**RabbitMQ**

# 消息中间件的定义

其实并没有标准定义。一般认为，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。

消息中间件就是消息的搬运工，接收生产者的消息，将消息发送给消费者。

# 为什么要使用消息中间件？

* 解耦
* 异步处理
* 缓冲能力
* 伸缩性:可以不断添加MQ来处理更多的请求。
* 扩展性

# 与RPC的区别

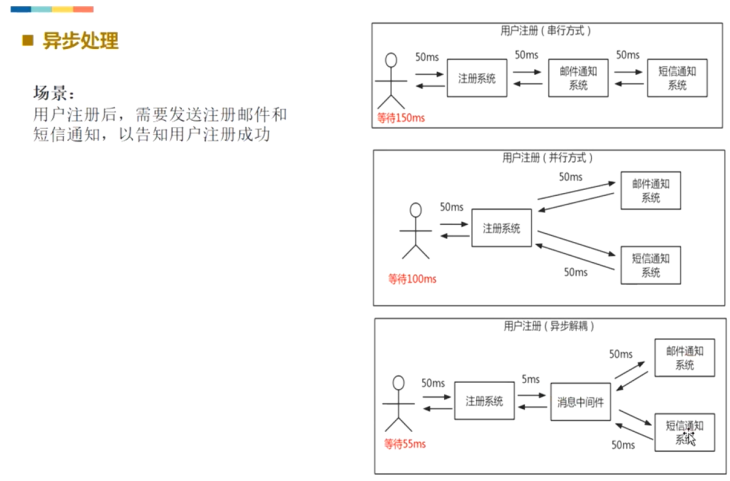
消息中间件是异步的，RPC一般是同步的。

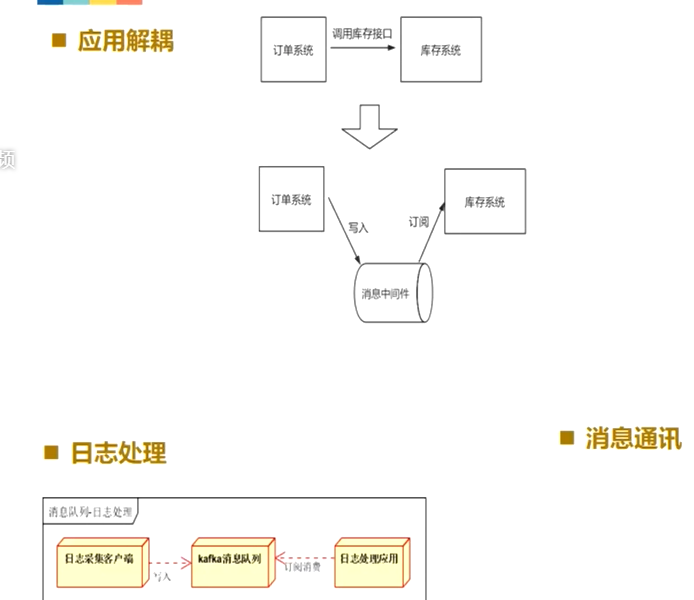
消息中间件是一个松耦合的架构。

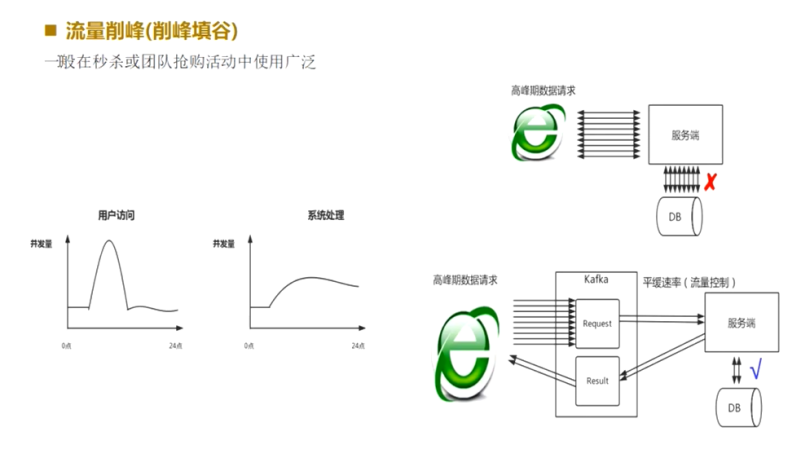
RPC实现服务的调用，调用者需要知道被调用者的接口，需要依赖接口，消息中间件中生产者完全不需要知道消费者的存在。

