大型企业云平台架构和关键技术实践

华为公司 华为软件云平台架构师 苗彩霞

自我介绍

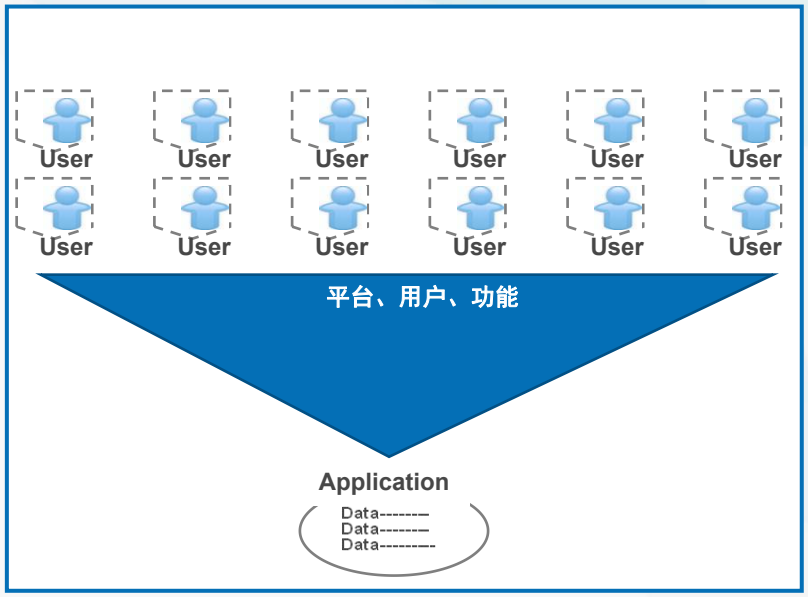
苗彩霞
华为软件云平台架构师

- 2000年加入华为，历经了华为软件的发展和技术演进与变革
  - 融合智能网业务平台
  - 分布式SOA中间件平台
  - 微服务化云平台DigitalCube

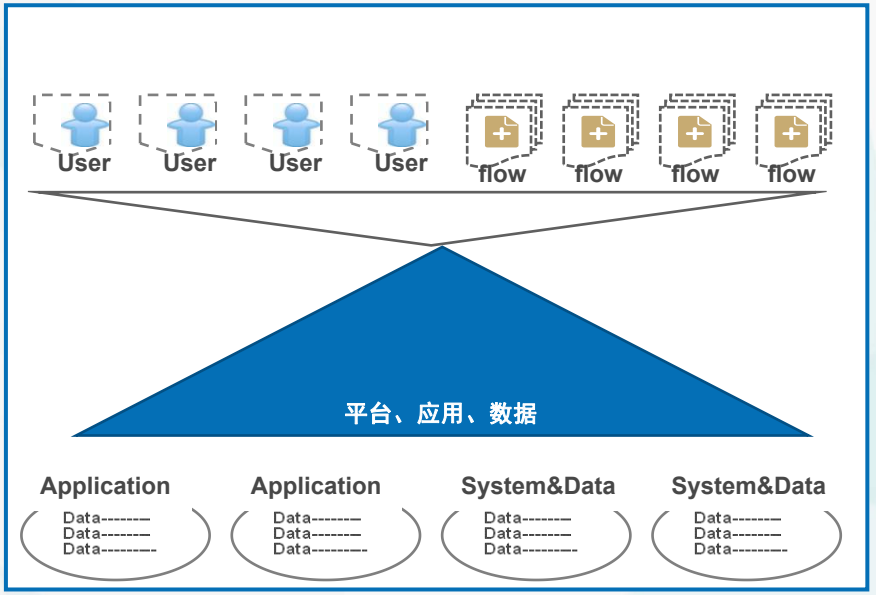
目录

复杂企业应用的特征和云化巨大挑战

B2C消费者应用



B2B企业应用



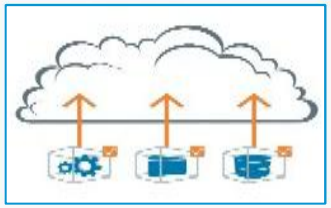
华为电信软件的挑战：

- 一套软件交付全球上千局点，新需求/新特性/特性变更频繁
- 业务复杂，周边几十上百个系统，千余人并行开发上百个模块
- 大/中/小局点（用户从数亿到数十万不等）要求部署灵活，部署维护成本低
- 契约化产品交付，5个9的可用性

