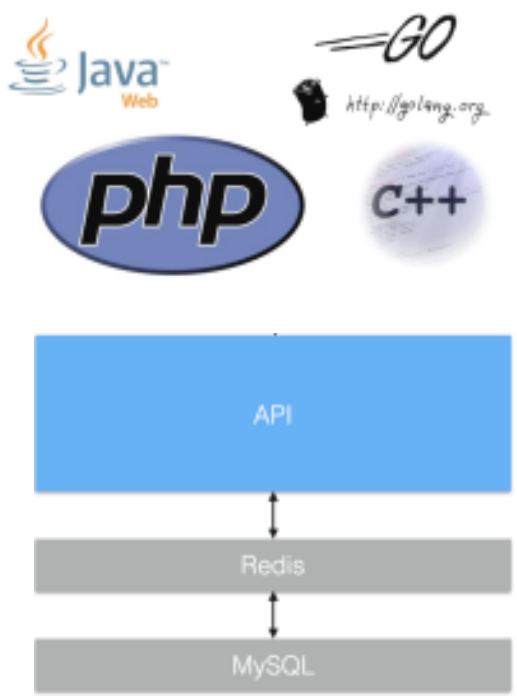
# 影响稳定性的因素

- **编码问题和变更导致的故障最多**
- **一系列巧合导致?**
  - 测试不到位
  - 环境不熟悉
  - 原理不清楚
  - 流程管理缺失
- **是人都犯错？**
  - 想当然
  - 怕麻烦
  - 盲目自信



# 难点与挑战
## -高速公路上换轮子

- 基础设施正在建设
  - 机房、网络、网关、DNS等重新规划
- 技术栈不统一
  - 人的技术协作沟通成本高代理
  - 需要解决跨语言访问问题
  - 中间件等基础SDK维护成本
- 业务层耦合比较多
  - 要挂一起
  - 代码不敢改



# 大纲

一．关于我

二．难点与挑战

三． 高可用体系化建设思路

- 压测体系、管控体系、监控体系、恢复体系和度量体系建设

四．全链路压测、服务治理等在滴滴场景下技术方案设计实践介绍

五．踩过的坑以及经验教训和总结

# 高可用建设思路

## — 规范建设

➢**日常规范**
- 日常测试、beta测试、发布前依赖检查

➢**问题发现**
- 准确实时的发现系统指标异常和业务指标异常
- 能够动态的调整异常指标阀值，减少异常误报率

➢**问题排查**
- 能够记录异常栈信息、分析异常指标的特征如那个机房？什么时间段？历史上是否出现过？
- 将异常指标关联起来，如load高是否QPS也高，当前是否有系统变更事件发生

➢**数据订正**
- 可以方便的添加要订正的数据源，提供自主化的测试界面

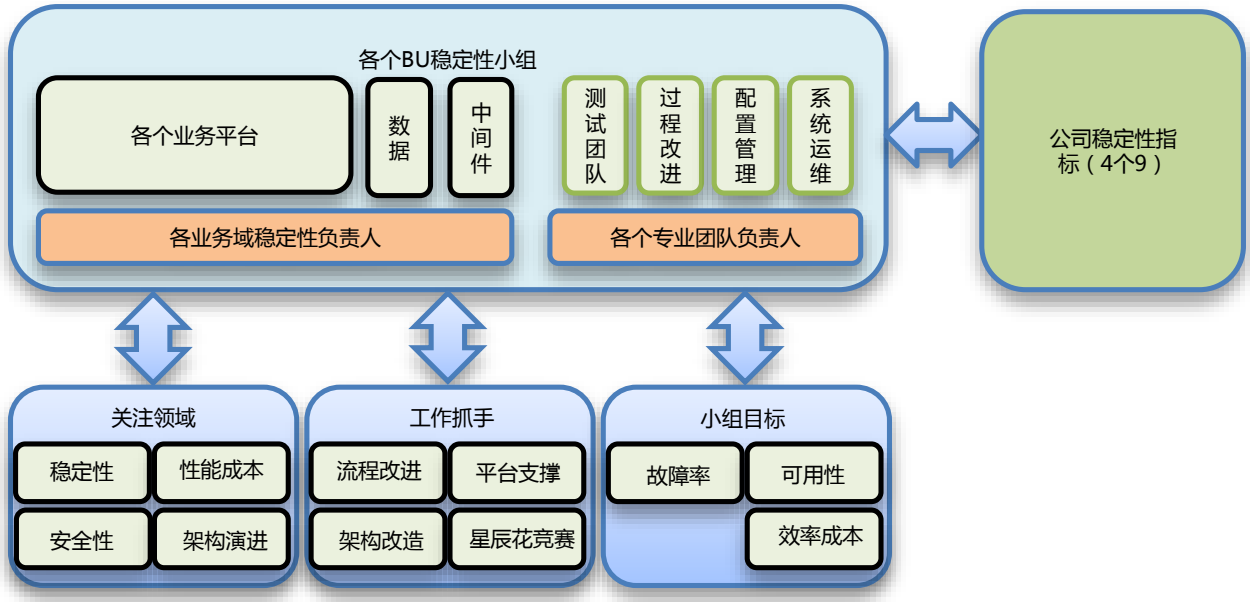
➢**总结沉淀**
- 发生故障要及时总结原因和改进措施，分享给其他及后来人，避免重复犯错

