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。

— *Microservices* by Martin Fowler & James Lewis

起源

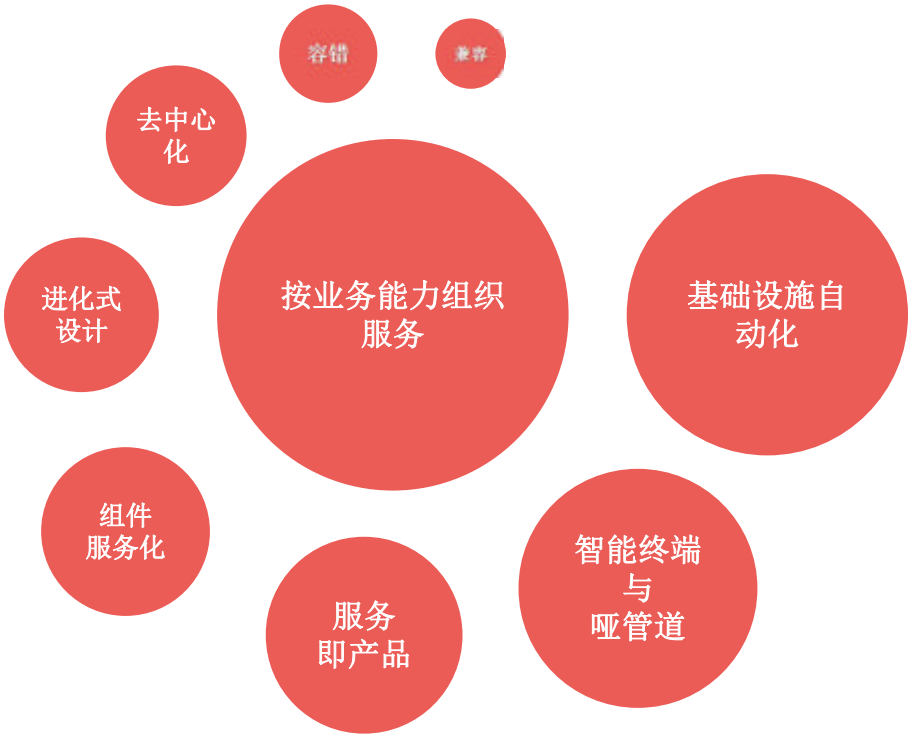
- 小即是美
- 一个程序只做好一件事
- 尽可能早地创建原型
- 可移植性比效率更重要

起源

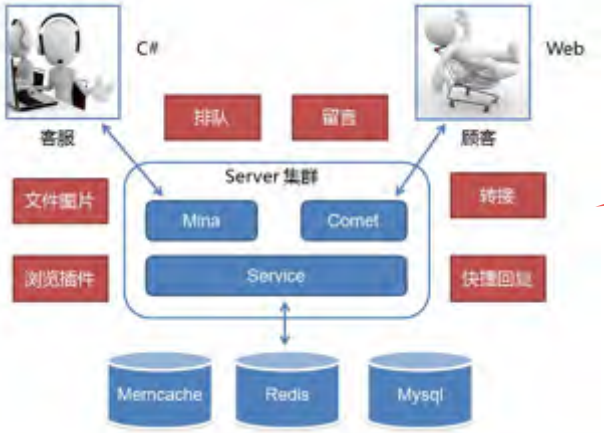
You should instead think of **Microservices** as a specific approach for **SOA** in the same way that XP or Scrum are specific approaches for Agile software development.

