各个消息队列框架的介绍

ActiveMQ：apache出品，能力强劲的开源消息总线，完全支持jms规范的消息中间件。api丰富，在传统行业的中小型企业中应用广泛。缺点：服务性能和数据存储性能不好。

kafka：apache顶级项目，追求高吞吐量。一开始的目的是用于日志收集和传输。不支持事务，对消息重复，丢失，错误没有严格的要求。适合产生大量数据的互联网服务的数据收集业务。

rocketMQ：阿里开源中间件，目前已经孵化为apache顶级项目，纯java开发。思路起源于kafka，对消息的可靠性传输和事务性做了优化。特点：高吞吐量，高可用，适合大规模分布式系统应用。目前在阿里集团被广泛使用，用于交易，充值，流计算，日志处理，消息推送等。

高性能，高可靠性，支持分布式，支持事务，集群水平的扩展，上亿级别消息的堆积，主从之间的自由切换等表现良好，但是商业版收费。很多功能不对外公布。

为什么选择RabbitMQ

RabbitMQ是一个开源的消息代理和队列服务器，用来通过普通协议在完全不同的应用之间共享数据，RabbitMQ是使用ErLang语言来编写的，并且是基于AMQP协议的。

1. 开源的消息中间件
2. 可以跨平台，跨语言。数据的生成和消费可以是不同的语言。

为什么使用？

> 开源，性能优秀
>
> 提供可靠性消息投递模式，返回模式
>
> 与springAMQP完美的整合
>
> 集群模式丰富，表达式配置，HA模式，镜像队列模型。

