1.什么是 ActiveMQ?

activeMQ 是一种开源的，实现了 JMS1.1 规范的，面向消息(MOM)的中间件，为应用程序提供高效的、可扩展的、稳定的和安全的企业级消息通信

2. ActiveMQ 服务器宕机怎么办？

这得从 ActiveMQ 的储存机制说起。在通常的情况下，非持久化消息是存储在内存中的，持久化消息是存储在文件中的，它们的最大限制在配置文件的<systemUsage>节点中配置。但是，在非持久化消息堆积到一定程度，内存告急的时候，ActiveMQ 会将内存中的非持久化消息写入临时文件中，以腾出内存。虽然都保存到了文件里，但它和持久化消息的区别是，重启后持久化消息会从文件中恢复，非持久化的临时文件会直接删除。

那如果文件增大到达了配置中的最大限制的时候会发生什么？我做了以下实验：

设置 2G 左右的持久化文件限制，大量生产持久化消息直到文件达到最大限制，此时生产者阻塞，但消费者可正常连接并消费消息，等消息消费掉一部分，文件删除又腾出空间之后，生产者又可继续发送消息，服务自动恢复正常。

设置 2G 左右的临时文件限制，大量生产非持久化消息并写入临时文件，在达到最大限制时，生产者阻塞，消费者可正常连接但不能消费消息，或者原本慢速消费的消费者，消费突然停止。整个系统可连接，但是无法提供服务，就这样挂了。

具体原因不详，解决方案：尽量不要用非持久化消息，非要用的话，将临时文件限制尽可能的调大。

3. 丢消息怎么办？

这得从 java 的 java.net.SocketException 异常说起。简单点说就是当网络发送方发送一堆数据，然后调用 close 关闭连接之后。这些发送的数据都在接收者的缓存里，接收者如果调用 read 方法仍旧能从缓存中读取这些数据，尽管对方已经关闭了连接。但是当接收者尝试发送数据时，由于此时连接已关闭，所以会发生异常，这个很好理解。不过需要注意的是，当发生 SocketException 后，原本缓存区中数据也作废了，此时接收者再次调用 read 方法去读取缓存中的数据，就会报 Software caused connection abort: recv failed 错误。

通过抓包得知，ActiveMQ 会每隔 10 秒发送一个心跳包，这个心跳包是服务器发送给客户端的，用来判断客户端死没死。如果你看过上面第一条，就会知道非持久化消息堆积到一定程度会写到文件里，这个写的过程会阻塞所有动作，而且会持续 20 到 30 秒，并且随着内存的增大而增大。当客户端发完消息调用 connection.close()时，会期待服务器对于关闭连接的回答，如果超过 15 秒没回答就直接调用 socket 层的 close 关闭 tcp 连接了。这时客户端发出的消息其实还在服务器的缓存里等待处理，不过由于服务器心跳包的设置，导致发生了 java.net.SocketException 异常，把缓存里的数据作废了，没处理的消息全部丢失。

解决方案：用持久化消息，或者非持久化消息及时处理不要堆积，或者启动事务，启动事务后，commit()方法会负责任的等待服务器的返回，也就不会关闭连接导致消息丢失了。

4. 持久化消息非常慢。

默认的情况下，非持久化的消息是异步发送的，持久化的消息是同步发送的，遇到慢一点的硬盘，发送消息的速度是无法忍受的。但是在开启事务的情况下，消息都是异步发送的，效率会有 2 个数量级的提升。所以在发送持久化消息时，请务必开启事务模式。其实发送非持久化消息时也建议开启事务，因为根本不会影响性能。

5. 消息的不均匀消费。

有时在发送一些消息之后，开启 2 个消费者去处理消息。会发现一个消费者处理了所有的消息，另一个消费者根本没收到消息。原因在于 ActiveMQ 的 prefetch 机制。当消费者去获取消息时，不会一条一条去获取，而是一次性获取一批，默认是 1000 条。这些预获取的消息，在还没确认消费之前，在管理控制台还是可以看见这些消息的 ,但是不会再分配给其他消费者 ,此时这些消息的状态应该算作“已分配未消费”，如果消息最后被消费 ,则会在服务器端被删除 ,如果消费者崩溃 ,则这些消息会被重新分配给新的消费者。但是如果消费者既不消费确认，又不崩溃，那这些消息就永远躺在消费者的缓存区里无法处理。更通常的情况是，消费这些消息非常耗时，你开了 10 个消费者去处理，结果发现只有一台机器吭哧吭哧处理，另外 9 台啥事不干。

