简单理解MQ

李力

大纲

- 1 基本概念
- 2 应用场景
- 3 ONS设计
- 4 消息乱序
- 5 消息重复
- 6 分布式事务
- 7 QA

基本概念

什么是消息传递

- 通信方式。

- 异步。

- 同步编程和异步编程

应用集成通信模式

- 1 文件传输
- 2 共享数据库
- 3 RPC
- 4 消息

什么是消息队列

- 消息的载体。

为什么使用消息传递和挑战

- 1 异步通信
- 2 平台和语言集成
- 3 节流
- 4 可靠通信
- 5 无连接运行

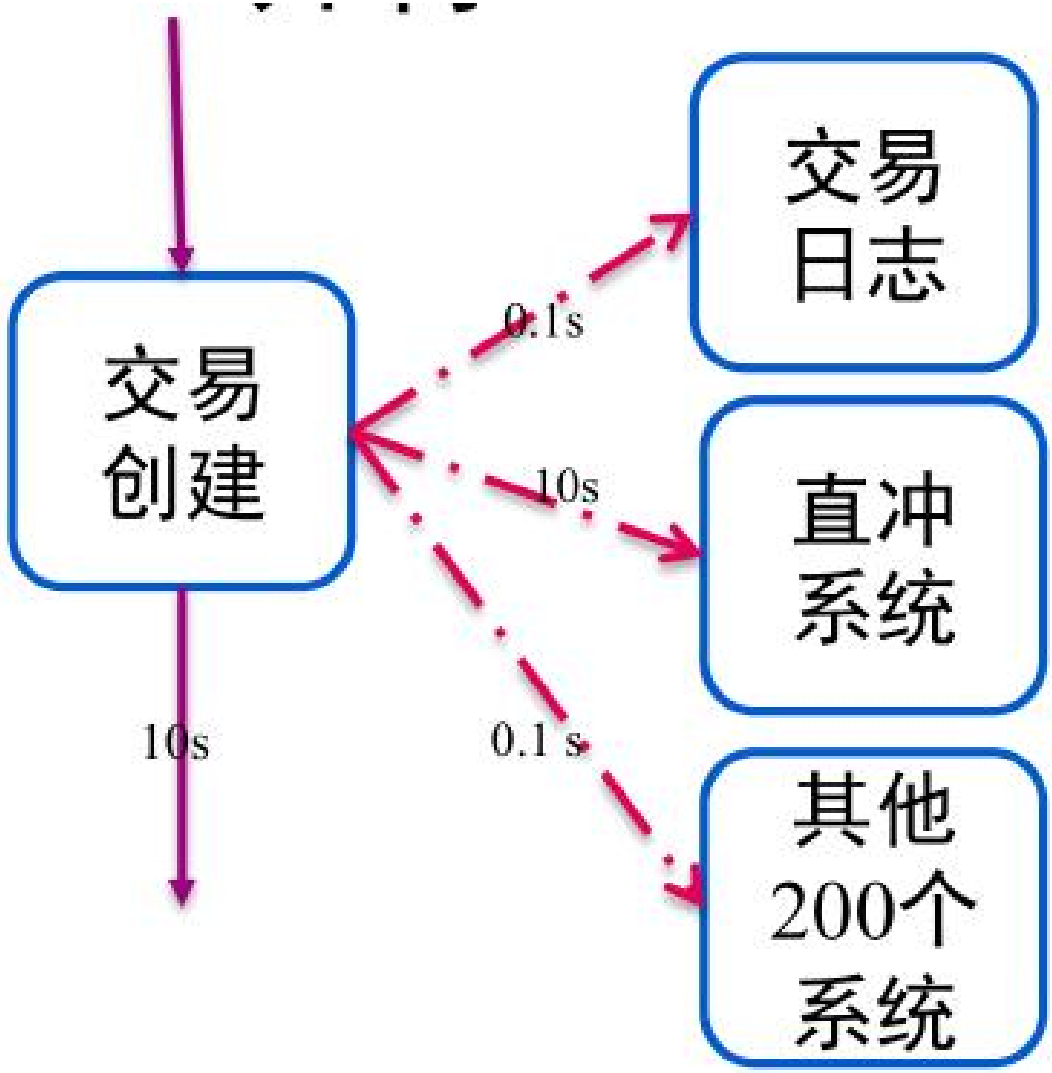
应用场景

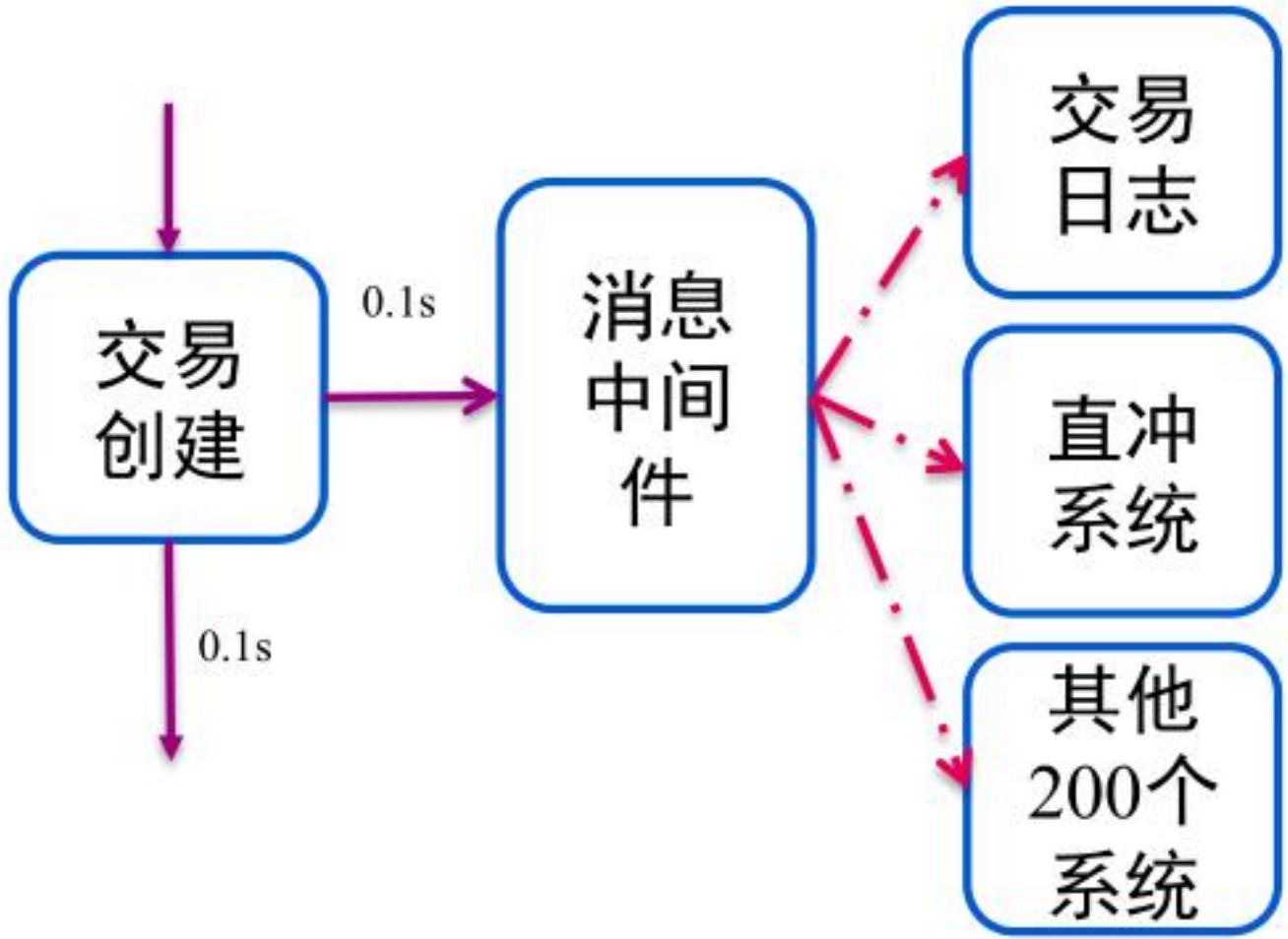
应用场景

- 异步-asynchronous
- 解耦-decoupling
- 最终一致
- 并行

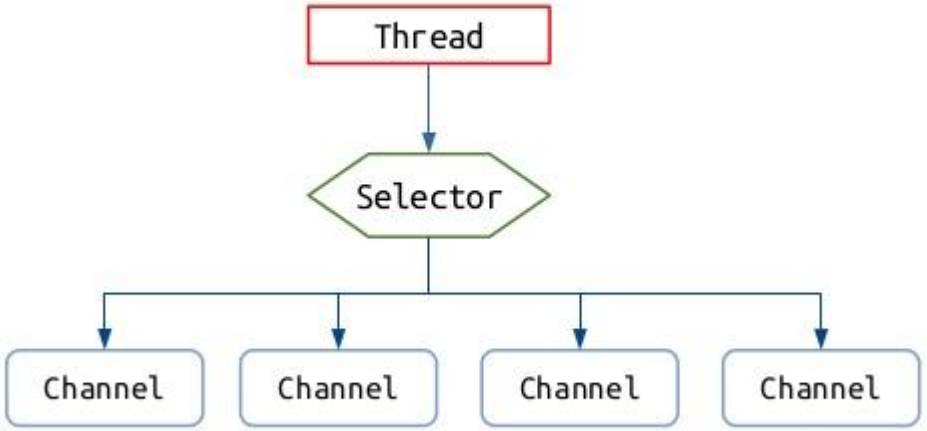
