1、消息队列的必要性

1、解耦

将原本紧密依赖的逻辑进行拆分为多个模块，降低各个模块之间的相互依赖关系，减少模块之间的强依赖。
提高 扩展性以及
A 中 同时会依赖 （B,C,D）,不利于扩展，新增逻辑的时候，需要修改源码，发布。

2、异步

将 业务数库交付给消息队列后，能够不依赖后续的逻辑处理；
提升吞吐量的同时，能够把后续的逻辑进行切分

3、削峰

当某一时刻，大量请求打到服务器后，处理业务的逻辑可能跟不上，此时，通过消息队列将请求缓存起来，等待消费者去处理逻辑
不止（链接资源）

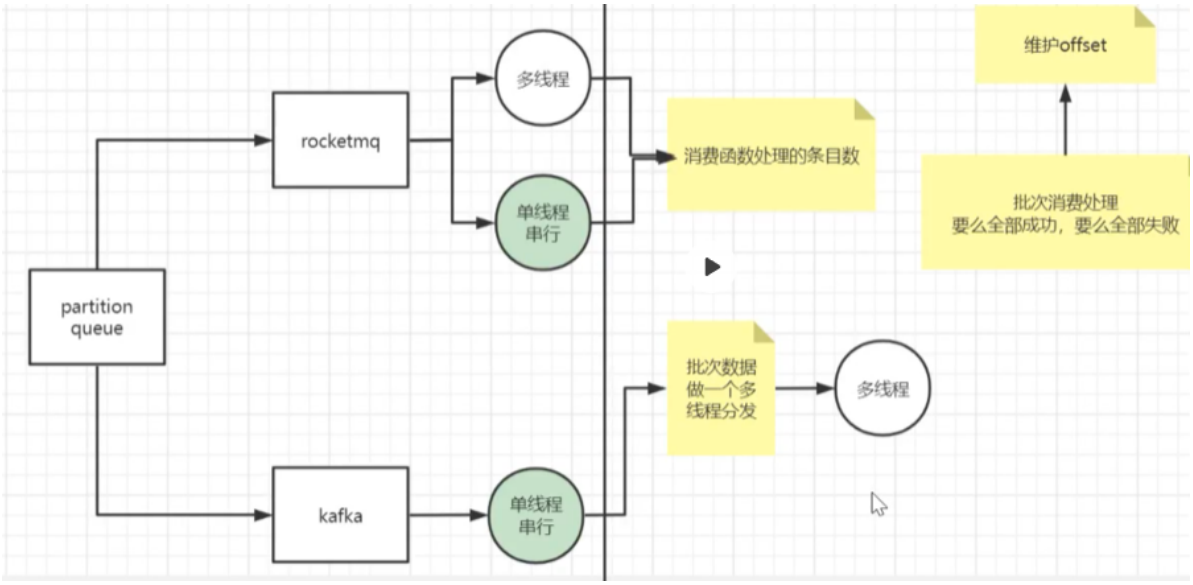
注意：
　　响应怎么处理？？？？

2、消息队列的选择

a、rocketMq
b、kafka
c、rocketMq

拉取一批 数据处理时，offset 的维护逻辑

	rabbitMq	kafka	rocketMq
出现			
传递性	消息级别	批量写入	消息级别
有序性	关注并发	保证有序 单线程拉取数据	关注并发



3、如何实现高吞吐

```
1、IO 是 瓶颈
    数据推送到消息队列 推送的体量（按集合推送）
```

4、消息幂等性

```
1、数据拉取 && 数据处理
    a、拉取数据后，自动提交 --> 数据的处理阶段处理失败，数据丢失
    b、拉取数据后，逻辑处理，手动提交 --> 需要结合幂等
    c、基于事务内部的消息应答 --> 事务提交的出错，消息丢失
2、幂等性的保证 --> 面向重复开放
```

5、消息不会丢失

```
1、生产者数据丢失（面向重复开放）
    a、可靠性数据投递（ack）
    b、ack 事务投递（性能影响）
2、broker 数据丢失（单点问题，副本可靠）
    a、增加副本
    b、数据持久化
3、消费者数据丢失
    参4
```

5、消息积压

```
1、增加消费能力 --> 设计的问题（分区partition/队列Queue）
    消费者线程比分区少（X）
2、增加分治负载能力 --> 消息会导致乱序的问题
    a、rocketMq 对 队列 进行读写权限的限定，可以控制一部分 Q 对消费者不可见
    b、
```

6、消息顺序的一致性

```
1、Queue -- FIFO -- 一个生产一个消费 -- 天然满足

2、业务逻辑的有序性 -- 业务有序
    控制生产者 按 业务要求有序性 的 推送到消息队列
```

7、redis + mq + mysql 可行

```
1、最终一致性
    1、增减类
    2、集合类

对于值覆盖的有序，？？？
```

7、延迟队列与定时消息

```
延迟队列 -- time 后 消息可见
定时消息 -- time 内 消息可见
```

8、可靠投递与事务消息

```
1、可靠投递
    生产者消息能不能保证投递成功
2、事务消息 -- 操作事务保证
    a、kafaka -- 消息提交，消费不可见，提交 commit-warkmark 确认后，消费者可见
    b、rocketMq -- 提交数据到 事务队列，消费不可见 ，commit ,转移到 真实队列 ，消费者可见
注意 ：kafaka 生产者保证事务；rocketMq 回查机制
```