消息队列

消息队列message queuing 使用消息将应用程序连接起来，这些消息通过想RabbitMQ这样的消息代理服务器在应用程序之间路由。

1990年，开始做。1993年，消息队列服务器软件IBM MQ产品系列面世。

1997年，微软也在消息通信市场崭露头角:微软消息队列。

2004年：AMQP（advanced Message Queuing Protocol，高级消息队列协议）出现。从一开始就设计成为开发标准，已解决众多的消息队列需求和拓扑结构问题。实现了从任何发布者到任何感兴趣的消费者之间的信息，通过一条总线实时动态连接起来。

RabbitMQ简史：

略。

在centos上安装RabbitMQ

Centos版本：centos7

erlang：版本20.1 ：http://erlang.org/download/

Rabbit：3.6.4 ：http://www.rabbitmq.com/install-rpm.html

以上版本在官网：

安装步骤：

1. 首先安装依赖环境

|  |
| --- |
| **yum -y install make gcc gcc-c++ kernel-devel m4 ncurses-devel openssl-devel unixODBC-devel** |

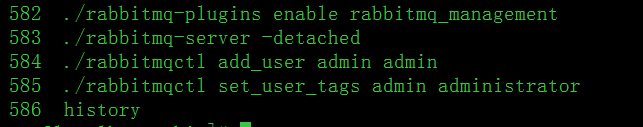
1. 安装最新版erlang：

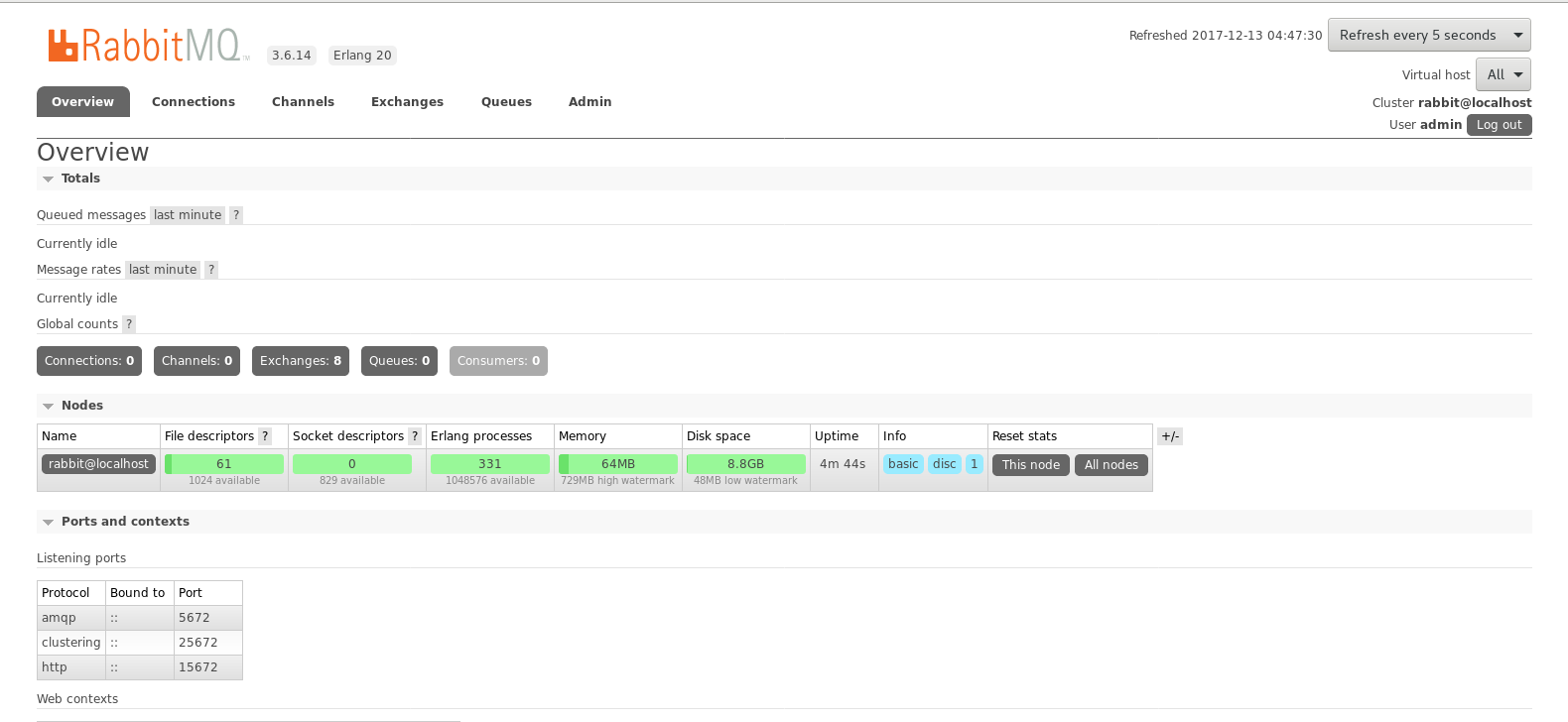
|  |
| --- |
| **wget http://erlang.org/download/otp\_src\_20.1.tar.gz** |
| **tar -xvzf otp\_src\_20.1.tar.gz otp\_src\_20.1** |
| **cd otp\_src\_20.1/** |
| **./configure --prefix=/usr/local/erlang --with-ssl -enable-threads -enable-smmp-support -enable-kernel-poll --enable-hipe --without-javac** |
| **make** |
| **make install** |
| **vi /etc/profile** |
| **source /etc/profile** |
| **erl** |



