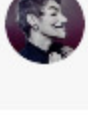


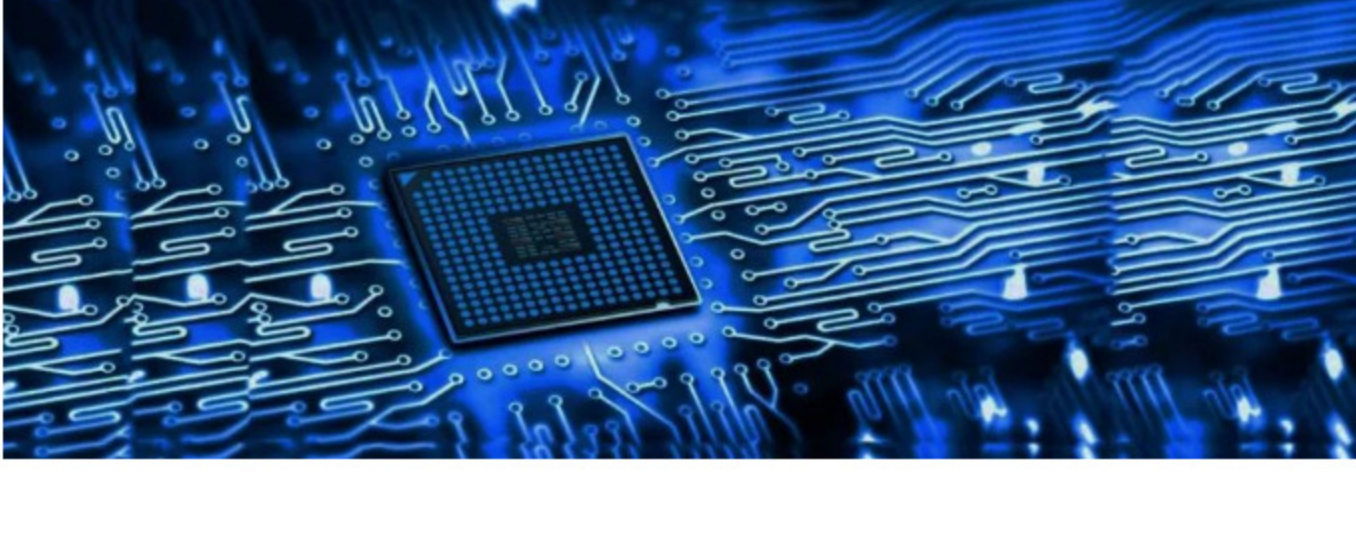
面试官：你会用Redisson实现分布式锁么？

程序员面试 5月23日

以下文章来源于冰河技术，作者冰河团队



冰河技术
分享各种编程语言、开发技术、分布式与微服务架构、分布式数据库、分布式事务、云...



写在前面

Redisson框架十分强大，基于Redisson框架可以实现几乎你能想到的所有类型的分布式锁。这里，我就列举几个类型的分布式锁，并各自给出一个示例程序来加深大家的理解。有关分布式锁的原理细节，后续专门攒一篇文章咱们慢慢聊！

1.可重入锁（Reentrant Lock）

Redisson的分布式可重入锁RLock Java对象实现了java.util.concurrent.locks.Lock接口，同时还支持自动过期解锁。

```
public void testReentrantLock(RedissonClient redisson){
    RLock lock = redisson.getLock("anyLock");
    try{
        // 1. 最常见的使用方法
        //lock.lock();
        // 2. 支持过期解锁功能,10秒钟以后自动解锁，无需调用unlock方法手动解锁
        //lock.lock(10, TimeUnit.SECONDS);
        // 3. 尝试加锁，最多等待3秒，上锁以后10秒自动解锁
        boolean res = lock.tryLock(3, 10, TimeUnit.SECONDS);
        if(res){ //成功
            // do your business
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    } finally {
        lock.unlock();
    }
}
```

Redisson同时还为分布式锁提供了异步执行的相关方法：

```
public void testAsyncReentrantLock(RedissonClient redisson){
    RLock lock = redisson.getLock("anyLock");
    try{
        lock.lockAsync();
        lock.lockAsync(10, TimeUnit.SECONDS);
        Future<Boolean> res = lock.tryLockAsync(3, 10, TimeUnit.SECONDS);
        if(res.get()){
            // do your business
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (ExecutionException e) {
        e.printStackTrace();
    } finally {
        lock.unlock();
    }
}
```

