**Zookeeper简单介绍：**

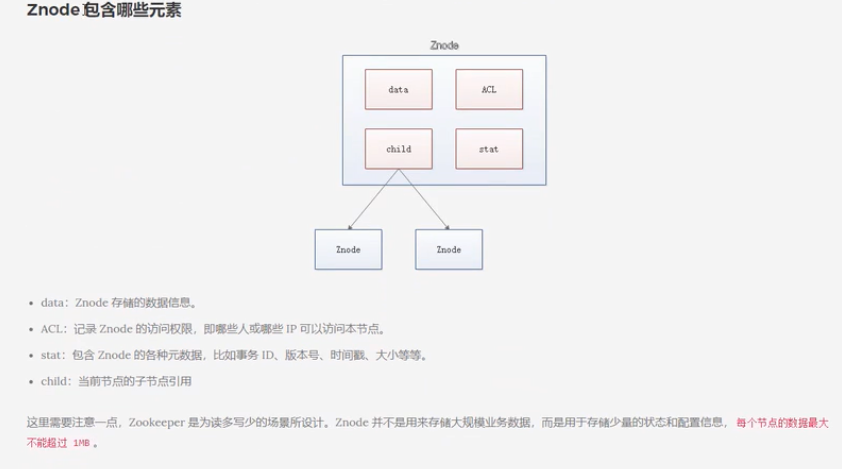
**基于树的数据结构组成： 节点znode;**

**对于不同树的节点，znode的引用方式是路径引用；类似于文件引用；**

**/动物/小狗**

**/汽车/路虎/发送机**

### Znode 单个节点的元素构成:

****

**注意：**

1. **zookeeper是为写少读多的场景设计的**
2. **Znode 不能够存储大量数据，1M为限制；**

**ZOOKEEPER的基本操作：**

1. **Create 创建节点**
2. **Delete 删除节点**
3. **Exist判断节点是否存在**
4. **getData获得一个数据的节点**
5. **setData设置一个节点的数据**
6. **获得节点下的所有的子节点；**