# 高可用建设思路
# — 工具建设

- **压测体系**
  - 发现系统瓶颈（后面会以全平台压测为例介绍）
- **管控体系**
  - 异常情况时做一些保护措施
- **监控体系**
  - 实时发现问题
- **恢复体系**
- **度量体系**
  - 知道是好是坏

# 高可用建设思路

## — 组织建设



# 大纲

一．关于我

二．难点与挑战

三． 高可用体系化建设思路

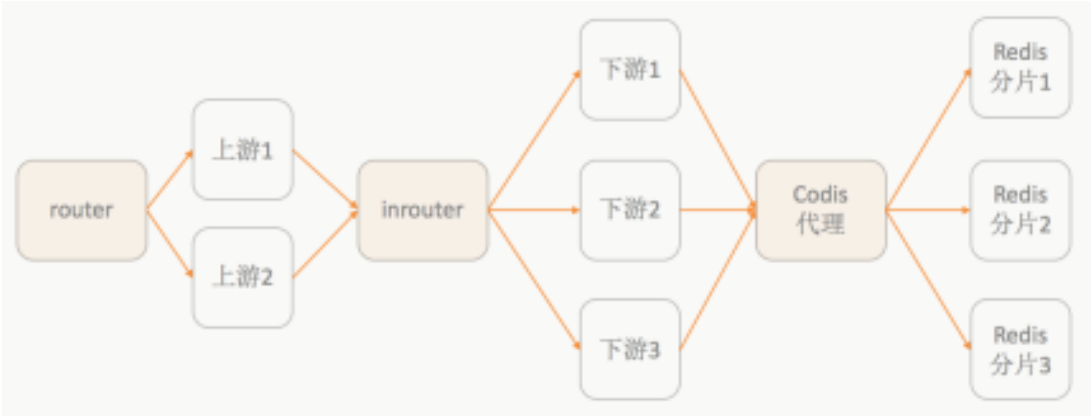
- 压测体系、管控体系、监控体系、恢复体系和度量体系建设

四．全链路压测、服务治理等在滴滴场景下技术方案设计实践介绍

五．踩过的坑以及经验教训和总结

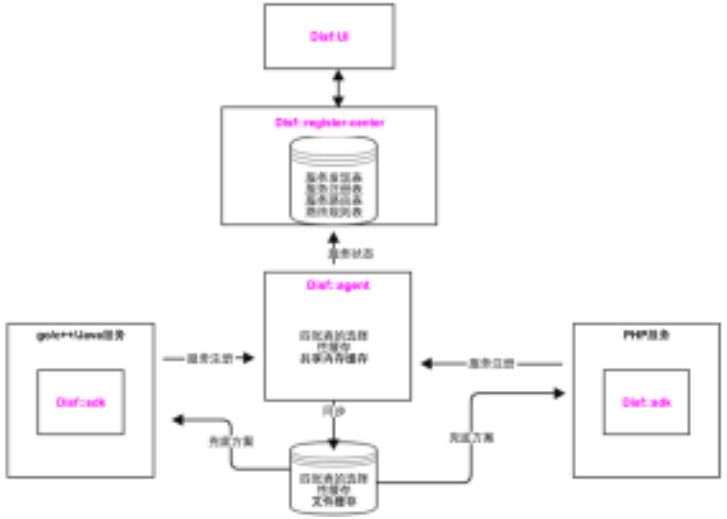
# 业务架构 — 服务治理

- 上下游依赖硬编码在代码里；
- 没有使用inrouter/tgw的ip:port摘除和扩容需要代码重新上线；
- Inrouter有网络链路稳定性隐患，以及效率上的损失；
- 没有清晰的服务目录，API文档以及SLA和监控。



