【组件设计】基于 Redis 实现高性能、低延迟的延时消息的方案演进

来自： 芋道快速开发平台 Boot + Cloud

芋道源码

2023年08月09日 08:03

目录：

作者：尼布斯
审稿：芋艿

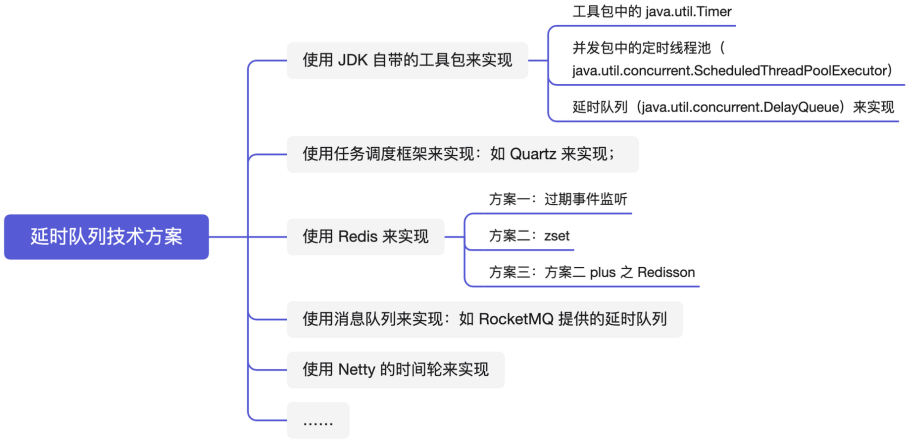
1、业务场景

延时消息来源于早期计算机对异步处理的需求，随着互联网的发展，越来越多的业务场景需要使用延队列。比如：

- 任务调度：延时队列可以用于任务调度，将需要在未来某个特定时刻执行的任务放入队列中。
- 消息延迟处理：延时队列可以用于消息系统，其中一些消息需要在一段时间后才能被消费。例如，在需要进行消息重试的情况下，可以将消息放入延时队列，并在一段时间后重新处理。
- 订单处理：在电商系统中，有时需要对订单进行一些处理，例如取消未支付的订单或处理退货请求。
- 会员到期提醒：如果你有一个会员制度，需要提醒会员其订阅即将到期，延时队列可以用于安排在到期日期前发送提醒通知。
- 缓解高峰期负载：在高峰期，系统可能会面临大量的请求和负载。延时队列可以用于将一些请求推迟处理，进行削峰处理。

2、技术方案

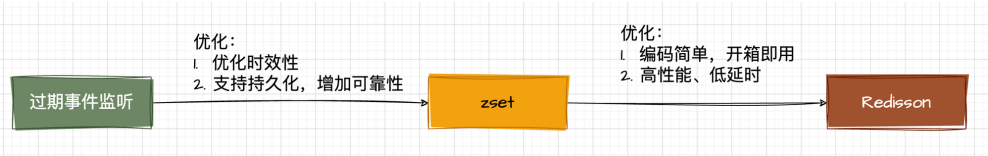
随着技术的进步，实现的方式也愈来愈多种多样，以 Java 语言为例，实现的方式也是多如牛毛：



本文主要探讨 Redis 实现延时消息的几种方式，如前文所述，使用 Redis 来实现延时消息的实现方式主要有 3 种：

1. 过期事件监听
2. Redis zset（有序集合）
3. Redisson

得益于 Redis 自身设计的优点，使用 Redis 完全可以支撑高性能的要求。
从可靠性和使用便利性上来考虑，三种方案优先级排序：Redisson > zset > 过期事件监听：



3、方案一：过期事件监听

3.1 核心设计与原理

Redis 的过期事件监听是基于 pub/sub 的，key 过期时会 pub 消息到一个内置的 channel 中，客户端可以通过监听这个 channel 获取到消息，进而实现延时队列的功能。

对 Redis pub/sub 不了解的同学，可以阅读 https://redisbook.readthedocs.io/en/latest/feature/pubsub.html 文档。