– *Building Microservices* by Sam Newman

特征

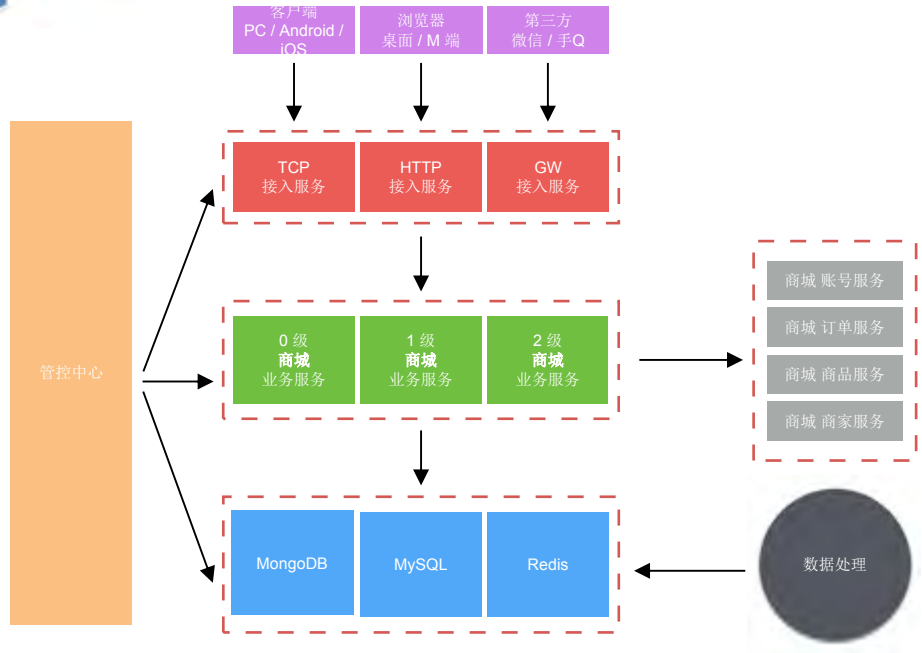


单体应用
咚咚面向业务架构



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

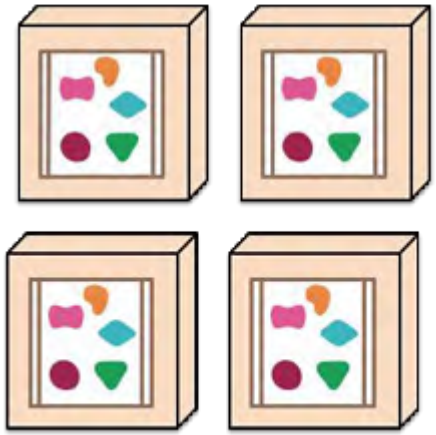
微服务化
咚咚面向平台架构

理论

服务拆分

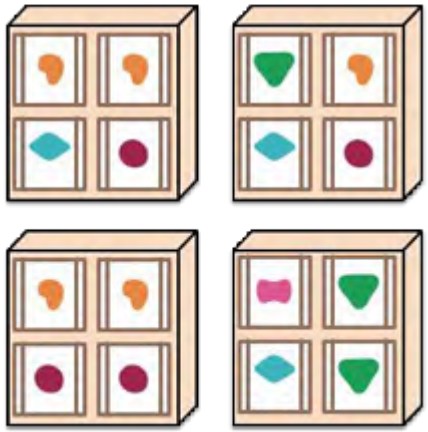
A monolithic application puts all its functionality into a single process...

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



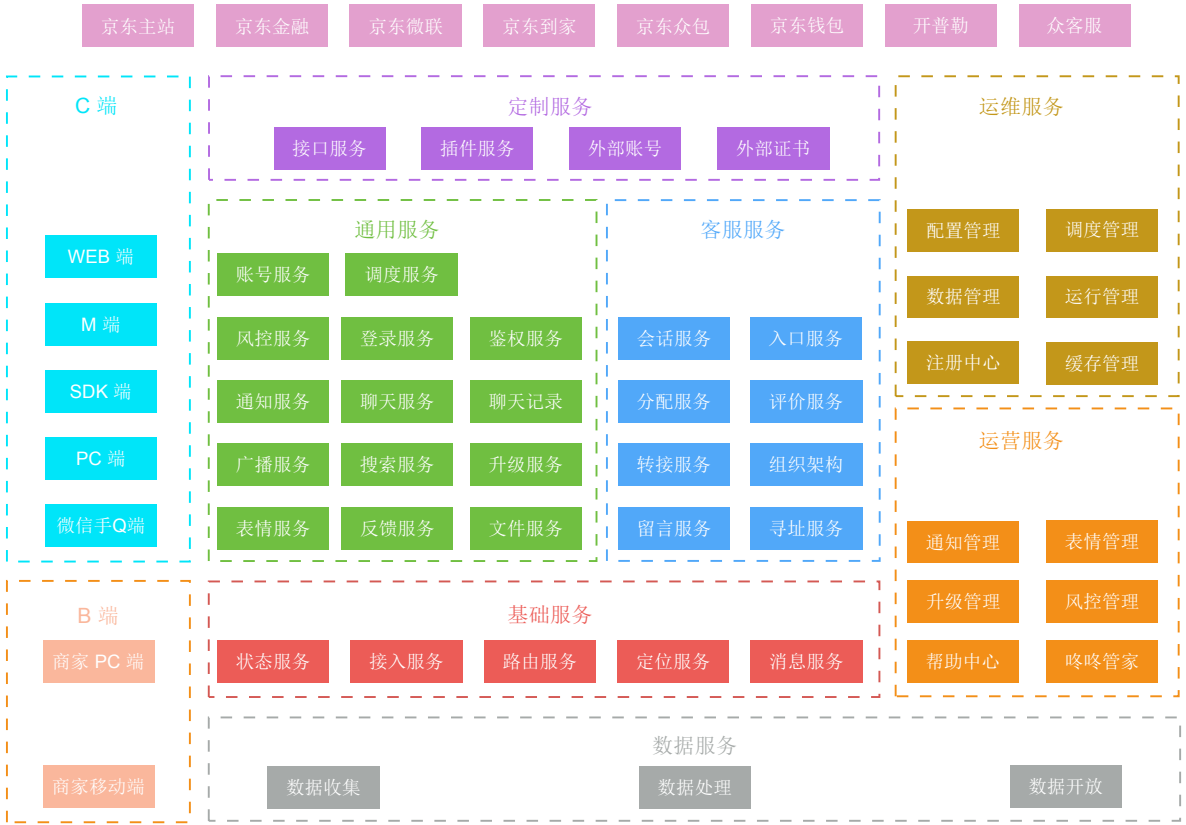
A microservices architecture puts each element of functionality into a separate service...

