支付宝系统架构（内部架构图）

2016-09-11 支付圈

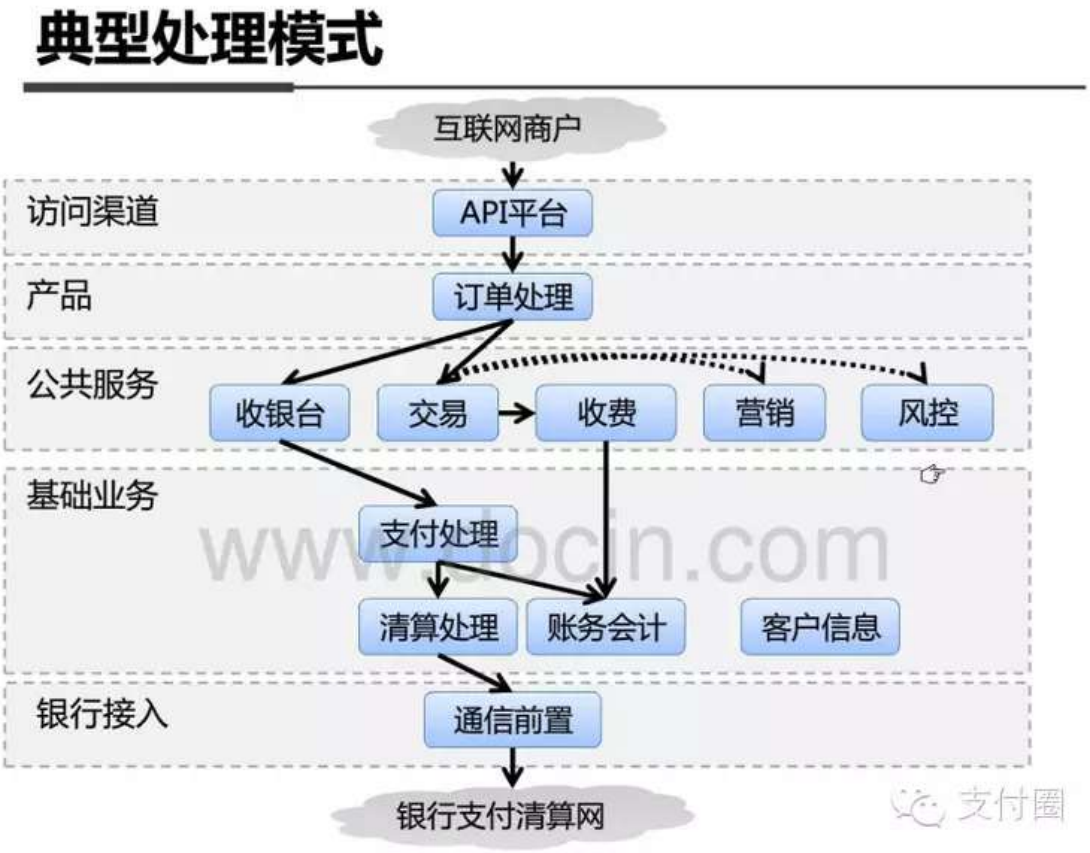
前言

支付宝是中国支付行业的一个标兵，无论是业务能力还是产品创都引领者中国支付行业的前沿，作为支付业务的基础系统的复杂性和稳定性是支付业务是否能够及时快速安全处理的根本，本期支付圈收集了支付宝的系统架构图包含：清算 客服 处理 资金 财务 等等 供其他支付公司进行参考！

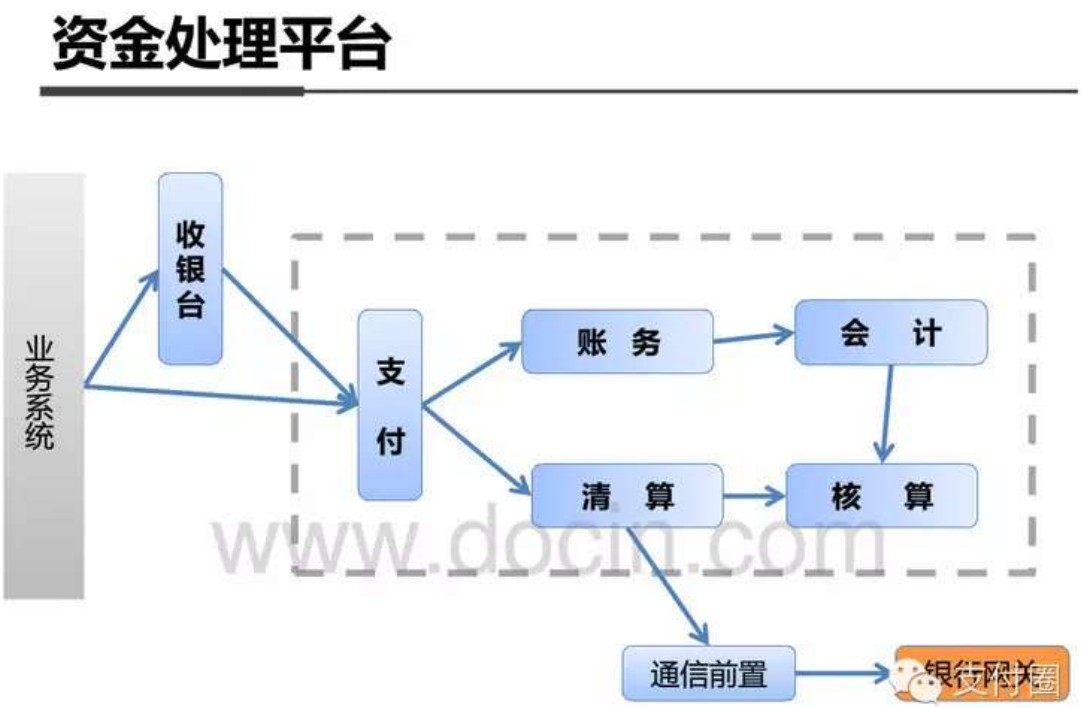
本文为网络收集信息，虽然不属于支付宝的最新系统架构信息但是作为支付行业的龙头，架构系统依然值得学习！

