唯品会应用系统架构设计思路和实践

唯品会
一家专门做特卖的网站

资深架构专家

2014年加盟唯品会，作为唯品会应用架构负责人，负责唯品会应用架构管理工作，主持公司架构评审运作；主持多个公司战略级项目的架构设计和支持工作；唯品会核心系统重构总架构师。

ebay

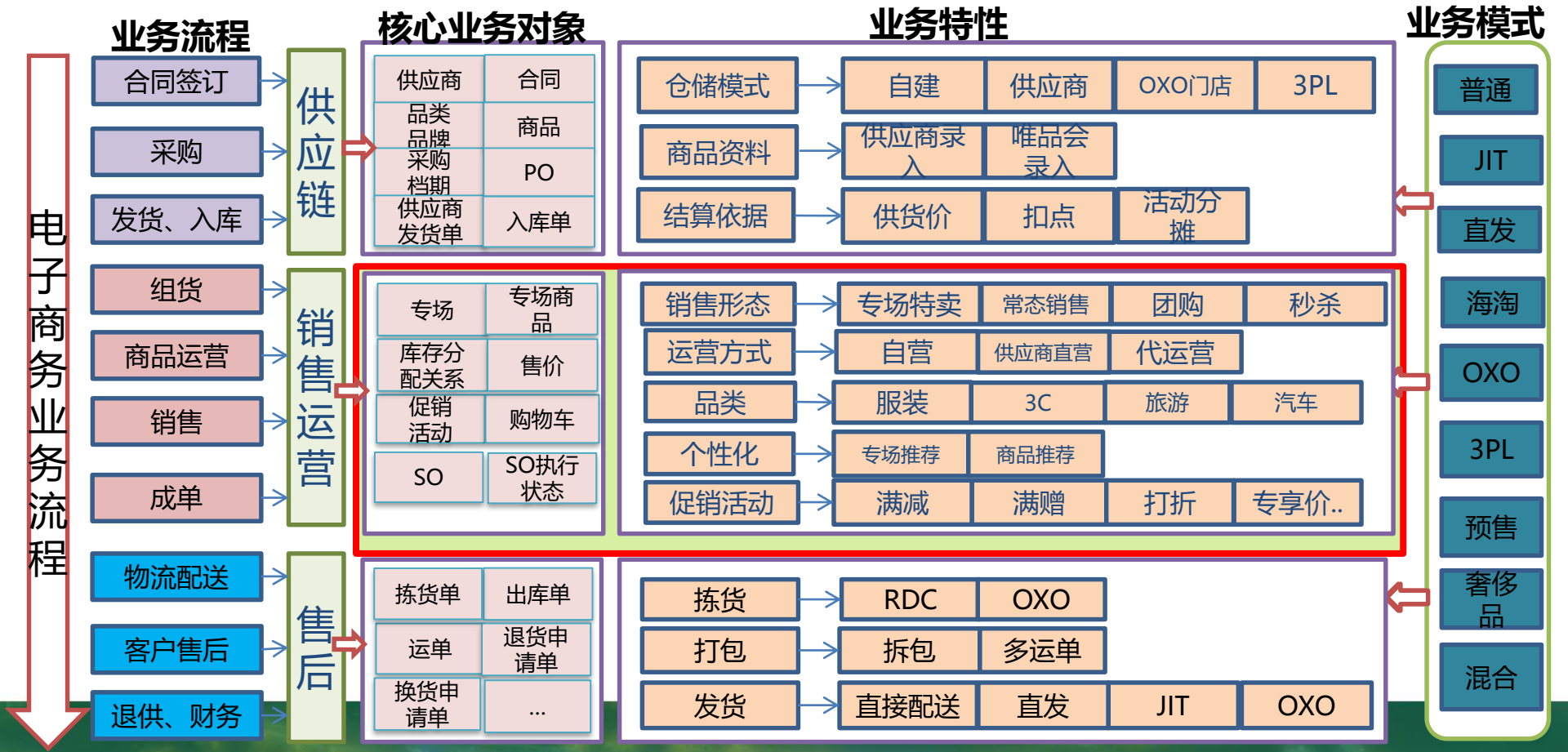
开发经理&Tech Leader

2014年前在eBay工作接近10年，负责eBay商城支付平台开发管理工作，PaaS平台架构师，电商平台系统Tech Leader

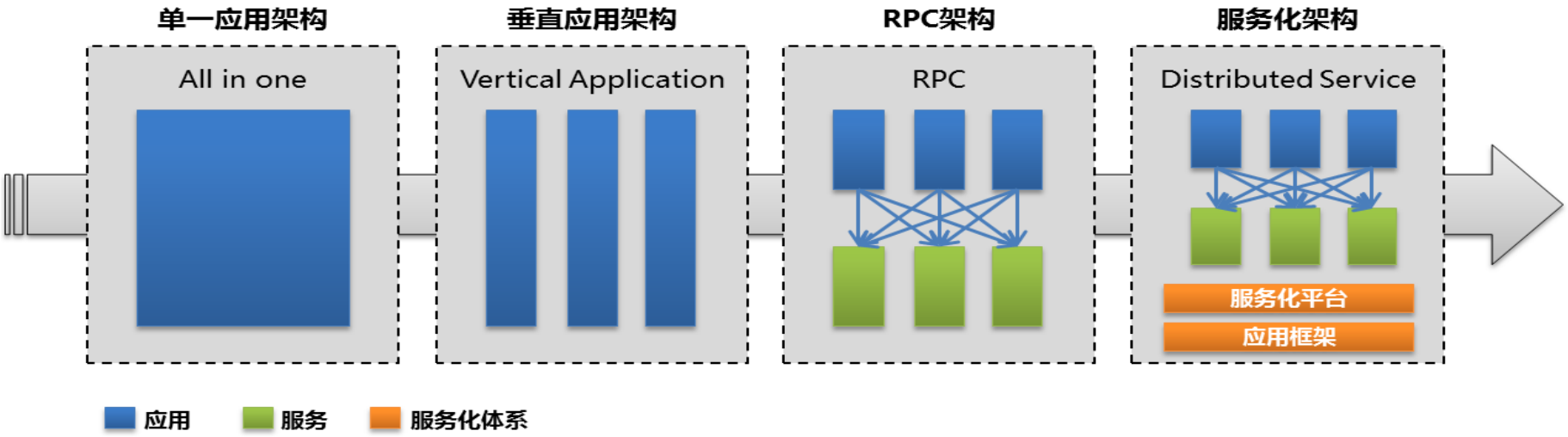
唯品会应用系统介绍

精选品牌正品 + 深度折扣 + 限时限量

唯品会业务流程图

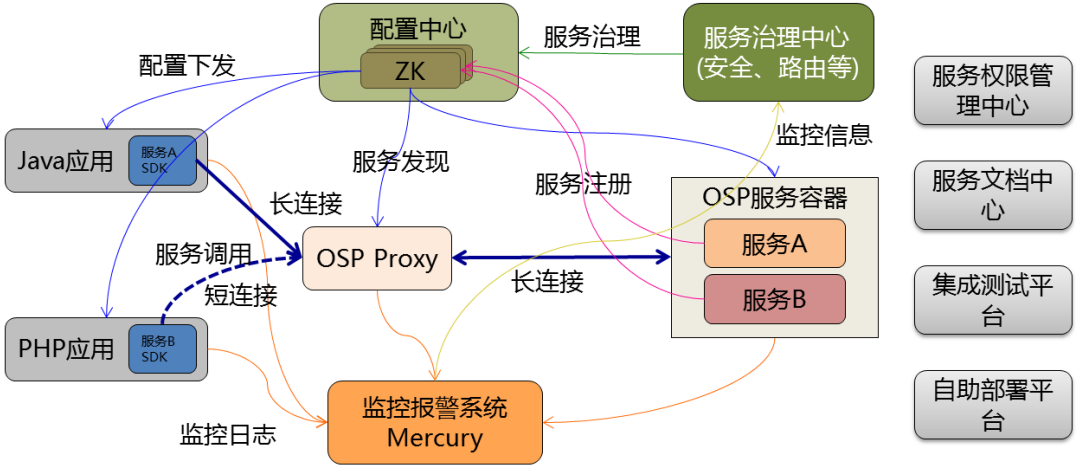

电子商务业务流程
业务流程
合同签订
采购
发货、入库
供应链
组货
商品运营
销售
成单
销售运营
物流配送
客户售后
退供、财务
售后
核心业务对象
供应商
合同
品类品牌
商品
采购档期
PO
供应商发货单
入库单
专场
专场商品
库存分配关系
售价
促销活动
