## 1、RabbitMQ是什么？

RabbitMQ是实现了高级消息队列协议（AMQP）的开源消息代理软件（亦称面向消息的中间件）。RabbitMQ服务器是用Erlang语言编写的，而群集和故障转移是构建在开放电信平台框架上的。所有主要的编程语言均有与代理接口通讯的客户端库。

PS:也可能直接问什么是消息队列？消息队列就是一个使用队列来通信的组件。

## 2、RabbitMQ特点？

（1）可靠性:

RabbitMQ使用一些机制来保证可靠性，如持久化、传输确认及发布确认等。

（2）灵活的路由:

在消息进入队列之前，通过交换器来路由消息。对于典型的路由功能，RabbitMQ己经提供了一些内置的交换器来实现。针对更复杂的路由功能，可以将多个交换器绑定在一起，也可以通过插件机制来实现自己的交换器。

（3）扩展性:

多个RabbitMQ节点可以组成一个集群，也可以根据实际业务情况动态地扩展集群中节点。

（4）高可用性:

队列可以在集群中的机器上设置镜像，使得在部分节点出现问题的情况下队列仍然可用。

（5）多种协议:

RabbitMQ除了原生支持AMQP协议，还支持STOMP，MQTT等多种消息中间件协议。

（6）多语言客户端:

RabbitMQ几乎支持所有常用语言，比如Java、Python、Ruby、PHP、C#、JavaScript等。

（7）管理界面:

RabbitMQ提供了一个易用的用户界面，使得用户可以监控和管理消息、集群中的节点等。

（8）插件机制:

RabbitMQ提供了许多插件，以实现从多方面进行扩展，当然也可以编写自己的插件。

## 3、AMQP是什么？

RabbitMQ就是AMQP协议的Erlang的实现(当然RabbitMQ还支持STOMP2、MQTT3等协议)

AMQP的模型架构和RabbitMQ的模型架构是一样的，生产者将消息发送给交换器，交换器和队列绑定。

RabbitMQ中的交换器、交换器类型、队列、绑定、路由键等都是遵循的AMQP协议中相应的概念。目前RabbitMQ最新版本默认支持的是AMQP0-9-1。

## 4、AMQP协议3层？

（1）ModuleLayer:

协议最高层，主要定义了一些客户端调用的命令，客户端可以用这些命令实现自己的业务逻辑。

（2）SessionLayer:

中间层，主要负责客户端命令发送给服务器，再将服务端应答返回客户端，提供可靠性同步机制和错误处理。

（3）TransportLayer:

最底层，主要传输二进制数据流，提供帧的处理、信道服用、错误检测和数据表示等。

## 5、AMQP模型的几大组件？

交换器(Exchange)：消息代理服务器中用于把消息路由到队列的组件。

队列(Queue)：用来存储消息的数据结构，位于硬盘或内存中。

绑定(Binding)：一套规则，告知交换器消息应该将消息投递给哪个队列。

## 6、说说生产者Producer和消费者Consumer?

（1）生产者

消息生产者，就是投递消息的一方。

消息一般包含两个部分：消息体（payload)和标签(Label)。

（2）消费者

消费消息，也就是接收消息的一方。

消费者连接到RabbitMQ服务器，并订阅到队列上。消费消息时只消费消息体，丢弃标签。

## 7、为什么需要消息队列？

从本质上来说是因为互联网的快速发展，业务不断扩张，促使技术架构需要不断的演进。

从以前的单体架构到现在的微服务架构，成百上千的服务之间相互调用和依赖。从互联网初期一个服务器上有100个在线用户已经很了不得，到现在坐拥10亿日活的微信。此时，我们需要有一个「工具」来解耦服务之间的关系、控制资源合理合时的使用以及缓冲流量洪峰等等。因此，消息队列就应运而生了。

它常用来实现：**异步处理、服务解耦、流量控制（削峰）**。

## 8、说说Broker服务节点、Queue队列、Exchange交换器？

（1）Broker

可以看做RabbitMQ的服务节点。一般请下一个Broker可以看做一个RabbitMQ服务器。

（2）Queue:

RabbitMQ的内部对象，用于存储消息。多个消费者可以订阅同一队列，这时队列中的消息会被平摊（轮询）给多个消费者进行处理。

（3）Exchange:

生产者将消息发送到交换器，由交换器将消息路由到一个或者多个队列中。当路由不到时，或返回给生产者或直接丢弃。

## 9、消息队列有什么优缺点

优点上面已经说了，就是在特殊场景下有其对应的好处，解耦、异步、削峰。

缺点有以下几个：

系统可用性降低系统引入的外部依赖越多，越容易挂掉。万一MQ挂了，MQ一挂，整套系统崩溃，你不就完了？

系统复杂度提高硬生生加个MQ进来，你怎么保证消息没有重复消费？怎么处理消息丢失的情况？

怎么保证消息传递的顺序性？问题一大堆。

一致性问题A系统处理完了直接返回成功了，人都以为你这个请求就成功了；但是问题是，要是BCD三个系统那里，BD两个系统写库成功了，结果C系统写库失败了，咋整？你这数据就不一致了。

