



微服务架构

在二手交易平台（转转）中的实践



孙玄

关于我

58

✓ **QCon** 全球软件开发大会 技术委员会

DTCC
2016中国数据库技术大会
DATABASE TECHNOLOGY CONFERENCE CHINA 2016

WOT
World Of Tech

✓ **SDCC** 中国软件开发大会
Software Developer Conference China

TOP1
技术型企业案例研究智库

Strata+Hadoop
WORLD

✓ **PROGRAMMER** 程序员 算法

✓ 百度高级工程

✓ 毕业于浙江大

✓ 代表公司多次

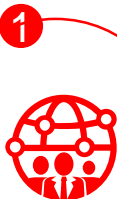




- 什么是微服务架构

In short, the microservice architectural style is an approach to developing a single application as a **suite of small services**, each **running in its own process** and communicating with lightweight mechanisms, often an HTTP resource API. These services are **built around business capabilities** and **independently deployable** by fully automated deployment machinery. There is a **bare minimum of centralized management** of these services, which may be written in different programming languages and use different data storage technologies.

-- James Lewis and Martin Fowler





微服务架构



微服务粒度



独立进程



围绕业务建模



轻量级通信



去中心化管理



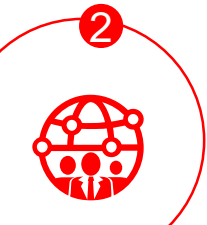
使用原因-转转是什么



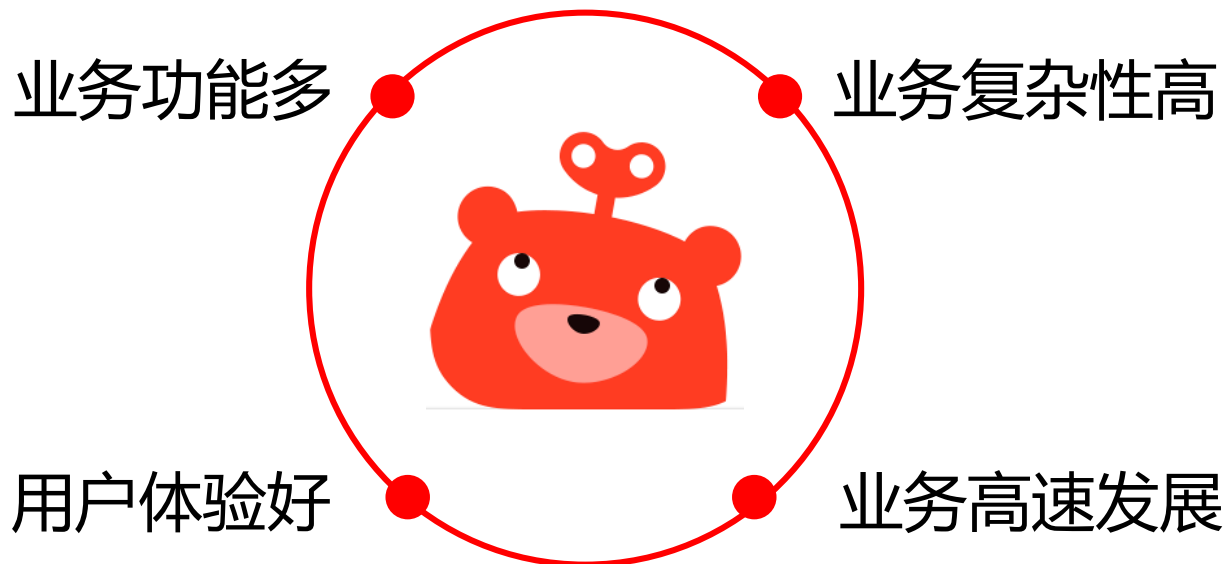
使用原因-二手交易平台功能



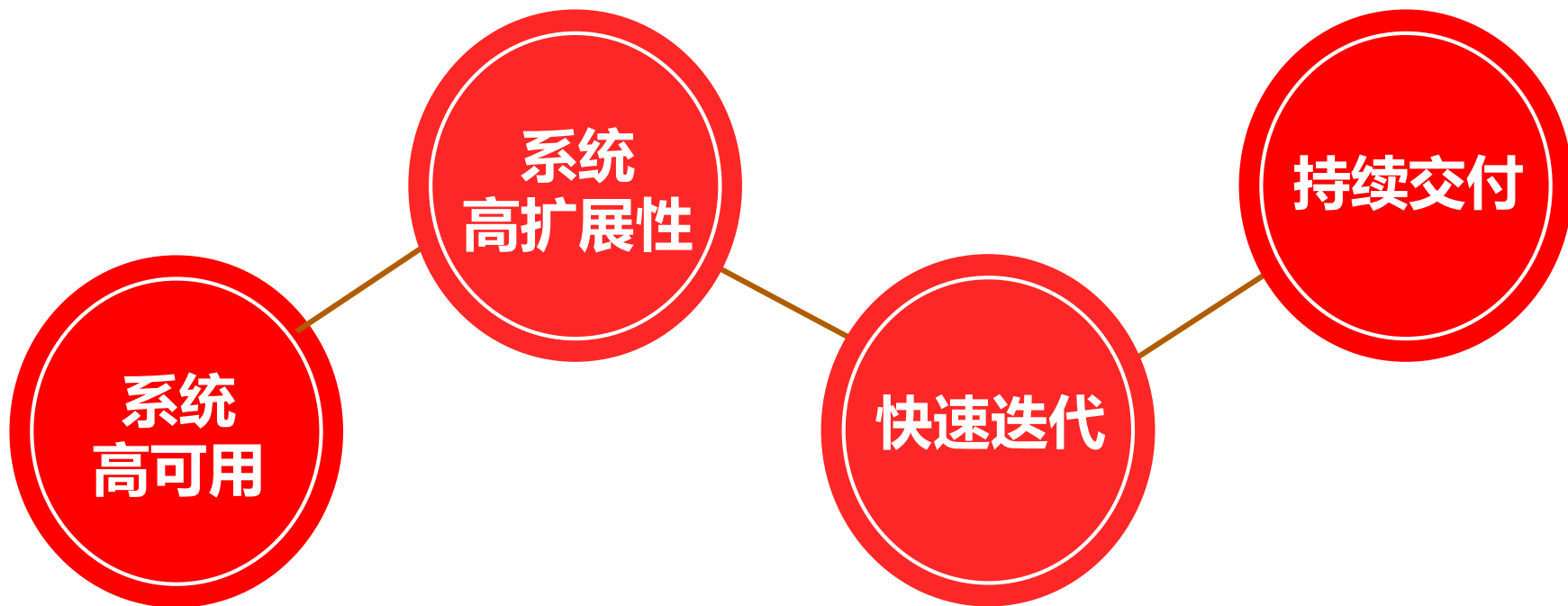
- ✓ 发布商品
- ✓ 分类搜索
- ✓ 关键词搜索
- ✓ 推荐商品
- ✓ 消息中心
 - 私信、留言
- ✓ 个人中心等等