购物车
SO
SO执行状态
拣货单
出库单
运单
退货申请单
换货申请单
...
业务特性
仓储模式
自建
供应商
OXO门店
3PL
商品资料
供应商录入
唯品会录入
结算依据
供货价
扣点
活动分摊
销售形态
专场特卖
常态销售
团购
秒杀
运营方式
自营
供应商直营
代运营
品类
服装
3C
旅游
汽车
个性化
专场推荐
商品推荐
促销活动
满减
满赠
打折
专享价..
拣货
RDC
OXO
打包
拆包
多运单
发货
直接配送
直发
JIT
OXO
业务模式
普通
JIT
直发
海淘
OXO
3PL
预售
奢侈品
混合

- 业务系统缺少长期规划，扩展困难
- 系统没有使用统一框架，效率低下
- 业务边界不清，责任不明确，耦合严重，效率低下
- 数据冗余严重，数据模型定义不严谨
- 系统可靠性低，治理能力弱,质量难以保证，运维成本高
- 服务颗粒度太粗,没有模块化设计,复用性差
- 系统可测试较差
- 容量扩展问题


单一应用架构
All in one
垂直应用架构
Vertical Application
RPC架构
RPC
服务化架构
Distributed Service
服务化平台
应用框架
应用
服务
服务化体系