## 10、如何保证消息的可靠性？

消息到MQ的过程中搞丢，MQ自己搞丢，MQ到消费过程中搞丢。

（1）生产者到RabbitMQ：事务机制和Confirm机制，注意：事务机制和Confirm机制是互斥的，两者不能共存，会导致RabbitMQ报错。

（2）RabbitMQ自身：持久化、集群、普通模式、镜像模式。

（3）RabbitMQ到消费者：basicAck机制、死信队列、消息补偿机制。

## 11、什么是RoutingKey路由键？

生产者将消息发送给交换器的时候，会指定一个RoutingKey,用来指定这个消息的路由规则，这个RoutingKey需要与交换器类型和绑定键(BindingKey)联合使用才能最终生效。

## 12、Binding绑定？

通过绑定将交换器和队列关联起来，一般会指定一个BindingKey,这样RabbitMq就知道如何正确路由消息到队列了。

## 13、交换器4种类型？

主要有以下4种。

（1）fanout:把所有发送到该交换器的消息路由到所有与该交换器绑定的队列中。

（2）direct:把消息路由到BindingKey和RoutingKey完全匹配的队列中。

（3）topic:

匹配规则：

RoutingKey`为一个点号'.':分隔的字符串。比如:`java.xiaoka.show

BindingKey和RoutingKey一样也是点号“.“分隔的字符串。

BindingKey可使用\*和#用于做模糊匹配，\*匹配一个单词，#匹配多个或者0个

（4）headers:不依赖路由键匹配规则路由消息。是根据发送消息内容中的headers属性进行匹配。性能差，基本用不到。

## 14、生产者消息运转？

（1）Producer先连接到Broker,建立连接Connection,开启一个信道(Channel)。

（2）Producer声明一个交换器并设置好相关属性。

（3）Producer声明一个队列并设置好相关属性。

（4）Producer通过路由键将交换器和队列绑定起来。

（5）Producer发送消息到Broker,其中包含路由键、交换器等信息。

（6）相应的交换器根据接收到的路由键查找匹配的队列。

（7）如果找到，将消息存入对应的队列，如果没有找到，会根据生产者的配置丢弃或者退回给生产者。

（8）关闭信道。

（9）管理连接。

## 15、消费者接收消息过程？

（1）Producer先连接到Broker,建立连接Connection,开启一个信道(Channel)。

（2）向Broker请求消费响应的队列中消息，可能会设置响应的回调函数。

（3）等待Broker回应并投递相应队列中的消息，接收消息。

（4）消费者确认收到的消息，ack。

（5）RabbitMq从队列中删除已经确定的消息。

（6）关闭信道。

（7）关闭连接。

## 16、交换器无法根据自身类型和路由键找到符合条件队列时，有哪些处理？

mandatory：true返回消息，给生产者。

mandatory:false直接丢弃。

## 17、死信队列？

DLX，全称为Dead-Letter-Exchange，死信交换器，死信邮箱。当消息在一个队列中变成死信(deadmessage)之后，它能被重新被发送到另一个交换器中，这个交换器就是DLX，绑定DLX的队列就称之为死信队列。

## 18、导致的死信的几种原因？

（1）消息被拒（Basic.Reject/Basic.Nack)且requeue=false。

（2）消息TTL过期。

（3）队列满了，无法再添加。

## 19、延迟队列？

存储对应的延迟消息，指当消息被发送以后，并不想让消费者立刻拿到消息，而是等待特定时间后，消费者才能拿到这个消息进行消费。

## 20、优先级队列？

优先级高的队列会先被消费。

可以通过x-max-priority参数来实现。

当消费速度大于生产速度且Broker没有堆积的情况下，优先级显得没有意义。

## 21、事务机制？

RabbitMQ客户端中与事务机制相关的方法有三个:

channel.txSelect用于将当前的信道设置成事务模式。

channel.txCommit用于提交事务。

channel.txRollback用于事务回滚，如果在事务提交执行之前由于RabbitMQ异常崩溃或者其他原因抛出异常，通过txRollback来回滚。

## 22、发送确认机制？

生产者把信道设置为confirm确认模式，设置后，所有再改信道发布的消息都会被指定一个唯一的ID，一旦消息被投递到所有匹配的队列之后，RabbitMQ就会发送一个确认（Basic.Ack)给生产者（包含消息的唯一ID)，这样生产者就知道消息到达对应的目的地了。