支付宝系统架构概况

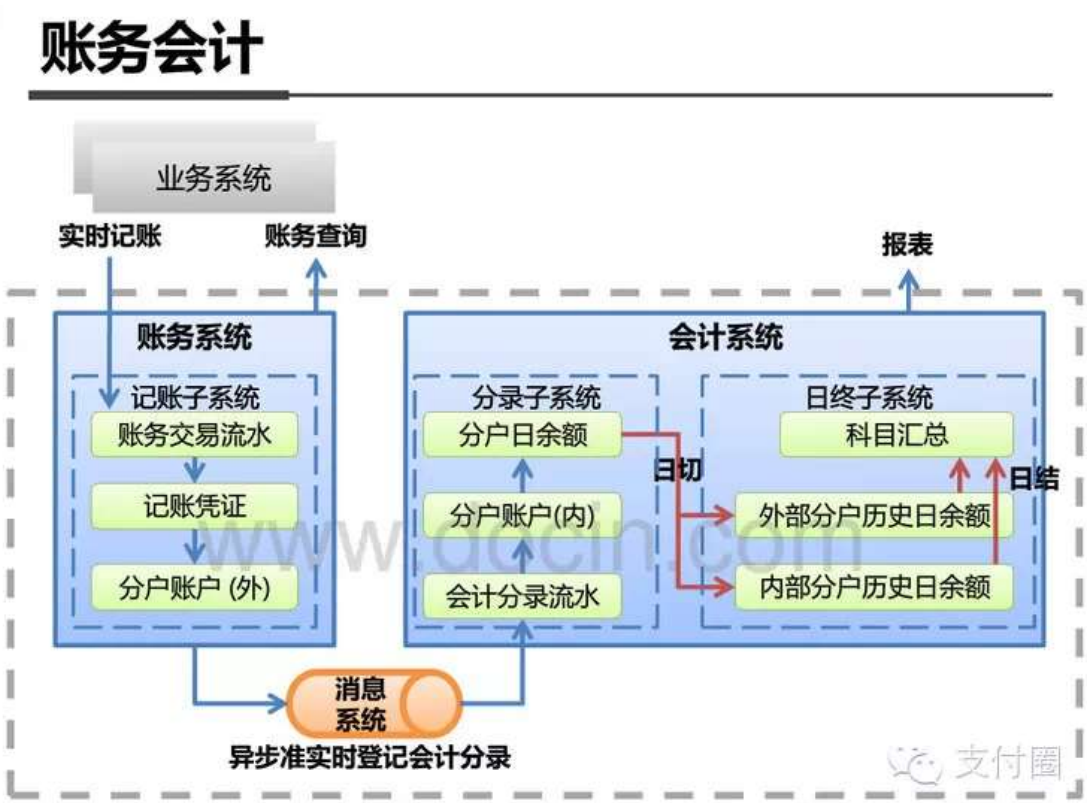
典型处理默认



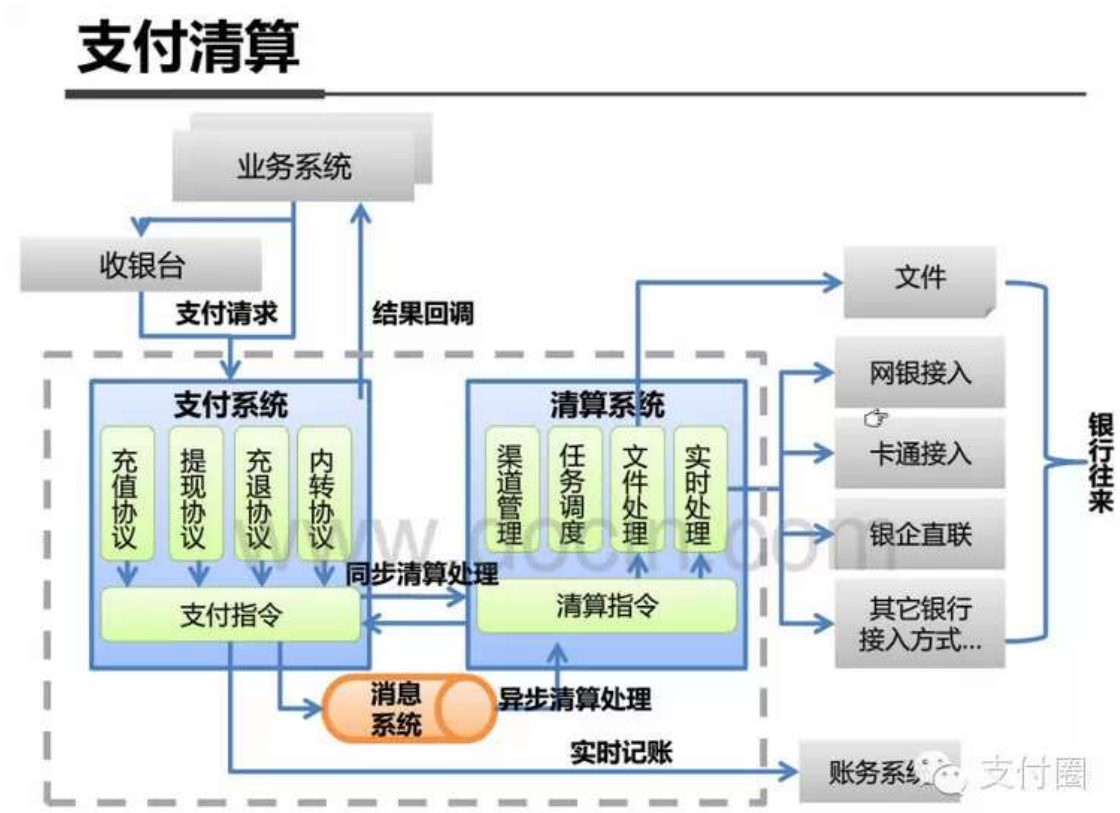
资金处理平台


资金处理平台
业务系统
收银台
支付
账 务
会 计
清 算
核 算
通信前置
银行网关
www.docin.com

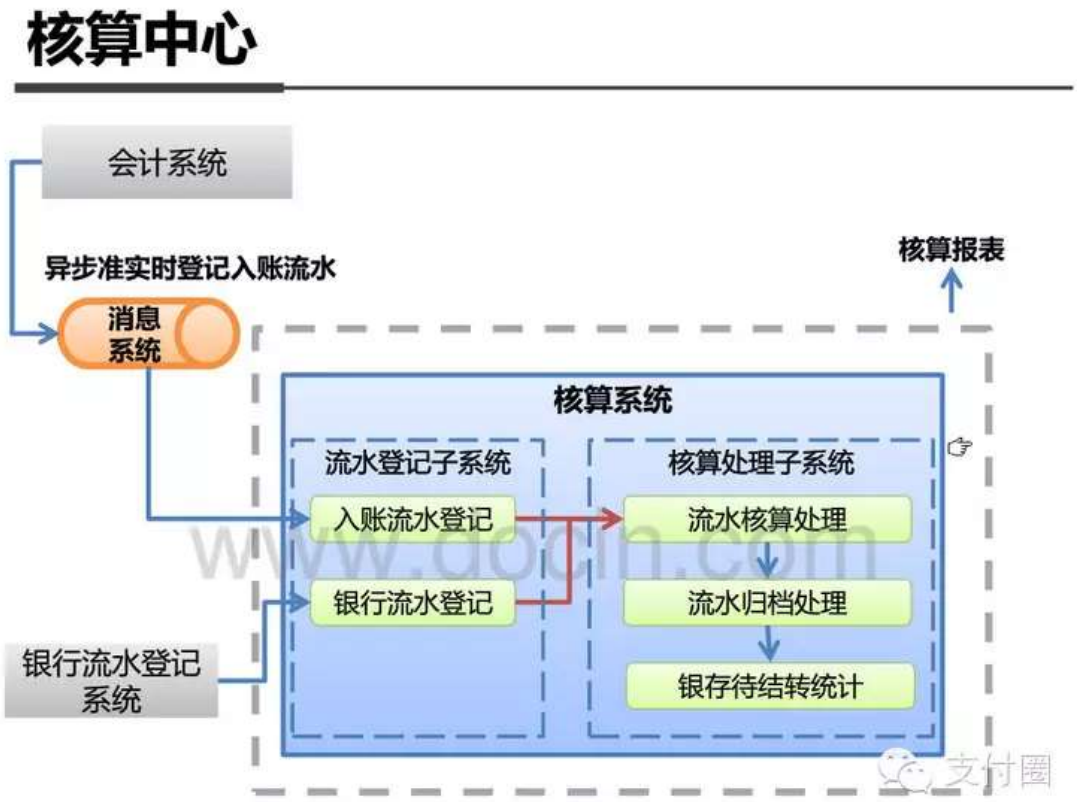
财务会计


账务会计
业务系统
实时记账
账务查询
报表
账务系统
记账子系统
账务交易流水
记账凭证
分户账户 (外)
会计系统
分录子系统