唯品会基础架构体系

唯品会技术平台—基础架构



- **智能服务能路由**
  - ✓ 服务上线/下线
  - ✓ 机房选择
  - ✓ 灰度发布
  - ✓ 读写分离
  - ✓ 优雅降级
- **容错机制**
  - ✓ 超时控制
  - ✓ 容错重试
  - ✓ 熔断隔离
  - ✓ 服务限流
  - ✓ 优雅停止

海量日志消息的记录，告警与分析

IT运维 / 监控中心人员

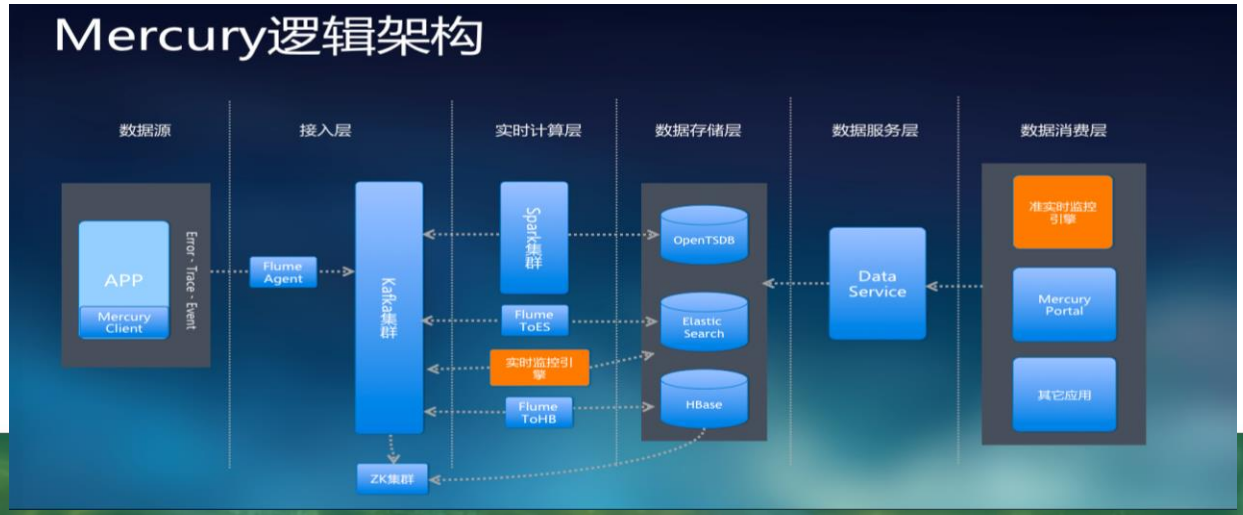
- 快速故障告警和问题定位
- 把握应用性能和容量评估
- 提供可追溯的性能数据