3. 安装rabbitMQ

|  |
| --- |
| **wget http://www.rabbitmq.com/releases/rabbitmq-server/v3.6.14/rabbitmq-server-generic-unix-3.6.14.tar.xz** |
| **xz -d rabbitmq-server-generic-unix-3.6.14.tar.xz** |
| **tar -xvf rabbitmq-server-generic-unix-3.6.14.tar** |
| **./rabbitmq-plugins enable rabbitmq\_management # 开启界面管理** |
| **./rabbitmq-server –detached # 启动服务** |
| **./rabbitmqctl add\_user admin admin #添加用户** |
| **./rabbitmqctl set\_user\_tags admin administrator # 给用户赋予权限** |
|  |





在window上安装RabbitMQ 非常简单。

下载对应的.exe即可。

2 理解消息通信

要点：

消息通信概念(消费者，生产者和代理)。

AMQP元素(交换器，队列和绑定)

虚拟主机

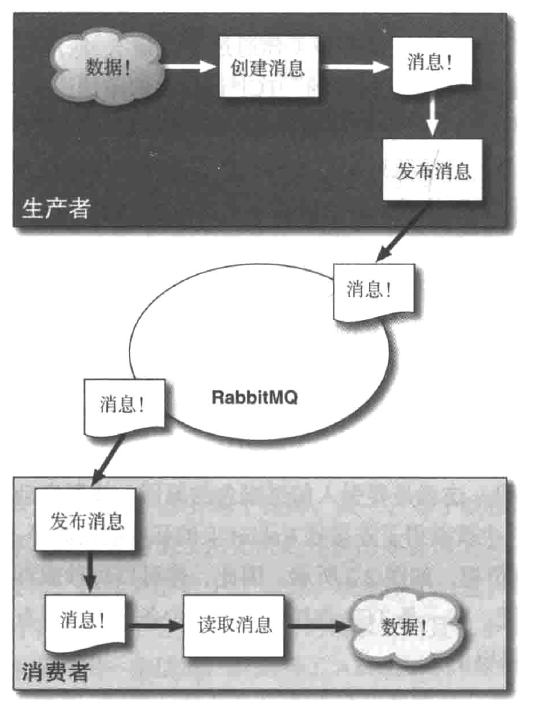
消息持久化

一条消息历经从生产者到消费者的生命周期

2.1消费者和生产者：

把RabbitMQ当做一种投递服务，应用程序可以发送和接口包裹。而数据所在的服务器也可以发送或者接收。RabbitMQ在应用程序和服务器扮演这路由器的角色。所以当应用程序连接到RabbitMQ时，它必须要做一个决定，我是在发送还是接收呢?或者从AMQP的角度思考，我是一个生产者还是一个消费者呢。

