

# 京东咚咚 微服务架构 从理论认识到实践落地

京东·成都研究院  
胡峰

# 目录

- 微服务理论认识
- 单体应用：咚咚面向业务架构
- 微服务化：咚咚面向平台架构
- 微服务架构演进路上的一些疑问和思考

# 微服务理论认识

# 定义

- **小：** 微服务架构即是采用一组小服务来构建应用的方法。
- **独立进程：** 运行在独立的进程中，不同服务通过一些轻量级交互机制来通信。
- **自动化：** 服务围绕业务能力来构建，并依赖自动部署机制来独立部署。

# 起源

- 小即是美
- 一个程序只做好一件事
- 尽可能早地创建原型
- 可移植性比效率更重要



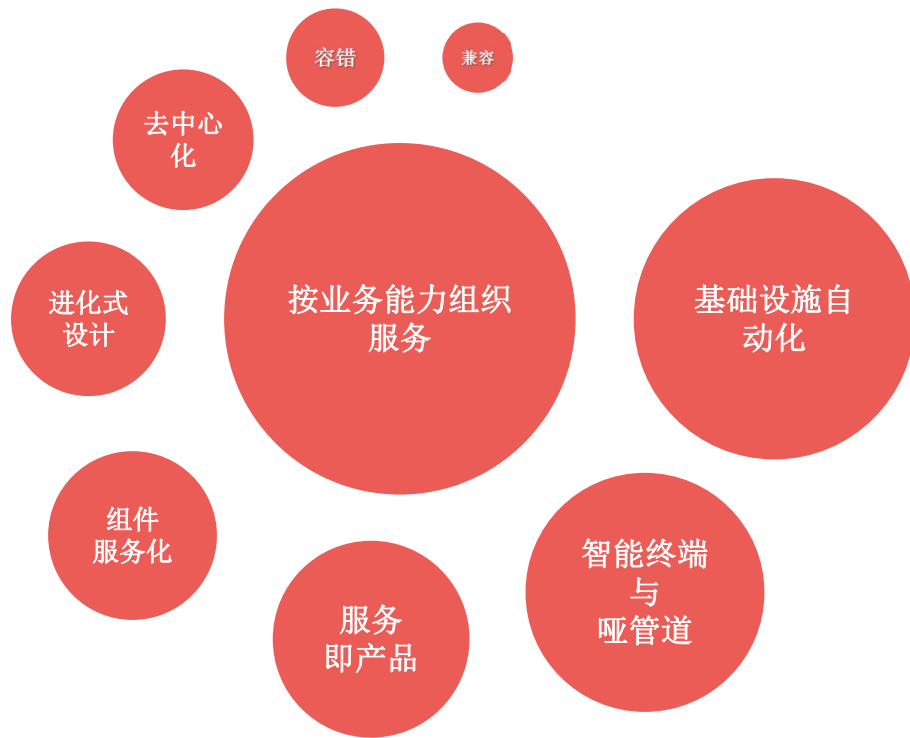
微服务  
就像把 **UNIX 哲学** 应用到了分  
布式系统

# 起源

You should instead think of **Microservices** as a specific approach for **SOA** in the same way that **XP** or **Scrum** are specific approaches for **Agile** software development.

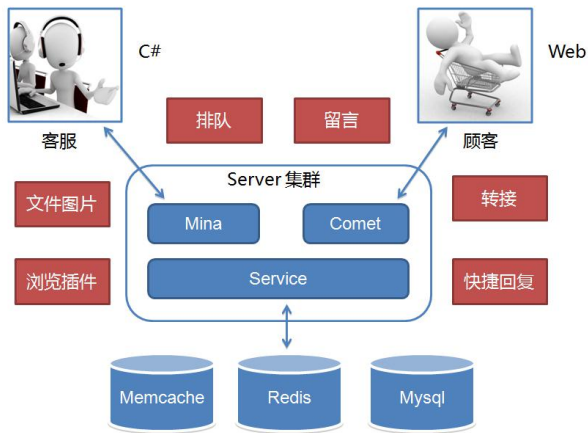
– *Building Microservices* by Sam Newman

# 特征



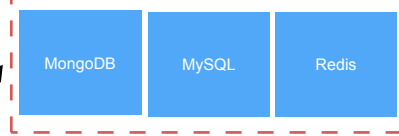
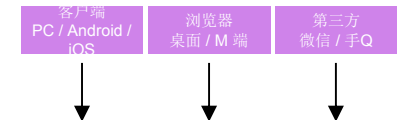
# 单体应用 咚咚面向业务架构





**2.0 成长期**  
全部业务逻辑实现部署在一个 Tomcat 中

**3.0 爆发期**  
对业务进行分级部署隔离



# 问题

- 代码量膨胀到 40 万行+
- 越来越不敏捷了



# 问题

- 新业务接入，扩展维护成本高

- 复制工程
- 根据业务差异定制开发
- 独立部署，每套部署含双机房主备和灰度环境，浪费资源



“什么时候接入我们？”

“给我们的排期定了没？！”

Xxx!!!!

