1. 有序消息，消费端如果无法使用多线程消费，那数据量很大的情况下，消费端怎么样去提高吞吐量？

注意： 场景 按顺序处理消息

--- 客户端 使用多线程一次并发处理多条消息，无法控制先后顺序 ---

冲突两个核心点：并发和顺序

数据量很大：需要保障有序的消息 分散到不同的“分区”（kakfa的topic，rocket queue， rabbitmq queue）

1. 老师能不能讲讲pageCache是，学习kafka后一直觉得这个地方不是太明白，它和堆内存相比差别在哪儿？（pageCache操作系统维护的缓冲区）
2. 消息重复场景下，幂等校验应该怎么样考虑？ （消息唯一ID，消费者记录它处理过的消息id， 处理的时候判断是否重复）
3. 用MQ实现任务调度 ，调度中心 vs quartz（需要侵入业务系统依赖数据库）  
   设计原则的体现 – 迪米特法则 – 最少知识原则
4. Kafka#James老师说过， 对于消息顺序的保证， 从某种角度来说比ActiveMQ和RabbitMQ更加好， 哪种角度啊？ --- RocketMQ
5. mq-任务调度"能保证实时性 --- 增加MQ延时 --- 理论来说是非常快。但是实际，很多调度中心，对于精度要求比较高， 分布式调度中心采取 调度中心 -RPC-业务系统 定时方法

MQ 接收 --- 存储 ---分发

原理：客户端 发送 – JVM本地缓冲区（jdk容器缓冲msg） （TCP网络连接 – 长连接，自定义协议）

MQ服务器接收 --- 网络编程监听端口，接收数据（如果有配置 – 一般就是并发级别，限流参数）

MQ服务器处理 – 存储 – 为了高可用 --- 1. 存文件 (刷盘) 2. 副本节点同步

分发-推 --

分发-拉 --   
重要点：每次提前获取(prefetch)的消息数量  
注意点：推拉 java API你可能看不出来~~

Java --- linkedblockingqueue --- thread循环消费

Disruptor --- JAVA单机消息队列

封装了 生产者 消费者 这种开发框架 ---

优化点1： 消费者的执行策略很多封装

优化点2： 容器 – 无锁编程 （CAS机制）

底层程度：和CPU打交道--- CPU亲和度