应用开发人员

- 定位线上服务性能瓶颈
- 持续优化代码和SQL
- 帮助快速解决线上问题

应用管理人员

- 全方位把握应用整体拓扑结构
- 定位全网应用瓶颈
- 帮助优化关键业务



架构设计思路

业务驱动

高可用性

高可扩展性和伸缩性

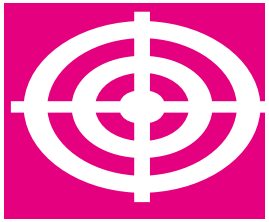
低成本

高性能

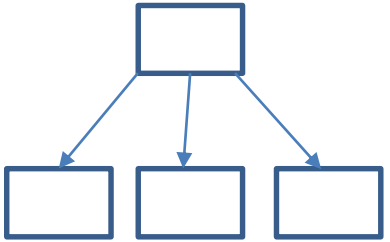
安全性

思路一

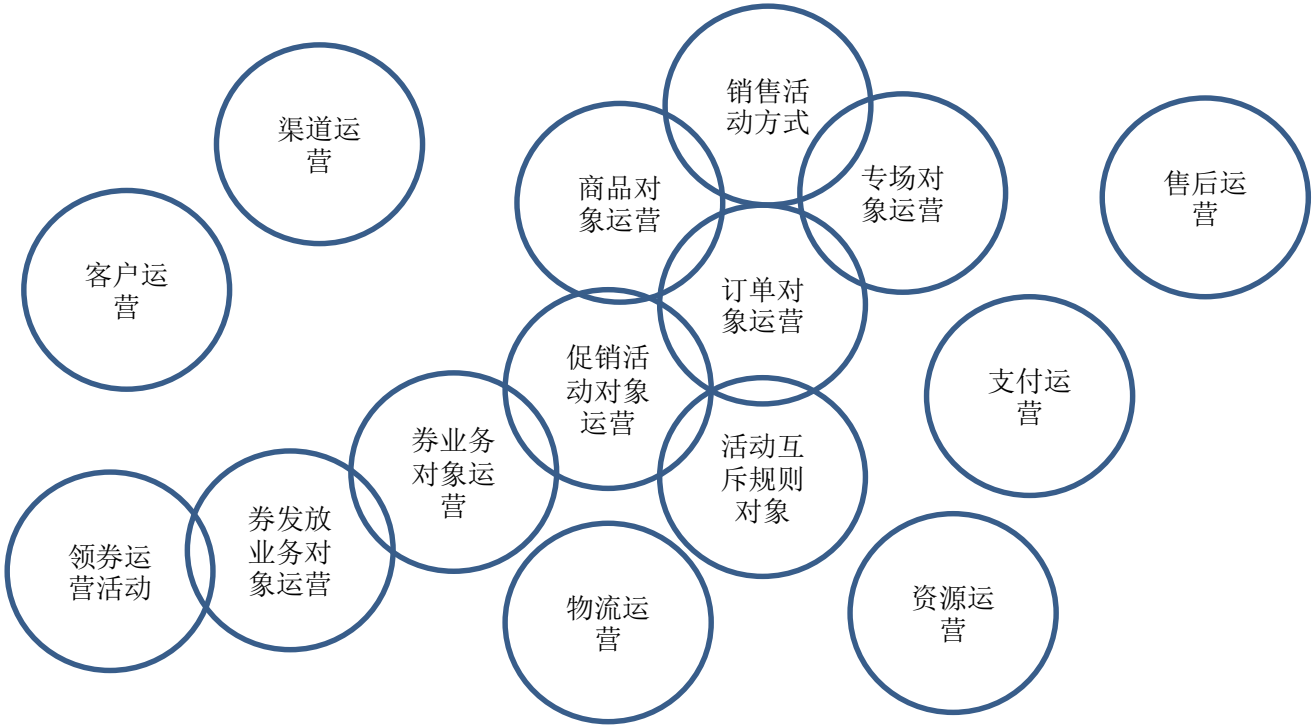
合理业务逻辑



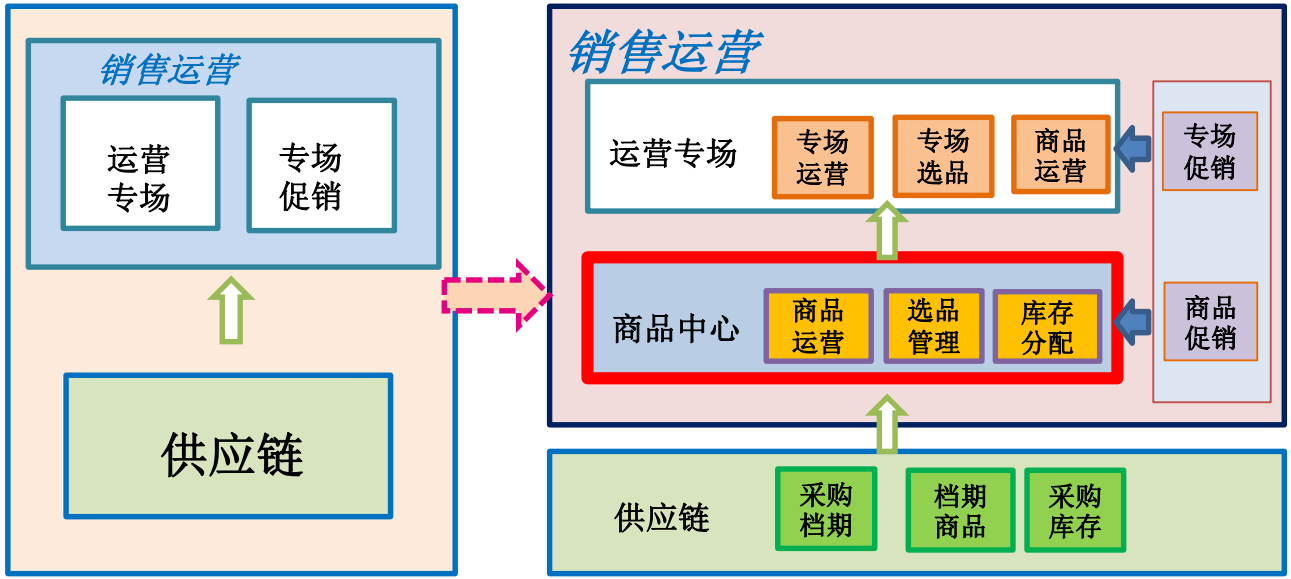
- 业务建模分析，通过对业务流程和用例进行分析，为每个业务对象定义合理的职责范围
- 根据业务建模结果，基于功能职责，进行垂直和水平分解，识别出业务功能或业务服务，将它们归类到子系统中相应模块中去
- 明确各业务域的定位，确定各个模块边界



核心业务对象分析规划


渠道运营
销售活动方式
商品对象运营
专场对象运营
售后运营
客户运营
订单对象运营
促销活动对象运营
支付运营
券业务对象运营
活动互斥规则对象
券发放业务对象运营
领券运营活动
物流运营
资源运营

