# 1.MQ

## 1.1如何进行消息队列的技术选型？

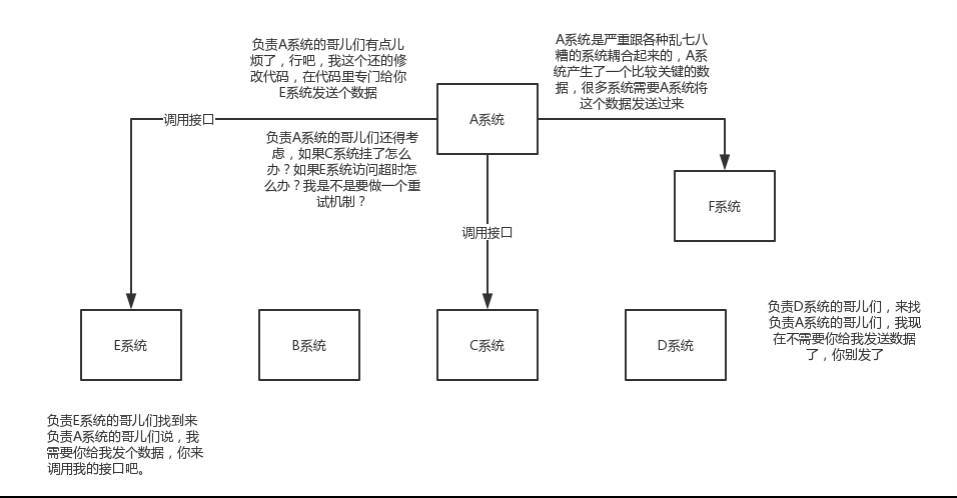
### 1.1.1为什么使用消息队列？

其实就是说消息队列都有哪些使用场景，在项目中具体是什么场景，这个业务场景有什么技术挑战，如果不用MQ可能会很麻烦，但是用了MQ之后带来了很多好处。

**场景有很多，核心的有3个：解耦、异步、削峰；**

**耦合时**：A系统发送个数据到BCD三个系统里面去，接口调用发送，那如果E系统也要这个数据呢？那如果C系统现在不需要了呢？现在A系统又要发送第二种数据了呢？A系统负责人濒临崩溃。再来点更加崩溃的事儿，A系统要时时刻刻考虑BCDE四个系统如果挂了咋办？我要不要重发？我要不要把消息存起来？