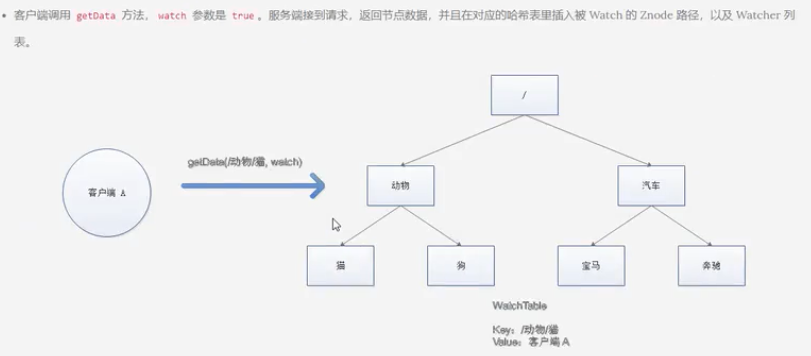
**Zookeeper客户端在设置读操作的时候，可以选择设置watch；**

### Zookeeper的事件通知：

**Watch为注册到特定znode 上的触发器；**

**当这个znode发生改变，调用了create delete，setData方法的时候，将会触发Znode上的注册的对应的事件； 之前对这个节点设置过监控的客户端会接收到响应的异步通知；**

1. **zookeeper中的其中一个znode注册watcher ；**

****

1. **如果这个特定节点被删除，则异步通知之前在这个节点上注册的watcher：**

****

**ZKClient在客户端维护了一个列表；记录树的节点地址和相对应的ip地址；**

**同时注册了watcher ，如果这台服务宕机，同时会通知注册了watcher的客户端的；**

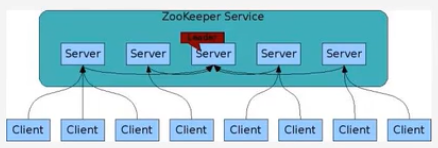
**Eureka的注册中心的维护是一个双向map存储；**

**Zookeeper则是树形结构存储；**

**Zookeeper 有变更只向注册此节点的watcher反馈--所以说zk还要维护一个watcher-table**

**Eureka 向所有的fetch的客户端反馈；**

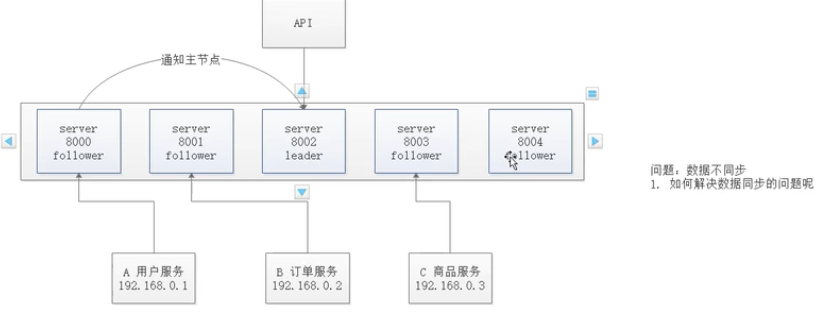
**Zookeeper的集群模式：**

****

**集群模式是一主多从机制：**

**Leader： 大哥**

**Flower： 小弟**

****

**进行了读写分离：**

**主从节点 : 只有大哥才能进行写操作哦**

**特殊的是：　小弟在接受到写入数据请求之后，必须先要告诉大哥；去写数据，然后再去　同步到其他从节点；－－需要Zookeeper　Atomic　BroadCast　（ZAB）协议；利用这个协议来完成数据同步的效果；**

**读：　读的话从任意节点都可以进行读取；**

**ZAB协议的作用有哪些？**

1. **控制了ZK集群（主节点）崩溃恢复的原理；**
2. **解决了主从数据同步的问题**

**ZAB协议定义的三种节点（代表了一个Zookeeper服务端）的状态：**

1. **Looking　：　选举状态**
2. **Following：　从节点所属的状态；**
3. **Leading：　leader节点所属的状态；**

**最大ZXID的概念：　代表了此节点的最新事务编号；**

**ZAB的崩溃恢复实现：**

1. **所有的节点都属于Looking状态，然后他们会像其他节点发起投票，投票当中包含自己的服务器ID和最新事务（ZXID）；**
2. **进行判定是否投票成功，A-》B，if（A.zxid < B.zxid ） 这样的话投票成功。不然进行重新选取其他的节点进行投票；（没有给他投票的节点进行淘汰）**
3. **这样循环直到某个节点得到半数以上的投票；这样的节点被选定为预定leader ;**
4. **如果由于网络问题出现两个leader的情况；这里要结合从节点重新进行选出最大的zxid，然后 同步自己的历史事务日志，**
5. **同步阶段，把leader收集来的最新的历史事务日志，同步给集群中所有的Follower ，只有当半数的Follower同步成功。这个准leader就程为真正leader；**

**Zookeeper如何确保数据一致性呢？**

1. **client发送写入数据到任意的一个Follower。**
2. **Follower把写入数据的请求转发给主节点；**
3. **然后主节点采取类似于事务的二阶段提交的方式；先把propose（提议）广播给Followers；**
4. **Followers接受到消息，写入日志成功后，返回ACK消息给leader；**
5. **Leader接到半数以上的ack，就返回成功给客户端，然后再发送commit广播给followers；**

