**消息中间件学习笔记**

# 1 消息中间件介绍

消息中间件是在消息的传输过程中保存消息的容器，是消息生产者和消息消费者之间的中间人。消息中间件主要目的是提供路由(生产者-->消费者)并保证消息可靠到达目的地，当消费者离线，暂时无法接收消息，消息中间件会保留消息队列，直到接收着上线接收消息为止。当然根据业务，可以设置消息队列的有效期，消息过期后会被删除。

消息中间件主要特点：

(1) 采用异步处理模式

消息发送折发送一个消息出去后无需等待响应，消息生产者将消息发送到一条的通道(主题或队列)上，消息消费者则订阅或监听该通道。一条消息可能最终转发给一个或多个消息消费者，这些消费者都无需对消息发送着做出同步回应，整个过程是异步的。

例如用户注册信息，注册完毕后一段事件后发送短息或邮件通知用户注册成功。

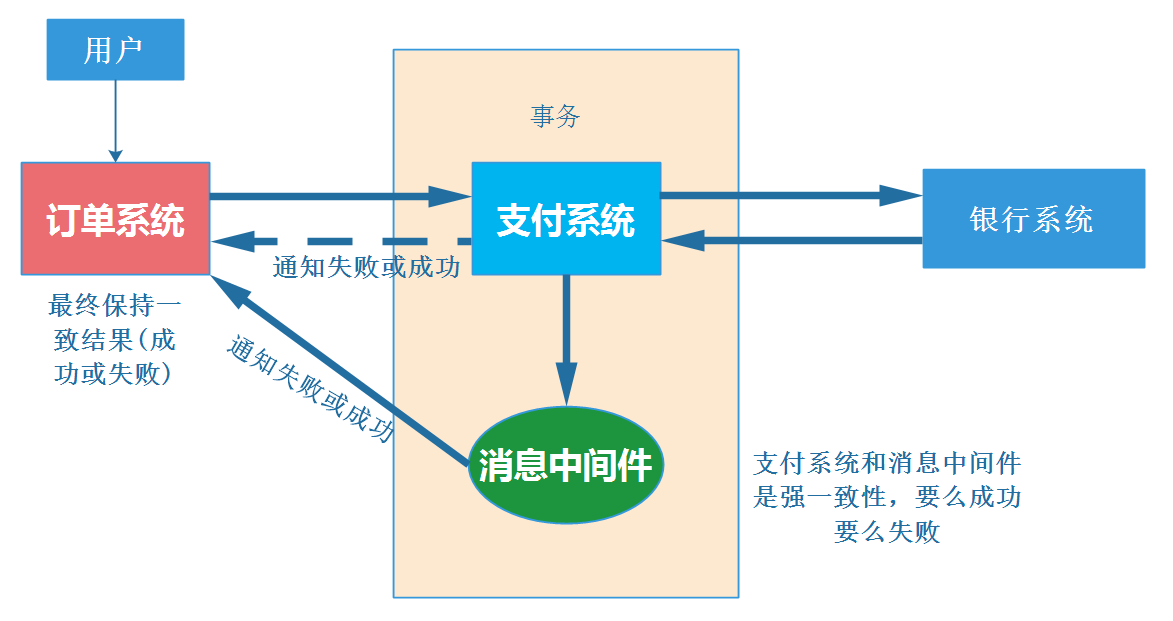
(2) 使得消息生产者和消费者之间松耦合

生产者和消费者不必了解对方，只需确认消息。

生产者和消费者不必同时在线。

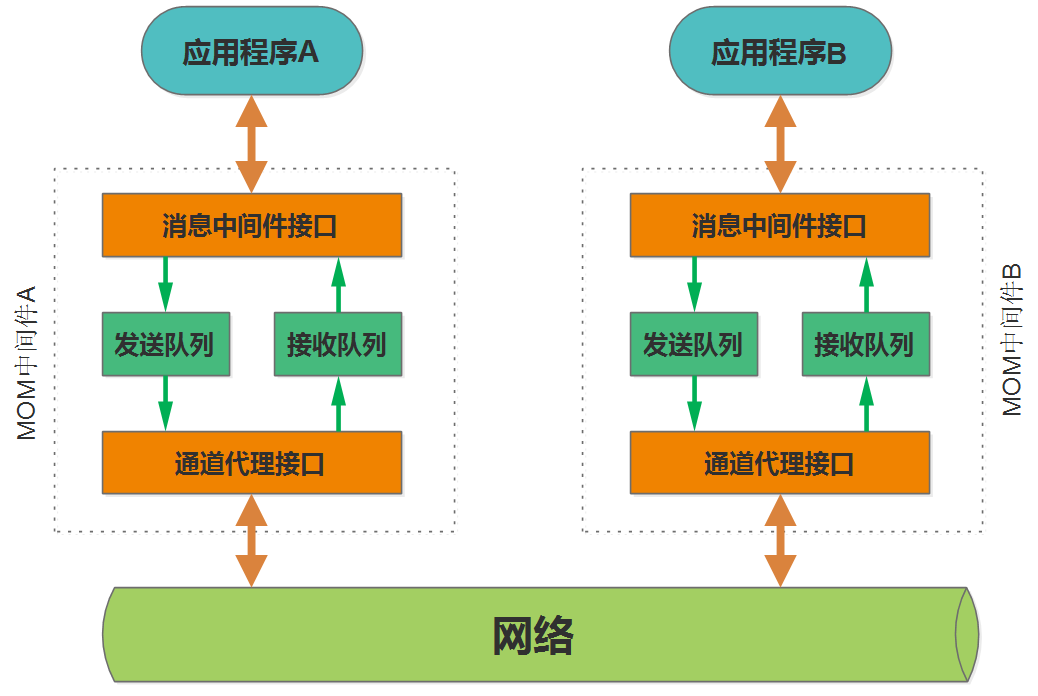
例如淘宝店，店主人不会一直在线，当有顾客在店里购买了商品，淘宝会通知店家发货给顾客。淘宝就相当于消息中间件，店主和顾客之间是解耦的，两者之间只需确认消息即可。

例如在线交易系统为了保证数据的最终一致，在支付系统能够处理完毕后把支付结果放到消息中间件里，消息中间件再发送消息通知订单系统修改订单支付状态，两个系统在消息中间件之间是解耦的。如下图所示：



