主讲老师：Fox

> 有道笔记地址：https://note.youdao.com/s/F6cC7lvp

# 1. Zookeeper 分布式锁实战

## 1.1 什么是分布式锁

在单体的应用开发场景中涉及并发同步的时候，大家往往采用Synchronized（同步）或者其他同一个JVM内Lock机制来解决多线程间的同步问题。在分布式集群工作的开发场景中，就需要一种更加高级的锁机制来处理跨机器的进程之间的数据同步问题，这种跨机器的锁就是分布式锁。

目前分布式锁，比较成熟、主流的方案：

（1）基于数据库的分布式锁。这种方案使用数据库的事务和锁机制来实现分布式锁。虽然在某些场景下可以实现简单的分布式锁，但由于数据库操作的性能相对较低，并且可能面临锁表的风险，所以一般不是首选方案。

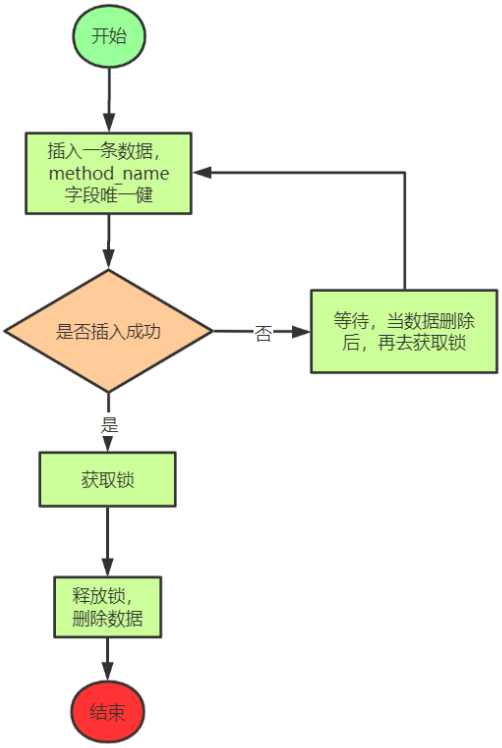
（2）基于Redis的分布式锁。Redis分布式锁是一种常见且成熟的方案，适用于高并发、性能要求高且可靠性问题可以通过其他方案弥补的场景。Redis提供了高效的内存存储和原子操作，可以快速获取和释放锁。它在大规模的分布式系统中得到广泛应用。

（3）基于ZooKeeper的分布式锁。这种方案适用于对高可靠性和一致性要求较高，而并发量不是太高的场景。由于ZooKeeper的选举机制和强一致性保证，它可以处理更复杂的分布式锁场景，但相对于Redis而言，性能可能较低。

> 基于Redis实现分布式锁
> https://vip.tulingxueyuan.cn/detail/p_602e53b3e4b035d3cdb8f856/6