**数据的一致性：**

**强一致性：**

**弱一致性：**

**顺序一致性： zookeeper利用了二阶段提交的方式；**

**Zookeeper的应用场景：**

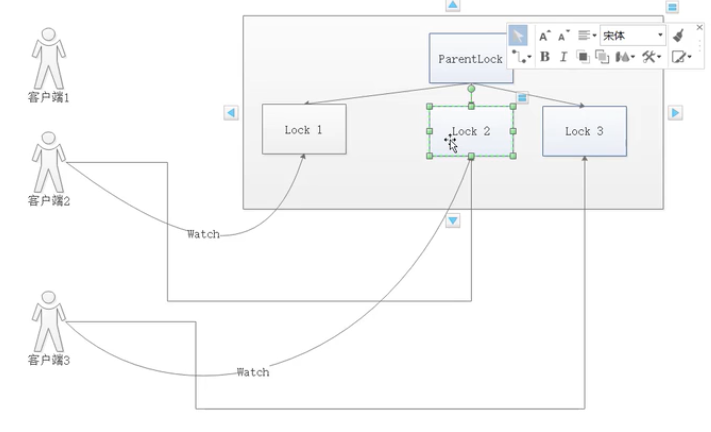
1. **分布式锁：雅虎设计的初衷就是为分布式锁而设计的；利用临时顺序节点；**
2. **服务的注册和发现，阿里的分布式框架dubbo；**
3. **共享配置和状态信息，维护一些组件集群的高可用，redis，kafka，hadoop；**

**Znode 节点的四种类型：**

1. **持久化节点；默认的节点类型；创建节点的客户端与zk断开后，该节点任然存在；持久化到当底的硬盘的；**
2. **持久节点顺序节点；**

**所谓的顺序节点，就是按照阶段的创建的事件顺序对节点进行编号；**

1. **临时节点：**
2. **临时顺序节点；**

**Zookeeper锁： **

**不需要设置超时机制： 因为临时节点会维护一个心跳，如果连接点宕机就会自动释放；**

**ZK的部署:**

1. **任何一个集群的部署数都为大于等于3的基数；**
2. **在局域网里面还可以用计算机名来访问；**

**Zk配置文件的解析：**

## **Zookeeper 的三种工作模式**

* 单机模式：存在单点故障
* 集群模式：在多台机器上部署 Zookeeper 集群，适合线上环境使用。
* 伪集群模式：在一台机器同时运行多个 Zookeeper 实例，仍然有单点故障问题，当然，其中配置的端口号要错开的，适合实验环境模拟集群使用。

## **Zookeeper 的三种端口号**

* 2181：客户端连接 Zookeeper 集群使用的监听端口号
* 3888：选举 leader 使用
* 2888：集群内机器通讯使用（Leader 和 Follower 之间数据同步使用的端口号，Leader 监听此端口）

## **Zookeeper 单机模式配置文件**

配置文件路径：/conf/zoo.cfg

clientPort=2181

dataDir=/data

dataLogDir=/datalog

tickTime=2000

**Zookeeper 集群模式配置文件**

* clientPort：这个端口就是客户端连接 Zookeeper 服务器的端口
* dataDir：Zookeeper 保存数据的目录。
* dataLogDir：Zookeeper 保存日志的目录。
* tickTime：一次心跳的时间间隔。

配置文件路径：/conf/zoo.cfg

clientPort=2181

dataDir=/data

dataLogDir=/datalog

tickTime=2000

initLimit=5 //连接到集群的初始化时间； 单位为 一个心跳间隔

syncLimit=2 //leader和follow之间请求和应答的时间间隔 单位：一个心跳间隔

autopurge.snapRetainCount=3

autopurge.purgeInterval=0

maxClientCnxns=60 //限制连接到zookeeper的连接数量

server.1=192.168.0.1:2888:3888

server.2=192.168.0.2:2888:3888

server.3=192.168.0.3:2888:3888

server.A=B：C：D：其中 A 是一个数字，表示这个是第几号服务器。B 是这个服务器的 IP 地址。C 表示的是这个服务器与集群中的 Leader 服务器交换信息的端口(2888)；D 表示的是万一集群中的 Leader 服务器挂了，需要一个端口来重新进行选举，选出一个新的 Leader，而这个端口就是用来执行选举时服务器相互通信的端口(3888)。如果是伪集群的配置方式，由于 B 都是一样，所以不同的 Zookeeper 实例通信端口号不能一样，所以要给它们分配不同的端口号。

注意： server.A 中的 A 是在 dataDir 配置的目录中创建一个名为 myid 的文件里的值

**Zk多种集群的配置方式：**

<https://www.jianshu.com/p/c486133a70e4>