# 2 消息传递服务模型

消息传递服务模型如下图所示：



假如应用程序A为消息生产者，应用程序B为消息消费者， A通过消息中间件接口把消息写入发送队列，然后通过通道代理接口发送到网络，B通道代理接口读取有消息过来，把消息写入接收队列，再由消息中间件接口把消息推给应用程序B。

当然消费者的消息中间件可以去掉，去掉后有个问题是，如果消息生产者频繁发送消息，消费者来不及处理或不在线，会造成发送队列大量堆积，造成整体性能下降。

消息传递服务模型分两类：点对点模型(PTP)和发布-订阅模型(Pub/Sub)

## 2.1 点对点模型

点对点模型也就是一个消息生成者对应一个消息消费者。消息生产者将消息发送到由某个名字标识(即消息服务队列)的特定消费者，在消息传递给消费者之前它被存储在这个队列中，队列小写可以放在内存中，也可以持久化，保证在消息服务出现故障时仍能够传递消息。

点对点模型特性：

1. 每个消息只用一个消费者
2. 发送者和接收者之间没有时间依赖
3. 接收者确认消息接收和处理成功



## 2.2 发布-订阅模型

发布-订阅模型支持向一个特定的消息主体生产消息。0个或多个订阅者可能对接收来自特定消息主体的消息感兴趣。在这种模型下，发布者和订阅者彼此不知道对方，类似于匿名公告版，生产者发布消息出去，多个消费者可以得到消息，发布者和订阅者之间存在时间的依赖，发布者首先需要建立一个订阅(subscription)，以便能够给消费者订阅。订阅者必须保持持续的活动状态才能接收消息，除非订阅者和发布者建立了持久的订阅关系，在这种情况下，当前订阅者离线也没关系，在订阅者连接时会重新发布。

