# Zookeeper基本概念

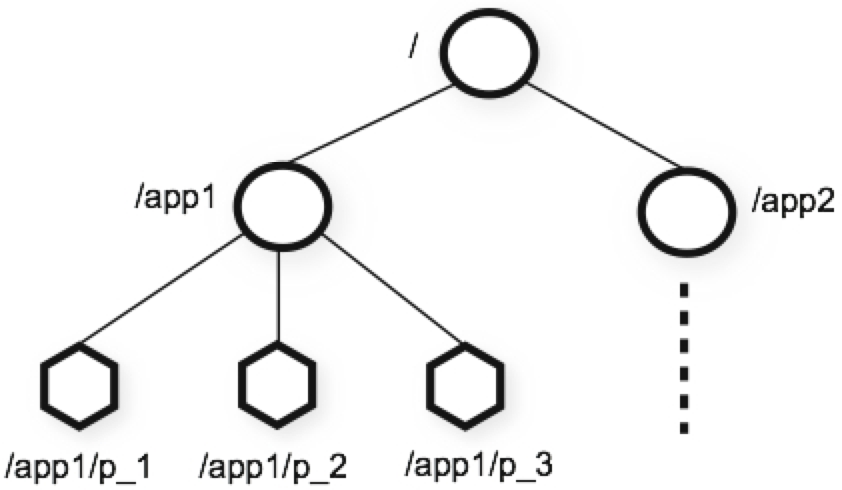
## 定位：

分布式协调工具

## 数据模型：key/value

key 类似文件存储系统的路径

value对应了一个节点node，节点可以**存储1M**大小数据



## node详解：

#### node类型：

###### 1:普通节点

create 创建的普通存储数据的节点

###### 2:临时节点

create –e 创建的节点，session结束之后，会话消失

###### 3:序列化节点

create –s 创建的节点，保证了重名之后，重命名的操作，可以和-e 一起使用

## node包含的信息：

创建事件zid

创建时间

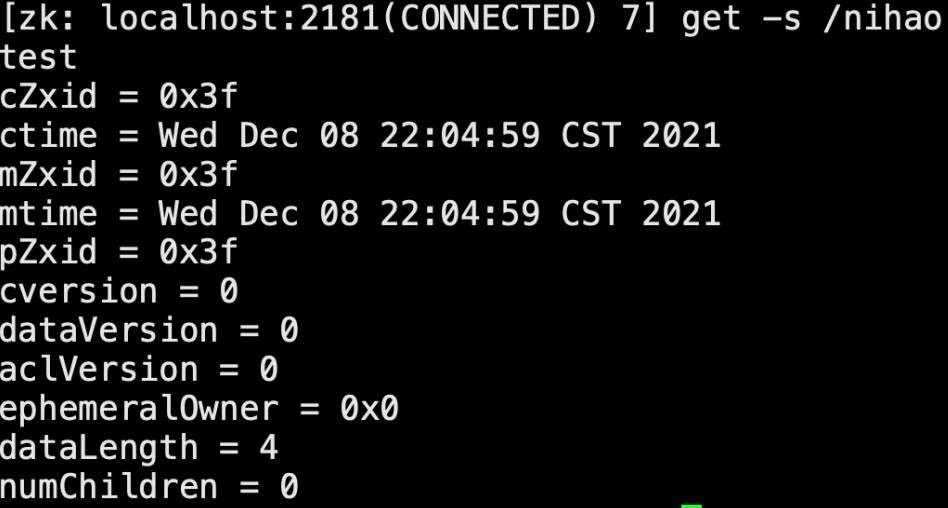
修改事件的zid

修改时间

子节点最后创建的事件id

拥有者：create –e的节点才有意义

ps：每个id都是8字节，前4字节代表leader纪元，后4字节代表 事件id



## Session：

每个连接都有 session的概念，session是zk集群共享的

# Zookeeper应用

## API:

create path [value]