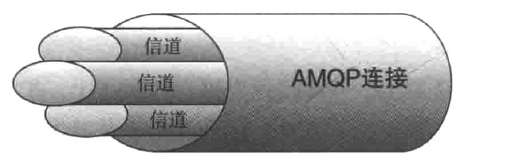
生产者（producer）创建消息，然后发布（发送）到代理服务器（RabbitMQ）。什么是消息呢？消息包含两个部分，有效载荷（payload）和标签（label）。有效载荷就是你想要传输的数据。可以是任何内容，json，，，等等、标签描述了有效载荷，并且RabbitMQ用它来决定谁将获得消息的拷贝。 -------生产者会创建消息并设置标签。



消费者连接到代理服务器上，并订阅到队列上。每当消息到达特定的队列上，RabbitMQ会将其发送给其中一个订阅的/监听的消费者。当消费者接收到消息时，他只得到消息的一部分：有效负载。如果想要明确知道是谁生产的AMQP消息的话，就要看生产者是否把发送发信息放入有效载荷中。

我们自己的应用程序是如何连接的RabbatMQ服务器的呢？

使用TCP？使用TCP对操作系统来说，代价是非常昂贵的。而且操作系统每秒也就能创建上千条连接，这就到了系统的瓶颈了。RabbitMQ使用了信道。线程启动后，会在现成的连接上创建一条信道，也就获得了连接到Rabbit上的私密通信路径。而不会给操作系统的TCP栈造成额外负担。在一条TCP连接上创建多少信道是没有限制的。如下图：



类似与光纤：TCP就像是电缆，而AMQP信道就像是一条条独立光纤束。

一旦建立到rabbitMQ代理服务器的连接，应用程序将创建多条信道：chan\_recv信道用于服务接收消息的线程。chan\_sendX（X是线程号）信道用于服务每一个应答线程。

消费者和生产者是消息发送和消息接收概念的体现。而不是客户端和服务器端。从总体上说，AMQP可以被当做是加强版的传输层。

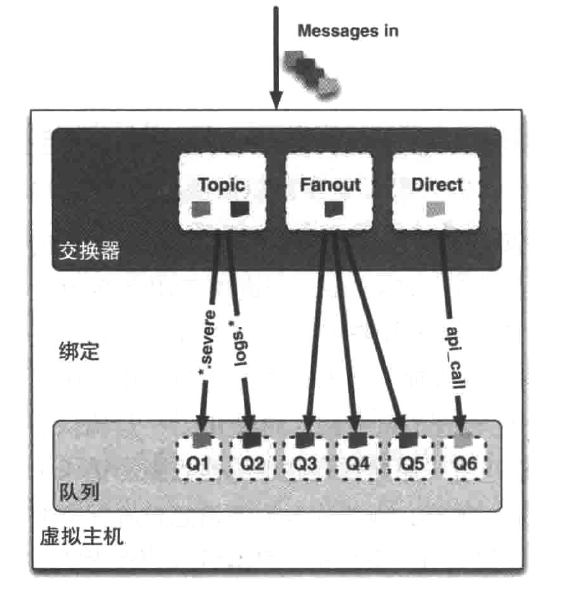
2017.12.12 21:42 畅游大厦

2.2 从底层开始构造 - 队列

从概念上来讲，AMQP消息路由必须有三部分：交换机，队列和绑定。

生产者把消息发布到交换机上；消息最终到达队列，并被消费者接收。绑定决定了消息如何从路由器到特定的队列。

首先先看一下队列的概念和工作原理：



消费者可以通过两种方式从特定的队列中接收消息。

1. 通过AMQP的basic.consume命令订阅。这样会将信道置为接收模式。直到取消对队列的订阅为止。订阅了消息之后，消费者在消费（或者拒绝）最近接收的那条消息后，就能从队列中自动接收下一条消息。如果消费者处理队列消息，并且/或者需要在消息已到达队列时就自动接收的话，就应该使用consume。
2. 向队列请求单条消息是通过AMQP的basic.get命令实现的。这样做可以让消费者接收队列中的下一条消息，然后就取消订阅。