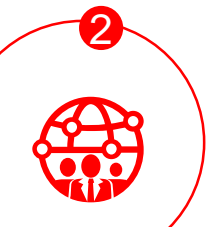
使用原因



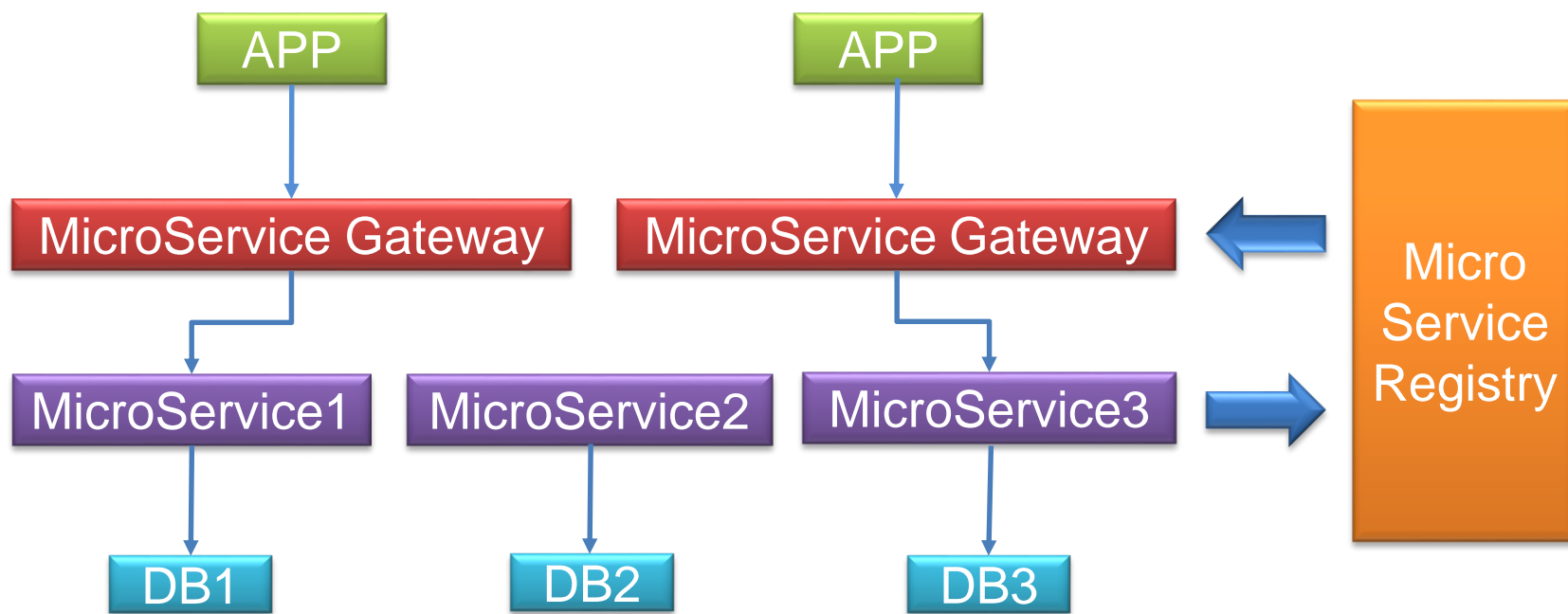
微服务架构特点&二手交易平台特点



转转使用微服务架构比较合适



微服务架构





整体设计

- ✓ 水平分层
- ✓ 垂直业务拆分



每层设计

- ✓ 微服务-业务单元垂直拆分
- ✓ 无状态化
- ✓ 独立进程、部署、运维

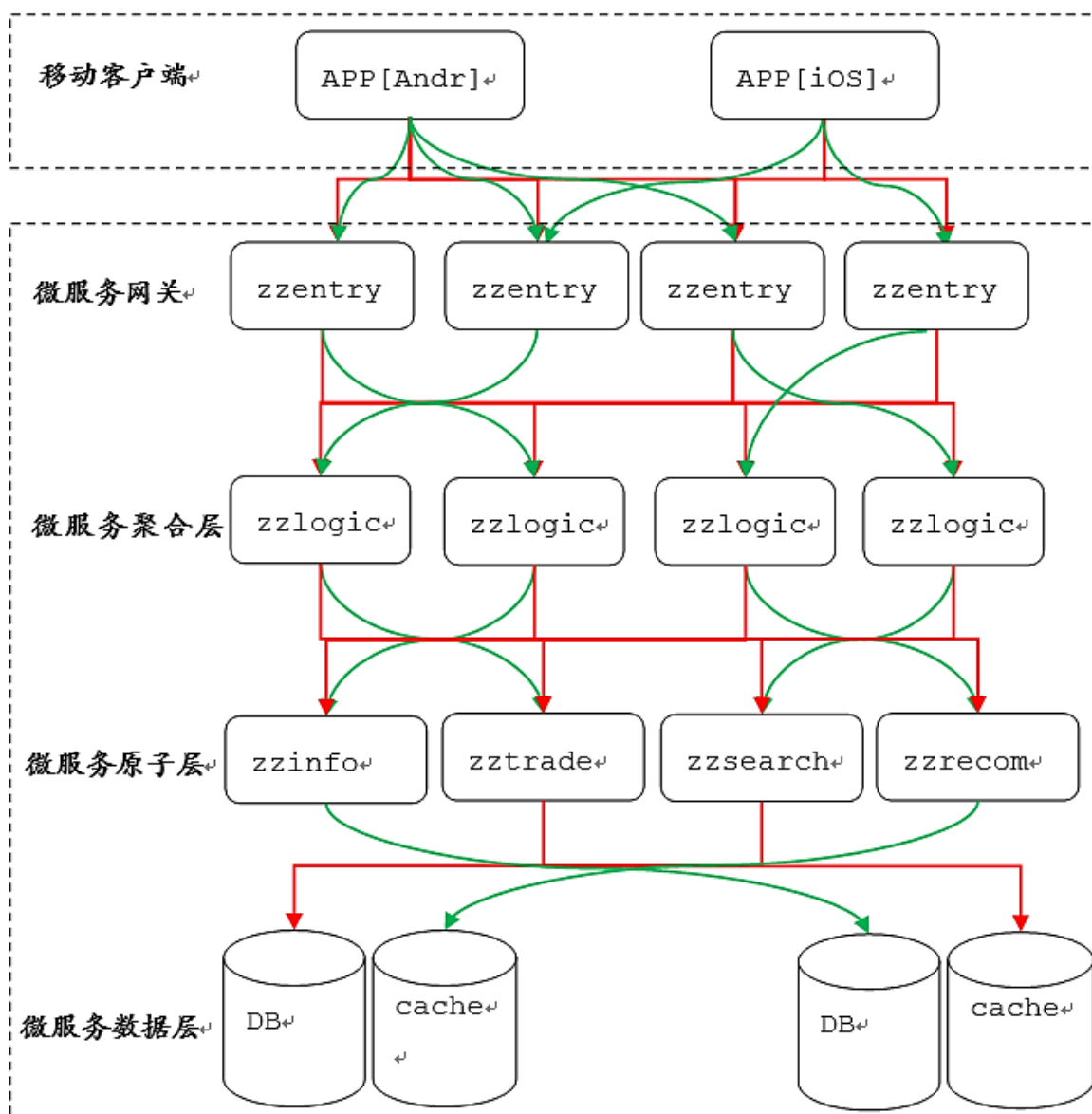


高可用

- ✓ 冗余
- ✓ 自动恢复

演进-总体架构设计

- ✓ 微服务网关
- ✓ 微服务聚合层
- ✓ 微服务原子层
- ✓ 微服务数据层
- ✓ 轻量级通信
 - HTTP
 - RPC
- ✓ 去中心化管理
 - 开发语言-java
- ✓ 微服务注册
- ✓ 微服务发现

