Redis 分布式锁

1.前言

说到 Redis，我们第一想到的功能就是可以缓存数据，除此之外，Redis 因为单进程、性能高的特点，它还经常被用于做分布式锁。

锁我们都知道，在程序中的作用就是同步工具，保证共享资源在同一时刻只能被一个线程访问，Java 中的锁我们都很熟悉了，像`synchronized`、`Lock`都是我们经常使用的，但是 Java 的锁只能保证单机的时候有效，分布式集群环境就无能为力了，这个时候我们就需要用到分布式锁。

分布式锁，顾名思义，就是分布式项目开发中用到的锁，可以用来控制分布式系统之间同步访问共享资源，一般来说，分布式锁需要满足的特性有这么几点：

1. **互斥性**： 在任何时刻，对于同一条数据，只有一台应用可以获取到分布式锁；
2. **高可用性**： 在分布式场景下，一小部分服务器宕机不影响正常使用，这种情况就需要将提供分布式锁的服务以集群的方式部署；
3. **防止锁超时**： 如果客户端没有主动释放锁，服务器会在一段时间之后自动释放锁，防止客户端宕机或者网络不可达时产生死锁；
4. **独占性**： 加锁解锁必须由同一台服务器进行，也就是锁的持有者才可以释放锁，不能出现你加的锁，别人给你解锁了；

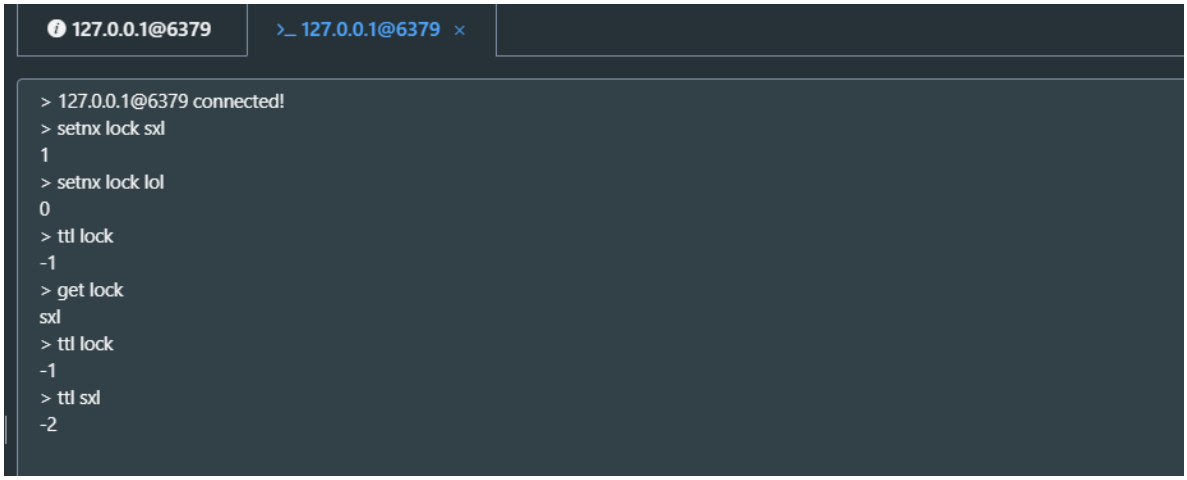
业界里可以实现分布式锁效果的工具很多，但操作无非这么几个：**加锁、解锁、防止锁超时**。既然本文说的是 Redis 分布式锁，那我们理所当然就以 Redis 的知识点来延伸。

2. 实现锁的命令

先介绍下 Redis 的几个命令，

1、`SETNX`，用法是`SETNX key value`

`SETNX`是『 SET if Not exists』(如果不存在，则 SET)的简写，设置成功就返回 1，否则返回 0。



可以看出，当把**key**为**lock**的值设置为"Java"后，再设置成别的值就会失败，看上去很简单，也好像独占了锁，但有个致命的问题，就是**key**没有过期时间，这样一来，除非手动删除 key 或者获取锁后设置过期时间，不然其他线程永远拿不到锁。

既然这样，我们给 key 加个过期时间总可以吧，直接让线程获取锁的时候执行两步操作：

返回

当 key 不存在时，返回 -2 。 当 key 存在但没有设置剩余生存时间时，返回 -1 。 否则，以秒为单位，返回 key 的剩余生存时间。

注意：在 Redis 2.8 以前，当 key 不存在，或者 key 没有设置剩余生存时间时，命令都返回 -1 。

2、SETEX，用法`SETEX key seconds value`

将值 `value` 关联到 `key` ，并将 `key` 的生存时间设为 `seconds` (以秒为单位)。如果 `key` 已经存在，`SETEX` 命令将覆写旧值。

这个命令类似于以下两个命令：

```
> setex lock 10 java
OK
> get lock
java
> ttl lock
-2
> get lock
null
```

3、PSETEX，用法`PSETEX key milliseconds value`

这个命令和`SETEX`命令相似，但它以毫秒为单位设置 `key` 的生存时间，而不是像`SETEX`命令那样，以秒为单位。

不过，从 Redis 2.6.12 版本开始，`SET`命令可以通过参数来实现和`SETNX`、`SETEX`、`PSETEX` 三个命令的效果。

就比如这条命令：

```
SET key value [EX seconds] [PX milliseconds] [NX|XX]
```

- EX seconds： 将键的过期时间设置为 seconds 秒。
- PX milliseconds： 将键的过期时间设置为 milliseconds 毫秒。
- NX： 只在键不存在时， 才对键进行设置操作。 执行 SET key value NX 的效果等同于执行 SETNX key value 。
- XX： 只在键已经存在时， 才对键进行设置操作。

```
> set test shen PX 6 XX
OK
> get test
null
```

3. 怎么释放锁

释放锁的命令就简单了，直接删除 key 就行，但我们前面说了，因为分布式锁必须由锁的持有者自己释放，所以我们必须先确保当前释放锁的线程是持有者，没问题了再删除，这样一来，就变成两个步骤了，似乎又违背了原子性了，怎么办呢？

不慌，我们可以用 lua 脚本把两步操作做拼装，就好像这样：