“我们要的功能能实现吗？！！”



# 微服务化 咚咚面向平台架构

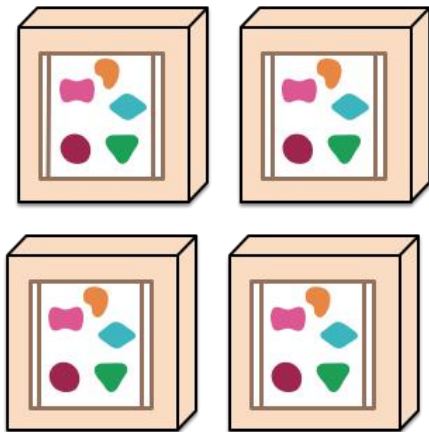
理论

# 服务拆分

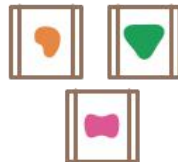
*A monolithic application puts all its functionality into a single process...*



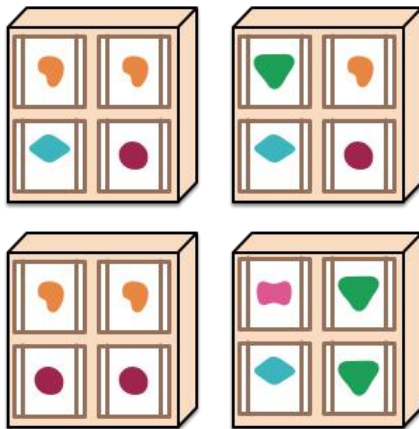
*... and scales by replicating the monolith on multiple servers*



*A microservices architecture puts each element of functionality into a separate service...*