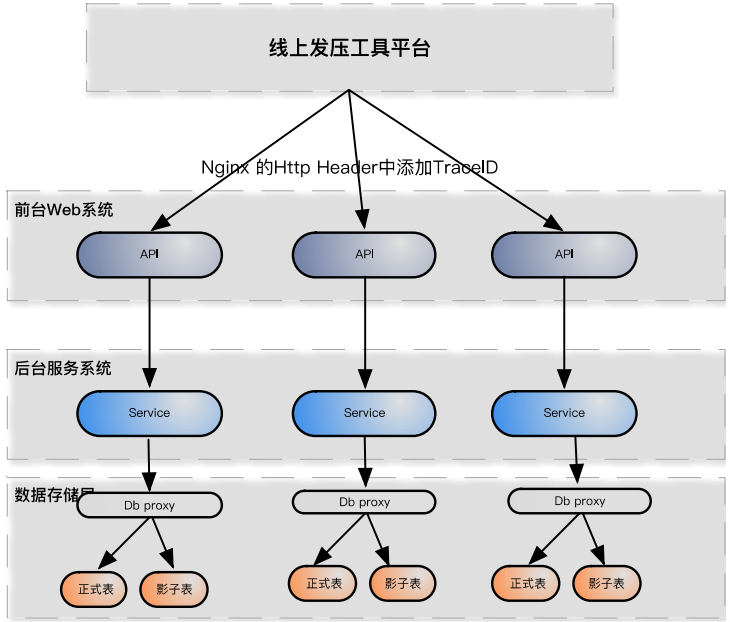
# 业务架构— DiSF服务框架

- 接口定义统一IDL描述
- 解决PHP不支持长连接问题
- 多语言SDK的支持，不同语言之间的服务可以互调
- 故障节点摘除、路由切换



# 压测体系
# —全平台压测思路

- 通过trace传递标识流量
- 中间件等通过标记将流量导到影子表



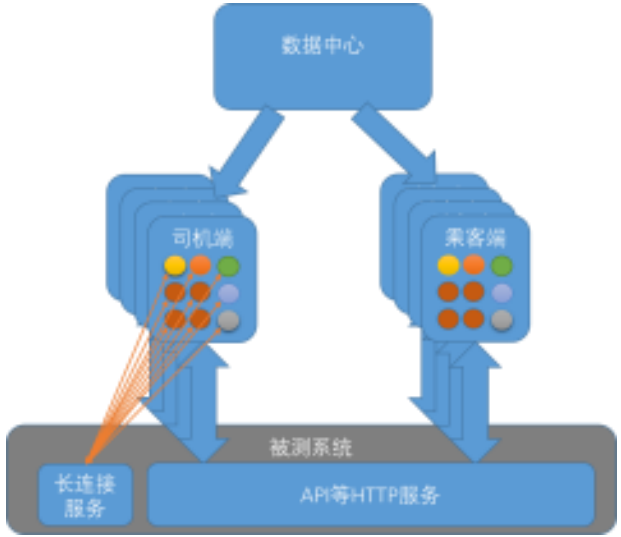
# 全平台压测 — 数据

- 司机按照线上司机切片位置投放
- 司机在切片位置中随机选点按路径规划运行
- 订单按照历史订单数据投放接单后司机、乘客
- 根据订单起终点按路径规划运行

- 隔离性
  - 通过路由测试请求到影子表，物理上做了隔离
  - 把可能用来做缓存key的字段按规则替换，避免key冲突
- 真实性
  - 数据量是线上高峰时期的数据量
  - 从线上数据迁移，保留数据的属性，保证真实性
  - 根据非业务字段筛选，保留线上真实比例
- 完整性
  - 关键字段按照统一规则替换，确保替换前后关联关系一致

