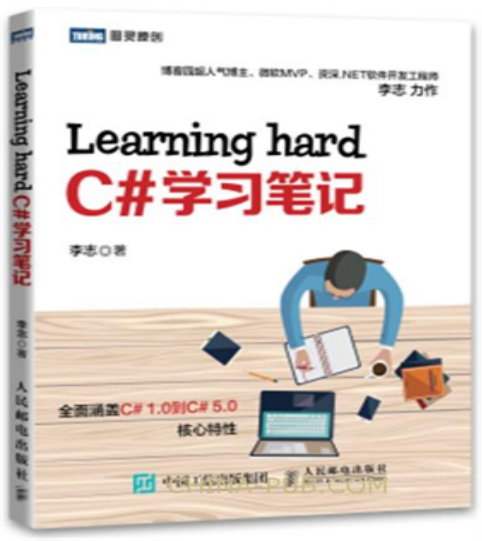
《Learning hard C#学习笔记》



[1] 互动网
[2] 京东
[3] 当当
[4]亚马逊
[5]天猫

昵称： Learning hard
园龄： 7年11个月
荣誉： 推荐博客
粉丝： 3740
关注： 170
+加关注

我的标签

DDD(9)
WPF(8)
Redis(4)
SignalR(4)
KnockoutJs(3)
Asp.net MVC(3)
Bootstrap(3)
分布式(3)
分布式缓存(3)
领域驱动设计(3)
更多

积分与排名

积分 - 510453
排名 - 462

随笔分类 (180)

.NET领域驱动设计系列(12)
Asp.net(5)
Asp.net MVC(8)
Asp.net Web API
C# 互操作入门系列(4)
C#多线程处理系列(7)

【分布式缓存系列】集群环境下Redis分布式锁的正确姿势

一、前言

在上一篇文章中，已经介绍了基于Redis实现分布式锁的正确姿势，但是上篇文章存在一定的缺陷——它加锁只作用在一个Redis节点上，如果通过sentinel保证高可用，如果master节点由于某些原因发生了主从切换，那么就会出现锁丢失的情况：

1. 客户端1在Redis的master节点上拿到了锁
2. Master宕机了，存储锁的key还没有来得及同步到Slave上
3. master故障，发生故障转移，slave节点升级为master节点
4. 客户端2从新的Master获取到了对应同一个资源的锁

于是，客户端1和客户端2同时持有了同一个资源的锁。锁的安全性被打破了。针对这个问题。Redis作者antirez提出了RedLock算法来解决这个问题

二、RedLock算法的实现思路

antirez提出的redlock算法实现思路大概是这样的。

客户端按照下面的步骤来获取锁：

1. 获取当前时间的毫秒数T1。
2. 按顺序依次向N个Redis节点执行获取锁的操作。这个获取锁的操作和上一篇中基于单Redis节点获取锁的过程相同。包括唯一UUID作为Value以及锁的过期时间(expireTime)。为了保证在某个在某个Redis节点不可用的时候算法能够继续运行，这个获取锁的操作还需要一个超时时间。它应该远小于锁的过期时间。客户端向某个Redis节点获取锁失败后，应立即尝试下一个Redis节点。这里失败包括Redis节点不可用或者该Redis节点上的锁已经被其他客户端持有。
3. 计算整个获取锁过程的总耗时。即当前时间减去第一步记录的时间。计算公司为T2=now()- T1。如果客户端从大多数Redis节点(>N/2 +1)成功获取到锁。并且获取锁总共消耗的时间小于锁的过期时间（即T2<expireTime）。则认为客户端获取锁成功，否则，认为获取锁失败
4. 如果获取锁成功，需要重新计算锁的过期时间。它等于最初锁的有效时间减去第三步计算出来获取锁消耗的时间，即expireTime - T2
5. 如果最终获取锁失败，那么客户端立即向所有Redis系欸但发起释放锁的操作。（和上一篇释放锁的逻辑一样）

虽然说RedLock算法可以解决单点Redis分布式锁的安全性问题，但如果集群中有节点发生崩溃重启，还是会锁的安全性有影响的。具体出现问题的场景如下：

假设一共有5个Redis节点：A, B, C, D, E。设想发生了如下的事件序列：

1. 客户端1成功锁住了A, B, C，**获取锁**成功（但D和E没有锁住）
2. 节点C崩溃重启了，但客户端1在C上加的锁没有持久化下来，丢失了
3. 节点C重启后，客户端2锁住了C, D, E，**获取锁**成功

这样，客户端1和客户端2同时获得了锁（针对同一资源）。针对这样场景，解决方式也很简单，也就是让Redis崩溃后延迟重启，并且这个延迟时间大于锁的过期时间就好。这样等节点重启后，所有节点上的锁都已经失效了。也不存在以上出现2个客户端获取同一个资源的情况了。