RabbitMQ的高性能之道是如何做到的

1. Erlang语言最初用于交换机领域，这样使RabbitMQ在Broker之间进行数据交互的性能是非常优秀的。
2. Erlang有着和原生Socket一样的延迟。

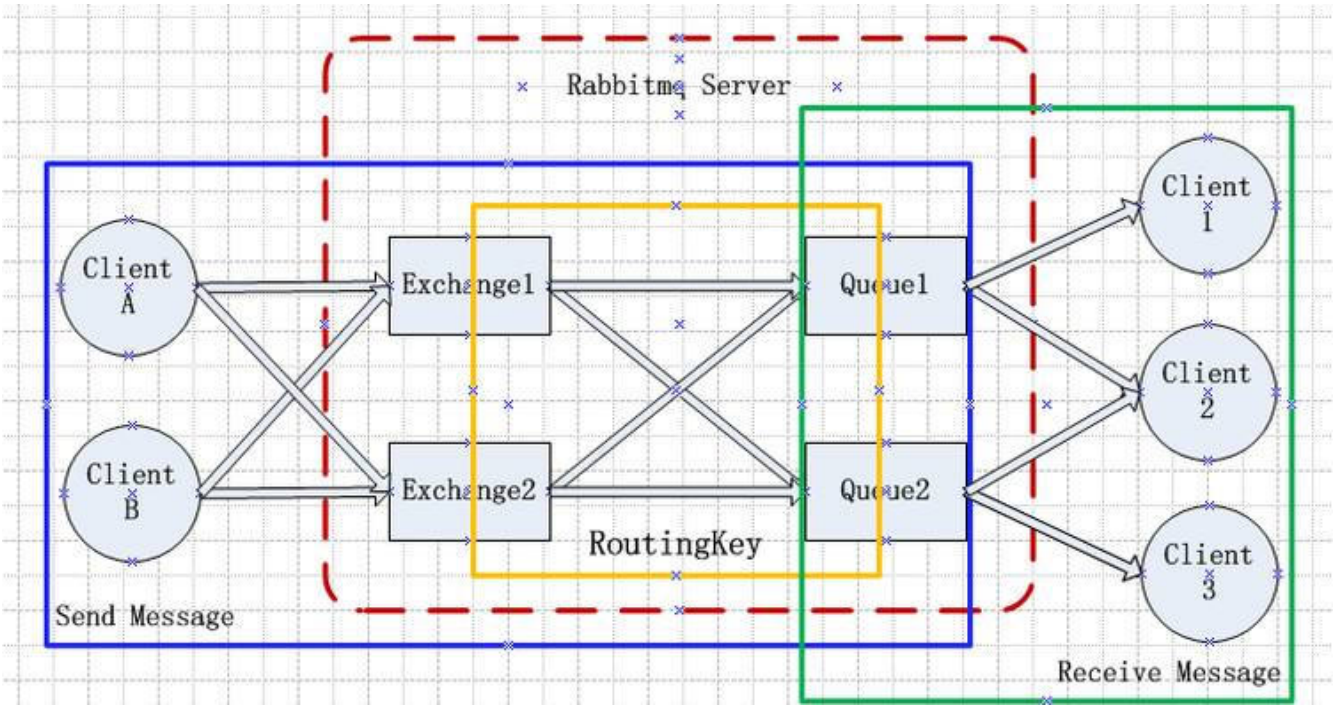
什么是AMQP协议

高级消息队列协议Advanced Message Queuing Protocol

是具有现代特征的二进制协议。是一个提供统一消息服务的的应用层标准高级消息队列协议，是应用层协议的一个开放标准，为面向消息的中间件设计。

是一个规范，里面有很多的概念，我们开发的时候按着他这个标准走就可以了。

AMQP核心概念



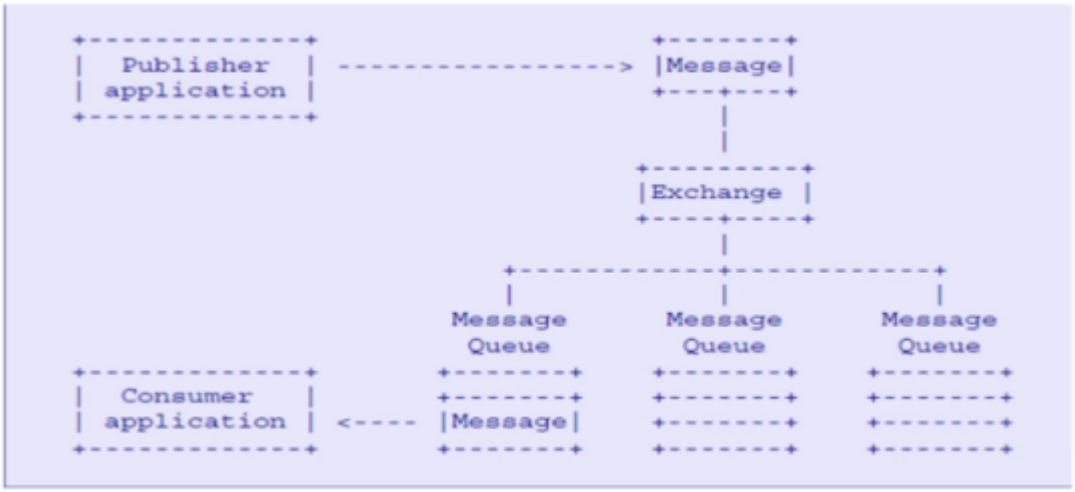
生产者把消息交给服务器，服务器里面有虚拟主机，主机里面有ampq的核心exchange交换机。生产者需要有服务器的ip和端口号，找到服务器，服务器需要把消息投递到哪个虚拟主机上。接下来，虚拟主机把消息交给交换机。我们的rabbitMq就是用来解耦了，做到这里，消息的生产者的任务就做完了。

