

es/674394?type=2)

技术小能手 (/users/o3u4mvu6rfxm2) 2018-12-03 09:12:43 浏览4942

分布式 (/tags/type\_blog-tagid\_21/) redis (/tags/type\_blog-tagid\_32/) 算法 (/tags/type\_blog-tagid\_37/)

源码 (/tags/type\_blog-tagid\_603/) uuid (/tags/type\_blog-tagid\_2199/)

普通实现

说道Redis分布式锁大部分人都会想到：setnx+lua，或者知道setkey value px milliseconds nx。后一种方式的核心实现命令如下：

service.weibo.com/share/share.php?

edlock%EF%BC%9ARedis%E5%88%86%E5%B8%83%E5%BC%8F%E9%94%81%E6%9C%8C%E5%B7%E5%8F%96%E9%94%81%E5%BC%88unique\_value%E5%8F%AF%E4%BB%A5%E

- 获取锁 (unique\_value可以是UUID等)

SET resource\_name unique\_value NX PX 30000

- 释放锁 (lua脚本中，一定要比较value，防止误解锁)

if redis.call("get",KEYS[1]) == ARGV[1] then


return redis.call("del",KEYS[1])

else

return 0

end

作者介绍



技术小能手 (/users/o3u4...)

7919

文章数

41620

粉丝数

所属云栖号：Java技术驿站 (/teams)

技术领域

数据库

分享40个数据库/脚本/SQL/SQL

下一篇

探访草根云计算大咖的成长历程 (/article

我是你爱豆 (/users/5s5aoldfi7ncs) 5234

这种实现方式有3大要点（也是面试概率非常高的地方）：

- set命令要用 setkey value px milliseconds nx；
- value要具有唯一性；
- 释放锁时要验证value值，不能误解锁；

事实上这类锁最大的缺点就是它加锁时只作用在一个Redis节点上，即使Redis通过sentinel保证高可用，如果这个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：

- 在Redis的master节点上拿到了锁；
- 但是这个加锁的key还没有同步到slave节点；
- master故障，发生故障转移，slave节点升级为master节点；
- 导致锁丢失。

正因为如此，Redis作者antirez基于分布式环境下提出了一种更高级的分布式锁的实现方式：**Redlock**。笔者认为，Redlock也是Redis所有分布式锁实现方式中唯一能让面试官高潮的方式。

## Redlock实现

antirez提出的redlock算法大概是这样的：

本文目录

1

普通实现

2

Redlock实现

3

Redlock源码

4

用法

5

唯一ID

6

获取锁

7

释放锁



在Redis的分布式环境中，我们假设有N个Redis master。这些节点完全互相独立，不存在主从复制或者其他集群协调机制。我们确保将在N个实例上使用与在Redis单实例下相同方法获取和释放锁。现在我们假设有5个Redis master节点，同时我们需要在5台服务器上面运行这些Redis实例，这样保证他们不会同时都宕掉。

为了拿到锁，客户端应该执行以下操作：

- 获取当前Unix时间，以毫秒为单位。
- 依次尝试从5个实例，使用相同的key和具有唯一性的**value**（例如UUID）获取锁。当向Redis请求获取锁时，客户端应该设置一个网络连接和响应超时时间，这个超时时间应该小于锁的失效时间。例如你的锁自动失效时间为10秒，则超时时间应该在5-50毫秒之间。这样可以避免服务器端Redis已经挂掉的情况下，客户端还在死死地等待响应结果。如果服务器端没有在规定时间内响应，客户端应该尽快尝试去另外一个Redis实例请求获取锁。
- 客户端使用当前时间减去开始获取锁时间（步骤1记录的时间）就得到获取锁使用的时间。当且仅当从大多数（ $N/2+1$ ，这里是3个节点）的**Redis**节点都拿到锁，并且使用的时间小于锁失效时间时，锁才算获取成功。
- 如果拿到了锁，key的真正有效时间等于有效时间减去获取锁所使用的时间（步骤3计算的结果）。
- 如果因为某些原因，获取锁失败（没有在至少 $N/2+1$ 个Redis实例拿到锁或者取锁时间已经超过了有效时间），客户端应该在所有的**Redis**实例上进行解锁（即便某些Redis实例根本就没有加锁成功，防止某些节点获取到锁但是客户端没有得到响应而导致接下来的一段时间不能被重新获取锁）。