解决方案：将 prefetch 设为 1，每次处理 1 条消息，处理完再去取，这样也慢不了多少。

6. 死信队列。

如果你想在消息处理失败后，不被服务器删除，还能被其他消费者处理或重试，可以关闭 AUTO_ACKNOWLEDGE，将 ack 交由程序自己处理。那如果使用了 AUTO_ACKNOWLEDGE，消息是什么时候被确认的，还有没有阻止消息确认的方法？有！

消费消息有 2 种方法，一种是调用 consumer.receive()方法，该方法将阻塞直到获得并返回一条消息。这种情况下，消息返回给方法调用者之后就自动被确认了。另一种方法是采用 listener 回调函数，在有消息到达时，会调用 listener 接口的 onMessage 方法。在这种情况下，在 onMessage 方法执行完毕后，消息才会被确认，此时只要在方法中抛出异常，该消息就不会被确认。那么问题来了，如果一条消息不能被处理，会被退回服务器重新分配，如果只有一个消费者，该消息又会重新被获取，重新抛异常。就算有多个消费者，往往在一个服务器上不能处理的消息，在另外的服务器上依然不能被处理。难道就这么退回--获取--报错死循环了吗？

在重试 6 次后，ActiveMQ 认为这条消息是“有毒”的，将会把消息丢到死信队列里。如果你的消息不见了，去 ActiveMQ.DLQ 里找找，说不定就躺在那里。

7. ActiveMQ 中的消息重发时间间隔和重发次数吗？

ActiveMQ : 是 Apache 出品，最流行的，能力强劲的开源消息总线。是一个完全支持 JMS1.1 和 J2EE 1.4 规范的 JMS Provider 实现。JMS（Java 消息服务）：是一个 Java 平台中关于面向消息中间件（MOM）的 API，用于在两个应用程序之间，或分布式系统中发送消息，进行异步通信。

首先，我们得大概了解下，在哪些情况下，ActiveMQ 服务器会将消息重发给消费者，这里为简单起见，假定采用的消息发送模式为队列（即消息发送者和消息接收者）。

① 如果消息接收者在处理完一条消息的处理过程后没有对 MOM 进行应答，则该消息将由 MOM 重发.

② 如果我们队某个队列设置了预读参数（consumer.prefetchSize），如果消息接收者在处理第一条消息时（没向 MOM 发送消息接收确认）就宕机了，则预读数量的所有消息都将被重发!

③ 如果 Session 是事务的，则只要消息接收者有一条消息没有确认，或发送消息期间 MOM 或客户端某一方突然宕机了，则该事务范围中的所有消息 MOM 都将重发。

④ 说到这里，大家可能会有疑问，ActiveMQ 消息服务器怎么知道消费者客户端到底是消息正在处理中还没来得急对消息进行应答还是已经处理完成了没有应答或是宕机了根本没机会应答呢？其实在所有的客户端机器上，内存中都运行着一套客户端的 ActiveMQ 环境，该环境负责缓存发来的消息，负责维持着和 ActiveMQ 服务器的消息通讯，负责失效转移（fail-over）等，所有的判断和处理都是由这套客户端环境来完成的。

我们可以来对 ActiveMQ 的重发策略（Redelivery Policy）来进行自定义配置，其中的配置参数主要有以下几个：

可用的属性

属性 默认值 说明

l collisionAvoidanceFactor 默认值 0.15， 设置防止冲突范围的正负百分比，只有启用 useCollisionAvoidance 参数时才生效。

l maximumRedeliveries 默认值 6， 最大重传次数，达到最大重连次数后抛出异常。为-1 时不限制次数，为 0 时表示不进行重传。

l maximumRedeliveryDelay 默认值-1， 最大传送延迟，只在 useExponentialBackOff 为 true 时有效（V5.5），假设首次重连间隔为 10ms，倍数为 2，那么第二次重连时间间隔为 20ms，第三次重连时间间隔为 40ms，当重连时间间隔大的最大重连时间间隔时，以后每次重连时间间隔都为最大重连时间间隔。

l initialRedeliveryDelay 默认值 1000L， 初始重发延迟时间

l redeliveryDelay 默认值 1000L， 重发延迟时间，当 initialRedeliveryDelay=0 时生效（v5.4）

l useCollisionAvoidance 默认值 false， 启用防止冲突功能，因为消息接收时是可以使用多线程并发处理的，应该是为了重发的安全性，避开所有并发线程都在同一个时间点进行消息接收处理。所有线程在同

一个时间点处理时会发生什么问题呢？应该没有问题，只是为了平衡 broker 处理性能，不会有时很忙，有时很空闲。

l useExponentialBackOff 默认值 false，启用指数倍数递增的方式增加延迟时间。

l backOffMultiplier 默认值 5，重连时间间隔递增倍数，只有值大于 1 和启用 useExponentialBackOff 参数时才生效。