**耦合时的图解**：

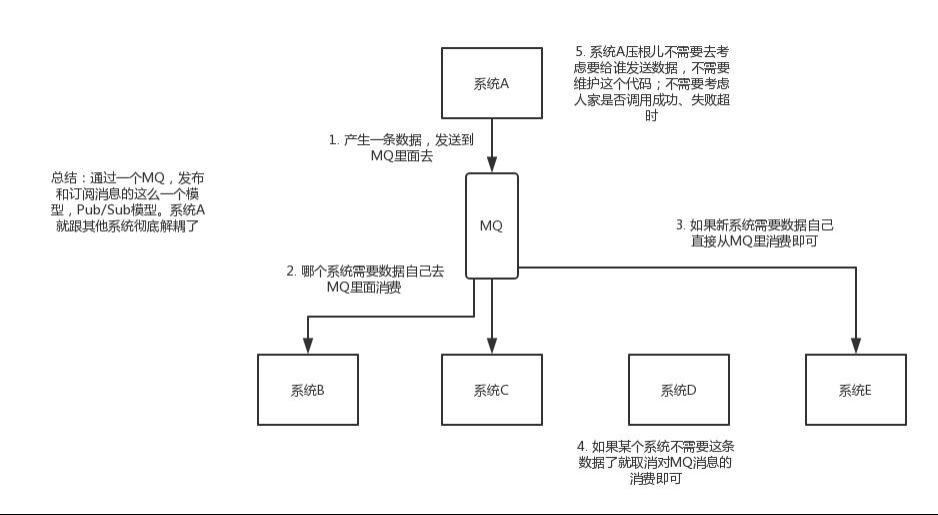


**解耦时：**A系统发送一条数据到MQ里面去，其它系统谁需要这条数据，谁就去消费，其它需要数据的系统的增减不再影响到A系统，A系统不用为了其它系统而频繁地改动、发布；

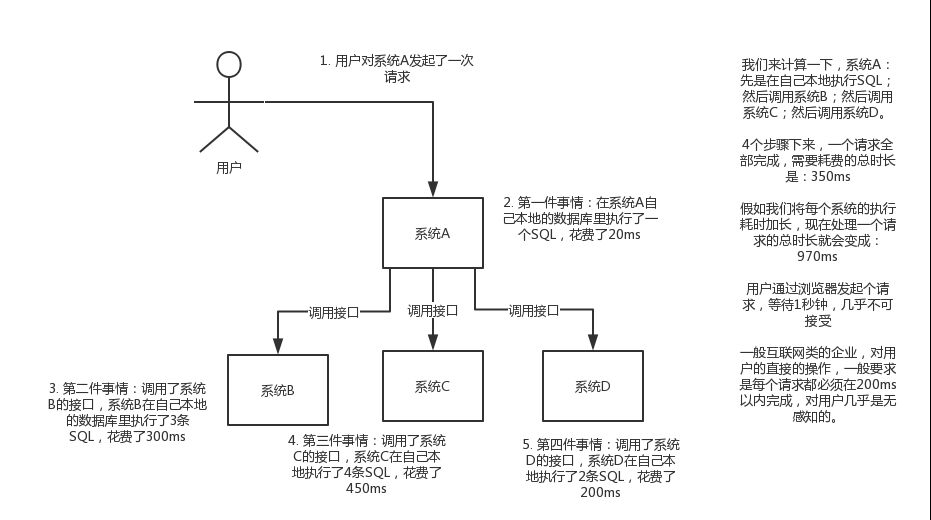
你需要去考虑一下你负责的系统中是否有类似的场景，就是一个系统或者一个模块，调用了多个系统或者模块，互相之间的调用很复杂，维护起来很麻烦。但是其实这个调用是不需要直接同步调用接口的，如果用MQ给他异步化解耦，也是可以的，你就需要去考虑在你的项目里，是不是可以运用这个MQ去进行系统的解耦。

例如对于最近工作的项目而言，关于设备指令，设备指令的统一处理在设备服务，而需要发送设备指令的业务分布在各个服务，许多相关的业务都需要触发设备指令的推送，我们首先服务之间是必须解耦的，因为不同服务都在不同的服务器，不能互相调用，也不能在controller层进行触发推送，因为对于service层的同一个业务场景下的不同情况会有不同的指令推送，这个时候就需要MQ，设备指令的统一处理时去MQ中订阅指令消息。