MQ 是生产者消费者模式。  
RPC 是请求响应模式。  
MQ 是面向数据的。  
RPC 是面向动作的。

# 消息中间件使用场景







①. 跨系统的异步通信，所有需要异步交互的地方都可以使用消息队列。就像我们除了打电话（同步）以外，还需要发短信，发电子邮件（异步）的通讯方式。

②. 多个应用之间的耦合，由于消息是平台无关和语言无关的，而且语义上也不再是函数调用，因此更适合作为多个应用之间的松耦合的接口。基于消息队列的耦合，不需要发送方和接收方同时在线。在企业应用集成（EAI）中，文件传输，共享数据库，消息队列，远程过程调用都可以作为集成的方法。

③. 应用内的同步变异步，比如订单处理，就可以由前端应用将订单信息放到队列，后端应用从队列里依次获得消息处理，高峰时的大量订单可以积压在队列里慢慢处理掉。由于同步通常意味着阻塞，而大量线程的阻塞会降低计算机的性能。

④. 消息驱动的架构（EDA），系统分解为消息队列，和消息制造者和消息消费者，一个处理流程可以根据需要拆成多个阶段（Stage），阶段之间用队列连接起来，前一个阶段处理的结果放入队列，后一个阶段从队列中获取消息继续处理。

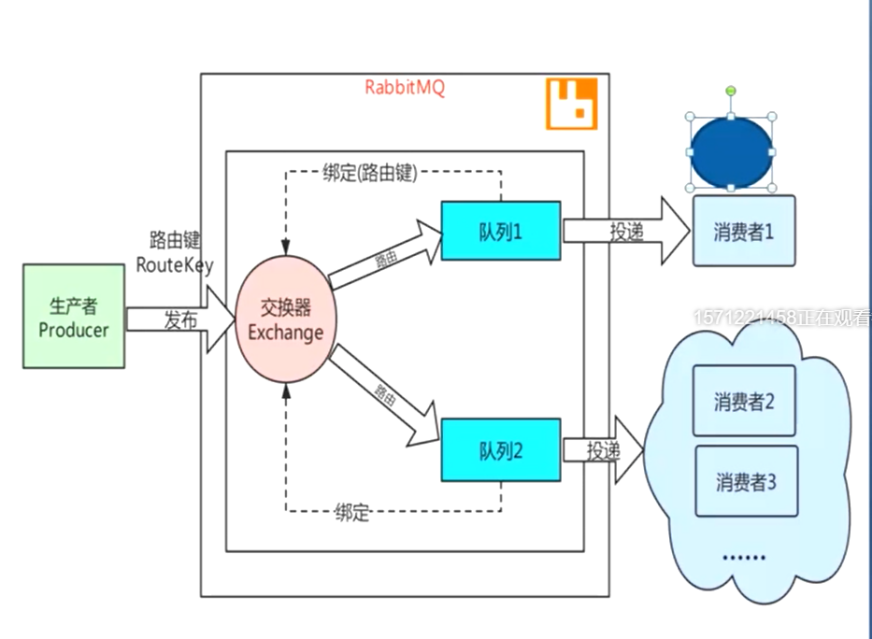
⑤. 应用需要更灵活的耦合方式，如发布订阅，比如可以指定路由规则。

⑥. 跨局域网，甚至跨城市的通讯（CDN行业），比如北京机房与广州机房的应用程序的通信。

# 如何选择消息中间件



# 内部构造



# 交换器的种类

## Direct exchange（直连交换机）

键必须完全匹配。

## Fanout exchange（扇型交换机）

和路由键没有任何关系。

## Topic exchange（主题交换机）

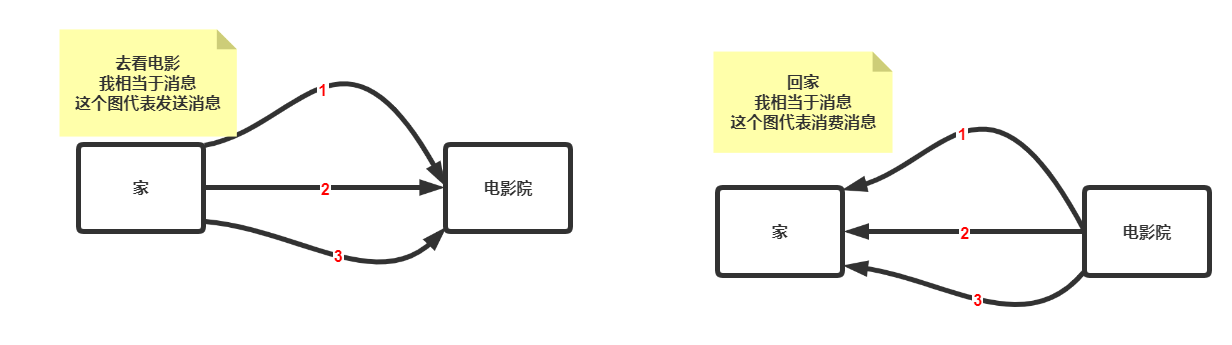
\*匹配一个 cart.info cart.# cart.\*

#可以匹配多个 cart.info.xxx cart.#可以匹配，cart.\*不可以匹配