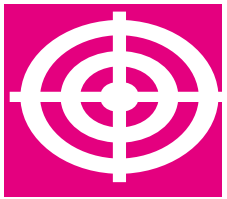
业务系统改造举例—改善销售运营选品



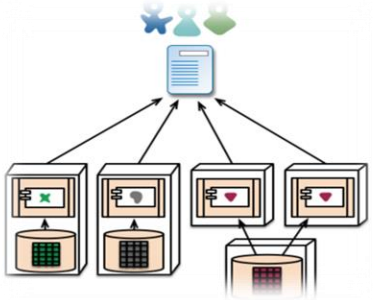
- 改变销售运营和供应链系统紧耦合关系
- 商品中心完成销售前的准备工作；
- 商品纬度的运营和促销
- 商品选品和库存分配管理

思路二

服务化解耦，提高系统复用性



- 系统分离成多个小的、相互独立的服务组成，这些服务运行在自己的进程中，开发和发布都没有依赖。可独立测试、独立部署、独立运行
- 新增的业务可以通过调用可复用的服务实现自身的业务逻辑
- 系统之间都是业务松耦合的，一个系统的修改并不会影响另外一个系统，为应用和服务的实现带来了更强的灵活性，缩短了服务交付周期简化了架构复杂度

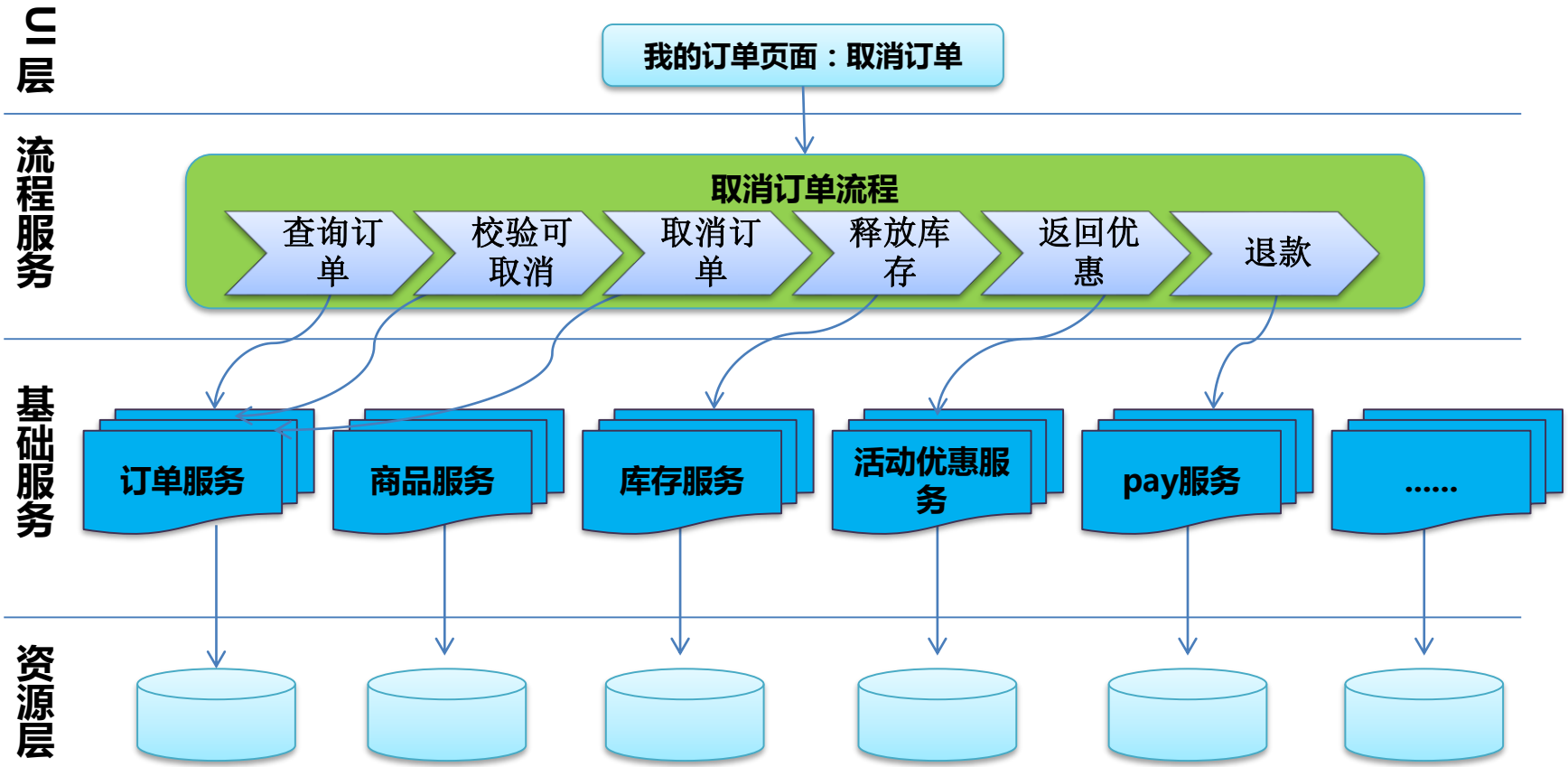


服务类型：

服务类型	注意事项	实例
中间层服务	✓ 中间层调用对应业务流程服务 ✓ 中间层不负责具体业务逻辑	移动中间层、不同业务系统之间dispatch服务
业务流程服务	✓ 避免直接访问数据库 ✓ 流程服务不能调用其它流程服务 ✓ 粗粒度服务	生成订单、取消订单、购物车操作等。
聚合服务	需要情况下创建聚合服务	专场查询聚合服务、价格聚合服务
业务基础服务	✓ 细粒度服务，可重用，灵活组合新的流程服务	比如商品查询／发布、订单查询、品牌查询、品类查询等