## Redlock源码

redisson已经有对redlock算法封装，接下来对其用法进行简单介绍，并对核心源码进行分析（假设5个redis实例）。

### POM依赖

```
<!-- https://mvnrepository.com/artifact/org.redisson/redisson -->
<dependency>

<groupId>org.redisson</groupId>

<artifactId>redisson</artifactId>

<version>3.3.2</version>

</dependency>
```

### 用法

首先，我们来看一下redisson封装的redlock算法实现的分布式锁用法，非常简单，跟重入锁（ReentrantLock）有点类似：

```
Config config = new Config();

config.useSentinelServers().addSentinelAddress("127.0.0.1:6369", "127.0.0.1:6379", "127.0.0.1:6389")

.setMasterName("masterName")

.setPassword("password").setDatabase(0);

RedissonClient redissonClient = Redisson.create(config);
```



```
// 还可以getFairLock(), getReadWriteLock()

RLock redLock = redissonClient.getLock("REDLOCK_KEY");

boolean isLock;

try {

    isLock = redLock.tryLock();

    // 500ms拿不到锁, 就认为获取锁失败。10000ms即10s是锁失效时间。

    isLock = redLock.tryLock(500, 10000, TimeUnit.MILLISECONDS);

    if (isLock) {

        //TODO if get lock success, do something;

    }

} catch (Exception e) {

} finally {

    // 无论如何, 最后都要解锁

    redLock.unlock();

}
```

### 唯一ID

实现分布式锁的一个非常重要的点就是set的value要具有唯一性, redisson的value是怎样保证value的唯一性呢? 答案是**UUID+threadId**。入口在redissonClient.getLock("REDLOCK\_KEY"), 源码在Redisson.java和RedissonLock.java中:

```
protected final UUID id = UUID.randomUUID();

String getLockName(long threadId) {

    return id + ":" + threadId;

}
```

### 获取锁

获取锁的代码为redLock.tryLock()或者redLock.tryLock(500, 10000, TimeUnit.MILLISECONDS), 两者的最终核心源码都是下面这段代码, 只不过前者获取锁的默认租约时间 (leaseTime) 是LOCK\_EXPIRATIONINTERVAL\_SECONDS, 即30s:

```
<T> RFuture<T> tryLockInnerAsync(long leaseTime, TimeUnit unit, long threadId, RedisStrictCommand<T> command) {

    internalLockLeaseTime = unit.toMillis(leaseTime);

    // 获取锁时向5个redis实例发送的命令

    return commandExecutor.evalWriteAsync(getName(), LongCodec.INSTANCE, command,

        // 首先分布式锁的KEY不能存在, 如果确实不存在, 那么执行hset命令 (hset REDLOCK_KEY uuid+threadId 1), 并通过pexpire

        "if (redis.call('exists', KEYS[1]) == 0) then " +

        "redis.call('hset', KEYS[1], ARGV[2], 1); " +
```



```

"redis.call('pexpire', KEYS[1], ARGV[1]);" +

"return nil;" +

"end;" +

// 如果分布式锁的KEY已经存在，并且value也匹配，表示是当前线程持有的锁，那么重入次数加1，并且设置失效时间

"if (redis.call('hexists', KEYS[1], ARGV[2]) == 1) then" +

"redis.call('hincrby', KEYS[1], ARGV[2], 1);" +

"redis.call('pexpire', KEYS[1], ARGV[1]);" +

"return nil;" +

"end;" +

// 获取分布式锁的KEY的失效时间毫秒数

"return redis.call('pttl', KEYS[1]);",

// 这三个参数分别对应KEYS[1]，ARGV[1]和ARGV[2]

Collections.<Object>singletonList(getName()), internalLockLeaseTime, getLockName(threadId));

}

```

获取锁的命令中，

- **KEYS[1]** 就是Collections.singletonList(getName())，表示分布式锁的key，即REDL  
OCK\_KEY；
- **ARGV[1]** 就是internalLockLeaseTime，即锁的租约时间，默认30s；
- **ARGV[2]** 就是getLockName(threadId)，是获取锁时set的唯一值，即UUID+threadI  
d：