庞大复杂的企业软件如何做到高效敏捷、弹性、自动化？

解决问题的关键思路

大系统小做，快速创新



集约化云化建设，
可扩展性和伸缩性

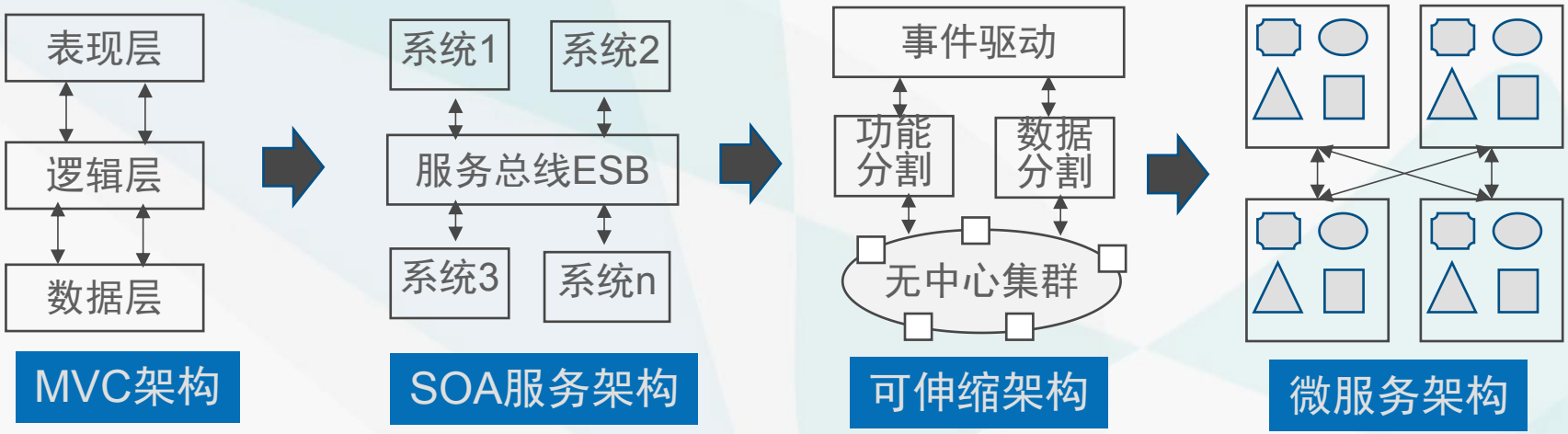
敏捷运营，Easy运维

海量灵活变更、快速上线

自动部署、弹性伸缩

分而治之、自动运维

服务架构演进之路



1、MVC：解决前后端、界面、控制逻辑和业务逻辑分层问题

2、SOA：服务化架构，企业级资产重用和异构系统间的集成对接

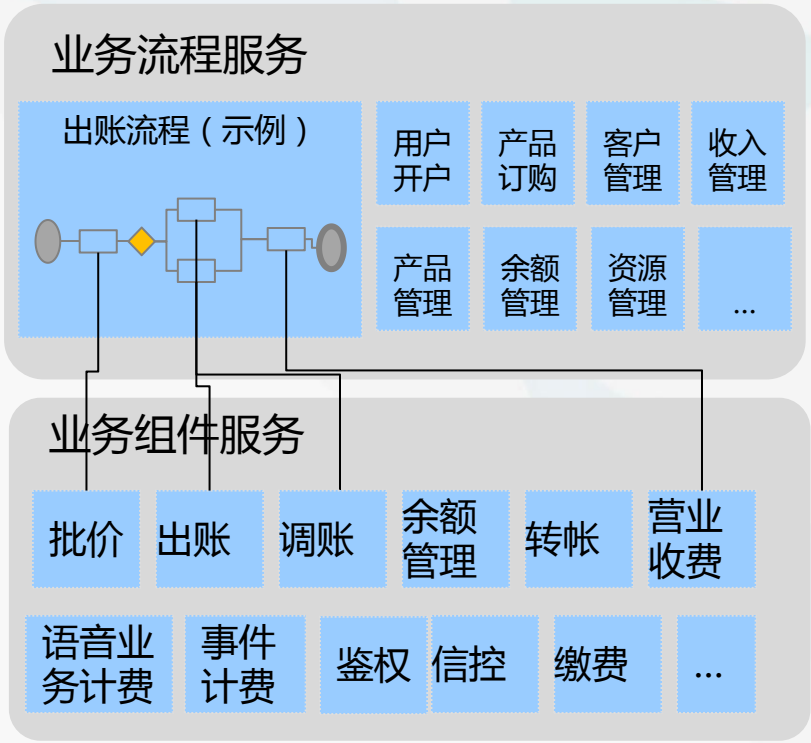
3、RPC/事件驱动可伸缩架构：远程过程调用，无中心集群，本质是系统解耦和分布式通信

4、微服务化架构：功能细粒度解耦，服务自治，敏捷交付、云计算、容器化发展的产物

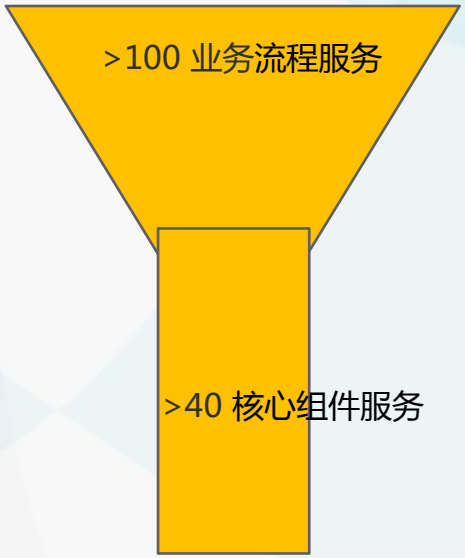
华为软件微服务架构视图

#1 微服务的设计

1、服务化拆分切莫忘了初衷
2、识别业务领域模型，以业务对象为核心进行微服务设计
3、不要一味追求扁平化的微服务，做好微服务分层和聚合
4、服务的数据完整性是关键