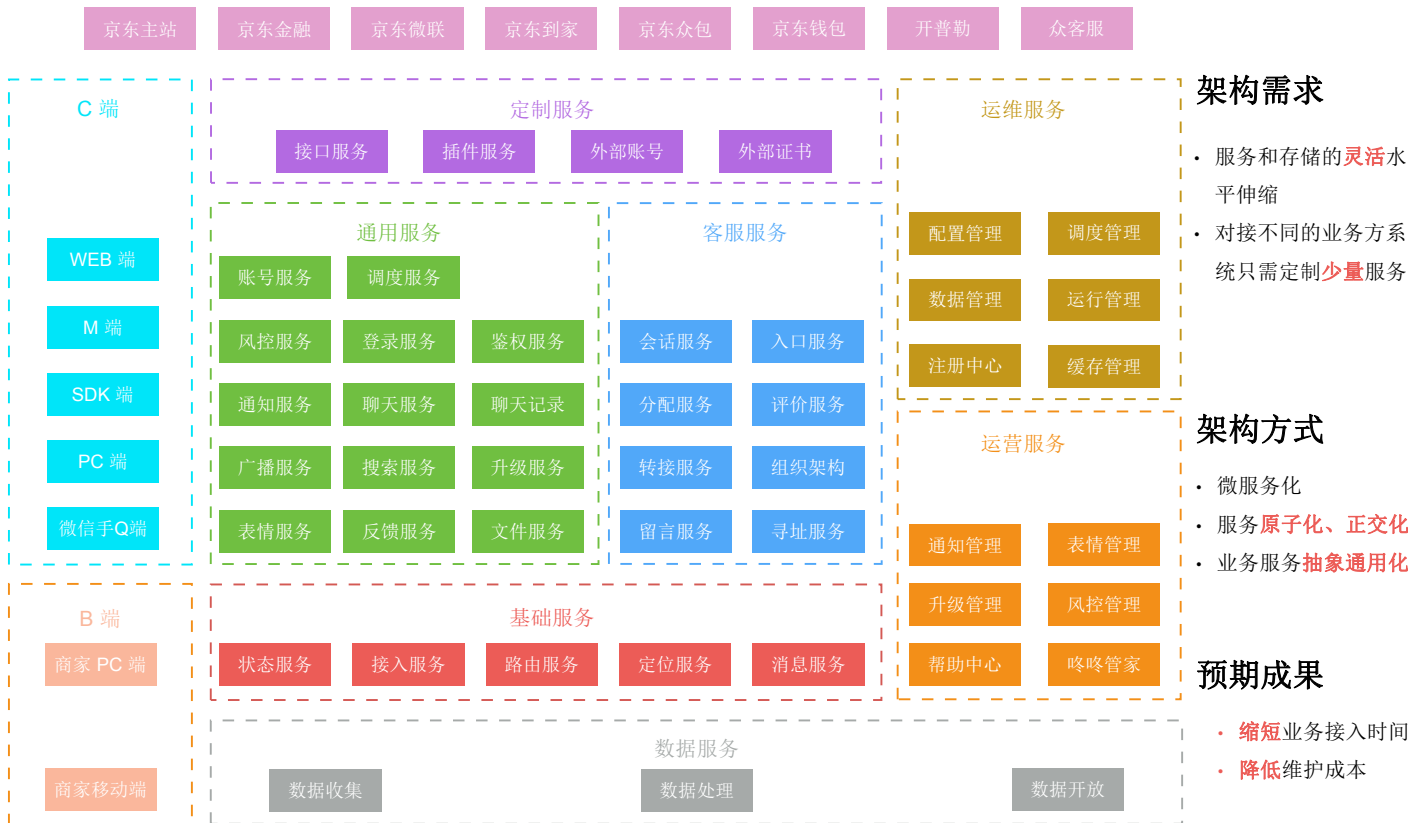


*... and scales by distributing these services across servers, replicating as needed.*



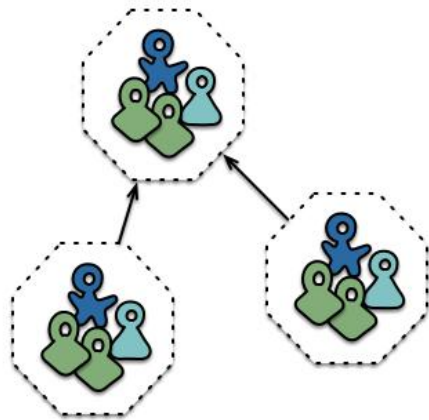
实践

# 服务拆分

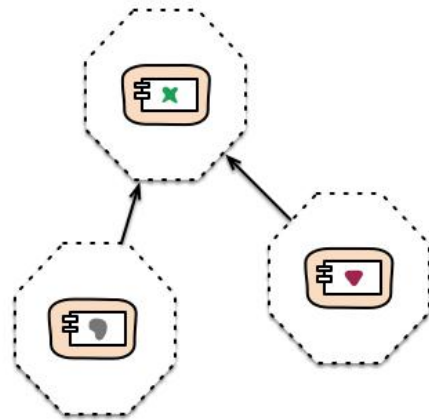


理论

# 服务协作



全栈团队  
Cross-functional teams...



按业务能力组织服务  
... organised around capabilities  
Because Conway's Law

康威定律

Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.

