# 0721技术整理-消息中间件

## 概念

消息中间件属于分布式系统中一个子系统，关注于数据的发送和接收，利用高效可靠的异步消息传递机制对分布式系统中的其余各个子系统进行集成。

它具有 **低耦合**、**可靠投递**、**广播**、**流量控制**、**最终一致性** 等一系列功能。

优势：

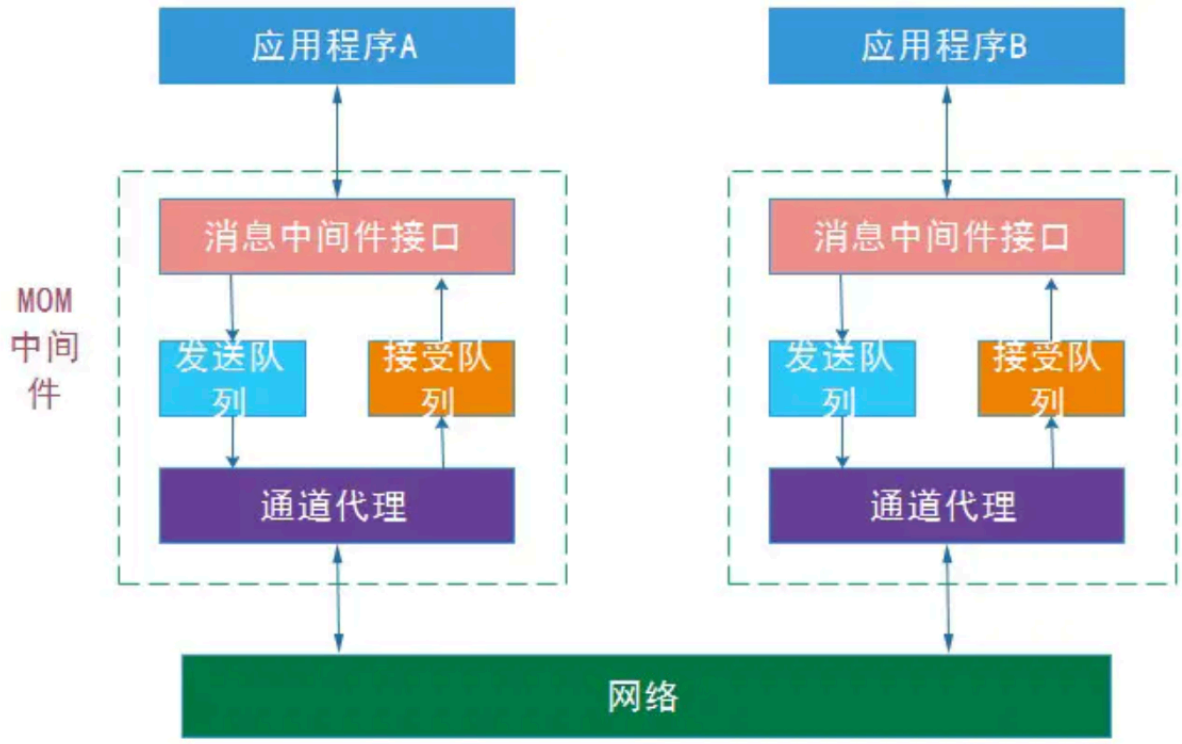
- 低耦合，不管是程序还是模块之间，使用消息中间件进行间接通信。
- 异步通信能力，使得子系统之间得以充分执行自己的逻辑而无需等待。
- 缓冲能力，消息中间件像是一个巨大的蓄水池，将高峰期大量的请求存储下来慢慢交给后台进行处理，对于秒杀业务来说尤为重要。

和RPC有什么区别?

RPC方式是典型的同步方式，让远程调用像本地调用。消息中间件方式属于异步方式。消息队列是系统级、模块级的通信。RPC是对象级、函数级通信。

## 传递服务模型/传输模式

**传递服务模型**




**传输模式**

点对点模型

**点对点模型** 用于 **消息生产者** 和 **消息消费者** 之间 **点到点** 的通信。消息生产者将消息发送到由某个名字标识的特定消费者。这个名字实际上对于消费服务中的一个 **队列**（`Queue`），在消息传递给消费者之前它被 **存储** 在这个队列中。**队列消息** 可以放在 **内存** 中也可以 **持久化**，以保证在消息服务出现故障时仍然能够传递消息。