漏斗形的服务模型，支撑快速构建新功能和新流程



微服务设计的实践总结

微服务拆分原则：围绕业务功能进行垂直和水平拆分。大小粒度是难点，也是团队争论的焦点。

不好的实践：

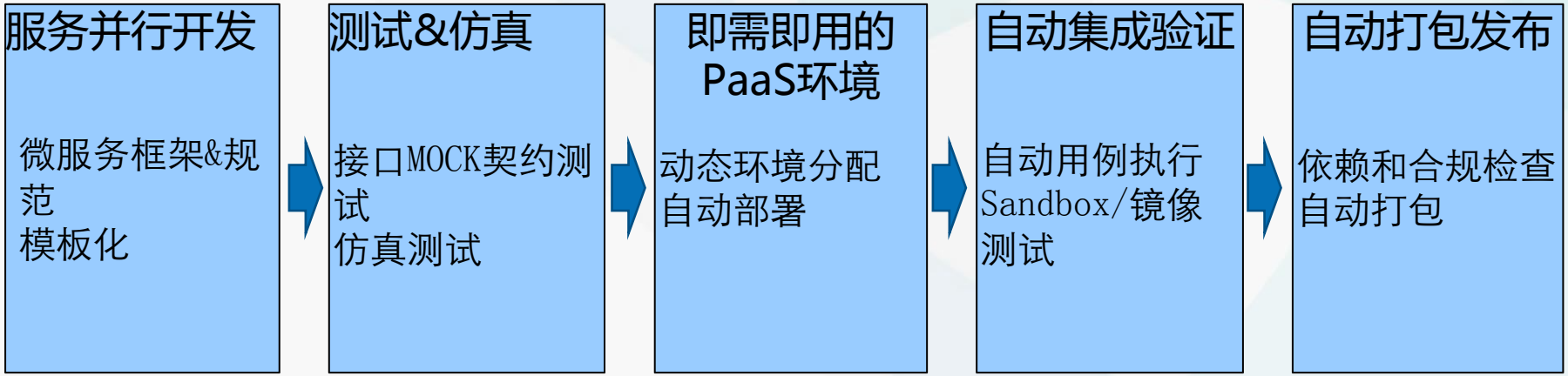
- 以代码量作为衡量标准，例如500行以内
- 拆分的粒度越小越好
- 只关注功能不关注数据

建议的原则：

- 以领域模型为中心、功能完整性、职责单一性
- 粒度适中，团队可接受
- 迭代演进，非一蹴而就
- API和兼容性优先考虑
- 交互性能、部署综合考虑

#2 基于微服务的开发测试和构建

1、众多的微服务，接口交互和业务协同复杂问题通常引发灾难
2、微服务模板化、框架化、工程化是保证开发效率和质量的基础
3、做到 “开发即测试+每日端到端集成构建 ” 是关键



微服务的开发构建的实践总结

微服务开发构建原则：接口先行，语言中立，并行开发，提升产能；自动测试和版本级持续构建，持续集成联调。

不好的实践：

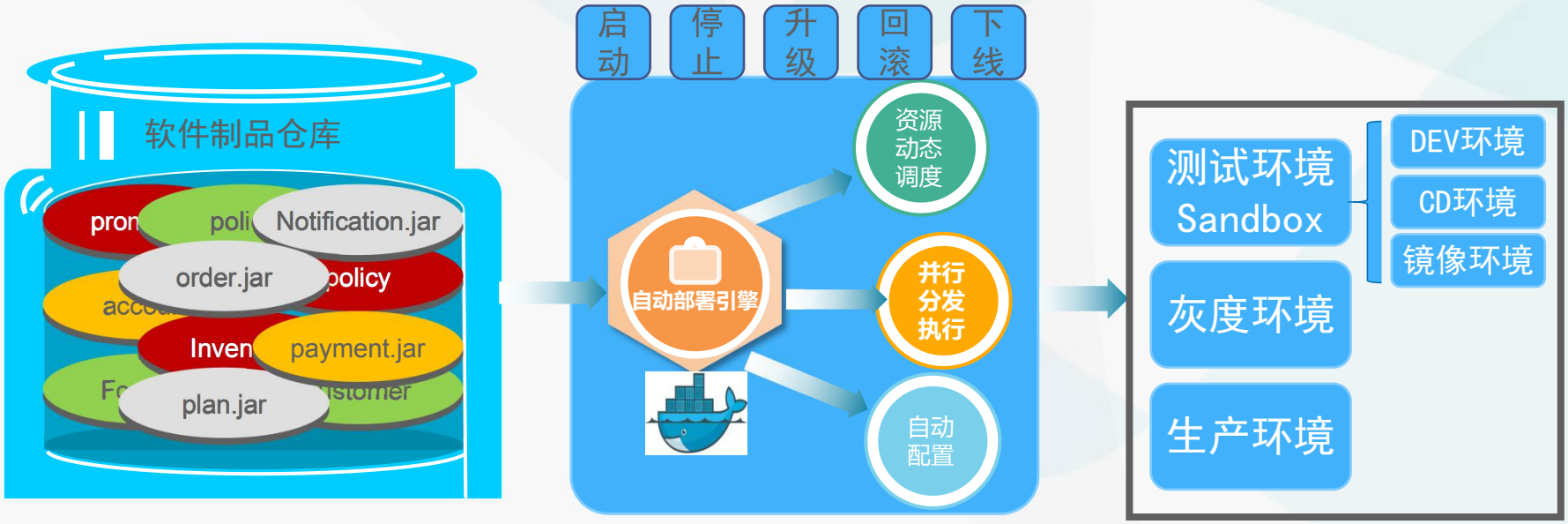
- 服务提供者专注于内部实现，而不是优先提供契约化的接口
- 担心接口变更，迟迟不提供接口契约，消费者无法并行开发
- 只重视快速开发，堆到最后联调测试发现问题