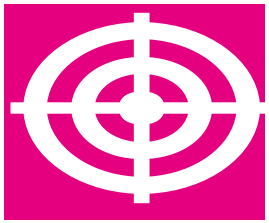

业务应用系统
订单中心
客服系统
CRM
支付
其它
Order.api(>200接口)
数据资源
DB
Redis
配置中心
服务治理
监控预警(Mercury)

服务示例：取消订单

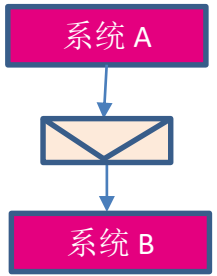

UI层
我的订单页面：取消订单
流程服务
取消订单流程
查询订单
校验可取消
取消订单
释放库存
返回优惠
退款
基础服务
订单服务
商品服务
库存服务
活动优惠服务
pay服务
.....
资源层

思路三

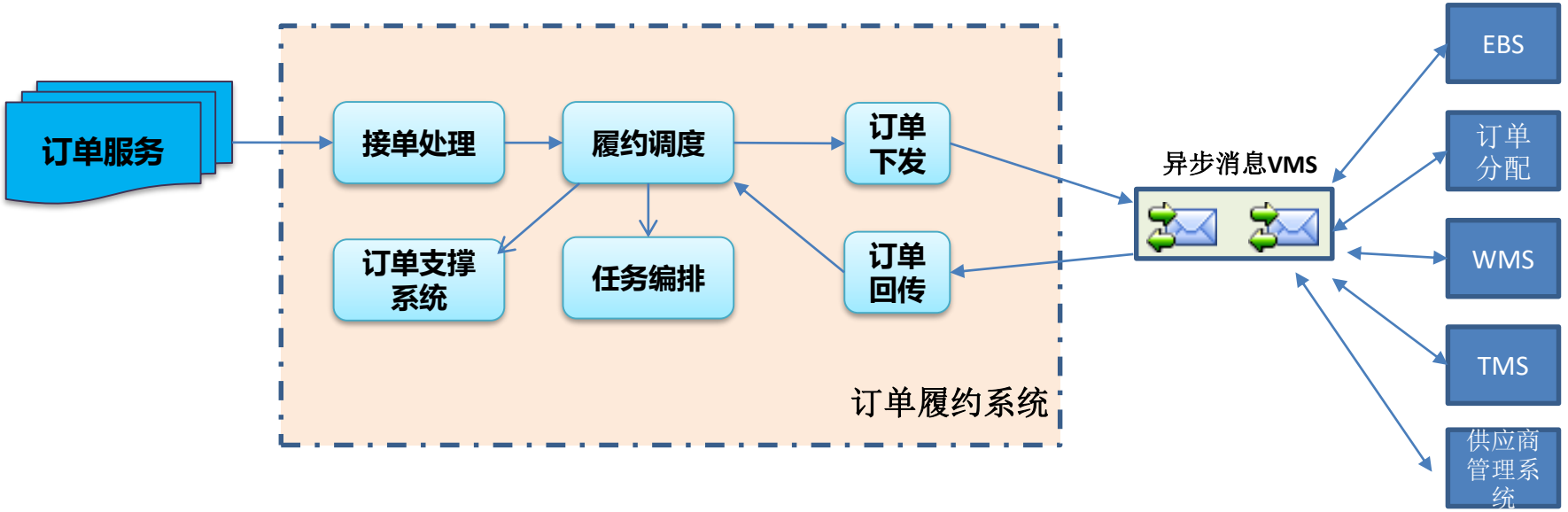
系统间通讯增加异步处理，减少同步处理



- 通过异步调用通知非主要流程，加快了系统主要业务流程的反应速度和性能。
- 异步调用实现系统隔离解耦，缓冲上游系统冲击，保护下游系统
- 分布式异步消息队列服务器可在宕机后确保消息不丢失，从而提高系统可用性、健壮性和扩展性
- 实现事务中的最终一致性