## 23、消费者获取消息的方式？

推、拉

## 24、消费者某些原因无法处理当前接受的消息如何来拒绝？

channel.basicNackchannel.basicReject

## 25、消息传输保证层级？

Atmostonce:最多一次。消息可能会丢失，但不会重复传输。

Atleastonce：最少一次。消息绝不会丢失，但可能会重复传输。

Exactlyonce:恰好一次，每条消息肯定仅传输一次。

## 26、了解VirtualHost吗？

每一个RabbitMQ服务器都能创建虚拟的消息服务器，也叫虚拟主机(virtualhost)，简称vhost。默认为“/”。

## 27、集群中的节点类型？

内存节点：ram,将变更写入内存。

磁盘节点：disc,磁盘写入操作。

RabbitMQ要求最少有一个磁盘节点。

## 28、队列结构？

通常由以下两部分组成？

（1）rabbit\_amqqueue\_process:

负责协议相关的消息处理，即接收生产者发布的消息、向消费者交付消息、处理消息的确认(包括生产端的confirm和消费端的ack)等。

（2）backing\_queue:

是消息存储的具体形式和引擎，并向rabbitamqqueueprocess提供相关的接口以供调用。

## 29、RabbitMQ中消息可能有的几种状态？

（1）alpha:消息内容(包括消息体、属性和headers)和消息索引都存储在内存中。

（2）beta:消息内容保存在磁盘中，消息索引保存在内存中。

（3）gamma:消息内容保存在磁盘中，消息索引在磁盘和内存中都有。

（4）delta:消息内容和索引都在磁盘中。

## 30、在何种场景下使用了消息中间件？

接口之间耦合比较严重

面对大流量并发时，容易被冲垮

存在性能问题

## 31、生产者如何将消息可靠投递到MQ？

（1）Client发送消息给MQ；

（2）MQ将消息持久化后，发送Ack消息给Client，此处有可能因为网络问题导致Ack消息无法发送到Client，那么Client在等待超时后，会重传消息；

（3）Client收到Ack消息后，认为消息已经投递成功。

## 32、MQ如何将消息可靠投递到消费者？

（1）MQ将消息push给Client（或Client来pull消息）

（2）Client得到消息并做完业务逻辑

（3）Client发送Ack消息给MQ，通知MQ删除该消息，此处有可能因为网络问题导致Ack失败，那么Client会重复消息，这里就引出消费幂等的问题；

（4）MQ将已消费的消息删除

## 33、如何保证RabbitMQ消息队列的高可用？

RabbitMQ有三种模式：单机模式，普通集群模式，镜像集群模式。

（1）单机模式：

就是demo级别的，一般就是你本地启动了玩玩儿的，没人生产用单机模式。

（2）普通集群模式：

意思就是在多台机器上启动多个RabbitMQ实例，每个机器启动一个。

（3）镜像集群模式：

这种模式，才是所谓的RabbitMQ的高可用模式，跟普通集群模式不一样的是，你创建的queue，无论元数据(元数据指RabbitMQ的配置数据)还是queue里的消息都会存在于多个实例上，然后每次你写消息到queue的时候，都会自动把消息到多个实例的queue里进行消息同步。

34、RabbitMQ使用场景

（1）顺序消息

（2）定时任务

（3）请求削峰

（4）异步通信

RabbitMQ基本概念

Broker： 消息队列服务器实体

Exchange： 消息交换机，它指定消息按特定规则，路由到哪个队列

Queue： 消息队列载体，每个消息都会被投入到一个或多个队列

Binding： 绑定，它的作用就是把exchange和queue按照路由规则绑定起来

Routing Key： 路由关键字，exchange根据这个关键字进行消息投递

VHost： 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，拥有独立的权限系统，可以做到 vhost 范围的用户控制。

Producer： 消息生产者，

Consumer： 消息消费者

Channel： 消息通道，在客户端的每个连接里，可建立多个channel，每个channel代表一个会话任务

工作模式

简单模式

工作模式

发布订阅模式

路由模式

主题模式

发布确认模式

## 消息基于什么传输？

RabbitMQ 使用信道的方式来传输数据。(由于 TCP 连接的创建和销毁开销较大，且并发数受系统资源限制，会造成性能瓶颈。)信道是建立在真实的 TCP 连接内的虚拟连接，且每条 TCP 连接上的信道数量没有限制。

如何保证RabbitMQ消息的可靠传输？

消息不可靠的情况可能是消息丢失，劫持等原因

丢失又分为：

生产者丢失消息、消息列表丢失消息、消费者丢失消息；

① 生产者丢失消息：从生产者弄丢数据这个角度来看，

RabbitMQ提供transaction机制和confirm模式来确保生产者不丢消息；

transaction机制就是说：发送消息前，开启事务（channel.txSelect()）,然后发送消息，如果发送过程中出现什么异常，事务就会回滚（channel.txRollback()）,如果发送成功则提交事务（channel.txCommit()）。

confirm模式用的居多：一旦channel进入confirm模式，所有在该信道上发布的消息都将会被指派一个唯一的ID（从1开始），一旦消息被投递到所有匹配的队列之后； RabbitMQ就会发送一个ACK给生产者（包含消息的唯一ID），这就使得生产者知道消息已经正确到达目的队列了；

