链滴

利用redis实现分布式锁

作者：guobing

原文链接：https://ld246.com/article/1472188407659

来源网站：链滴

利用redis来实现分布式锁。一般就用setnx和getset两个命令。
>* `NX`是`Not eXists`的缩写，如`SETNX`命令就应该理解为：`SET if Not eXists`.
>* `getset`是同步的

java之jedis实现
`expireMsecs` 锁持有超时，防止线程在入锁以后，无限的执行下去，让锁无法释放
`timeoutMsecs` 锁等待超时，防止线程饥饿，永远没有入锁执行代码的机会

代码如下：
```
private int timeoutMsecs
    private String lockKey
    private static long expireMsecs = 1000 * 60 * 5  // min 锁持有超时

    // timeoutMsecs 表示锁等待超时
    public JedisLock(Integer timeoutMsecs, String lockKey) {
        this.timeoutMsecs = timeoutMsecs
        this.lockKey = lockKey
    }

    public synchronized boolean acquire(Jedis jedis) throws InterruptedException {
        int timeout = timeoutMsecs

        while (timeout >= 0) {
            long expires = System.currentTimeMillis() + expireMsecs + 1
            String expiresStr = String.valueOf(expires) //锁到期时间

            if (jedis.setnx(lockKey, expiresStr) == 1) {
                return true
            }

            String currentValueStr = jedis.get(lockKey); //redis里的时间
            // 表示已经锁失效，要重新设置锁
            if (currentValueStr != null && Long.parseLong(currentValueStr) < System.currentTim
Millis()) {
                //判断是否为空，不为空的情况下，如果被其他线程设置了值，则第二个条件判断是过不
的
                // lock is expired

                String oldValueStr = jedis.getSet(lockKey, expiresStr)
                //获取上一个锁到期时间，并设置现在的锁到期时间，
                //只有一个线程才能获取上一个线上的设置时间，因为jedis.getSet是同步的
                if (oldValueStr != null && oldValueStr.equals(currentValueStr)) {
                    //如过这个时候，多个线程恰好都到了这里，但是只有一个线程的设置值和当前值相同
他才有权利获取锁
                    return true
                }
            }

            timeout -= 100
            Thread.sleep(100)
        }
        return false
    }
```

```

所以大致的思路就是
>* 先定义好一个锁等待时间和锁超时时间，我们使用中分别设置了300ms和5min，
>* 线程执行进方法后，设置锁到期时间为当前时间加5分钟，就是`System.currentTimeMillis() + 100*60*5 + 1`
>* 执行setnx，执行完返回1就是获取锁成功。返回0时，表示未获取到锁，执行下面一步
>* 判断是否已经锁到期，`currentValueStr`和当前时间比较，如果未过期，锁等待时间减100ms，sleep一下继续重复这个动作。如果已经到期，执行下面一步。
>* 通过`getSet`把redis中的锁超时时间设为自己的时间（因此自己要获取锁），这个方法是同步的保证只能有个线程能拿到锁。执行这步返回的值是旧值，
>* 判断旧值和之前的锁超时的值是否一致。一致则获取锁成功。失败的话锁等待时间减1，sleep一继续重复之前的逻辑。

再总结一下：
> 用最简单的办法就是，多个进程执行以下Redis命令：SETNX lock.foo
如果 SETNX 返回1，说明该进程获得锁，SETNX将键 lock.foo 的值设置为锁的超时时间（当前时间 + 锁的有效时间）。
如果 SETNX 返回0，说明其他进程已经获得了锁，进程不能进入临界区。进程可以在一个循环中不断尝试 SETNX 操作，以获得锁。

但是这里有个问题，如何解决死锁，
> 考虑一种情况，如果进程获得锁后，断开了与 Redis 的连接（可能是进程挂掉，或者网络中断），果没有有效的释放锁的机制，那么其他进程都会处于一直等待的状态，即出现“死锁”。

锁超时时，我们不能简单地使用 DEL 命令删除键 lock.foo 以释放锁。因为这样可能同时会有多个线获得锁。
比如就是p2，p3两个进程同时发现锁已经超时。p2执行删除操作，添加自己的key，返回1，获取锁功。接着p3又删除p2设置的值，自己获取锁。这样子显然是不行的。

为了解决上述算法可能出现的多个进程同时获得锁的问题，我们再来看以下的算法。
> 超时时我们可以这样处理。比如p4,p5发现超时之后，通过getset去更新key值，这个是同步的，能证线程安全。然后根据getset返回的值是否大于当前时间来判断获取锁是否成功。如果小于当前时间那获取成功。如果大于当前时间，说明已经有线程修改过了。再继续等待。
```