# 参考

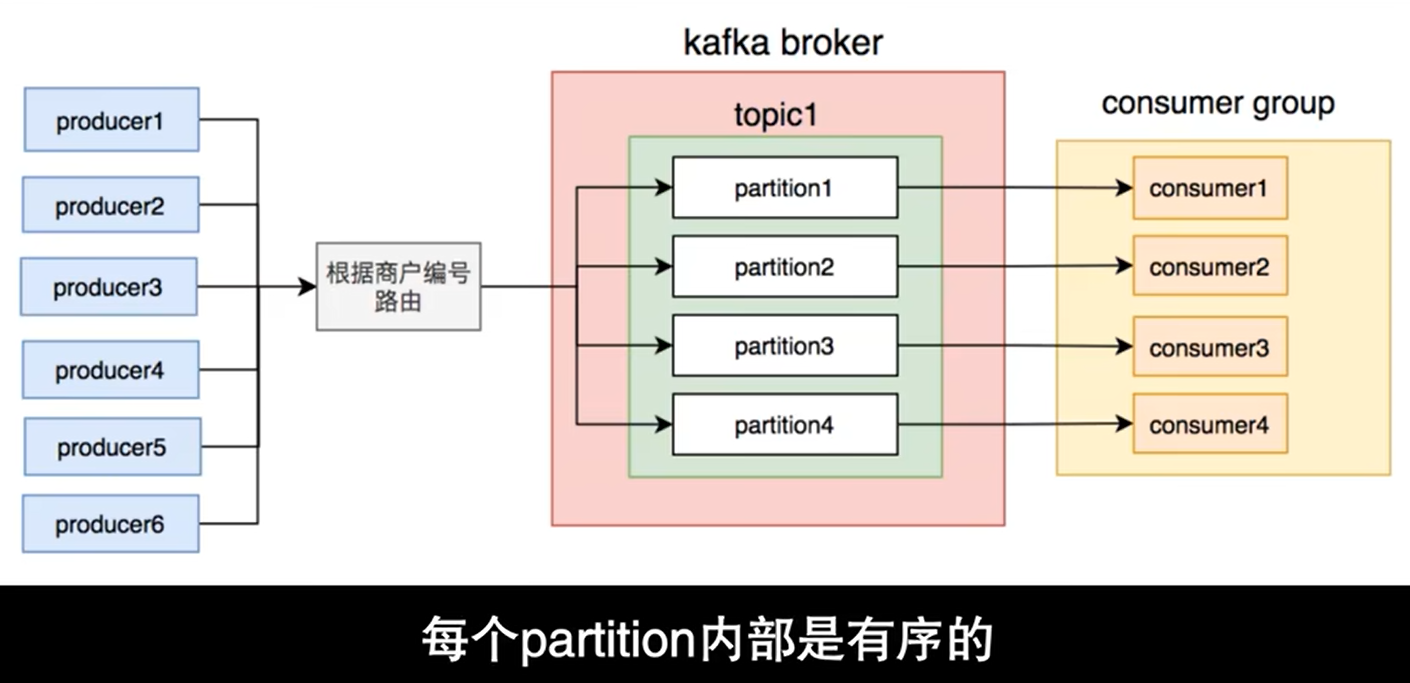
<https://www.bilibili.com/video/BV1gm4y1w7tN/?spm_id_from=333.337.search-card.all.click>

