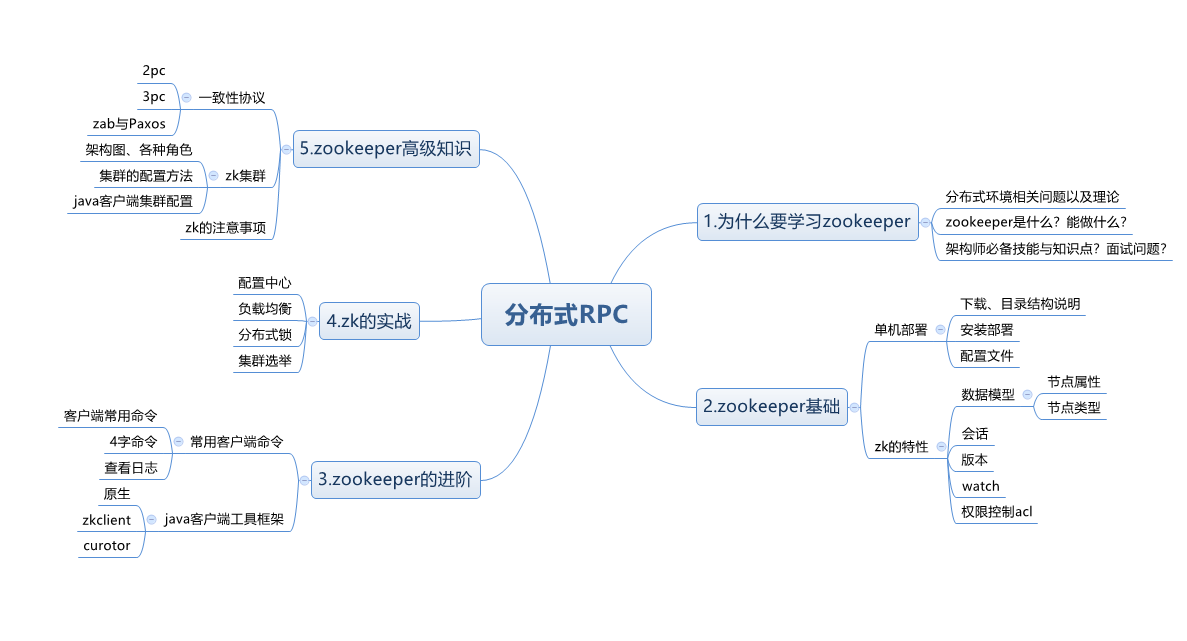
# Zookeeper综述

## 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

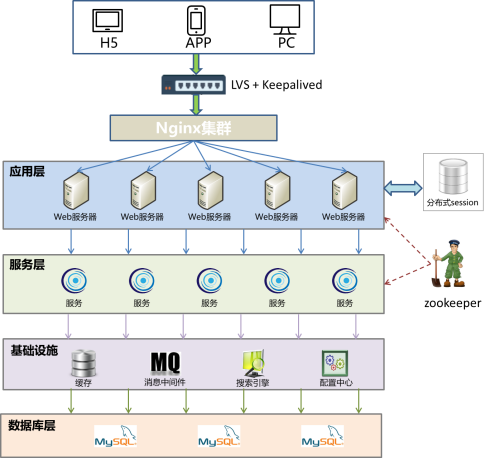
在分布式系统中，节点是可能反正任意位置的，而每个位置，每个节点都有自己的时间系统，因此在分布式系统中，很难定义两个事务纠结谁先谁后，原因就是因为缺乏一个全局的时钟序列进行控制，当然，现在这已经不是什么大问题了，已经有大把的时间服务器给系统调用

#### 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”

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。

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

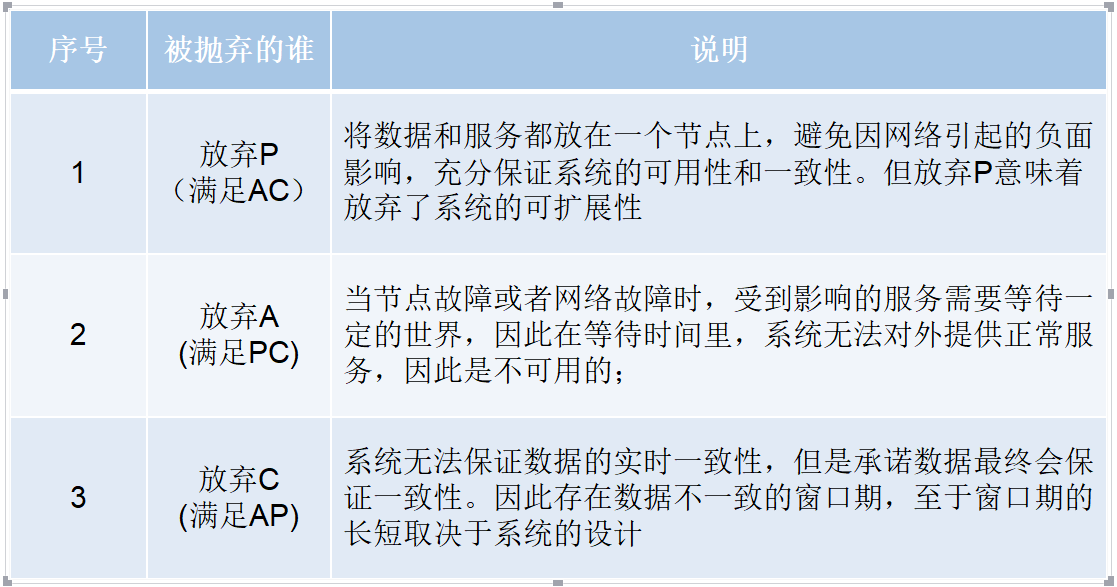
分布式系统在遇到任何网络分区故障的时候，仍然需要能够对外提供满足一致性和可用性的服务，除非是整个网络环境都发生了故障。

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；

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：对于每个读请求，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；

### 哪些常见需要用到ZK（why）

数据发布订阅

负载均衡

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？