分户日余额
分户账户(内)
会计分录流水
日终子系统
科目汇总
外部分户历史日余额
内部分户历史日余额
日切
日结
消息系统
异步准实时登记会计分录
www.docin.com
支付圈

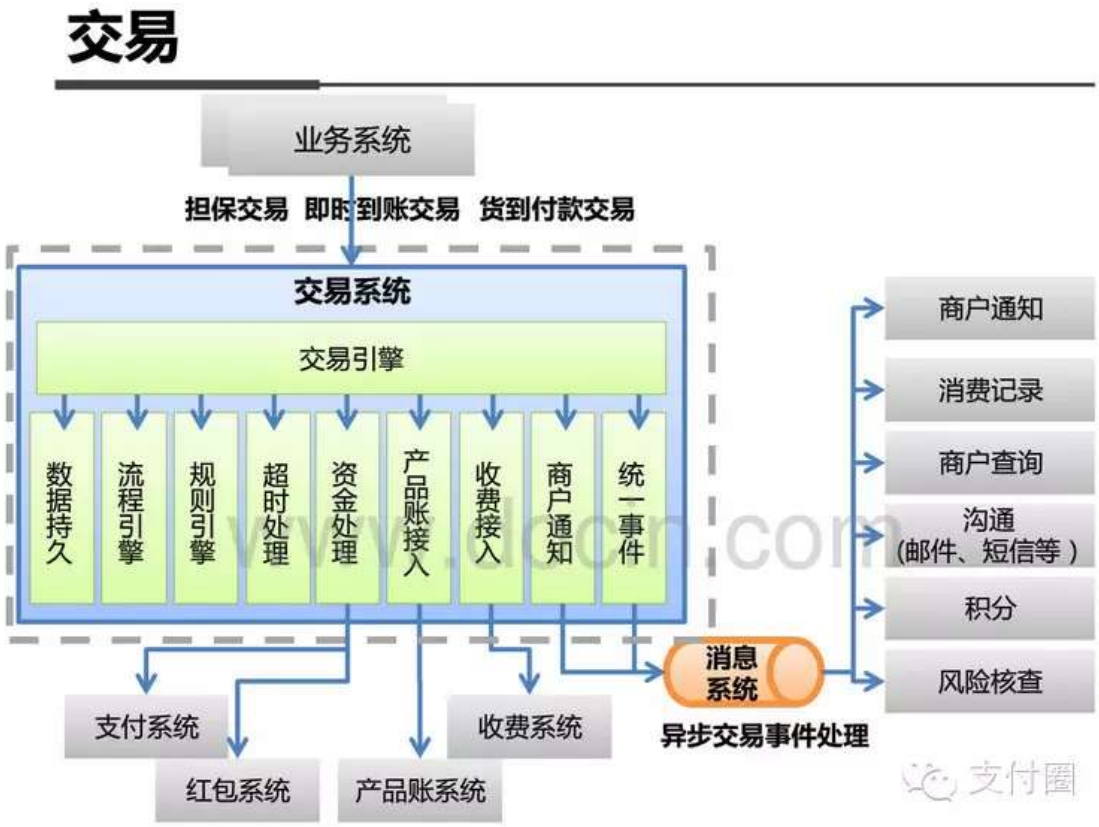
支付清算


支付清算
业务系统
收银台
支付请求
结果回调
支付系统
充值协议
提现协议
充退协议
内转协议
支付指令
同步清算处理
清算系统
渠道管理
任务调度
文件处理
实时处理
清算指令
消息系统
异步清算处理
实时记账
文件
网银接入
卡通接入
银企直联
其它银行接入方式...
银行往来
支付圈