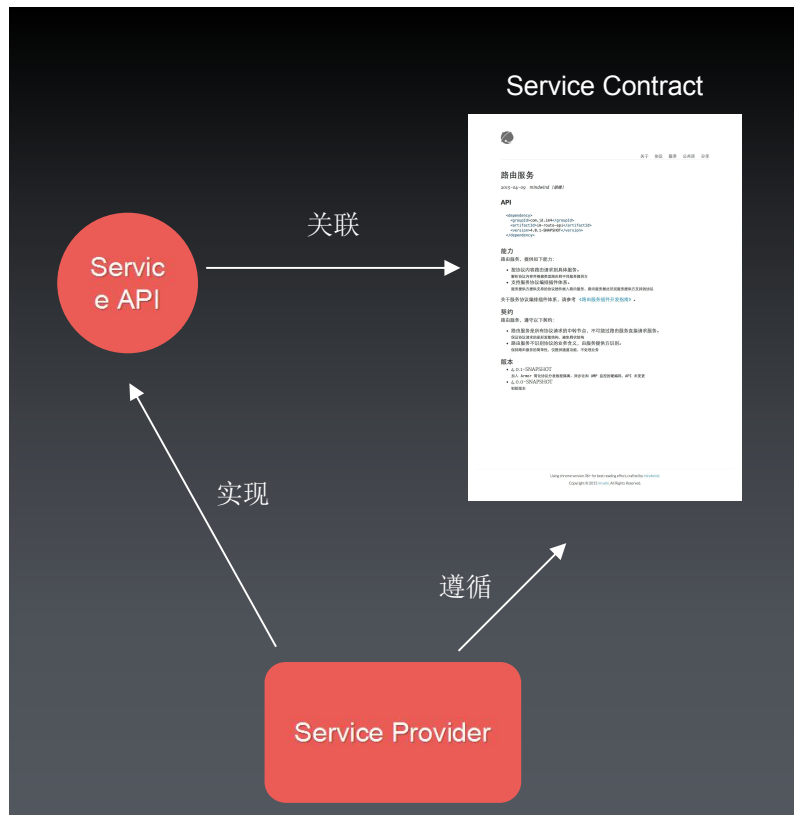
– Melvin Conway

实践

# 服务协作

- 契约式开发协作

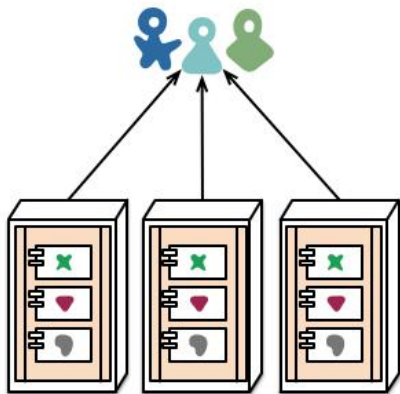
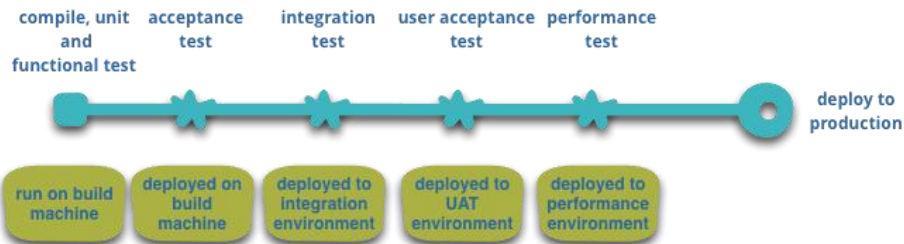
- API
- 能力
- 契约
- 版本



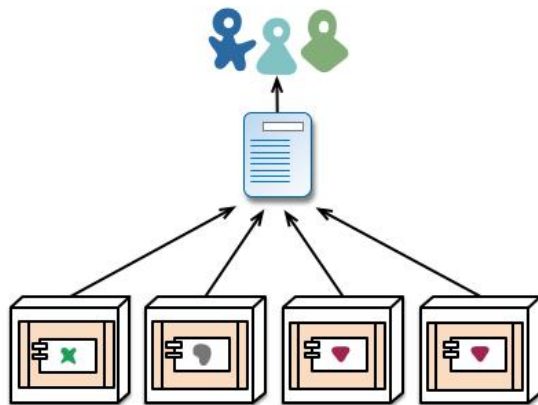


理论

# 服务部署



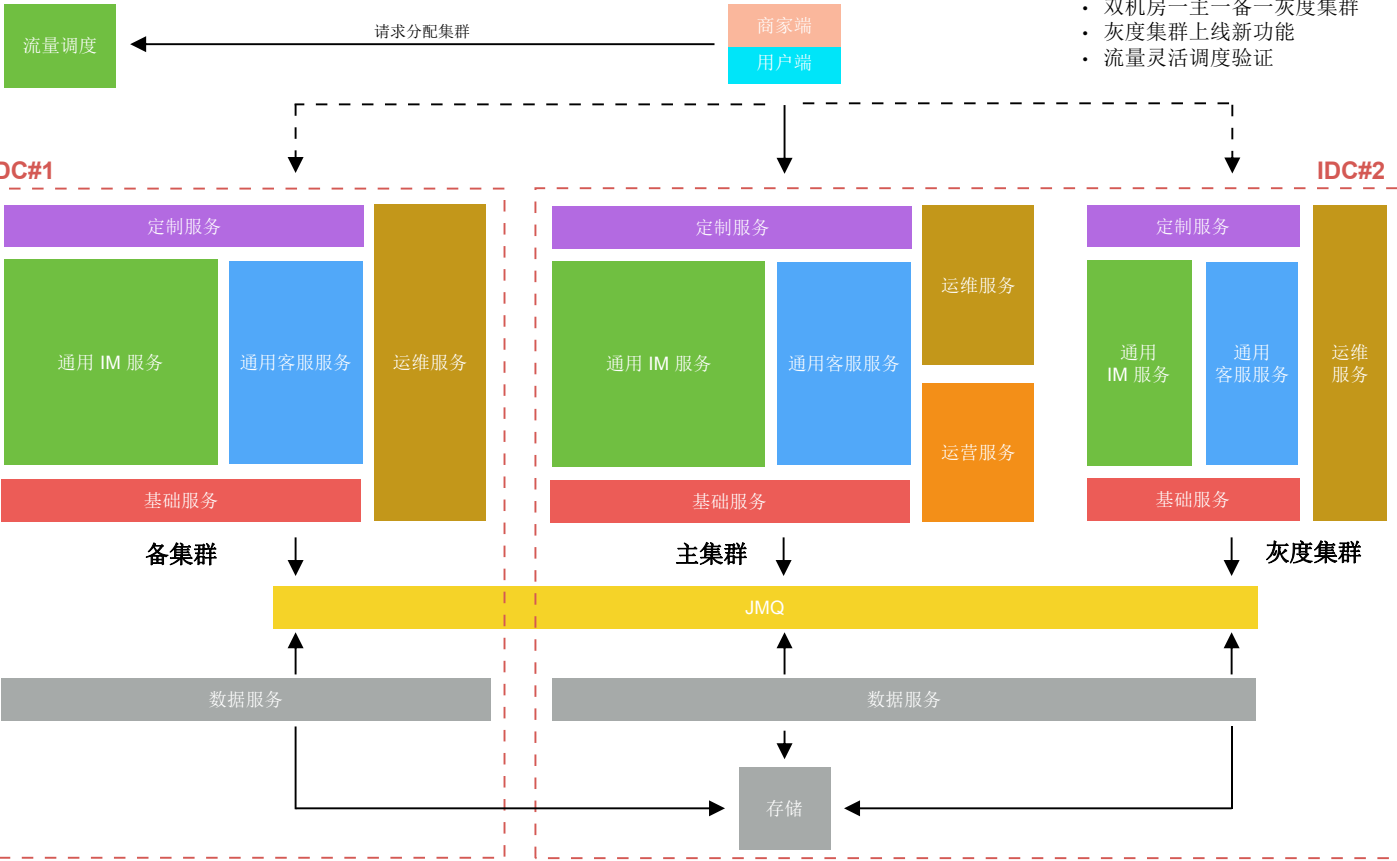
monolith - multiple modules in the same process