如果至少有一个消费者订阅了队列的话，消息会立即发送给这些订阅的消费者。但是如果消息到达了无人订阅的队列呢？在这种情况下，消息会在队列中等待。一旦有消费者订阅到该队列，那么队列上的消息就会发送到消费者。

那么当有多个消费者订阅到同一个队列上的时候，消息是如何分发的呢？

队列收到的消息将以循环的方式发送给消费者，并且每条消息只会发送给一个订阅的消费者。

消费者接收到每一条消息都必须进行确认。消费者可以通过AMQP的basic.ack命令显示的向RabbitMQ发送一个确认。或者订阅到队列的时候，就将auto\_ack参数设置为true。消费者通过确认命令告诉RabbitMQ它已经正确接收了消息，通过RabbitMQ才能把消息从队列中删除。

如果消费者接收到一条消息，但是没有确认，就从rabbitMQ断开了或者从队列上取消了订阅，RabbitMQ会认为这条消息没有分发，然后重新分发给下一个订阅的消费者。

在收到消息后，如果想要明确拒绝而不是确认收到消息的话，该如何呢？

在处理消息的时候，遇到了不可恢复的错误，但是由于硬件问题，只影响到当前的消费者。只要消息尚未确认，有以下两种发誓选择：

1. 把消费者从RabbitMQ服务器上断开连接。
2. 如果是>2.20.0 的版本可以 使用AMQP的basic.reject命令。
   1. 如果把reject命令的requeue参数设置为true的时候，RabbitMQ会将消息重新发送到下一个订阅者。
   2. 如果是false，RabbitMQ立即会把消息从消息队列中移除。不发送给新的消费者。

**还有一个比较重要的问题：如何创建队列**

消费者和生产者都能使用AMQP的queue.declare命令来创建队列。但是，如果消费者在同一条信道上订阅了另一个队列的话，就无法再声明队列了。必须首先取消订阅，将信道置为“传输模式”。

创建队列的时候可以使用的参数：

1. exclusive -----如果为true的时候，队列可以变成私有的，此时只有你的应用程序才能够消费队列消息。
2. auto-delete----------当最后一个消费者取消订阅的时候，队列就会自动移除。

如果要创建一个已经存在的队列，只要声明的参数完全匹配现存的队列的话，RabbitMQ就什么不做，参数不匹配就会创建不失败。

队列是AMQP消息通信的基础模块：

1.为消息提供了住所，消息在此等待消费。

2.对负载均衡来说，队列是绝佳方案，只要附加一堆消费者，并让RabbitMQ以循环的方式均匀的发配发来的消息。

3.队列是Rabbit中消息的最后的终点\*（除非消息进入了最后的黑洞）

* 1. 联合起来：交换机和绑定

当想要把消息投递到队列时，就要通过把消息发送给交换机来完成。然后，根据确定的规则，RabbitMQ将会决定消息该投递到哪个队列。这些规则被称作路由键。队列通过路由键绑定到交换机。队列通过路由键绑定到交换器. 当把消息发送到代理服务器，消息将拥有一个路由键-即便是空的-RabbitMQ也会将其和绑定的路由键进行匹配。如果相匹配的话，那么消息将投递到该队里，如果路由的消息不匹配任何绑定模式的话，消息将进入黑洞。

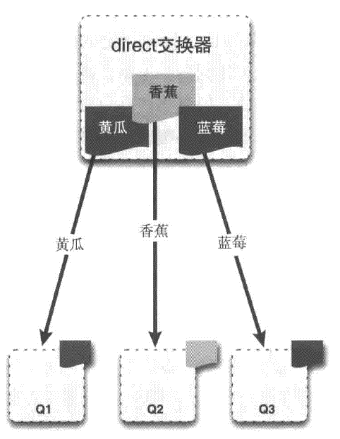
服务器会更具路由键将消息从交换器路由到队列。但是它是如何处理投递到多个队列的情况的呢？