- 1 登录系统
- 2 下单
- 3 转账


交易
创建
交易
日志
直冲
系统
其他
200个
系统

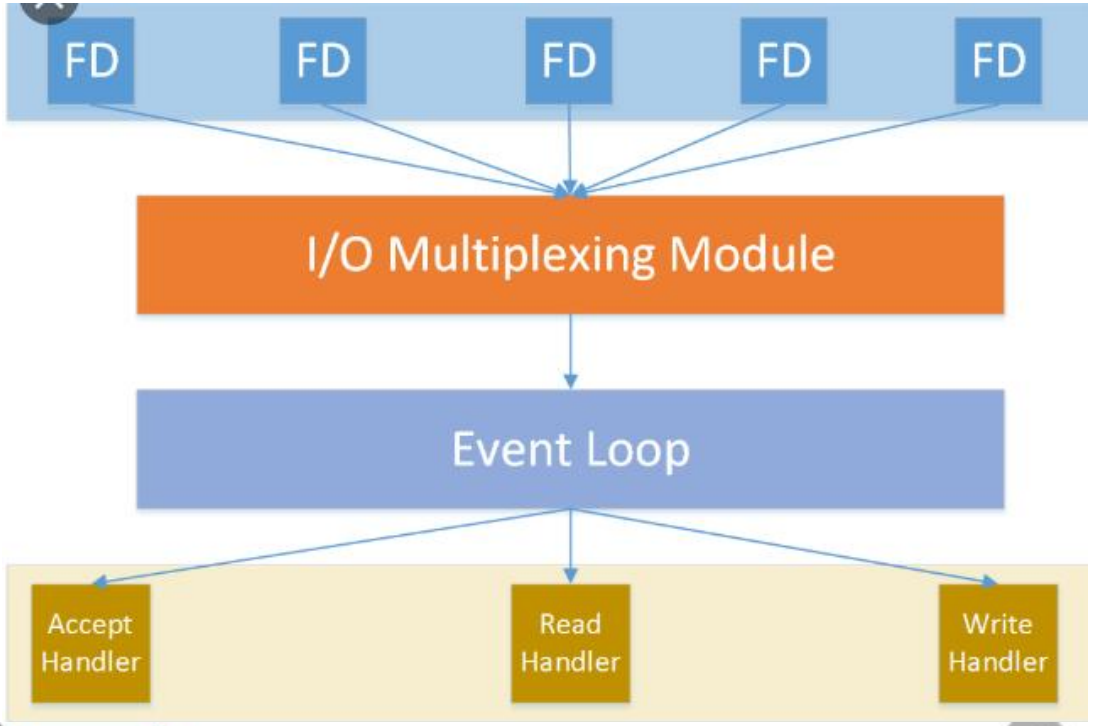

交易
创建
交易
日志
0.1s
直冲
系统
10s
其他
200个
系统
10s
0.1 s

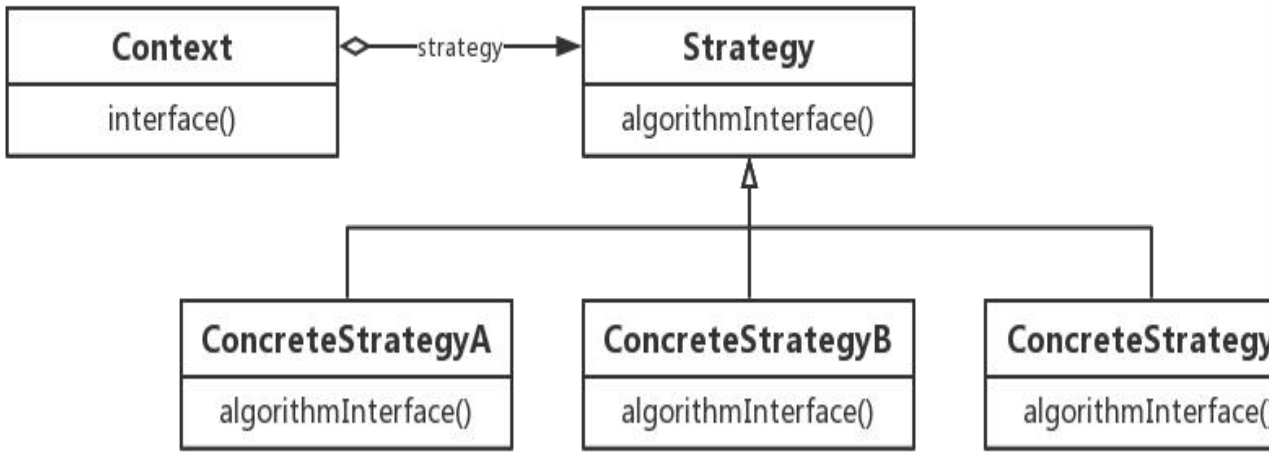

交易
创建
0.1s
0.1s
消息
中间
件
交易
日志
直冲
系统
其他
200个
系统

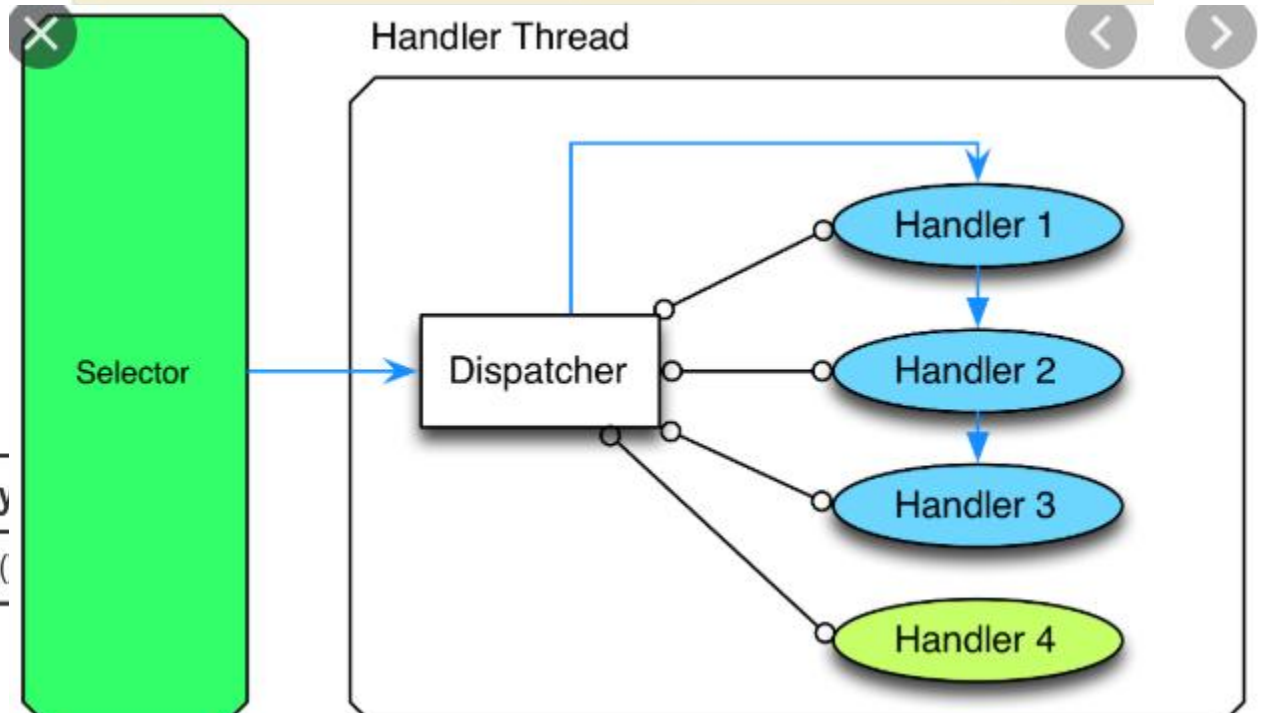
Java.NIO – NIO server design



instinctools™







- 容易理解的模型性能往往不好，性能好的模式往往不容易理解。这就是生活。

ONS设计分析

MQ理想状态

