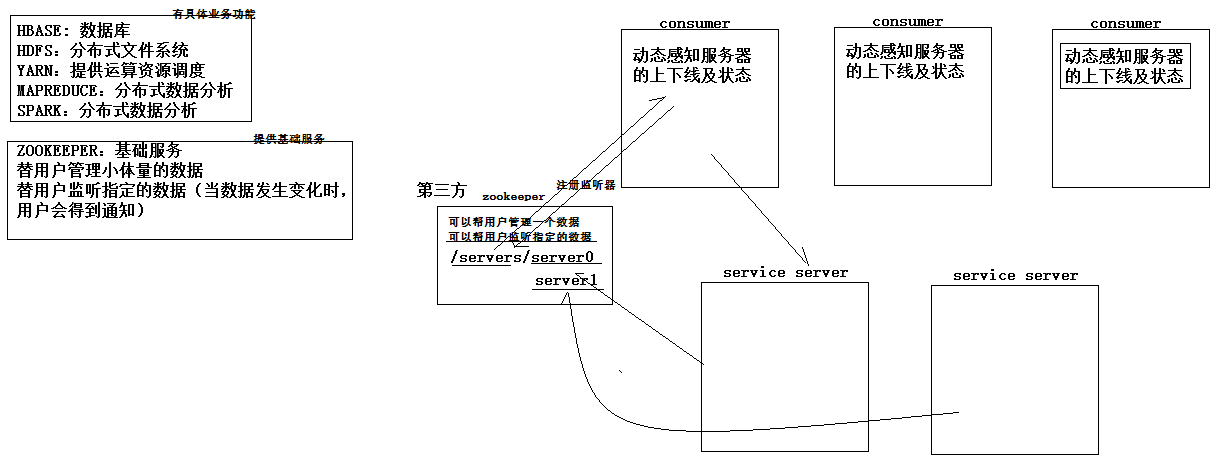
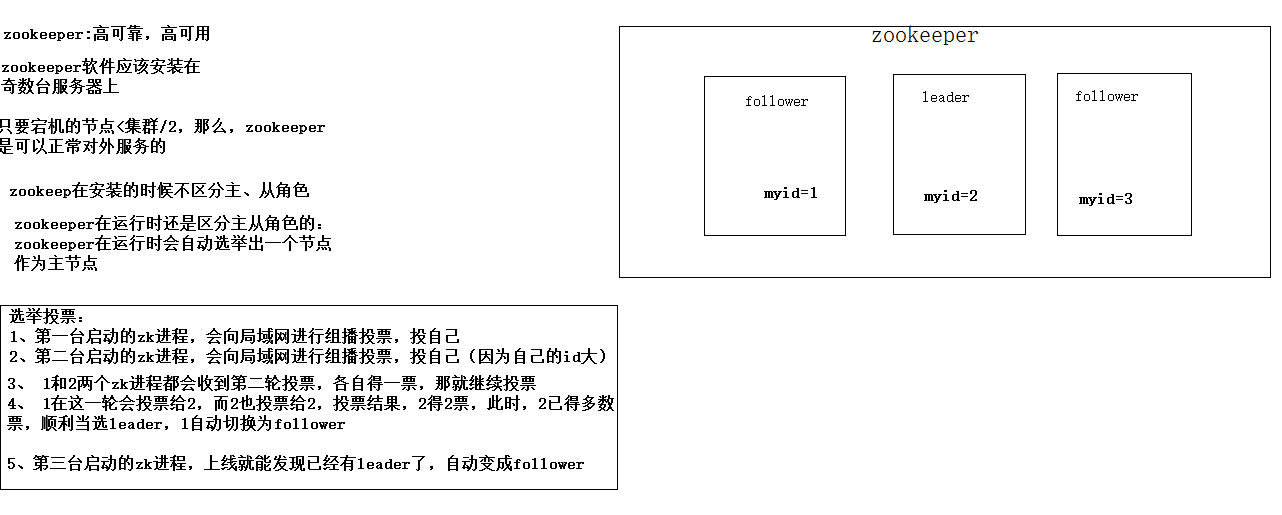
## zookeeper的基本功能和应用场景