建议的原则：

- 接口优先
- 契约驱动测试，实现服务提供者和消费者解耦
- IDL，代码骨架自动生成
- 每日构建、持续集成联调，快速发现和解决问题
- 允许契约的变更，但项目组需就接口的兼容性做约束

#3 微服务的云化自动部署

1、服务独立打包，物理交付件独立
2、基础设施自动化，异构I层（PM/VM/Docker）
3、自动部署（部署引擎、并行分发和执行、Docker）
4、持续交付流水线（CD pipeline、sandbox）



微服务的大规模节点部署

量变到质变，企业级大规模部署需要考虑的问题：

1、如何支持多国多地部署？

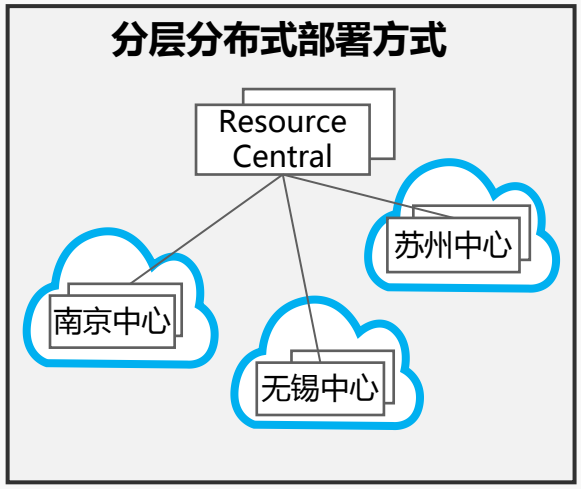
->分层分布式部署：资源分域、服务分组、AA容灾

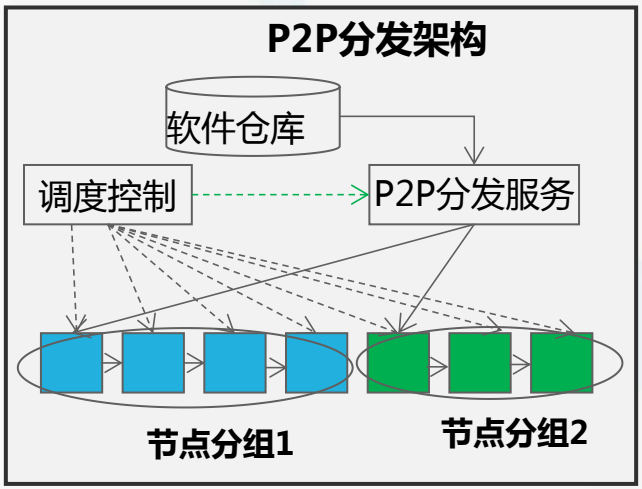
2、千级万级节点同时部署，带宽开销与节点数成倍增加？

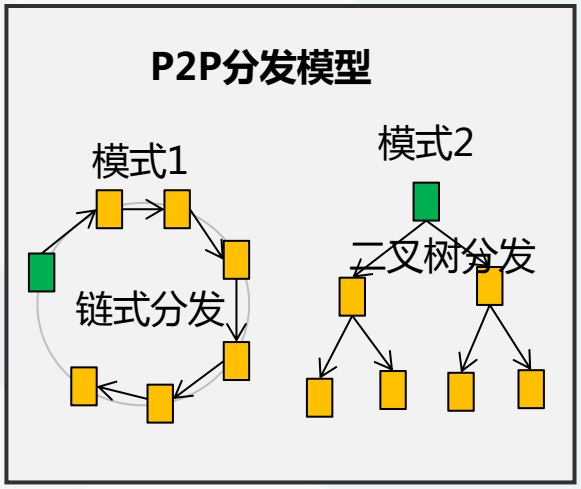
->分布式软件仓库、P2P高速软件分发

3、部署中遇到各种部分异常，如何处理？

->断点续传、断点续做、失败重做







微服务部署的实践总结

微服务部署原则：独立交付件、独立生命周期管理、基础设施自动化、部署自动化并行化

不好的实践：

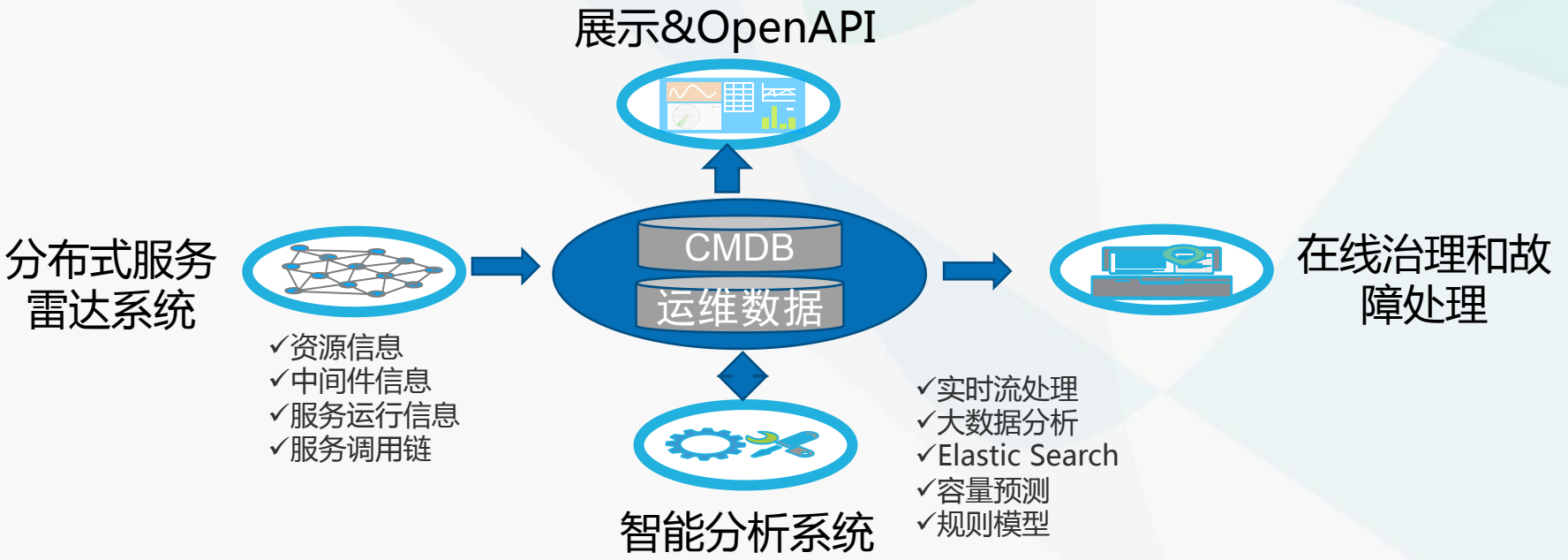
- 微服务混在一起部署，无法独立伸缩和启停
- 无法分批部署/升级，需要串行操作
- 无法优雅升级，升级中断业务