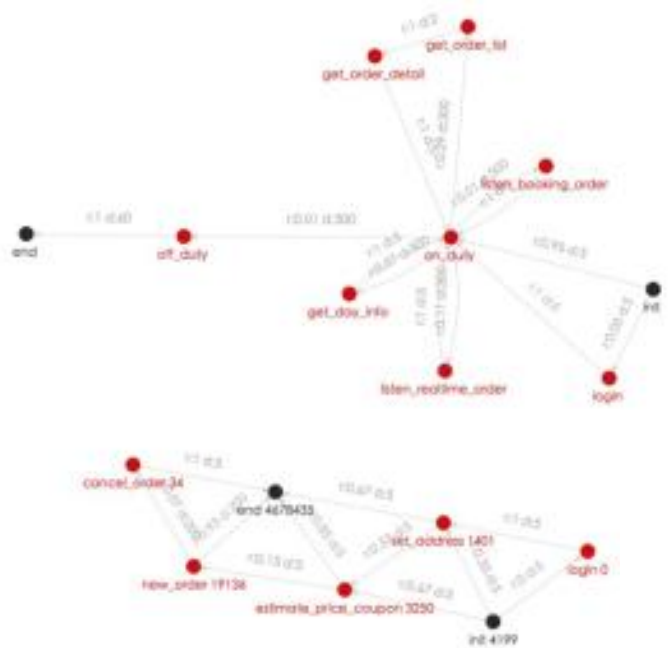
# 全平台压测 — 发压工具

- 数据中心负责机器人的数量投放、大脑配置、位置和订单数据配置
- 单个进程模拟N个机器人
- 每个机器人有独立的长连接



# 全平台压测 — 状态流转

- 真实模拟司机的状态流转
- 模拟乘客的状态流转

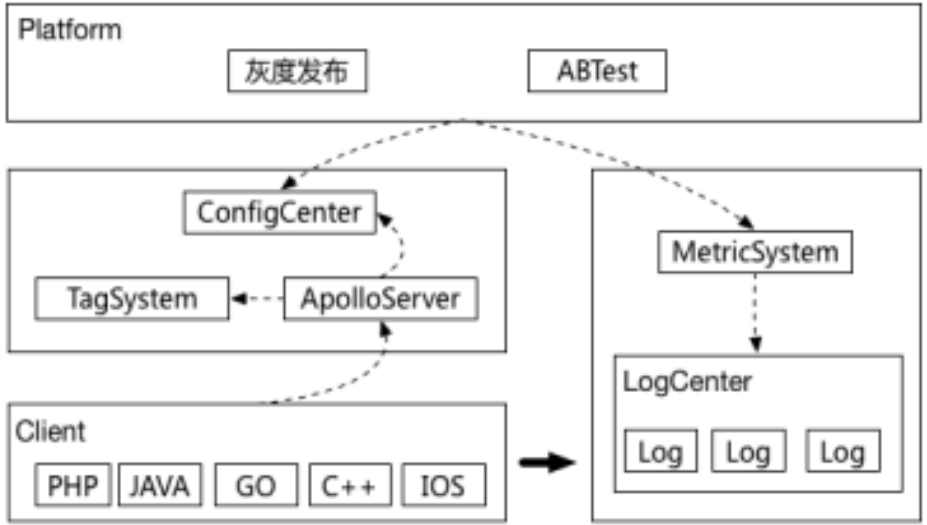


# 全平台压测 — 经验

- **支持测试标记路由**
  - 支持标识透传
  - 支持测试数据路由影子表
  - 支持全链路全局开关
- **解决缓存错乱问题**
  - 缓存覆盖保护
  - 防击穿处理问题（正常请求带标预防）
  - 持久化缓存失效
- **非核心系统隔离**
  - NOTIFY隔离非核心系统
  - Metaq隔离非核心系统
  - 非核心系统禁止写接口调用
- **数据脱敏**
  - 压测日志隔离
  - Bi数据隔离
- **链路风险**
  - 安全拦截
  - 安全验证码
  - 从CDN压，模拟真实流量