释放锁

释放锁的代码为redLock.unlock()，核心源码如下：

```

protected RFuture<Boolean> unlockInnerAsync(long threadId) {

// 向5个redis实例都执行如下命令

return commandExecutor.evalWriteAsync(getName(), LongCodec.INSTANCE, RedisCommands.EVAL_BOOLEAN,

// 如果分布式锁KEY不存在，那么向channel发布一条消息

"if (redis.call('exists', KEYS[1]) == 0) then" +

"redis.call('publish', KEYS[2], ARGV[1]);" +

"return 1;" +

"end;" +

// 如果分布式锁存在，但是value不匹配，表示锁已经被占用，那么直接返回

"if (redis.call('hexists', KEYS[1], ARGV[3]) == 0) then" +

"return nil;" +

"end;" +

```



```
// 如果就是当前线程占有分布式锁，那么将重入次数减1

"local counter = redis.call('hincrby', KEYS[1], ARGV[3], -1); " +

// 重入次数减1后的值如果大于0，表示分布式锁有重入过，那么只设置失效时间，还不能删除

"if (counter > 0) then " +

"redis.call('pexpire', KEYS[1], ARGV[2]); " +

"return 0; " +

"else " +

// 重入次数减1后的值如果为0，表示分布式锁只获取过1次，那么删除这个KEY，并发布解锁消息

"redis.call('del', KEYS[1]); " +

"redis.call('publish', KEYS[2], ARGV[1]); " +

"return 1; "+

"end; " +

"return nil;";

// 这5个参数分别对应KEYS[1]，KEYS[2]，ARGV[1]，ARGV[2]和ARGV[3]

Arrays.<Object>asList(getName(), getChannelName()), LockPubSub.unlockMessage, internalLockLeaseTime, getLockName(thr

}

}
```

参考：<https://redis.io/topics/distlock>

原文发布时间为： 2018-12-02

本文作者：阿飞的博客

本文来自云栖社区合作伙伴“Java技术驿站 (/go/articleRenderRedirect?url=https%3A%2F%2Fmp.weixin.qq.com%2Fs%2FalsSIkQnIOEXw-q4HQ3ew)”，了解相关信息可以关注“Java技术驿站 (/go/articleRenderRedirect?url=https%3A%2F%2Fmp.weixin.qq.com%2Fs%2FalsSIkQnIOEXw-q4HQ3ew)”。

- 如果您发现本社区中有涉嫌抄袭的内容，欢迎发送邮件至：[yqgroup@service.aliyun.com](mailto:yqgroup@service.aliyun.com) 进行举报，并提供相关证据，一经查实，本社区将立刻删除涉嫌侵权内容。

## 网友评论

写下你的评论...

登录 ([https://account.aliyun.com/login/login.htm?from\\_type=yqclub&oauth\\_callback=https%3A%2F%2Fyq.aliyun.com%2Farticles%2F674394%3Fdo%3Dlogin](https://account.aliyun.com/login/login.htm?from_type=yqclub&oauth_callback=https%3A%2F%2Fyq.aliyun.com%2Farticles%2F674394%3Fdo%3Dlogin))后评论

0/500

评论

阿里云优惠券  
(/users/gmgrws4cue6ek)

gydtep (/users/gmgrws4cue6ek)

2018-12-03 13:37:26



阿里云服务器2折起，可免费领取1888元代金券，官方活动网址：<https://promotion.aliyun.com/ntms/yunparter/invite.html?userCode=2a7uv47d> (<https://promotion.aliyun.com/ntms/yunparter/invite.html?userCode=2a7uv47d>) 新用户和老用户都可以用。

👍 0    💬 0



(/users/5i4oeokgoi35y)

1204835698096614 (/users/5i4oeokgoi35y)

2019-04-10 00:21:03

如果集群有50个节点， 那个加锁开销也太夸张了？

236333543819969997 赞同

👍 1    💬 0



(/users/znzbvrdqkaqds)

游客znzbvrdqkaqds (/users/znzbvrdqkaqds)

