ZK采用的是观察者模式，服务端向ZK中注入文件，然后客户端观察ZK，当ZK中的数据改变时通知客户端，所以ZK=文件存储 + 通知机制

ZK集群(Leader/Follower)，集群要保证半数以上的节点存数，集群才可用

集群中的每个客户端保存的数据都一样的

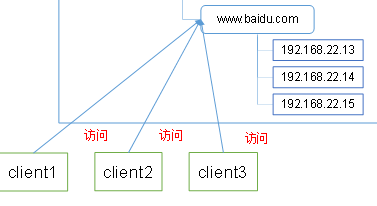
客户端对ZK的写操作是顺序执行的，并且具有原子性，要么成功，要么失败

ZK存储的文件数据量非常少，所以能在一定时间范围内保证客户端读取的数据是最新的

ZK存数数据采用的树形结构，每个ZNode默认能够存储1MB的数据，ZK将数据存储在ZNODE中

使用场景：

统一命名服务：



客户端读取baidu节点，获取数据

统一配置管理：

分布式环境下，每个节点的配置文件都是一样的，可以将配置文件写到ZK中，每个节点客户端监听ZNODE，一旦数据被修改，节点从ZK中读取

统一集群管理：

分布式环境中需掌握每个节点的状态，可以将节点的信息写到ZNODE中，然后监听这个ZNODE

服务器节点动态上下线：

软负载均衡：

安装：

在ZK的bin目录下有两个常用的脚本zkServer.sh/zkCli.sh

zkServer.sh start用于启动ZK服务端，

zkServer.sh status查看ZK的状态，

zkServer.sh stop关闭ZK服务端

zkCli.sh用于启动ZK客户端，主要查看ZK的节点信息

配置文件：

tickTime =2000：通信心跳数，Zookeeper服务器与客户端心跳时间，单位毫秒，每个tickTime时间就会发送一个心跳，检测服务端与客户端连接是否正常，响应时间是2倍tickTime(来回)

initLimit =10：LF初始通信时限，ZK集群在启动时，Leader和Follower之间初始连接时的时间，10个tickTime

syncLimit =5：LF同步通信时限，ZK集群启动后，Leader和Follower响应时间超过syncLimit个tickTime，Leader会认为Follower死掉，将从集群中移出

dataDir：存储数据文件目录以及数据持久化的路径，一般安装完成后，新建dataDir目录存储文件

ZNODE类型：ZNODE是否可持久化

持久：客户端与服务器断开后，节点不会删除

短暂：客户端与服务器断开后，节点会删除

而节点又分为是否编号：节点创建后，会自动在后面生成一个0000n的编号(单调递增，由父节点维护)

这个编号可以说明节点创建的顺序

每个ZONDE中都包含一个Stat结构体，有一下信息：

czxid：

修改ZNODE状态时，会收到一个zxid形式的时间戳，也就是ZooKeeper事务ID。

事务ID是ZooKeeper中所有修改总的次序。每个修改都有唯一的zxid，如果zxid1小于zxid2，那么zxid1在zxid2之前发生。

ctime：

znode被创建的毫秒数

mzxid：

znode最后更新的事务zxid

mtime：

znode最后修改的毫秒数(从1970年开始)

pZxid：

znode最后更新的子节点zxid

cversion：

znode子节点变化号，znode子节点修改次数

dataversion：

znode数据变化号

aclVersion：

znode访问控制列表的变化号

ephemeralOwner：

如果是临时节点，这个是znode拥有者的session id。如果不是临时节点则是0。

dataLength：

znode的数据长度

numChildren：

znode子节点数量

主要是后两个属性

集群：

1. 在ZKData下创建myid(必须这样命名)文件
2. 编辑myid文件，设置集群中server的编号
3. 在zoo.cfg文件后添加如下配置：

#######################cluster##########################

server.2=hadoop102:2888:3888

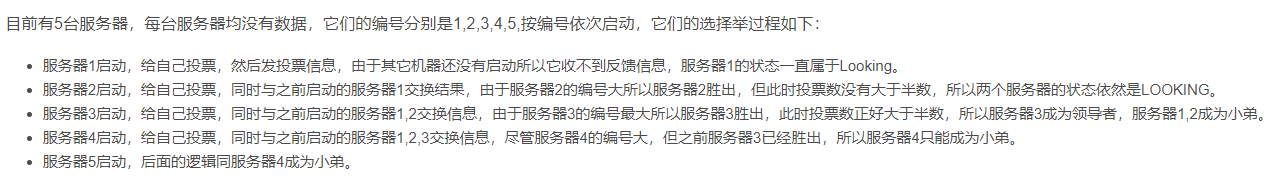
server.3=hadoop103:2888:3888

server.4=hadoop104:2888:3888

server.A=B:C:D

A为每个集群中myid中的编号，B为对应节点的ip地址，C为F与L进行数据信息交换端口，D为万一集群中L挂了，通过D端口投票选举

选举机制：



客户端命令：

help：查看所有命令

ls /：查看指定路径下的节点名称

ls2 /：查看指定路径下的节点的Stat信息

create /路径/newNode “value”：在指定路径下创建newNode子节点，节点存储的值为value，ZK规定，在创建节点时，必须写入内容

get /路径：获取指定路径下节点的值 + Stat信息

create –e /路径/newNode “value”：创建临时节点

create –s /路径/newNode “value”：创建带序号的节点

set /路径/node “newValue”：修改node的属性值

delete /路径/node：删除指定路径下的node

rmr /路径/node：递归删除

stat /路径/node：查看节点状态

监听：

get /路径/node watch：监听指定路径下node的状态，当集群中有对此node的值进行修改，监听将会被触发，一般使用process函数处理监听被触发后的操作，但是监听是会被监听一次，下次再修改时，监听将会失效。

ls /路径/node watch：监听路径，和监听节点类似

写操作过程：ZK接收到客户端的写请求后，如果ZK不是leader，会将请求转发给leader，而leader会将写请求广播给所以follower，follower写成功后会返回通知给leader，只有当大多数leader的写成功请求后，采用返回给客户端写成功。

Node权限：

对节点的操作权限有：CREATE、READ、WRITE、DELETE、ADMIN 也就是 增、删、改、查、管理权限，5种权限缩写：crwda(delete是对子节点的权限，其他的4种指对自身的权限)

客户端认证的方式：

World：都可以访问，默认

Auth：通过认证的用户

Digest：即用户名：密码认证方式

Ip：使用ip地址认证

新建节点默认是：world，cdrwa

ZK中不会继承父节点权限

命令：

getAcl path ：读取节点权限

setAcl path acl ：设置节点权限

addauth 认证方式 auth：添加认证用户

world：

create /node value

setAcl /node world:anyone: crwda

ip：

create /node value

setAcl /node ip:192.168.187.238:crwda

只有指定ip的ZK客户端才能访问此节点，操作权限是crwda

Auth：

create /node value

addauth digest josiah:123456

setAcl /node auth:Josiah:crwda

当其他客户端要访问时，必须执行addauth digest josiah:123456才能获取到节点

create /node value

setAcl /node4 digest:josiah:UvJWhBril5yzpEiA2eV7bwwhfLs=:cdrwa

这种方式和auth类型，只是auth要先addauth才执行setAcl，这里直接一步完成，这里的UvJWhBril5yzpEiA2eV7bwwhfLs=指123456加密后的密文(必须是密文)，访问此节点之前要先认证addauth digest josiah:123456

API使用：见代码