QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2017

微服务在小米消息推送的实践和感悟

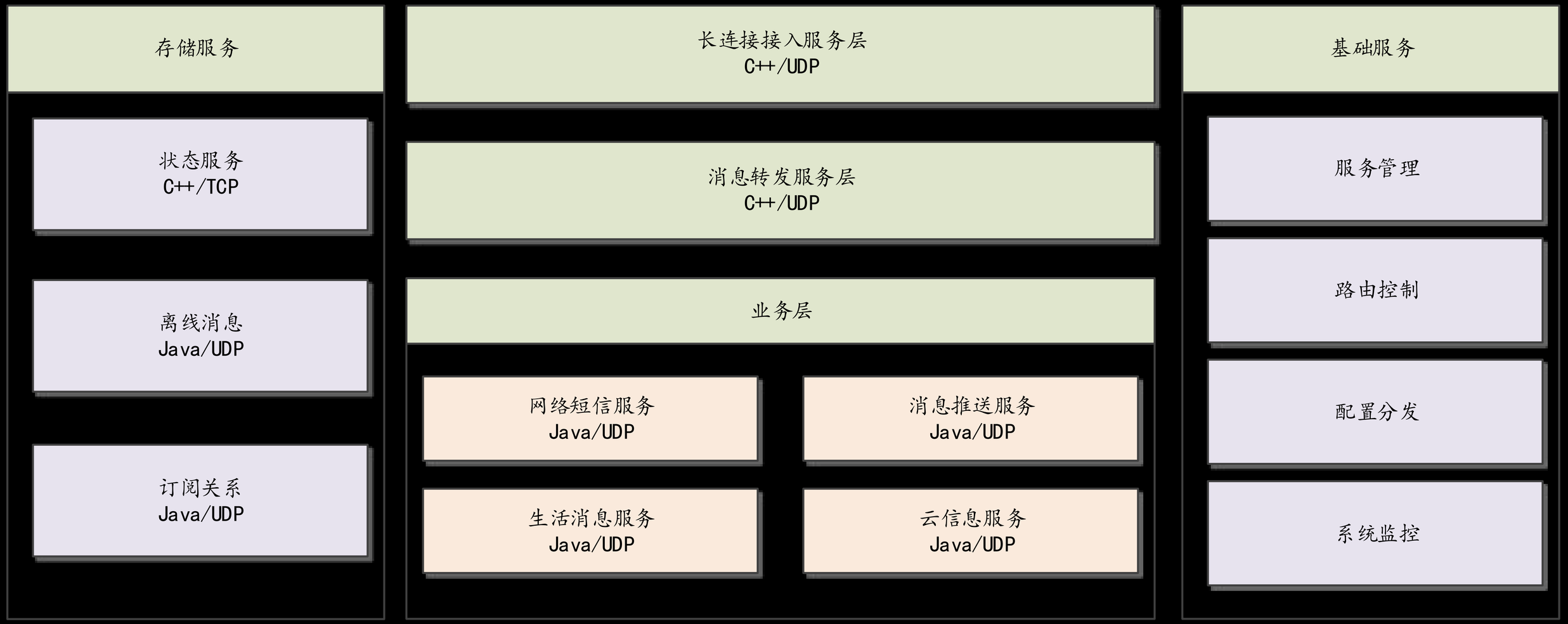
夏超

主办方 Geekbang. 极客邦科技 InfoQ

目录

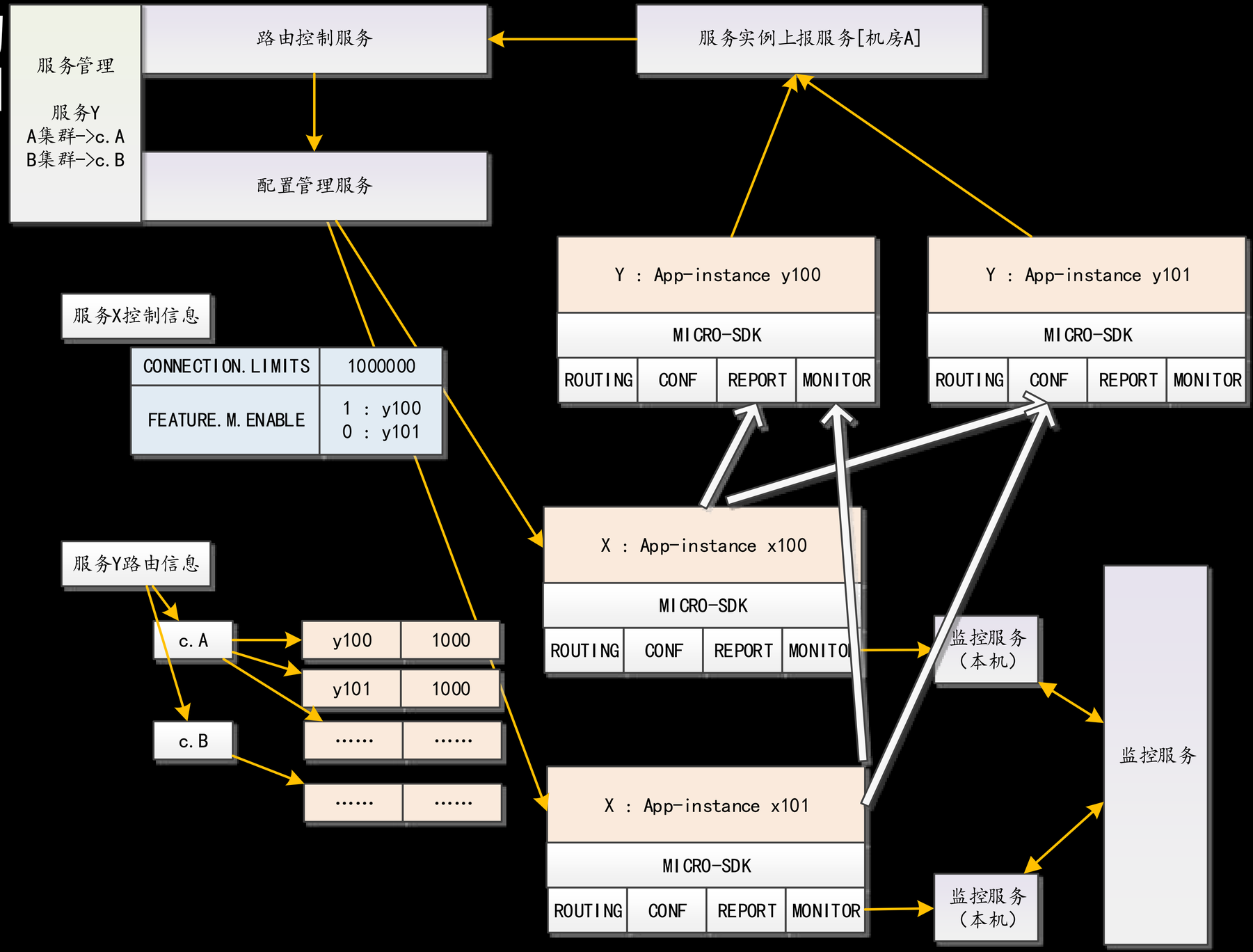
- 动机
  - 降低系统耦合
  - 提升系统可扩展性
  - 提升开发效率
- 挑战
  - 系统的整体稳定性
  - 排查定位问题困难
- 治理
  - 服务管理
  - 路由控制
  - 监控升级
  - 有状态服务改造

总体架构


存储服务
状态服务
C++/TCP
离线消息
Java/UDP
订阅关系
Java/UDP
长连接接入服务层
C++/UDP
消息转发服务层
C++/UDP
业务层
网络短信服务
Java/UDP
消息推送服务
Java/UDP
生活消息服务
Java/UDP
云信息服务
Java/UDP
基础服务
服务管理
路由控制
配置分发
系统监控