delete path

ls path

get path

set path value

## Wather：

允许客户端对key进行监听，当监听的key变化的时候，server会通知到client

## 应用场景：

1：分布式锁

kafka用此分布式锁进行选broker【HA】

2：注册发现

利用watcher机制，server给client推数据

4：统一配置

5：统一命名

create –s

# zookeeper分布式

## ZAB协议：

角色：

1:leader：主节点

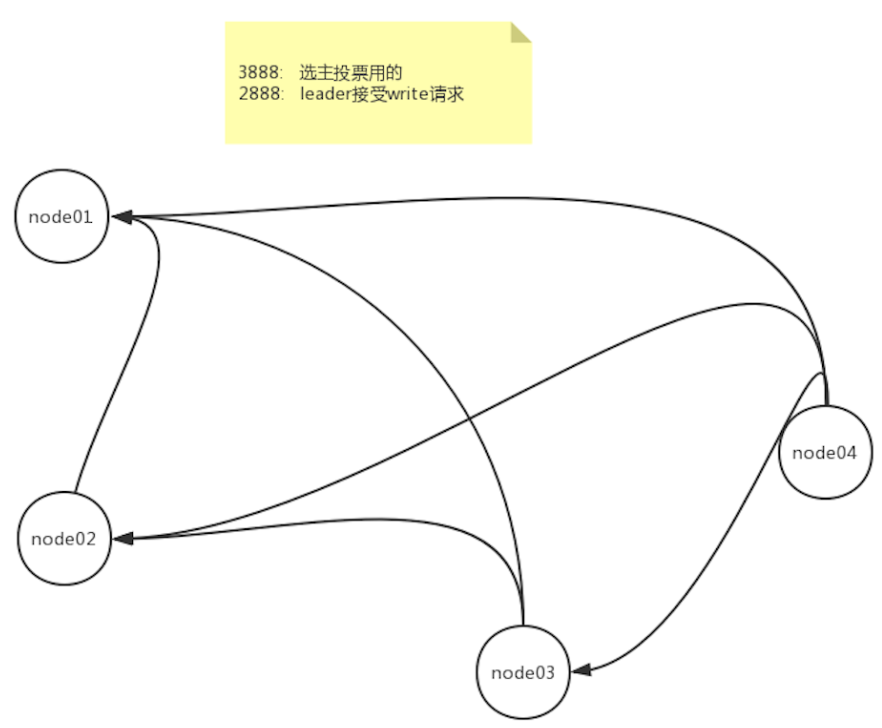
2:follower：参与选举，数据投票

3:observer：只提供查询过程

投票流程：

leader发起投票，发送事务id，follower接收到投票，发现比自己存储的事务id大，那么同意该投票，leader发现半数都同意了，让follower更新数据（两阶段）。