建议的原则：

- 微服务可独立发布、部署、升级、扩容、灰度
- 服务基于性能和成本可考虑合设，但核心服务和非核心服务隔离部署
- 基于docker部署，线上线下环境一致
- 部分失败不影响部署任务，支持可人工干预失败处理

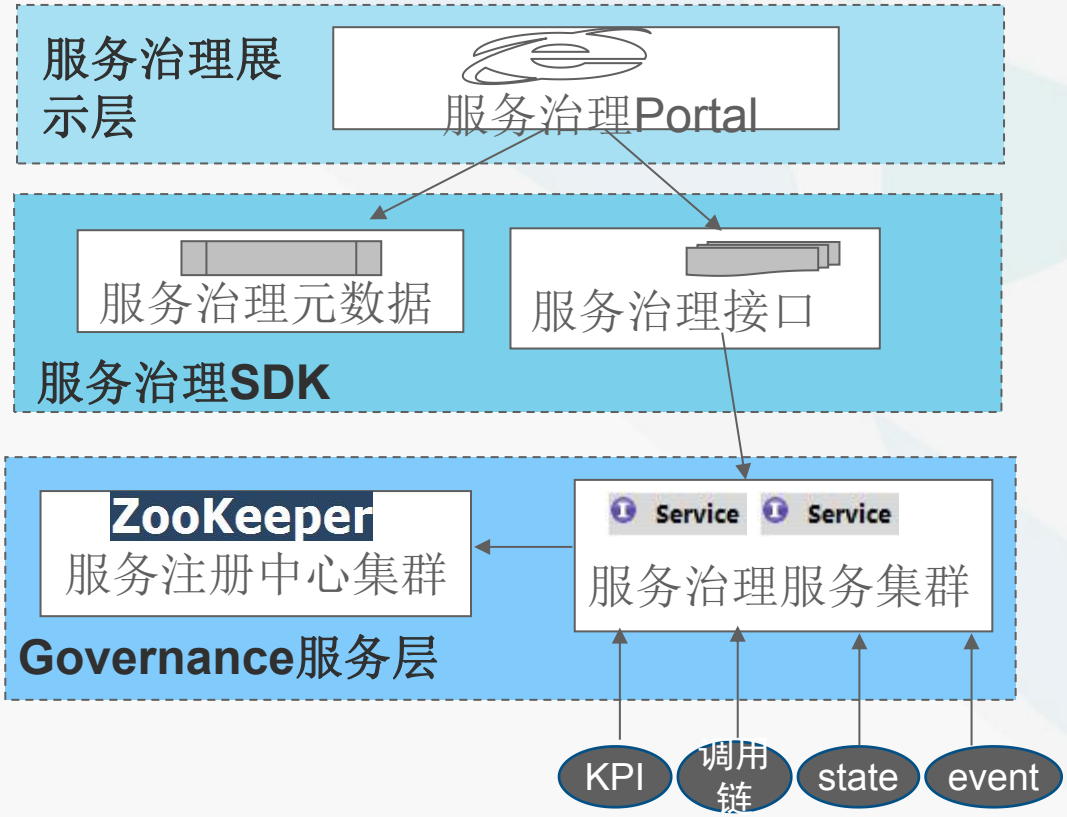
#4 微服务的治理和运维

1、微服务运维能力内置在服务框架中，框架化、自动化、实时化
2、运维数据：CMDB系统信息+实时动态运维数据+分析数据
3、服务调用链非常关键，并建立性能瓶颈分析、用户体验分析、故障分析模型和规则
4、一定考虑运维数据的对外开放性