发布-订阅模型的特性：

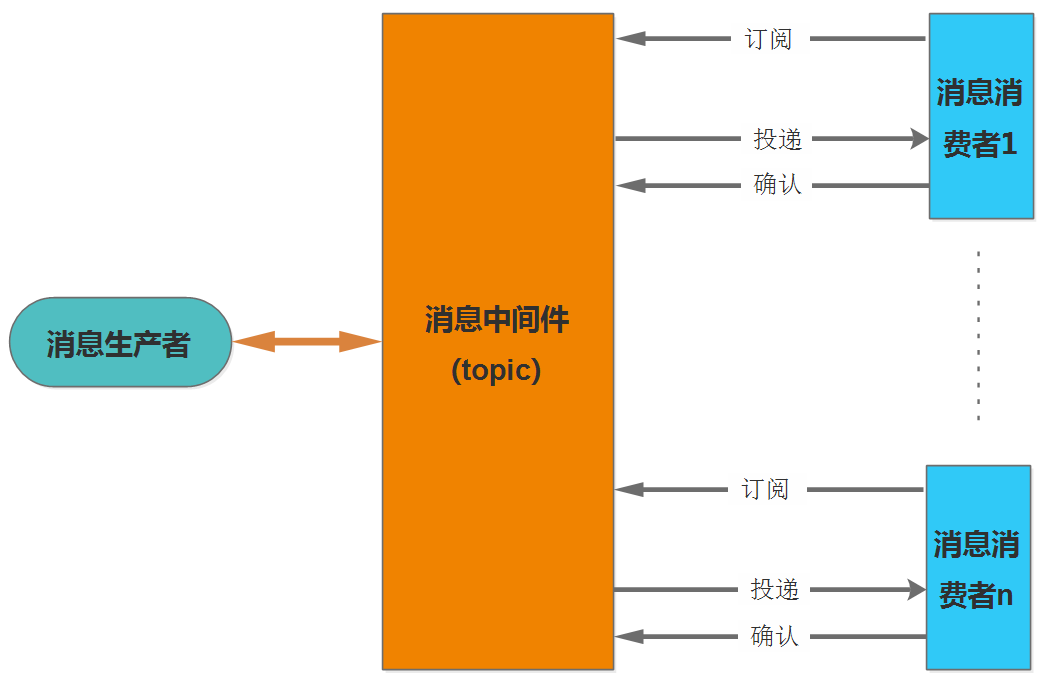
1. 每个消息可以有多个订阅者
2. 发布者和订阅者有时间依赖

接收者和发布者只有建立订阅关系，接收者才能接收到消息，类似如用户没有订报纸的话，不可能免费发报纸给用户

1. 分为持久订阅和非持久订阅

持久订阅：订阅关系建立后，消息就不会消失，不管订阅者是否在线，比较常用。

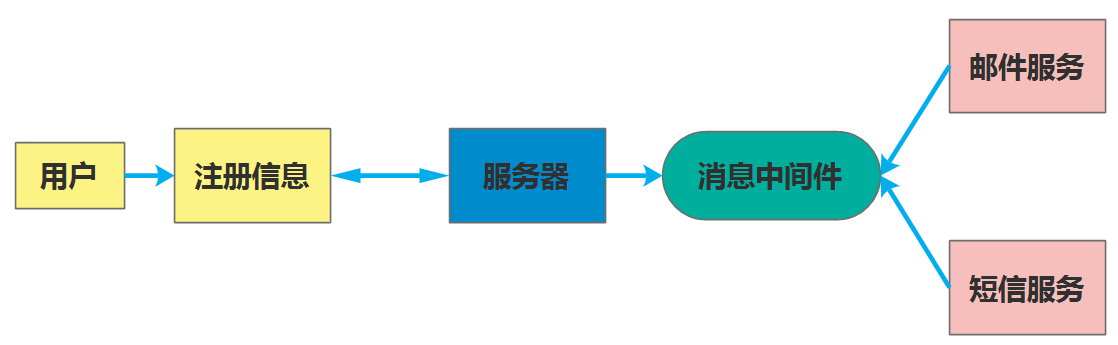
非持久订阅：订阅者为了接收消息，必须一直在线，当只有一个订阅者时类似与点对点模型。比较少用，例如商家促销搞活动。



# 3 互联网消息中间件的应用场景

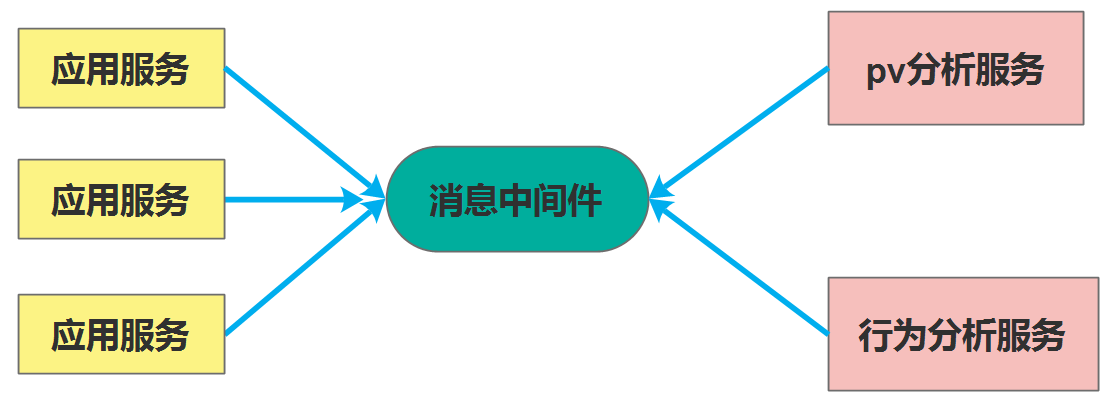
## 3.1 网站用户注册(发布-订阅)