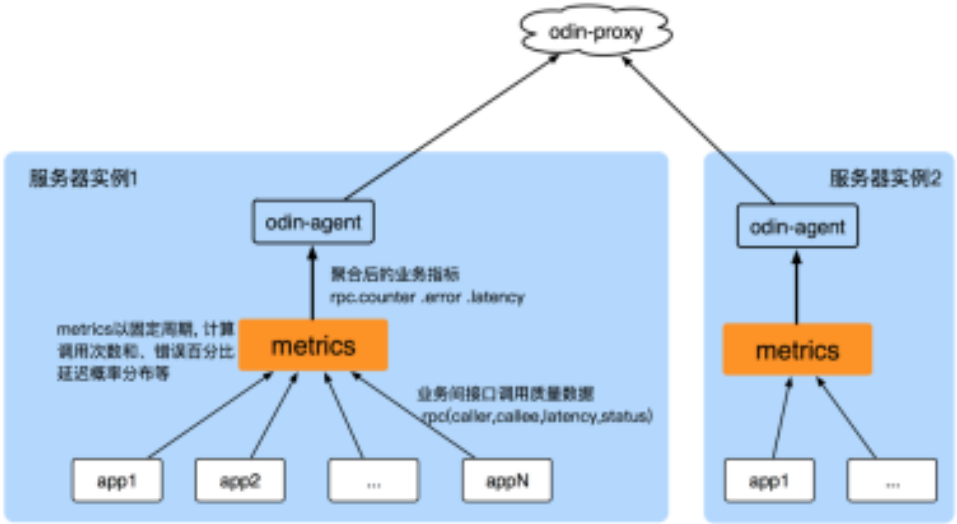
# 管控体系
# —开关平台

- 支持多种业务场景：定向投放、灰度发布、A/BTest
- 支持多语言client的接入
- 可以支持基于内存和持久化的
- 开关操作单机和集群操作都支持



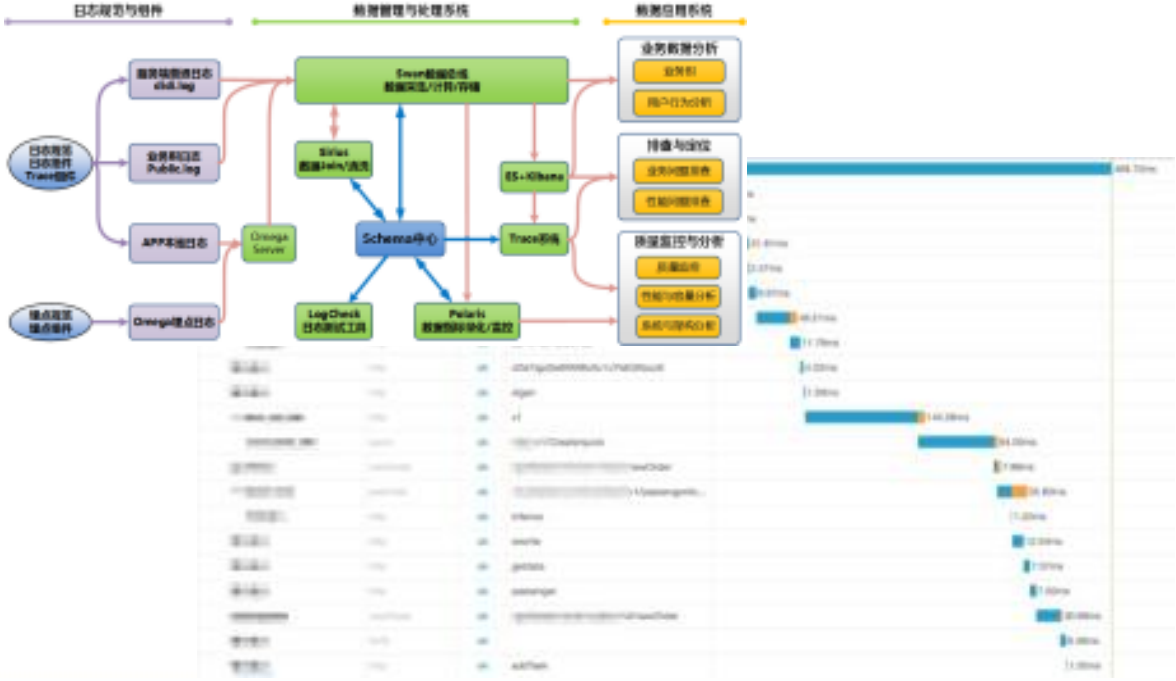
# 监控体系
# —metrics监控平台

- 内存采集数据，实时性非常高
- 采用UDP协议收集数据



# 监控体系
# —数据链路监控

- 通过Trace关联调用链路，全息展示业务调用轨迹
- 每个调用节点的qps、rt等都可以看到



# 恢复体系
# —数据订正平台

- 支持可以录入规则的一个数据订正平台
- 打通各种数据源，只需要写订正逻辑
- 目前还在建设中。。。

# 度量体系
# —容量、性能基线

- 性能基线：知道各个业务系统的当前的单机性能
- 链路基线：知道各个请求调用的依赖关系的变化
- 机器容量基线：知道各个系统的机器水位情况
- 正在建设中。。。