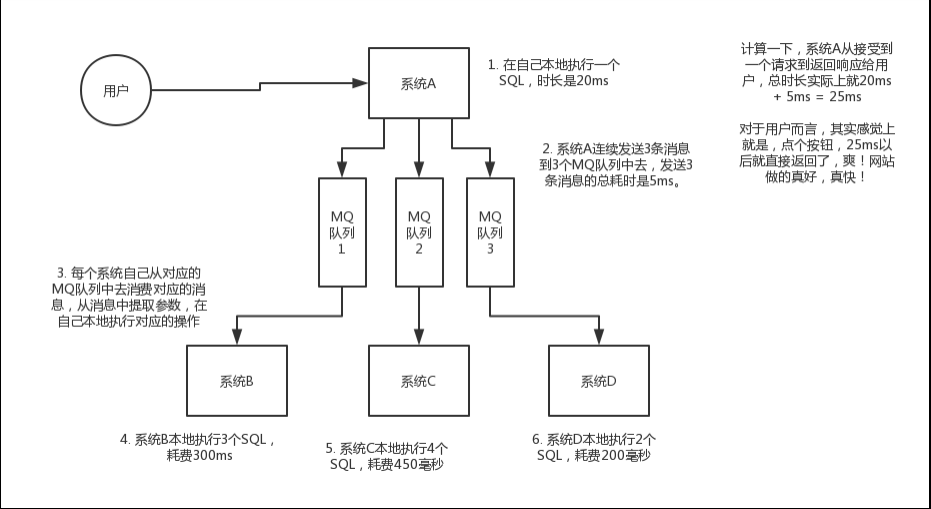
**解耦时图解：**



**同步高延时请求场景**：

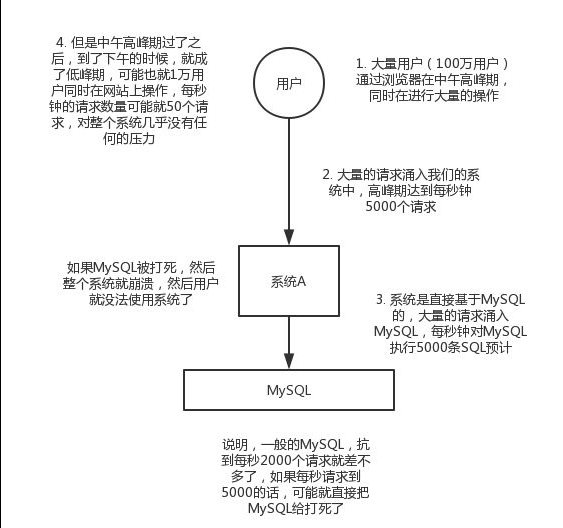


**使用MQ异步之后**：

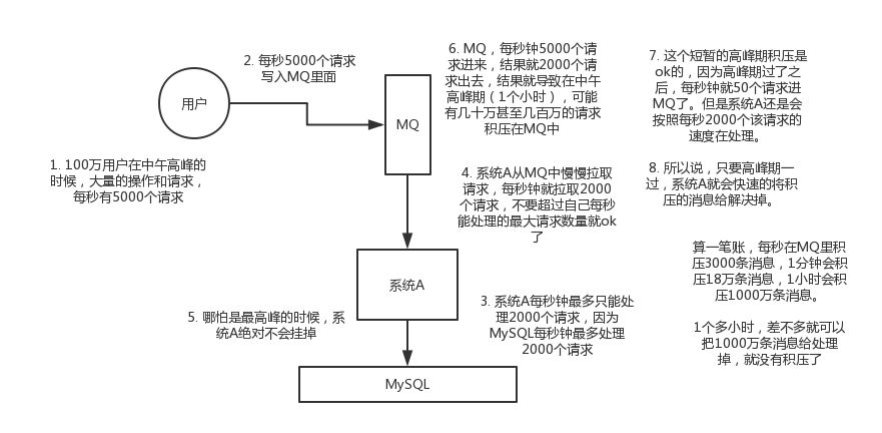


**关于削峰场景**：每天0点到11点，A系统风平浪静，每秒并发请求数量就100个。结果每次一到11点~1点，每秒并发请求数量突然会暴增到1万条。但是系统最大的处理能力就只能是每秒钟处理1000个请求，怎么办？

**无削峰**：



**有削峰**：



### 1.1.2消息队列有什么优点和缺点？

优点上面已经说了，就是在特殊场景下有其对应的好处，解耦、异步、削峰；

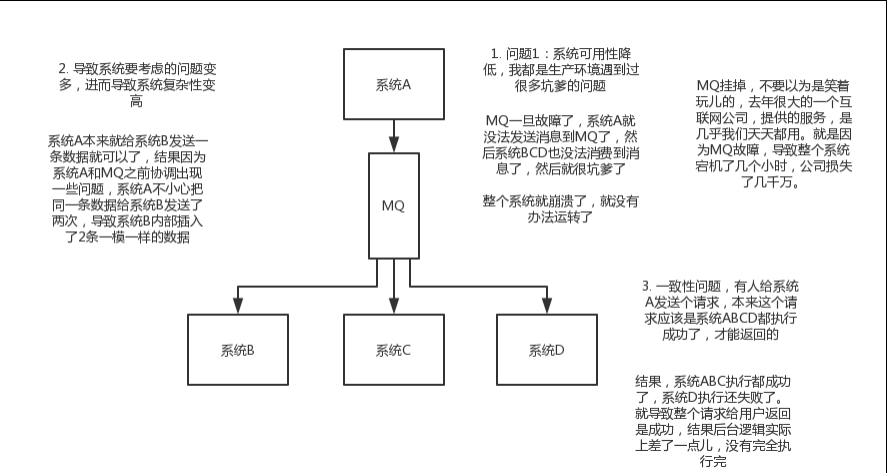
缺点呢？显而易见的。

系统可用性降低：系统引入的外部依赖越多，越容易挂掉，本来你就是A系统调用BCD三个系统的接口就好了，人ABCD四个系统好好的，没啥问题，你偏加个MQ进来，万一MQ挂了咋整？MQ挂了，整套系统崩溃了，你不就完了么；

系统复杂性提高：硬生生加个MQ进来，你怎么保证消息没有重复消费？怎么处理消息丢失的情况？怎么保证消息传递的顺序性？头大头大，问题一大堆，痛苦不已；

一致性问题：A系统处理完了直接返回成功了，人都以为你这个请求就成功了；但是问题是，要是BCD三个系统那里，BD两个系统写库成功了，结果C系统写库失败了，咋整？你这数据就不一致了；

所以消息队列实际是一种非常复杂的架构，你引入它有很多好处，但是也得针对它带来的坏处做各种额外的技术方案和架构来规避掉，最好之后，你会发现，妈呀，系统复杂度提升了一个数量级，也许是复杂了10倍。但是关键时刻，用，还是得用的。



### 1.1.3 Kafka、activemq、RabbitMQ、rocketmq都有什么区别以及适合哪些场景？