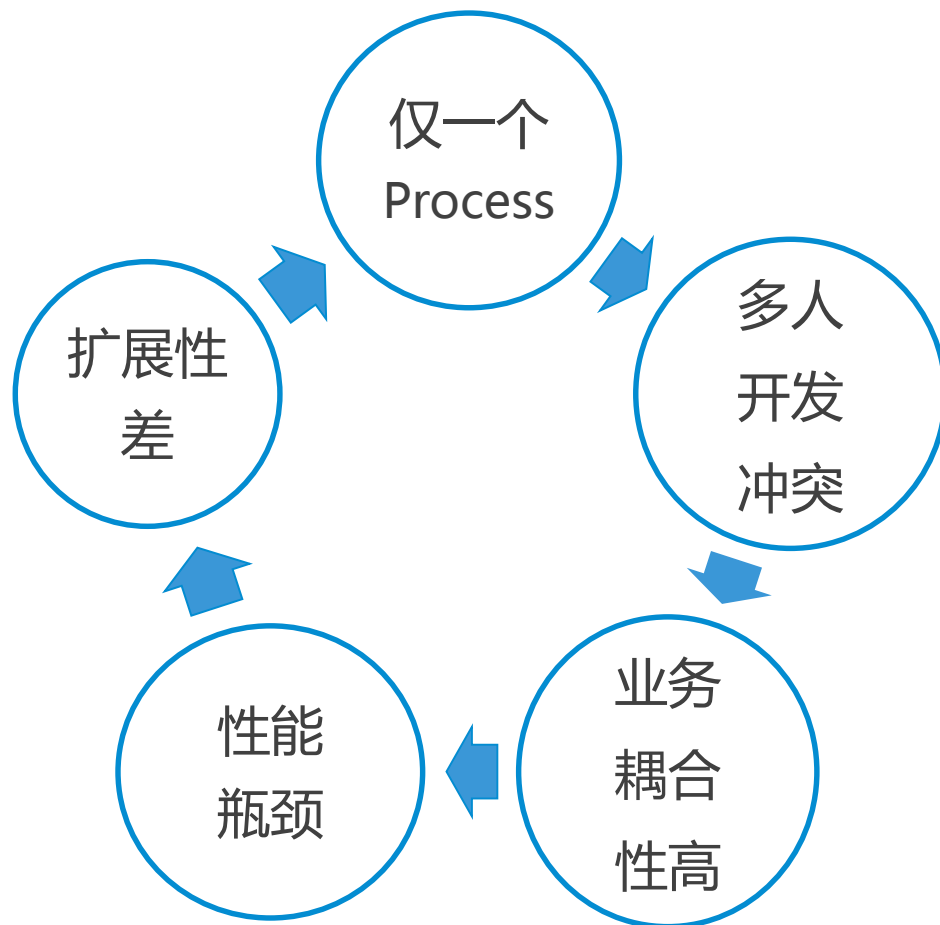


二手交易特点

- ✓ 业务发展快
- ✓ 业务越来越复杂

微服务聚合层

- ✓ 开发瓶颈集中
-微服务聚合层



