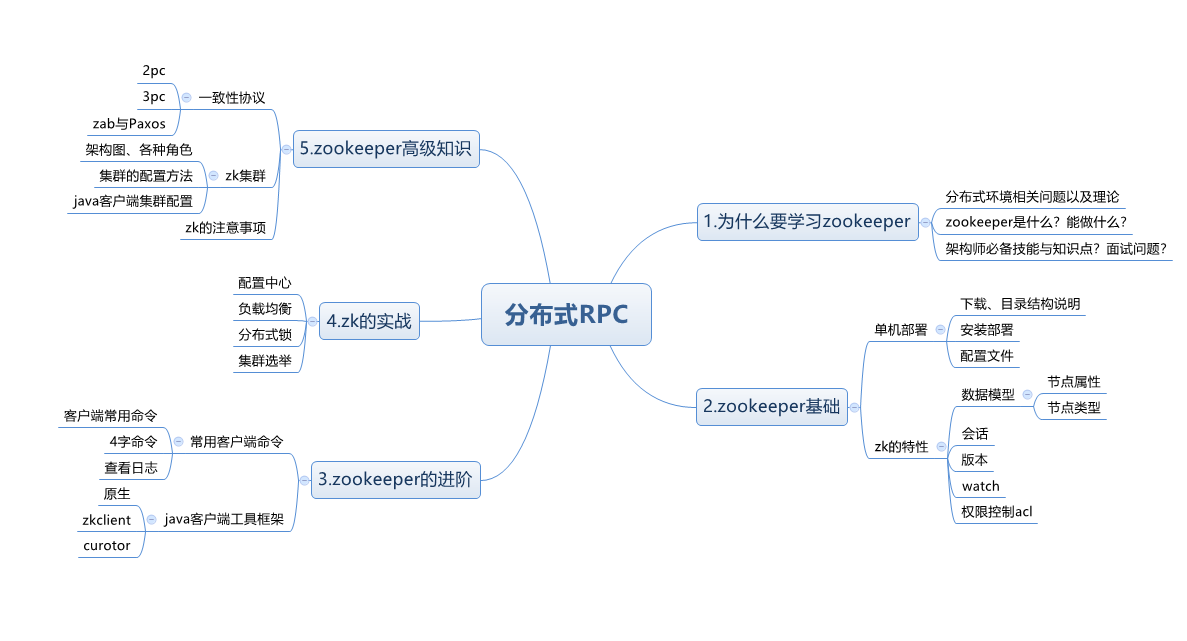
# Zookeeper综述

## 课程总体介绍



如上图所示，整个ZK体系会从入门开始，到基础，进阶以及实战，最后把相关的理论给介绍一下。

ZK在后面讲dubbo和kafka都会用到，请务必重视起来。

另外对zk了解比较深入的同学可以课后通过视频的方式加快学习进度。

### 为什么学习ZK

应该重点掌握分布式环境的演进过程，从一个单节点开始，慢慢过渡到分布式，为什么单节点不行，传统一个tomcat打天下有什么有点，缺点又是什么，当一个tomcat搞不定的时候，分布式的架构图又是什么样的，

传统的单节点架构自然有问题，到了分布式的架构中，问题肯定也有不少，这些问题就是我们学习ZK要解决的，但学习这些解决方案之前，还是需要有点理论基础。

接下来就要了解下什么是zk,为什么学习zk, Zk在分布式架构中扮演了什么样的角色。以及面试的时候经常会问到的问题，心里要有个大概的了解。

### zookeeper基础

了解zk是什么玩意后，接下来就把zk安装好，先来讲解的是zk单机部署，这个非常容易，而且后面课程绝大部的时间都是使用单节点来使用的，至于集群的配置放到后面来讲，正在在工作中，集群的维护应该是运维来做，哪怕没有运维，在后面通过详细的学习通过也能在20分钟内完全搭建起来，同学们不用着急。

Zk的特性：这块是重点之中的重点，后面学习的一切操作，包括实战，都是建立在这基础之上的。其中数据模型和watch机制又是最最重要的。

### zk进阶

安装好了zk,对zk基础有一定的了解后，接下来就学习怎么操作zk,首先了解基础zk客户端的使用。

客户端的简单使用能解决一些问题，方便查看信息，简单省事，但真正对于我们java架构师来说最重要的是java客户端，包括原生的，zkclient以及curotor，同学们最少要熟练使用其中的一种，另外的在工作中也要很快的百度得出来。

### zk的实战

一般来说zk实战是放到最后来讲得，但这一块确是面试得时候经常会碰到的，所以先提前，这一块是整个课程的重点，我会带同学一行行的代码来学习，也请同学务必能够跟着deer老师一个个案例吃透。

会讲到配置中心，负载均衡，分布式锁，集群选举，尤其是负载均衡，如果这块没听懂peter老师会找我麻烦的，后面他讲dubbo的核心就是这玩意，至于具体的请到后面课程中收听。