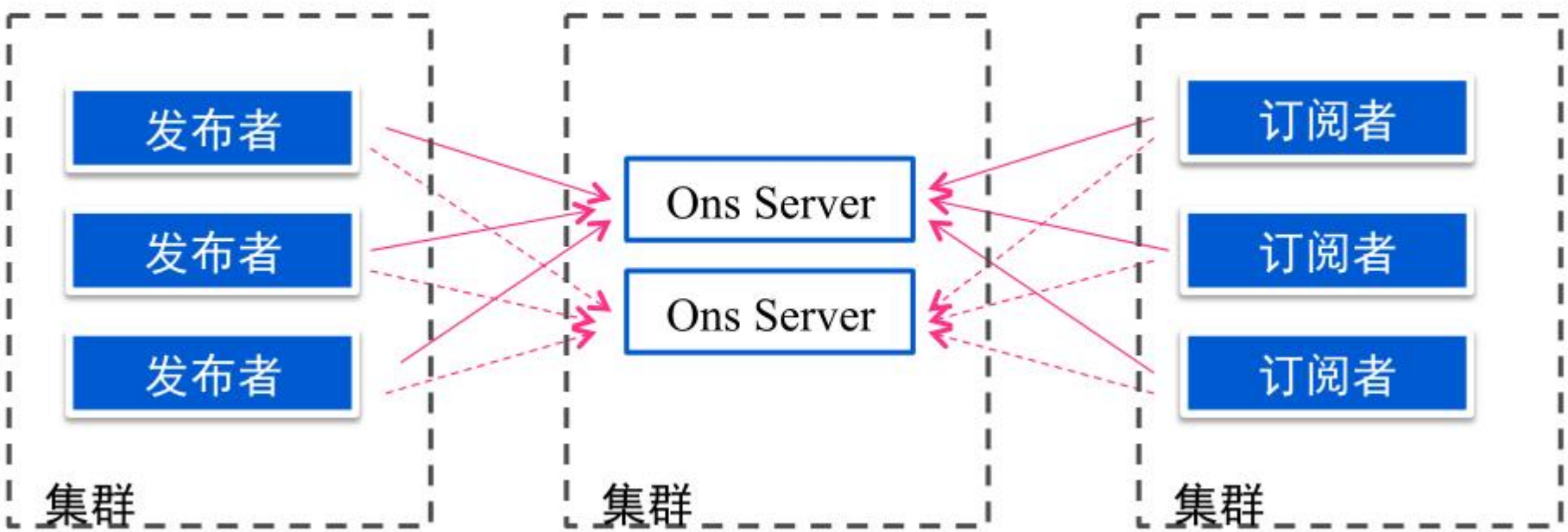
- 1 消息堆积无限不降低延时。
- 2 有发必收。
- 3 无限扩展。

ONS设计思路和关键概念

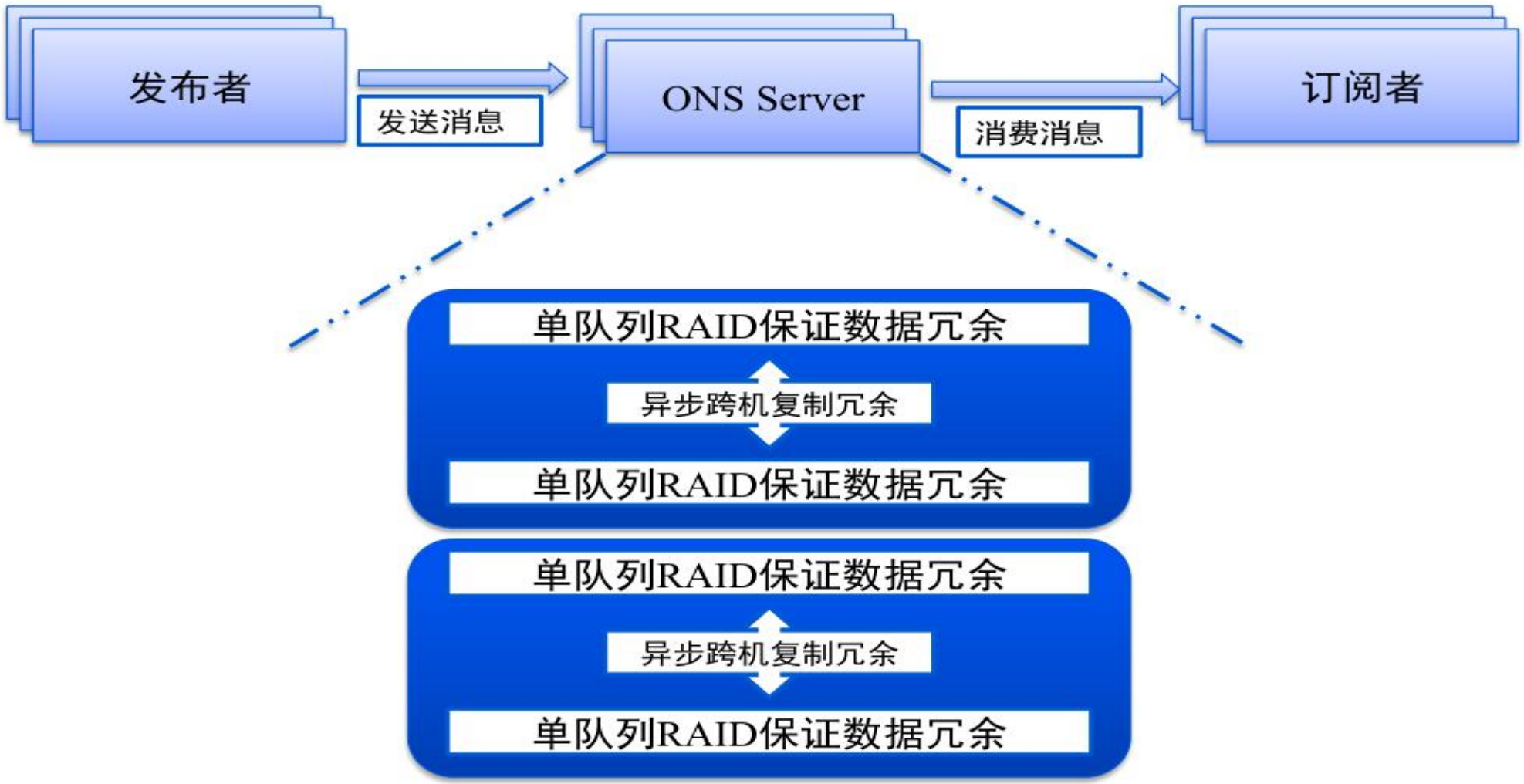
- 设计假定:
- – 每台PC机器都可能down机不可服务
- – 任意集群都可能处理能力不足
- – 最坏情况一定会发生
- – 内网环境需要低延迟来提供最佳用户体验
- 关键设计
- – 分布式集群化
- – 强数据安全
- – 海量数据堆积
- – 毫秒级投递延迟

无单点集群化设计

- 理论上无限的处理能力
- 集群级别高可用



强数据安全和高可用


发布者
发送消息
ONS Server
消费消息
订阅者
单队列RAID保证数据冗余
异步跨机复制冗余
单队列RAID保证数据冗余
单队列RAID保证数据冗余
异步跨机复制冗余
单队列RAID保证数据冗余

消息丢失怎么办?