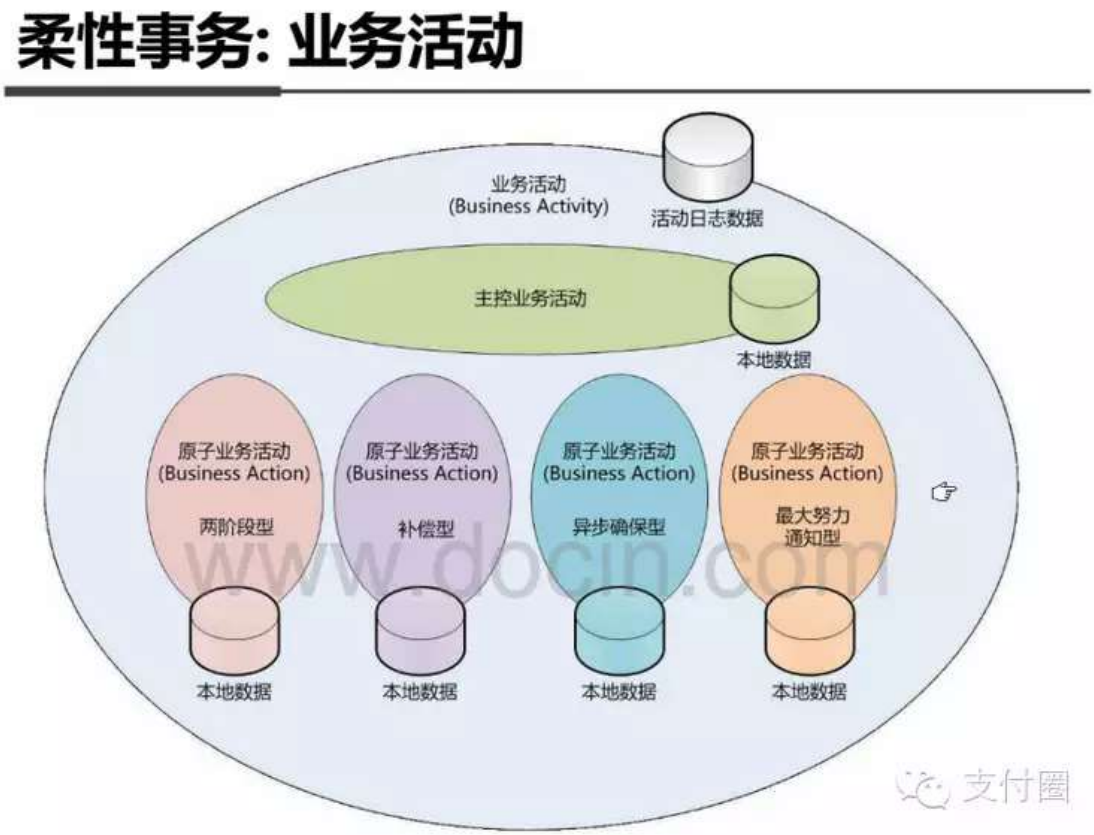
核算中心


核算中心
会计系统
异步准实时登记入账流水
消息系统
核算报表
核算系统
流水登记子系统
入账流水登记
银行流水登记
核算处理子系统
流水核算处理
流水归档处理
银存待结转统计
银行流水登记系统
支付圈

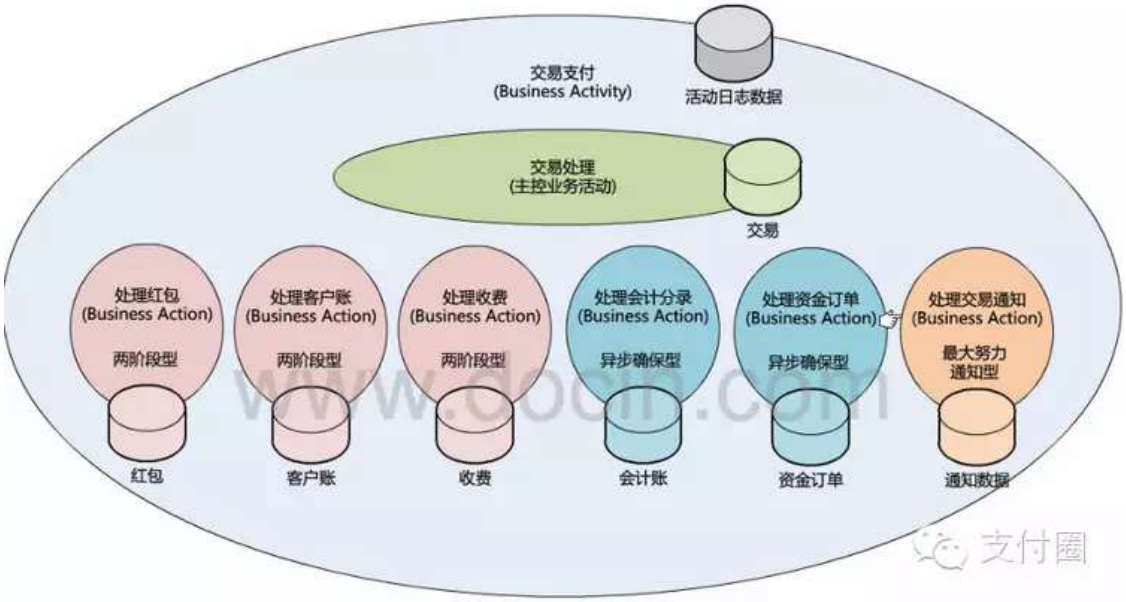
交易


交易
业务系统
担保交易 即时到账交易 货到付款交易
交易系统
交易引擎
数据持久
流程引擎
规则引擎
超时处理
资金处理
产品账接入
收费接入
商户通知
统一事件
商户通知
消费记录
商户查询
沟通
(邮件、短信等)
积分
风险核查
消息系统
异步交易事件处理
支付系统
收费系统
红包系统
产品账系统
支付圈