rabbitMQ会把消息交给消费者。消费者会监听消息队列message queue。交换机和消息队列会进行绑定。

具体的概念：

1. server：又称broker，接受客户端的链接，实现amqp实体服务。
2. Connection：链接，应用程序跟broker的网络链接。
3. channel：网络信道，几乎所有的操作都是在channel中进行。数据的流转都要在channel上进行。channel是进行消息读写的通道。客户端可以建立多个channel，每个channel代表一个会话任务。
4. message：消息，服务器与应用程序之间传送的数据，由Properties和body组成。Properties可以对消息进行修饰，比如消息的优先级，延迟等高级特性。body则就是消息体的内容。
5. virtual host：虚拟地址，用于进行逻辑隔离，最上层的消息路由。一个虚拟地址里面可以有多个交换机exchange和消息队列message queue。
6. exchange：交换机，接收消息，根据路由机转发消息到绑定的队列。
7. binding：绑定。交换机和队列之间的虚拟链接。绑定中可以包含routing key。
8. routing key：一个路由规则，虚拟机可以用它来确定如何路由一个特定消息。
9. queue：消息队列，保存消息并将它们转发给消费者。

消息如何流转



生产者生产出消息，投递到交换机上，一个交换机可以绑定多个消息队列。

为什么只有一个消息队列中有消息？

交换机接收到消息后会根据路由规则找到指定的消息队列。所以生产者生产消息时需要指定消息的routing key。交换机会把消息交给消息队列。消费者就可以监听消息队列，从消息队列中获取消息。

安装和使用

1. 安装linux必要依赖包。
2. 官网下载rabbitMQ的安装包，先安装erlang。
3. 下载rabbitMQ的必须安装包。
4. 配置rabbitMQ。

   核心配置文件rabbit.app可以看到默认的端口号是5672.

   lookback_users的用户名[guest]

ctl：控制相关的命令

plugins：插件管理

server：启停服务

启动rabbitMQ

rabbitmq-server start &

启动时如果发生提示已经启动需要手动kill进程

ps -ef | grep rabbit

启动后可以通过lsof -i：5672查看启动进程

停止rabbitMQ服务

rabbitmqctl stop_app

管理插件：rabbitmq-plugins list可以查看现在的插件列表

启动管控台：rabbitmq-plugins enable rabbitmq_management

访问地址：http://ip:15672

访问前可以关闭防火墙：**systemctl stop firewalld**

命令行和管控台

基础操作：

1. rabbitmqctl stop_app 关闭应用
2. rabbitmqctl start_app 启动应用
3. rabbitmqctl status 查看节点状态
4. rabbitmqctl add_user username password 添加用户
5. rabbitmqctl list_users 列出所有用户
6. rabbitmqctl delete_user username 删除用户
7. rabbitmqctl clear_permissions -p vhostpath username 清除用户权限
8. rabbitmqctl reset 移除所有数据，要在rabbitmqctl stop_app之后使用
9. rabbitmqctl list_queues 列出所有的消息队列
10. rabbitmqctl list_exchanges 列出所有的交换机

控制台：

交换机页面的介绍

演示添加虚拟主机，给虚拟主机指定用户操作。

管控台可以操作的界面ctl都可以通过命令操作。

介绍管控台的概览。

消息的生产和消费

ConnectionFactory：获取连接的工厂

Connection:一个客户端和server的连接

Channel：数据通信信道，可以发送和接受消息

Queue：具体的消息存储队列

QueuingConsumer：消息队列的消费者

Delivery：mq对消息和channel的java封装

消息的生产者代码：

```
public static void main(String[] args) throws IOException, TimeoutException {
        ConnectionFactory factory=new ConnectionFactory();
```

```
        factory.setHost("192.168.157.128");
        factory.setPort(5672);
        factory.setVirtualHost("/");

        Connection conn = factory.newConnection();
        Channel channel=conn.createChannel();
        for(int i=0;i<5;i++){
            String body="hello rabbitmq!";
            //第一个参数是交换机的名称，为空表示使用默认交换机，第二个参数表示路由键，第三个参数表示消息的
属性。最后表示消息的内容
            channel.basicPublish("","test001",null,body.getBytes());
        }
        channel.close();
        conn.close();
    }
```