### zk高级知识

这一块不需要同学完全听懂，掌握40%-50%就够了，工作中很少用到，但确实面试的时候的加分项，

包括2pc,3pc提交协议，集群的安装等等

### 总结：

不管怎么样，其中的zk基础，zk进阶客户端的使用，以及zk项目实战请务必重视起来。

## 分布式系统基础知识

一个tomcat打天下的时代，不能说完全淘汰了，在一个管理系统，小型项目中还经常使用，这并不过分，出于成本的考虑，这反而值得提倡。

但同学们最终是要成为架构师，架构师就必然要了解分布式系统：

### 分布式系统是什么

分布式系统：一个硬件或软件组件分布在不同的网络计算机上，彼此之间仅仅通过消息传递进行通信和协调的系统

这是分布式系统，在不同的硬件，不同的软件，不同的网络，不同的计算机上，仅仅通过消息来进行通讯与协调

这是他的特点，更细致的看这些特点又可以有：分布性、对等性、并发性、缺乏全局时钟、

故障随时会发生。

#### 分布性

既然是分布式系统，最显著的特点肯定就是分布性，从简单来看，如果我们做的是个电商项目，整个项目会分成不同的功能，专业点就不同的微服务，比如用户微服务，产品微服务，订单微服务，这些服务部署在不同的tomcat中，不同的服务器中，甚至不同的集群中，整个架构都是分布在不同的地方的，在空间上是随意的，而且随时会增加，删除服务器节点，这是第一个特性

#### 对等性

对等性是分布式设计的一个目标，还是以电商网站为例，来说明下什么是对等性，要完成一个分布式的系统架构，肯定不是简单的把一个大的单一系统拆分成一个个微服务，然后部署在不同的服务器集群就够了，其中拆分完成的每一个微服务都有可能发现问题，而导致整个电商网站出现功能的丢失。

比如订单服务，为了防止订单服务出现问题，一般情况需要有一个备份，在订单服务出现问题的时候能顶替原来的订单服务。

这就要求这两个（或者2个以上）订单服务完全是对等的，功能完全是一致的，其实这就是一种服务副本的冗余。

还一种是数据副本的冗余，比如数据库，缓存等，都和上面说的订单服务一样，为了安全考虑需要有完全一样的备份存在，这就是对等性的意思。

#### 并发性

并发性其实对我们来说并不模式，在学习多线程的时候已经或多或少学习过，多线程是并发的基础。

但现在我们要接触的不是多线程的角度，而是更高一层，从多进程，多JVM的角度，例如在一个分布式系统中的多个节点，可能会并发地操作一些共享资源，如何准确并高效的协调分布式并发操作。

后面实战部分的分布式锁其实就是解决这问题的。

#### 缺乏全局时钟

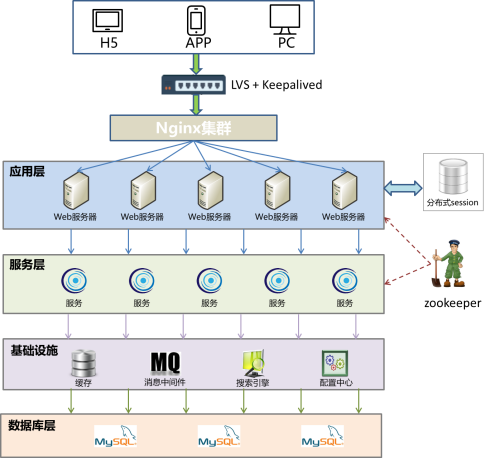
在分布式系统中，节点是可能反正任意位置的，而每个位置，每个节点都有自己的时间系统，因此在分布式系统中，很难定义两个事务纠结谁先谁后，原因就是因为缺乏一个全局的时钟序列进行控制，当然，现在这已经不是什么大问题了，已经有大把的时间服务器给系统调用

#### 故障随时会发生

任何一个节点都可能出现停电，死机等现象，服务器集群越多，出现故障的可能性就越大，随着集群数目的增加，出现故障甚至都会成为一种常态，怎么样保证在系统出现故障，而系统还是正常的访问者是作为系统架构师应该考虑的。

### 大型网站架构图回顾

知道什么是分布式系统，接下来具体来看下大型网站架构图，这个图在前面分布式架构演进应该已经讲过，首先整个架构分成很多个层，应用层，服务层，基础设施层与数据服务层，每一层都由若干节点组成，这是典型的分布式架构，后面一大把的时间就是系统的学习里面的每一个部分



那么zookeeper在其中又是扮演什么角色呢，如果可以把zk扮演成交警的角色，而各个节点就是马路上的各种汽车（汽车，公交车），为了保证整个交通（系统）的可用性，zookeeper必须知道每一节点的健康状态（公交车是否出了问题，要派新的公交【服务注册与发现】），公路在上下班高峰是否拥堵，在某一条很窄的路上只允许单独一个方向的汽车通过【分布式锁】。

如果交通警察是交通系统的指挥官，而zookeeper就是各个节点组成分布式系统的指挥官。

#### 分布式系统协调“方法论”

* + - * 1. 分布式系统带来的问题

如果把分布式系统和平时的交通系统进行对比，哪怕再稳健的交通系统也会有交通事故，分布式系统也有很多需要攻克的问题，比如：通讯异常，网络分区，三态，节点故障等。

通信异常

通讯异常其实就是网络异常，网络系统本身是不可靠的，由于分布式系统需要通过网络进行数据传输，网络光纤，路由器等硬件难免出现问题。只要网络出现问题，也就会影响消息的发送与接受过程，因此数据消息的丢失或者延长就会变得非常普遍。

网络分区

网络分区，其实就是脑裂现象，本来有一个交通警察，来管理整个片区的交通情况，一切井然有序，突然出现了停电，或者出现地震等自然灾难，某些道路接受不到交通警察的指令，可能在这种情况下，会出现一个零时工，片警零时来指挥交通。

