**推送的概念**

**<https://zhuanlan.zhihu.com/p/19801751>**

### 任意门推送-开源实时信息推送服务器

**<http://www.ddpush.net/overview>**

[dwr 后台推送技术](http://blog.csdn.net/is_zhoufeng/article/details/7641234)

<http://blog.csdn.net/is_zhoufeng/article/details/7641234>

Js可以直接调用web后台Java代码。

JPush ,APP推送

<https://www.mailgun.com/> 邮件推送

activeMQ 是Apache出品，最流行的，能力强劲的开源消息总线。

Kafka是一种高吞吐量的[分布式](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F" \t "https://baike.baidu.com/item/Kafka/_blank)发布订阅消息系统。

http://orchome.com/6

## ****为什么需要消息系统****

1. **解耦：**　　允许你独立的扩展或修改两边的处理过程，只要确保它们遵守同样的接口约束。

**2.冗余：**

消息队列把数据进行持久化直到它们已经被完全处理，通过这一方式规避了数据丢失风险。许多消息队列所采用的"插入-获取-删除"范式中，在把一个消息从队列中删除之前，需要你的处理系统明确的指出该消息已经被处理完毕，从而确保你的数据被安全的保存直到你使用完毕。

**3.扩展性：**

因为消息队列解耦了你的处理过程，所以增大消息入队和处理的频率是很容易的，只要另外增加处理过程即可。

**4.灵活性 & 峰值处理能力：**

在访问量剧增的情况下，应用仍然需要继续发挥作用，但是这样的突发流量并不常见。如果为以能处理这类峰值访问为标准来投入资源随时待命无疑是巨大的浪费。使用消息队列能够使关键组件顶住突发的访问压力，而不会因为突发的超负荷的请求而完全崩溃。

**5.可恢复性：**

系统的一部分组件失效时，不会影响到整个系统。消息队列降低了进程间的耦合度，所以即使一个处理消息的进程挂掉，加入队列中的消息仍然可以在系统恢复后被处理。

**6.顺序保证：**

在大多使用场景下，数据处理的顺序都很重要。大部分消息队列本来就是排序的，并且能保证数据会按照特定的顺序来处理。（Kafka 保证一个 Partition 内的消息的有序性）

**7.缓冲：**

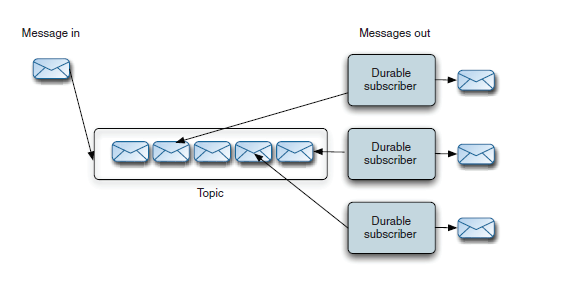
有助于控制和优化数据流经过系统的速度，解决生产消息和消费消息的处理速度不一致的情况。

**8.异步通信：**

　　很多时候，用户不想也不需要立即处理消息。消息队列提供了异步处理机制，允许用户把一个消息放入队列，但并不立即处理它。想向队列中放入多少消息就放多少，然后在需要的时候再去处理它们。

使用消息队列，可以动态实现负载均衡，入队和出队处理速率。

MOM 就是面向消息中间件（Message-oriented middleware），是用于以分布式应用或系统中的异步、松耦合、可靠、可扩展和安全通信的一类软件。MOM 的总体思想是它作为消息发送器和消息接收器之间的消息中介,这种中介提供了一个全新水平的松耦合。



每一类的消息称之为一个主题(Topic).

****Topic：****是消息的分类名。如果从技术角度，topic标签实际就是队列，生产者把所有“鸡蛋（消息）”都放到对应的队列里了，消费者到指定的队列里取。

通常来讲，消息模型可以分为两种， 队列和发布-订阅式。 队列的处理方式是 一组消费者从服务器读取消息，一条消息只有其中的一个消费者来处理。在发布-订阅模型中，消息被广播给所有的消费者，接收到消息的消费者都可以处理此消息。

Web后台发短信的示例代码

<http://ask.csdn.net/questions/23097>

MPush

Jetty

根据 CAP 定理：

    Consistency(一致性), 数据一致更新，所有数据变动都是同步的。

    Availability(可用性), 好的响应性能。

    Partition tolerance(分区容错性) 可靠性。

    定理：**任何分布式系统只可同时满足二点，没法三者兼顾。**

在处理某条数据的时候报错了，且重试了3次仍然不成功，后面还有大量数据等着推送，线程不能阻塞，那么只能将推送失败的数据记录下来，继续处理后面来的数据。

如果推送失败，不要阻塞后面的数据，采用将数据记录下来（放内存里面），等后面一批数据来后，同时处理，再次推送，如果这次推送仍然失败，则再重复这个过程，一共重试3~5次。

根据具体的业务场景，如果当某类数据多次无法推送时，主动关闭该数据的推送，同时触发监控程序，通知维护人员，由维护人员决定，是否去处理故障，如果不需要处理，则该数据就一直关闭，如果需要处理，则当故障恢复时，重新启动该数据的推送任务。也可以由monitor主动去监控数据推送是否恢复正常，恢复正常后，自动恢复推送。