microservices - modules running in different processes

实践

# 服务部署



实践

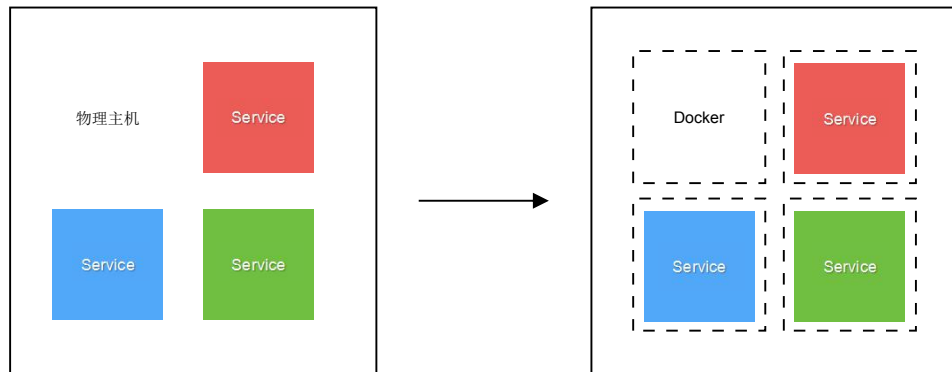
# 服务部署



一主机多服务

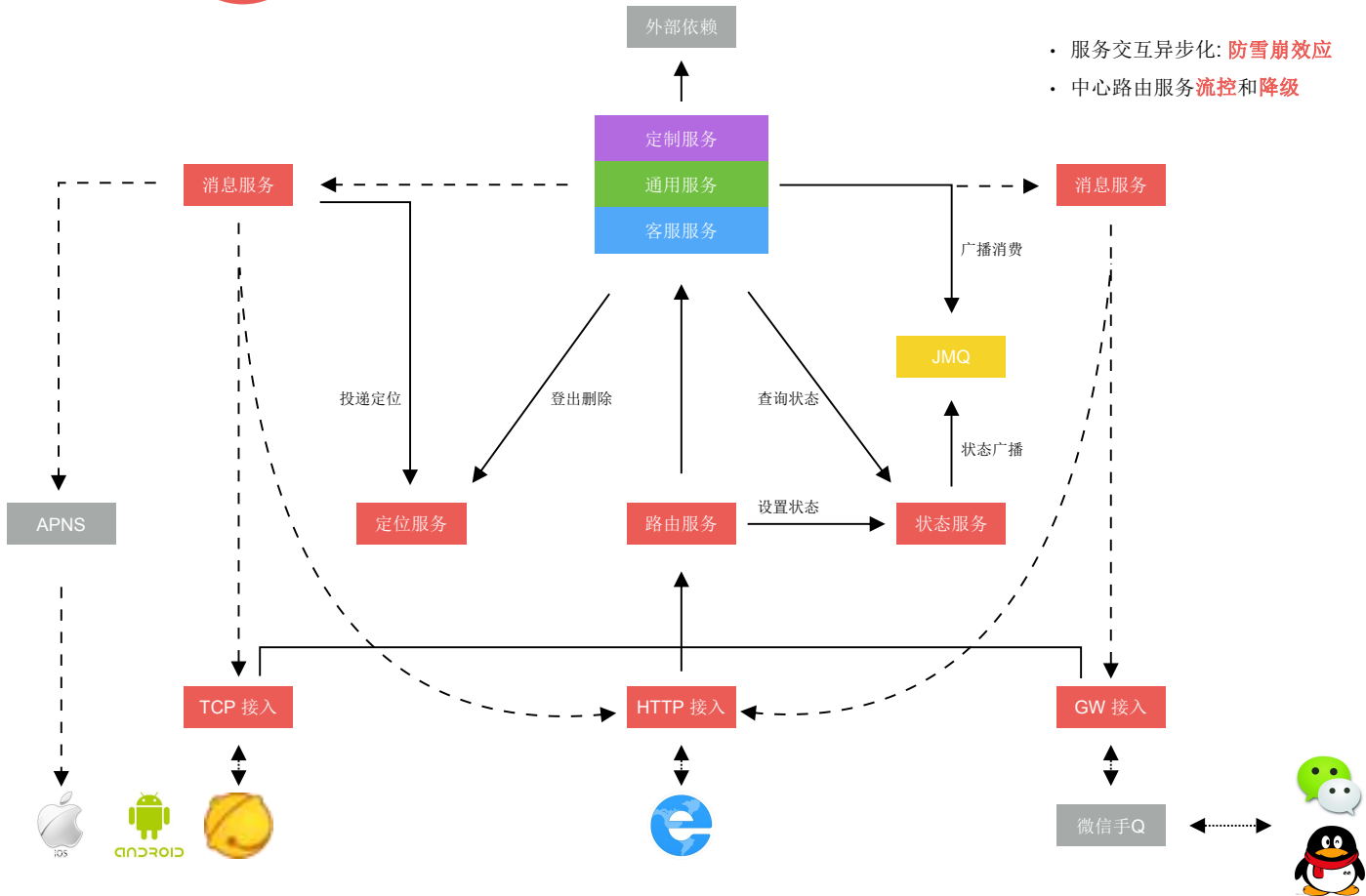
