订单超时自动取消如何实现?

如果在用户在一定的时间（比如 24 小时）内没有付款，那么订单会自动被取消，这是电商和外卖平台很常见的一个功能。当你的简历中包含商城、外卖类型项目的话，问到这个问题的概率还是比较大的。

这个问题的本质是在考察你对延时任务解决方案的了解。类似的问题还有：

1. 会议预定成功后，会议开始前 30 分钟通知所有预定该会议的用户，如何实现?
2. 用户订单完成后未点击收货，10 天后自动同意收货，如何实现?
3. 用户发起退款后，三天之内未被处理自动退款给用户，如何实现?
4. ……

我这篇文章已经详细总结了定时和延时任务的常见解决方案：Java 定时任务详解 <https://javaguide.cn/system-design/schedule-task.html> ，一定要看看。

单机延时任务方案有 `Timer` 、 `ScheduledExecutorService` 、 `DelayQueue` 、Spring Task 和时间轮，其中最常用也是比较推荐使用的是时间轮。

分布式延时任务的解决方案有 Redis 和 MQ。实际项目中，MQ 延时任务用的更多一些，可以降低业务之间的耦合度。大部分消息队列，例如 RocketMQ、RabbitMQ，都支持定时/延时消息。不过，在使用 MQ 定时/延时消息之前一定要看清楚其使用限制，以免不适合项目需求，例如 RocketMQ 定时时长最大值默认为 24 小时且不支持自定义修改、定时精度为秒。

如果你的项目没有用到 MQ 或者用到的 MQ 对延迟任务支持不好的话，Redis 延时任务也是可以考虑的。基于 Redis 实现延时任务的功能无非就下面两种方案：

1. Redis 过期事件监听
2. Redisson 内置的延时队列（推荐）

《后端面试高频系统设计&场景题》中有一篇文章详细介绍到这两种方案的优缺点，这里就不重复介绍了。

如何基于 Redis 实现延时任务?

类似的问题： 订单在 10 分钟后未支付就失效，如何用 Redis 实现？红包 24 小时未被查收自...

《后端面试高频系统设计&场景题》

这里顺带分享一下 RabbitMQ 延迟队列的两种实现方式和选择，以及 RocketMQ 新版本对延时消息的优化，面试的时候能聊到就更好了。

RabbitMQ 延迟队列的两种实现方式：

1. RabbitMQ 3.6.x 之前我们一般采用 DLX（Dead Letter Exchange，死信队列）+TLL（Time To Live，过期时间）。具体的原理是：将消息发送到一个设置了 TTL 的队列中，当消息过期后，会被转发到另一个设置了 DLX 的队列中，消费者监听这个 DLX 队列来获取延迟消息。
2. RabbitMQ 3.6.x 开始，RabbitMQ 官方提供了延迟队列的插件 rabbitmq_delayed_message_exchange <https://github.com/rabbitmq/rabbitmq-delayed-message-exchange> 。这种方式用的更多一些，比较简单方便，解决了 DLX+TLL 存在的一些问题（后面会提到）。通过这个插件，我们可以声明 x-delayed-message 类型的 Exchange（一种新的交换器类型），消息发送时指定消息头 x-delay 以毫秒为单位将消息进行延迟投递。这个插件支持的最大延迟时间是(2^32)-1 毫秒，大约 49 天。

DLX+TLL 这种方式存在一些问题比如：

- **存在消息阻塞问题** （主要问题）：如果设置的是队列统一过期时间放到死信队列，那么就不会有阻塞问题，因为所有的消息都会在同一时间过期，然后被投递到死信队列。但是这种方案也有一个缺点，就是它不能实现不同的延迟时间，所有的消息都必须等待同样的时间才能被消费。如果是设置每条消息的 TTL 不同，那么可能会出现这阻塞问题。因为过期时间的检测也是从消息队列头部开始的，而队列又遵循先进先

出，如果一个过期时间较长的消息在头部的话，可能就会导致阻塞其他过期时间较短的消息。

- **要为不同的延迟时间创建多个队列**：我们上面也说了队列统一过期时间可以解决头阻塞问题，但不能实现不同的延迟时间。如果想要实现不同的延迟时间，就需要单独为每一种过期时间创建一个对应的消息队列。如果延迟时间是动态可配置的，那么就需要动态地创建和删除队列。这样会增加系统复杂度、资源消耗和维护难度。而且，并不灵活，如果延迟时间是无规律的，那这种方式也不合适了。
- **不适合延迟时间较长的任务**：会占用原队列和死信队列的空间。如果消息过期时间太长，那么它们就会在队列中存储很久，占用内存或磁盘空间。

更推荐 RabbitMQ 延迟队列插件这种方式。这种方式中，消息并不会立即进入队列，而是先把他们保存在 Mnesia 数据库（Erlang 运行时中自带的一个分布式数据库管理系统，详细介绍可参考Mnesia 数据库 <https://elixirschool.com/zh-hans/lessons/storage/mnesia> ）中，然后通过一个定时器去查询需要被投递的消息，再把他们投递到 x-delayed-message 队列中。这种方式不存在消息阻塞问题，还可以实现灵活的延迟时间。并且，还避免过期时间太长的消息在队列中堆积，导致占用内存或磁盘空间。

RocketMQ 新版本对延时消息的优化：弥补延时消息的不足，RocketMQ 基于时间轮算法实现了定时消息！ <https://mp.weixin.qq.com/s/I91QRel-7CraP7zCRh0ISw>（RocketMQ 5.0 基于时间轮算法引入了定时消息，解决了延时级别只有 18 个、延时时间不准确等问题）。

Quartz、Elastic-Job、XXL-JOB 和 PowerJob 这几个是专门用来做分布式调度的框架，也可以实现延时任务，但一般不推荐这么做，它们更适合执行周期性的定时任务。如果使用这些分布式调度框架来说延时任务的话，可能会出现时间精度较差、性能无法满足等问题。另外，如果你需要用到一些高级特性比如支持任务在分布式场景下的分片和高可用、任务可视化管理（除了 Quartz，其他几个都支持）的话，那 Elastic-Job、XXL-JOB 和 PowerJob 或许是更好的选择（其他优秀的分布式任务调度框架也行）。

url=https%3A%2F%2Fwww.yuque.com%2Fsnailclimb%2Ftangw3%2Fsgageiesw7cqi1o7&pic=nul