- 1　MQ增量更新Mysql课程信息到ES构建倒排索引，但是MQ挂了。课程数据不更新了。

- 2　10个节点网关API数据内存实时更新，但是MQ挂了，请求报错。

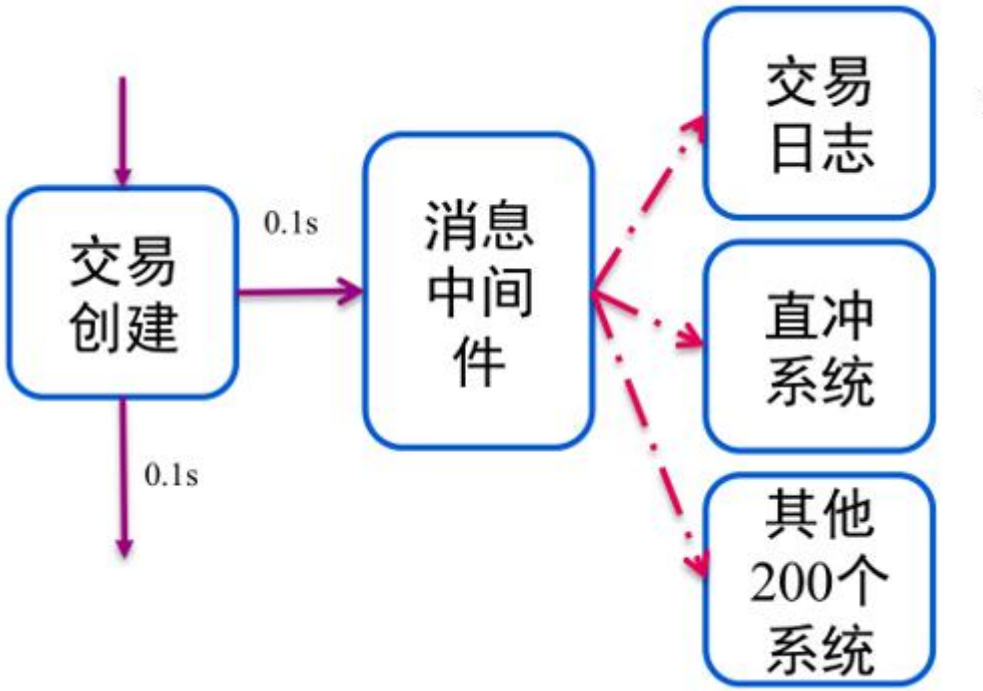
消息持久化问题

- (1). 持久化到数据库，例如 Mysql。
- (2). 持久化到 KV 存储，例如 levelDB、伯克利 DB 等 KV 存储系统。
- (3). 文件记录形式持久化，例如 Kafka，RocketMQ
- (4). 对内存数据做一个持久化镜像，例如 beanstalkd，VisiNotify

- 持久化做好的ActiveMQ。LevelDB，JDBC， KahaDB
- http://activemq.apache.org/persistence.html

海量数据堆积能力

- 任何集群都有可能处理能力不足
- 消息堆积是常态



海量数据堆积能力

- 面向堆积设计
  - 百亿级别的消息大量堆积，系统稳定，延迟不增
  - 堆积能力
  - 双十一多年考验
  - 单消息server不可用数据不丢
- 默认落磁盘策略，并针对磁盘吞吐做优化。
- 集群可无限扩展，保证足够堆积能力。