# 问题排查一些思路

- **案例**
  - kafka proxy触发的一次故障
  - redis集群一次故障
- **重现问题**
  - 相同的客户端/服务器/环境
  - 相同的用户
  - 能够重现，就能够快速定位问题，并解决问题
- **不好重现问题**
  - 确定影响范围
  - 定位问题可能产生的范围
    - 并发导致
    - Cache引起
    - 单台机器的环境不一致
    - 数据边界
    - 机器负载高

# 一些教训

- 勇于尝试
- 充分测试
- 熟悉线上环境
- 掌握工作原理
- 掌握工具
- 从事故中学习

- 应用等级划分
- 故障等级划分
- 故障review/action
- 故障责任追究

# 大纲

一．关于我

二．难点与挑战

三． 高可用体系化建设思路

- 压测体系、管控体系、监控体系、恢复体系和度量体系建设

四．全链路压测、服务治理等在滴滴场景下技术方案设计实践介绍

五．踩过的坑以及经验教训和总结

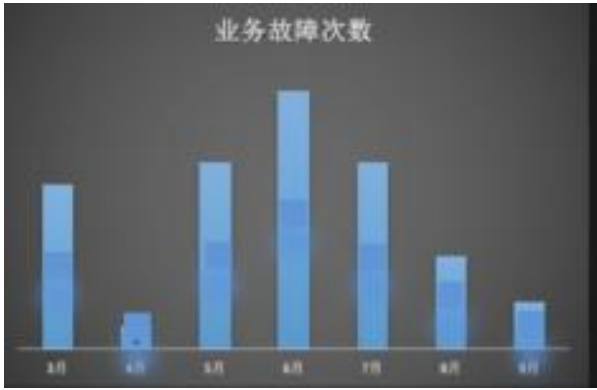
# 一些经验

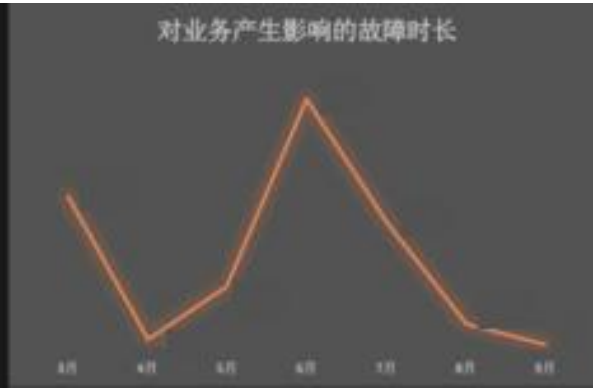
- **变更之中出现问题第一时间回滚**
- **在变更之前必须制定回滚方案**
  - 对变更内容设置开关，出现问题可以快速通过开关关闭新功能
  - 接口变更、数据结构变更，回滚要考虑第三方依赖

- **指导原则：**
  - 将故障清晰描述和暴露出来，获取第一手资料，找到问题反馈源头
  - 先解决问题，消除故障
  - 找到对应系统和业务的直接负责人
- **处理流程：**
  - 问题发现后第一时间上报到“消防群”
  - 组建应急处理小组
  - 跨团队合作，通知到对方系统的负责人，P1故障要通知到客服、公关接口人，尽量做到集中办公
  - 问题处理完毕，立即总结和制定改进方案
  - 系统TL负责，改进方案的执行情况

# 滴滴稳定性建设成果

- **下半年滴滴在稳定性建设上提升很大**
  - 从KPI到规范、制度、运营等多重组织建设
  - 从单系统到横向架构的优化
  - 从客户端到后端应用系统再到基础平台等全流程稳定性建设
- **稳定性需要持续建设**
  - 单元化、异地多活
  - 容量规划、弹性部署



希望大家继续关注和使用滴滴

# THANKS


- 邮箱:xulingbo0201@163.com
- http://xulingbo.net