但注意，原来的交通警察其实还在，只是通讯系统中断了，这时候就会出现问题了，在同一个片区的道路上有不同人在指挥，这样必然引擎交通的阻塞混乱。

这种由于种种问题导致同一个区域（分布式集群）有两个相互冲突的负责人的时候就会出现这种精神分裂的情况，在这里称为脑裂，也叫网络分区。

三态

三态是什么？三态其实就是成功，与失败以外的第三种状态，当然，肯定不叫变态，而叫超时态。

在一个jvm中，应用程序调用一个方法函数后会得到一个明确的相应，要么成功，要么失败，而在分布式系统中，虽然绝大多数情况下能够接受到成功或者失败的相应，但一旦网络出现异常，就非常有可能出现超时，当出现这样的超时现象，网络通讯的发起方，是无法确定请求是否成功处理的。

节点故障

这个其实前面已经说过了，节点故障在分布式系统下是比较常见的问题，指的是组成服务器集群的节点会出现的宕机或“僵死”的现象，这种现象经常会发生。

* + - * 1. CAP理论

前面花费了很大的篇幅来了解分布式的特点以及会碰到很多会让人头疼的问题，这些问题肯定会有一定的理论思想来解决问题的。

接下来花点时间来谈谈这些理论，其中CAP和BASE理论是基础，也是面试的时候经常会问到的

首先看下CAP，CAP其实就是一致性，可用性，分区容错性这三个词的缩写

一致性

一致性是事务ACID的一个特性【原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）、持久性（Durability）】，学习数据库优化的时候deer老师讲过。

这里讲的一致性其实大同小异，只是现在考虑的是分布式环境中，还是不单一的数据库。

在分布式系统中，一致性是数据在多个副本之间是否能够保证一致的特性，这里说的一致性和前面说的对等性其实差不多。如果能够在分布式系统中针对某一个数据项的变更成功执行后，所有用户都可以马上读取到最新的值，那么这样的系统就被认为具有【强一致性】。

可用性

可用性指系统提供服务必须一直处于可用状态，对于用户的操作请求总是能够在有限的时间内访问结果。

这里的重点是【有限的时间】和【返回结果】

为了做到有限的时间需要用到缓存，需要用到负载，这个时候服务器增加的节点是为性能考虑；

为了返回结果，需要考虑服务器主备，当主节点出现问题的时候需要备份的节点能最快的顶替上来，千万不能出现OutOfMemory或者其他500，404错误，否则这样的系统我们会认为是不可用的。

分区容错性

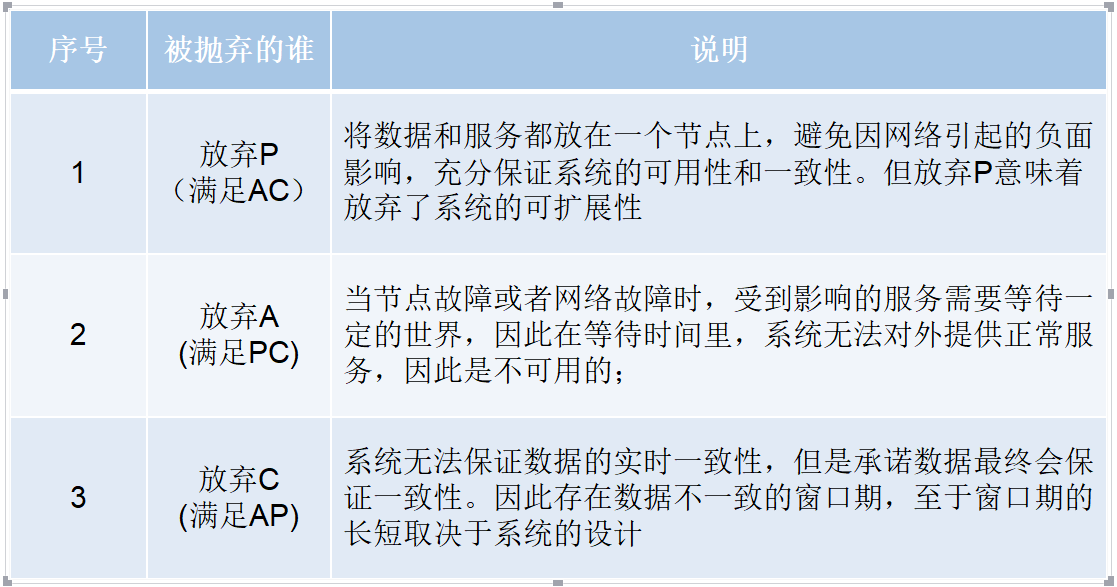
分布式系统在遇到任何网络分区故障的时候，仍然需要能够对外提供满足一致性和可用性的服务，除非是整个网络环境都发生了故障。