单微服务聚合层



多微服务聚合层



业务逻辑拆分

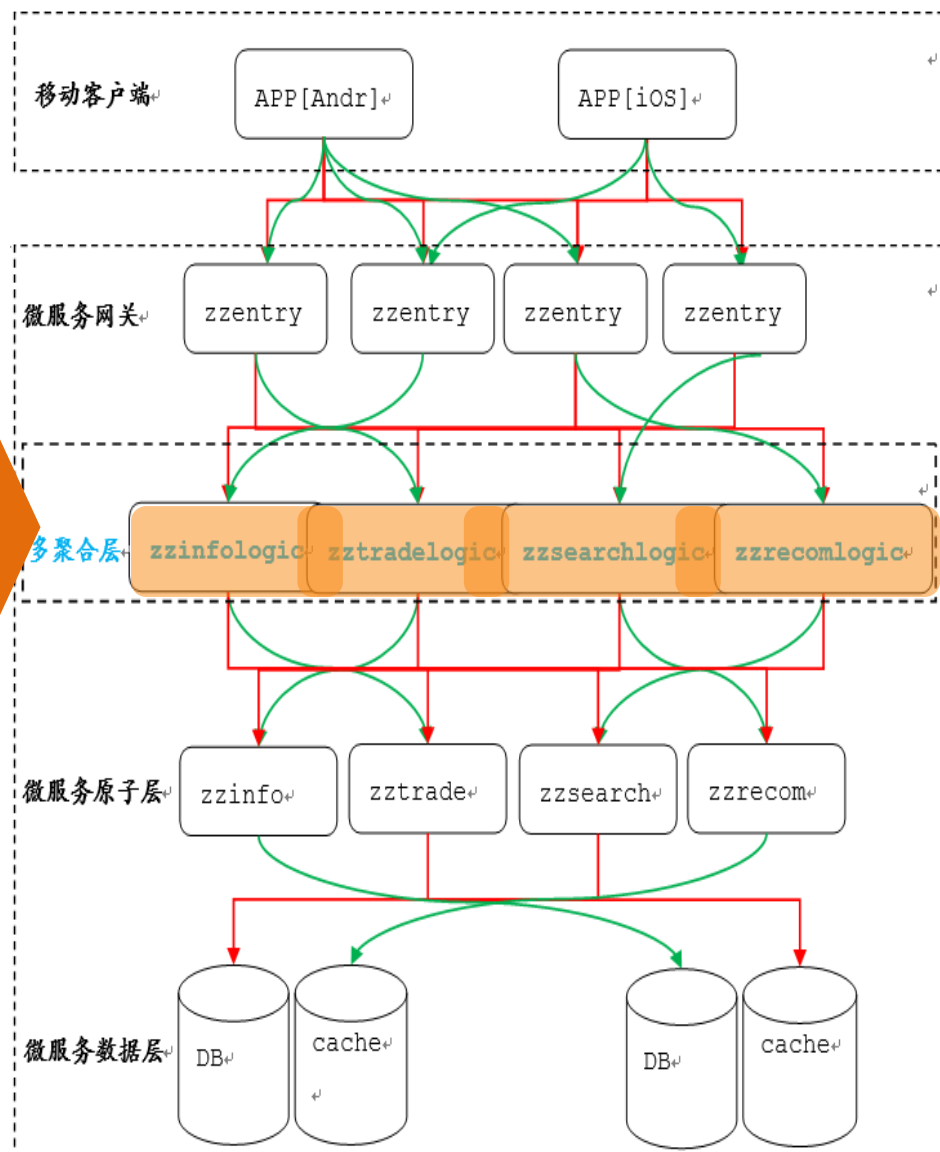
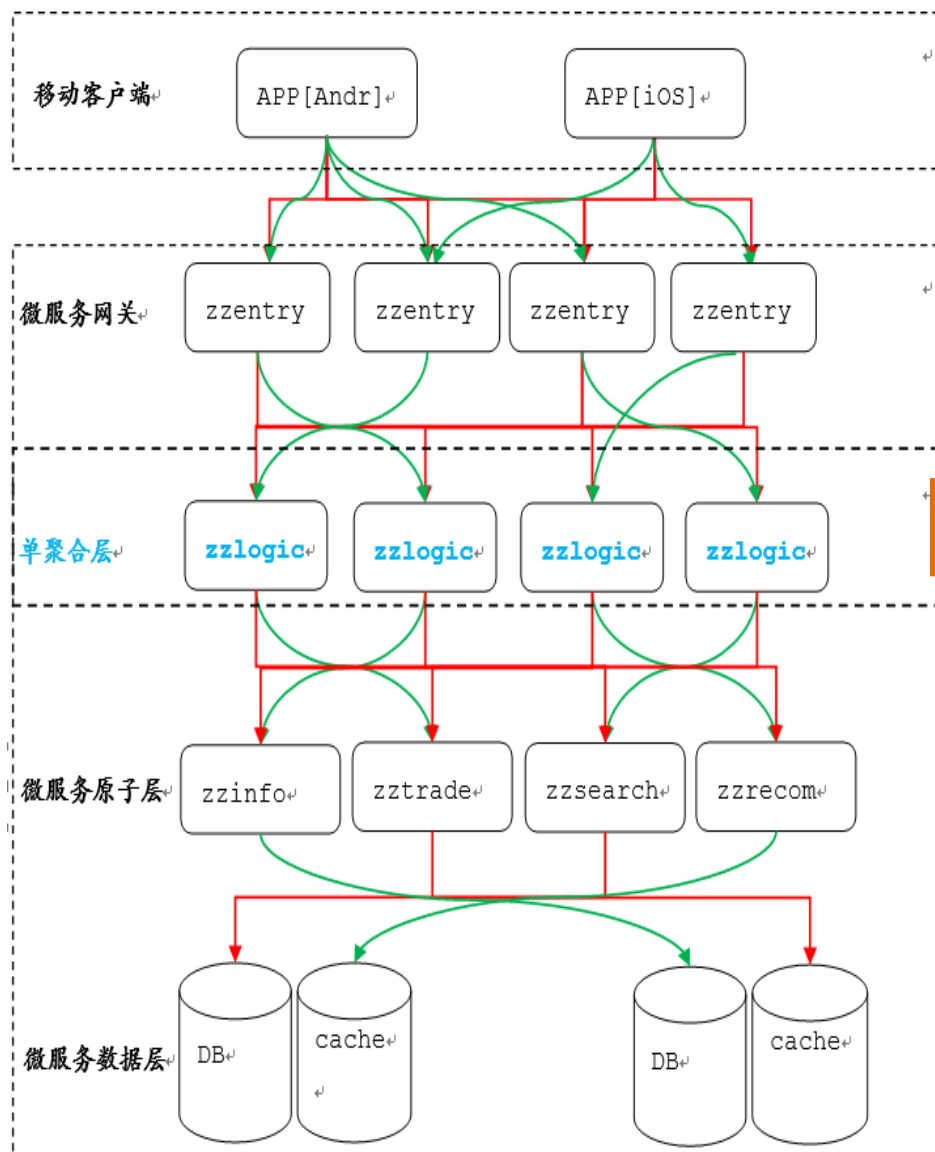