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



实践

服务拆分



架构需求

- 服务和存储的**灵活**水平伸缩
- 对接不同的业务方系统只需定制**少量**服务

架构方式

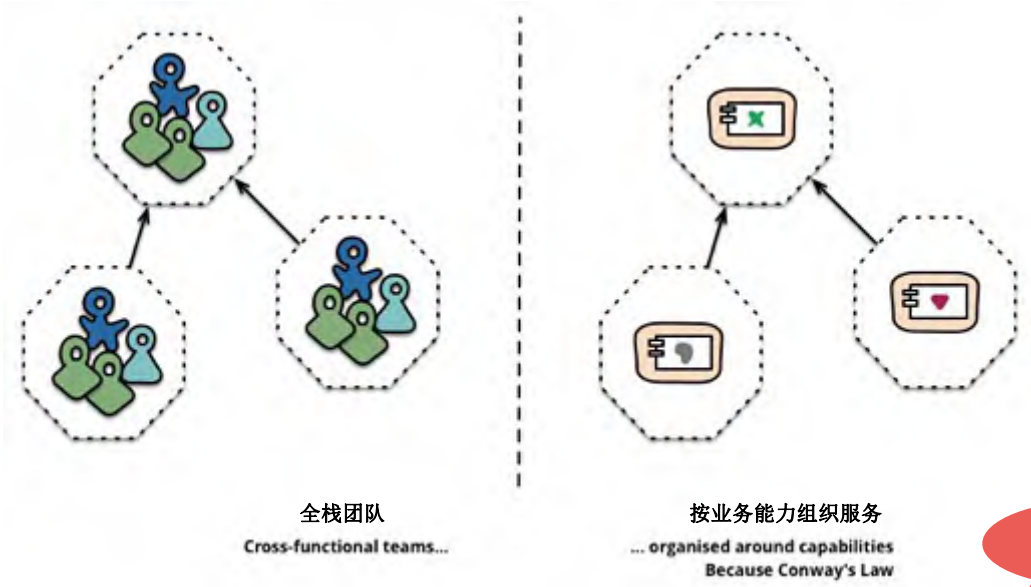
- 微服务化
- 服务**原子化、正交化**
- 业务服务**抽象通用化**

预期成果

- **缩短**业务接入时间
- **降低**维护成本

理论

服务协作



康威定律

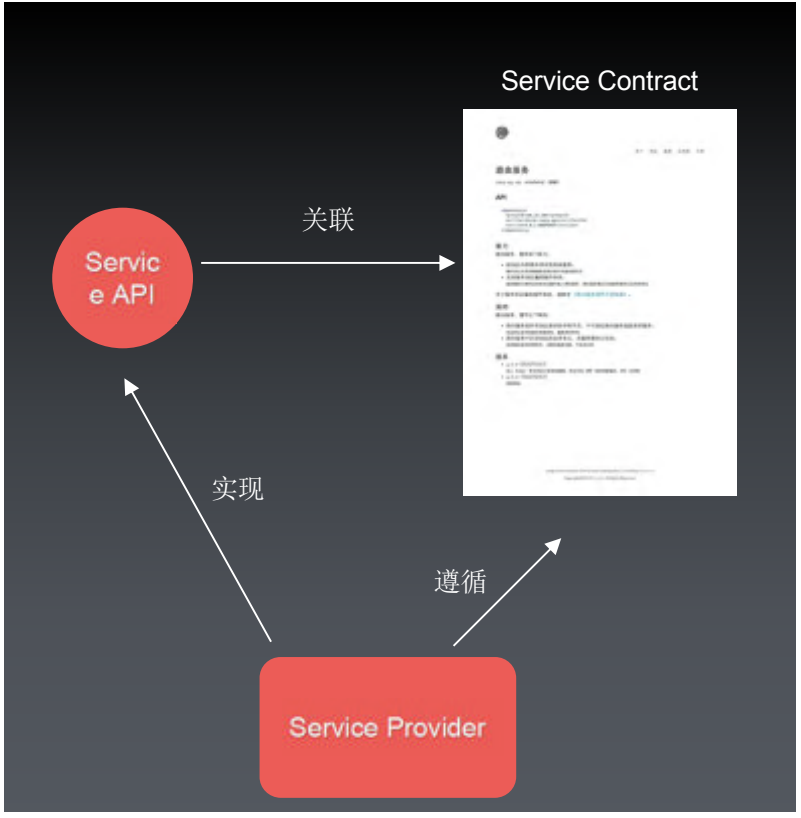
Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.