注册成功后会过一会发送邮件或短信确认，使用的是发布-订阅模型



## 3.2 日志分析(发布-订阅)

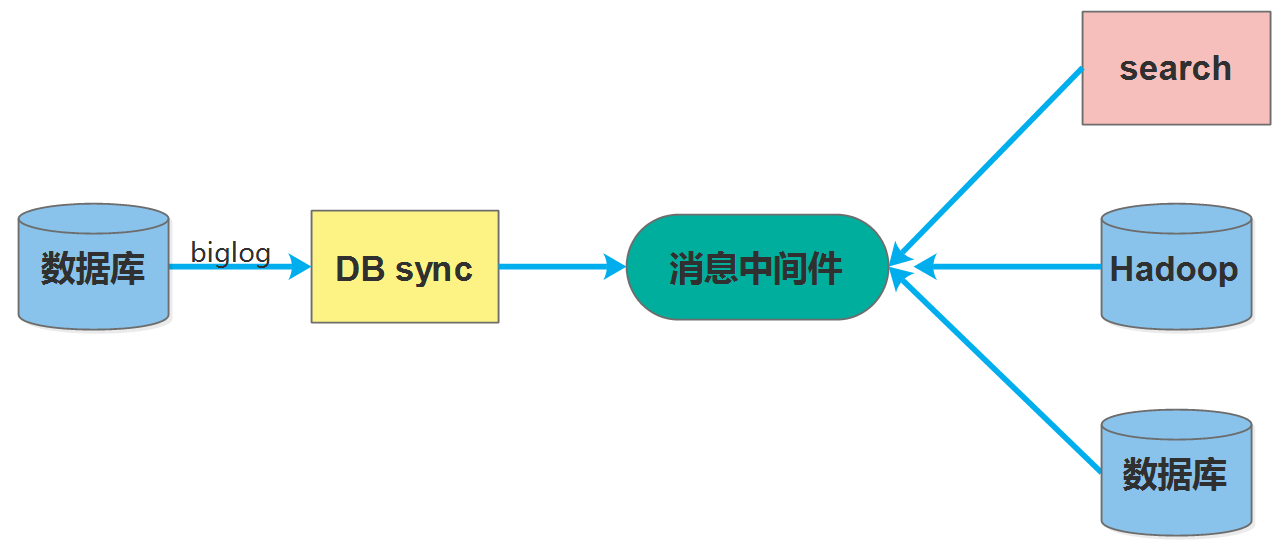
把服务器产生的日志进行集中收集，用于计算pv、用户行为分析。例如应用程序改版后，先要试运营，在访问旧版本用户分流很小一部分给新版本，然后收集用户行为，通过行为分析出新版本应用比旧版本好在哪里。