2.公平锁（Fair Lock）

Redisson分布式可重入公平锁也是实现了java.util.concurrent.locks.Lock接口的一种RLock对象。在提供了自动过期解锁功能的同时，保证了当多个Redisson客户端线程同时请求加锁时，优先分配给先发出请求的线程。

```
public void testFairLock(RedissonClient redisson){
    RLock fairLock = redisson.getFairLock("anyLock");
    try{
        // 最常见的使用方法
        fairLock.lock();
        // 支持过期解锁功能，10秒钟以后自动解锁，无需调用unlock方法手动解锁
        fairLock.lock(10, TimeUnit.SECONDS);
        // 尝试加锁，最多等待100秒，上锁以后10秒自动解锁
        boolean res = fairLock.tryLock(100, 10, TimeUnit.SECONDS);
    } catch (InterruptedException e) {
        e.printStackTrace();
    } finally {
        fairLock.unlock();
    }
}
```

Redisson同时还为分布式可重入公平锁提供了异步执行的相关方法：

```
RLock fairLock = redisson.getFairLock("anyLock");
fairLock.lockAsync();
fairLock.lockAsync(10, TimeUnit.SECONDS);
Future<Boolean> res = fairLock.tryLockAsync(100, 10, TimeUnit.SECONDS);
```

3.联锁（MultiLock）

Redisson的RedissonMultiLock对象可以将多个RLock对象关联为一个联锁，每个RLock对象实例可以来自于不同的Redisson实例。

```
public void testMultiLock(RedissonClient redisson1,RedissonClient redisson2, RedissonClient redisson3){
    RLock lock1 = redisson1.getLock("lock1");
    RLock lock2 = redisson2.getLock("lock2");
    RLock lock3 = redisson3.getLock("lock3");
    RedissonMultiLock lock = new RedissonMultiLock(lock1, lock2, lock3);
    try {
        // 同时加锁：lock1 lock2 lock3，所有的锁都上锁成功才算成功。
        lock.lock();
        // 尝试加锁，最多等待100秒，上锁以后10秒自动解锁
        boolean res = lock.tryLock(100, 10, TimeUnit.SECONDS);
    } catch (InterruptedException e) {
        e.printStackTrace();
    } finally {
        lock.unlock();
    }
}
```

4.红锁（RedLock）

Redisson的RedissonRedLock对象实现了Redlock介绍的加锁算法。该对象也可以用来将多个RLock对象关联为一个红锁，每个RLock对象实例可以来自于不同的Redisson实例。

```
public void testRedLock(RedissonClient redisson1,RedissonClient redisson2, RedissonClient redisson3){
    RLock lock1 = redisson1.getLock("lock1");
    RLock lock2 = redisson2.getLock("lock2");
    RLock lock3 = redisson3.getLock("lock3");
    RedissonRedLock lock = new RedissonRedLock(lock1, lock2, lock3);
    try {
        // 同时加锁：lock1 lock2 lock3，红锁在大部分节点上加锁成功就算成功。
        lock.lock();
        // 尝试加锁，最多等待100秒，上锁以后10秒自动解锁
        boolean res = lock.tryLock(100, 10, TimeUnit.SECONDS);
    } catch (InterruptedException e) {
        e.printStackTrace();
    } finally {
        lock.unlock();
    }
}
```

5.读写锁（ReadWriteLock）

Redisson 的 分 布 式 可 重 入 读 写 锁 RReadWriteLock,Java 对 象 实 现 了 java.util.concurrent.locks.ReadWriteLock接口。同时还支持自动过期解锁。该对象允许同时有多个读取锁，但是最多只能有一个写入锁。

```
RReadWriteLock rwlock = redisson.getLock("anyRWLock");
// 最常见的使用方法
rwlock.readLock().lock();
// 或
rwlock.writeLock().lock();
// 支持过期解锁功能
// 10秒钟以后自动解锁
// 无需调用unlock方法手动解锁
rwlock.readLock().lock(10, TimeUnit.SECONDS);
// 或
rwlock.writeLock().lock(10, TimeUnit.SECONDS);
// 尝试加锁，最多等待100秒，上锁以后10秒自动解锁
boolean res = rwlock.readLock().tryLock(100, 10, TimeUnit.SECONDS);
// 或
boolean res = rwlock.writeLock().tryLock(100, 10, TimeUnit.SECONDS);
...
lock.unlock();
```

6.信号量（Semaphore）

Redisson 的 分 布 式 信 号 量（Semaphore）Java 对 象 RSemaphore 采 用 了 与 java.util.concurrent.Semaphore相似的接口和用法。

```
RSemaphore semaphore = redisson.getSemaphore("semaphore");
semaphore.acquire();
//或
semaphore.acquireAsync();
semaphore.acquire(23);
semaphore.tryAcquire();
//或
semaphore.tryAcquireAsync();
semaphore.tryAcquire(23, TimeUnit.SECONDS);
//或
semaphore.tryAcquireAsync(23, TimeUnit.SECONDS);
semaphore.release(10);
semaphore.release();
//或
semaphore.releaseAsync();
```

7.可过期性信号量（PermitExpirableSemaphore）

Redisson的可过期性信号量（PermitExpirableSemaphore）实在RSemaphore对象的基础上，为每个信号增加了一个过期时间。每个信号可以通过独立的ID来辨识，释放时只能通过提交这个ID才能释放。

