1、上千万条消息在mq中积压了几个小时还没解决：

1）先修复consumer的问题，确保其恢复消费速度，然后将现有consumer都停掉；
2）新建一个topic，partition是原来的10倍，临时建立好原先10倍或者20倍的queue数量；
3）然后写一个临时的分发数据的consumer程序，这个程序部署上去消费积压的数据；
消费之后不做耗时的处理，直接均匀轮询写入临时建立好的10倍数量的queue；
4）接着临时征用10倍的机器来部署consumer，每一批consumer消费一个临时queue的数据；
5）这种做法相当于是临时将queue资源和consumer资源扩大10倍，以正常的10倍速度来消费数据；
6）等快速消费完积压数据之后，得恢复原先部署架构，重新用原先的consumer机器来消费消息。
总结：
1. 修复并停掉consumer；
2. 新建一个topic，partition是原来的10倍，建立临时queue，数量是原来的10倍或20倍；
3. 写临时consumer程序，临时征用10倍的机器去消费数据；
4. 消费完成之后，恢复原先consumer；

2、rabbitmq设置过期时间，部分消息丢失：

采取批量重导方法：将丢失的那批数据查询导入到mq里面。

3、RabbitMQ 上的一个 queue 中存放的 message 是否有数量限制?

可以认为是无限制，因为限制取决于机器的内存，但是消息过多会导致处理效率的下降。

4、分布式部署：

RabbitMQ无法容忍不同数据中心之间网络延迟，但是可以通过3种方式实现分布式部署：Federation和Shovel。

5、如何确保消息正确地发送至RabbitMQ?

RabbitMQ使用发送方确认模式，确保消息正确地发送到RabbitMQ。

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（not acknowledged，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。

6、如何确保消息接收方消费了消息?

接收方消息确认机制：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。

> 特殊情况：
> 1、如果消费者接收到消息，在确认之前断开了连接或取消订阅，RabbitMQ会认为消息没有被分发，然后重新分发给下一个订阅的消费者。（可能存在消息重复消费的隐患，需要根据bizId去重）
> 2、如果消费者接收到消息却没有确认消息，连接也未断开，则RabbitMQ认为该消费者繁忙，将不会给该消费者分发更多的消息。

7、如何避免消息重复投递或重复消费?

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。

8、消息基于什么传输?

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。RabbitMQ使用信道的方式来传输数

据。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。

> 1、RabbitMQ采用类似NIO（Non-blocking I/O）做法，选择TCP连接复用，不仅可以减少性能开销，同时也便于管理。
> 2、每个线程把持一个信道，所以信道服用了Connection的TCP连接。同时RabbitMQ可以确保每个线程的私密性，就像拥有独立的连接一样。

9、消息如何分发?

若该队列至少有一个消费者订阅，消息将以循环（round-robin）的方式发送给消费者。每条消息只会分发给一个订阅的消费者（前提是消费者能够正常处理消息并进行确认）。

10、消息怎么路由?

从概念上来说，消息路由必须有三部分：交换器、路由、绑定。生产者把消息发布到交换器上；绑定决定了消息如何从交换器路由到特定的队列；消息最终到达队列，并被消费者接收。

1、消息发布到交换器时，消息将拥有一个路由键（routing key），在消息创建时设定。
2、通过队列路由键，可以把队列绑定到交换器上。
3、消息到达交换器后，RabbitMQ会将消息的路由键与队列的路由键进行匹配（针对不同的交换器有不同的路由规则）。
4、如果能够匹配到队列，则消息会投递到相应队列中；如果不能匹配到任何队列，消息将进入“黑洞”。

11、如何确保消息不丢失?

消息持久化的前提是：将交换器/队列的durable属性设置为true，表示交换器/队列是持久交换器/队列，在服务器崩溃或重启之后不需要重新创建交换器/队列（交换器/队列会自动创建）。

如果消息想要从Rabbit崩溃中恢复，那么消息必须：

1、在消息发布前，通过把它的“投递模式”选项设置为2（持久）来把消息标记成持久化
2、将消息发送到持久交换器
3、消息到达持久队列

> RabbitMQ确保持久性消息能从服务器重启中恢复的方式是，将它们写入磁盘上的一个持久化日志文件，当发布一条持久性消息到持久交换器上时，Rabbit会在消息提交到日志文件后才发送响应（如果消息路由到了非持久队列，它会自动从持久化日志中移除）。一旦消费者从持久队列中消费了一条持久化消息，RabbitMQ会在持久化日志中把这条消息标记为等待垃圾收集。如果持久化消息在被消费之前RabbitMQ重启，那么Rabbit会自动重建交换器和队列（以及绑定），并重播持久化日志文件中的消息到合适的队列或者交换器上。

12、使用RabbitMQ有什么好处?