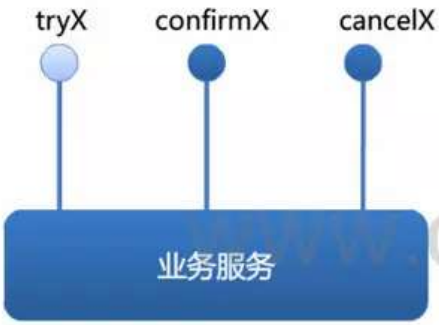
柔性事务


柔性事务: 业务活动
业务活动
(Business Activity)
活动日志数据
主控业务活动
本地数据
原子业务活动
(Business Action)
两阶段型
原子业务活动
(Business Action)
补偿型
原子业务活动
(Business Action)
异步确保型
原子业务活动
(Business Action)
最大努力
通知型
本地数据
本地数据
本地数据
本地数据
支付圈

柔性事务: 业务活动举例



柔性事务: TCC型业务服务



Try: 尝试执行业务

- 完成所有业务检查(一致性)
- 预留必须业务资源(准隔离性)

Confirm:确认执行业务

- 真正执行业务
- 不作任何业务检查
- 只使用Try阶段预留的业务资源
- Confirm操作满足幂等性

Cancel: 取消执行业务

- 释放Try阶段预留的业务资源
- Cancel操作满足幂等性

柔性事务: TCC服务事务协调模式

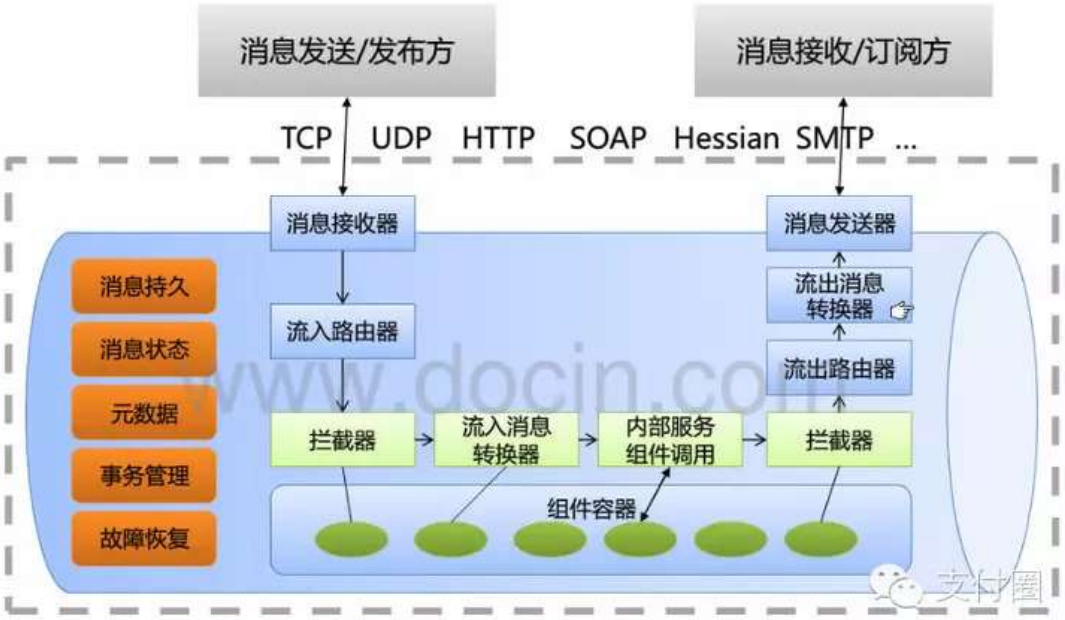
实现

- 一个完整的业务活动由一个主业务服务与若干从业务服务组成
- 主业务服务负责发起并完成整个业务活动
- 从业务服务提供TCC型业务操作
- 业务活动管理器控制业务活动的一致性，它登记业务活动中的操作，并在业务活动提交 时确认所有的TCC型操作的confirm操作，在业务活动取消时调用所有TCC型操作的cancel操作