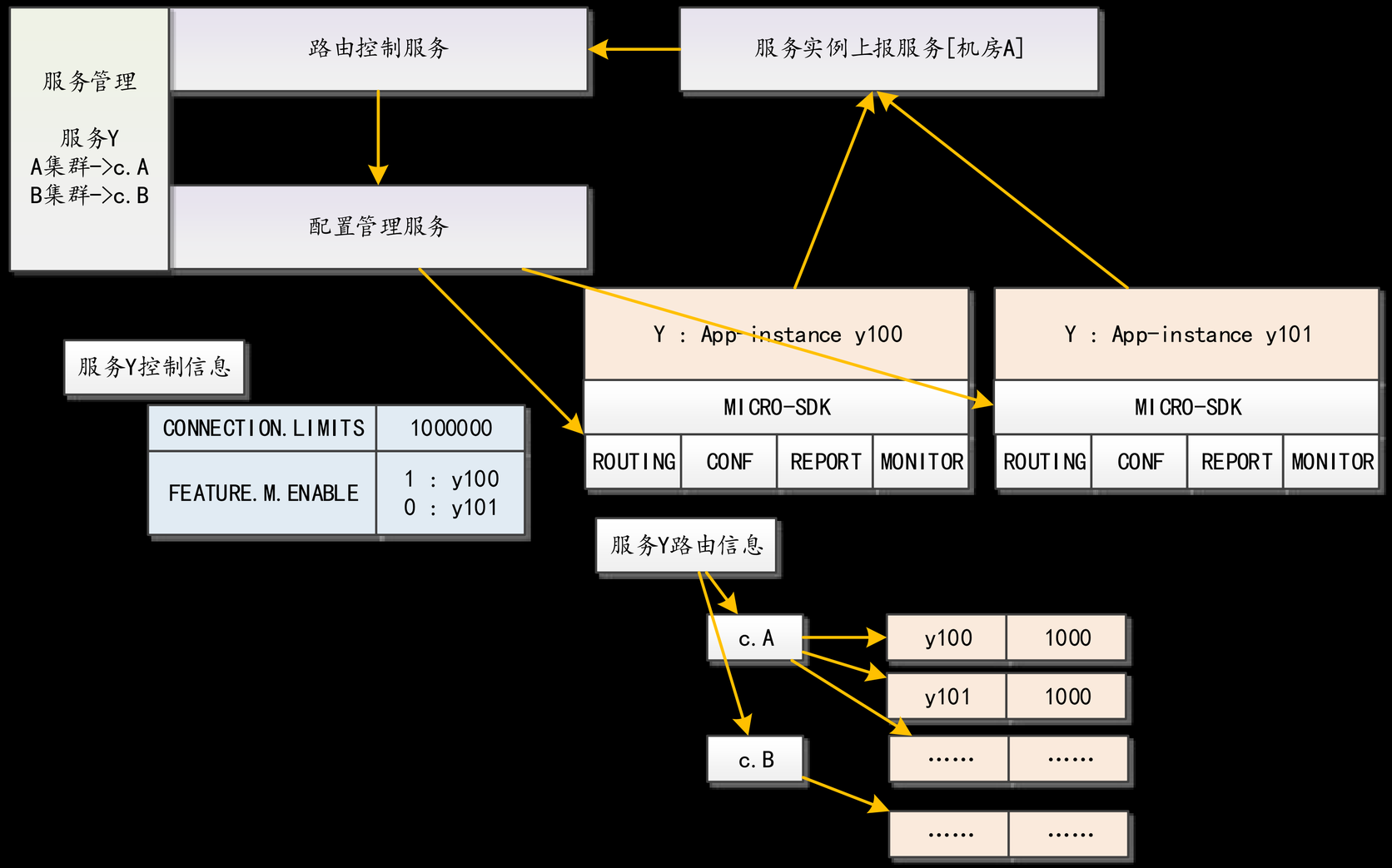
总体介绍

- 服务X和服务Y的相互关联，以及其和整体系统的交互



微服务治理——服务管理

- 服务的注册和发现
- 服务的限流控制
- 服务的发布和升级
  - 使用动态配置
  - 高级灰度控制



微服务治理——服务管理——灰度控制

- 程序开发规范
  - 灰度开关
  - 选择逻辑配置化
- 逐步灰度
  - 实例维度
  - 流量维度
  - 组合维度

```
EANBLE_NEW_FEATURE (CONFIGURE)

INPUTS=string CLUSTER, string MACHINE, string API,long ID

RULE.CLUSTER_A = CLUSTER IN (A)
RULE.CLUSTER_B = CLUSTER IN (B)
RULE.API = API IN (LOGIN, LOGOUT)
RULE.ID = ID MOD 1000 IN (100..200,450..600)

JUDGE = RULE.CLUSTER_A && RULE.ID
JUDGE = RULE.CLUSTER_B && RULE.API
```

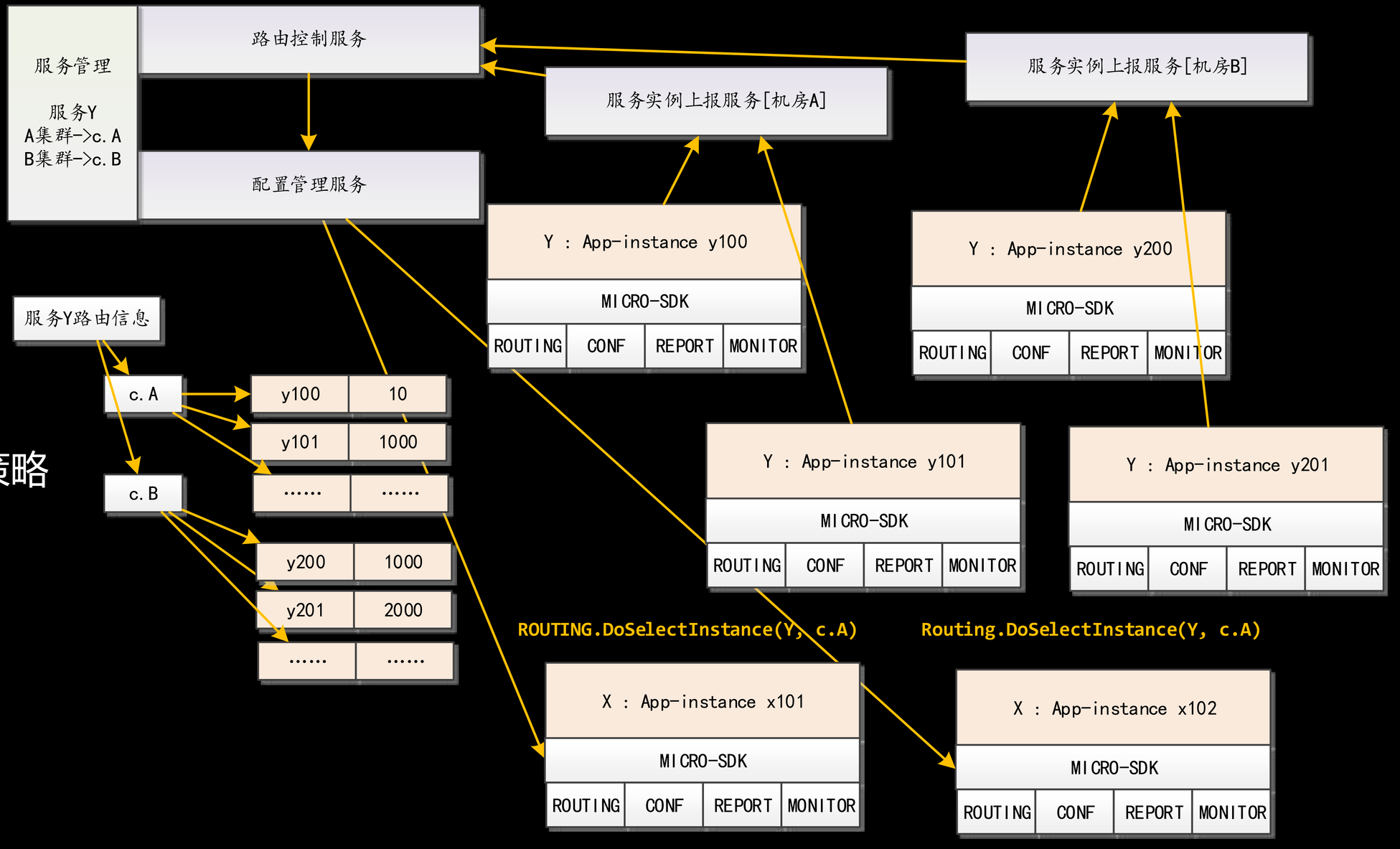
JAVA API

```
ConditionJudgeOutput enableNewFeature =ConditionJudge( “ENABLE_NEW_FEATURE” )
        .CLUSTER(getLocalCluster()).MACHINE(getLocalMachine())
        .API(context.api()).ID(context.id()).judge();

if (enableNewFeature.isHit()) {
    newFeature();
} else {
    oldFeature();
}
```