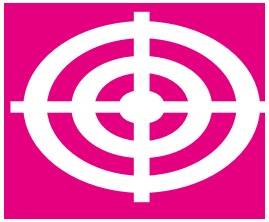


异步消息示例：订单履约

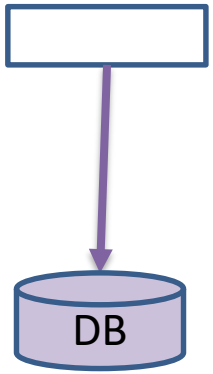

订单服务
接单处理
履约调度
订单下发
订单支撑系统
任务编排
订单回传
订单履约系统
异步消息VMS
EBS
订单分配
WMS
TMS
供应商管理系统

思路四

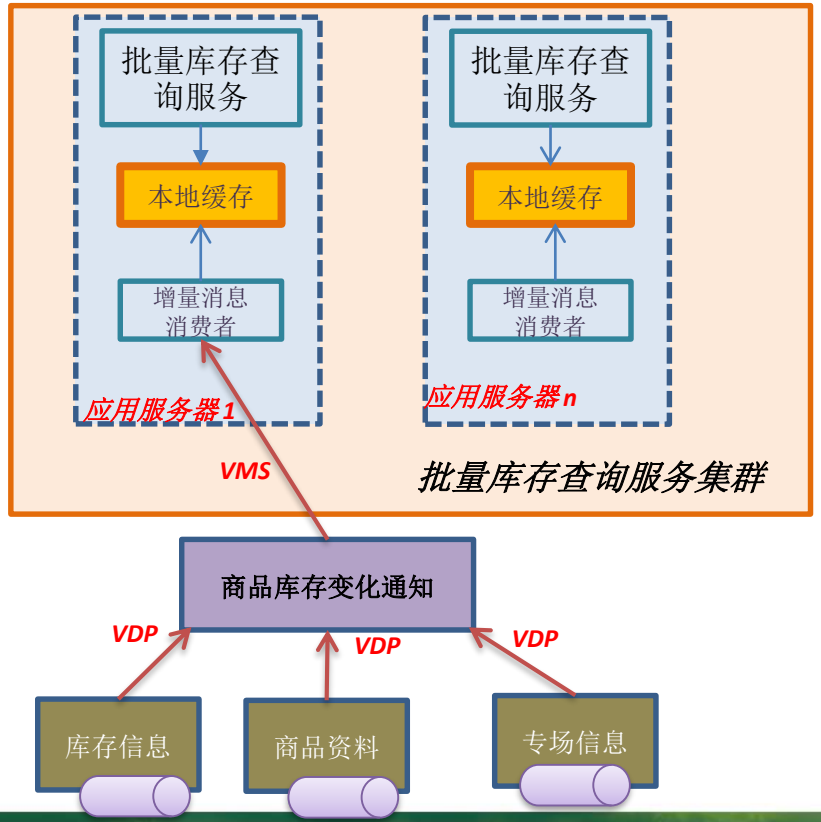
优化数据访问



- 访问量大的数据库做读写分离
- 数据库有能力支撑时，尽量不要引入缓存
- 合理利用缓存提高访问性能
- 访问量和数据量都很大的数据库，通过数据库分库分表
- 不同业务域数据库做分区隔离
- 重要数据配置备库



优化数据访问例子



背景：

- ✓ 商品数量多，性能要求高，数据库或分布式缓存查询难以满足性能要求
- ✓ 库存变化频繁

解决方案：

- ✓ 通过本地缓存保存全量库存信息
- ✓ 库存变化通过数据推送器vdp触发异步消息
- ✓ 应用服务器消费库存变动消息并同步本地缓存
- ✓ 批量查询服务直接查询本地缓存获取库存信息