相比之下，RedLock安全性和稳定性都比前一篇文章中介绍的实现要好很多，但要说完全没有问题不是。例如，如果客户端获取锁成功后，如果访问共享资源操作执行时间过长，导致锁过期了，后续客户端获取锁成功了，这样在同一个时刻又出现了2个客户端获得了锁的情况。所以针对分布式锁的应用的时候需要多测试。服务器台数越多，出现不可预期的情况也越多。如果客户端获取锁之后，在上面第三步发生了GC得情况导致GC完成后，锁失效了，这样同时也使得同一时间有2个客户端获得了锁。如果系统对共享资源有非常严格要求得情况下，还是建议需要做数据库锁得得方案来补充。如飞机票或火车票座位得情况。对于一些抢购获取，针对偶尔出现超卖，后续可以人为沟通置换得方式采用分布式锁得方式没什么问题。因为可以绝大部分保证分布式锁的安全性。

三、分布式场景下基于Redis实现分布式锁的正确姿势

目前redisson包已经有对redlock算法封装，接下来就具体看看使用redisson包来实现分布式锁的正确姿势。

具体实现代码如下代码所示：

随笔档案 (164)

文章分类 (0)

分布式专题

文章档案 (9)


```
public interface DistributedLock {
    /**
     * 获取锁
     * @author zhi.li
     * @return 锁标识
     */
    String acquire();

    /**
     * 释放锁
     * @author zhi.li
     * @param indentifier
     * @return
     */
    boolean release(String indentifier);
}

public class RedisDistributedRedLock implements DistributedLock {

    /**
     * redis 客户端
     */
    private RedissonClient redissonClient;

    /**
     * 分布式锁的键值
     */
    private String lockKey;

    private RLock redLock;

    /**
     * 锁的有效时间 10s
     */
    int expireTime = 10 * 1000;

    /**
     * 获取锁的超时时间
     */
    int acquireTimeout  = 500;

    public RedisDistributedRedLock(RedissonClient redissonClient, String lockKey) {
        this.redissonClient = redissonClient;
        this.lockKey = lockKey;
    }

    @Override
    public String acquire() {
        redLock = redissonClient.getLock(lockKey);
        boolean isLock;
        try{
            isLock = redLock.tryLock(acquireTimeout, expireTime, TimeUnit.MILLISECONDS);
            if(isLock){
                System.out.println(Thread.currentThread().getName() + " " + lockKey + "获得了
锁");

                return null;
            }
        }catch (Exception e){
            e.printStackTrace();
        }
        return null;
    }

    @Override
    public boolean release(String indentifier) {
        if(null != redLock){
            redLock.unlock();
            return true;
        }

        return false;
    }
}
```

2015年8月(2)
2014年10月(1)
2014年9月(2)
2013年8月(1)
2013年5月(1)

友情链接

个人开发历程知识库
VC 知识库
WCF
SQL Server Study 2
SQL Server Study 1
WPF
WPF 2
WPF 3
Asp.net MVC
SQL Server Study 3
框架设计
框架设计2

最新评论

1. Re:[.NET领域驱动设计实战系列]专题二：结合领域驱动设计的面向服务架构来搭建网上书店
评论里真是八仙过海，各显神通啊
--ChangeTheWorldInCode
2. Re:[C# 网络编程系列]专题十二：实现一个简单的FTP服务器
参考博主的代码，改成了VB.NET，博主的编程思路也加深了我对网络线程的理解，给博主打了赏
--大鹏集成
3. Re:WPF快速入门系列(9)——WPF任务管理工具实现
缺少 config.xml
--jasonlai2016
4. Re:【分布式缓存系列】Redis实现分布式锁的正确姿势
分析的不全，如果锁超时后再去释放锁是不是又会出现问题
--haibiscuit
5. Re:【分布式缓存系列】Redis实现分布式锁的正确姿势
@
我明白了-。- 第一次没有设置成功就自动暂停一秒让他超过锁的等待期，就不会一直去设置锁了
--numbol

阅读排行榜

1. C#设计模式总结(115239)
2. C#设计模式(1)——单例模式(99786)
3. C#设计模式(2)——简单工厂模式(57839)
4. C#设计模式(3)——工厂方法模式(48515)
5. C#设计模式(4)——抽象工厂模式(41919)
6. VSTO之旅系列(一)：VSTO入门(33723)
7. [你必须知道的异步编程]C# 5.0 新特性——Async和Await使异步编程更简单(32114)
8. C#设计模式(5)——建造者模式（Builder Pattern）(31490)
9. WPF快速入门系列(1)——WPF布局概览(30949)
10. [C# 网络编程系列]专题五：TCP编程(27404)
11. C#设计模式(17)——观察者模式（Observer Pattern）(24553)
12. C#设计模式(7)——适配器模式（Ada

由于RedLock是针对主从和集群场景准备。上面代码采用哨兵模式。所以要让上面代码运行起来，需要先本地搭建Redis哨兵模式。本人的环境是Windows,具体Windows 哨兵环境搭建参考文章：redis sentinel部署(Windows下实现)。