微服务治理——路由控制

- 生成路由
  - 负载均衡
  - 熔断策略
- 高级策略
  - 多集群混合策略
  - 过载保护

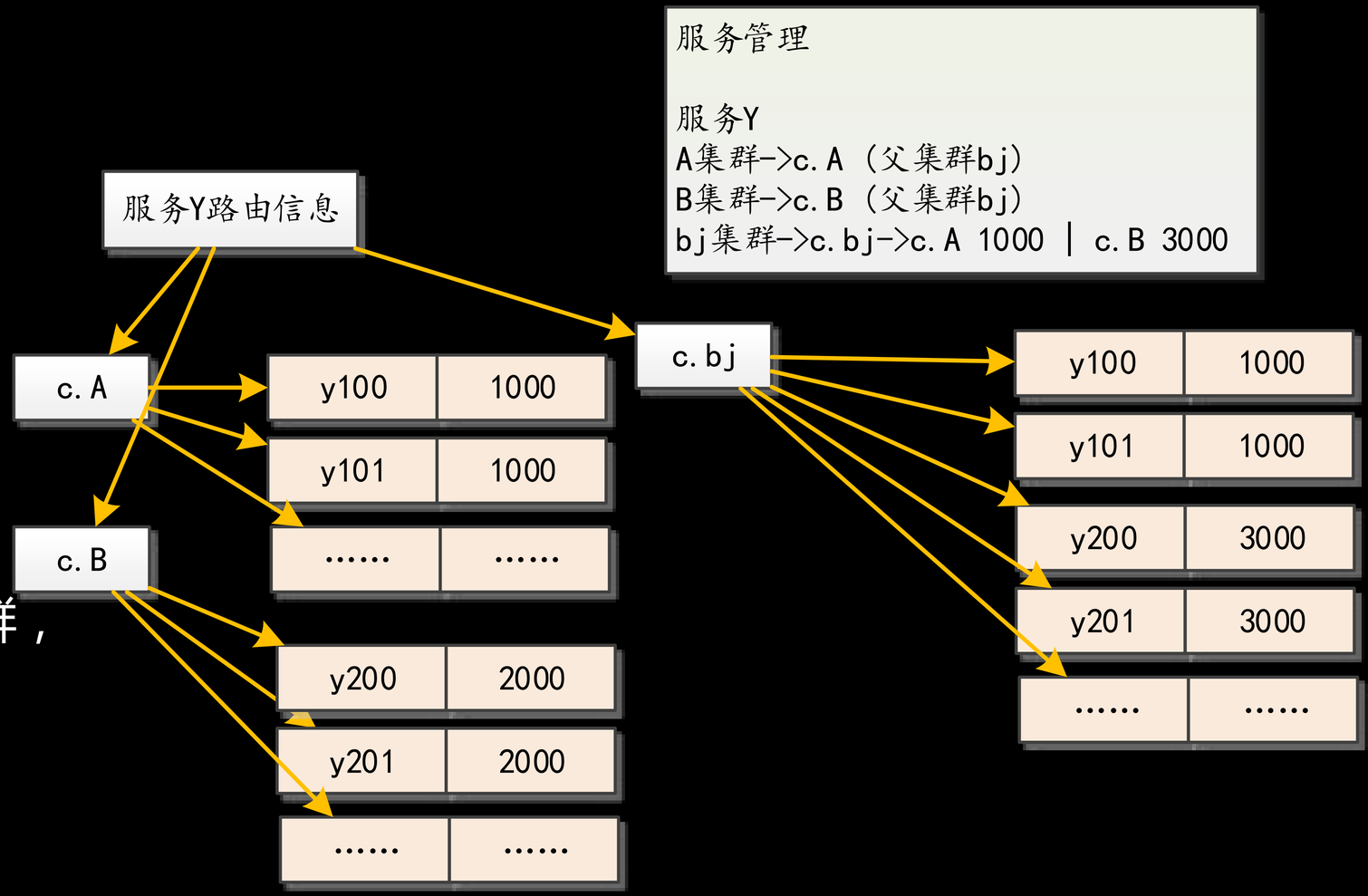


微服务路由——路由策略

- 负载均衡——服务端权重/客户端自适应?
  - 服务端为主：路由稳定
  - 客户端为主：反应速度快
- 熔断控制——防止雪崩
  - 服务端熔断保护策略：设置路由变化阀值
  - 客户端熔断保护策略：
    - 触发条件：RPC调用异常（错误/超时）结果超过阀值
    - 初步处理：时间窗口临时禁止实例
    - 进一步处理（禁止实例超过比例）：服务降级或拒绝

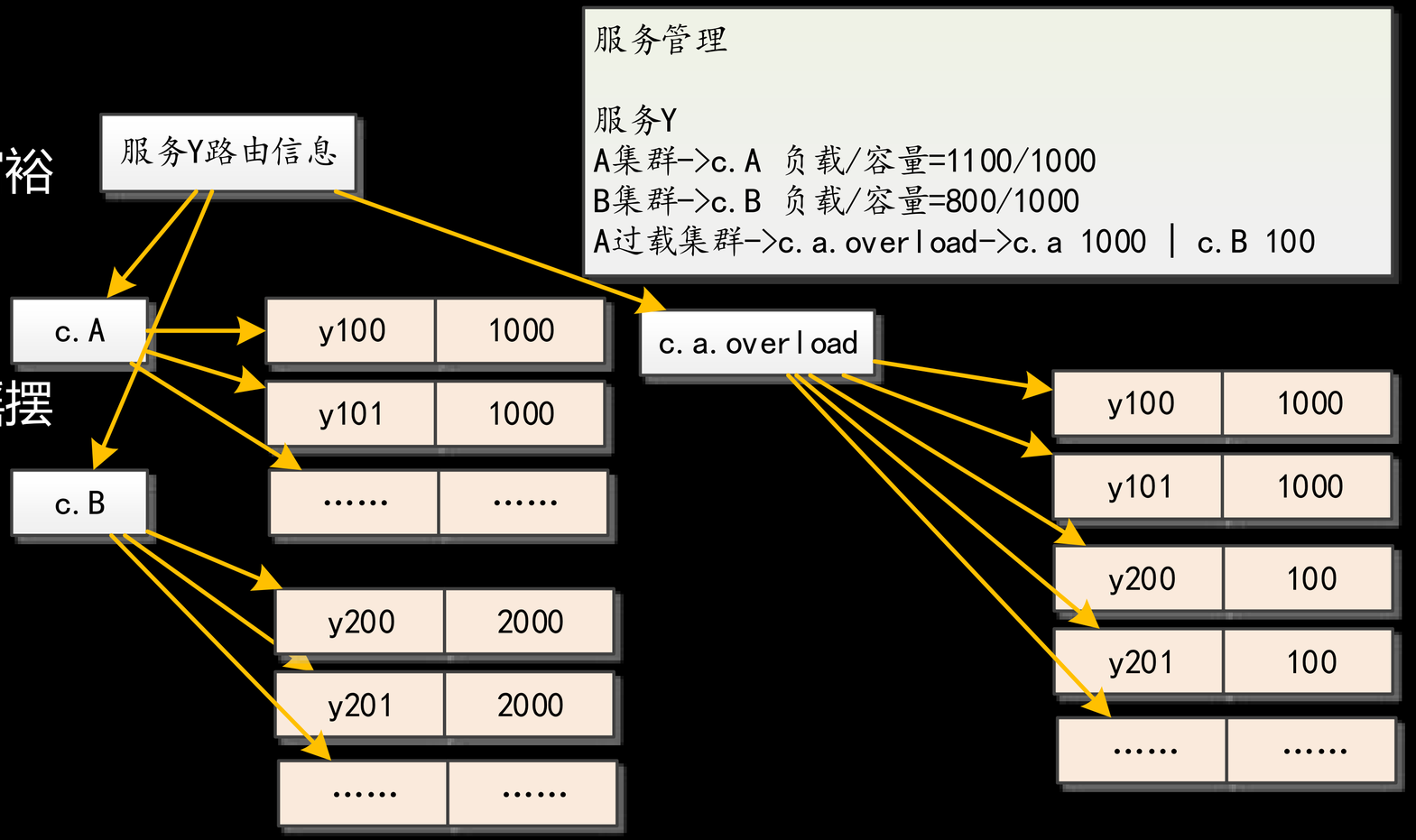
微服务路由——高级路由策略

- 多集群调度：多集群权重混合
  - bj集群是按照A/B两个集群1：3流量来配比
- 容灾策略：兄弟集群成为备选
  - bj集群是A/B两个集群的父亲集群，若A集群访问出现问题，回退访问bj集群下面的B集群