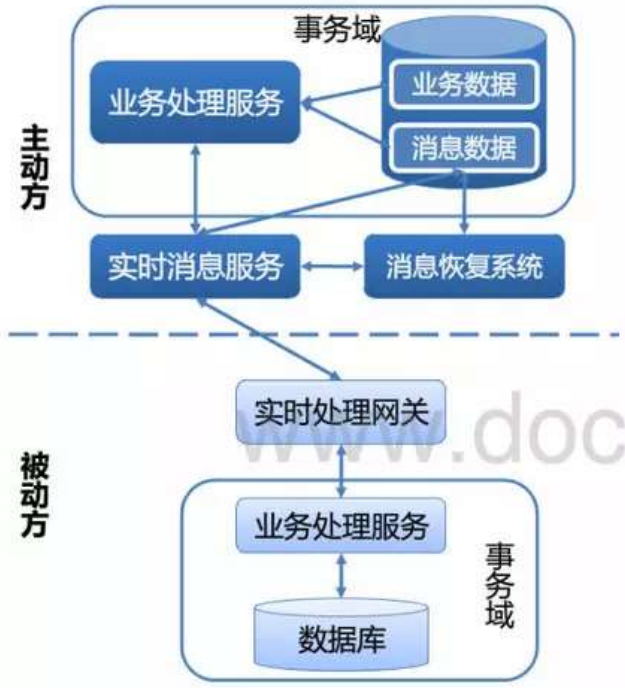
适用范围

- 强隔离性、严格一致性要求的业务活动
- 适用于执行时间较短的业务

消息系统



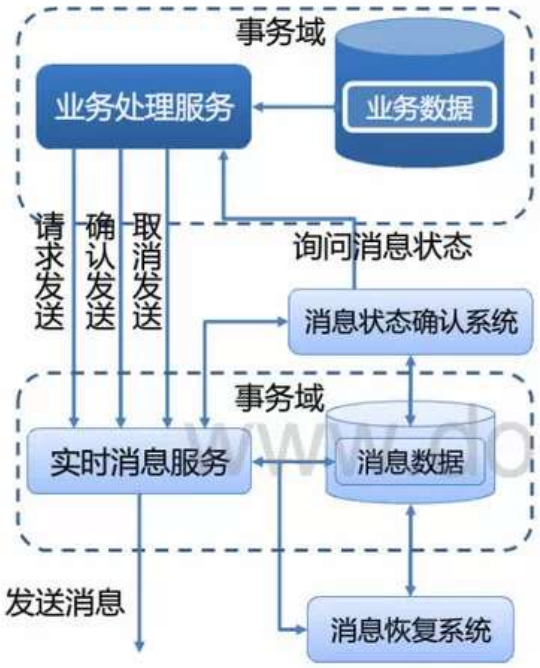
消息系统: 消息事务模式(1)



实现

- 业务活动的主动方，在完成业务处理的同一个本地事务中，记录消息数据
- 业务处理事务提交后、通过实时消息服务通知业务被动方，实时通知成功后删除消息数据
- 消息恢复系统定期找到未成功发送的消息，交给实时消息服务补发送

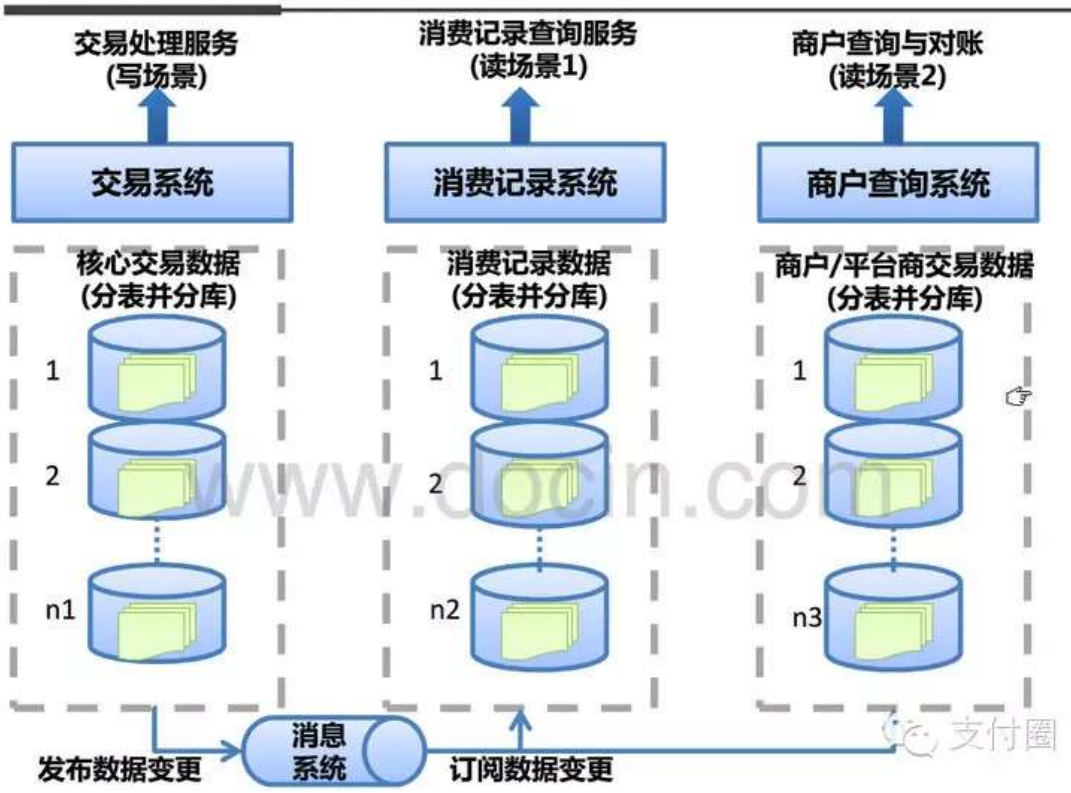
消息系统: 消息事务模式(2)