毫秒级别投递延迟

- 采用长轮询/推送方式

- 应用准备好时候，如何消费一个消息？轮询消费者
- 消息达到时候如何让应用自动消费一个消息？事件驱动消费者

关键概念-主题Topic

- 第一级消息类型
- 书的标题
- 交易消息

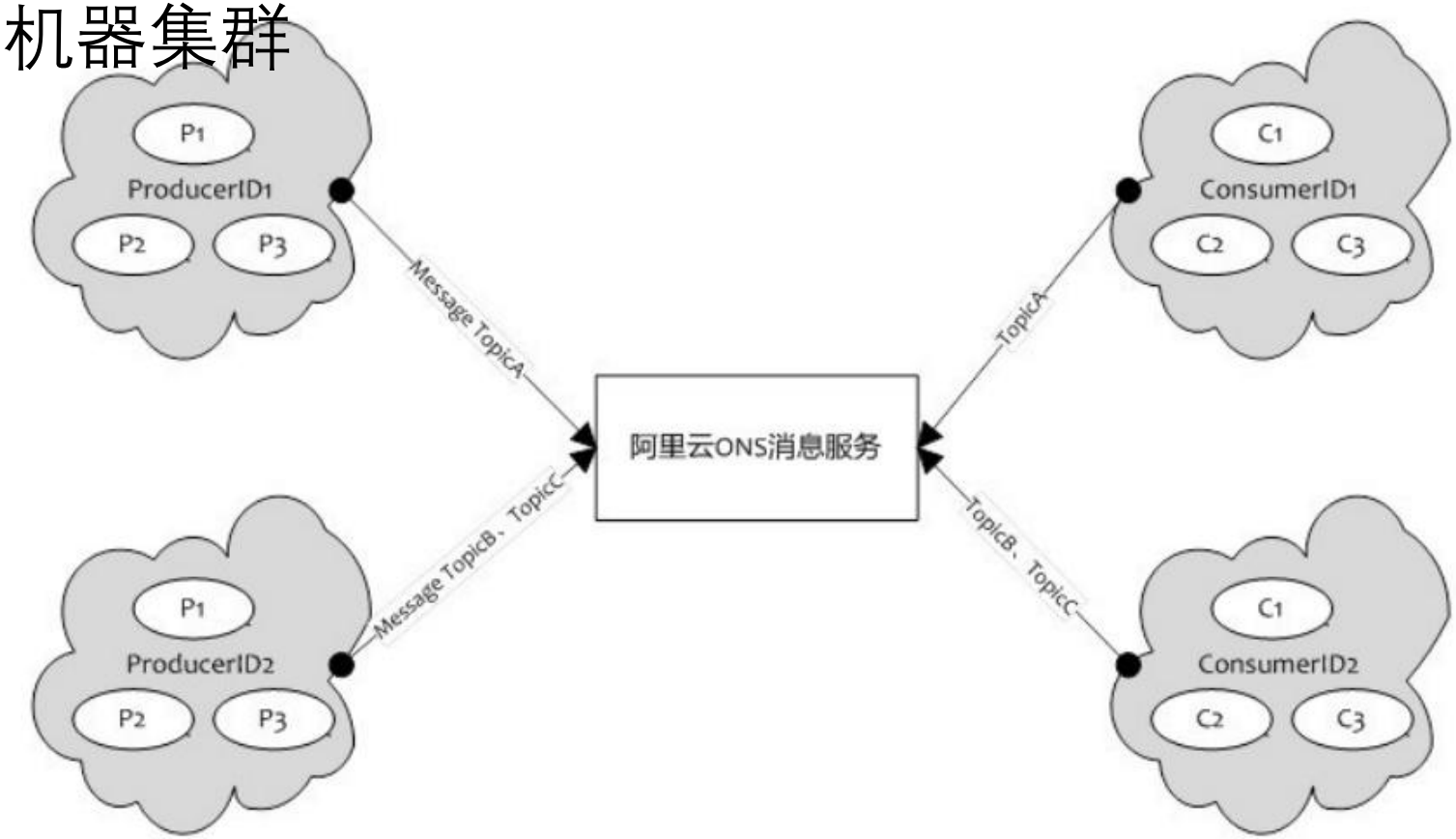
消息类型Tag

- 第二级消息类型
- 书的目录
- -方便检索

- 交易消息
  - 交易创建
  - 交易完成

发送和订阅组

- 发送/接受机器集群

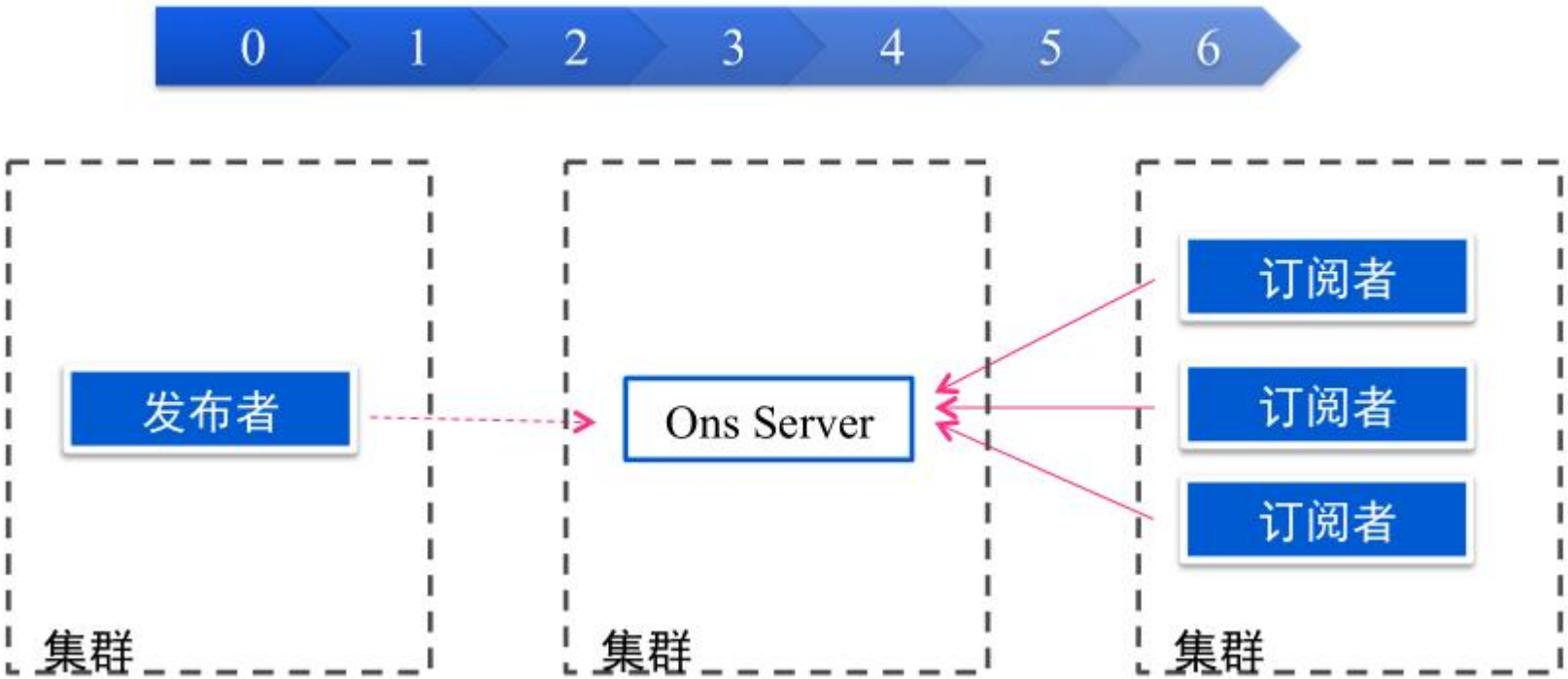


消息乱序问题

- 产生原因
- 有序队列优劣分析

产生原因

- 吞吐+容错 VS 方便，容易理解



有序队列优劣分析

- 优势:
  - 容易理解
  - 处理问题容易

- 劣势:
  - 并行度瓶颈
  - 异常处理

- BUT
  - 我们需要集群容错性和高吞吐!

- 在世界上解决计算机问题的最简单方法:
  - “恰好”不需要解决它。
  - 因为解决任何一个问题都有付出代价。

订单举例

- 一笔订单有三个状态（创建，付款，发货）
  - 订单间没有先后顺序，所以乱序无所谓。
  - 某应用只关注付款

转账

事务单元

操作指令	耗时
锁定Bob账户	0.001ms
查看Bob是否有100元	1ms
从Bob账号中减少100元	2ms
解锁Bob账户	0.001ms

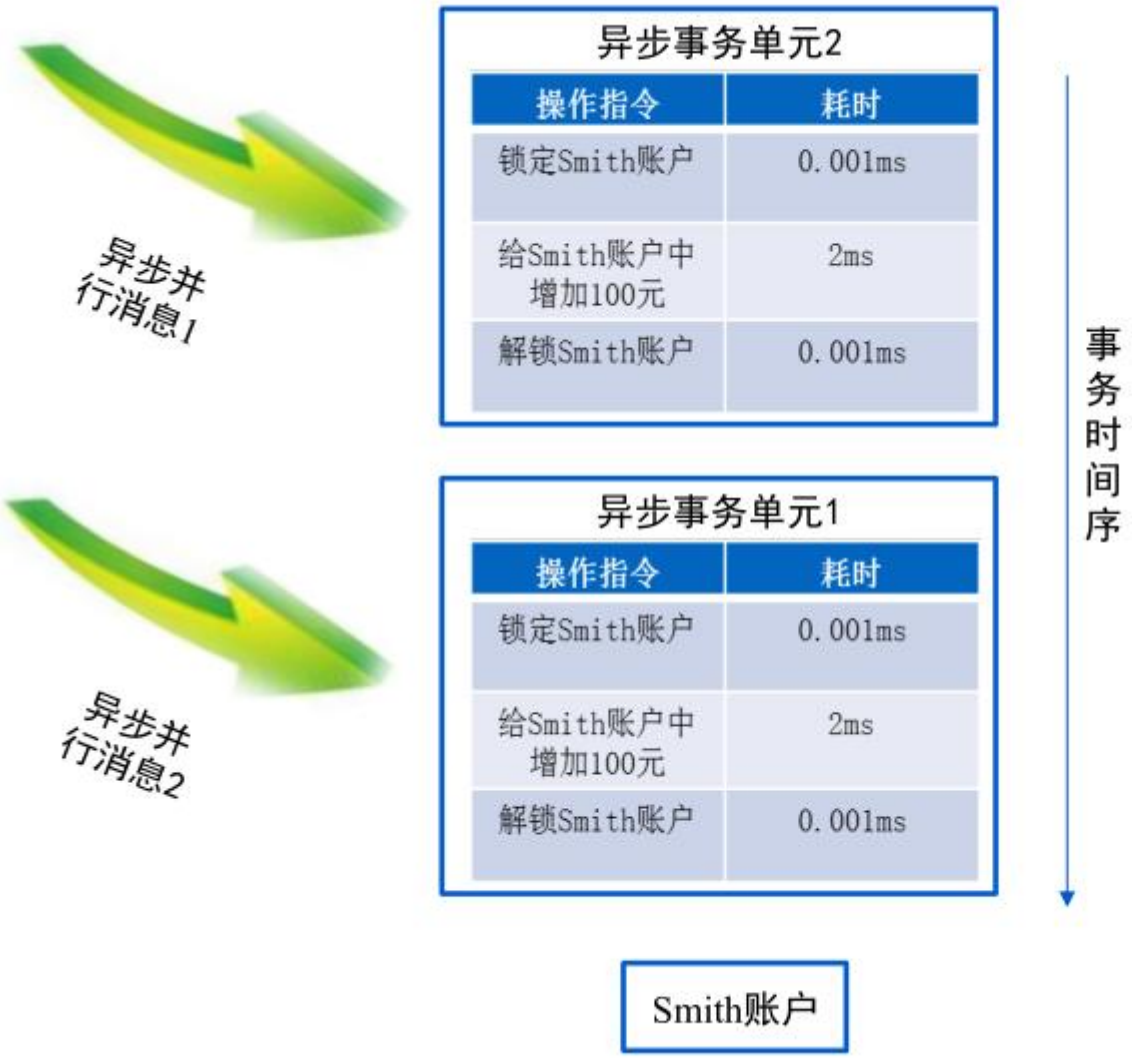
异步并行消息

异步事务单元

操作指令	耗时
锁定Smith账户	0.001ms
给Smith账户中增加100元	2ms
解锁Smith账户	0.001ms

事务时间序

多人通过消息转账


异步事务单元2
操作指令
耗时
锁定Smith账户
0.001ms
给Smith账户中增加100元
2ms
解锁Smith账户
0.001ms
异步并行消息1
异步事务单元1
操作指令
耗时
锁定Smith账户
0.001ms
给Smith账户中增加100元
2ms
解锁Smith账户
0.001ms
异步并行消息2
事务时间序
Smith账户

是否真的需要顺序?

