zxid分高32位、低32位。

提问1：如果低32位被占满了，怎么办？

回答1：如果低32溢出，这个时候重新进行选举。

提问2：为什么启动zk1、2、3，有时候1是leader，有时2是leader

回答2：因为在启动2台节点的时候，就可以进行选举了。

# Zookeeper在大型分布式系统中的应用

1. 有些分布式系统master-slave，master是一个单节点（备份master-back）。

实际的案例：Hadoop（NameNode、ResourceManager），普通的部署NameNode、ResourceManager仅仅是单节点。Hadoop HA（NameNode和ResourceManager有多个备份）

说明：统一的一个临时节点：ActiveOrStandByLock（/distibuted\_system/ActiveOrStandByLock仅仅这样一个节点）

第一步：zk有这样一个持久节点/distibuted\_system

第二步：master1和master2同时启动，同时向向/distributed\_system这个节点申请创建临时子节点ActiveOrStandByLock（同一时间只有一个请求能够创建成功）。

如果master1创建成功，这个节点（ActiveOrStandByLock）就不允许master2创建（锁的机制）

master1：active===》真正的master。路径：/distributed\_system/ActiveOrStandByLock

master2：改为standby（master-back）。

同时对/distributed\_system/ActiveOrStandByLock注册事件监听。

第三步：master1挂掉或者超过一定时间。节点会被删除（事件机制就会起作用），就会通知master2，master2就会在/distributed\_system/ActiveOrStandByLock，同时修改状态为active。

备注：假如master1并没有挂掉，只有由于网络延时导致，当网络顺畅的时候就会出现“脑裂”状态。都认为自己是active。

解决脑裂的办法：对/distributed\_system/ActiveOrStandByLock加一个权限ACL控制。master1对于这个节点/distributed\_system/ActiveOrStandByLock没有权限。自己把状态改成standby。

# Zookeeper实现分布式锁

分布式锁主要用于在分布式环境中包括跨主机、跨进程、跨网络，导致共享资源不一致的问题，保证数据的一致性。

zookeeper

LOCK

Client\_1

Client\_2

Client\_3

Client\_n

Node-...n

Node-...3

Node-...2

Node-...1

分布式锁的思路

获取锁

锁是否被占用

创建节点

是否成功

占用锁

完成逻辑

释放锁

等待锁

缺点：“惊群效应”

优化后的算法：

创建临时顺序节点Node-n

获取/LOCK下面的所有子节点

排序所有子节点

判断子节点是否最小

占用锁

完成逻辑

释放锁

找出比Node-n次小的节点

对次小节点进行监听

等待锁

是

否

等待Watcher通知

通知

创建临时顺序节点Node-n

获取/LOCK下面的所有子节点

排序所有子节点

判断子节点是否最小

占用锁

完成逻辑

释放锁

找出比Node-n次小的节点

对次小节点进行监听

等待锁

是

否

等待Watcher通知

通知

# Zookeeper Leader选举源码分析

Election选举类

QuorumCnxManager：主要完成服务器间的网络交互

Vote 选票

QuorumPeer 是zk服务器实例类

QuorumPeer.QuorumServer 存储zk server连接信息

主要实现函数：

lookForLeader：选举过程（主要函数）

totalOrderPredicate：比较选票

termPredicate：是否结束选举

senderWorkerMap：主要用于发送

queueSendMap：发送队列

SendWorker：发送器

RecvWorker：接收器

lastMessageSent：最后一条发送消息

Notification：表示该服务器节点收到选票信息

recvset：存储本节点来自其他节点的选票(投票用)

outofelection：存储本节点从其他节点由following、leading状态发送来的选票（选举确认用）

|  |  |
| --- | --- |
| **第一节** | **第二节** |
| Zokeeper安装配置及简单使用 | Zookeeper API如何进行调用 |
| 分布式架构详解 | 一致性协议详解 |
| Paxos算法及ZAB协议 | Chubby技术架构解析 |
| ZkClient、Curator | ZK大型分布式系统中的应用 |
| Zk分布式锁、发布订阅 | Zk命名服务、Master选举 |
| Zk高级特性：服务器角色 | Zk高级特性：系统模型 |
| Zk高级特性：会话与客户端 | Zk高级特性：数据与存储 |
| Zk高级特性：序列化与通信协议 | Zk高级特性：Leader选举解析 |

今天讲解的内容用颜色标识。