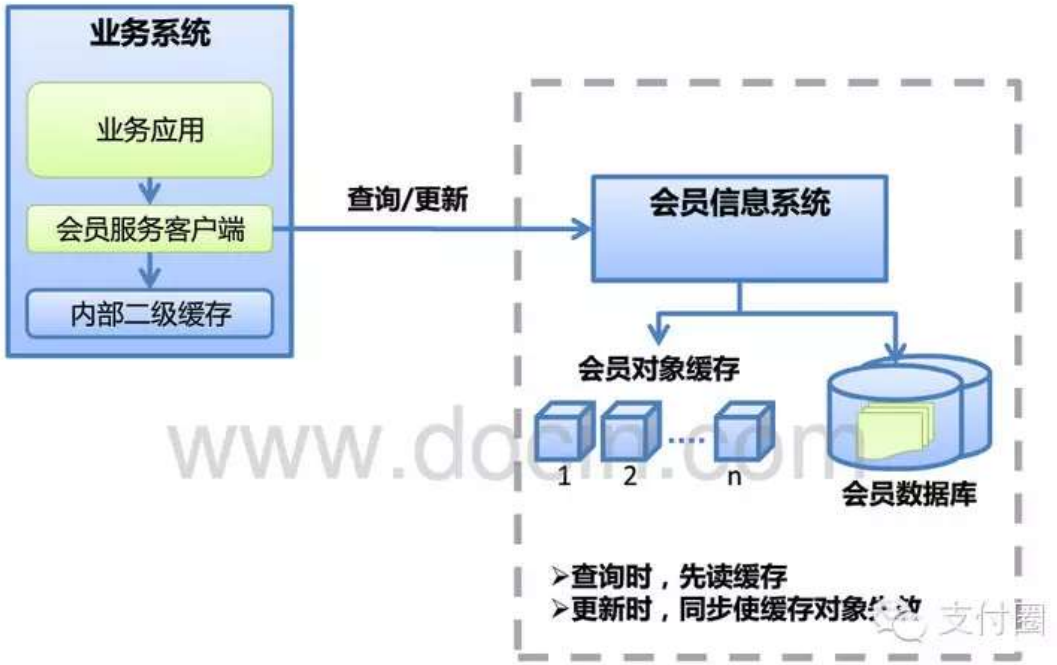
实现

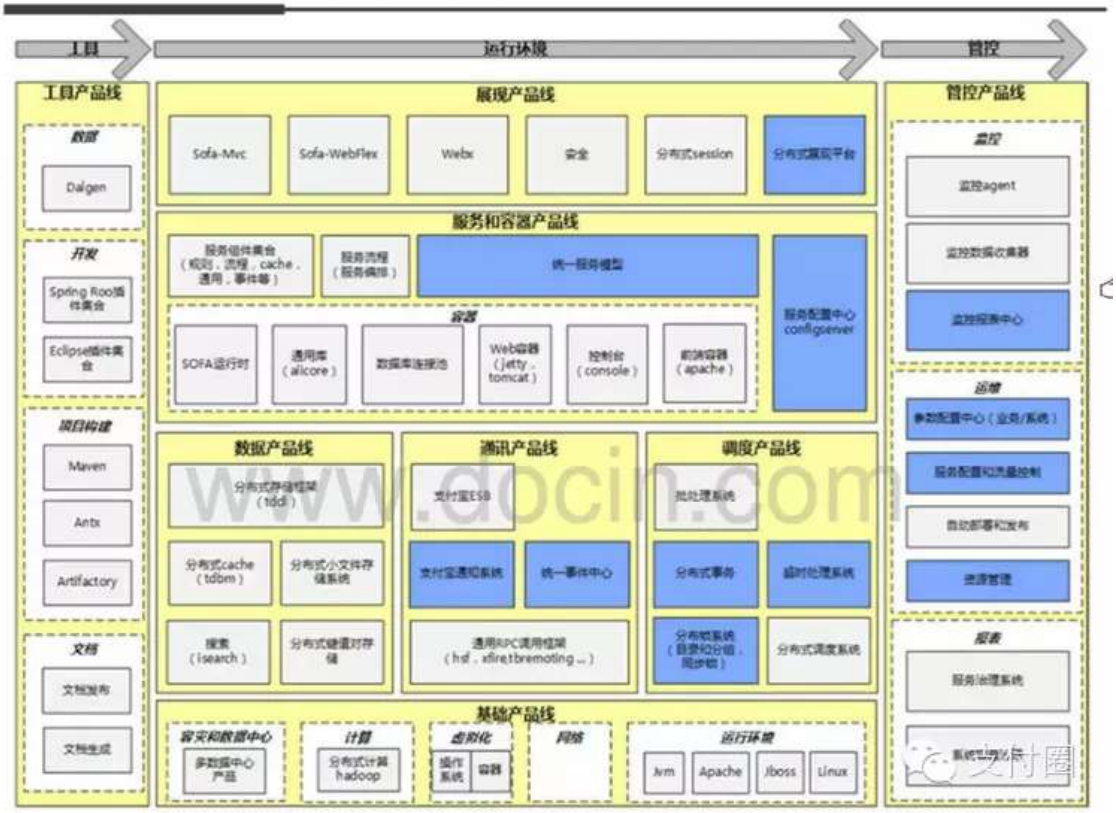
- 业务处理服务在业务事务提交前，向实时消息服务请求发送消息，实时消息服务只记录消息数据，而不真正发送
- 业务处理服务在业务事务提交后，向实时消息服务确认发送。只有在得到确认发送指令后，实时消息服务才真正发送消息
- 业务处理服务在业务事务回滚后，向实时消息服务取消发送
- 消息状态确认系统定期找到未确认发送或回滚发送的消息，向业务处理服务询问消息状态，业务处理服务根据消息ID或消息内容确定该消息是否有效

数据分布: 交易数据拆分



数据缓存





支付宝的开源分布式消息中间件--Metamorphosis(MetaQ)

Metamorphosis (MetaQ) 是一个高性能、高可用、可扩展的分布式消息中间件，类似于LinkedIn的Kafka，具有消息存储顺序写、吞吐量大和支持本地和XA事务等特性，适用于大吞吐量、顺序消息、广播和日志数据传输等场景，在淘宝和支付宝有着广泛的应用，现已开源。

Metamorphosis是淘宝开源的一个Java消息中间件。关于消息中间件，你应该听说过JMS规范，以及一些开源实现，如ActiveMQ和HornetQ等。Metamorphosis也是其中之一。

Metamorphosis的起源是我从对linkedin的开源MQ--现在转移到apache的kafka的学习开始的，这是一个设计很独特的MQ系统，它采用pull机制，而不是一般MQ的push模型，它大量利用了zookeeper做服务发现和offset存储，它的设计理念我非常欣赏并赞同，强烈建议你阅读一下它的设计文档，总体上说metamorphosis的设计跟它是完全一致的。但是为什么还需要meta呢？

简单概括下我重新写出meta的原因：

- Kafka是scala写，我对scala不熟悉，并且kafka整个社区的发展太缓慢了。
- 有一些功能是kakfa没有实现，但是我们却需要：事务、多种offset存储、高可用方案(HA)等