- 不关注乱序的应用大量存在。

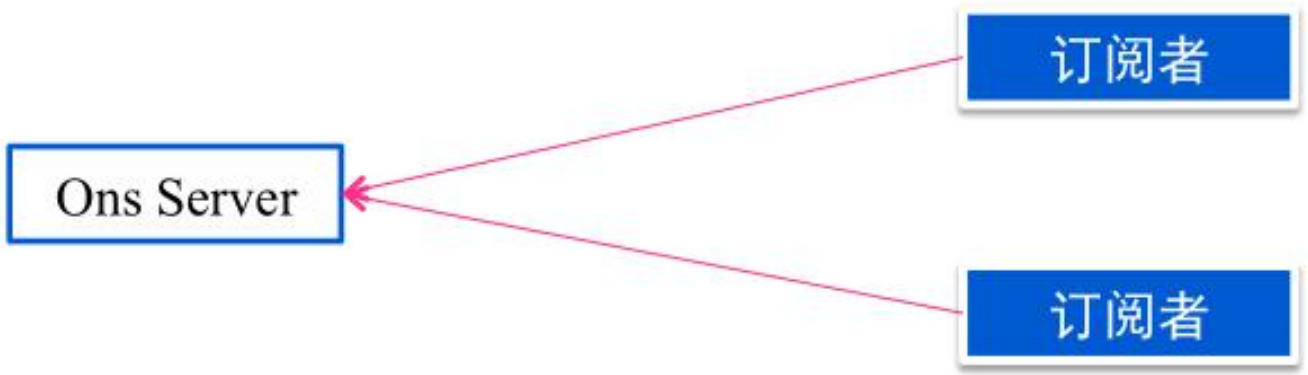
- 队列无序不意味着消息无序
  - TCP协议
  - 可以通过发送编号和接收端恢复方式的恢复顺序（重排器模式ReSequencer）

  - https://www.enterpriseintegrationpatterns.com/patterns/messaging/Resequencer.html

消息重复问题

消息重复问题

- 产生原因
  - 网络不可达- 如果发送者只发送了1条消息，是否不会重复呢?



解决方案

- 最好解决方案：恰好不需要解决（不需要mq内部解决）
- 幂等　f(x)=f(f(x)) 无论操作多少次，结果都一样。

  - 幂等 – 无论做多少次结果都一样
    - insert into T (col1) values (1)
    - update T set col = 2 where col = 1
    - delete from T where col = 1
  - 非幂等
    - update set col = col + 1

如何获得幂等？

- 幂等消息本身不需要去重
- 非幂等消息去重：
  - 显示去重：保证有唯一ID标记每一条消息，保证消息处理成功和去重表日志同时出现。代价？
    TCP协议消除包重复。
  - 重新定义消息语义，使之幂等。（持久层去重）

- https://www.enterpriseintegrationpatterns.com/patterns/messaging/IdempotentReceiver.html

TCP协议给我们的启示

- 解决包顺序问题。
- 解决包重复问题。

- 解决消息顺序。
- 解决消息重复。MQ内部 VS 应用系统

- 为什么市面主流MQ对顺序消息支持少，对消息重复支持少呢？为什么消息重复要应用系统做呢？
- End-To-End Arguments in System Design

分布式事务与消息队列

单机事务实现

- 2PL
- MVCC

分布式事务

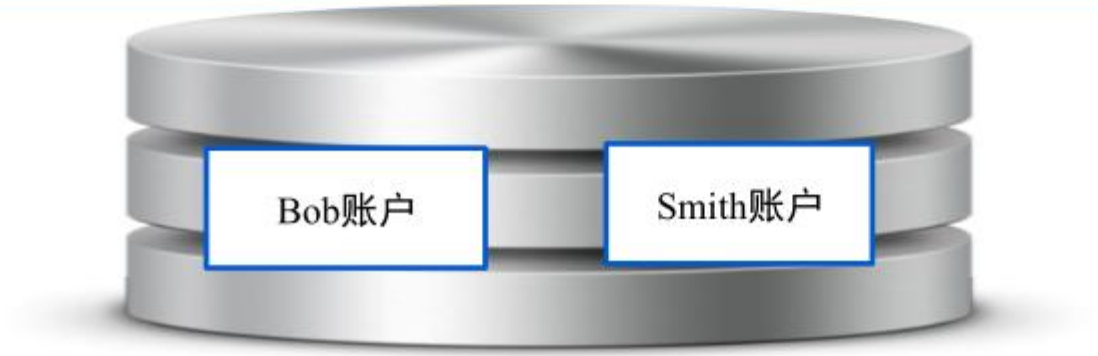
- 2PC
- Why not MVCC?

事务的分布式优化

事务单元

操作指令	耗时	总耗时
锁定Bob账户	0.001ms	5.004ms
锁定Smith账户	0.001ms	
查看Bob是否有100元	1ms	
从Bob账号中减少100元	2ms	
给Smith账户中增加100元	2ms	
解锁Bob账户	0.001ms	
解锁Smith账户	0.001ms	

