

## 说说RabbitMQ延迟队列实现原理?

=====

使用 RabbitMQ 和 RocketMQ 的人是幸运的，因为这两个 MQ 自身提供了延迟队列的实现，不像用 Kafka 的同学那么苦逼，还要自己实现延迟队列。当然，这都是题外话，今天咱们重点来聊聊 RabbitMQ 延迟队列的实现原理，以及 RabbitMQ 实现延迟队列的优缺点有哪些？

> 很多人知道使用 RabbitMQ 是可以实现延迟队列的，但对于 RocketMQ 自身也提供了延迟队列这件事却持有不同态度，这是因为网上有些资料说 RocketMQ 和 Kafka 没有内置延迟队列。其实这种说法是因为，RocketMQ 在早期版本中确实没有内置延迟队列，但在 4.x 就内置了 18 个级别的延迟队列了（最长支持 2 小时的延迟队列），5.x 就支持随机延迟时间的延迟队列了，所以这里需要特殊强调一下。

### 1.什么是延迟队列?

-----

延迟队列（Delay Queue）是一种特殊类型的队列，它的主要特点是可以让进入队列的元素在指定的延迟时间之后才被取出进行处理。

延迟队列的主要使用场景有以下这些：

1. **\*\*订单超时处理\*\***：在电商系统中，如果用户下单后未在一定时间内支付，订单可能会被自动取消。可以将订单放入延迟队列，在设定的延迟时间（如 30 分钟）后取出处理取消操作。
2. **\*\*任务重试\*\***：当某个任务执行失败时，将其放入延迟队列，等待一段时间（如 5 分钟）后重新执行。
3. **\*\*消息延迟发送\*\***：某些消息不需要立即发送，而是在指定的延迟时间后发送，例如定时提醒消息。
4. **\*\*缓存过期处理\*\***：缓存中的数据可能有一定的有效期，将即将过期的数据放入延迟队列，到期后进行删除或更新操作。

### 2.延迟任务实现方法

-----

那么延迟队列的实现方式有哪些呢？

延迟队列的实现方式通常有以下几种：

1. **\*\*基于 JDK 提供的 DelayQueue 来实现\*\***：它是内存级别的延迟队列，重启应用之后消息会丢失，并且只支持单机版延迟队列，所以一般不用。
2. **\*\*基于 MQ 的延迟队列\*\***：例如使用 RabbitMQ 来实现延迟队列，他们适合处理动态和临时延迟任务，不像定时任务一样，适合处理正式的、固定的延迟任务。
3. **\*\*基于定时任务组件实现延迟任务\*\***：例如 XXLJob 或 Quartz 等框架来实现延迟任务，他们适合处理固定（执行）频率的延迟任务。

> 我们通常会使用延迟队列来存储（和实现）延迟消息，所以大部分时候，我们说的延迟队列和延迟消息其实是一回事。

### 3.使用RabbitMQ实现延迟队列

-----

使用 RabbitMQ 实现延迟队列有以下两种实现方式：

1. **\*\*通过死信队列实现延迟任务\*\***：将正常的消息放到没有消息订阅者的消息队列（消息自然就会过期），等消息过期之后会进入死信队列，通过订阅死信队列消费消息，从而实现延迟队列，如下图所示：

![image.png](https://p6-xtjj-sign.byteimg.com/tos-cn-i-73owjymdk6/de966fa2aee04ccda7f398d9b3fad868~tplv-73owjymdk6-watermark.image?rk3s=f64ab15b&x-expire=1721730933&x-signature=wEzszwhefJtqtElrnMHt3snDH8l%3D)

2. **\*\*通过官方提供的延迟插件实现延迟功能\*\***。

早期大部分公司都会采用第一种方式，而随着 RabbitMQ 3.5.7（2015 年底发布）的延迟插件的发布，因为其使用更简单、更方便，所以它现在才是大家普遍会采用的，实现延迟队列的方式。

### ### 3.1 实现原理分析

使用延迟插件的实现原理是通过创建一个延迟交换机 (Delay Exchange)，延迟消息首先会把消息投递到延迟交换机，并不是直接将消息投递业务队列（所以不会立即执行），由延迟交换机控制消息在延迟一段时间后，再将消息投递到真正的队列中进行消费，从而实现延迟队列，它的实现流程如下图所示：

![image.png](https://p6-xtjj-sign.byteimg.com/tos-cn-i-73owjymdk6/0be38128319348c7a404a0a6fe181109~tplv-73owjymdk6-watermark.image?rk3s=f64ab15b&x-expire=1721730933&x-signature=GfFJYD%2Bt0bsT16ncoskXOurS%2Fy4%3D)

> 其中 Mnesia 可以理解为基于文件存储的数据库。

### ### 3.2 优缺点分析

使用死信队列实现延迟任务有个缺点，它不能实现随机延迟任务，每个无消费者的队列上只能设置一个 ttl（消息过期时间），所以只能实现固定过期时间的延迟任务。

使用延迟插件实现延迟任务有以下两个缺点：

1. **\*\*消息丢失问题\*\***：消息在真的被投递到目标消息队列之前，是存放在接收到了这个消息的服务端本地的 Mnesia 里面。也就是说，如果这个时候还没有刷新磁盘，那么消息就会丢失；如果这个节点不可用了，那么消息也同样会丢失。
2. **\*\*高并发问题\*\***：这种实现方式不支持高并发场景，因为它只有一个延迟交换机，当高并发或数据量比较大时执行效率就会比较低。

### 课后思考

-----

如何解决 RabbitMQ 延迟插件造成的这两个问题？如何实现 Kafka 的延迟队列？

> 本文已收录到我的面试小站 [www.javacn.site](http://cxyroad.com/"https://www.javacn.site")，其中包含的内容有：Redis、JVM、并发、并发、MySQL、Spring、Spring MVC、Spring Boot、Spring Cloud、MyBatis、设计模式、消息队列等模块。

原文链接: <https://juejin.cn/post/7392070976093519935>