不能出现脑裂的情况

具体描述

来看下CAP理论具体描述：

**一个分布式系统不可能同时满足一致性、可用性和分区容错性这三个基本需求，最多只能同时满足其中的两项**



TIPS：不可能把所有应用全部放到一个节点上，因此架构师的精力往往就花在怎么样根据业务场景在A和C直接寻求平衡；

* + - * 1. BASE理论

根据前面的CAP理论，架构师应该从一致性和可用性之间找平衡，系统短时间完全不可用肯定是不允许的，那么根据CAP理论，在分布式环境下必然也无法做到强一致性。

BASE理论：即使无法做到强一致性，但分布式系统可以根据自己的业务特点，采用适当的方式来使系统达到最终的一致性；

Basically Avaliable 基本可用

当分布式系统出现不可预见的故障时，允许损失部分可用性，保障系统的“基本可用”；体现在“时间上的损失”和“功能上的损失”；

e.g：部分用户双十一高峰期淘宝页面卡顿或降级处理；

Soft state 软状态

其实就是前面讲到的三态,既允许系统中的数据存在中间状态，既系统的不同节点的数据副本之间的数据同步过程存在延时，并认为这种延时不会影响系统可用性；

e.g：12306网站卖火车票，请求会进入排队队列；

Eventually consistent 最终一致性

所有的数据在经过一段时间的数据同步后，最终能够达到一个一致的状态；

e.g：理财产品首页充值总金额短时不一致；

## Zookeeper简介

### Zookeeper简介（what）

ZooKeeper致力于提供一个高性能、高可用，且具备严格的顺序访问控制能力的分布式协调服务，是雅虎公司创建，是Google的Chubby一个开源的实现，也是Hadoop和Hbase的重要组件。

#### 设计目标

* 简单的数据结构：共享的树形结构，类似文件系统，存储于内存；
* 可以构建集群：避免单点故障，3-5台机器就可以组成集群，超过半数正常工作就能对外提供服务；
* 顺序访问：对于每个写请求，zk会分配一个全局唯一的递增编号，利用这个特性可以实现高级协调服务；
* 高性能：基于内存操作，服务于非事务请求，适用于读操作为主的业务场景。3台zk集群能达到13w QPS；1

### 哪些常见需要用到ZK（why）

数据发布订阅

负载均衡

客户端负载 zk 、

服务端负载 ngnix

命名服务

Master选举

集群管理

配置管理

分布式队列

分布式锁

### 为什么要学习zookeeper？（why）

互联网架构师必备技能

高端岗位必考察的知识点

zk面试问题全解析

* Zookeeper是什么框架
* 应用场景
* Paxos算法& Zookeeper使用协议
* 选举算法和流程
* Zookeeper有哪几种节点类型
* Zookeeper对节点的watch监听通知是永久的吗？
* 部署方式？集群中的机器角色都有哪些？集群最少要几台机器
* 集群如果有3台机器，挂掉一台集群还能工作吗？挂掉两台呢？
* 集群支持动态添加机器吗？

# Zookeeper基础

## 单基部署

先把ZK安装起来，后面的很多操作，都是的前提都是由ZK的操作环境，先来把ZK安装好，

### Zookeeper windows环境安装

环境要求:必须要有jdk环境,本次讲课使用jdk1.8

1.安装jdk

2.安装Zookeeper. 在官网[http://zookeeper.apache.org/](http://zookeeper.apache.org/" \t "_blank)下载zookeeper.我下载的是zookeeper-3.4.12版本。

解压zookeeper-3.4.6至D:\machine\zookeeper-3.4.12.

在D:\machine 新建data及log目录。

3.ZooKeeper的安装模式分为三种，分别为：单机模式（stand-alone）、集群模式和集群伪分布模式。ZooKeeper 单机模式的安装相对比较简单，如果第一次接触ZooKeeper的话，建议安装ZooKeeper单机模式或者集群伪分布模式。

安装单击模式。 至D:\machine\zookeeper-3.4.12\conf 复制 zoo\_sample.cfg 并粘贴到当前目录下，命名zoo.cfg.