消息的消费者代码：

```
ConnectionFactory factory=new ConnectionFactory();
factory.setHost("192.168.157.128");
factory.setPort(5672);
factory.setVirtualHost("/");

Connection conn = factory.newConnection();
Channel channel=conn.createChannel();
String queueName="test001";
//声明队列
//第一个参数是队列的名称，第二个参数表示是否持久化消息队列，第三个参数表示channel独占，包装顺序的消费。
// 第四个参数表示是否自动删除。当队列没有绑定交换机就自动删除。第五个参数是扩展参数
channel.queueDeclare(queueName,true,false,false,null);
//创建消费者
QueueingConsumer consumer = new QueueingConsumer(channel);
//设置channel，
/*
        第一个参数是队列名.第二个参数表示ack消息是否自动签收，第三个是消费者对象
                */
channel.basicConsume(queueName,true,consumer);
//获取消息
while(true){
    QueueingConsumer.Delivery delivery = consumer.nextDelivery();
    String message=new String(delivery.getBody());
    System.out.println("消费端："+message);
}
```

交换机详解

exchange：接收消息，并根据路由键转发消息所绑定的队列。

交换机的属性：

name：交换机的名称

type：交换机的类型direct，topic，fanout，headers

durability：是否需要持久化，true为持久化。

auto delete：当最后一个绑定到exchange上的队列删除后，自动删除该exchange。

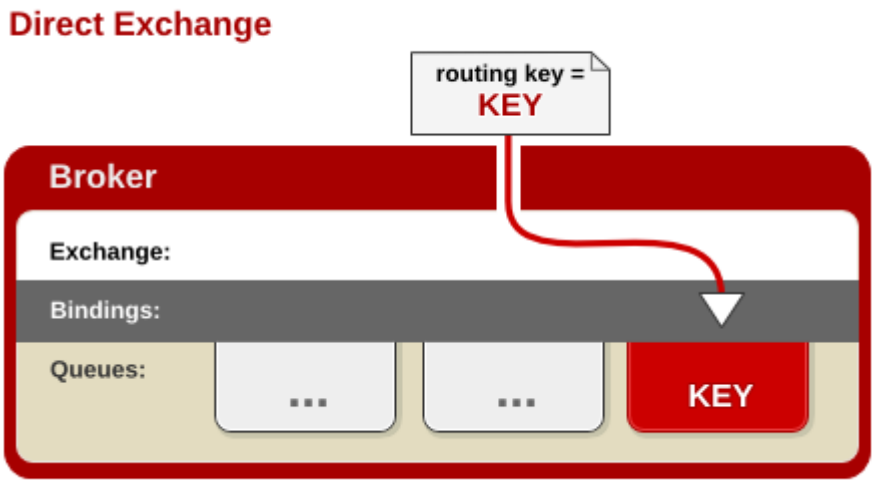
internal：当前exchange是否用于rabbitMQ内部使用，默认为false。

arguments：可扩展参数。用户自自定义的交换机时，用到的参数。

交换机的类型

direct:

1. 所有发送到directExchange的消息被转发到RouteKey中指定的Queue
2. rabbitmq有一个自带的exchange叫default exchange，这个交换机是direct类型的。rabbitmq会让路由键跟队列名相等进行绑定。



topic:

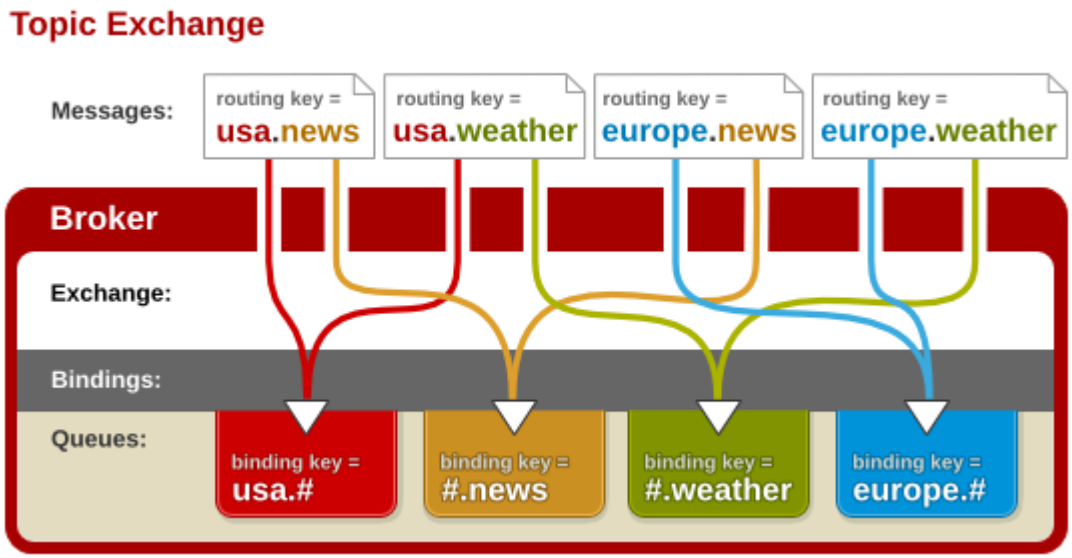
1. 所有发送到topic exchange的消息被转发到所有关心RouteKey的Queue上
2. Exchange将RouteKey和某些队列进行模糊匹配，此时队列需要绑定一个Topic

   模糊匹配可以使用通配符：

   #可以匹配一个或多个词

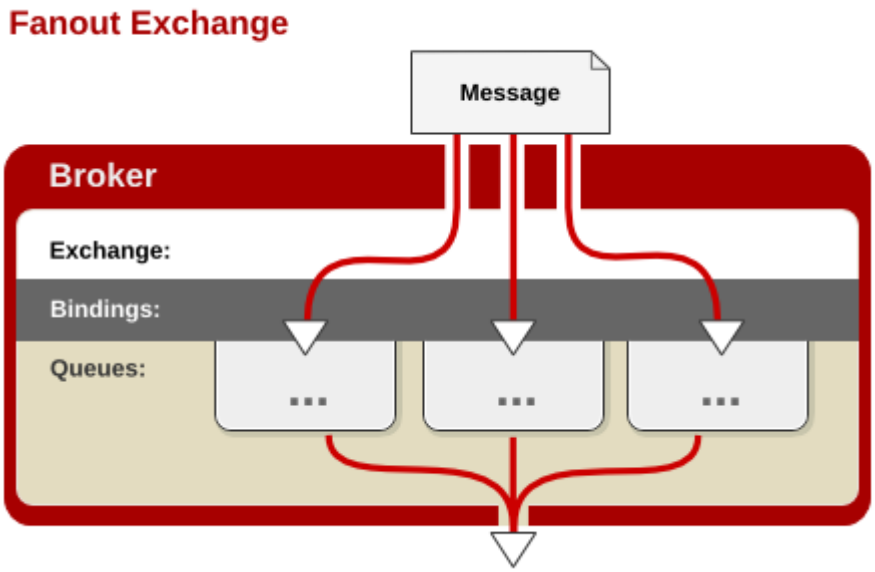
   *只能匹配一个词

   比如："log.#"可以匹配到“log.info.oa”。“log.*”只会匹配到“log.error”



Fanout：

1. 不处理路由键，只需要简单的将队列绑定到交换机上。
2. 发送到交换机的消息都会被转发到与该交换机绑定的所有队列上
3. fanout交换机转发消息是最快的。



队列，绑定，虚拟主机，消息

绑定：交换机跟队列的连接关系，两个交换机的连接关系。包含routing key

消息队列：实际存储消息数据的。

durability：是否持久化

auto_delete:当最后一个监听被移除之后，该队列是否被自动删除。

消息：服务器和应用之间传递的数据。有properties和payload(body)组成。

消息的常见属性：

delivery mode：送达模式，是否持久化

headers：自定义消息属性，消息里面有常规的一些属性，一些特别的属性需要在headers里面进行定义

content_type:消息内容的类型

content_encoding:编码集

priority:优先级，从0-9，数字越大优先级越高

虚拟主机：用于进行逻辑隔离，最上层的消息路由。一个虚拟主机里面可以有若干个交换机和消息队列。