如果RabbitMQ没能处理该消息，则会发送一个Nack消息给你，你可以进行重试操作。

②消息队列丢数据：消息持久化。

处理消息队列丢数据的情况，一般是开启持久化磁盘的配置

将队列的持久化标识durable设置为true,则代表是一个持久的队列 发送消息的时候将 deliveryMode=2 这样设置以后，即使RabbitMQ挂了，重启后也能恢复数据

③消费者丢失消息：消费者丢数据一般是因为采用了自动确认消息模式，改为手动确认消息

消费者在收到消息之后，处理消息之前，会自动回复RabbitMQ已收到消息； 如果这时处理消息失败，就会丢失该消息；

解决方案：处理消息成功后，手动回复确认消息。

如何确保消息正确地发送至 RabbitMQ？ 如何确保消息接收方消费了消息？

发送方确认模式

将信道设置成 confirm 模式（发送方确认模式），则所有在信道上发布的消息都会被指派一个唯一的 ID。

一旦消息被投递到目的队列后，或者消息被写入磁盘后（可持久化的消息），信道会发送一个确认给生产者（包含消息唯一 ID）。 如果 RabbitMQ 发生内部错误从而导致消息丢失，会发送一条 nack（notacknowledged，未确认）消息。

发送方确认模式是异步的，生产者应用程序在等待确认的同时，可以继续发送消息。当确认消息到达生产者应用程序，生产者应用程序的回调方法就会被触发来处理确认消息。

接收方确认机制

消费者接收每一条消息后都必须进行确认（消息接收和消息确认是两个不同操作）。只有消费者确认了消息，RabbitMQ 才能安全地把消息从队列中删除。 这里并没有用到超时机制，RabbitMQ 仅通过 Consumer 的连接中断来确认是否需要重新发送消息。也就是说，只要连接不中断，RabbitMQ 给了 Consumer 足够长的时间来处理消息。保证数据的最终一致性。

### 5. 如何避免消息的重复消费和重复投递

在消息生产时，MQ内部针对每条生产者发送的消息生成一个inner-msg-id，作为去重和幂等的依据（消息投递失败并重传），避免重复的消息进入队列；在消息消费时，要求消息体中必须要有一个bizId（对于同一业务全局唯一，如支付ID、订单ID、帖子ID等）作为去重和幂等的依据，避免同一条消息被重复消费

消息幂等性

我对一个动作进行操作，我们肯能要执行100次1000次，对于这1000次执行的结果都必须一样的。比如单线程方式下执行update count-1的操作执行一千次结果都是一样的，所以这个更新操作就是一个幂等的，如果是在并发不做线程安全的处理的情况下update一千次操作结果可能就不是一样的，所以并发情况下的update操作就不是一个幂等的操作。对应到消息队列上来，就是我们即使受到了多条一样的消息，也和消费一条消息效果是一样的。

高并发的情况下如何避免消息重复消费

唯一id+加指纹码，利用数据库主键去重。

优点：实现简单

缺点：高并发下有数据写入瓶颈。

利用Redis的原子性来实习。

使用Redis进行幂等是需要考虑的问题

是否进行数据库落库，落库后数据和缓存如何做到保证幂等（Redis   和数据库如何同时成功同时失败）？

如果不进行落库，都放在Redis中如何这是Redis和数据库的同步策略？还有放在缓存中就能百分之百的成功吗？

confirm 确认消息、Return返回消息

理解confirm消息确认机制

消息的确认，指生产者收到投递消息后，如果Broker收到消息就会给我们  的生产者一个应答，生产者接受应答来确认broker是否收到消息。

如何实现confirm确认消息。

在Channel上开启确认模式：channel.confirmSelect()

在channel上添加监听：addConfirmListener，监听成功和失败的结果，具体结果对消息进行重新发送或者记录日志。

return消息机制

Return消息机制处理一些不可路由的消息，我们的生产者通过指定一个Exchange和Routinkey，把消息送达到某一个队列中去，然后我们消费者监听队列进行消费处理！

在某些情况下，如果我们在发送消息的时候当Exchange不存在或者指定的路由key路由找不到，这个时候如果我们需要监听这种不可到达的消息，就要使用Return Listener！

Mandatory 设置为true则会监听器会接受到路由不可达的消息，然后处理。如果设置为false，broker将会自动删除该消息。

-----------------------------------

RabbitMQ面试总结二

https://blog.51cto.com/u\_15257216/2862247

<https://blog.51cto.com/u_15257216/2862247>