微服务路由——高级路由调度

- 过载保护
  - 从负载过高的集群向尚且有富裕容量的集群调度流量
  - 服务整体过载不进行调度
  - 负载自适应策略：防止流量摇摆



微服务监控——链路监控

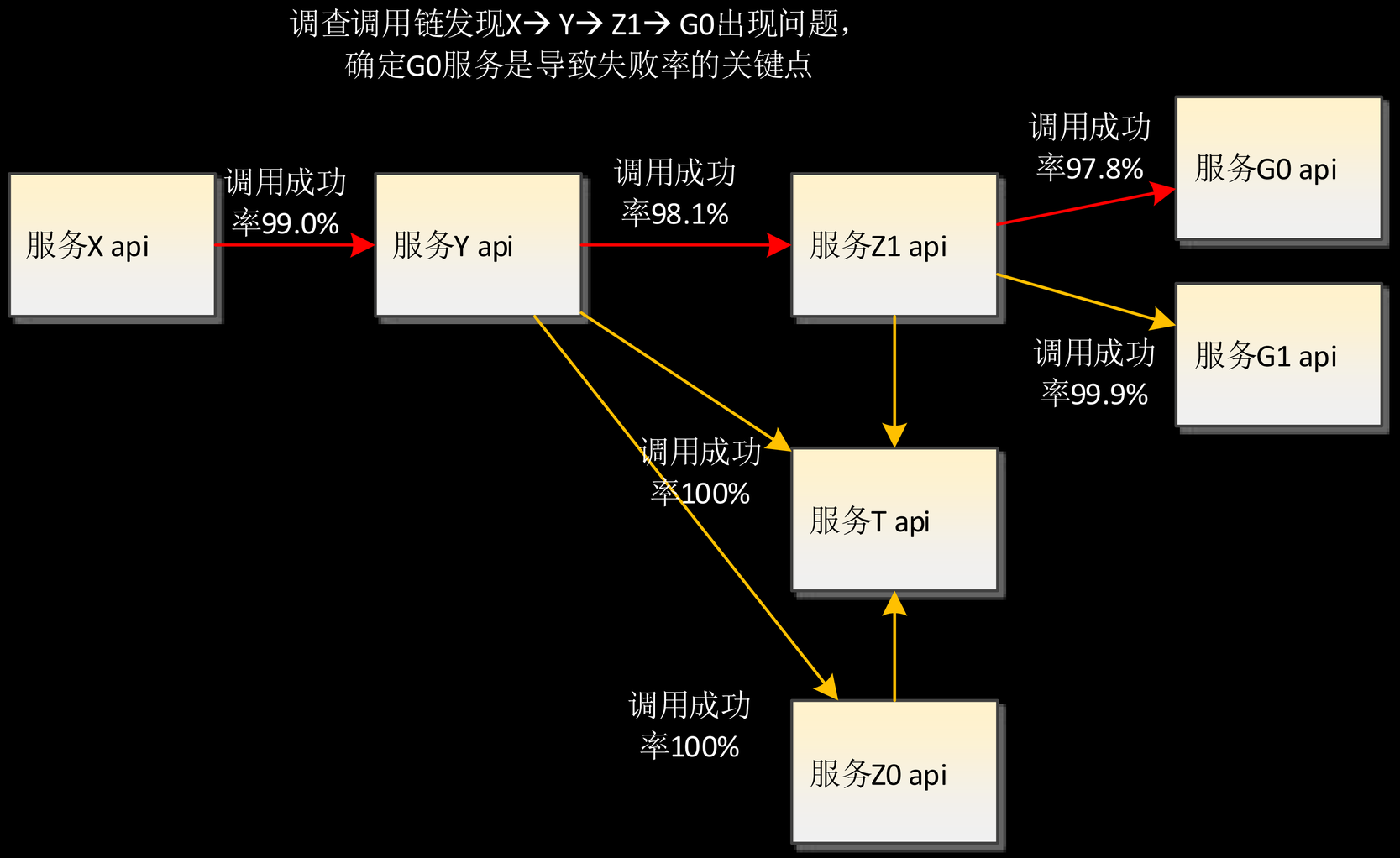
- 传统监控的问题
  - 报警模糊
  - 定位问题慢
- 目标
  - 迅速定位排查问题
- 链路监控的实现
  - 链路基本调用情况
  - 整合请求链调用系统

微服务监控——监控思路

- 记录上下游服务的ip，调用的api，延时，成功失败情况，错误代码和错误的id
- 提供api，机器，机房，错误代码，时间等多个维度的聚合结果，和报警系统结合，提供更加精准的报警
- 错误id，机器ip和trace调用链系统整合来提供调用链调查