2019-08-15 12:13:10

Redis集群需要使用复制来保证高可用，RedLock不支持复制的话，那我怎么保证集群的高可用。

👍 0    💬 0

相关文章

(/articles/674395)

Redisson实现Redis分布式锁的N种姿势

技术小能手👉 (/users/o3u4mvu6rfxm2)    ⌚ 2018-12-03 09:16:26    👁 浏览2077

(/articles/86680)

《Redis官方文档》用Redis构建分布式锁

青衫无名 (/users/s7cq5fhefigma)    ⌚ 2017-05-22 09:27:00    👁 浏览1054

(/articles/16963)

《Redis官方文档》用Redis构建分布式锁

ali清英 (/users/dtyz3662dkx3a)    ⌚ 2016-04-01 11:36:20    👁 浏览1359

(/articles/584969)

体验一键php/java环境安装工具oneinstack

bboysoul👉 (/users/x63t45s3xlkqq)    ⌚ 2018-04-25 21:46:43    👁 浏览3316

(/articles/533138)

最牛逼的自媒体平台今日头条申请秘籍，包过！

suboysugar (/users/gouchxrqo5i2g)    ⌚ 2016-04-07 14:09:00    👁 浏览416

(/articles/393516)

网友最牛逼的回复，笑死人

y0umer (/users/tu34v6lsxt7eu)    ⌚ 2010-01-27 10:50:00    👁 浏览389

(/articles/681270)

京东内部那些牛逼的技术点终于汇总了

java填坑路 (/users/o56geandm2j2o)    ⌚ 2018-12-07 15:13:33    👁 浏览626

(/articles/80333)

基于Redis的分布式锁真的安全吗？（上）

稀奇古怪 (/users/fointt4u7wz4q)    ⌚ 2017-05-16 10:06:00    👁 浏览1429

(/articles/466040)

人工智能终究会抢了我们程序员的饭碗

littletigerbj (/users/vhbobbuppv3q)    ⌚ 2017-04-05 21:16:00    👁 浏览488

(/articles/639958)

探索Redis设计与实现15：Redis分布式锁进化史

黄小斜👉 (/users/alppkiuo2tyco)    ⌚ 2018-02-14 20:10:40    👁 浏览879

(/articles/502950)

Redis分布式锁

玄学酱 (/users/xlyqrd2zxlspg)    ⌚ 2018-02-27 16:02:00    👁 浏览1779





<div>(/articles/694253)</div> <div>redis实现分布式锁</div> <div>快乐崇拜007👉 (/users/2lwhbrw5nyudy)🕒 2019-03-19 10:23:24 👁 浏览645</div>
<div>(/articles/625308)</div> <div>redis系列：分布式锁</div> <div>勿妄 (/users/4jfig4lpq4ecu)🕒 2018-08-15 09:20:34 👁 浏览1575</div>
<div>(/articles/660420)</div> <div>redis系列：分布式锁</div> <div>java小胡哥👉 (/users/ojk3sa42sqizj)🕒 2018-08-15 21:09:00 👁 浏览404</div>
<div>(/articles/503094)</div> <div>supervisor学习笔记</div> <div>余二五 (/users/fy5hholq4xfqs)🕒 2017-11-15 16:53:00 👁 浏览781</div>
<div>(/articles/676290)</div> <div>大公司为什么还在采用过时的技术</div> <div>java隋七哥 (/users/qeocmagg5wsos)🕒 2018-10-29 22:05:00 👁 浏览335</div>
<div>(/articles/80353)</div> <div>基于Redis的分布式锁真的安全吗？（下）</div> <div>稀奇古怪 (/users/fointt4u7wz4q)🕒 2017-05-16 10:28:00 👁 浏览1043</div>
<div>(/articles/443930)</div> <div>Base PyQt4, Simple Web APP Framwork</div> <div>北之燕 (/users/kkaiqaq6t6cq6)🕒 2012-01-15 15:54:00 👁 浏览510</div>
<div>(/articles/621136)</div> <div>为什么分布式要有分布式锁！</div> <div>技术小能手👉 (/users/o3u4mvu6rfxm2)🕒 2018-08-01 14:43:34 👁 浏览4209</div>
<div>(/articles/674720)</div> <div>12月3日云栖精选夜读   Python程序员的30个常见错误</div> <div>yq传送门👉 (/users/dwbolajahkeg4)🕒 2018-12-03 17:15:10 👁 浏览959</div>
<div>下拉加载更多</div>