1. 应用解耦（系统拆分）
2. 异步处理（预约挂号业务处理成功后，异步发送短信、推送消息、日志记录等，可以大大减小响应时间）
3. 消息分发
4. 流量削峰：将请求发送到队列中，短暂的高峰期积压是允许的。
5. 消息缓冲
...

13、消息队列有什么缺点?

1. 系统可用性降低：消息队列出问题影响业务；
2. 系统复杂性增加：加入消息队列，需要考虑很多方面的问题，比如：一致性问题、如何保证消息不被重复消费、如何保证消息可靠性传输等。

14、MQ如何选型?

特性	ActiveMQ	RabbitMQ	RocketMQ	kafka
开发语言	java	erlang	java	scala
单机吞吐量	万级	万级	10万级	10万级
时效性	ms级	us级	ms级	ms级以内
可用性	高(主从架构)	高(主从架构)	非常高(分布式架构)	非常高(分布式架构)
功能特性	成熟的产品，在很多公司得到应用；有较多的文档；各种协议支持较好	基于erlang开发，所以并发能力很强，性能极其好，延时很低;管理界面较丰富	MQ功能比较完备，扩展性佳	只支持主要的MQ功能，像一些消息查询，消息溯等功能没有提供，是为大数据准备的，在数据领域应用广。

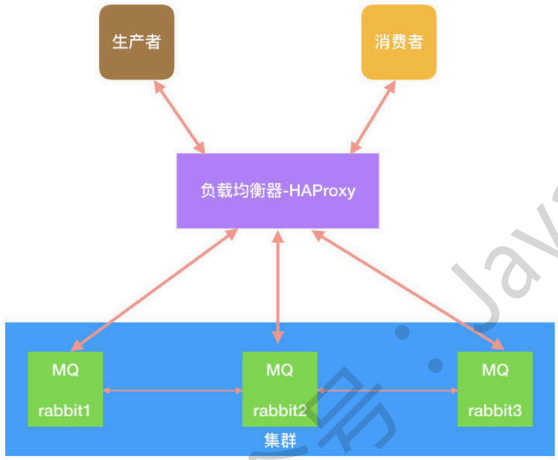
1. 中小型公司首选RabbitMQ：管理界面简单，高并发。

2. 大型公司可以选择RocketMQ：更高并发，可对rocketmq进行定制化开发。

3. 日志采集功能，首选kafka，专为大数据准备。

15、如何保证消息队列高可用?

1. 集群：

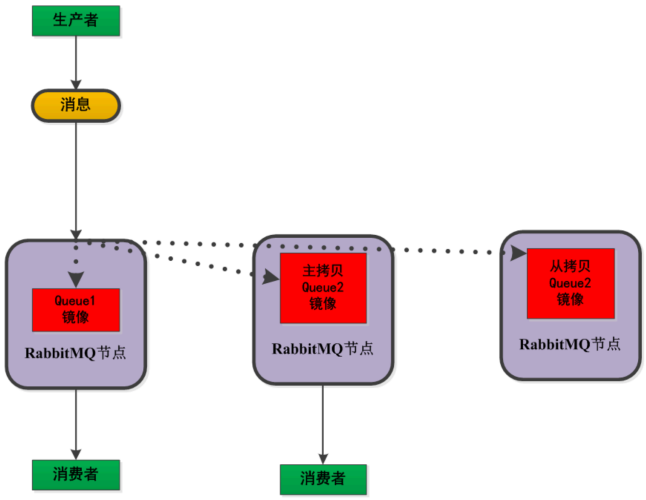


RabbitMQ Cluster 架构图

集群可以扩展消息通信的吞吐量，但是不会备份消息，备份消息要通过镜像队列的方式解决。

队列存储在单个节点、交换器存储在所有节点。

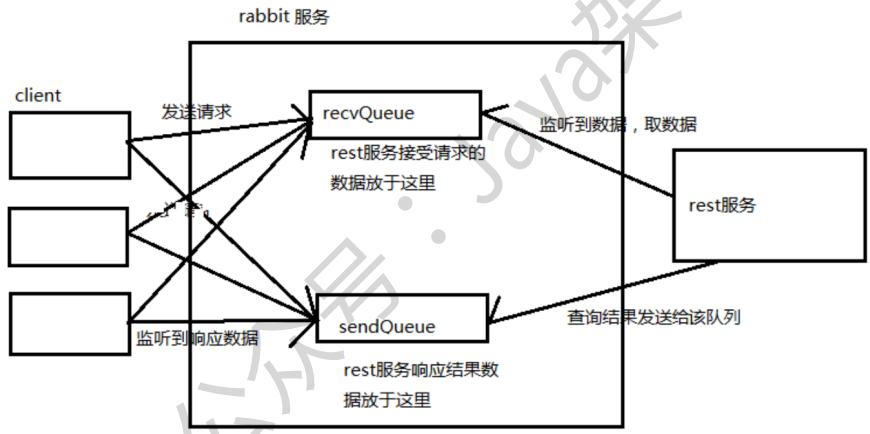
2、镜像队列：将需要消费的队列变为镜像队列，存在于多个节点，这样就可以实现RabbitMQ的HA高可用性。作用就是消息实体会主动在镜像节点之间实现同步，而不是像普通模式那样，在consumer消费数据时临时读取。缺点就是，集群内部的同步通讯会占用大量的网络带宽。