## Headers exchange（头交换机）

# 路由键

我的观点：路由键类似于我从家到电影院看电影，可以有多条路，我可以选择任何一条路去电影院，同样的，从交换机到队列也可以有多种发送途径，可以选择任意一条途径，接收消息也可以只选择从某条路径发送过来的。



# 消息发送时的权衡

失败重试

## 发送方确认

一般确认、批量确认。

启用发送者确认模式

|  |
| --- |
| channel.confirmSelect(); |

批量确认：

|  |
| --- |
| channel.waitForConfirmsOrDie(); |

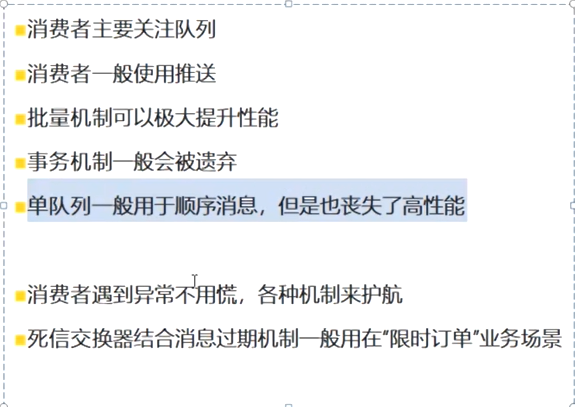
添加消息确认监听器：

|  |
| --- |
| *// 添加消息确认监听器* channel.addConfirmListener(**new** ConfirmListener() {  **public void** handleAck(**long** deliveryTag, **boolean** multiple) **throws** IOException {  System.***out***.println(**"ack message: deliveryTag = "** + deliveryTag + **", multiple = "** + multiple);  }   **public void** handleNack(**long** deliveryTag, **boolean** multiple) **throws** IOException {  System.***out***.println(**"nack message: deliveryTag = "** + deliveryTag + **", multiple = "** + multiple);  } }); |