– Melvin Conway

实践

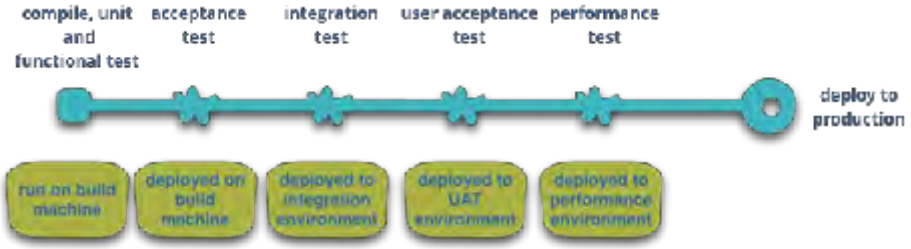
服务协作

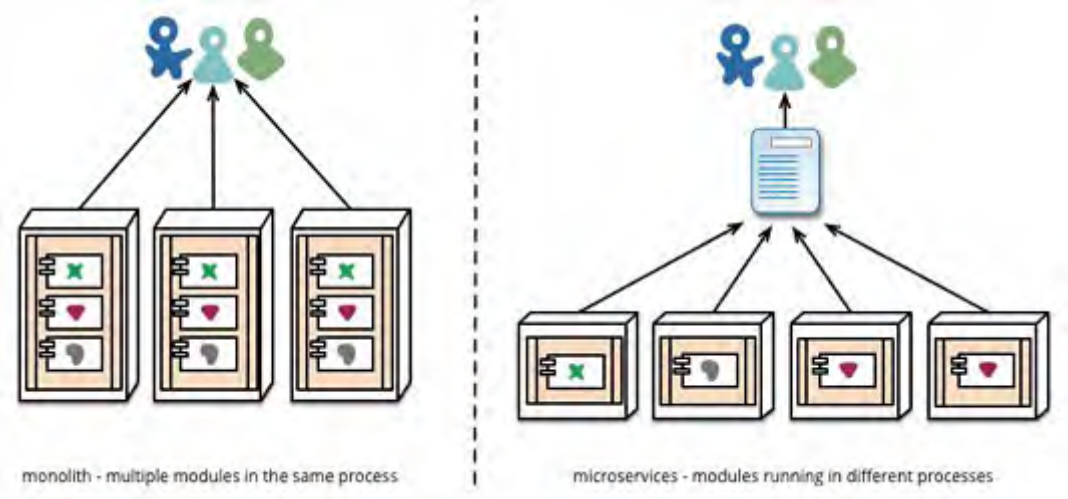
- 契约式开发协作
  - API
  - 能力
  - 契约
  - 版本



理论

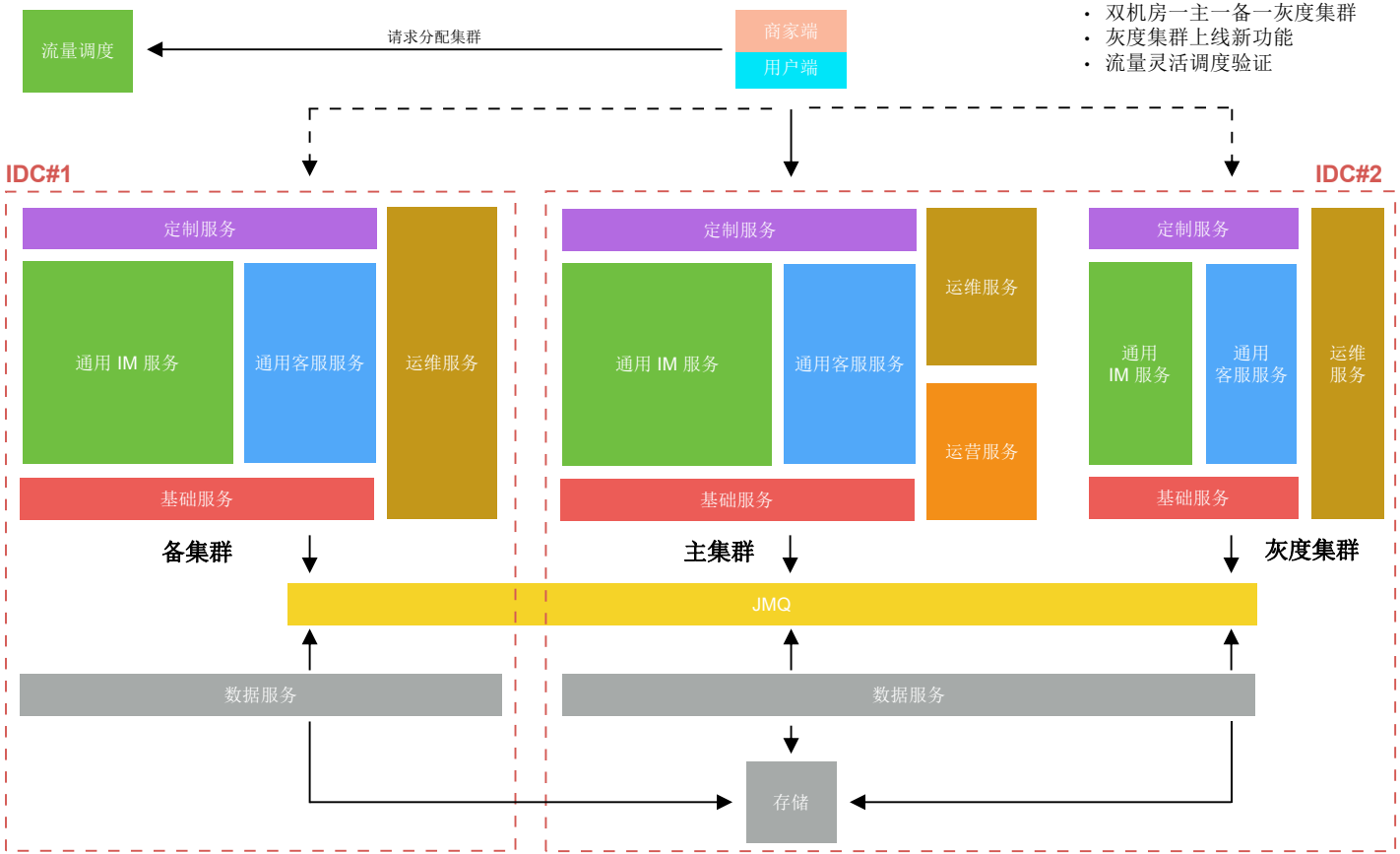
服务部署


compile, unit and functional test
acceptance test
integration test
user acceptance test
performance test
deploy to production
run on build machine
deployed on build machine
deployed to integration environment
deployed to UAT environment
deployed to performance environment


monolith - multiple modules in the same process
microservices - modules running in different processes