# 生产问题

## 顺序消费问题

同一个订单必须先下单，支付，。。。

基于同一Partition消息有序



## 网络超时（异步重试）问题

业务接口时不时超时，连不上数据库，导致创建订单失败，从而后续的支付等状态刷新失败。

下单-》支付-》。。。

解决：失败重试

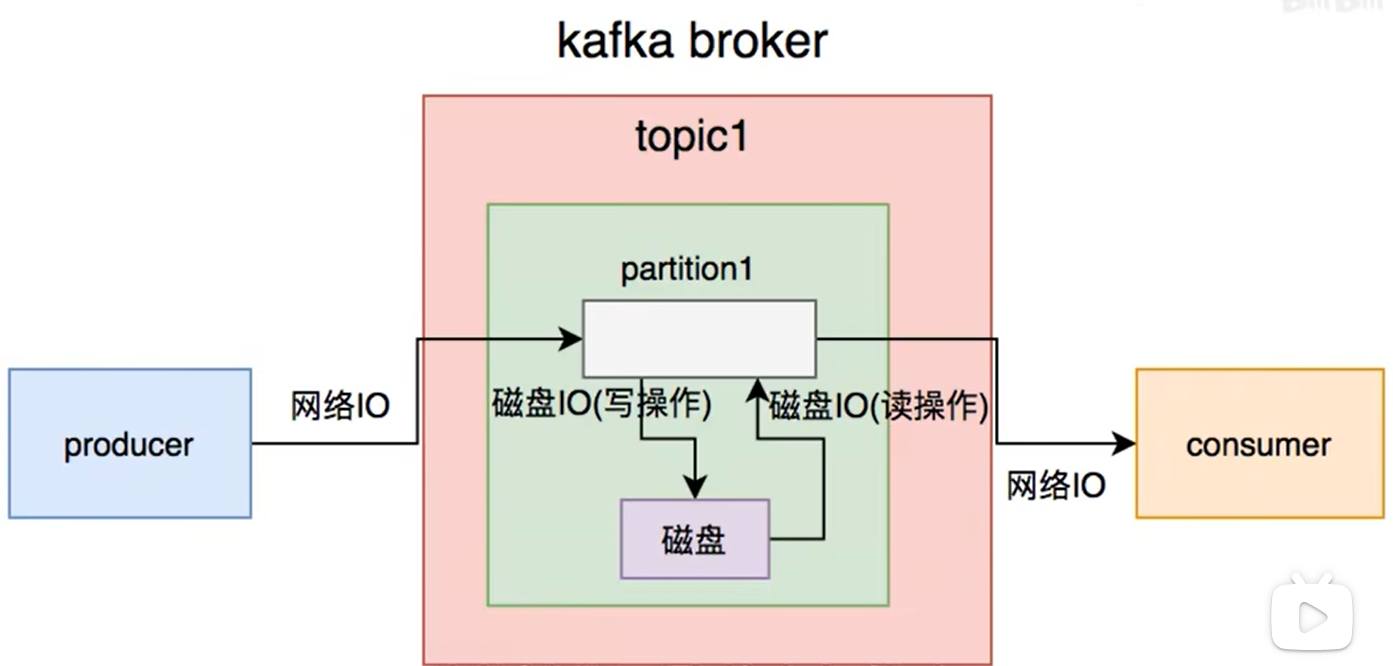
方案1：同步重试：

1. 消费失败，尝试重试3-5次，但有些请求第6次才成功怎么办？
2. 会阻塞其他商户订单消息的读取，降低吞吐量

**方案2：异步重试**

1. 处理失败的消息保存到专门重试的表；某消息到达时，如果前序消息有失败，则将该消息也放到重试表；否则直接进行处理。如果处理失败，则放到专门重试的表；
2. 基于elastic-job建立失败重试
3. 如果重试7次仍失败，则将该消息标记失败，并通知开发介入处理