一主机一服务

- **50+** 微服务
- **2000+** Docker 容器



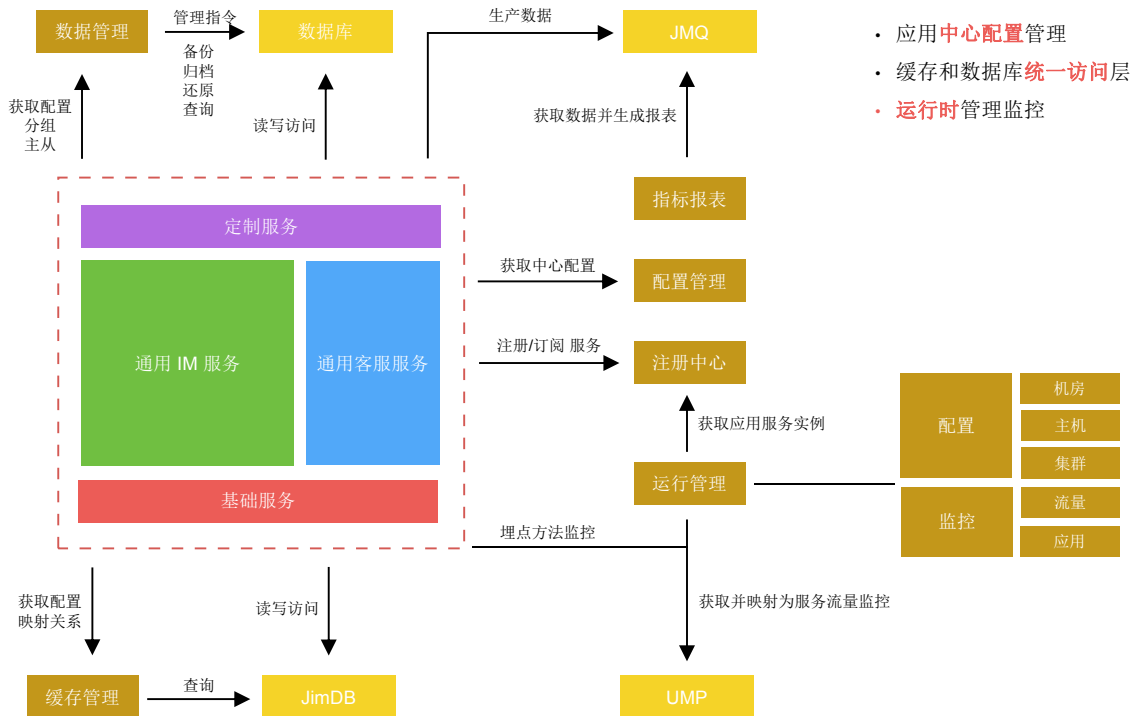
实践

# 服务编排



实践

# 服务运维



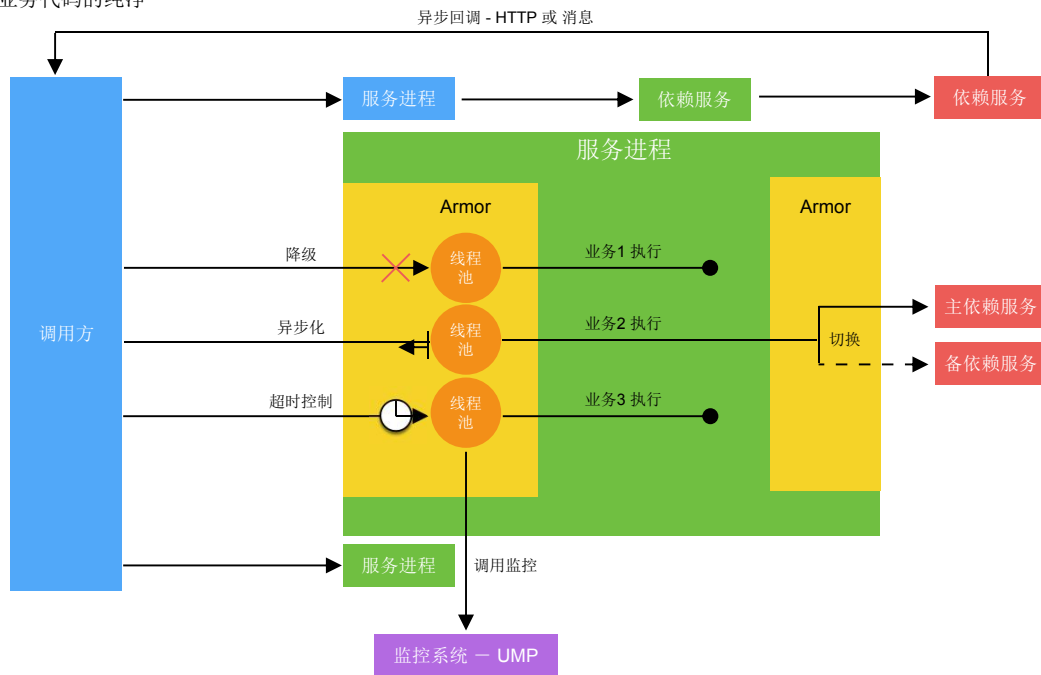
- 应用中心配置管理
- 缓存和数据库统一访问层