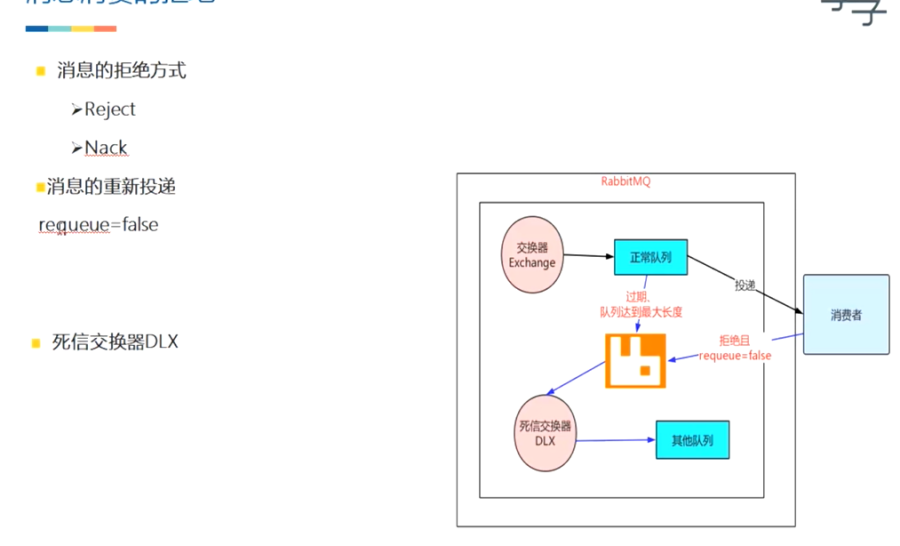
### 备用交换器



## 消费者消费时的权衡



# 消息消费的拒绝



# 队列

## 临时队列

自动删除的判断条件：消费者断开

消费者独占：

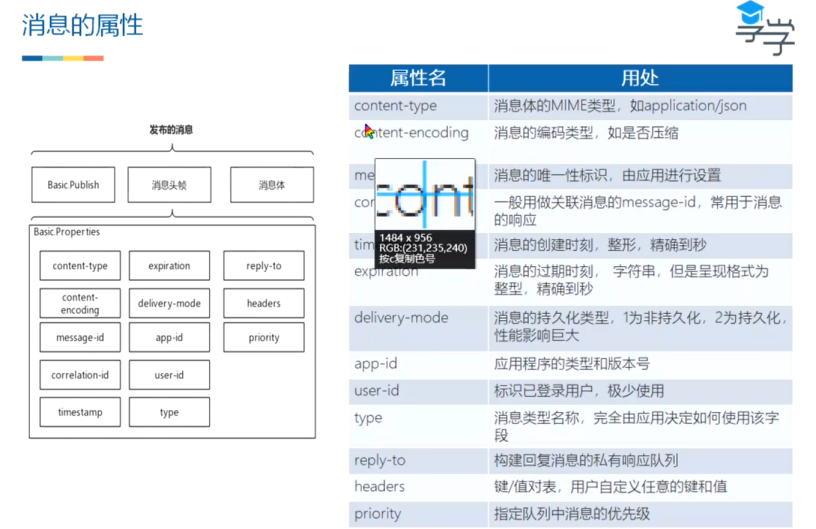


过期自动删除：



## 持久队列

# 消息的属性



# ****rabbitmq 有哪些重要的角色？****

RabbitMQ 中重要的角色有：生产者、消费者和代理：

生产者：消息的创建者，负责创建和推送数据到消息服务器；

消费者：消息的接收方，用于处理数据和确认消息；

代理：就是 RabbitMQ 本身，用于扮演“快递”的角色，本身不生产消息，只是扮演“快递”的角色。

# ****rabbitmq 有哪些重要的组件？****

ConnectionFactory（连接管理器）：应用程序与Rabbit之间建立连接的管理器，程序代码中使用。

Channel（信道）：消息推送使用的通道。

Exchange（交换器）：用于接受、分配消息。

Queue（队列）：用于存储生产者的消息。

RoutingKey（路由键）：用于把生成者的数据分配到交换器上。

BindingKey（绑定键）：用于把交换器的消息绑定到队列上。

# ****rabbitmq 中 vhost 的作用是什么？****

vhost 可以理解为虚拟 broker ，即 mini-RabbitMQ server。其内部均含有独立的 queue、exchange 和 binding 等，但最最重要的是，其拥有独立的权限系统，可以做到 vhost 范围的用户控制。当然，从 RabbitMQ 的全局角度，vhost 可以作为不同权限隔离的手段（一个典型的例子就是不同的应用可以跑在不同的 vhost 中）。