微服务的在线治理架构

✓**服务自治，服务精细化管控**
✓**在线治理，动态实时生效**



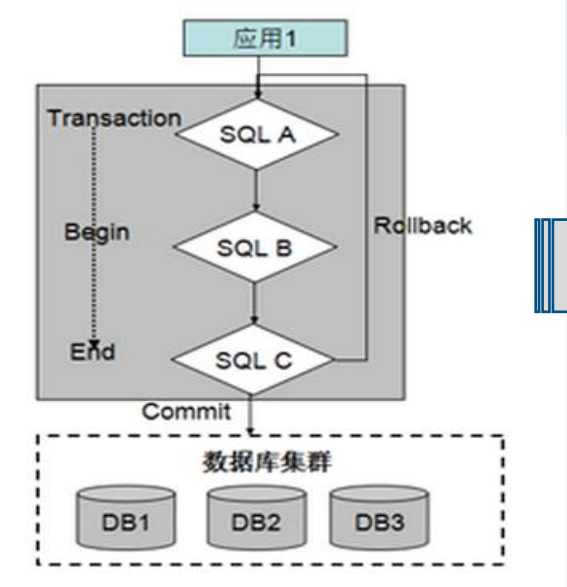
微服务治理策略：

1. 流量控制：动态、静态流控
2. 服务降级
3. 超时控制
4. 优先级调度
5. 服务智能路由
6. 服务实例接管/流量迁移
7. 服务迁入迁出
8. 调用链跟踪和分析
9. SLA策略控制
10. 服务伸缩...