## zookeeper的整体运行机制



## zookeeper的数据存储机制

### 数据存储形式

zookeeper中对用户的数据采用kv形式存储

只是zk有点特别：

key：是以路径的形式表示的，那就以为着，各key之间有父子关系，比如

/ 是顶层key

用户建的key只能在/ 下作为子节点，比如建一个key： /a 这个key可以带value数据

也可以建一个key： /b

也可以建key： /a/x

zookeeper中，对每一个数据key，称作一个znode

### znode类型

zookeeper中的znode有多种类型：

1. PERSISTENT 持久的：创建者就算跟集群断开联系，该类节点也会持久存在与zk集群中
2. EPHEMERAL 短暂的：创建者一旦跟集群断开联系，zk就会将这个节点删除
3. SEQUENTIAL 带序号的：这类节点，zk会自动拼接上一个序号，而且序号是递增的

组合类型：

PERSISTENT ：持久不带序号

EPHEMERAL ：短暂不带序号

PERSISTENT 且 SEQUENTIAL ：持久且带序号

EPHEMERAL 且 SEQUENTIAL ：短暂且带序号

## zookeeper的集群部署

1. 上传安装包到集群服务器
2. 解压
3. 修改配置文件

进入zookeeper的安装目录的conf目录

cp zoo\_sample.cfg zoo.cfg

vi zoo.cfg

|  |
| --- |
| # The number of milliseconds of each tick  tickTime=2000  initLimit=10  syncLimit=5  dataDir=/root/apps/zkdata  clientPort=2181  #autopurge.purgeInterval=1  server.1=node1:2888:3888  server.2=node2:2888:3888  server.3=node3:2888:3888 |

对3台节点，都创建目录 mkdir /root/apps/zkdata

对3台节点，在工作目录中生成myid文件，但内容要分别为各自的id： 1,2,3

Node1上： echo 1 > /root/apps/zkdata/myid

Node2上： echo 2 > /root/apps/zkdata/myid

Node3上： echo 3 > /root/apps/zkdata/myid

1. 从node1上scp安装目录到其他两个节点

scp -r zookeeper-3.4.6/ node2:$PWD

scp -r zookeeper-3.4.6/ node3:$PWD

1. 启动zookeeper集群

zookeeper没有提供自动批量启动脚本，需要手动一台一台地起zookeeper进程

在每一台节点上，运行命令：

bin/zkServer.sh start

启动后，用jps应该能看到一个进程：QuorumPeerMain

但是，光有进程不代表zk已经正常服务，需要用命令检查状态：

bin/zkServer.sh status

能看到角色模式：为leader或follower，即正常了。

## zookeeper的命令行客户端操作

### zookeeper的数据存储形式：

1. zookeeper中存储数据的基本形式为： key ， value
2. zookeeper中的key是用路径表示的：

/a: hello

/rr: 666

每一个key-value称为一个znode（zookeeper数据节点）

1. zookeeper中的数据节点有4种类型：
2. 持久节点：客户端一旦建立，zk会持久保存，除非有客户端手动删除
3. 短暂节点：创建这个节点的客户端一旦断开与zookeeper集群的联系，zookeeper集群就会自动将该节点删除
4. 带序号的节点：在同一个父节点下，建带序号的子节点，zk会自动给客户端指定的子节点名后拼接一个自增的序号
5. 不带序号的节点：

上述4中类型，可以有以下组合类型：

持久-带序号

持久-不带序号

短暂-带序号

短暂-不带序号

### 数据管理功能：

创建节点： create /3a xx

查看节点下的子节点： ls /3a

获取节点的value： get /3a

修改节点的value： set /3a kkk

删除节点：rmr /3a

### 数据监听功能：

ls /3a watch

## 查看/aaa的子节点的同时，注册了一个监听“节点的子节点变化事件”的监听器

get /3a watch

## 获取/3a 的value的同时，注册了一个监听“节点value变化事件”的监听器

注意这里只监听了一次