聊一聊定时任务重复执行以及解决方案

大家好，我是小趴菜，关于定时任务大家都有接触过，项目中肯定也使用过，只需要在项目中的启动类上加上 @EnableScheduling 注解就可以实现了

现在是单个节点部署，倒是没什么问题。如果是多节点部署呢?

假设现在我们系统要每天给用户新增10积分，那么更新的SQL如下

```
update user set point = point + 10 where id = 1;
```

这时候你的服务部署了两台，那么每一台都会来执行这条更新语句，那么用户的积分就不是+10了。而是+20。

当然这里我们只是举了个例子来引出 @EnableScheduling 的定时任务会存在定时任务重复执行的问题。而且有可能会因为重复执行导致数据不一致的问题

使用数据库乐观锁

在使用乐观锁的时候，首先我们会新增一个字段，也就是**更新日期**，但是这个更新日期不是指我们修改数据的那个**更新时间**,比如说今天是2024-03-25,那么到了明天,第一台机器更新成功了，这个值就更新成2024-03-26，其它机器的线程来更新判断这个值是否是2024-03-25,如果不是，说明已经有线程更新了，那么就不需要再次更新了

```
update user set point = point + 10,modifyTime = 2023-03-26 where id = 1 and modifyTime = 2024-03-25
```

基于乐观锁的方式有什么缺点呢？？

现在我们只有两台服务器，那如果有1千台，1万台呢，对于同一条数据的，那么这1万台服务器都会去执行更新操作，但是其实在这1万次更新操作中，只有一次操作是成功的，其余的操作都是不需要执行的

所以使用这种方式当机器数量很多的时候，对数据库的压力是非常大的

分布式锁

我们还可以使用分布式锁的方式来实现，比如要执行这个定时任务之前要先获取一把锁，这个锁是对每一条记录都分别有对应的一把锁

当线程来更新某一条数据的时候，首先要获取这条记录的一个分布式锁，拿到锁了就可以去更新了，没有拿到锁的也不要去等待获取锁了，就直接更新下一条数据即可，同样的步骤，只有拿到某条数据的锁，才可以更新

但是这里有一个注意的点，就是比如**服务-1**先获取到id=100的这条记录的锁，然后执行更新，但是此时因为某些原因，导致某台服务器过了一会才来执行id=100的这条数据，因为服务-1已经执行完了，所以会释放掉这把锁，所以这台服务器来就能获取到锁，那么也会执行更新操作

所以在更新的时候，还是做一下判断，判断这条记录是否已经被更新了，如果已经更新了，那么就不要再次更新了

分布式锁相对于乐观锁来说，减少了大量的无用的更新操作，但还是会存在极少量的重复更新操作，但是相对来说，对数据库的压力就减少了很多

但是与此同时，这就依赖于Redis，要保证Redis的服务可用

消息队列

我们可以将要更新的数据提前加载到消息队列中去，然后每台服务就是一个消费者，保证一条记录只能让一个消费者消费，这样也就可以避免重复更新的问题了

但是消费失败的记录不要重回队列，可以在数据库记录，让人工进行处理

使用消息队列会有什么问题呢?

如果你的消息数据特别多，比如有上亿条，那么消息队列就会被堆满，，而且每天都要把数据库的数据都加载到消息队列中去

或许有人说，数据量大我可以多弄几个消息队列，这样确实可以解决一个消息队列堆积消息过多的问题，但是你要如何控制某些服务器只访问某个队列呢？不能每台服务都循环获取每一个队列中的消息吧

而且如果你的消息队列是单点的，那么服务宕机那么所有数据都没法更新了，这时候你还要去弄一个集群
，这成本就有点大了

所以不推荐使用这种方式

分布式任务调度-xxl-job

最后就是使用分布式定时任务调度框架 xxl-job了，关于xxl-job的使用大家可以网上自己搜一下资料。

XXL-JOB是一个开源的分布式任务调度框架，它主要用于解决大规模分布式任务的调度和执行问题。该框架提供了[任务调度中心**](http://cxyroad.com/”https://www.baidu.com/s?wd=%E4%BB%BB%E5%8A%A1%E8%B0%83%E5%BA%A6%E4%B8%AD%E5%BF%83&rsv_idx=2&tn=baiduhome_pg&usm=2&ie=utf-8&rsv_pq=d0981ad3003b606b&oq=XXL-JOB%E6%98%AF%E4%BB%80%E4%B9%88&rsv_t=3753aOTyGmHQroRTQws2q8VT%2FqsbtqjaNGb6RqZlZYludjXanWUg7m%2BjHQudtWdAK7Ri&sa=re_dqa_zy&icon=1”)、[执行器**](http://cxyroad.com/

”https://www.baidu.com/s?wd=%E6%89%A7%E8%A1%8C%E5%99%A8&rsv_idx=2&tn=baiduhome_pg&usm=2&ie=utf-8&rsv_pq=d0981ad3003b606b&oq=XXL-JOB%E6%98%AF%E4%BB%80%E4%B9%88&rsv_t=f056i0MzbOODCLuSDhRGbNHQ%2Bap0Hsqbf4koEW44Ye7xp5ie9eH0mmMv%2FVZmlaVB%2Bhiz&sa=re_dqa_zy&icon=1”)和[任务日志**](http://cxyroad.com/”https://www.baidu.com/s?wd=%E4%BB%BB%E5%8A%A1%E6%97%A5%E5%BF%97&rsv_idx=2&tn=baiduhome_pg&usm=2&ie=utf-8&rsv_pq=d0981ad3003b606b&oq=XXL-JOB%E6%98%AF%E4%BB%80%E4%B9%88&rsv_t=f056i0MzbOODCLuSDhRGbNHQ%2Bap0Hsqbf4koEW44Ye7xp5ie9eH0mmMv%2FVZmlaVB%2Bhiz&sa=re_dqa_zy&icon=1”)等组件，支持任务的定时调度、动态添加和删除、执行情况监控和日志记录等功能。

总结

--

以上就是为大家提供的定时任务重复执行的解决方案，大家可以根据自己的实际情况来选择不同的方案来实现

原文链接: https://juejin.cn/post/7350167062364979226