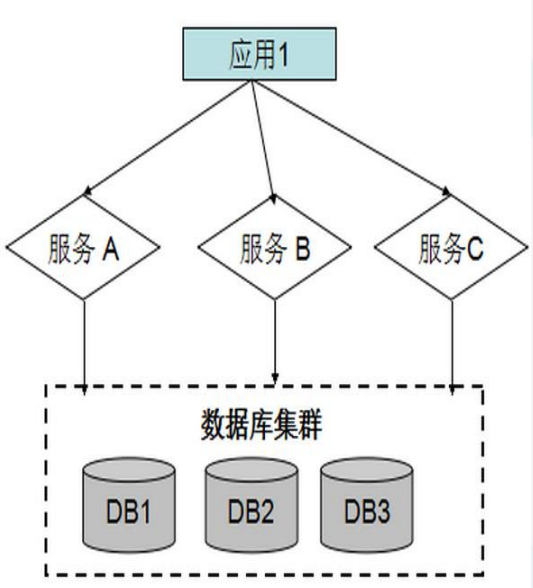
微服务运行需要考虑的问题：分布式事务

事务一致性：大部分采用最终一致性、极少部分需要强一致性

本地事务：



分布式事务：



策略：

1. 最终一致性，消息中间件
2. 强一致性，TCC（Try-Confirm-Cancel）

微服务运行需要考虑的问题：性能和时延

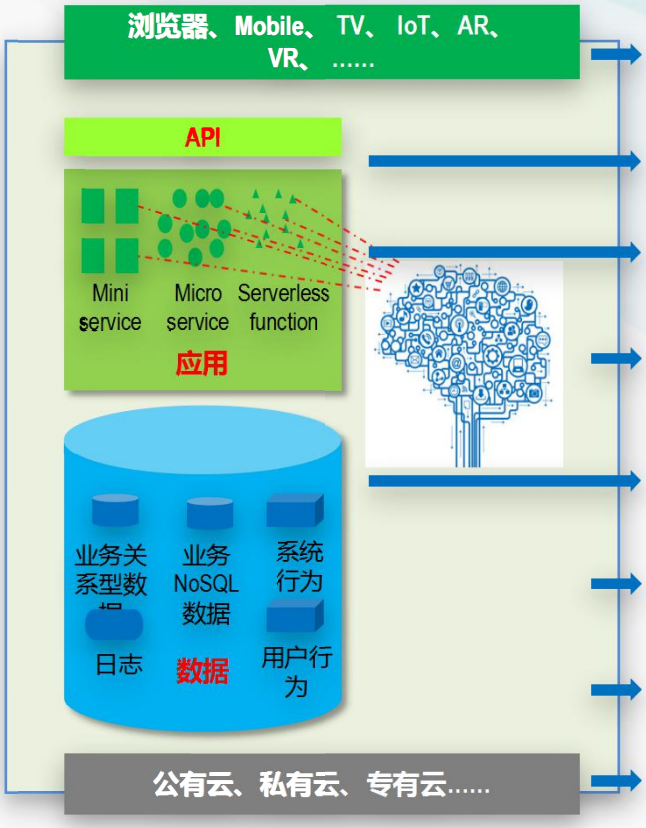
微服务技术上的优化策略：

1. P2P通信、长链接
2. 二进制协议
3. 码流压缩
4. 本地短路策略，本地-本机通信优先&优化
5. 同时支持同步/异步

业务上的策略：

1. 并行请求，减少等待
2. 服务分层和接口聚合，减少调用次数
3. 利用缓存，减少反复调用
4. 长调用等待，建议用异步消息机制

新的关注点



✓**Serverless**：从微服务到Serverless，应用被切分得越来越细，更高资源利用和调度，更易弹性、演进

✓**API First**：API成为云服务的核心要素，API驱动微服务设计和构建

✓**CD Pipeline**：以微服务为中心的自动持续交付流水线

✓**部署混合化 (Hybrid Cloud)**:企业使用混合的云环境来运行应用，满足其经济性、敏捷性和安全等方面的平衡

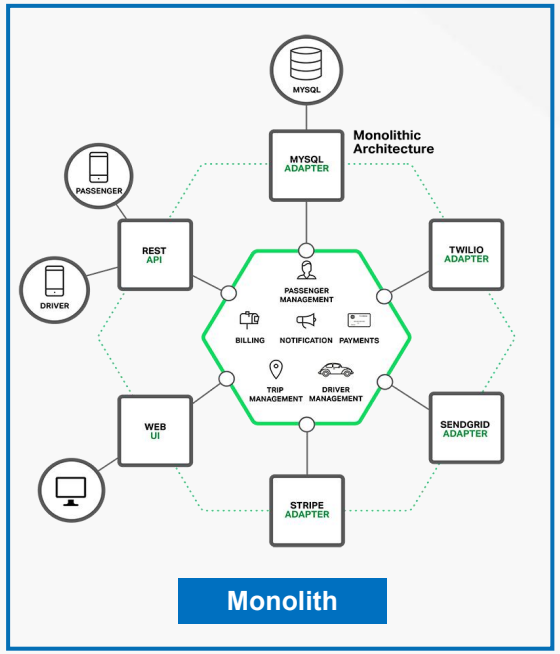
从微服务到Serverless的发展

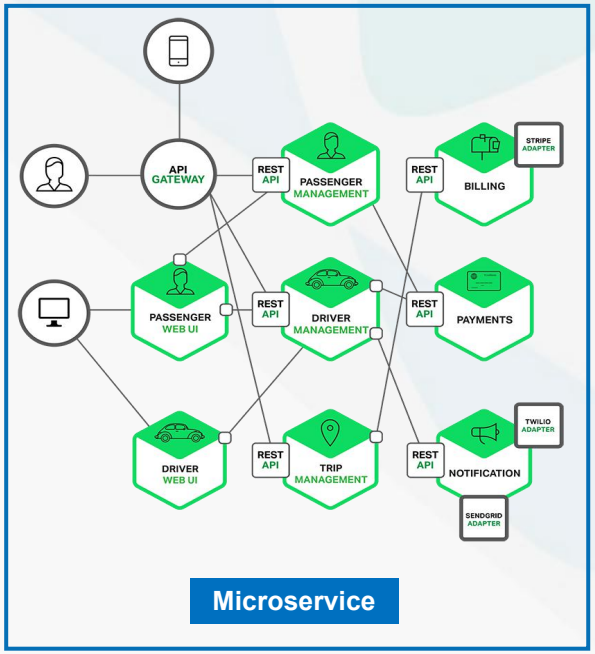
Serverlesss：“Run code, not servers, reducing IT operational costs and deploy services faster”