16、如何保证消息的顺序性?

1. 通过某种算法，将需要保持先后顺序的消息放到同一个消息队列中(kafka中就是partition,rabbitMq中就是queue)。然后只用一个消费者去消费该队列。
2. 可以在消息体内添加全局有序标识来实现。

17、使用RabbitMQ增加rest服务吞吐量。



18、RabbitMQ交换器有哪些类型?

1. fanout交换器：它会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交换器：direct类型的交换器路由规则很简单，它会把消息路由到哪些BindingKey和RoutingKey完全匹配的队列中；
3. topic交换器：匹配规则比direct更灵活。
4. headers交换器：根据发送消息内容的headers属性进行匹配（由于性能很差，不实用）。

> 常用的交换器主要分为以下三种：
> 1、direct：如果路由键完全匹配，消息就被投递到相应的队列
> 2、fanout：如果交换器收到消息，将会广播到所有绑定的队列上
> 3、topic：可以使来自不同源头的消息能够到达同一个队列。 使用topic交换器时，可以使用通配符，比如：“*”匹配特定位置的任意文本，“.”把路由键分为了几部分，“#”匹配所有规则等。特别注意：发往topic交换器的消息不能随意的设置选择键（routing_key），必须是由"."隔开的一系列的标识符组成。

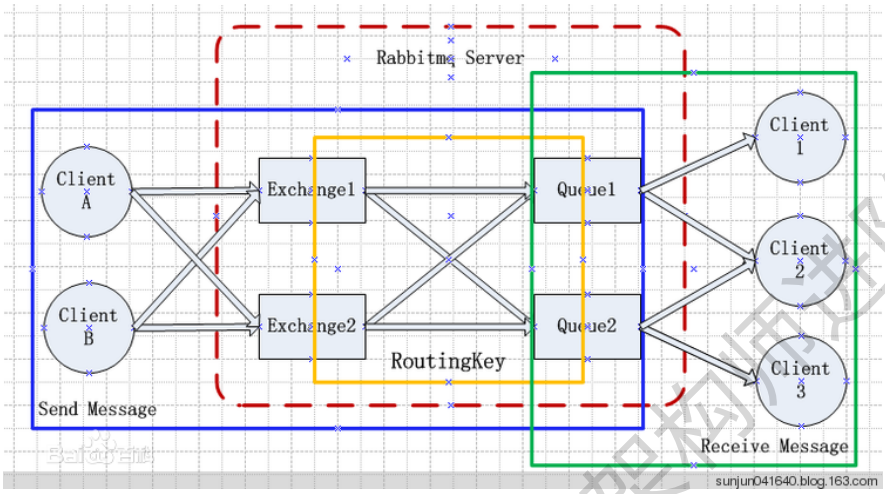
19、RabbitMQ如何保证数据一致性?

1. 生产者确认机制：消息持久化后异步回调通知生产者，保证消息已经发出去；
2. 消息持久化：设置消息持久化；
3. 消费者确认机制：消费者成功消费消息之后，手动确认，保证消息已经消费。

20、RabbitMQ消费者自动扩展数量

SimpleMessageListenerContainer可根据RabbitMQ消息堆积情况自动扩展消费者数量。

21、RabbitMQ结构：



a. Broker：简单来说就是消息队列服务器实体。

b. Exchange：消息交换机，它指定消息按什么规则，路由到哪个队列。

c. Queue：消息队列载体，每个消息都会被投入到一个或多个队列。

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

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

f. vhost：虚拟主机，一个broker里可以开设多个vhost，用作不同用户的权限分离。

g. producer：消息生产者，就是投递消息的程序。

h. consumer：消息消费者，就是接受消息的程序。

i. 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。

22、rabbitmq队列与消费者的关系?

1. 一个队列可以绑定多个消费者；
2. 消息默认以循环的方式发送给消费者；
3. 消费者收到消息默认自动确认，也可以改成手动确认。