实践

服务部署

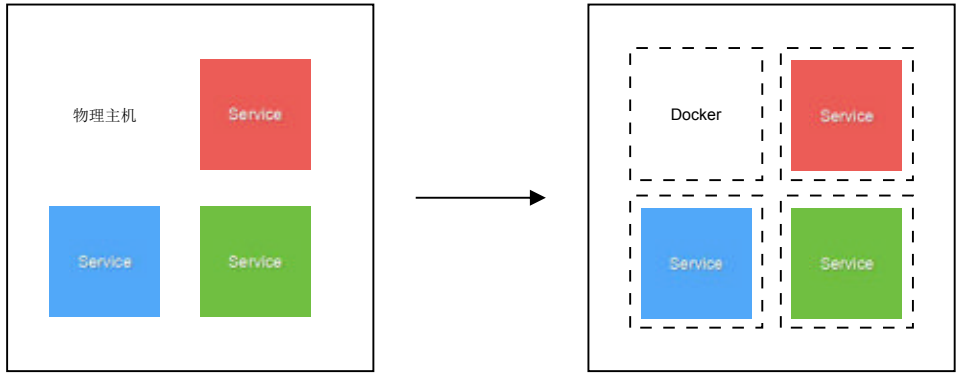

流量调度
请求分配集群
商家端
用户端
• 双机房一主一备一灰度集群
• 灰度集群上线新功能
• 流量灵活调度验证
IDC#1
IDC#2
定制服务
通用 IM 服务
通用客服服务
运维服务
基础服务
运营服务
备集群
主集群
灰度集群
JMQ
数据服务
存储