具体测试代码如下所示：

```
public class RedisDistributedRedLockTest {
    static int n = 5;
    public static void secskill() {
        if(n <= 0) {
            System.out.println("抢购完成");
            return;
        }

        System.out.println(--n);
    }
    public static void main(String[] args) {

        Config config = new Config();
        //支持单机，主从，哨兵，集群等模式
        //此为哨兵模式
        config.useSentinelServers()
                .setMasterName("mymaster")
                .addSentinelAddress("127.0.0.1:26369","127.0.0.1:26379","127.0.0.1:26389")
                .setDatabase(0);
        Runnable runnable = () -> {
            RedisDistributedRedLock redisDistributedRedLock = null;
            RedissonClient redissonClient = null;
            try {
                redissonClient = Redisson.create(config);
                redisDistributedRedLock = new RedisDistributedRedLock(redissonClient,
"stock_lock");
                redisDistributedRedLock.acquire();
                secskill();
                System.out.println(Thread.currentThread().getName() + "正在运行");
            } finally {
                if (redisDistributedRedLock != null) {
                    redisDistributedRedLock.release(null);
                }

                redissonClient.shutdown();
            }
        };

        for (int i = 0; i < 10; i++) {
            Thread t = new Thread(runnable);
            t.start();
        }
    }
```

具体的运行结果，如下图所示：

评论排行榜

推荐排行榜


```
1
Thread-2正在运行
23:47:53.786 [Thread-2] DEBUG org.redisson.comman
23:47:53.786 [redisson-14-7] DEBUG org.redisson.c
23:47:53.788 [redisson-netty-10-2] DEBUG org.redi
23:47:53.788 [Thread-1] DEBUG org.redisson.comman
23:47:53.791 [Thread-1] DEBUG org.redisson.comman
Thread-1 stock_lock获得了锁
0
Thread-1正在运行
23:47:53.793 [Thread-1] DEBUG org.redisson.comman
23:47:53.793 [redisson-netty-1-5] DEBUG org.redis
23:47:53.795 [Thread-4] DEBUG org.redisson.comman
23:47:53.796 [redisson-netty-3-1] DEBUG org.redis
Thread-4 stock_lock获得了锁
抢购完成
Thread-4正在运行
```

四、总结

到此，基于Redis实现分布式锁的就告一段落了，由于分布式锁的实现方式主要有：数据库锁的方式、基于Redis实现和基于Zookeeper实现。接下来的一篇文章将介绍基于Zookeeper分布式锁的正确姿势。

本文所有代码地址：https://github.com/learninghard-lizhi/common-util

分类: 分布式专题

标签: 分布式, 分布式缓存, Redis

分类: 分布式专题

标签: 分布式, 分布式缓存, Redis

Learning hard
关注 - 170
粉丝 - 3740

推荐博客

+加关注

« 上一篇： 【分布式缓存系列】Redis实现分布式锁的正确姿势

» 下一篇： 大型分布式项目实战视频教程，帮你实现加薪升职

posted @ 2019-01-23 22:30 Learning hard 阅读(3587) 评论(6) 编辑 收藏

评论列表

#1楼 2019-01-24 13:56 学亮

redis分布式锁存在争议，别用啦

支持(0) 反对(0)

#2楼 2019-03-07 13:56 捞月亮的猴子

@ 学亮
大佬这两年很安静呀

支持(0) 反对(0)

#3楼 2019-05-29 03:12 孵蛋的鸵鸟

大佬有木有计划写有关node.js和vue.js的文章呀

支持(0) 反对(0)

#4楼 2019-07-11 20:12 树上的疯子

有人说用setnx 有人说用set 感觉晕了已经

支持(0) 反对(0)

#5楼 2019-10-08 12:55 仍是少年

大佬这两年不耍.net啦

支持(0) 反对(0)

#6楼 [楼主] 2019-10-08 22:33 Learning hard

@ 仍是少年
目前转Java了，可以加我QQ详聊

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

相关博文：
· Redis分布式锁服务(八)
· 分布式缓存技术redis系列（五）——redis实战（redis与spring整合，分布式锁实现）
· redis系列：基于redis的分布式锁
· 【原创】分布式之抉择分布式锁
· jedisLock—redis分布式锁实现
» 更多推荐...

独家下载电子书 | 前端必看！阿里这样实现前端代码智能生成

最新 IT 新闻:
· 12个国家考虑发行央行数字货币
· Ghostcat漏洞影响过去13年发布的所有Apache Tomcat版本
· 阿里拟退出博雅天下，曾运营《人物》《财经天下》《博客天下》等
· 微软 Edge 扩展商店已经拥有超 1000 个扩展
· 为了鼓励人们少开车，这个国家推出全国公交免费政策
» 更多新闻...

历史上的今天:
2013-01-23 C# 互操作性入门系列(三)：平台调用中的数据封送处理