Distributed Lock

简述

- 传统多线程
- 分布式锁

分布式锁的常见应用场景

分布式锁的概念和常见实现方式

几个重要的概念：

- 分布式锁管理(DLM): Distributed Lock Manager，分配给节点锁和释放锁，对分布式锁进行统一的管理
- 租约(lease): 从DLM，获得锁的有效期，超过有效期，DLM自动回收锁，租约保证了一个客户端不会占用锁很长时间。一定程度上保证了高效性。

常见实现方式：

- 基于数据库
- 基于Redis
- 基于zookeeper

基于zookeeper

- 给每一个进程创建一个临时顺序节点，节点会监听锁的状态
- 当锁被释放以后，每个子节点都会判断自己的序号是否是最小的，
- 如果是，就获得锁
- 如果不是的话，会做如下判断：
  - 如果是读，前面的子节点都是读，那么可以读
  - 如果是写，前面有子节点是写，那么就继续等待

基于zookeeper的优缺点

- 羊群效应(Herd Effect) - 应每个节点创建者只关注比自己序号小的那个节点
- 大量的监听和节点信息的获取非常消耗资源

基于Redis

- **setnx()** - Redis 通常可用setnx(key, value) 函数来实现分布式锁 - 返回1，说明该服务器器获得锁，setnx 将key 对应的value 设置为当前时间 + 锁的有效时间；返回0，说明其他服务器器已经获得了锁，进程不能进入临界区。该服务器器可以不断尝试setnx 操作，以获得锁。
- Redlock
- Redission

基于Redis的优缺点

- 性能更好。数据被存放在内存，而不是磁盘，避免了频繁的IO 操作
- 很多缓存可以跨集群部署，避免了单点故障问题
- 很多缓存服务都提供了可以用来实现分布式锁的方法，比如Redis 的setnx 方法等
- 可以直接设置超时时间来控制锁的释放，因为这些缓存服务器器一般支持自动删除过期数据

- 时间很难控制
- 时间短，可能错误释放
- 时间长，会拖慢系统处理速度

基于数据库 - Demo

```
select .......for update
```

基于数据库的优缺点

三种方式比较

理解容易度: 数据库> 缓存> zk

性能: 缓存> zk > 数据库

可靠性: zk > 缓存> 数据库