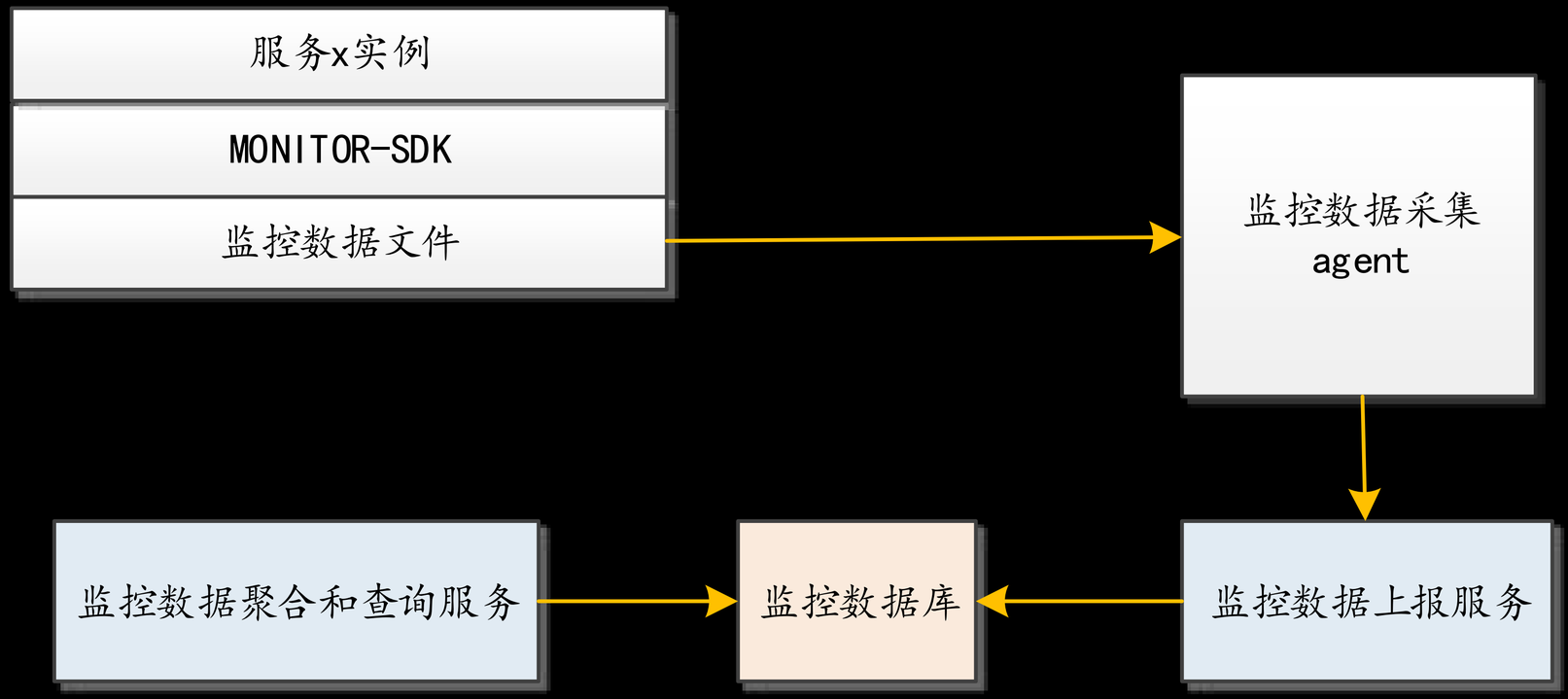
微服务监控——排查过程

- 确定导致服务可用性下降的关键点。
- 确定出现问题的服务，查看错误是否集中的特点。
- 从上游集群中抽出一个出现问题id，查看服务链。



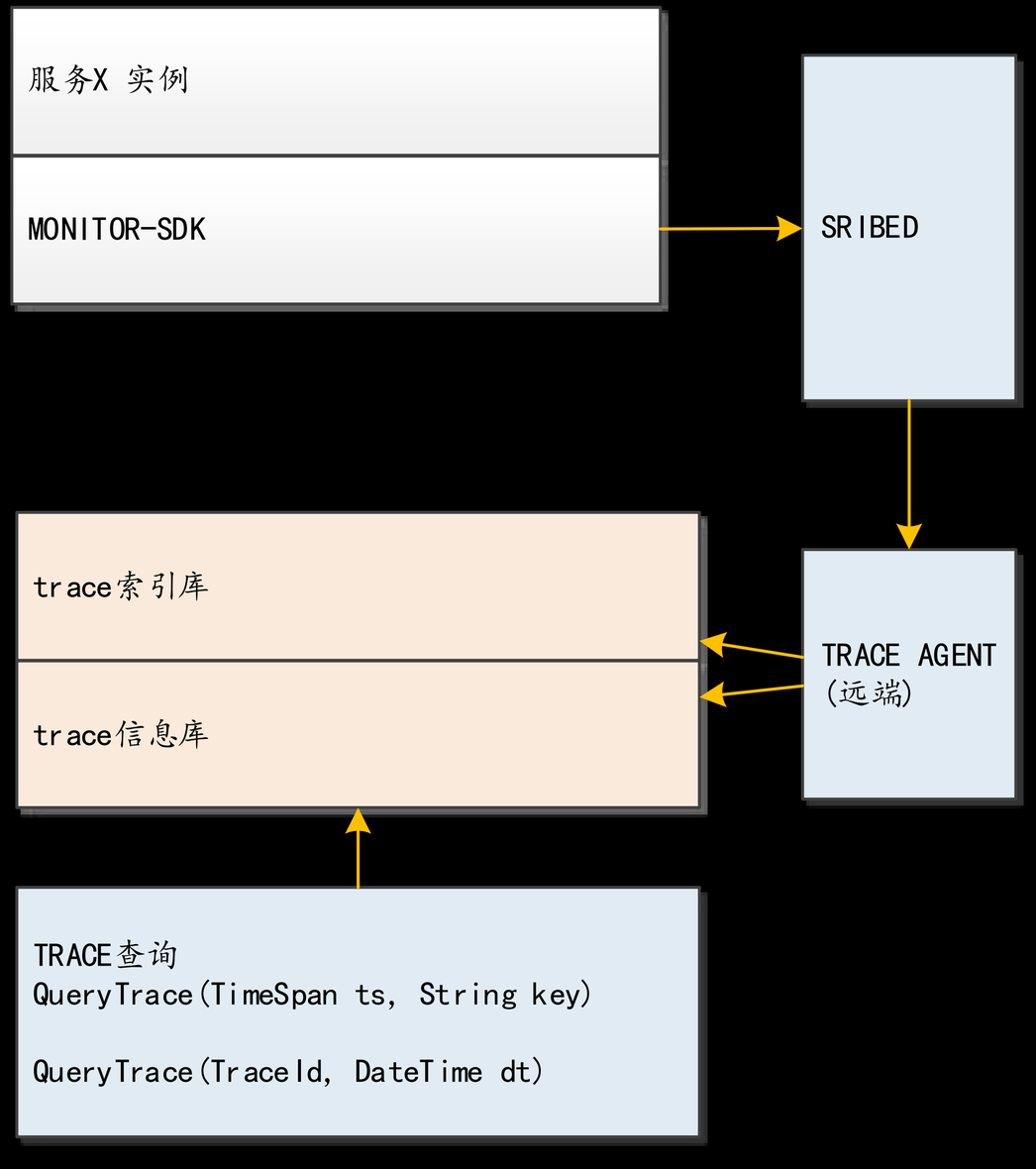
微服务监控——基本调用情况采集和监控

- 程序中SDK分钟级聚合调用情况
- 延时采用平均数，90分位数采用预估
- 提供小时/天时间维度聚合并提供统计报告



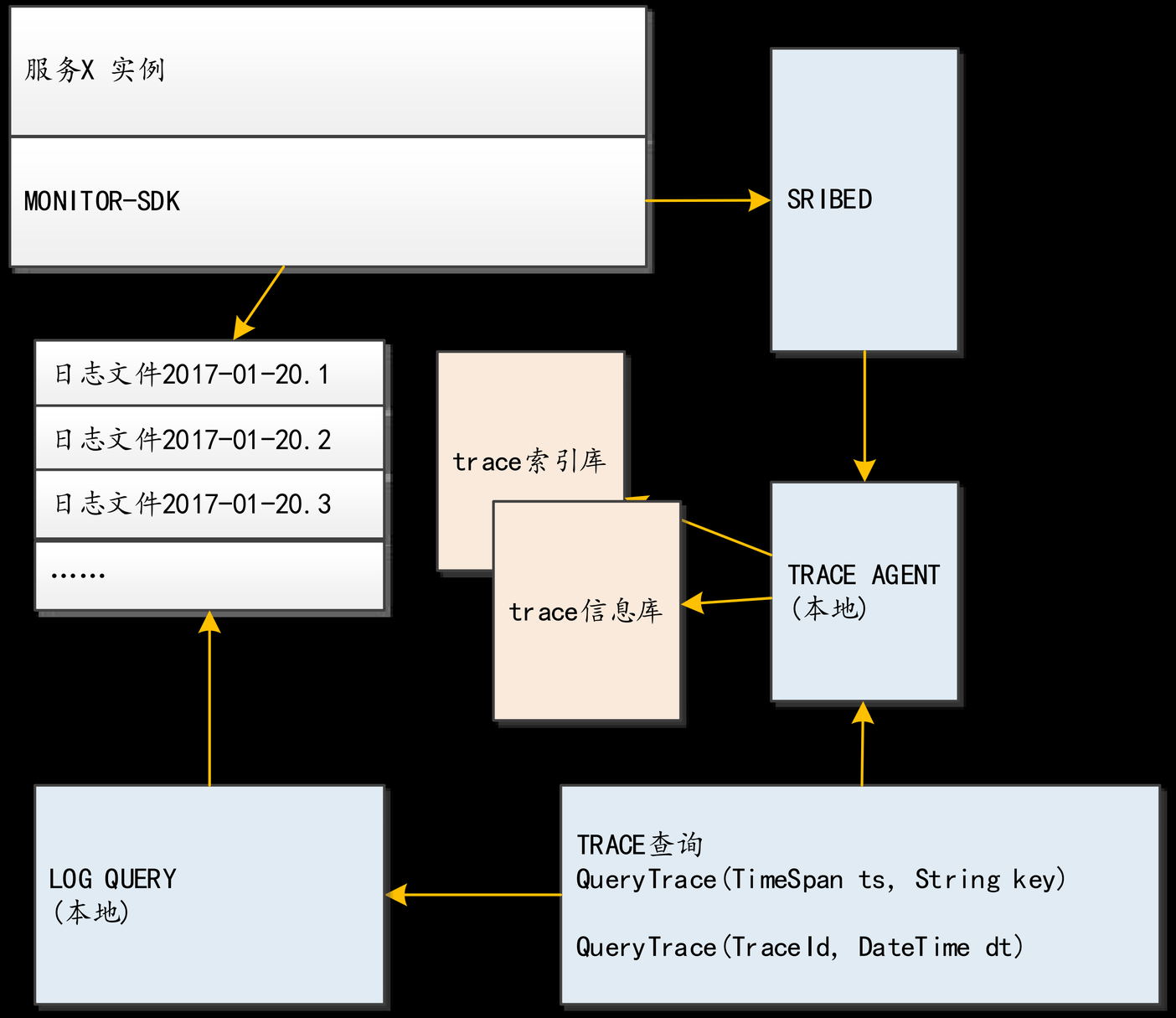
微服务监控——链路调用情况采集和监控

- 问题调查需要全采样