## X：主从结构



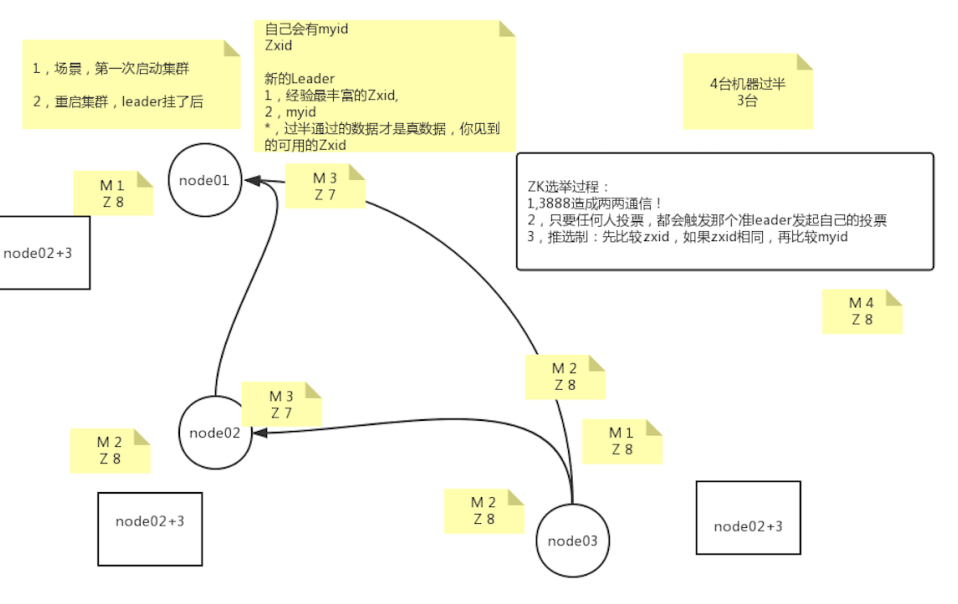
只能主节点write，从节点提供read服务

###### 主节点HA：

1:使用zab协议，节点发现主节点失联，此时发起选主投票

2:收到投票，如果自己的事务id比它的小，同意投票；否则自己发起投票

3:leader收到半数投票，告诉follower，自己是主



###### 主从同步：（半数同意+使用队列 + sync 最终一致性）

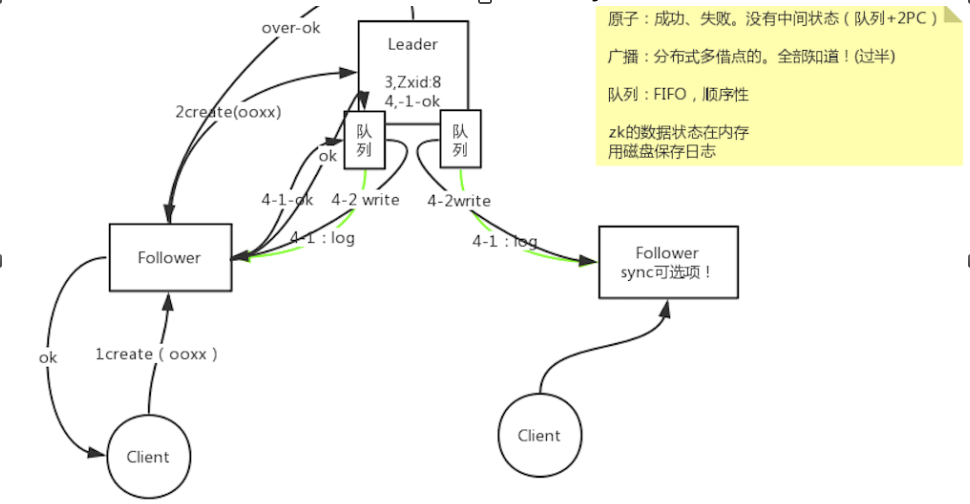
1:使用zab协议，主节点发起数据更新请求（包含zid），并且给每个follower维护一个队列

2:follower发现zid比本地最大的zid大， 同意数据更新

3:leader发现半数通过，统治follower更新数据（两阶段）

4:没有及时同步的，client每次get前可以先sync一把，再get





## Y/Z轴：不要使用zookeeper来存储大量数据，不存在X/Y；zookeeper的本质是分布式协调工具，不是分布式存储

## Zookeeper Server的四种状态：

1：leading

2：following

3：observing

4：looking：不能提供服务

# 分布式配置：

#### 原理：

使用watch机制

#### 流程：

配置：

1：create 配置key value

2：set 配置key value

服务启动：

1：get 配置信息key

2：watch 配置信息key

#### 其他组件是否可以实现：

1:redis：可以，使用add，get来存取数据，客户端定时去查是否变更

注意，不能使用redis的publish和subscribe来实现，这个是实时的，比如新增一个客户端，或者客户端重启了，那么就读不到配置了。

2:mysql：可以，同reids，客户端需要定时去获取。

相比于zookeeper的watch，zookeeper的方式更简单，服务端主动去通知客户端。客户端不用主动维护一个线程。

# Zookeeper实现分布式锁

## 原理：

1:CountDownlatch阻塞，放行

2:使用create –s –e实现公平的锁

## 步骤：

1:trylock

create –s –e；

watchCreateCallback

countDownLatch.await;

2:createCallback

成功-> getChildren

3:getChildrenCallback

3-1:是第一个countDownLatch.countDown()

3-2:不是第一个 判断上一个是否exists，并且注册watchr

4:

4-1: existscallback：不存在则getChildren；存在不做任何事

4-2:上一个节点删除时，回调getChildrenCallback注册的watcher

# 精益求精

使用Zookeeper实现一个可重入锁