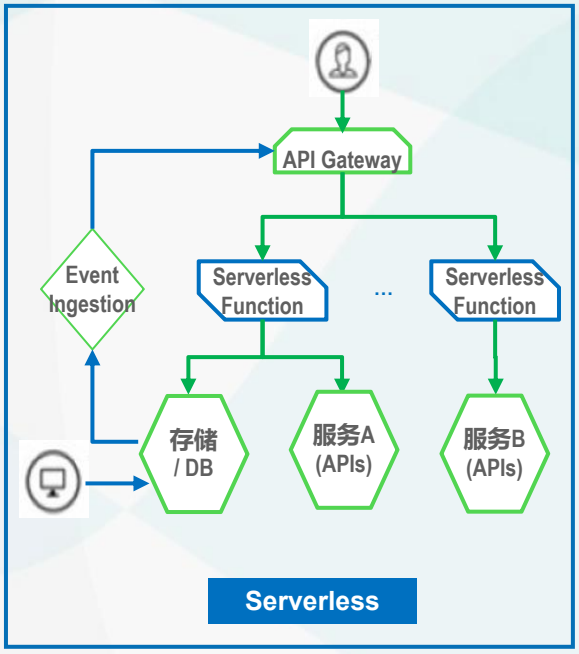
单体架构
Monolith application

微服务架构
Microservice application

Serverless架构
Nanoservice without Servers







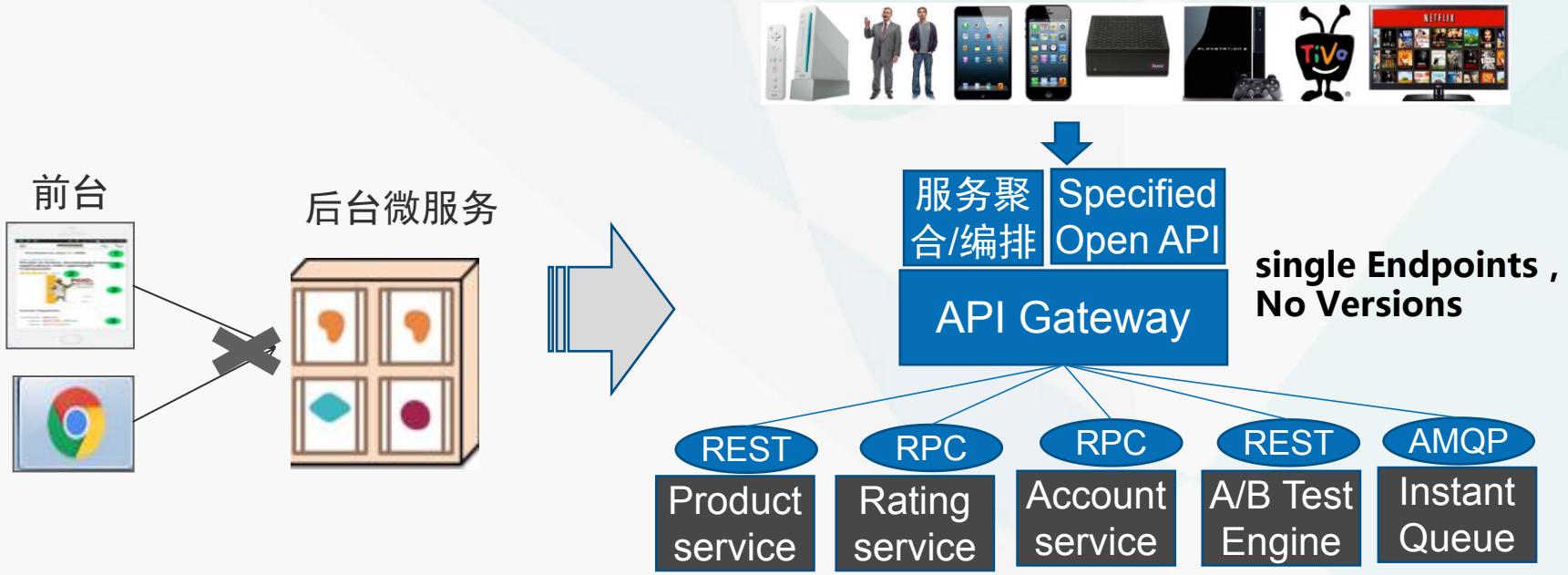
API GW+微服务的自然融合

API GW让微服务更纯净、更好适配性、开放性：

✓解决客户端/外部系统与微服务直接通信复杂的问题（网络性能、私有协议、多屏重复逻辑等）

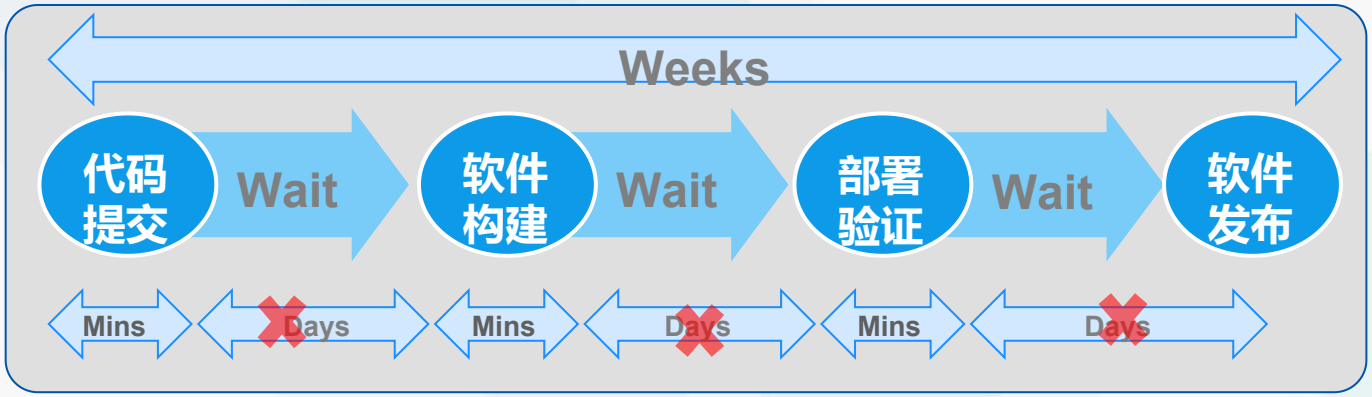
✓解决微服务间聚合编排的问题

✓屏蔽微服务异构协议、框架、多版本、重构问题

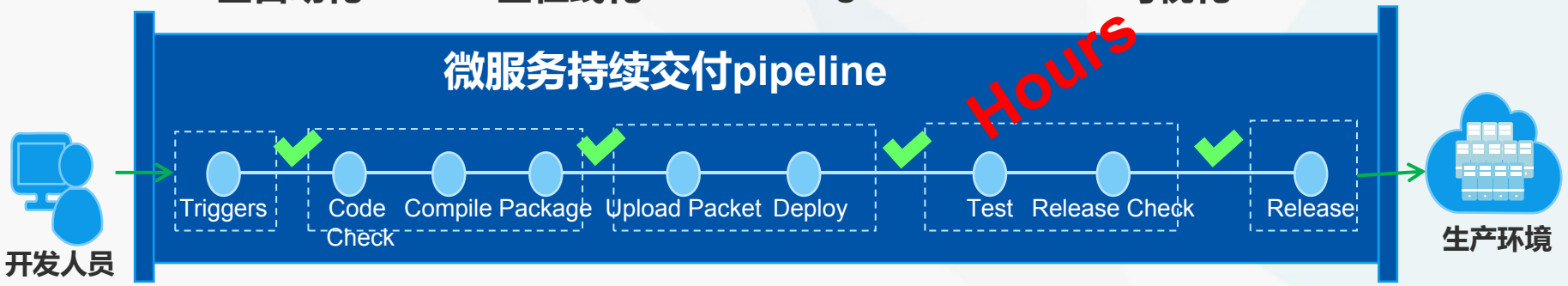


微服务CD Pipeline

Before 应用持续交付存在诸多互相等待、协同和复杂的配合



After 全自动化 全在线化 Stage标准化 可视化



最后的一些建议

1. 微服务化是个系统工程，技术保障和管理协同并进，架构&框架、组织结构、业务流程都要配套
2. 微服务的构建、部署、运维能力要构建齐备，否则微服务化之后复杂度倍增，效率并不能提升
3. 微服务不是银弹，有其使用的范围，对于数据处理为中心的逻辑不适合微服务化

QA

欢迎大家随时微信交流！
欢迎有意愿者加入华为软件云平台大家庭！

微信：miao-deer
邮箱：miaocaixia@huawei.com