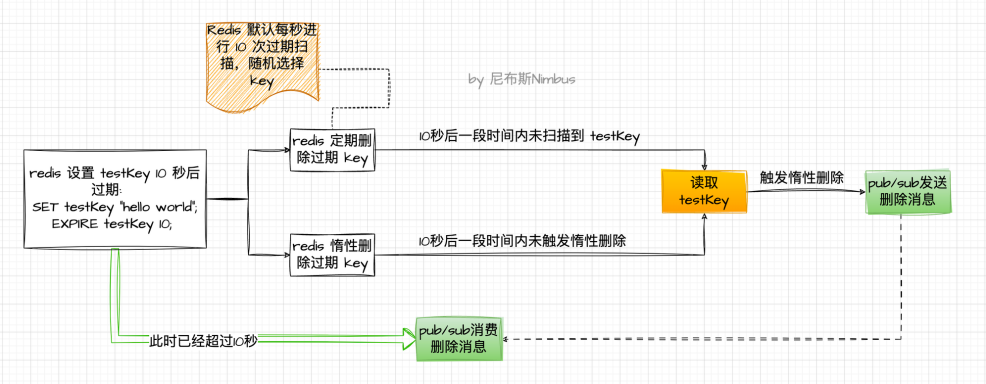
3.2 方案缺陷

① 缺陷 1：不支持持久化，可靠性低
pub/sub 模式下的过期事件监听，消息并不会做持久化，会有消息丢失的风险。

② 缺陷 2：不保证及时性
过期事件监听的方案听起来很好理解，也似乎很完美，但事实并非如此。主要是借助 Redis 删除过期 key 的消息监听，不保证及时性。

Redis 的单线程设计，如果需要支持定时过期，在 Redis 高负载的情况下或者有大量过期键需要同时处理时，会造成 Redis 服务器卡顿，影响主业务执行。
考虑到可用性，Redis 的单线程机制并不能很好地支持定时删除过期 key 的场景，所以 Redis 使用惰性删除 + 定期删除key 的方式。
也正因为 Redis key 的删除不是过期即删除的，所以 key 删除时发送消息的时间也不一定是 key 的过期时间。

失效场景可以参考下图：



另外 Redis 官方 Doc 有一篇文档详细《Monitor changes to Redis keys and values in real time》（https://redis.io/docs/manual/keyspace-notifications/#timing-of-expired-events），阐述了实时监听 Redis k-v 变化的使用，包括如何开启监听配置、哪些命令可以监听到、Redis 6.0 与 Redis 7.0 的命令补充、有什么坑……

4、方案二：zset

Redis 提供了有序集合 zset，我们也可以利用 zset 封装延迟消息。

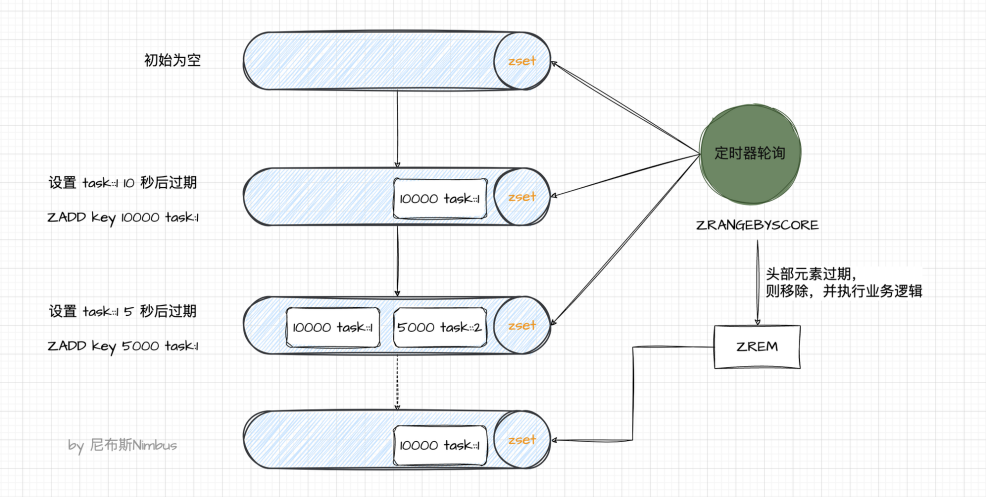
zset 的常用命令及解释：

```
ZADD key score member [score member ...]：向有序集合 key 中添加一个或多个成员，每个成员都带有一个分值 score

ZRANGEBYSCORE key min max [WITHSCORES]：返回有序集合 key 中分值介于 min 和 max 之间的成员。可选的 WITHSCORES 参数表示同时返回成员和分值

ZREM key member [member ...]：从有序集合 key 中移除一个或多个成员
```