- 运行时管理监控

# 服务隔离

- **进程隔离:** 微服务独立进程天然隔离
- **线程隔离:**

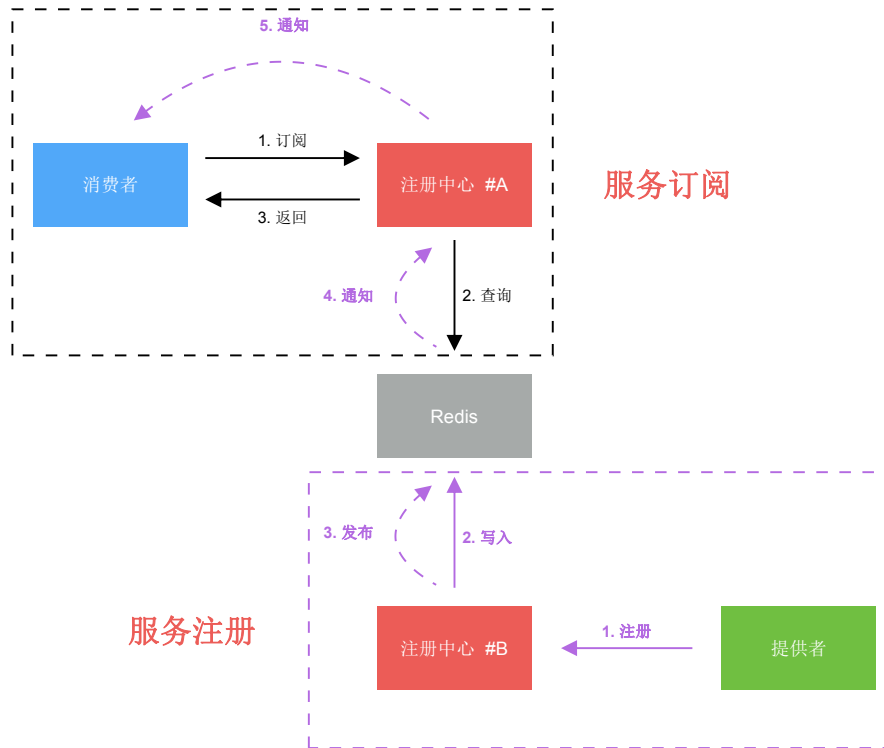
中心路由服务针对不同业务消息使用独立线程池  
利用 AOP 技术切入 RPC 和 业务代码之间  
既隔离了业务线程池，同时保证了业务代码的纯净