业务物理拆分

转转业务领域模型拆分



演进



演进-多微服务聚合层优点



独立

- ✓ 进程
- ✓ 开发
- ✓ 部署
- ✓ 运维

高效

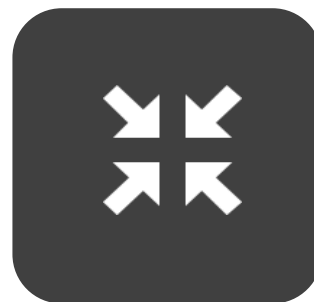
- ✓ 快速迭代
- ✓ 持续交付



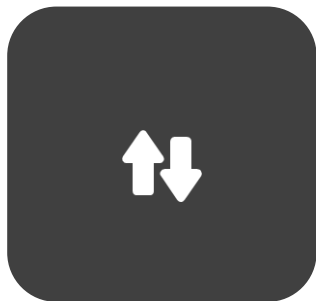
通信协议-轻量级通信协议



REST
✓ HTTP



HAL
✓ REST



RPC
✓ Thrift
✓ gRpc
✓ dubbo



**消息
队列**



通信协议选择-我们的选择

REST API
✓ HTTPS
✓ JSON

转转
微服务网关

```
cmd : zz.com/zz/userquery
method : GET
request : { "uid" : 9664058}
response : {" respCode" : 0,
respData:{ "nickname" : musciml}}
```