## 消息积压

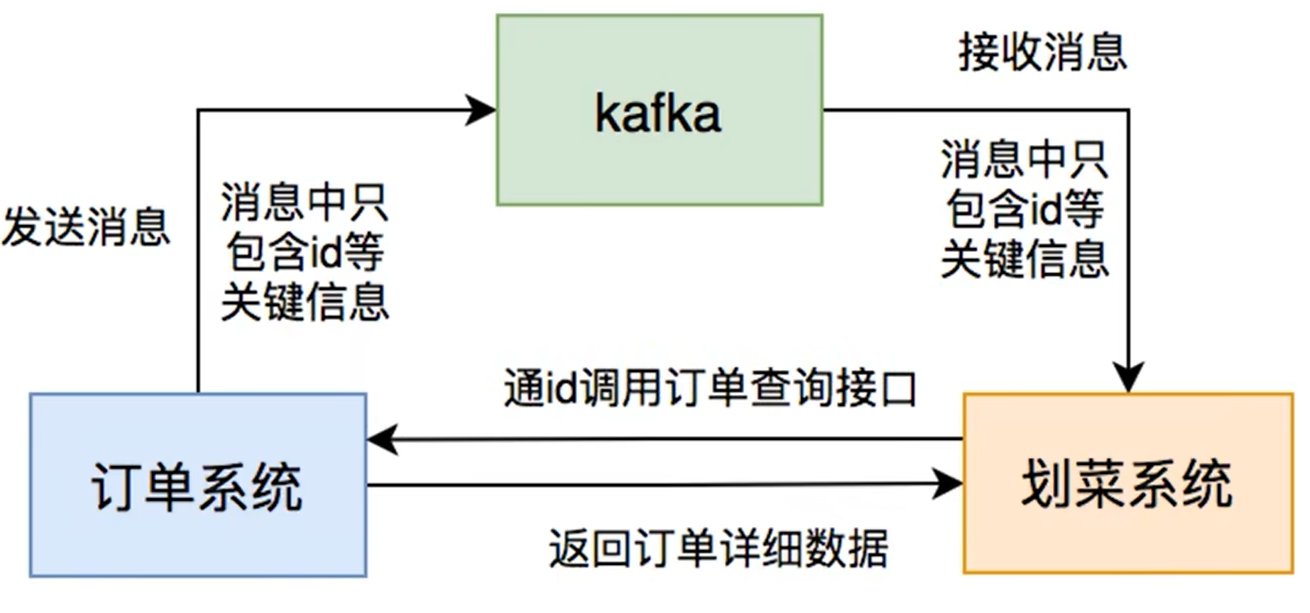


从左到右共2次网络IO和两次磁盘IO

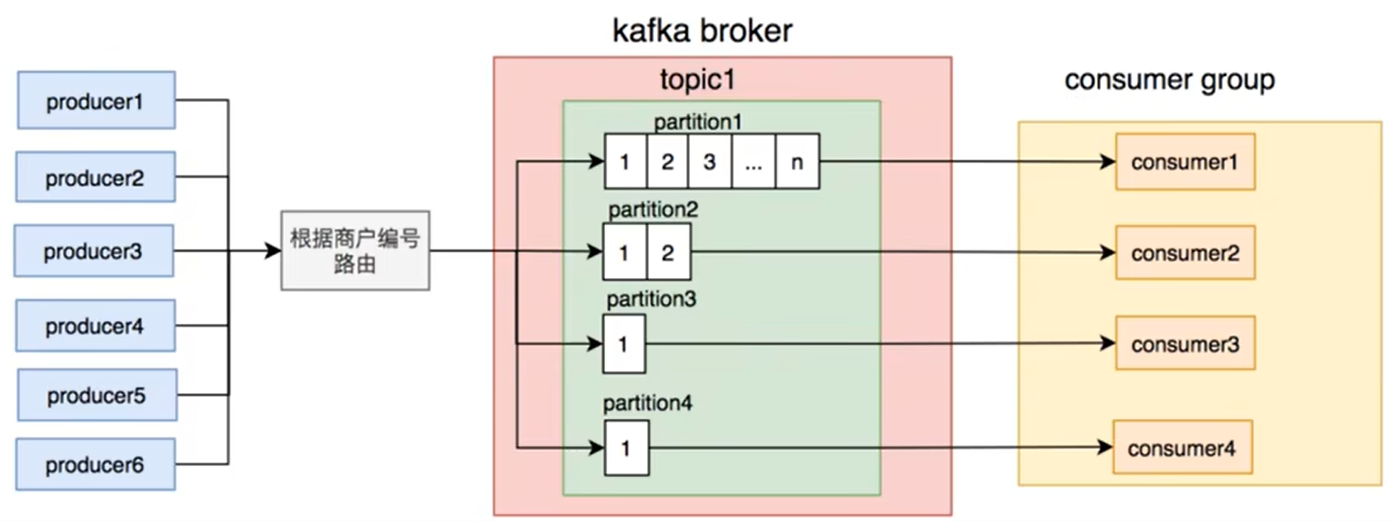
消息体过大

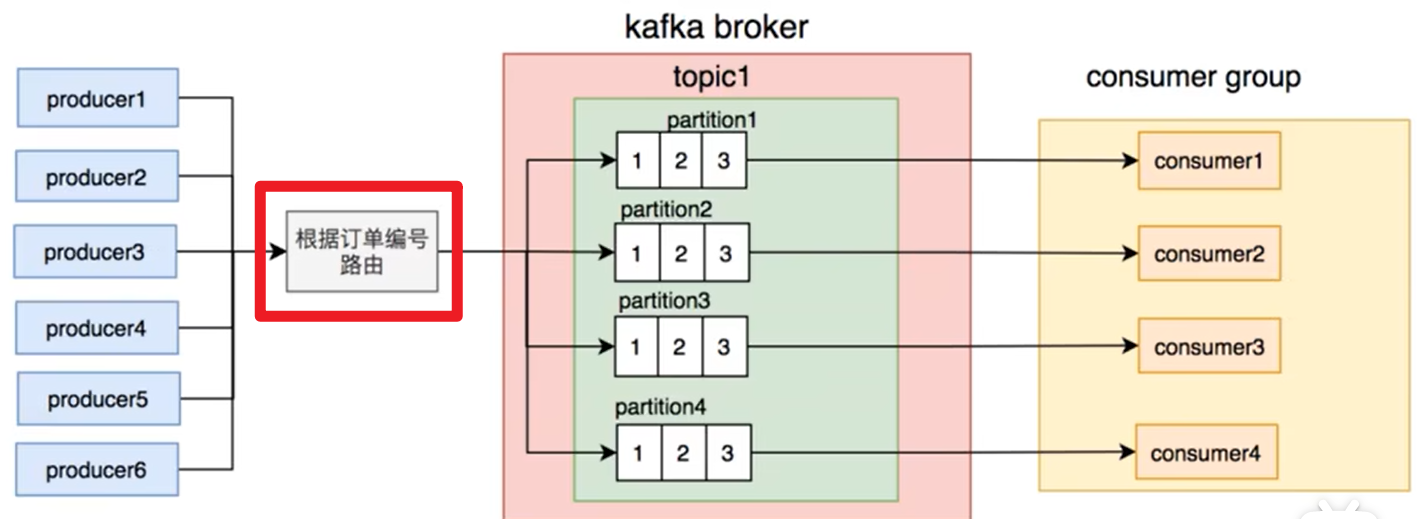
**解决：**

1. **减少消息大小**



**2、将对商家号Partition进行改成基于订单号Partition，避免不同分区大小不均问题**





## 已有Partition积压的消息处理

问题：大量历史消息积压需要处理

解决：多线程并发处理，提升消息消费速度

问题：DB中数量过多，千万级，导致更新记录操作缓慢

解决：识别业务不需要消息，将更早消息转储，提升SQL处理速度。

## 插入主键冲突问题

先尝试插入数据，如果记录已存在，则进行字段更新