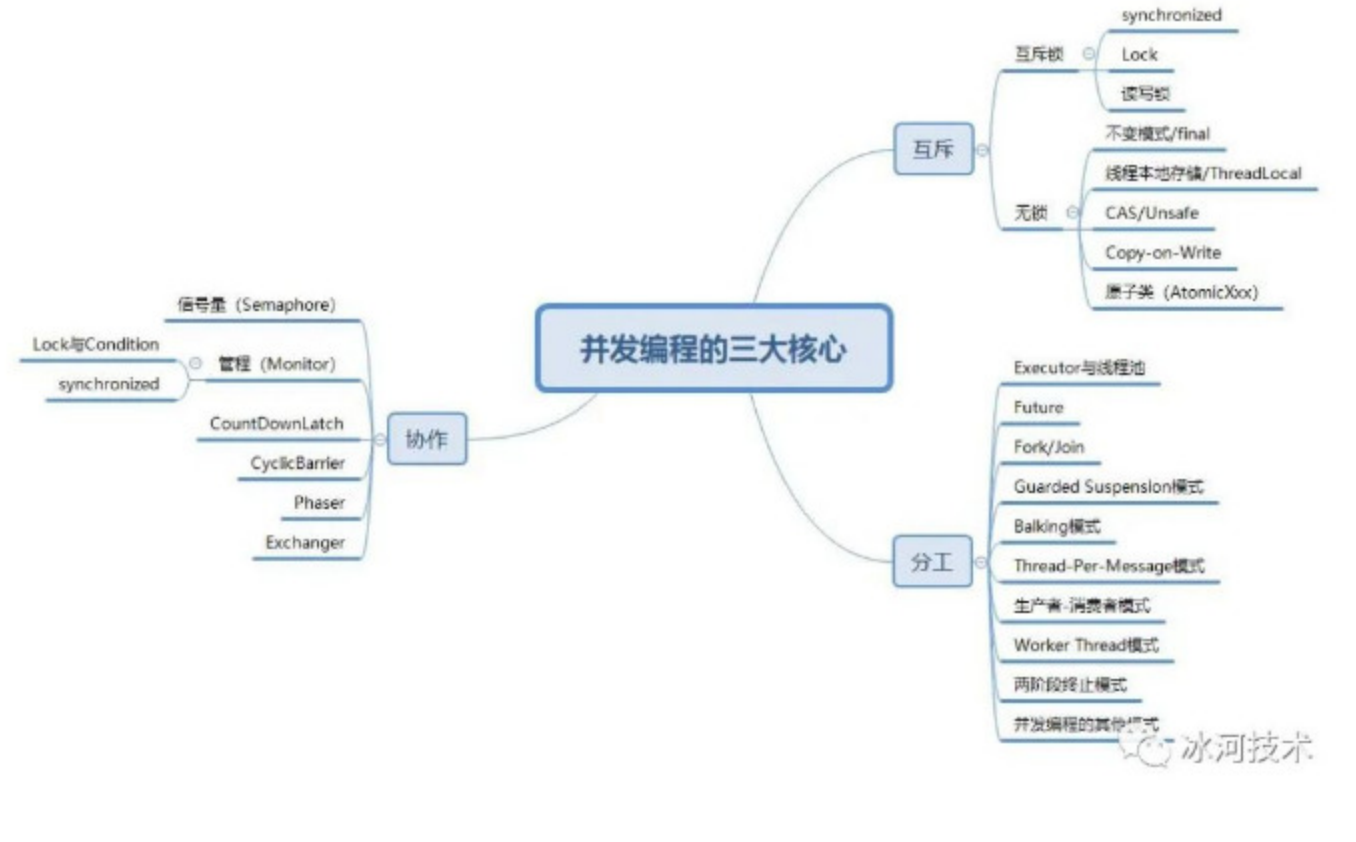
```
RPermitExpirableSemaphore semaphore = redisson.getPermitExpirableSemaphore("mySemaphore");
String permitId = semaphore.acquire();
// 获取一个信号，有效期只有2秒钟。
String permitId = semaphore.acquire(2, TimeUnit.SECONDS);
// ...
semaphore.release(permitId);
```

8.闭锁（CountDownLatch）

Redisson 的 分 布 式 闭 锁（CountDownLatch）Java 对 象 RCountDownLatch 采 用 了 与 java.util.concurrent.CountDownLatch相似的接口和用法。

```
RCountDownLatch latch = redisson.getCountDownLatch("anyCountDownLatch");
latch.trySetCount(1);
latch.await();
// 在其他线程或其他JVM里
RCountDownLatch latch = redisson.getCountDownLatch("anyCountDownLatch");
latch.countDown();
```

最后，附上并发编程需要掌握的核心技能知识图，祝大家在学习并发编程时，少走弯路。



长按订阅更多面经分享