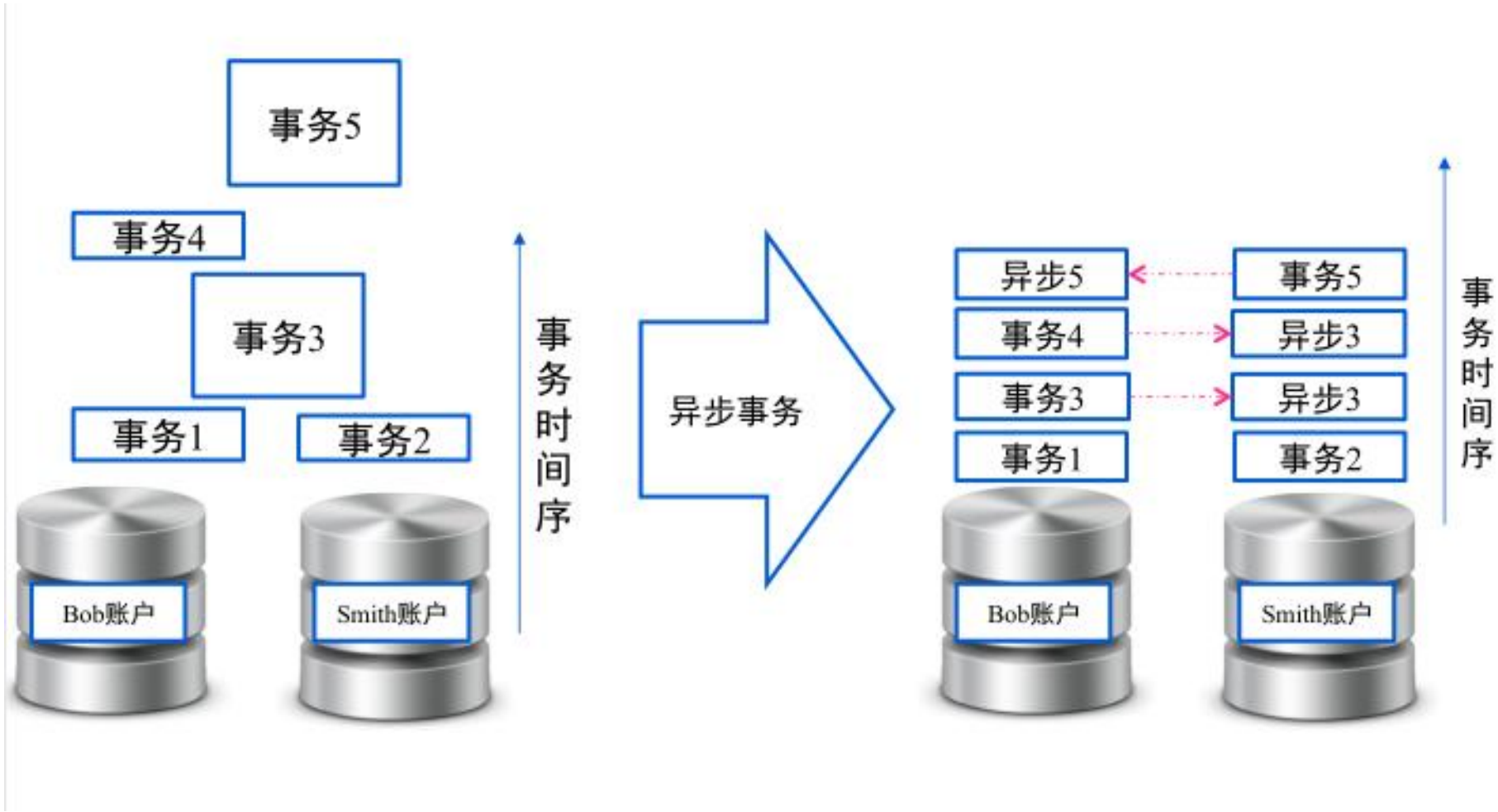
事务时间序



延迟增加
用户体验
下降

操作指令	耗时	总耗时
锁定Bob账户	0.001ms	11.004ms
通过网络锁定Smith账户	2ms+0.001ms	
查看Bob是否有100元	1ms	
从Bob账号中减少100元	2ms	
通过网络给Smith账户中增加100元	2ms+2ms	
解锁Bob账户	0.001ms	
通过网络解锁Smith账户	2ms+0.001ms	

事务时间序


事务5
事务4
事务3
事务1
事务2
Bob账户
Smith账户
事务时间序
异步事务
异步5
事务4
事务3
事务1
事务5
异步3
异步3
事务2
Bob账户
Smith账户
事务时间序

事务单元

操作指令	耗时
锁定Bob账户	0.001ms
查看Bob是否有100元	1ms
从Bob账号中减少100元	2ms
解锁Bob账户	0.001ms

异步并行消息

异步事务单元

操作指令	耗时
锁定Smith账户	0.001ms
给Smith账户中增加100元	2ms
解锁Smith账户	0.001ms

事务时间序

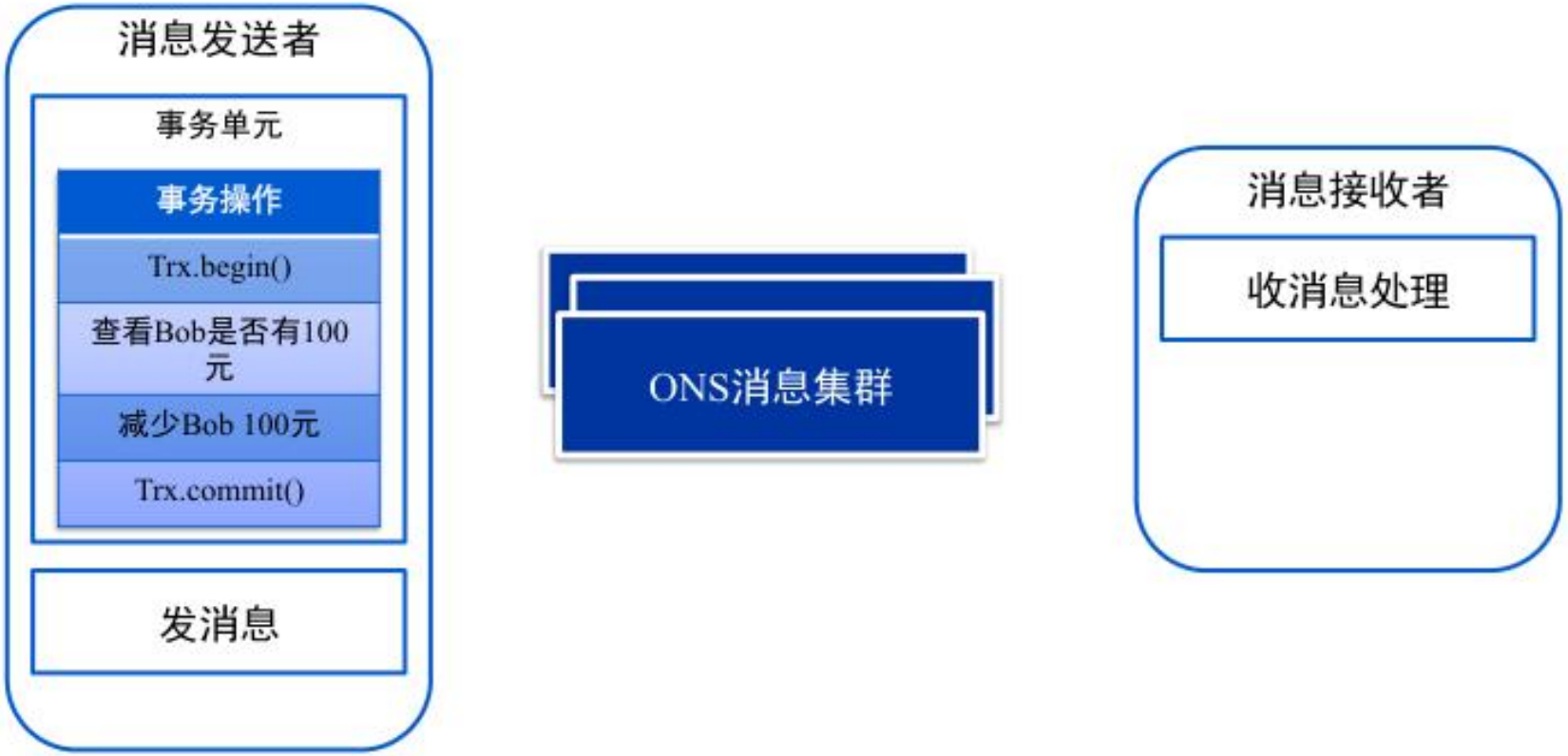
消息与事务转账

- 关键设计难点
  - 如何保证消息发出与Bob账户减钱同时成功或同时失败?
  - 消息处理超时如何解决？努力送达模型
  - 消息处理失败如何解决？人工介入

- https://www.enterpriseintegrationpatterns.com/patterns/messaging/TransactionalClient.html