  - 消耗大量带宽和存储
  - 场景的时效性
  - 考虑结合日志

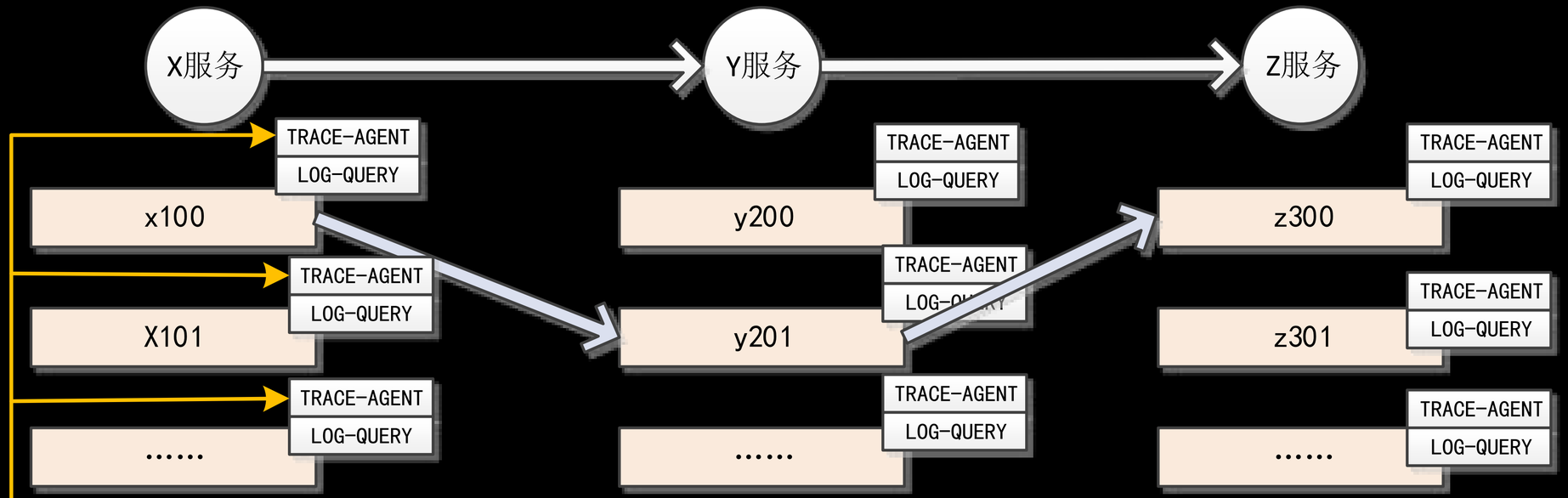


微服务监控——链路调用情况采集和监控

- 记录trace信息的上下游服务ip，关键的ID，调用发生的时间和日志的位置。
- trace存储不再集中。



微服务监控——链路查询示例



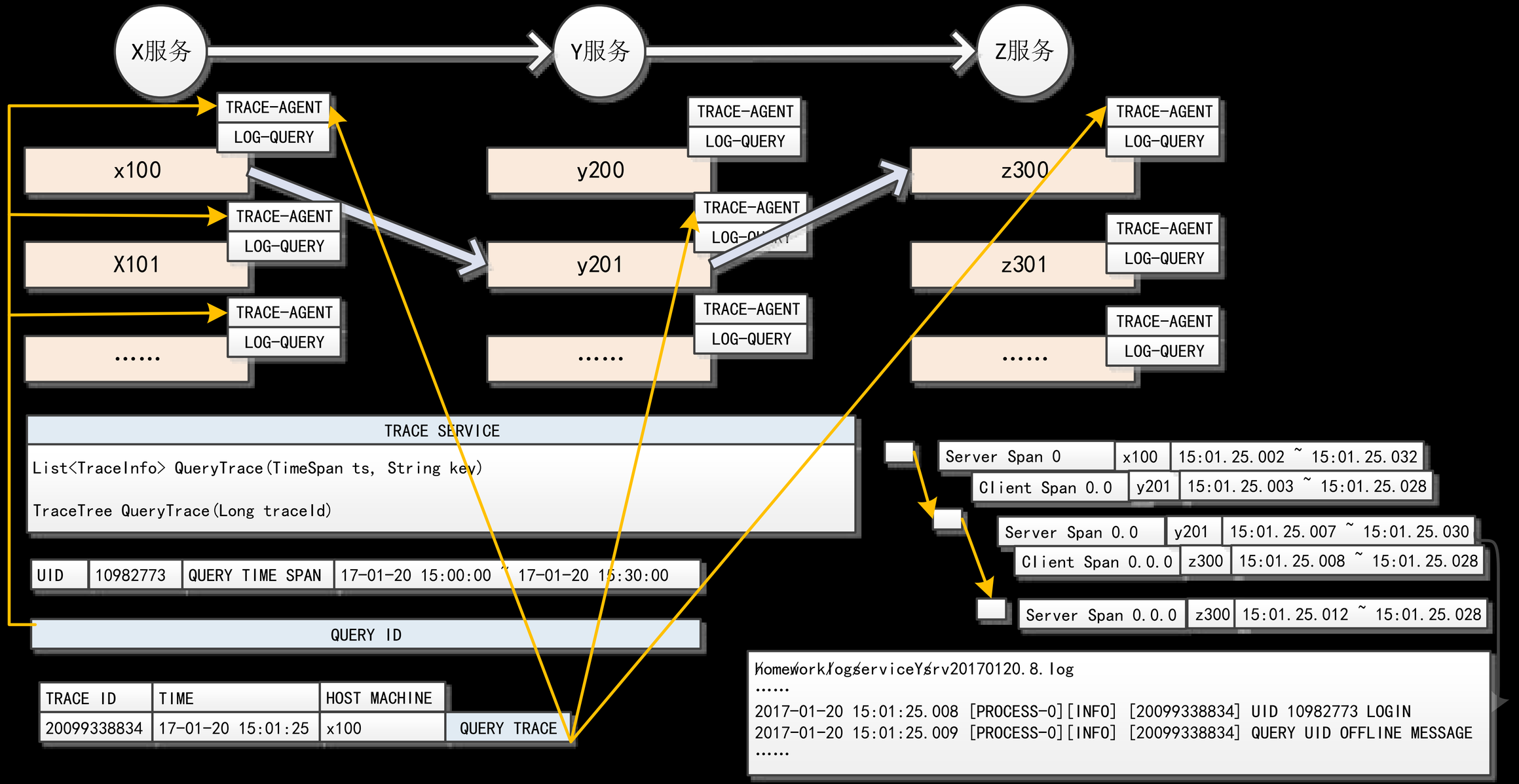
TRACE SERVICE
List<TraceInfo> QueryTrace(TimeSpan ts, String key)
TraceTree QueryTrace(Long traceId)