# ****rabbitmq 的消息是怎么发送的？****

首先客户端必须连接到 RabbitMQ 服务器才能发布和消费消息，客户端和 rabbit server 之间会创建一个 tcp 连接，一旦 tcp 打开并通过了认证（认证就是你发送给 rabbit 服务器的用户名和密码），你的客户端和 RabbitMQ 就创建了一条 amqp 信道（channel），信道是创建在“真实” tcp 上的虚拟连接，amqp 命令都是通过信道发送出去的，每个信道都会有一个唯一的 id，不论是发布消息，订阅队列都是通过这个信道完成的。

# ****rabbitmq 怎么保证消息的稳定性？****

提供了事务的功能。

通过将 channel 设置为 confirm（确认）模式。

# ****rabbitmq 怎么避免消息丢失？****

消息持久化

ACK确认机制

设置集群镜像模式

消息补偿机制

# ****要保证消息持久化成功的条件有哪些？****

声明队列必须设置持久化 durable 设置为 true.

消息推送投递模式必须设置持久化，deliveryMode 设置为 2（持久）。

消息已经到达持久化交换器。

消息已经到达持久化队列。

# ****rabbitmq 持久化有什么缺点？****

持久化的缺地就是降低了服务器的吞吐量，因为使用的是磁盘而非内存存储，从而降低了吞吐量。可尽量使用 ssd 硬盘来缓解吞吐量的问题。

# ****rabbitmq 有几种广播类型？****

三种广播模式：

fanout: 所有bind到此exchange的queue都可以接收消息（纯广播，绑定到RabbitMQ的接受者都能收到消息）；

direct: 通过routingKey和exchange决定的那个唯一的queue可以接收消息；

topic:所有符合routingKey(此时可以是一个表达式)的routingKey所bind的queue可以接收消息；

# ****rabbitmq 怎么实现延迟消息队列？****

通过消息过期后进入死信交换器，再由交换器转发到延迟消费队列，实现延迟功能；

使用 RabbitMQ-delayed-message-exchange 插件实现延迟功能。

# ****rabbitmq 集群有什么用？****

集群主要有以下两个用途：

高可用：某个服务器出现问题，整个 RabbitMQ 还可以继续使用；

高容量：集群可以承载更多的消息量。

# ****rabbitmq 节点的类型有哪些？****

磁盘节点：消息会存储到磁盘。

内存节点：消息都存储在内存中，重启服务器消息丢失，性能高于磁盘类型。

# ****rabbitmq 集群搭建需要注意哪些问题？****

各节点之间使用“--link”连接，此属性不能忽略。

各节点使用的 erlang cookie 值必须相同，此值相当于“秘钥”的功能，用于各节点的认证。

整个集群中必须包含一个磁盘节点。

# ****rabbitmq 每个节点是其他节点的完整拷贝吗？为什么？****

不是，原因有以下两个：

存储空间的考虑：如果每个节点都拥有所有队列的完全拷贝，这样新增节点不但没有新增存储空间，反而增加了更多的冗余数据；

性能的考虑：如果每条消息都需要完整拷贝到每一个集群节点，那新增节点并没有提升处理消息的能力，最多是保持和单节点相同的性能甚至是更糟。

# ****rabbitmq 集群中唯一一个磁盘节点崩溃了会发生什么情况？****

如果唯一磁盘的磁盘节点崩溃了，不能进行以下操作：

不能创建队列

不能创建交换器

不能创建绑定

不能添加用户

不能更改权限

不能添加和删除集群节点

唯一磁盘节点崩溃了，集群是可以保持运行的，但你不能更改任何东西。

# ****rabbitmq 对集群节点停止顺序有要求吗？****

RabbitMQ 对集群的停止的顺序是有要求的，应该先关闭内存节点，最后再关闭磁盘节点。如果顺序恰好相反的话，可能会造成消息的丢失。