### 目录结构

bin 存放系统脚本

conf 存放配置文件

contrib zk附加功能支持

dist-maven maven仓库文件

docs zk文档

lib 依赖的第三方库

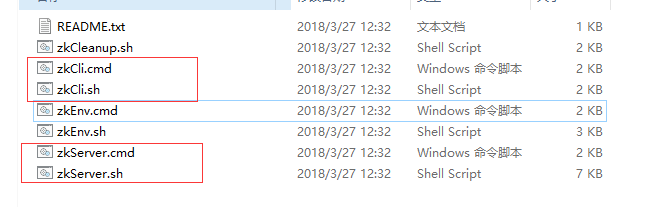
recipes 经典场景样例代码

src zk源码

其中bin和conf是非常重要的两个目录，平时也是经常使用的。

#### bin目录

先看下bin目录



其中

zkServer为服务器，启动后默认端口为2181

zkCli为命令行客户端

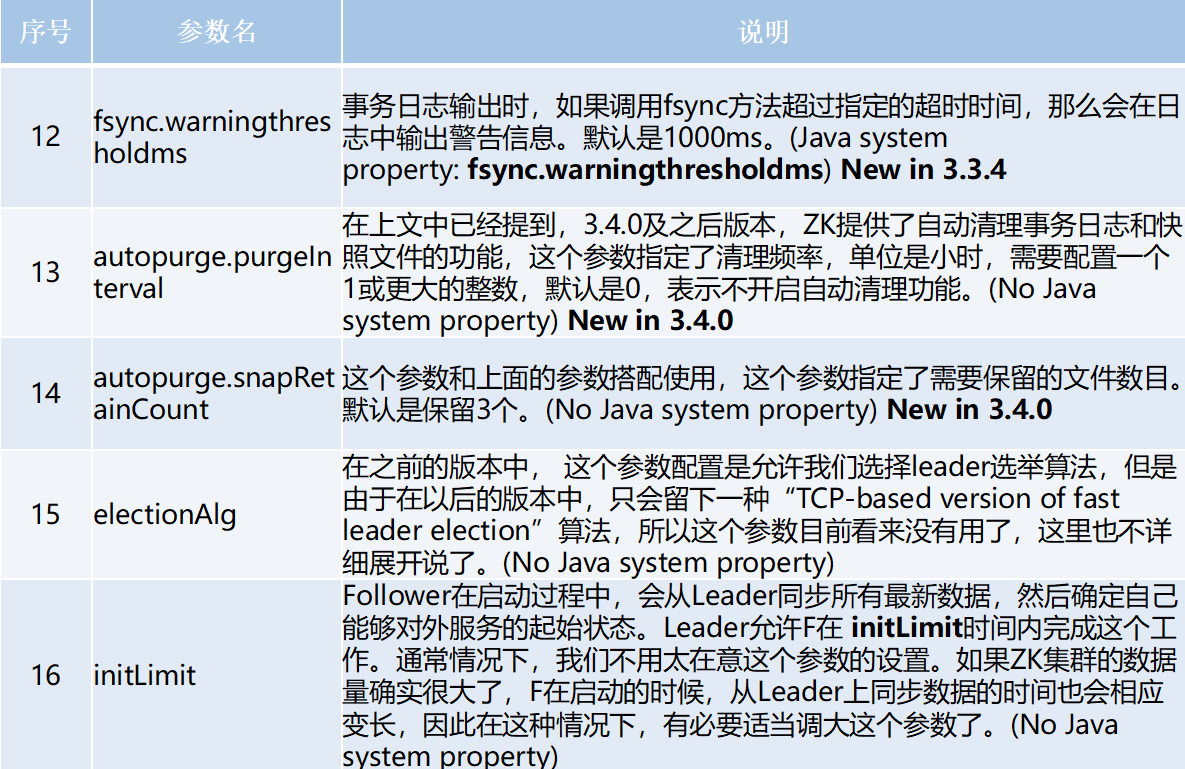
#### conf目录

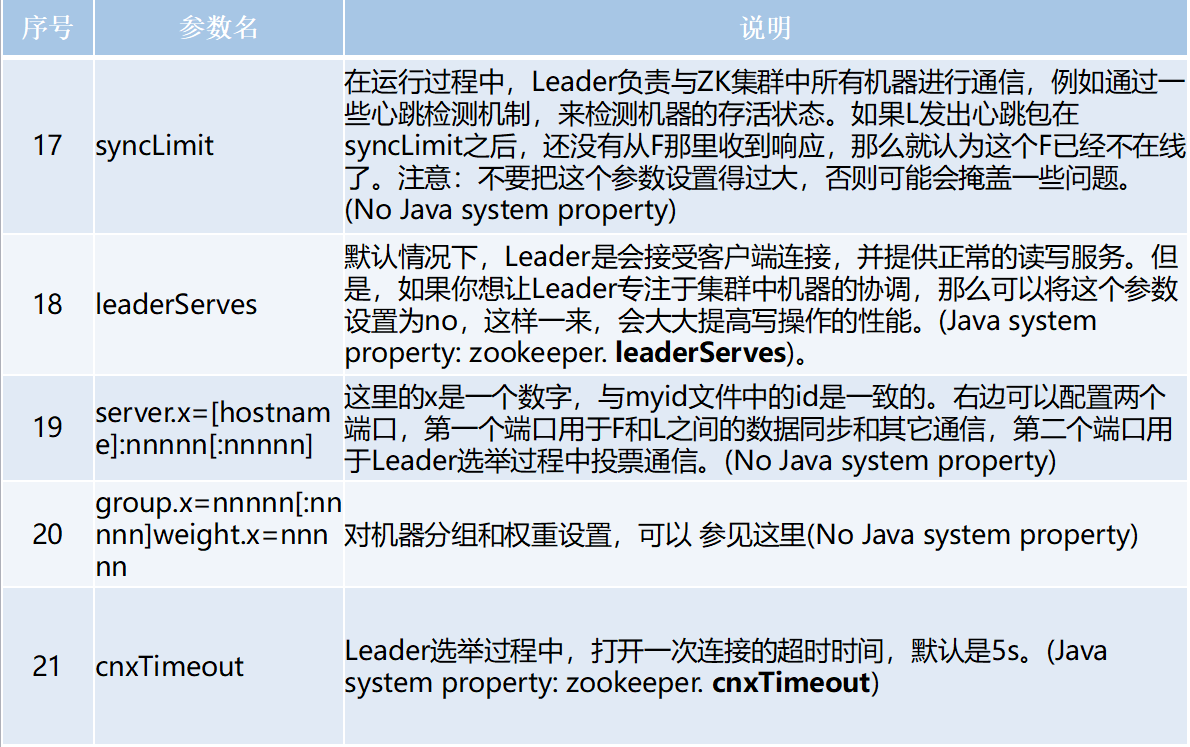
Conf目录为配置文件存放的目录，zoo.cfg为核心的配置文件

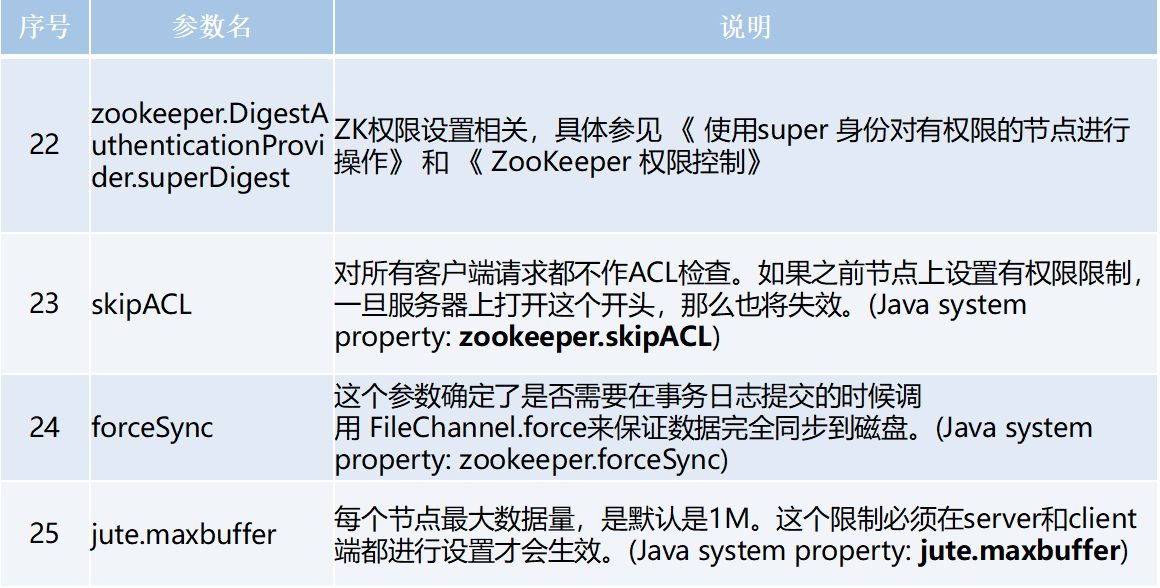
这里面的配置很多，这配置是运维的工作，目前没必要，也没办法全部掌握。











在这挑选几个讲解：

clientPort：参数无默认值，必须配置，用于配置当前服务器对外的服务端口，客户端必须使用这端口才能进行连接

dataDir：用于存放内存数据库快照的文件夹，同时用于集群的myid文件也存在这个文件夹里（注意：一个配置文件只能包含一个dataDir字样，即使它被注释掉了。）

dataLogDir：用于单独设置transaction log的目录，transaction log分离可以避免和普通log还有快照的竞争

dataDir：新安装zk这文件夹里面是没有文件的，可以通过snapCount参数配置产生快照的时机

以下配置集群中才会使用，后面再讨论

tickTime：心跳时间，为了确保连接存在的，以毫秒为单位，最小超时时间为两个心跳时间