实践

服务部署

一主机多服务　　一主机一服务

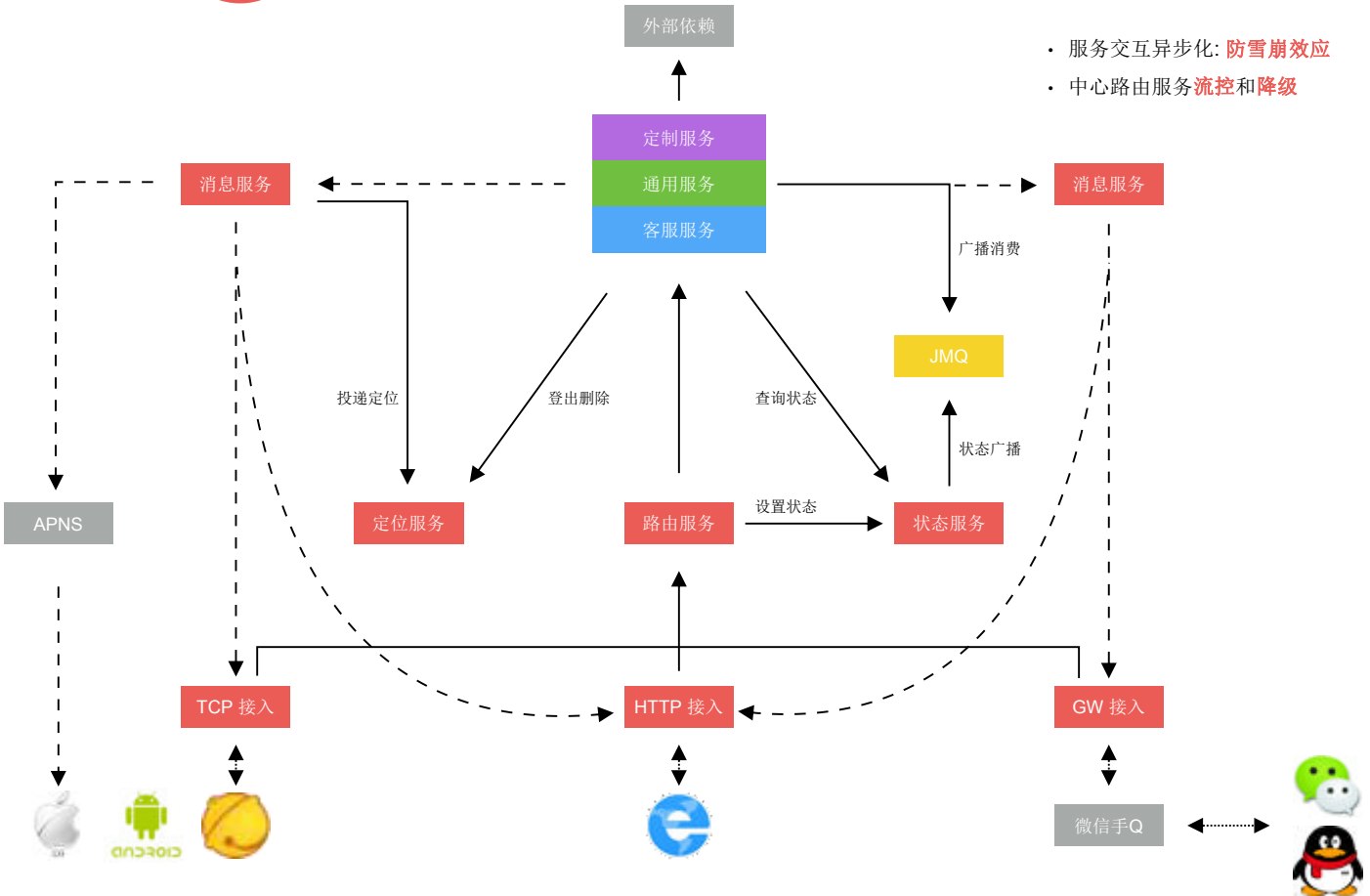
- **50+** 微服务
- **2000+** Docker 容器



实践

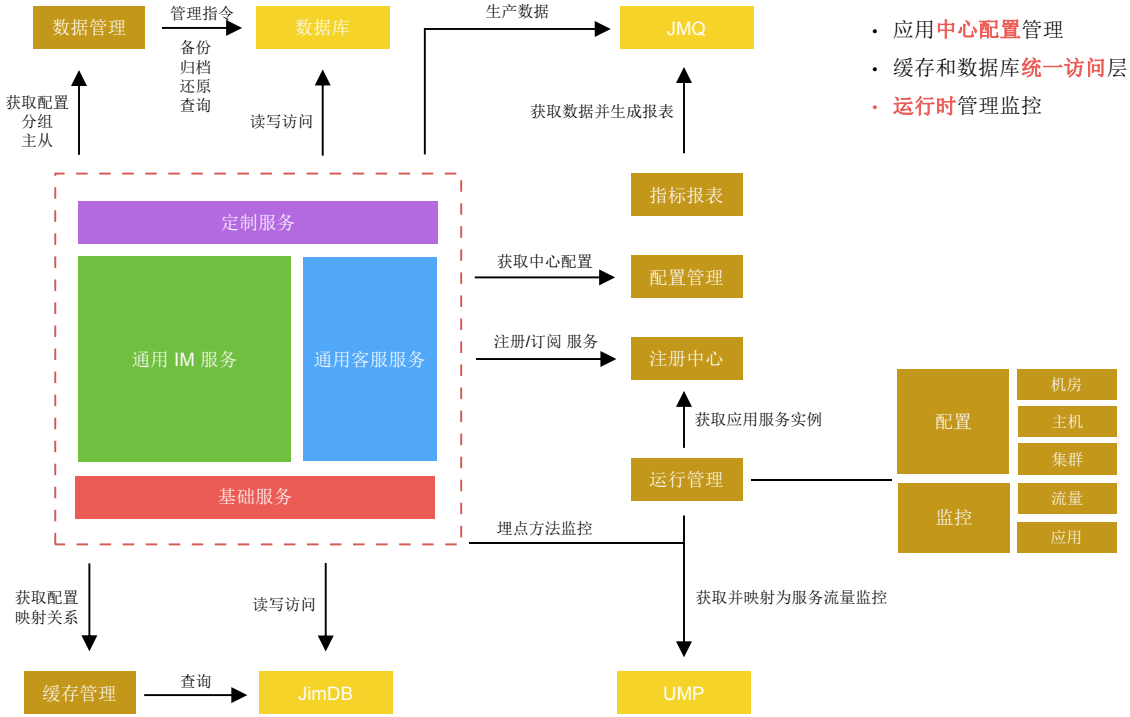
服务编排

- 服务交互异步化: 防雪崩效应
- 中心路由服务流控和降级



实践

服务运维



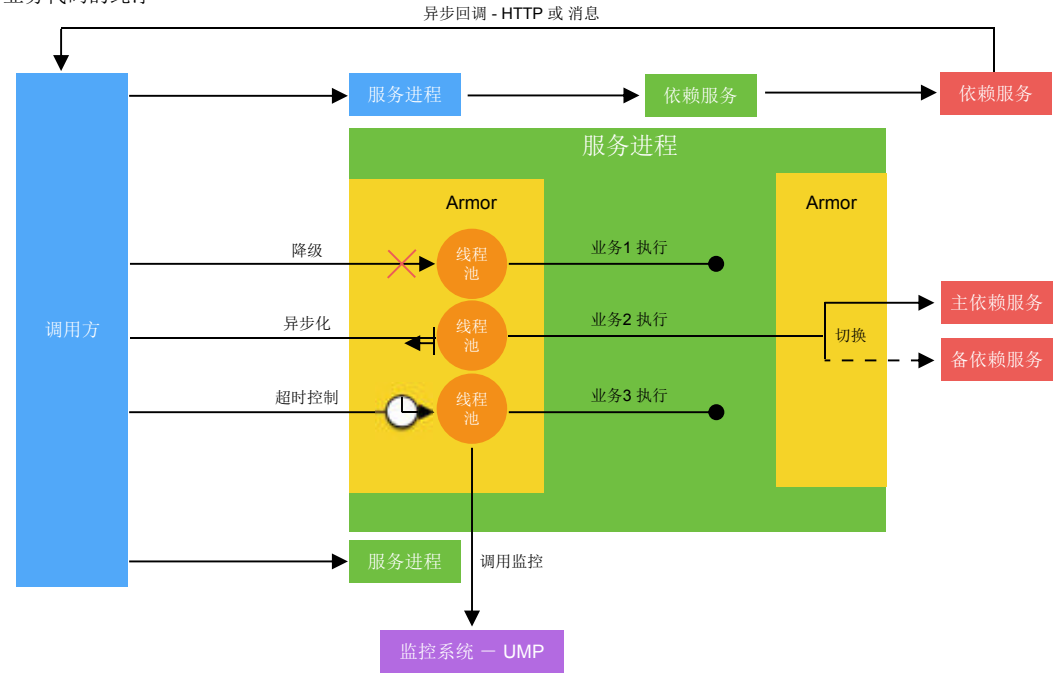
- 应用**中心配置**管理
- 缓存和数据库**统一访问**层
- **运行时**管理监控