RPC
✓ SCF
✓ 私有协议

转转
微服务聚合
层、原子层

服务注册/发现

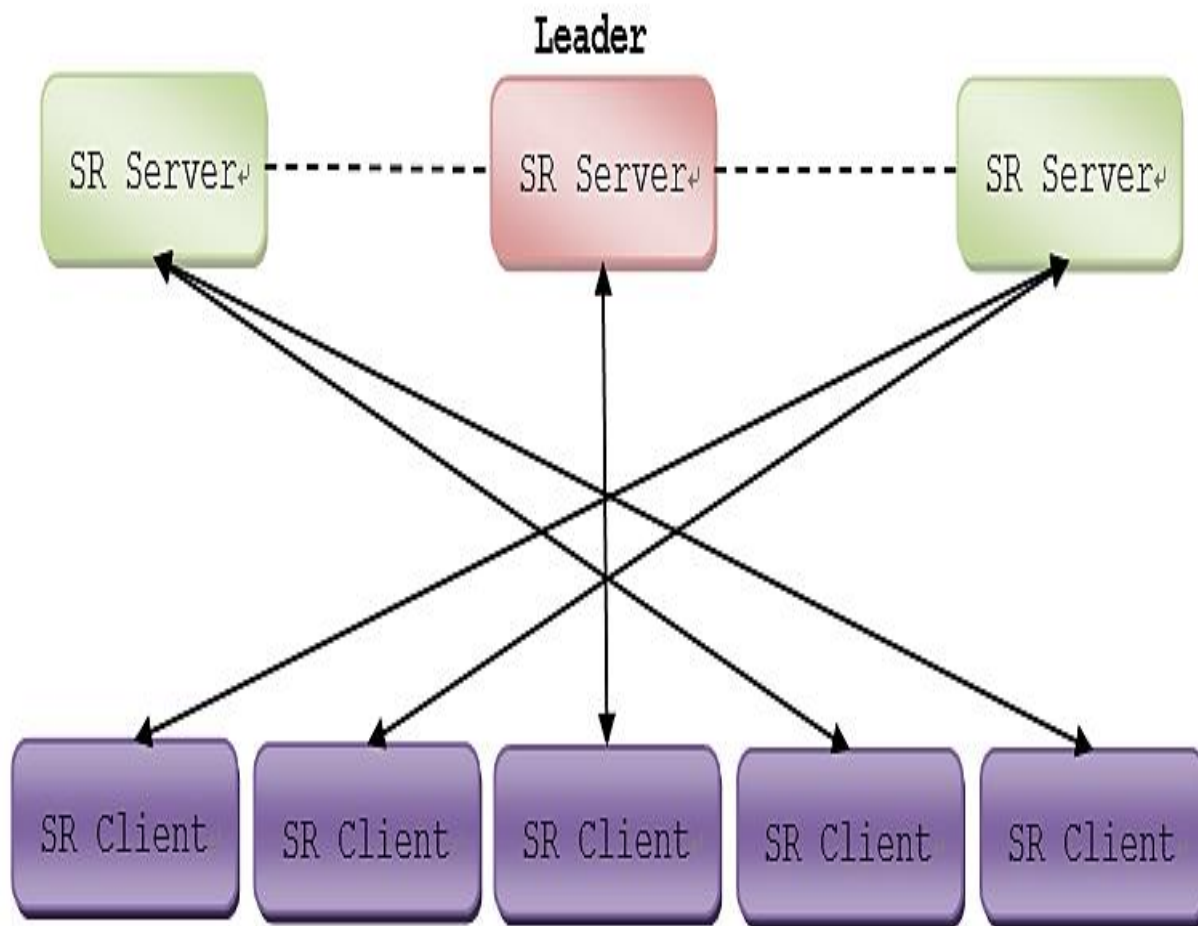


- **微服务注册中心**

- ✓ Host
- ✓ Port
- ✓ Config等

- **微服务发现**

- ✓ 下游发现
- ✓ 负载均衡



服务注册/发现-微服务发现

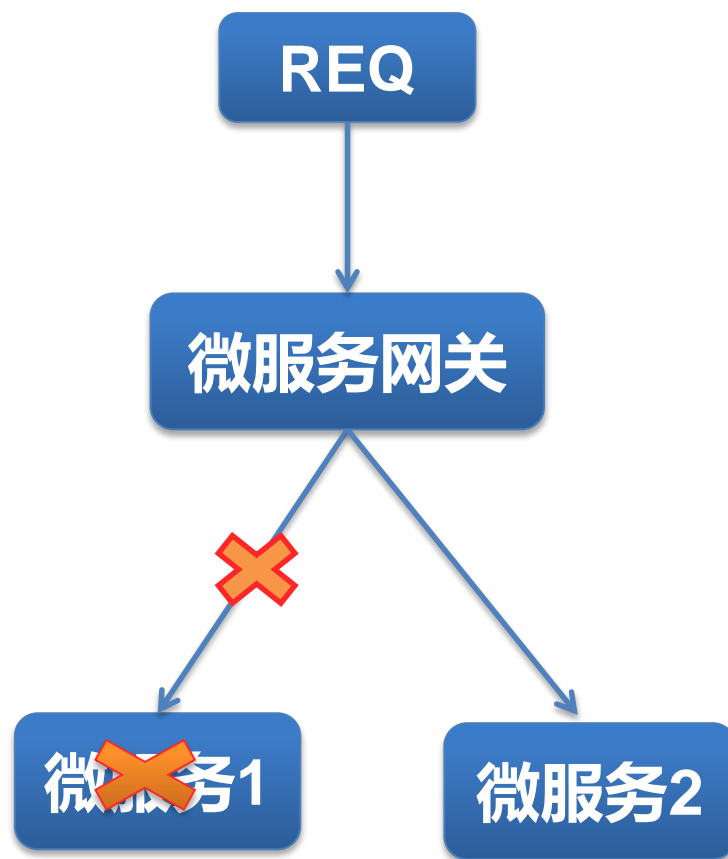


下游发现

负载均衡

✓ Keepalive

✓ Retry



柔性可用实践-为什么需要

流量高峰期

短时请求量大

服务能力有限

性能下降

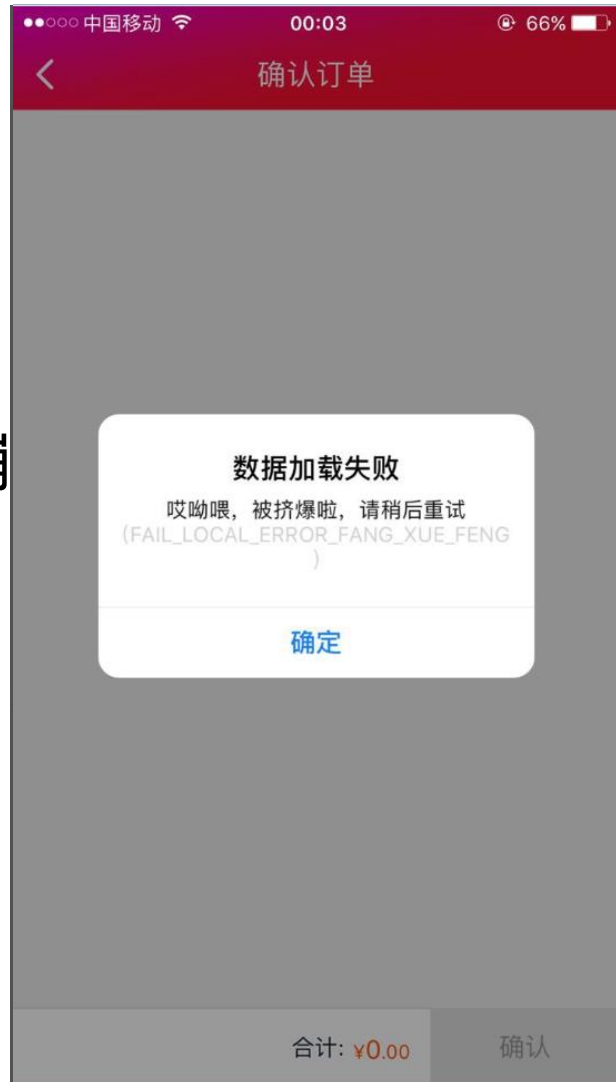
服务宕机

系统雪崩

怎么办




微服务柔性可用



柔性可用实践-柔性设计如何做

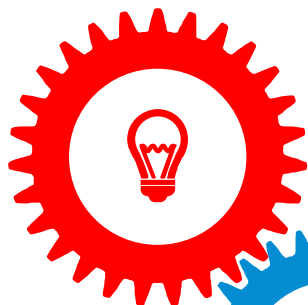


 **目标**：保证核心服务可用；非核心服务弱可用，甚至不可用

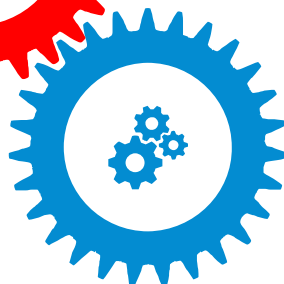


5

柔性可用实践-系统降级



拒绝部分请求



关闭部分服务（业务紧密）



5

1

拒绝部分老请求

- ✓ 减轻微服务请求处理数量
- ✓ 确保“新”请求正常响应
- ✓ RPC队列方式（请求入队、出队时间处理请求时，检查请求在队列请求时间超过一定时间[比如1s]，直接丢弃）