热点导航	阿里云618大促 (/https://www.aliyun.com/acts/hi618/index)云计算 (/https://www.aliyun.com/)网络安全 (/https://market.aliyun.com/security)互联网架构 (/https://www.aliyun.com/aliware) ECS升级配置 (/https://yq.aliyun.com/ask/53742)物联网 (/https://www.aliyun.com/product/iot)中间件比赛 (/https://tianchi.aliyun.com/competition/entrance/231714/introduction) 云栖博客 (/https://yq.aliyun.com/articles/)
用户关注	自动化测试 (/https://bbs.aliyun.com/read/301499.html)解决方案 (/https://www.aliyun.com/solution/all)linux命令 (/https://yq.aliyun.com/articles/34777)云服务 (/https://www.aliyun.com/ ) JavaScript 函数 (/https://yq.aliyun.com/articles/92145)服务器监控 (/https://yq.aliyun.com/articles/48786)Python语言 (/https://yq.aliyun.com/roundtable/56407)移动数据分析 (/https://www.aliyun.com/roundtable/56407)
更多推荐	用户体验 (/https://yq.aliyun.com/articles/132294)云数据库Rds (/https://help.aliyun.com/product/26090.html)负载均衡 (/https://www.aliyun.com/product/slb/)域名注册 (/https://wanwang.aliyun.com/domain/)Whois查询 (/https://whois.aliyun.com/)数据可视化 (/https://help.aliyun.com/product/43570.html)ICP备案查询 (/https://beian.aliyun.com/)主题地图 (/https://yq.aliyun.com/zt)阿里云大学 (/https://ed.cn域名 (/https://wanwang.aliyun.com/domain/cn/)移动站 (/https://m.aliyun.com/yunqi/)IT论坛 (/https://bbs.aliyun.com/)阿里云开年Hi购季 (/https://www.aliyun.com/acts/product-section-2019/ho企业邮箱 (/https://mail.aliyun.com/)新用户优惠 (/https://www.aliyun.com/acts/product-section-2019/new-users)

关于我们 (/https://www.aliyun.com/about)      法律声明及隐私权政策 (/https://terms.aliyun.com/legal-agreement/terms/suit\_bu1\_ali\_cloud/suit\_bu1\_ali\_cloud201710161525\_98396.html)      廉正举报 (/https://jubao.alibaba.com/index.html?site=ALIYUN) (/https://www.aliyun.com/contact)      加入阿里云 (/https://www.aliyun.com/careers)

阿里巴巴集团 (http://www.alibabagroup.com/cn/global/home)    淘宝网 (/https://www.taobao.com/)    天猫 (/https://www.tmall.com/)    聚划算 (/https://ju.taobao.com/)    全球速卖通 (/https://www.aliexpress.com/)    阿里巴巴国际交易市场 (/https://www.alibaba.com/)    1688 (/https://www.1688.com/)    阿里妈妈 (/https://www.alimama.com/index.htm)    飞猪 (/https://www.fliggy.com)    阿里云计算 (/https://www.aliyun.com/)    AliOS (/https://www.alios.cn/)    阿里通信 (/https://aliquin.tmall.com/)    万网 (/https://wanwang.aliyun.com/)    高德 (http://www.autonavi.com/)    UC (http://www.uc.cn/)    友盟 (/https://www.umeng.com/)    虾米 (/https://www.xiami.com/)    优酷 (/https://www.youku.com/)    钉钉 (/https://www.dingtalk.com/?lwfrom=20150205111943449)    支付宝 (https://www.alipay.com/)    达摩院 (https://damo.alibaba.com/)

© 2009-2019 Aliyun.com 版权所有 ICP证：浙B2-20080101  
 (http://idinfo.zjaic.gov.cn/bscx.do?method=lzxx&id=3301963301080000025024)  
 浙公网安备 33010602009975号 (http://www.beian.gov.cn/portal/registerSystemInfo?COLLCC=2716891795&recordcode=33010602009975)