实践

服务隔离

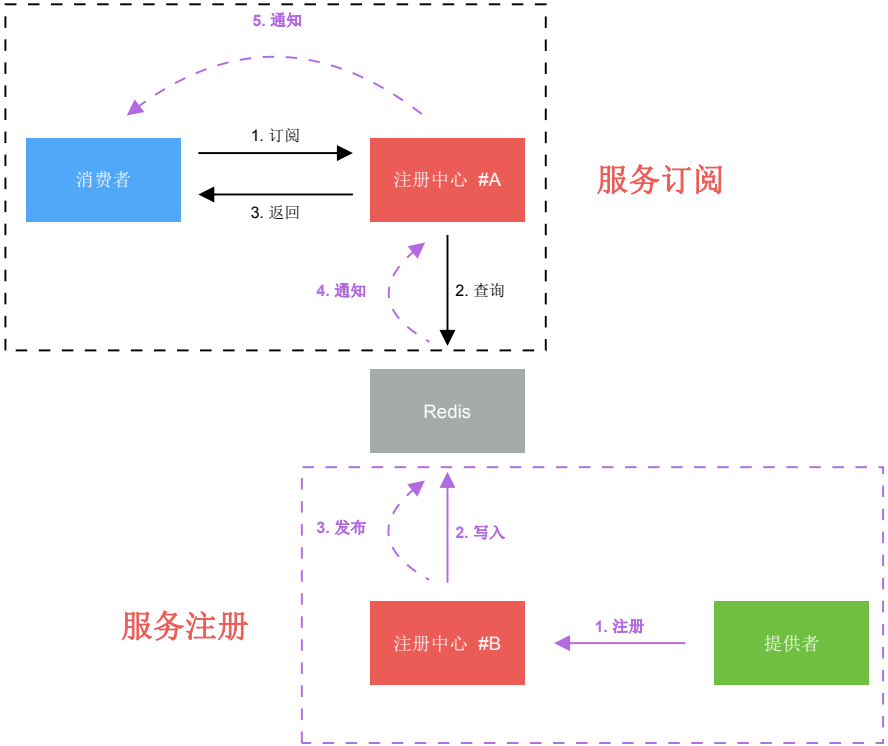
- **进程隔离：**微服务独立进程天然隔离
- **线程隔离：**
  中心路由服务针对不同业务消息使用独立线程池
  利用 AOP 技术切入 RPC 和 业务代码之间
  既隔离了业务线程池，同时保证了业务代码的纯净

- 进程隔离
- 线程隔离
- 依赖隔离



实践

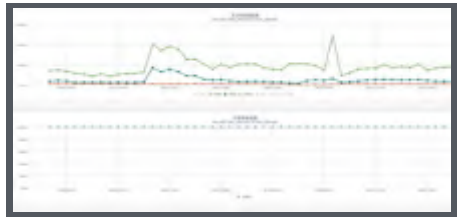
服务发现

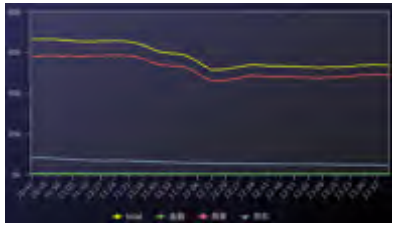

5. 通知
1. 订阅
消费者
注册中心 #A
3. 返回
4. 通知
2. 查询
服务订阅
Redis
3. 发布
2. 写入
服务注册
注册中心 #B
1. 注册
提供者