## 1.2 基于数据库设计思路

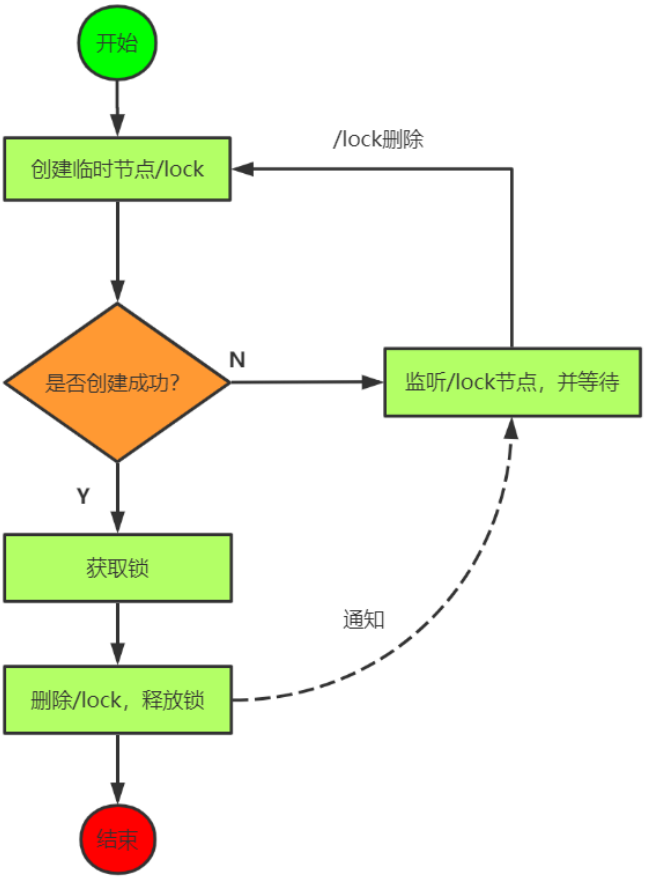
可以利用数据库的唯一索引来实现，唯一索引天然具有排他性




思考：基于数据库实现分布式锁存在什么问题？

## 1.3 基于Zookeeper设计思路一

使用临时 znode 来表示获取锁的请求，创建 znode成功的用户拿到锁。



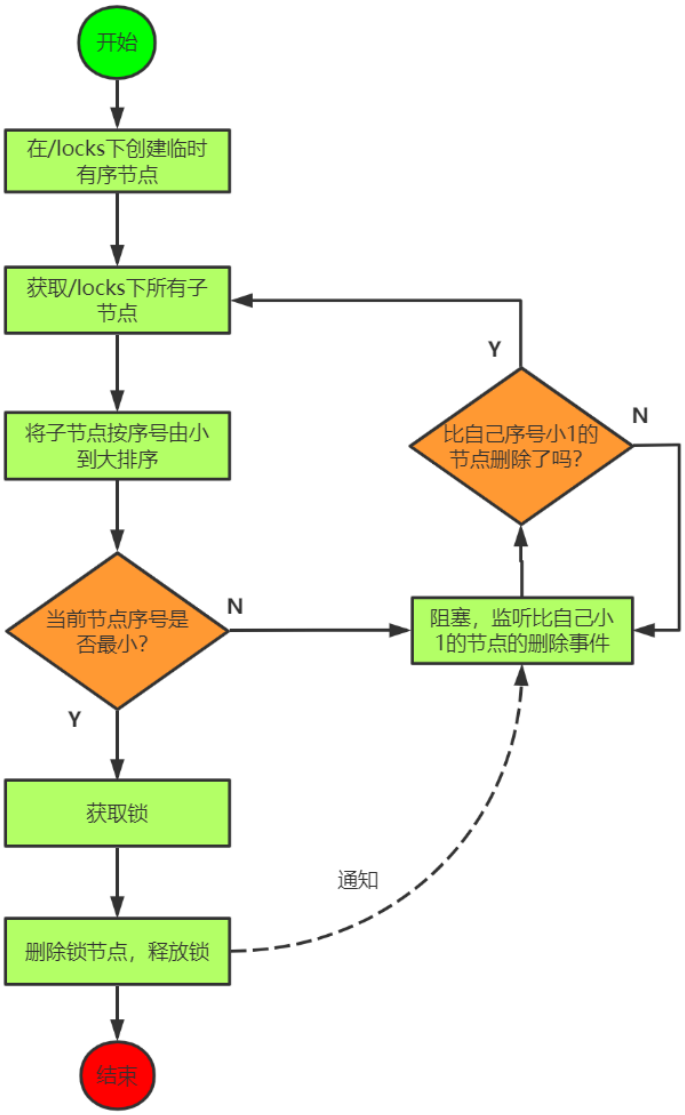

思考：上述设计存在什么问题?

