1、为什么使用分布式锁

2、分布式锁需要实现什么效果

3、实现方式和对比

4、zk的具体实现

先看下这个业务场景，在单节点系统中，有共享变量count=10，接口A对count加1操作，接口B做count减1操作。现在同时请求A和B接口，A和B取到count的变量值都是10，然后A做+1操作，B做-1操作，A先操作完成，count赋值为11，然后B操作完成，count赋值为9。预期结果count值是10不变，输出结果为9。这是多线程中常见的数据一致性问题，解决方法很简单，每次修改操作对count加锁就可以了。

可是分布式系统中，多个系统中的线程同时操作一个临界资源呢，就没法通过Lock来加锁了，如下图。



三个系统同时操作变量A，怎么来实现系统间线程的互斥加锁呢，这就需要分布式锁了。

一、分布式锁的具备的条件

1、在分布式系统环境下，一个方法在同一时间只能被一个机器的一个线程执行；

2、高可用的获取锁与释放锁；

3、高性能的获取锁与释放锁；

4、具备可重入特性；

5、具备锁失效机制，防止死锁；

6、具备非阻塞锁特性，即没有获取到锁将直接返回获取锁失败。

二、分布式锁的实现方式

实现分布式锁首要条件的是有一个公共区域存储锁的互斥量，各个系统都能访问这个公共区域，才能互斥的加锁。这个互斥量的存放方案有很多：数据库、redis、zookeeper等等，具体是怎么实现呢

1、数据库实现

在数据库中创建一张表，如下

CREATE TABLE `cloudlock` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`lockName` varchar(128) DEFAULT NULL COMMENT '锁名称',

`expire` datetime DEFAULT NULL COMMENT '过期时间',

`createTime` datetime DEFAULT NULL COMMENT '创建时间',

PRIMARY KEY (`id`),

UNIQUE KEY `lock` (`lockName`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

1. 加锁，进行一次数据的插入，插入成功获取到锁，否则加锁失败

INSERT INTO `cloudlock`(`lockName`,`expire`,`createTime`) VALUES ('lockName','2020-06-15 23:39:48','2020-06-15 21:39:58');

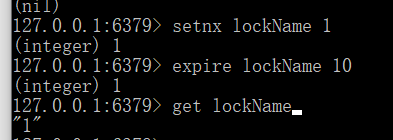
2）解锁，删除数据

DELETE FROM cloudLock WHERE lockName='lockName'

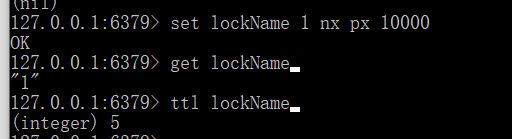
2、Redis实现

1）加锁，有两种方法

setnx(原子的加锁) + expire（设置过期时间）



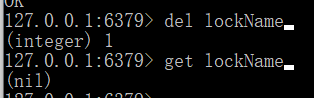
SET key lockName NX PX timeout（原子的加锁设置过期时间）



setnx+ expire 的加锁过程无法保证加锁的原子性，如果加锁完成，还没执行exprie的延期命令，redis就宕机了，就会出现无法解锁的问题。

1. 解锁

等待过期或者代码执行完后删除数据



## 可以选择的成熟框架Redisson

3、Zookeeper实现

使用zk