UID	10982773	QUERY TIME SPAN	17-01-20 15:00:00 ~ 17-01-20 15:30:00

QUERY ID

TRACE ID	TIME	HOST MACHINE	
20099338834	17-01-20 15:01:25	x100	QUERY TRACE

微服务监控——链路查询示例

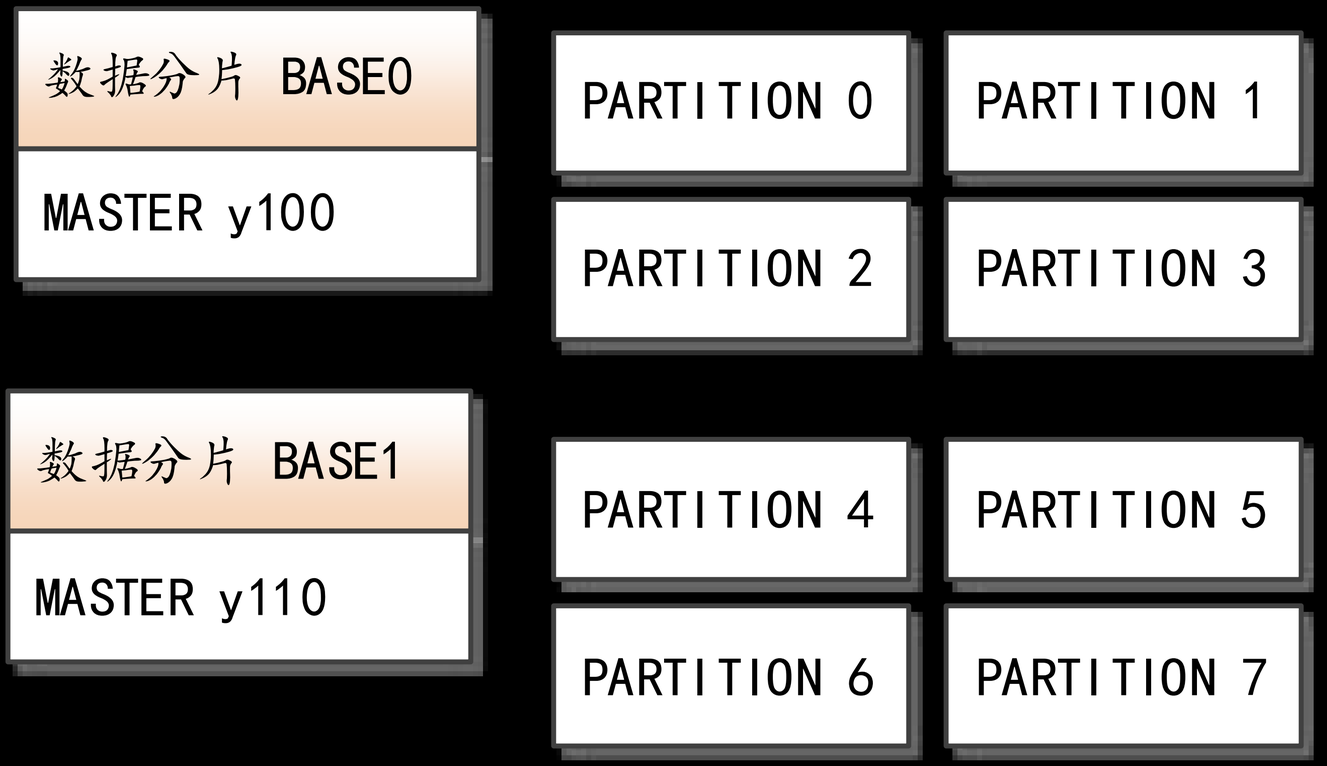


有状态服务的微服务化改造

- 纳入路由系统统一管理
- 考虑数据的迁移场景
- 典型的状态服务类型
  - 半持久化数据
  - 持久化数据

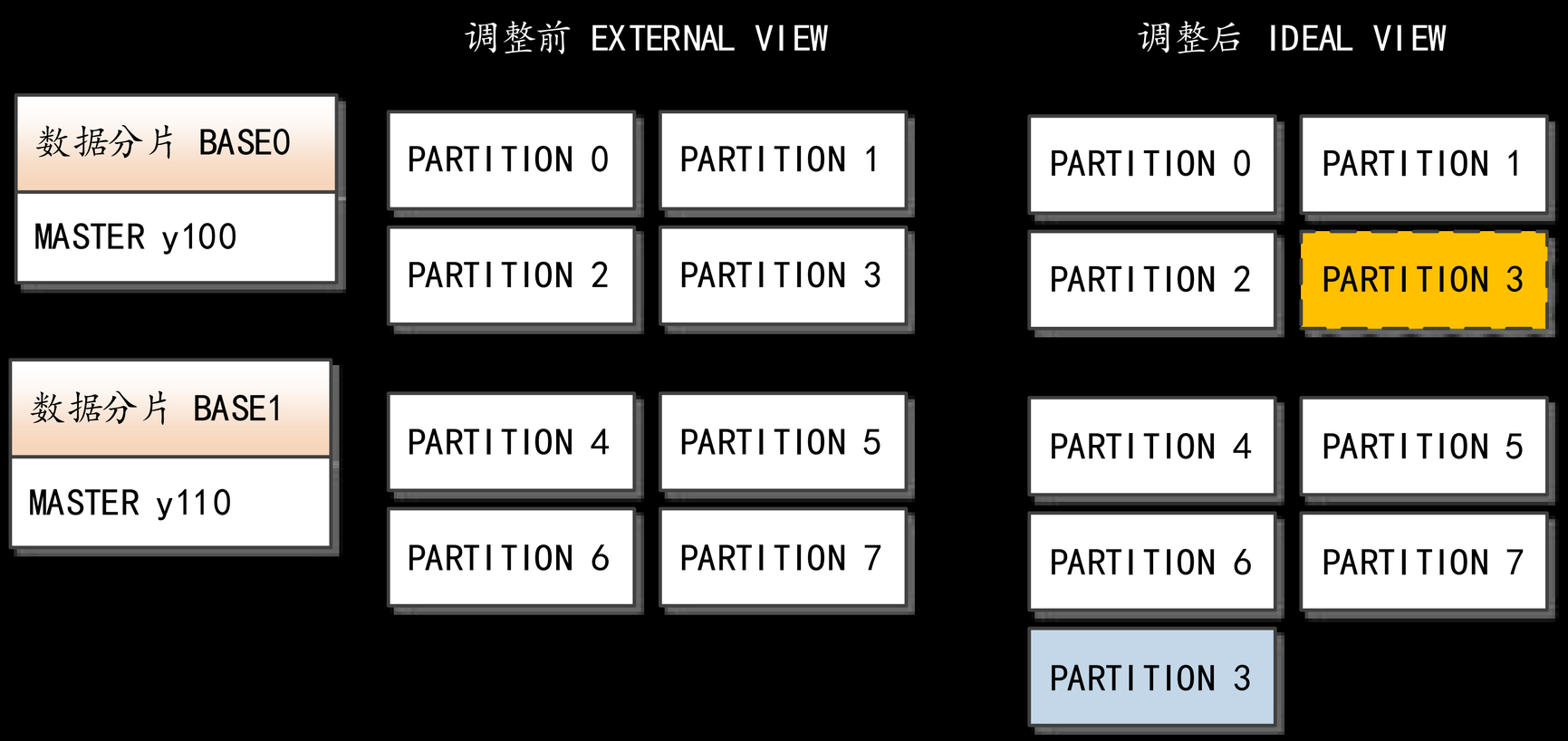
有状态vs无状态

- 集群路由信息包含数据分片。查找路由首先确定数据分片，然后确定对应的访问实例。
- 每个数据分片可能存在主/备分区。
- 无状态是一种特殊的有状态服务（仅一个数据分区）