## 3.3 数据复制(发布-订阅)

将源头数据复制到多个目的地，要求顺序或者保证因果序列，最终保证数据一致性。

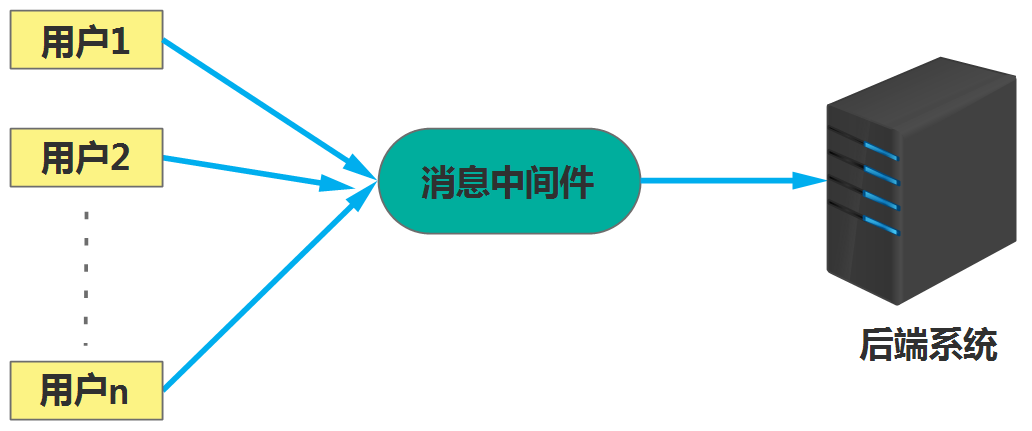
用于跨机房数据传输、搜索、离线数据计算等，如下图所示：



## 3.4 延时消息发送和暂存(点对点)

把消息中间件当成可靠的消息暂存地，然后进行定时消息投递，比如模拟用户秒杀访问，进行系统性能压测。

首先收集用户访问行为，把这些访问行为暂存在消息中间件里，暂时不消费消息，，把待测的后端系统接上来，消息中间件集中访问后端系统，这些消息消费了也不删除，可以重复利用，重复压测，能真实反映系统的性能。

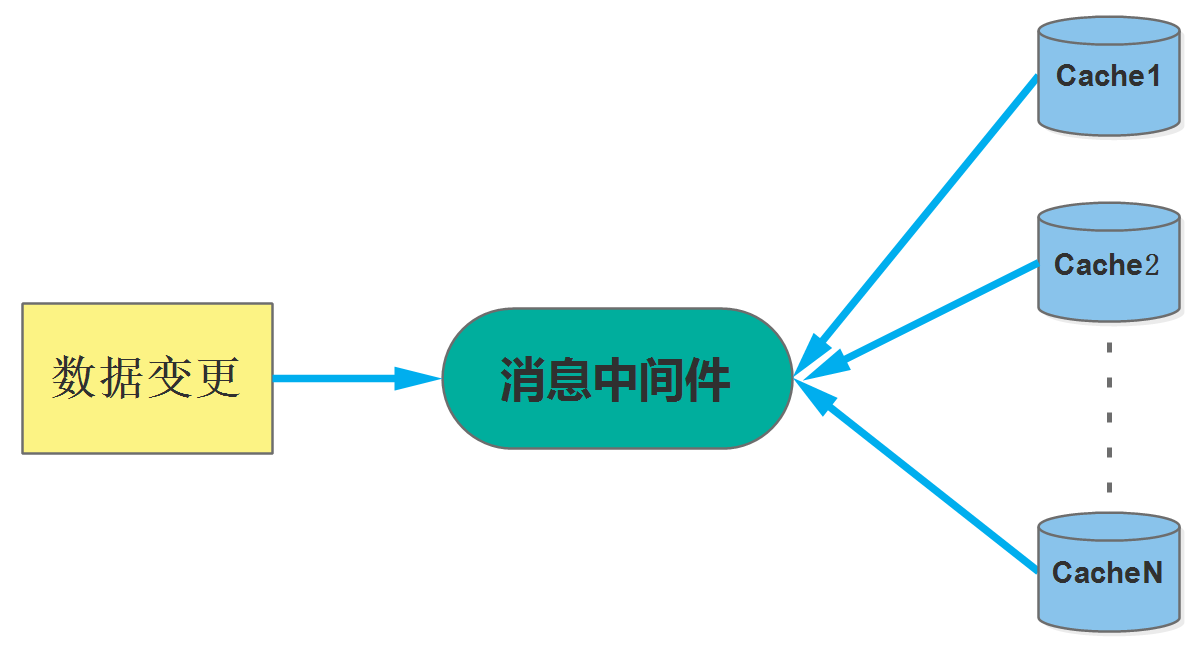


## 3.5 消息广播(发布-订阅)

(1) 缓存数据同步更新

(2) 网应用推送数据

为了提高性能，一般把数据缓存到内存，多个缓存服务器之间会有缓存数据一致性问题，用消息中间件往缓存服务器推送数据，最终保证数据一直性。例如修改商品价格，最终要保证该商品价格在各个缓存服务器是一致的。



# 4 消息中间件分类

(1) 推消息模型(push)：消息生产者将消息发送给消息中间件，消息中间件把消息推给消息消费者，时效性高。

(2) 拉消息模型(pull)：消息消费者主动从消息中间件拉取消息。时效性差，取决于消费者。