2

优先级请求方式

- ✓ 非核心请求直接丢弃
 - （1）业务紧密
 - （2）转转

3

随机拒绝方式

- ✓ 随机丢弃一定比例请求
- ✓ 网站一会可用，一会不可用

拒绝
部分请求



柔性可用实践-数据层降级

01

更新请求

- ✓ 持久到消息队列
- ✓ 只更新缓存

02

读请求

- ✓ 读缓存

03

数据补齐

- ✓ 消息队列->数据库



柔性可用实践-柔性可用策略



自动
打开

不依赖于人肉

演练

保证线上生效



5

为什么需要监控

监控什么



机器资源

进程状态

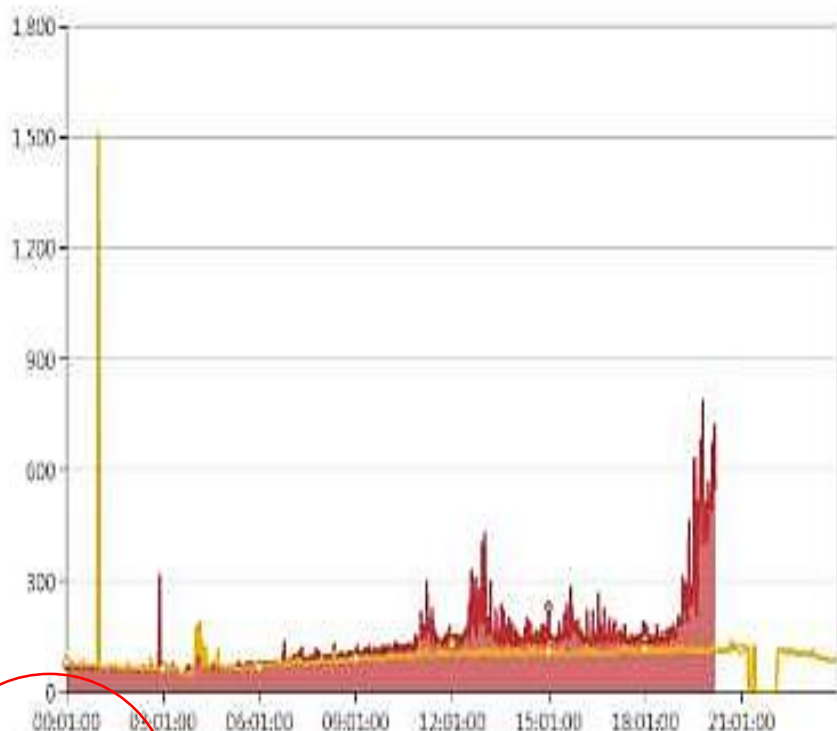


监控手段

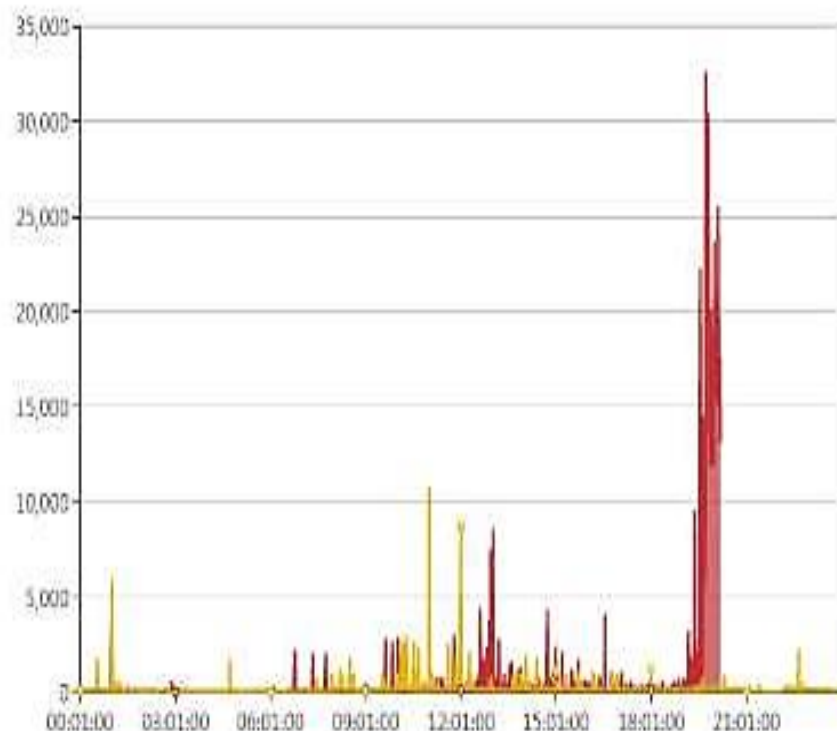


微服务实时监控

请求平均耗时



请求异常条数



服务治理-微服务监控框架



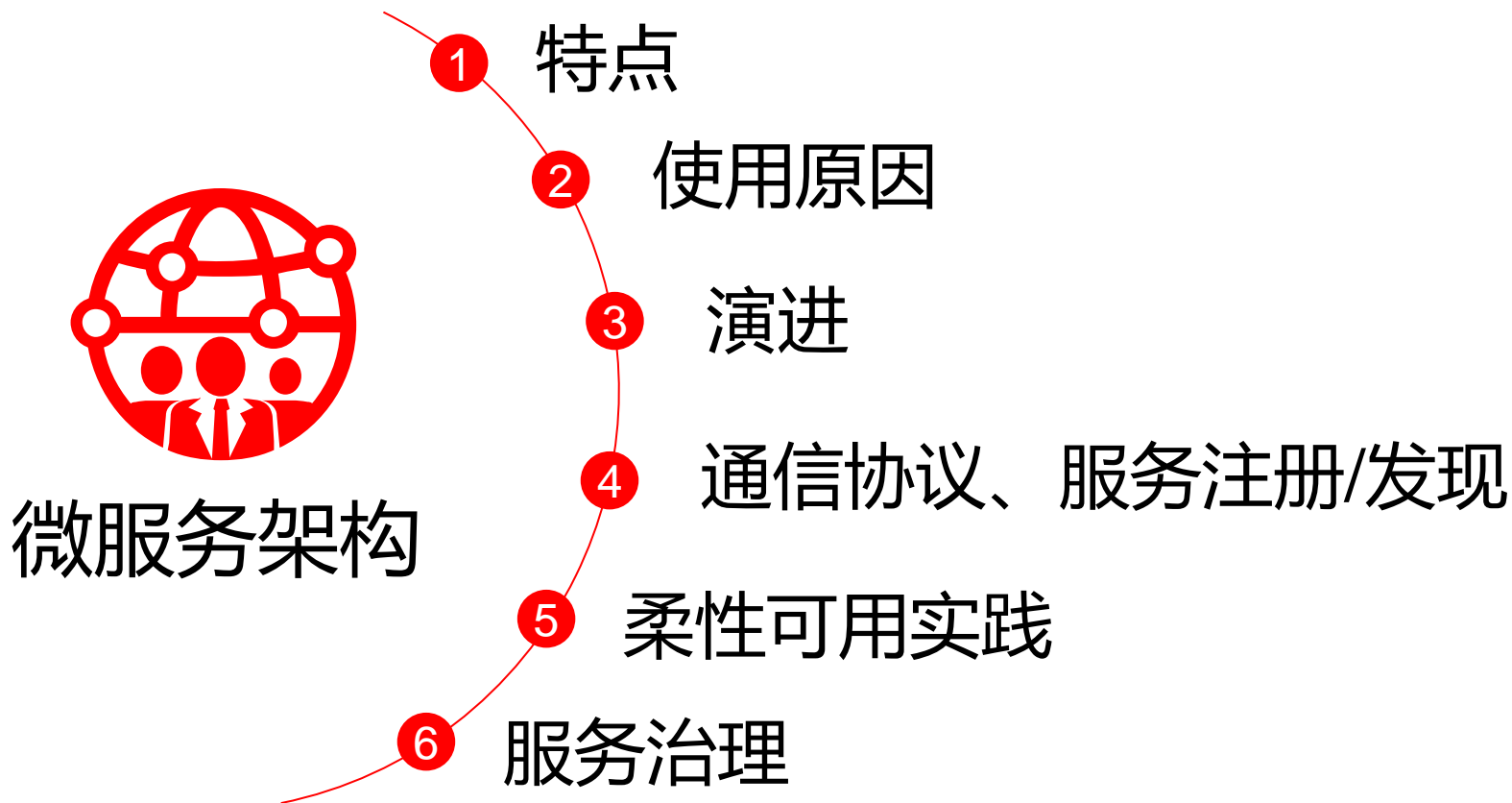
Open-falcon



定制开发



6



欢迎关注本人公众号 “架构之美”



后台开发、搜索、推荐算法职位
sunxuan@58.com

Thanks!

—— 让生活更简单 ——



L O O O L O O O

58集团技术专场