initLimit：多少个心跳时间内，允许其他server连接并初始化数据，如果ZooKeeper管理的数据较大，则应相应增大这个值

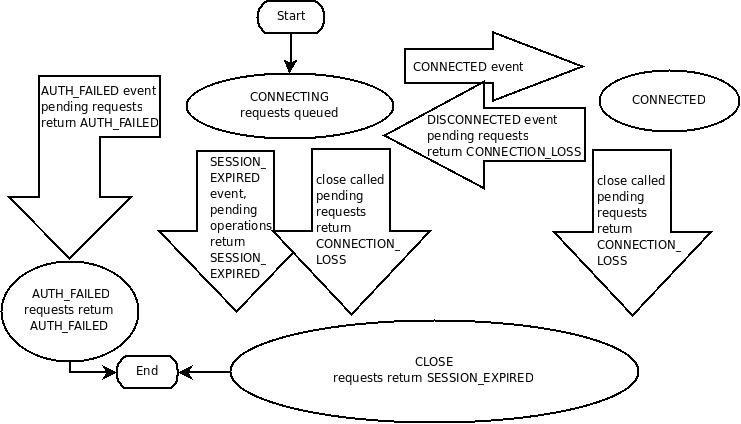
syncLimit：多少个tickTime内，允许follower同步，如果follower落后太多，则会被丢弃。

## ZK的特性

Zk的特性会从会话、数据节点，版本，Watcher，ACL权限控制，集群角色这些部分来了解，其中重点需要掌握的数据节点与Watcher

### 会话

客户端与服务端的一次会话连接，本质是TCP长连接，通过会话可以进行心跳检测和数据传输；



会话（session）是zookepper非常重要的概念，客户端和服务端之间的任何交互操作都与会话有关

**会话状态**

看下这图，Zk客户端和服务端成功连接后，就创建了一次会话，ZK会话在整个运行期间的生命周期中，会在不同的会话状态之间切换，这些状态包括：

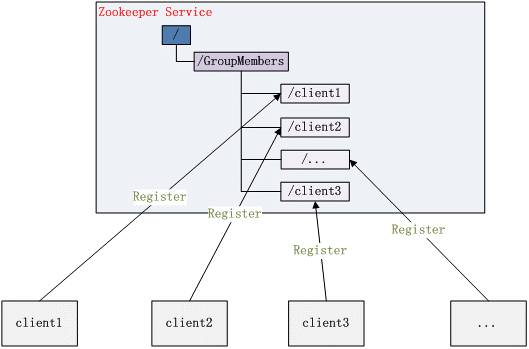
CONNECTING、CONNECTED、RECONNECTING、RECONNECTED、CLOSE

一旦客户端开始创建Zookeeper对象，那么客户端状态就会变成CONNECTING状态，同时客户端开始尝试连接服务端，连接成功后，客户端状态变为CONNECTED，通常情况下，由于断网或其他原因，客户端与服务端之间会出现断开情况，一旦碰到这种情况，Zookeeper客户端会自动进行重连服务，同时客户端状态再次变成CONNCTING，直到重新连上服务端后，状态又变为CONNECTED，在通常情况下，客户端的状态总是介于CONNECTING和CONNECTED之间。但是，如果出现诸如会话超时、权限检查或是客户端主动退出程序等情况，客户端的状态就会直接变更为CLOSE状态