如果所有的锁请求者都 watch 锁持有者，当代表锁持有者的 znode 被删除以后，所有的锁请求者都会通知到，但是只有一个锁请求者能拿到锁。这就是羊群效应。

## 1.4 基于Zookeeper设计思路二

使用临时有序znode来表示获取锁的请求，创建最小后缀数字 znode 的用户成功拿到锁。

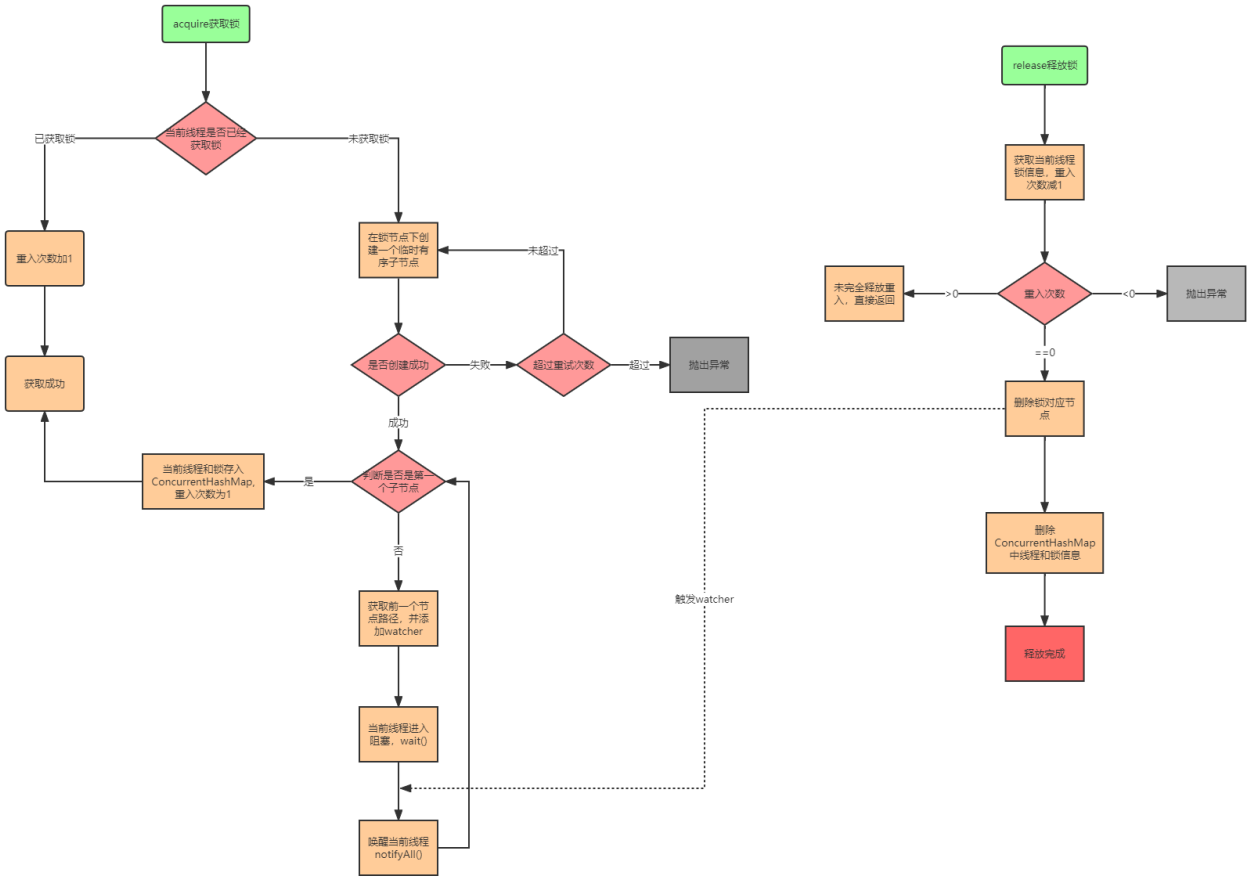
公平锁的实现




在实际的开发中，如果需要使用到分布式锁，不建议去自己“重复造轮子”，而建议直接使用Curator客户端中的各种官方实现的分布式锁，例如其中的InterProcessMutex可重入锁。

