# 基础知识篇

# 消息队列的概念

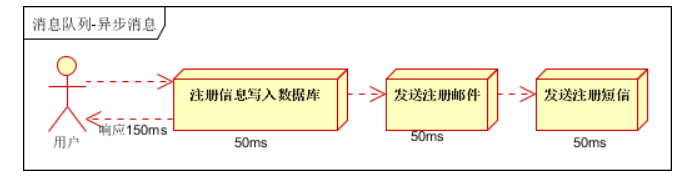
用来一个实现inter-process-communication,或者说是实现了inter-threads-communication within in a process.

强调的是用异步解耦的方式用于应用的解耦，异步消息，流量削峰的问题，最终实现分布式高可用，高性能，高一致性的架构。

## 概念模型

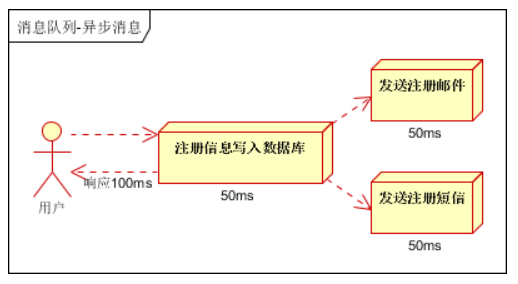
### 1. 异步

（1）在一般的**串行**模型中（常用的为邮件发送结构）：



这个模型总共响应时间为 3\*50ms=150ms

（2）通过**异步方式（并行）**解耦：

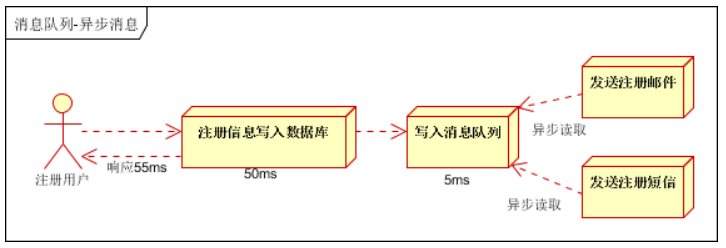


响应时间即刻降为2\*50 =100ms

因为CPU在单位时间内处理的请求数是一定的，假设CPU1秒内吞吐量是100次。则串行方式1秒内CPU可处理的请求量是7次（1000/150）。并行方式处理的请求量是10次（1000/100）（并行可以略高效与串行）

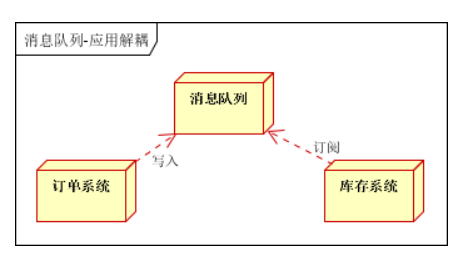
小结：如以上案例描述，传统的方式系统的性能（**并发量**，**吞吐量**，**响应时间**）会有瓶颈。如何解决这个问题呢？

（3）通过引入**消息队列**进行模型升级：



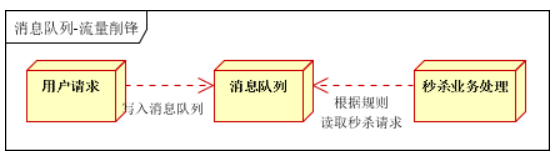
从图中就可以看出，注册这个**服务**对于整个应用而言是仅仅于在用户提交信息后将信息**写入数据库**后，继续将队列标识信息**写入消息队列**。整个过程 50ms+5ms=55ms

### 2． 应用解耦



这是一个MQ典型的pub/sub模式， 订单系统在用户下单成功后， 对MQ进行publish。 而库存系统对于消息队列中的内容进行订阅抽取后进行操作。看的出来消息队列可以很成功的把两个系统进行解耦。Kiss(keep it simple and stupid )让每个系统的部分独立，逻辑清晰。

### 3． 流量削峰



大量用户在面对秒杀系统的时候都有一个明显的特点，卡点，准时塞入大量请求，这个时候首先可以肯定的是串行式的系统必然爆炸。当然还有很多别的优化，在这里只针对MQ。

引入MQ，客户端或者浏览器只需要完成写入队列的这一步。后端业务系统只需要挨个取出相应处理。

**\***但我有个疑惑， 在流量削峰中的秒杀系统， 把用户请求写入队列， 系统取出队列挨个处理， 这个不就是很像代理模式吗，使用了代理模式，我们建立了一个代理类， 同样是用队列的结构去代理请求。 请问这两个的异同处在哪呢

### 4.日志处理

# 实现队列需要注意的地方