发布/订阅模型（Pub/Sub）

**发布者/订阅者** 模型支持向一个特定的 **消息主题** 生产消息。`0` 或 **多个订阅者** 可能对接收来自 **特定消息主题** 的消息感兴趣。

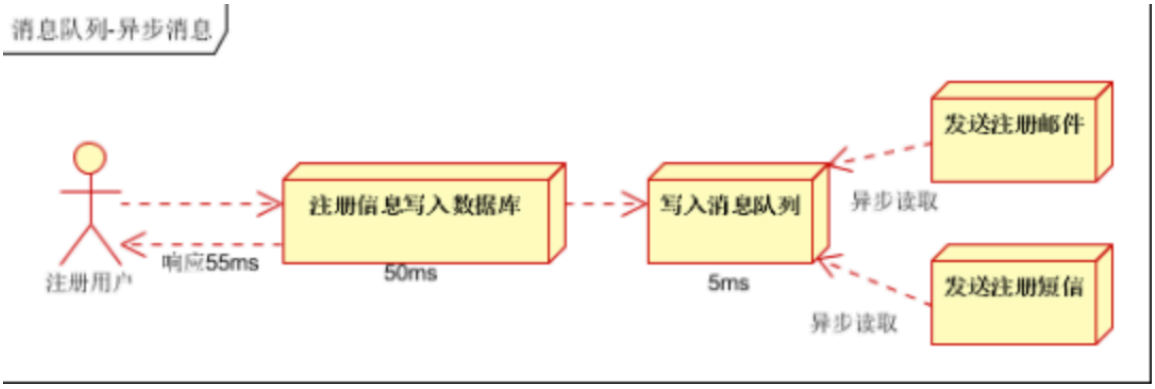
在这种模型下，发布者和订阅者彼此不知道对方，就好比是匿名公告板。这种模式被概况为：多个消费者可以获得消息，在 **发布者** 和 **订阅者** 之间存在 **时间依赖性**。发布者需要建立一个 **订阅**（`subscription`），以便能够消费者订阅。**订阅者** 必须保持 **持续的活动状态** 并 **接收消息**。

1. 持久订阅：订阅关系建立后，消息就不会消失，不管订阅者是否都在线；
2. 非持久订阅：订阅者为了接受消息，必须一直在线。 当只有一个订阅者时约等于点对点模式

# 使用场景

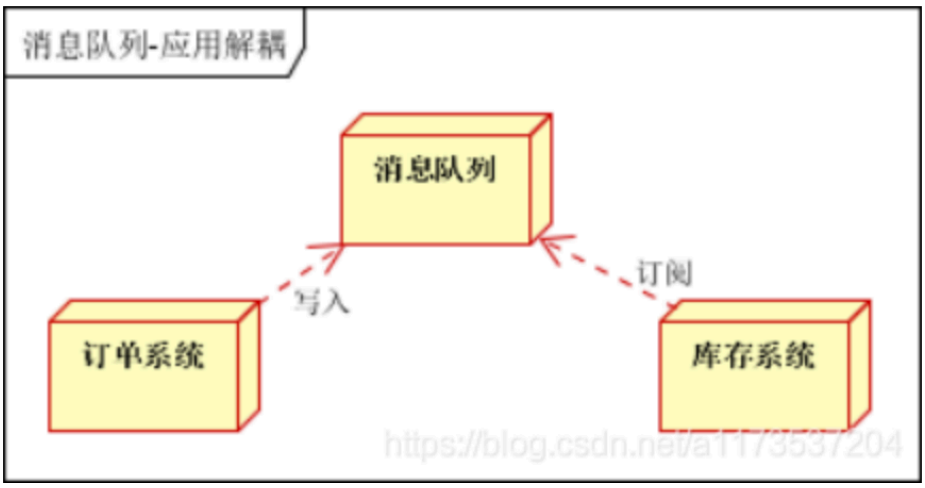
**异步处理**

将可以异步进行的部分，通过消息队列传递数据，实现异步处理。




**应用解耦**

使系统间不必互相依赖，耦合性降低。




订单系统：用户下单后，订单系统完成持久化处理，将消息写入消息队列，返回用户订单下单成功。

库存系统：订阅下单的消息，采用拉/推的方式，获取下单信息，库存系统根据下单信息，进行库存操作。

**流量削峰**

用户的请求，服务器接收后，首先写入消息队列。假如消息队列长度超过最大数量，则直接抛弃用户请求或跳转到错误页面；秒杀业务根据消息队列中的请求信息，再做后续处理。

**日志处理**

日志采集客户端，负责日志数据采集，定时写入Kafka队列：Kafka消息队列，负责日志数据的接收，存储和转发；日志处理应用：订阅并消费kafka队列中的日志数据；

**消息通讯**

消息通讯是指，消息队列一般都内置了高效的通信机制，因此也可以用在纯的消息通讯。比如实现点对点消息队列，或者聊天室等。

点对点通讯：客户端A和客户端B使用同一队列，进行消息通讯。

聊天室通讯：客户端A，客户端B，客户端N订阅同一主题，进行消息发布和接收。实现类似聊天室效果。

# RocketMQ

优势特点：

**单机** 支持 `1` 万以上 **持久化队列**；

`RocketMQ` 的所有消息都是 **持久化的**，先写入系统 `PAGECACHE`，然后 **刷盘**，可以保证 **内存** 与 **磁盘** 都有一份数据，而 **访问** 时，直接 **从内存读取**。

模型简单，接口易用（`JMS` 的接口很多场合并不太实用）；

**性能非常好**，可以允许 **大量堆积消息** 在 `Broker` 中；

支持 **多种消费模式**，包括 **集群消费**、**广播消费**等；

各个环节 **分布式扩展设计**，支持 **主从** 和 **高可用**；

开发度较活跃，版本更新很快。