- 进程隔离
- 线程隔离
- 依赖隔离



实践

# 服务发现



实践

# 服务监控

用户视角

业务监控

长周期趋势  
多维度



服务监控

响应时间  
流量 TPS  
全链路  
染色



基础监控

IDC  
网络  
主机  
OS  
VM



系统视角



# 微服务架构演进路上 的 一些疑问和思考

不能只开车，还得修路





乘车人 - 业务需求方



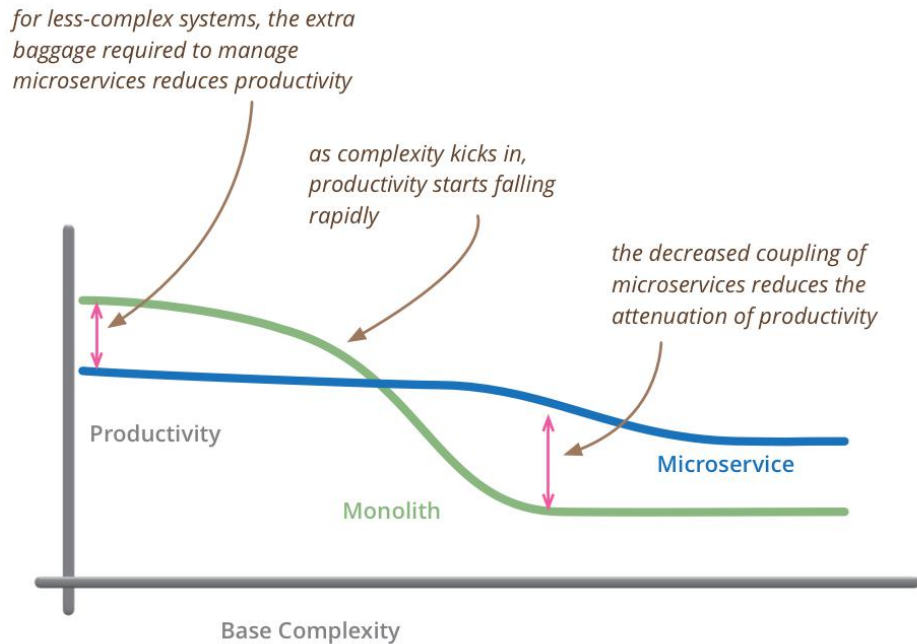
开车人 - 业务研发



修路人 - 基础研发

# 微服务实施前提

- 1 ~ 2k 行，从爬行到奔跑
- 2k ~ 20k 行，开车
- 20k+，开飞机



but remember the skill of the team will outweigh any monolith/microservice choice

# 原则

## 战略目标

### 业务扩张

- 加机器不加入

### 业务开拓

- 快速
- 边际成本低
- 自服务

### 业务创新

- 试错成本

## 架构原则

### 契约化开发

- 契约变更通知
- 考虑消费方

### 自动化文化

- 代码模板生成
- 编译、测试、部署
- 日志收集
- 告警处理

### 反脆弱性

- 错误隔离
- 超时管理
- 断路器
- 隔离仓
- 独立部署
- 高度可监控

## 设计与交付实践

### 标准化接口

- Dubbo RPC（内部）
- JSF RPC（外部）
- REST（跨语言）

### 标准化输出

- 日志错误
- 报警提示

### 标准化配置

- 配置文件
- 启动脚本
- 环境参数

### 标准化监控

- AOP 接入
- 埋点约定



# Q & A



微信公众号  
瞬间之间

Thanks