### ZK数据模型

ZooKeeper的视图结构和标准的Unix文件系统类似，其中每个节点称为“数据节点”或ZNode,每个znode可以存储数据，还可以挂载子节点，因此可以称之为“树”

第二点需要注意的是，每一个znode都必须有值，如果没有值，节点是不能创建成功的。



* 在Zookeeper中，znode是一个跟Unix文件系统路径相似的节点，可以往这个节点存储或获取数据
* 通过客户端可对znode进行增删改查的操作，还可以注册watcher监控znode的变化。

### Zookeeper节点类型

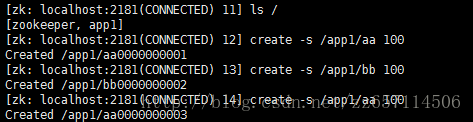
节点类型非常重要，是后面项目实战的基础。

a、Znode有两种类型：

短暂（ephemeral）（create -e /app1/test1 “test1” 客户端断开连接zk删除ephemeral类型节点）   
持久（persistent） （create -s /app1/test2 “test2” 客户端断开连接zk不删除persistent类型节点）

b、Znode有四种形式的目录节点（默认是persistent ）

PERSISTENT   
PERSISTENT\_SEQUENTIAL（持久序列/test0000000019 ）   
EPHEMERAL   
EPHEMERAL\_SEQUENTIAL

c、创建znode时设置顺序标识，znode名称后会附加一个值，顺序号是一个单调递增的计数器，由父节点维护   
       

d、在分布式系统中，顺序号可以被用于为所有的事件进行全局排序，这样客户端可以通过顺序号推断事件的顺序

### Zookeeper节点状态属性



### ACL保障数据的安全

ACL机制，表示为scheme:id:permissions，第一个字段表示采用哪一种机制，第二个id表示用户，permissions表示相关权限（如只读，读写，管理等）。

zookeeper提供了如下几种机制（scheme）：

**world:** 它下面只有一个id, 叫anyone, world:anyone代表任何人，zookeeper中对所有人有权限的结点就是属于world:anyone的

**auth:** 它不需要id, 只要是通过authentication的user都有权限（zookeeper支持通过kerberos来进行authencation, 也支持username/password形式的authentication)

**digest:** 它对应的id为username:BASE64(SHA1(password))，它需要先通过username:password形式的authentication

**ip:** 它对应的id为客户机的IP地址，设置的时候可以设置一个ip段，比如ip:192.168.1.0/16, 表示匹配前16个bit的IP段

现在看这可能懵懵懂懂，不过没有关系，等会在客户端操作的时候会有详细的操作

## 命令行

### 服务端常用命令

在准备好相应的配置之后，可以直接通过zkServer.sh 这个脚本进行服务的相关操作

启动ZK服务:       sh bin/zkServer.sh start 

查看ZK服务状态: sh bin/zkServer.sh status 

停止ZK服务:       sh bin/zkServer.sh stop 

重启ZK服务:       sh bin/zkServer.sh restart

### 客户端常用命令

使用 zkCli.sh -server 127.0.0.1:2181 连接到 ZooKeeper 服务，连接成功后，系统会输出 ZooKeeper 的相关环境以及配置信息。 命令行工具的一些简单操作如下：

* 显示根目录下、文件： ls / 使用 ls 命令来查看当前 ZooKeeper 中所包含的内容
* 显示根目录下、文件： ls2 / 查看当前节点数据并能看到更新次数等数据
* 创建文件，并设置初始内容： create /zk "test" 创建一个新的 znode节点“ zk ”以及与它关联的字符串  [-e] [-s] 【-e 零时节点】 【-s 顺序节点】
* 获取文件内容： get /zk 确认 znode 是否包含我们所创建的字符串  [watch]【watch 监听】
* 修改文件内容： set /zk "zkbak" 对 zk 所关联的字符串进行设置 
* 删除文件： delete /zk 将刚才创建的 znode 删除，如果存在子节点删除失败
* 递归删除：rmr /zk将刚才创建的 znode 删除，子节点同时删除
* 退出客户端： quit 
* 帮助命令： help

### ACL命令常用命令

再回过头来看下ACL权限

Zookeeper的ACL(Access Control List)，分为三个维度：scheme、id、permission

通常表示为：scheme:id:permission

* schema:代表授权策略
* id:代表用户
* permission:代表权限

#### Scheme

world：

默认方式，相当于全世界都能访问

auth：

代表已经认证通过的用户(可以通过addauth digest user:pwd 来添加授权用户)

digest：

即用户名:密码这种方式认证，这也是业务系统中最常用的

ip：

使用Ip地址认证

#### id

id是验证模式，不同的scheme，id的值也不一样。

scheme为auth时：

username:password

scheme为digest时:  
 username:BASE64(SHA1(password))

scheme为ip时:  
 客户端的ip地址。

scheme为world时  
 anyone。

#### Permission

CREATE、READ、WRITE、DELETE、ADMIN 也就是 增、删、改、查、管理权限，这5种权限简写为crwda(即：每个单词的首字符缩写)