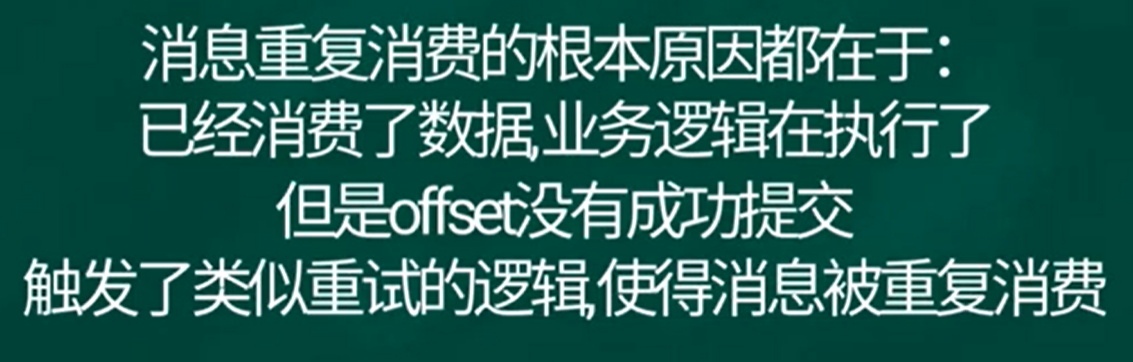
## 数据库主从延迟问题

解决方案：如果查询数据为空或只查到某种不完整数据，加入重试表，进行重试查询

## 重复消费问题

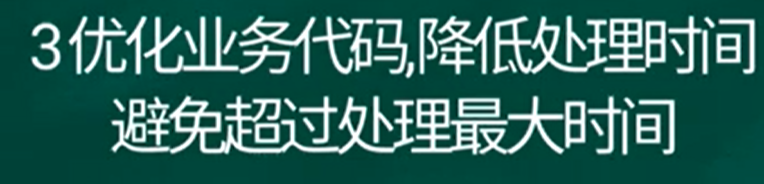
Kafka的3种消费模式

1. at most once：最多一次
2. at least once：至少一次 ---》业务必须做幂等设计
3. exactly once： 精确传递一次



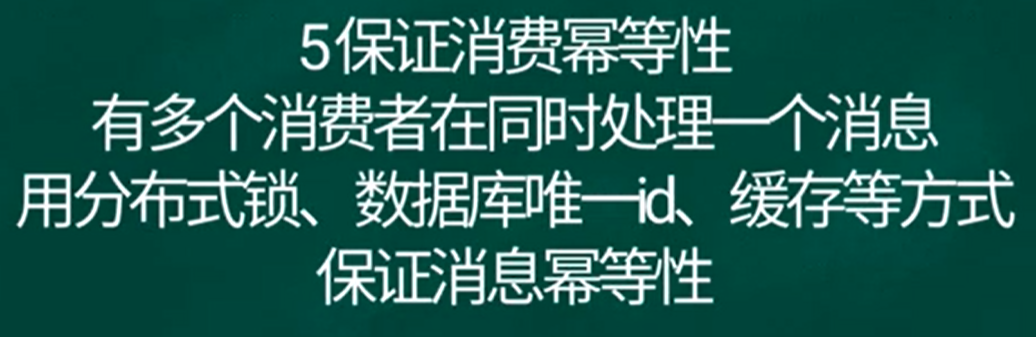












## 多环境消费问题

环境分为 pre （预发布环境）和prod（生产环境）

两个环境共数据库和共kafka集群

kafka的topic增加不同前缀区分不同环境的topic