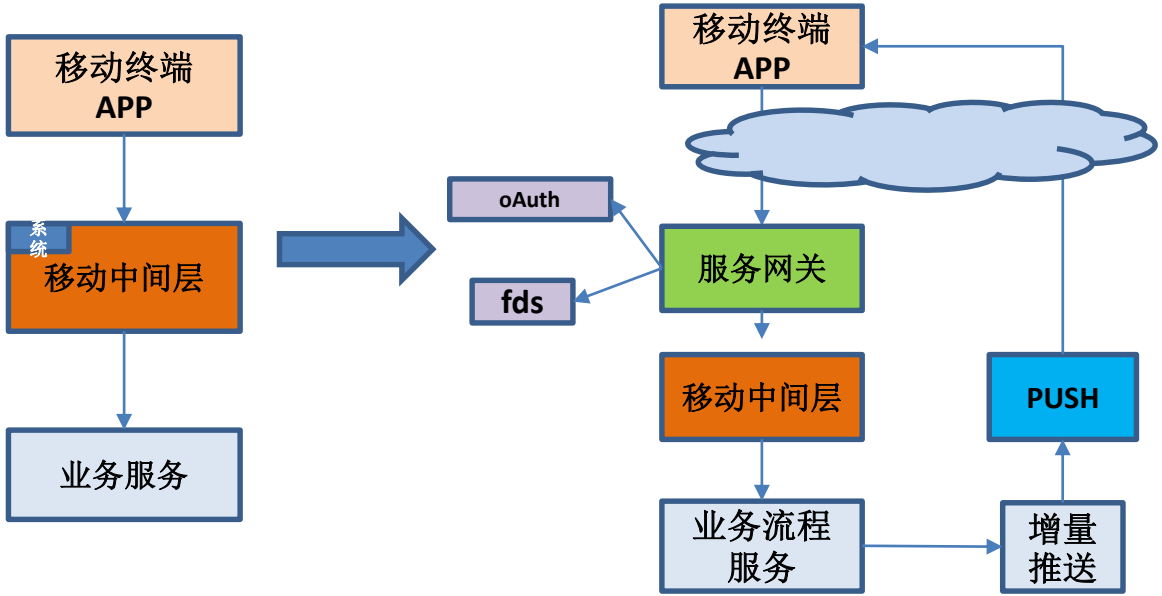
思路五

统一数据标准，减少数据冗余

- 明确各关键信息项目的定义并在全公司范围内推行
- 统一数据字典，便于各系统固定编码定义的统一
- 规范化数据模型设计，遵循3范式要求进行设计，并约一的技术字段
- 明确数据归档定义
- 明确数据维护范围，减少系统间拷贝

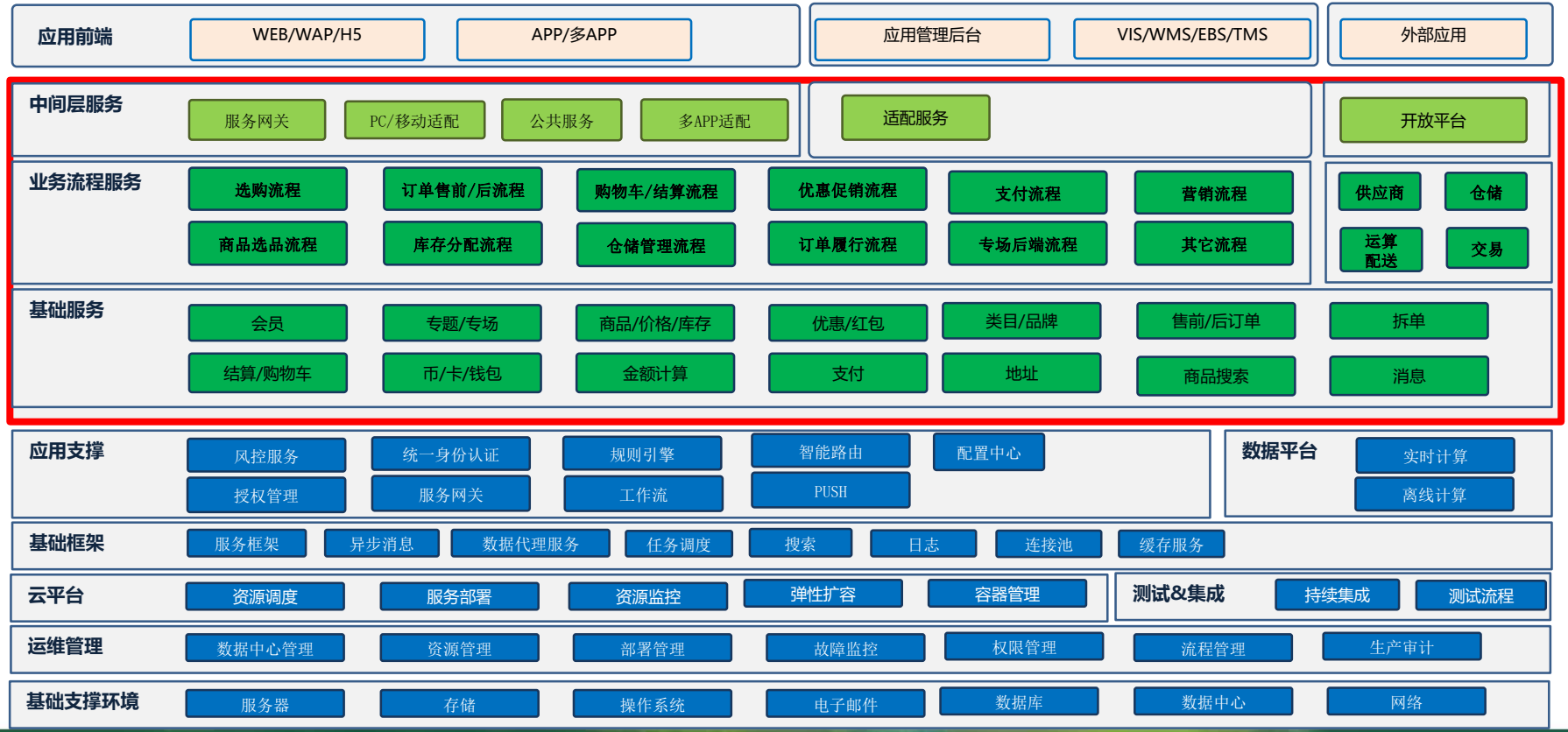
思路六

优化移动APP接入，引入服务网关



- 服务网管负责公共逻辑，如协议适配、安全策略、流量管理、熔断机制、降级容错、智能路由、监控
- 中间层业务逻辑下沉，简化中间层业务逻辑
- 由业务流程服务提供对应聚合服务
- 增量数据通过推送中心推送到移动APP

唯品会电商系统应用整体架构



谢谢！