CREATE(c)：创建子节点的权限

DELETE(d)：删除节点的权限

READ(r)：读取节点数据的权限

WRITE(w)：修改节点数据的权限

ADMIN(a)：设置子节点权限的权限

#### ACL命令

* + - * 1. getAcl

获取指定节点的ACL信息

create /testDir/testAcl deer # 创建一个子节点

getAcl /testDir/testAcl # 获取该节点的acl权限信息

* + - * 1. setAcl

设置指定节点的ACL信息

setAcl /testDir/testAcl world:anyone:crwa # 设置该节点的acl权限

getAcl /testDir/testAcl # 获取该节点的acl权限信息，成功后，该节点就少了d权限

create /testDir/testAcl/xyz xyz-data # 创建子节点

delete /testDir/testAcl/xyz # 由于没有d权限，所以提示无法删除

* + - * 1. addauth

注册会话授权信息

Auth

addauth digest user1:123456 # 需要先添加一个用户

setAcl /testDir/testAcl auth:user1:123456:crwa # 然后才可以拿着这个用户去设置权限

getAcl /testDir/testAcl # 密码是以密文的形式存储的

create /testDir/testAcl/testa aaa

delete /testDir/testAcl/testa # 由于没有d权限，所以提示无法删除

退出客户端后：

ls /testDir/testAcl #没有权限无法访问

create /testDir/testAcl/testb bbb #没有权限无法访问

addauth digest user1:123456 # 重新新增权限后可以访问了

Digest

auth与digest的区别就是，前者使用明文密码进行登录，后者使用密文密码进行登录

create /testDir/testDigest data

addauth digest user1:123456

setAcl /testDir/testDigest digest:user1:HYGa7IZRm2PUBFiFFu8xY2pPP/s=:crwa # 使用digest来设置权限

注意：这里如果使用明文，会导致该znode不可访问

**通过明文获得密文**

shell>

java -Djava.ext.dirs=/soft/zookeeper-3.4.12/lib -cp /soft/zookeeper-3.4.12/zookeeper-3.4.12.jar org.apache.zookeeper.server.auth.DigestAuthenticationProvider deer:123456

deer:123456->deer:ACFm5rWnnKn9K9RN/Oc8qEYGYDs=

* + - * 1. acl命令行ip

create /testDir/testIp data

setAcl /testDir/testIp ip:192.168.30.10:cdrwa

getAcl /testDir/testIp

### 常用四字命令

ZooKeeper 支持某些特定的四字命令字母与其的交互。用来获取 ZooKeeper 服务的当前状态及相关信息。可通过 telnet 或 nc 向 ZooKeeper 提交相应的命令 ：

当然，前提是安装好了nc

echo stat|nc 127.0.0.1 2181 来查看哪个节点被选择作为follower或者leader 

使用echo ruok|nc 127.0.0.1 2181 测试是否启动了该Server，若回复imok表示已经启动。 

echo dump| nc 127.0.0.1 2181 ,列出未经处理的会话和临时节点。 

echo kill | nc 127.0.0.1 2181 ,关掉server 

echo conf | nc 127.0.0.1 2181 ,输出相关服务配置的详细信息。 

echo cons | nc 127.0.0.1 2181 ,列出所有连接到服务器的客户端的完全的连接 / 会话的详细信息 

echo envi |nc 127.0.0.1 2181 ,输出关于服务环境的详细信息（区别于 conf 命令）。 

echo reqs | nc 127.0.0.1 2181 ,列出未经处理的请求。 

echo wchs | nc 127.0.0.1 2181 ,列出服务器 watch 的详细信息。 

echo wchc | nc 127.0.0.1 2181 ,通过 session 列出服务器 watch 的详细信息，它的输出是一个与 watch 相关的会话的列表。 

echo wchp | nc 127.0.0.1 2181 ,通过路径列出服务器 watch 的详细信息。它输出一个与 session 相关的路径。

### ZooKeeper 日志可视化

前面以及讲了两个非常重要的配置一个是dataDir，存放的快照数据，一个是dataLogDir，存放的是事务日志文件

java -cp /soft/zookeeper-3.4.12/zookeeper-3.4.12.jar:/soft/zookeeper-3.4.12/lib/slf4j-api-1.7.25.jar org.apache.zookeeper.server.LogFormatter log.1

win7

java -cp D:\tool\zookeeper-3.4.14\zookeeper-3.4.14.jar;D:\tool\zookeeper-3.4.14\lib\slf4j-api-1.7.25.jar org.apache.zookeeper.server.LogFormatter log.1

java -cp /soft/zookeeper-3.4.12/zookeeper-3.4.12.jar:/soft/zookeeper-3.4.12/lib/slf4j-api-1.7.25.jar org.apache.zookeeper.server.SnapshotFormatter log.1

win7

java -cp D:\tool\zookeeper-3.4.14\zookeeper-3.4.14.jar;D:\tool\zookeeper-3.4.14\lib\slf4j-api-1.7.25.jar org.apache.zookeeper.server.SnapshotFormatter snapshot.5

## Java客户端框架（\*重要）

### Zookeeper原生客户端

### ZkClient

### Curator

## Zk入门

* 会话
* 节点属性
* 节点类型
* 版本
* 监听器机制
* 安全权限
* 单机部署
* Zk命令

## Zk开发基础

* 原生客户端
* Zkclient
* Curotor

# Zookeeper进阶

## Zk集群

* Zk集群部署安装
* 客户端连接zk集群及原理

## Zk的注意事项

# Zookeeper使用举例

## 配置中心

## 分布式锁

## 集群选举

## 命名服务