对 Redis zset 不了解的同学，可以阅读 http://redisdoc.com/sorted_set/ 文档。

4.1 核心设计与原理

核心设计为使用 zset + 定时轮询器，基于 zset 的 ZRANGEBYSCORE 命令获取已过期的延时任务，流程如下图所示：



4.2 方案缺陷

直接使用 zset 也存在一些弊端~

① 缺陷 1：额外的资源消耗
使用有序集合作为延时队列，并且需要定期地检查有序集合中的任务是否需要被处理，会占用CPU资源

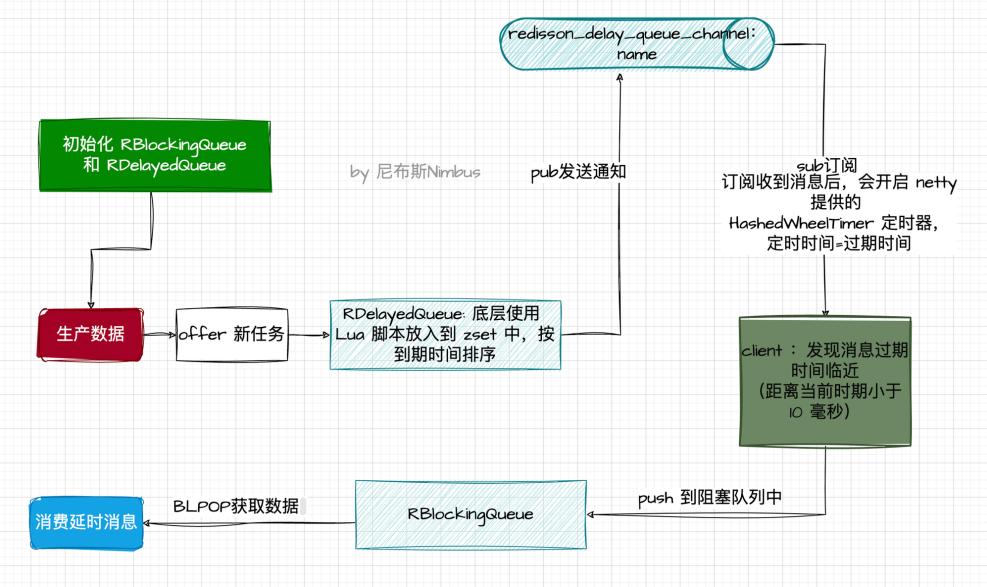
② 缺陷 2：使用上不够友好
需要自行封装，增加编码，在维护上增加了许多成本

5、方案三：Redisson

基于 4.2 小节的考虑，我们可以考虑使用 Redisson 提供的延时队列，它也是基于 zset 实现。

5.1 核心设计与原理

Redisson 封装的延时队列源码还是蛮多的，限于篇幅，之后看情况再补充源码分析篇。本文只讲解源码核心链路的封装。
Redisson 封装了两个核心队列：RBlockingQueue 和 RDelayedQueue，其中 RDelayedQueue 作为中间队列，RBlockingQueue 作为目标消费队列。



5.2 方案优点

1. 简单易用：可以通过注册监听的方式获取延时消息。
2. 批量处理，低延迟：Redisson 延时队列使用定时任务进行批量处理，而不是每个任务都单独处理。这种批处理方式减少了与Redis的通信次数，提高了处理效率。
3. 异步处理：Redisson的延时队列支持异步任务处理，减少了等待时间和阻塞，提升了整体的吞吐量和响应性能。
4. 支持分布式：Redisson的延时队列可以在分布式环境中使用，并提供了分布式锁和协调机制，确保多个节点或实例之间的任务处理的一致性和可靠性。
5. 延迟更低：Redisson 底层使用了 HashedWheelTimer，基于时间轮算法，具有高性能、较高精确度、内存友好、并发安全的特点。

6、编码练习 for 面试

1. 手撕代码：用 Java 实现一个本地延时队列
2. 模仿 Redisson 的 RDelayedQueue 实现一个延时队列

7、扩展资料

- Redisson 源码：https://github.com/redisson/redisson/blob/master/redisson/src/main/java/org/redisson/api/RDelayedQueue.java
- 京东技术（定时任务原理方案综述）：https://mp.weixin.qq.com/s/u6EFPVql4IuoG9-NJLDhsA
- vivo 互联网技术（时间轮原理及其在框架中的应用）：https://mp.weixin.qq.com/s/ZmfvujXMXUe9O0PjXuq-uA
- 有赞延迟队列设计：http://tech.youzan.com/queuing_delay/
- 基于 Redis 实现延时任务：https://juejin.cn/post/6844903817713025032