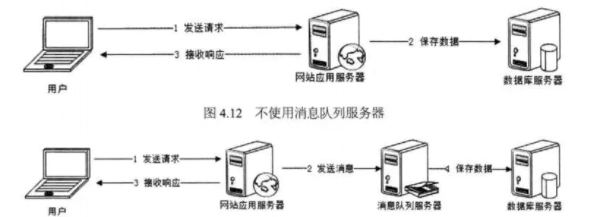
**1、消息队列**

我们可以把消息队列比作是一个存放消息的容器，当我们需要使用消息的时候可以取出消息供自己使用。消息队列是分布式系统中重要的组件，

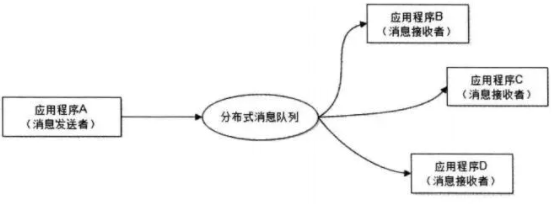
**优点：**

1、通过异步处理提高系统性能（削峰、减少响应所需时间）



在不使用消息队列服务器的时候，用户的请求数据直接写入数据库，在高并发的情况下数据库压力剧增，使得响应速度变慢。但是在使用消息队列之后，用户的请求数据发送给消息队列之后立即 返回，再由消息队列的**消费者进程**从消息队列中获取数据，异步写入数据库。由于消息队列服务器处理速度快于数据库（消息队列也比数据库有更好的伸缩性），因此响应速度得到大幅改善。

2、降低系统耦合性。



消息队列使利用**发布-订阅**模式工作，消息发送者（生产者）发布消息，一个或多个消息接受者（消费者）订阅消息。 从上图可以看到消息发送者（生产者）和消息接受者（消费者）之间没有直接耦合，消息发送者将消息发送至分布式消息队列即结束对消息的处理，消息接受者从分布式消息队列获取该消息后进行后续处理，并不需要知道该消息从何而来。对新增业务，只要对该类消息感兴趣，即可订阅该消息，对原有系统和业务没有任何影响，从而实现网站业务的可扩展性设计。

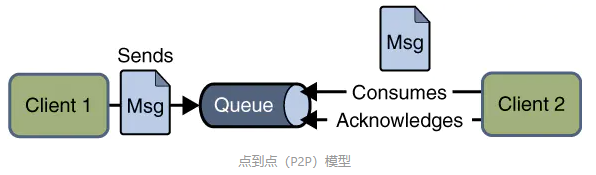
**问题：**

1、系统复杂性提高： 加入MQ之后，你需要保证消息没有被重复消费、处理消息丢失的情况、保证消息传递的顺序性等等问题！

2、一致性问题： 我上面讲了消息队列可以实现异步，消息队列带来的异步确实可以提高系统响应速度。但是，万一消息的真正消费者并没有正确消费消息怎么办？这样就会导致数据不一致的情况了!

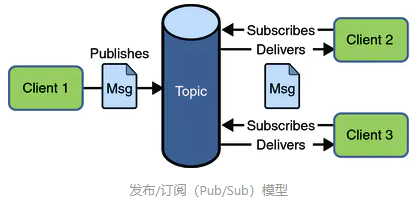
**2、消息队列模型**

**1、点到点（P2P）模型**



使用队列（Queue）作为消息通信载体；满足**生产者与消费者**模式，一条消息只能被一个消费者使用，未被消费的消息在队列中保留直到被消费或超时。比如：我们生产者发送100条消息的话，两个消费者来消费一般情况下两个消费者会按照消息发送的顺序各自消费一半。

**2、发布/订阅（Pub/Sub）模型**



发布订阅模型（Pub/Sub） 使用**主题（Topic）**作为消息通信载体，类似于广播模式；发布者发布一条消息，该消息通过主题传递给所有的订阅者，在一条消息广播之后才订阅的用户则是收不到该条消息的。

**3、进程调度算法**

1、先来先服务调度算法

2、短作业优先调度算法（非抢占/抢占）

3、优先级调度算法

4、高响应比优先调度算法

5、时间片轮转调度算法RR

**4、分区分配算法**

1、首次适应算法：地址递增

2、循环首次适应算法：地址递增

3、最佳适应算法：容量递增

4、最坏适应算法：容量递减

**5、页面置换算法**

**1、最佳置换算法（OPT）**（理想）：从主存中移出永远不再需要的页面；如无这样的页面存在，则选择最长时间不需要访问的页面。

**2、先进先出置换算法（FIFO）**：是最简单的页面置换算法，总是选择驻留主存时间最长的页面进行淘汰，即先进入主存的页面先淘汰。

**3、最近最久未使用（LRU）算法**：总是选择在最近一段时间内最久不用的页面予以淘汰。

**5、同步IO、异步IO、阻塞IO、非阻塞IO**

**同步：**  
所谓同步，就是在发出一个功能调用时，在没有得到结果之前，该调用就不返回。也就是必须一件一件事做,等前一件做完了才能做下一件事。

**异步：**  
异步的概念和同步相对。当一个异步过程调用发出后，调用者不能立刻得到结果。实际处理这个调用的部件在完成后，通过状态、通知和回调来通知调用者。

**阻塞：**  
阻塞调用是指调用结果返回之前，当前线程会被挂起（线程进入非可执行状态，在这个状态下，cpu不会给线程分配时间片，即线程暂停运行）。函数只有在得到结果之后才会返回。

**非阻塞：**  
非阻塞和阻塞的概念相对应，指在不能立刻得到结果之前，该函数不会阻塞当前线程，而会立刻返回。