Meta相对于kafka特有的一些功能：

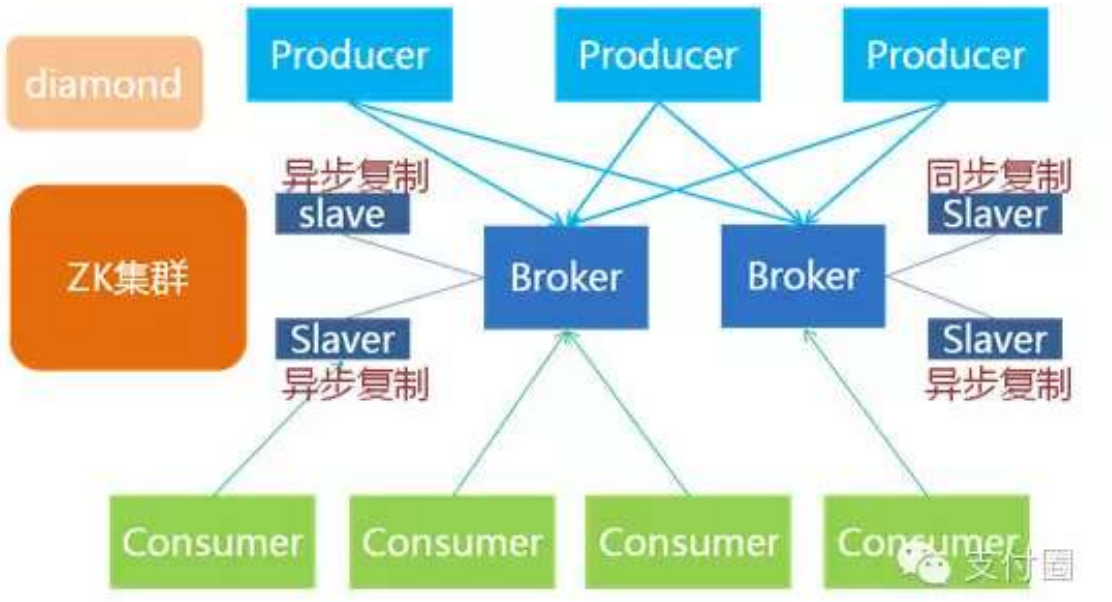
- 文本协议设计，非常透明，支持类似memcached stats的协议来监控broker
- 纯Java实现，从通讯到存储，从client到server都是重新实现。
- 提供事务支持，包括本地事务和XA分布式事务
- 支持HA复制，包括异步复制和同步复制，保证消息的可靠性
- 支持异步发送消息
- 消费消息失败，支持本地恢复
- 多种offset存储支持，数据库、磁盘、zookeeper，可自定义实现
- 支持group commit，提升数据可靠性和吞吐量。
- 支持消息广播模式
- 一系列配套项目：python客户端、twitter storm的spout、tail4j等。

因此meta相比于kafka的提升是巨大的。meta在淘宝和支付宝都得到了广泛应用，现在每天支付宝每天经由meta路由的消息达到120亿，淘宝也有每天也有上亿的消息量。

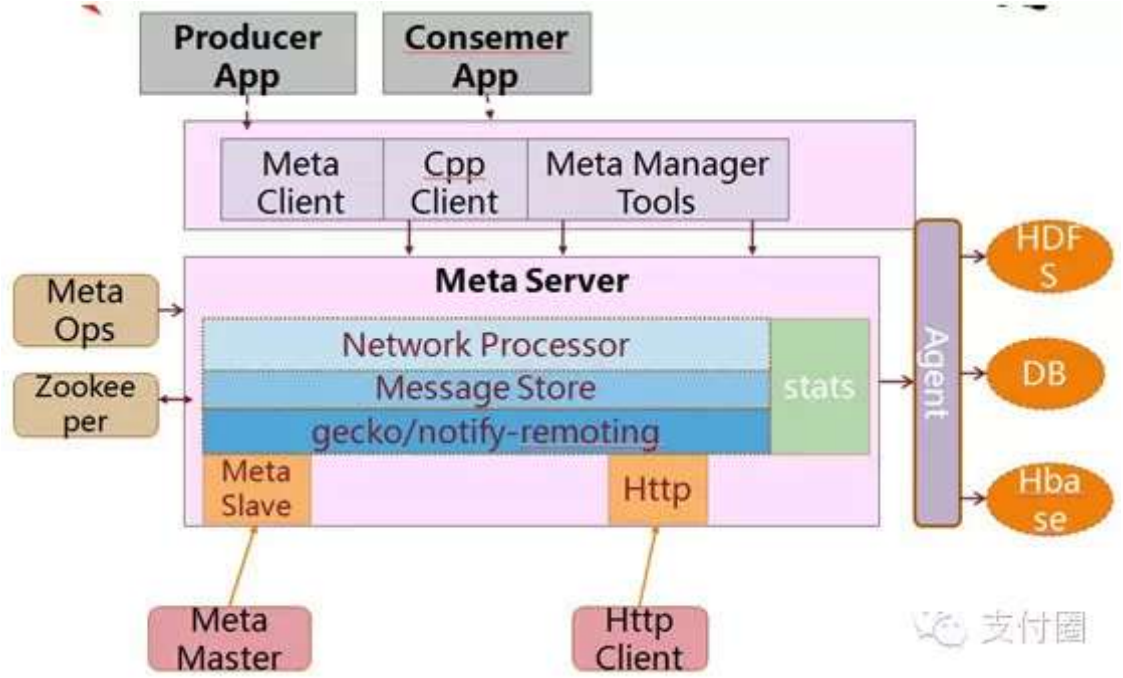
Meta适合的应用：

- 日志传输，高吞吐量的日志传输本来就是kafka的强项
- 消息广播功能，如广播缓存配置失效。
- 数据的顺序同步功能，如mysql binlog复制
- 分布式环境下（broker,producer,consumer都为集群）的消息路由，对顺序和可靠性有极高要求的场景。
- 作为一般MQ来使用的其他功能

总体结构：



内部结构：



来源【fd2012】

支付圈今天推荐一下我们的**备用号**，你可以点击添加，更多信息尽在关注！

长按二维码关注支付圈