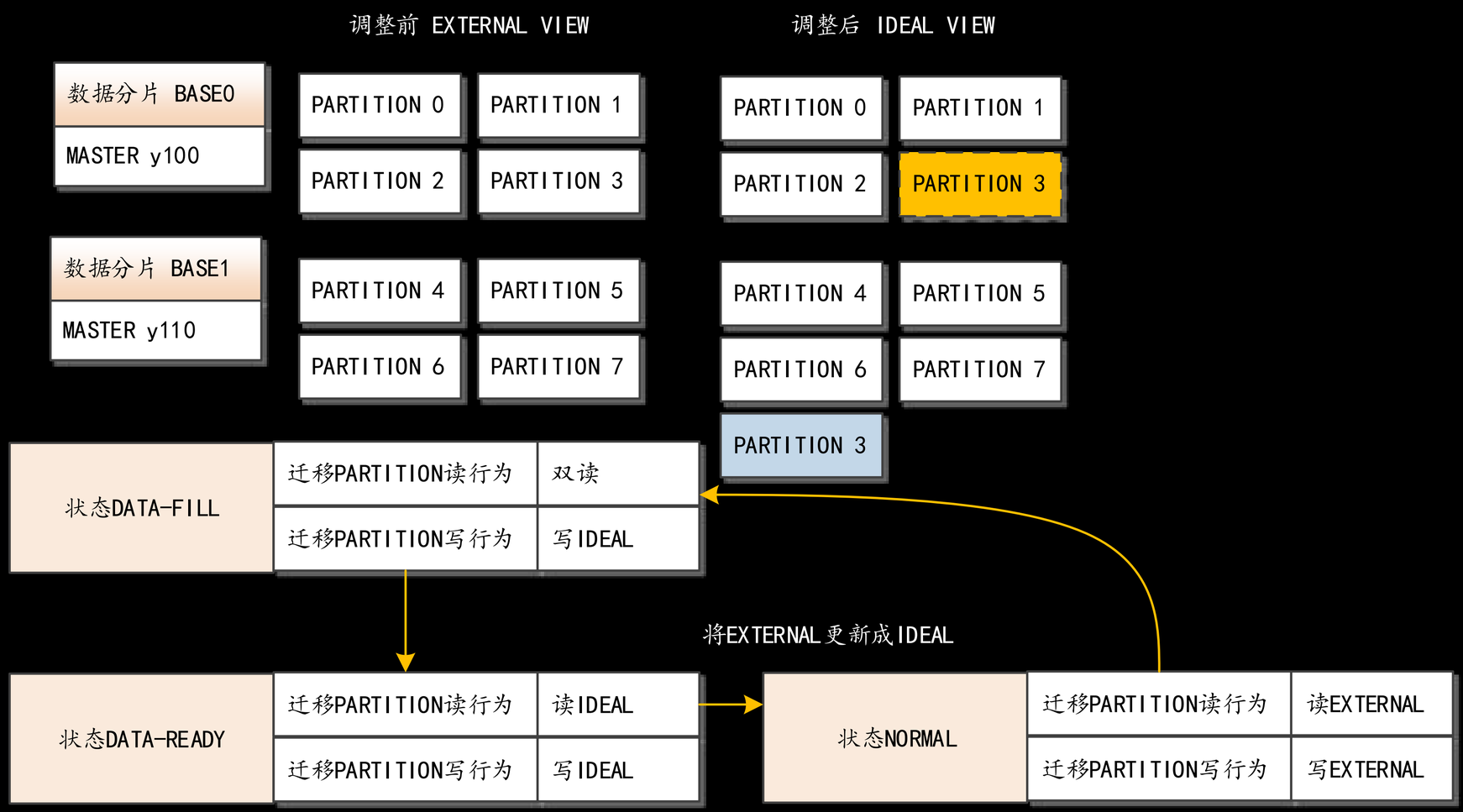
有状态服务的数据迁移

- 定义数据迁移前和迁移后的两张分片信息
- 定义迁移状态来控制数据读写



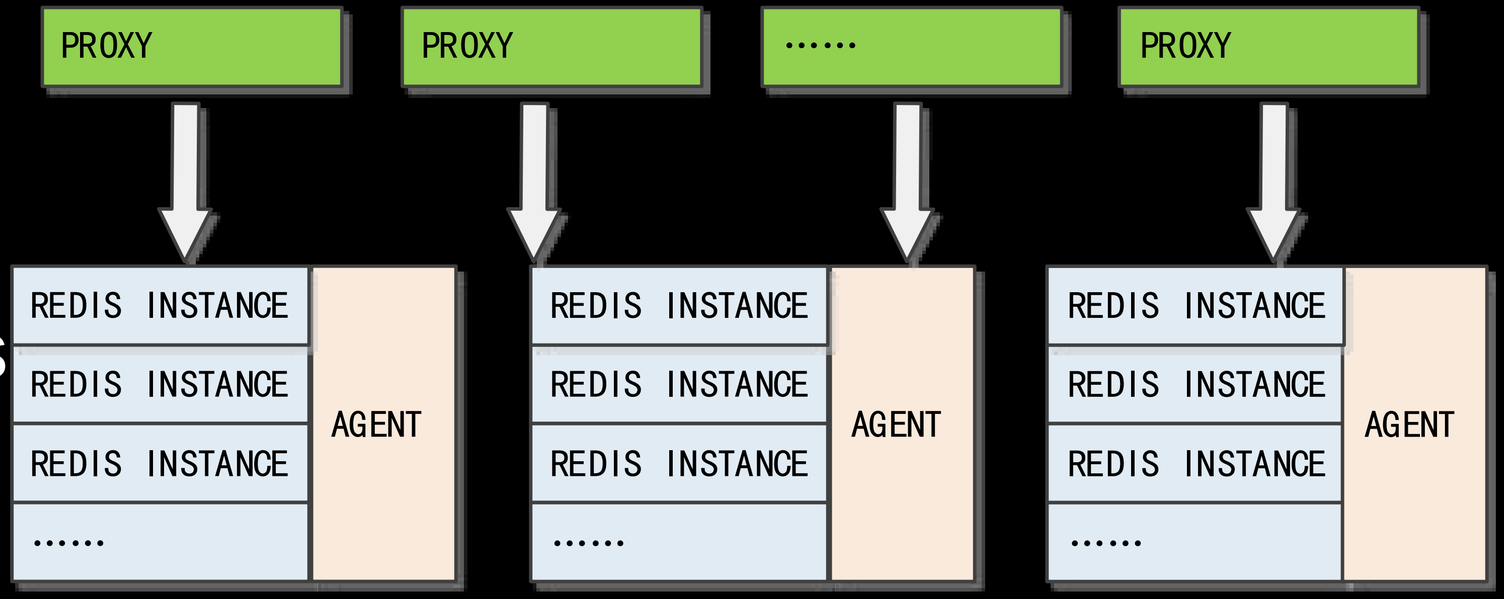
有状态服务——半持久化模式——登录状态

- 利用用户平均30分钟的登录时长的经验值来简化迁移过程
- 定义两个迁移状态和迁移数据的读写行为

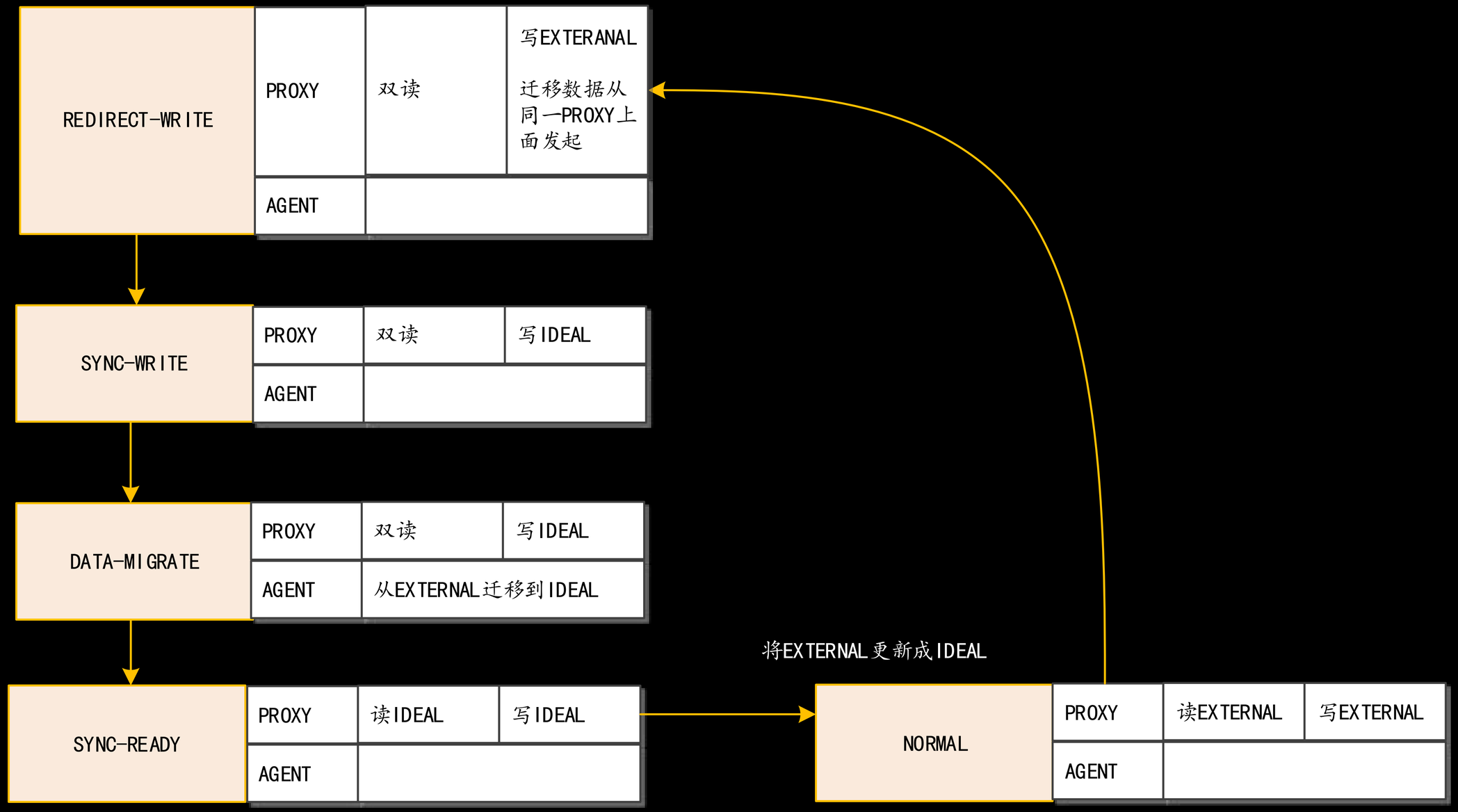


有状态服务——持久化模式——离线消息系统

- 基于redis，添加proxy层用于屏蔽数据迁移
- proxy负责读取状态和控制读写，直接和redis实例通讯
- agent负责起停redis实例，上报实例状态和对数据做dump迁移



有状态服务——持久化模式——离线消息系统



总结和感悟

- 坏消息传播的慢
  - 熔断、限流、灰度
- 坏消息知道的快
  - 链路监控、链路定位
- 自动化本身有代价
  - 自适应策略
- 微服务体系建设目标
  - 稳定、便捷、高效

关注QCon微信公众号，
获得更多干货！

Thanks!

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

主办方 Geekbang. 极客邦科技 InfoQ