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

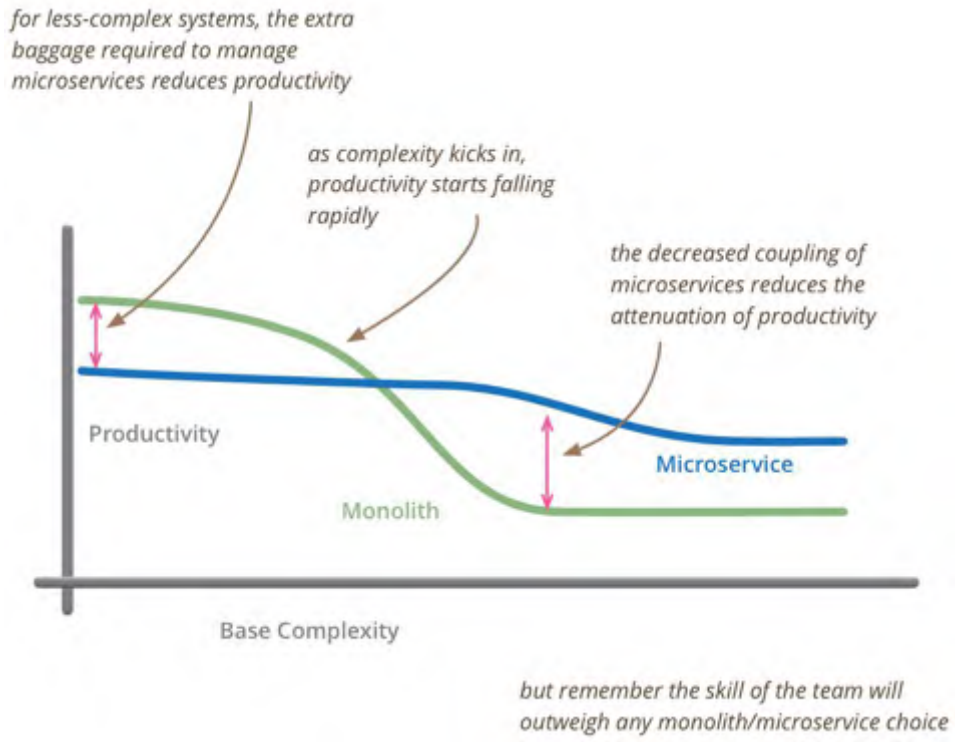

乘车人 - 业务需求方


开车人 - 业务研发

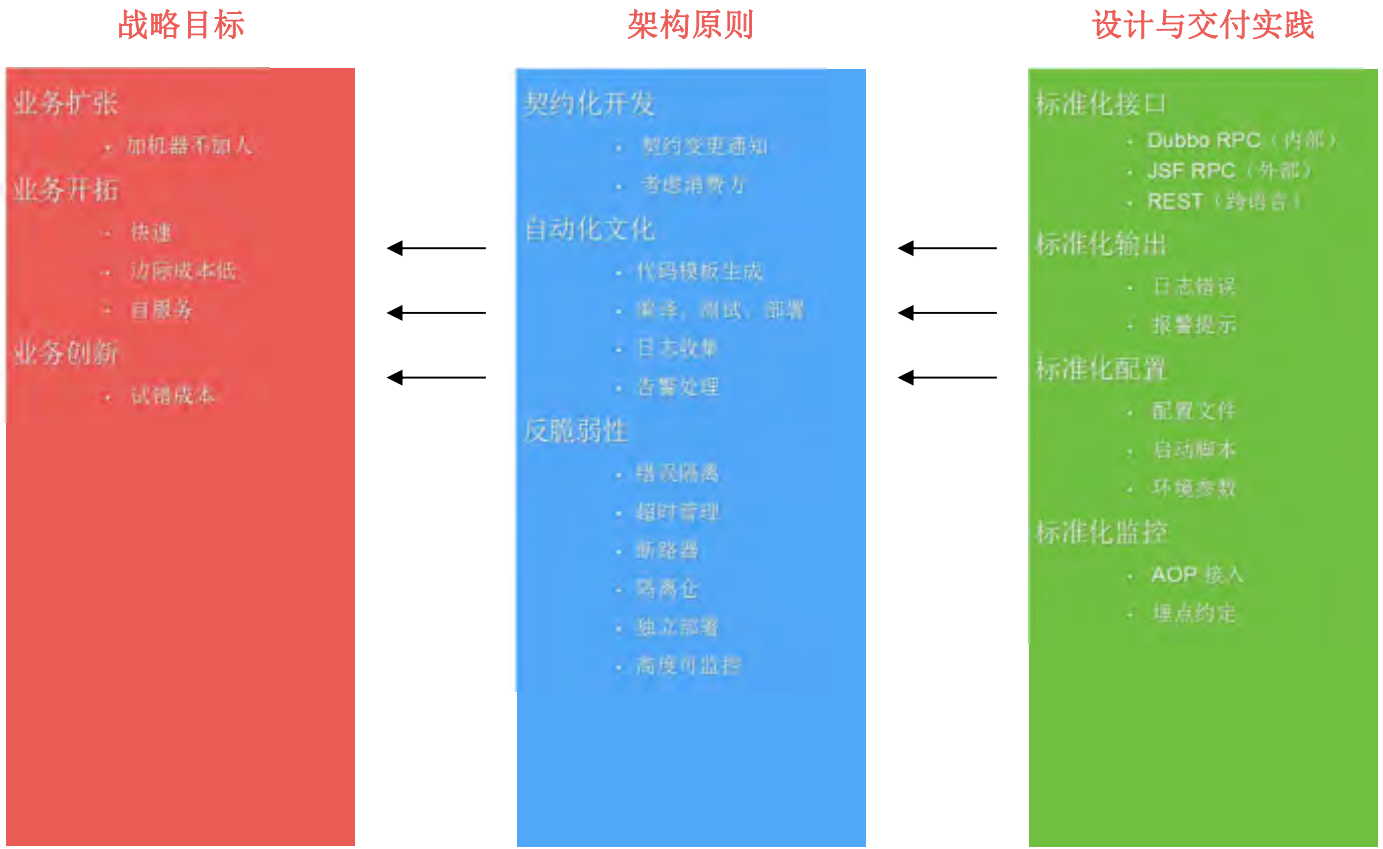

修路人 - 基础研发

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



原则


战略目标
业务扩张
加机器不加人
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
集成、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
瞬息之间


Thanks