AMQP协议中定义了四种不同类型的交换器，direct，fanout，topic，headers。每一种算法实现了不同的路由算法。

**Headers交换器**允许你匹配AMQP消息的header，而非路由键。除此之外，header交换器和direct交换器完全一致，但是性能差很多

**Direct交换器**

如果路由键匹配的话，消息就被投递到对应的队列。



服务器必须实现direct类型的交换器，包含一个空白字符串名称的默认交换器。当声明一个队列的时候，他会自动绑定到默认交换器，并以队列名称作为路由键。

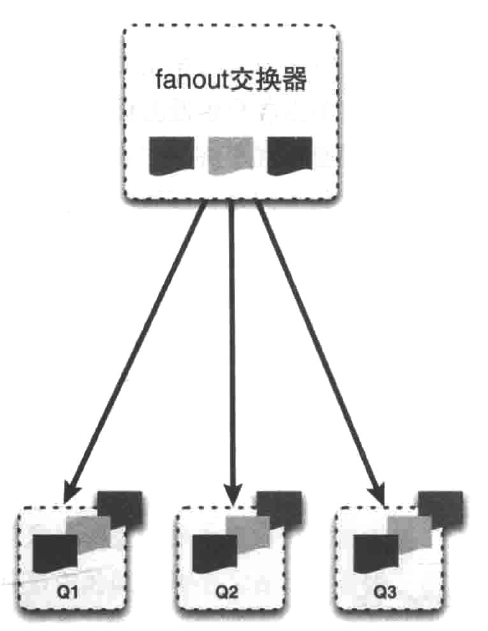
channelInstance.basicPublish(massage,’’,queue).

第一个参数是想要发送的内容，第二个参数是空字符串，指定了使用默认的交换器。第三个参数是路由键。也就是用来声明队列的名称。

**Fanout交换器：**

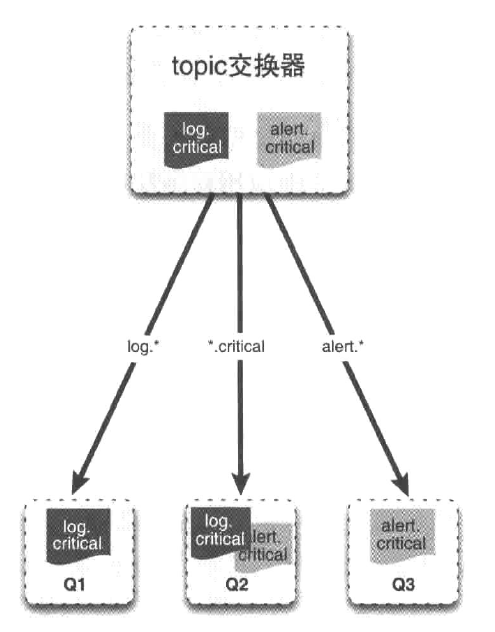
将收到的消息广播到绑定的队列上。

消息通信模式：当你发送一条消息到fanout交换器时，他会把消息投递给所有附加在此交换器上的队列。这允许你对不同的消息做不同方式的反应。



Topic 交换器：

它可以使来自不同源头的消息到达同一个队列。



Channel.basicPublish(msg,’logs-exchange’,’error.msg-inbox’); f发送消息。

Channel.queueBind(‘msg-inbox-errors’,’logs-exchange’,’error.msg-inbox’);

这样就为队列绑定操作和消息发送路由键制定了相同的error.msg-inbox字符串作为绑定规则。这样，消息就会路由到msg-inbox-errors队列。

如果想要一个队列监听msg-inbox模块的所有error级别的话，可以通过新的队列绑定到已有的同一个交换器来实现。

Channel.queueBind(‘msg-inbox-logs’,’logs-exchange’,’\*.msg-inbox’);