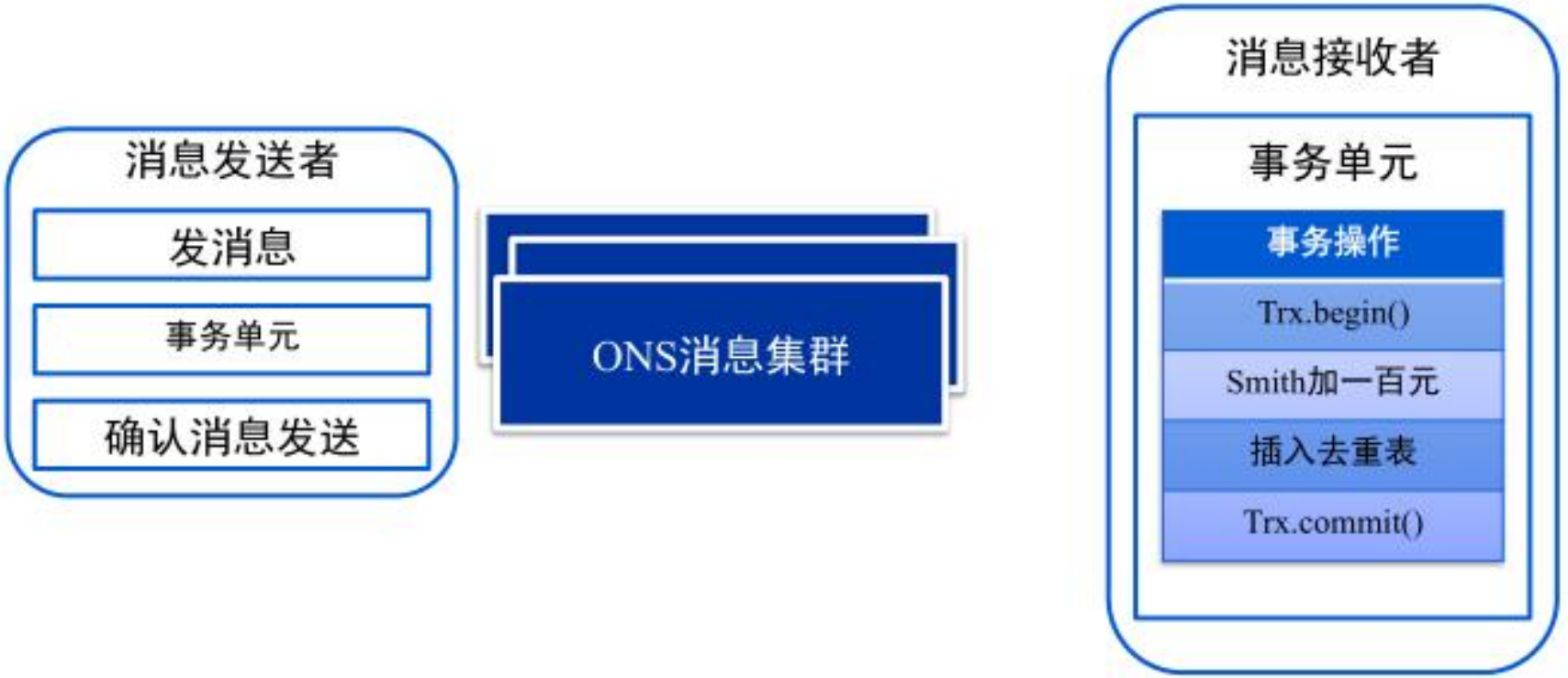

消息发送者
发消息
事务单元
事务操作
Trx.begin()
查看Bob是否有100元
减少Bob 100元
Trx.commit()
ONS消息集群
消息接收者
收消息处理

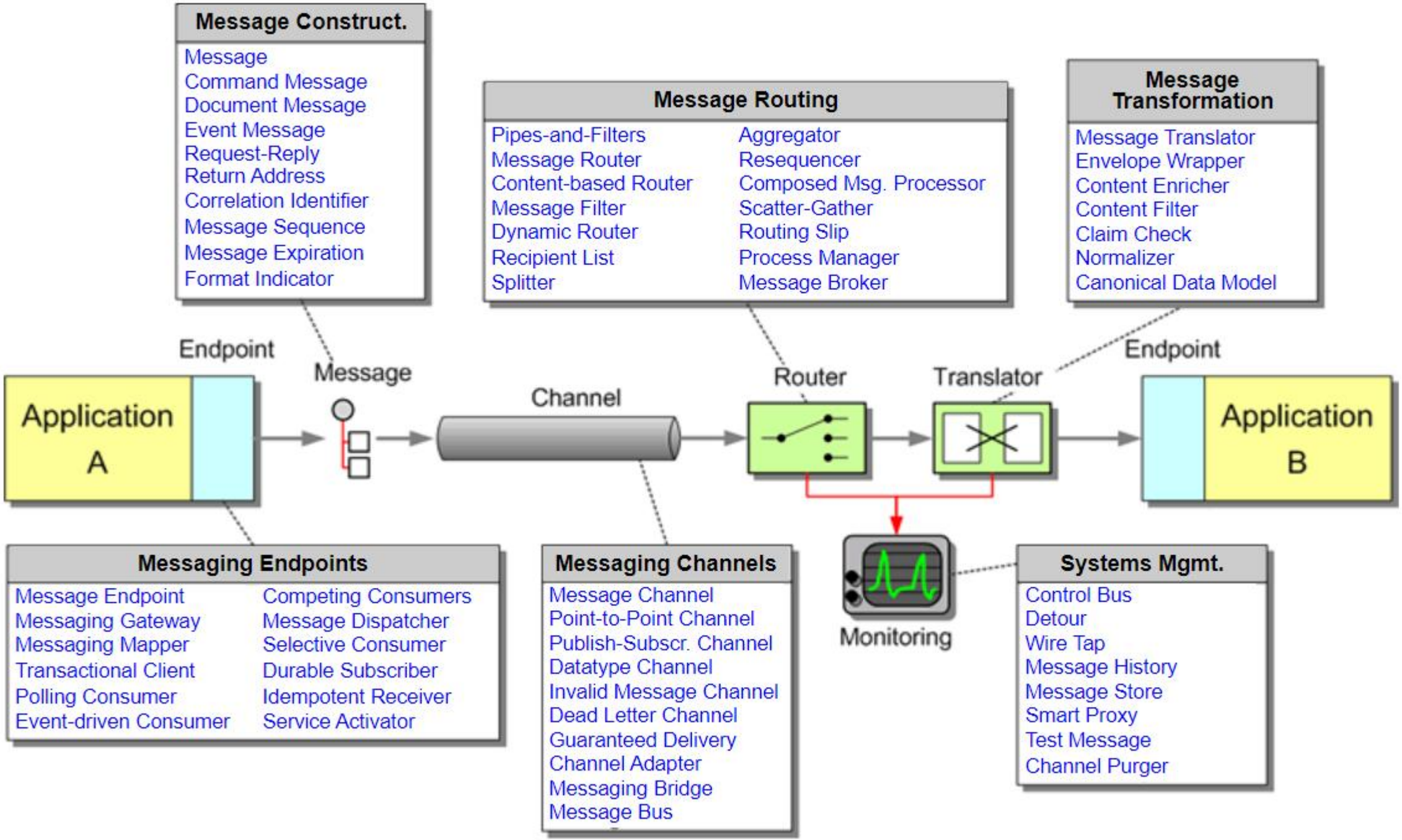
同时成功，同时失败（事务消息）




消息发送者
发消息
事务单元
事务操作
Trx.begin()
查看Bob是否有100元
减少Bob 100元
Trx.commit()
确认消息发送
ONS消息集群
消息接收者
收消息处理

处理超时问题


消息发送者
发消息
事务单元
确认消息发送
ONS消息集群
消息接收者
事务单元
事务操作
Trx.begin()
Smith加一百元
插入去重表
Trx.commit()


Message Construct.
Message
Command Message
Document Message
Event Message
Request-Reply
Return Address
Correlation Identifier
Message Sequence
Message Expiration
Format Indicator
Message Routing
Pipes-and-Filters
Message Router
Content-based Router
Message Filter
Dynamic Router
Recipient List
Splitter
Aggregator
Resequencer
Composed Msg. Processor
Scatter-Gather
Routing Slip
Process Manager
Message Broker
Message Transformation
Message Translator
Envelope Wrapper
Content Enricher
Content Filter
Claim Check
Normalizer
Canonical Data Model
Endpoint
Message
Channel
Router
Translator
Endpoint
Application A
Application B
Messaging Endpoints
Message Endpoint
Messaging Gateway
Messaging Mapper
Transactional Client
Polling Consumer
Event-driven Consumer
Competing Consumers
Message Dispatcher
Selective Consumer
Durable Subscriber
Idempotent Receiver
Service Activator
Messaging Channels
Message Channel
Point-to-Point Channel
Publish-Subscr. Channel
Datatype Channel
Invalid Message Channel
Dead Letter Channel
Guaranteed Delivery
Channel Adapter
Messaging Bridge
Message Bus
Monitoring
Systems Mgmt.
Control Bus
Detour
Wire Tap
Message History
Message Store
Smart Proxy
Test Message
Channel Purger

事件驱动系统

- 把消息构造成事件消息。

- https://www.enterpriseintegrationpatterns.com/patterns/messaging/EventMessage.html

思考

- 1 哪些应该应用系统做，哪些应该MQ做？或许有一天MQ也不做了怎么办？

- 2 是否真的需要顺序和去重？还是恰好“不需要解决”

- 3 具体到消息产品时候具备哪些特点和这些特点到底能保证什么？

Q&A

感谢聆听