distributed lock solution

Frank Han

Date: 1568268686

Base on

Redis

总结

分布式锁的实现思路和需求

实现思路：任何对于分布式系统共享/可见的资源

原子方式获取/释放就可以作为分布式锁(获取代表加锁，反之释放锁)

至于资源数量，根据锁的类型改变而改变（独享锁资源数量为1，共享锁资源数量为∞，限制锁资源数量为n(n>=1&&n<∞)）

关键字：原子 获取 释放 数量 资源

考察

- REDIS能否作为分布式锁?

可以！原因：

redis&redis的key对于分布式系统可见/共享

redis是kv存储的数据库，key可以作为数量为1的资源

redis可以原子的设置资源key

REDIS 分布式锁实现v1.0

- 加锁：使用SETNX 命令
- SETNX lockname

什么是setnx？

setnx：如果该key不存在，返回1代表设置成功

如果已存在　　返回0代表设置失败

这样的话我们认为返回1代表加锁成功，反之失败

REDIS 分布式锁实现v1.0

- 释放锁:

```
DEL lockname
```

代表锁删除成功

可能存在的问题

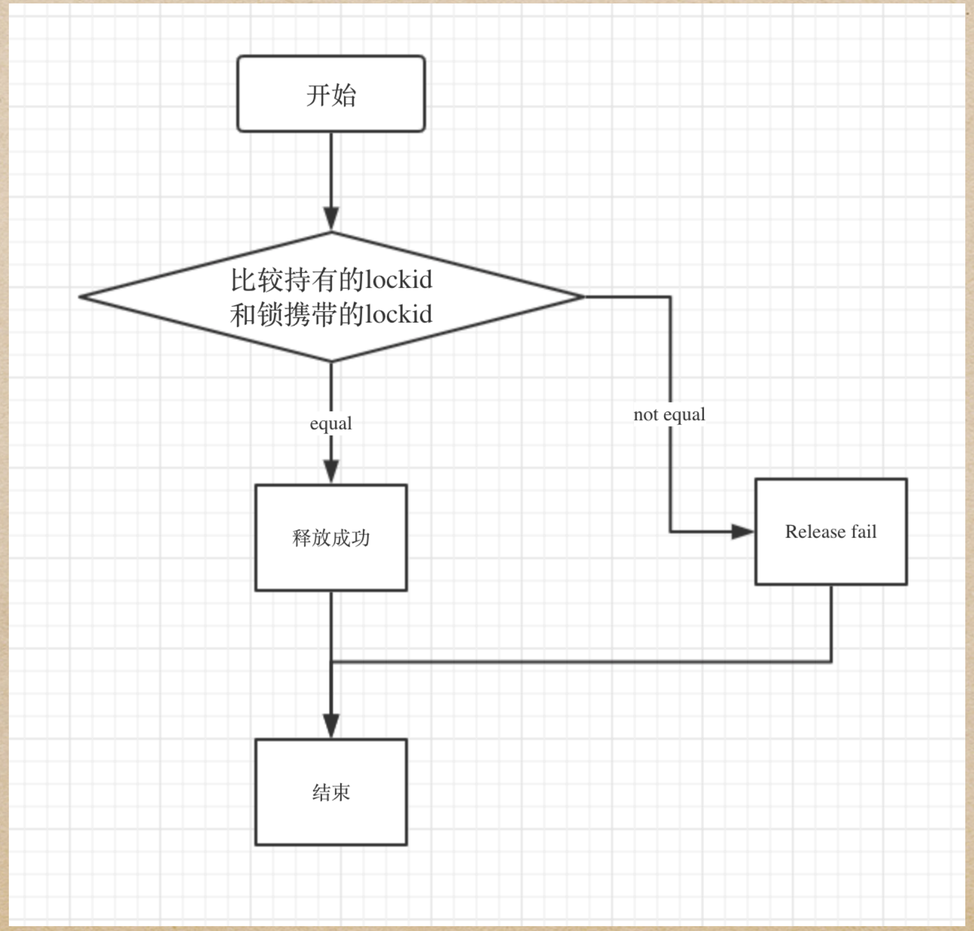
- 如果程序卡死，锁不能被释放
- 任何程序都可以释放锁，非常不安全

解决问题

SET lockname lockid ex 15 NX
使用这个命令解决了问题
1.时间太长释放
2.设置lockid为value，确保只有持有ID的线程才能释放锁
3.使用EXPIRE去更新剩余时间

如何解锁?

- 流程:


开始
比较持有的lockid
和锁携带的lockid
equal
not equal
释放成功
Release fail
结束

新的问题

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

如何解决？！！

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。
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。

这种方式如何加锁?

- 依次尝试从5个实例，使用相同的key和**具有唯一性的**ID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节点都取到锁，并且使用的时间小于锁失效时间时，锁才算获取成功**。
- 如果取到了锁，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实例取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）。

这个方案是否还有问题?