Msg-inbox-logs队列将会接收从msg-inbox模块发来的所有error，warning和info的日志消息。

* 1. 多租户模式---虚拟主机和隔离

每一个RabbitMQ服务器都能创建虚拟消息服务器，称之为虚拟主机vhost。Vhost是mini版的RabbitMQ服务器。拥有自己的队列，交换器和绑定。还有自己的权限机制。

Vhost是AMQP概念的基础，必须在连接的时候就进行指定。默认的vhost是：“/”.在Rabbit里创建一个而用户，用户通常会指派给至少一个vhost。并且只能访问被指派vhost内的队列，交换机和绑定。

Vhost和权限控制非常独特，他们是AMQP中唯一无法通过AMQP协议创建的基元。

可以通过rabbitmqctl命令来创建。

rabbitmqctl add\_vhost <hostName>

rabbitmqctl delete\_vhost <hostName>

rabbitmqctl list\_vhosts

2.5 持久化和你的策略

重启rabbitMQ服务器后，队列和交换器就都消失了。原因是每个队列和交换器的durable属性默认为false。它决定了RabbtMQ是否在崩溃和重启之后重新创建队列。

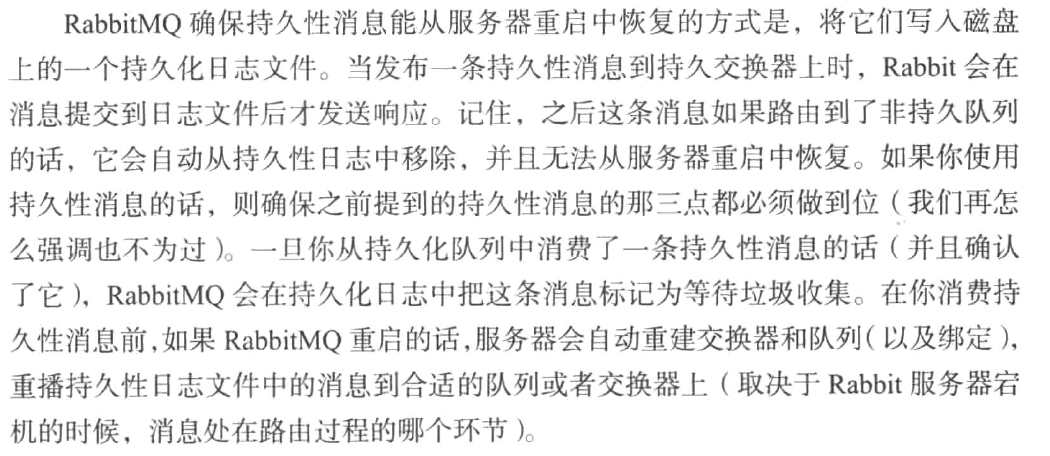
当时，是指durable=true，还是不能让消息在重启之后，幸免于难。

能从AMQP服务器崩溃中恢复的消息我们称之为持久化消息。如果消息要想从Rabbit崩溃中恢复，那么消息必须：

A.把它的投递模式选项设置为2

B.发送到持久化的交换器

C.到达持久化的队列



**AMQP事务：**

在AMQP中，在把信道设置为事务模式后，摩天平面广告IP信道发送那些想要确认的消息，之后还有多个其他AMQP命令。如果发送失败的话，其他AMQP命令将不会执行。事务降低了RabbitMQ的性能。后来Rabbit团队拿出了更好的方案保证投递。

**发送方确认模式。**

将信道设置为confirm模式，而且你只能通过重新创建信道来关闭该设置。一旦信道进入confirm模式，所有在信道上发布的消息都会被指派一个唯一的ID号，一旦消息被投递给所有匹配的队列后，信道会发送一个发送发确认模式给生产者应用程序。这使得生产者知晓消息已经到达目的队列了。

发送发模式是异步的。发布了一条消息。生产者应用程序就可以在等待确认的同时继续发送下一条。当确认消息最终收到的时候，生产者应用的回调方式就会被触发来处理该确认消息。

2.6 一条消息的生命周期。