## 1.5 Curator 可重入分布式锁工作流程

https://www.processon.com/view/link/5cadacd1e4b0375afbef4320




## 1.6 总结

优点：ZooKeeper分布式锁（如InterProcessMutex），具备高可用、可重入、阻塞锁特性，可解决失效死锁问题，使用起来也较为简单。

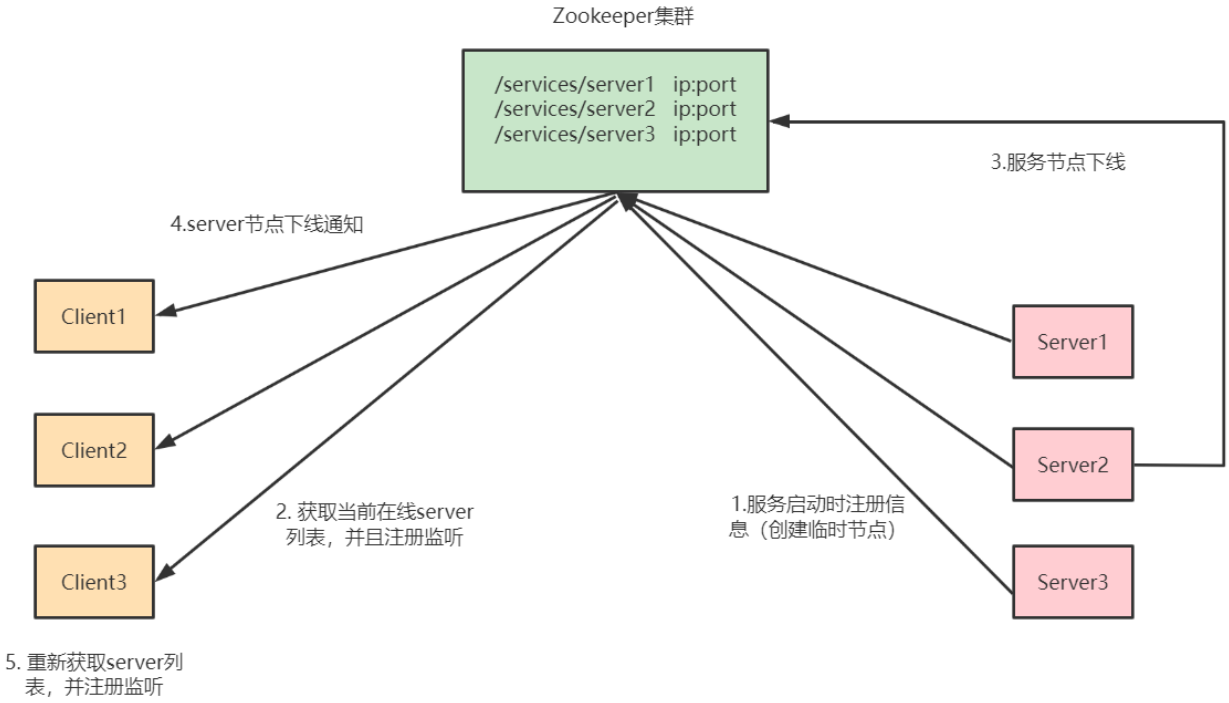
缺点：因为需要频繁的创建和删除节点，性能上不如Redis。

在高性能、高并发的应用场景下，不建议使用ZooKeeper的分布式锁。而由于ZooKeeper的高可靠性，因此在并发量不是太高的应用场景中，还是推荐使用ZooKeeper的分布式锁。

# 2. 基于Zookeeper实现服务的注册与发现

基于 ZooKeeper 本身的特性可以实现服务注册中心

## 2.1 设计思路




## 2.2 Zookeeper实现注册中心的优缺点

**优点：**

- 高可用性：ZooKeeper是一个高可用的分布式系统，可以通过配置多个服务器实例来提供容错能力。如果其中一个实例出现故障，其他实例仍然可以继续提供服务。
- 强一致性：ZooKeeper保证了数据的强一致性。当一个更新操作完成时，所有的服务器都将具有相同的数据视图。这使得ZooKeeper非常适合作为服务注册中心，因为可以确保所有客户端看到的服务状态是一致的。
- 实时性：ZooKeeper的监视器（Watcher）机制允许客户端监听节点的变化。当服务提供者的状态发生变化时（例如，上线或下线），客户端会实时收到通知。这使得服务消费者能够快速响应服务的变化，从而实现动态服务发现。

**缺点：**

- 性能限制：ZooKeeper的性能可能不如一些专为服务注册中心设计的解决方案，如nacos或Consul。尤其是在大量的读写操作或大规模集群的情况下，ZooKeeper可能会遇到性能瓶颈。

## 2.3 整合Spring Cloud Zookeeper实现微服务注册中心

https://spring.io/projects/spring-cloud-zookeeper#learn

第一步：在父pom文件中指定Spring Cloud版本

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
```

```
    <version>2.3.2.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
<properties>
    <java.version>1.8</java.version>
    <spring-cloud.version>Hoxton.SR8</spring-cloud.version>
</properties>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>${spring-cloud.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

注意： springboot和springcloud的版本兼容问题

第二步：微服务pom文件中引入Spring Cloud Zookeeper注册中心依赖

```
<!-- zookeeper服务注册与发现 -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-zookeeper-discovery</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.apache.zookeeper</groupId>
            <artifactId>zookeeper</artifactId>
        </exclusion>
    </exclusions>
</dependency>

<!-- zookeeper client -->
<dependency>
    <groupId>org.apache.zookeeper</groupId>
```

```
    <artifactId>zookeeper</artifactId>
    <version>3.8.0</version>
</dependency>
```

注意： zookeeper客户端依赖和zookeeper sever的版本兼容问题

Spring Cloud整合Zookeeper注册中心核心源码入口： ZookeeperDiscoveryClientConfiguration

第三步： 微服务配置文件application.yml中配置zookeeper注册中心地址

```
spring:
  cloud:
    zookeeper:
      connect-string: localhost:2181
      discovery:
        instance-host: 127.0.0.1
```

注册到zookeeper的服务实例元数据信息如下：

注意：如果address有问题，会出现找不到服务的情况，可以通过instance-host配置指定

第四步：整合feign进行服务调用

```
@RequestMapping(value = "/findOrderByUserId/{id}")
public R  findOrderByUserId(@PathVariable("id") Integer id) {
    log.info("根据userId:"+id+"查询订单信息");
    //feign调用
    R result = orderFeignService.findOrderByUserId(id);
    return result;
}
```

测试：http://localhost:8040/user/findOrderByUserId/1