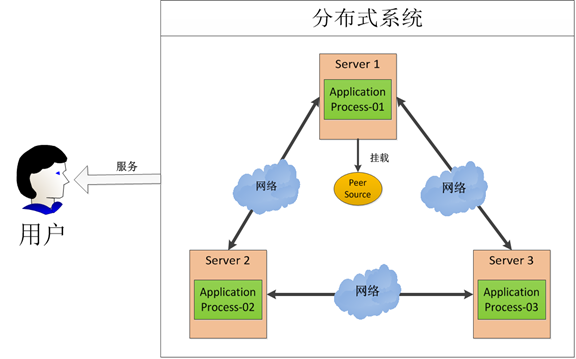
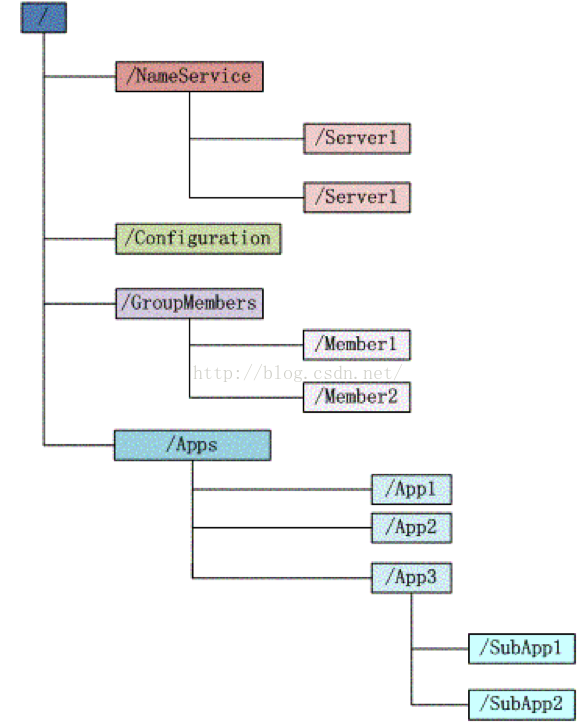
1. Zookeeper简介
2. 定义：

ZooKeeper是一种**为分布式应用所设计的高可用、高性能且一致的开源协调服务**，它提供了一项基本服务：**分布式锁服务**。由于ZooKeeper的开源特性，后来我们的开发者在分布式锁的基础上，摸索了出了其他的使用方法：配置维护、组服务、分布式消息队列、分布式通知/协调等。



1. 作用：
2. 要是用来解决分布式应用中经常遇到的一些数据管理问题，如：统一命名服务、状态同步服务、集群管理、分布式应用配置项的管理等。简化分布式应用协调及其管理的难度，提供高性能的分布式服务。
3. ZooKeeper的目标就是封装好复杂 易出错的关键服务，将简单易用的接口和性能高效、功能稳定的系统提供给用户。
4. ZooKeeper所提供的服务主要是通过：数据结构+原语+watcher机制，三个部分来实现的。
5. Zookeeper的数据模型 -- Znode



ZooKeeper的数据模型，在结构上和标准文件系统的非常相似，都是采用这种树形层次结构，ZooKeeper树中的每个节点被称为—Znode。

1. 数据模型特点

和文件系统的目录树一样，ZooKeeper树中的每个节点可以拥有子节点。但也有不同之处：

1. 引用方式

Zonde通过路径引用，如同Unix中的文件路径。路径必须是绝对的，因此他们**必须由斜杠字符来开头**。除此以外，他们必须是唯一的，也就是说每一个路径只有一个表示，因此这些路径不能改变。在ZooKeeper中，路径由Unicode字符串组成，并且有一些限制。**字符串"/zookeeper"用以保存管理信息**，比如关键配额信息。

1. Znode结构

ZooKeeper命名空间中的Znode，兼具文件和目录两种特点。**既像文件一样维护着数据、元信息、ACL、时间戳等数据结构，又像目录一样可以作为路径标识的一部分。**图中的每个节点称为一个Znode。 每个Znode由3部分组成:

① stat：此为状态信息, 描述该Znode的版本, 权限等信息

② data：与该Znode关联的数据

③ children：该Znode下的子节点

* ZooKeeper虽然可以关联一些数据，但并没有被设计为常规的数据库或者大数据存储，相反的是，它用来管理调度数据，比如分布式应用中的配置文件信息、状态信息、汇集位置等等。这些数据的共同特性就是它们都是很小的数据，通常以KB为大小单位。ZooKeeper的服务器和客户端都被设计为严格检查并限制每个Znode的数据大小至多1M，但常规使用中应该远小于此值。

1. 数据访问 -- 原子性操作和访问控制

ZooKeeper中的每个节点存储的数据要被**原子性的操作**。也就是说读操作将获取与节点相关的所有数据，写操作也将替换掉节点的所有数据。另外，每一个节点都拥有自己的ACL(**访问控制**列表)，这个列表规定了用户的权限，即限定了特定用户对目标节点可以执行的操作。

1. 节点类型 -- 创建即确定，不可改变。

ZooKeeper中的节点有两种，分别为临时节点和永久节点。节点的类型在**创建时即被确定，并且不能改变。**

① 临时节点：该节点的生命周期依赖于创建它们的会话。一旦会话(Session)结束，临时节点将被自动删除，当然可以也可以手动删除。虽然每个临时的Znode都会绑定到一个客户端会话，但他们对所有的客户端还是可见的。另外，ZooKeeper的临时节点**不允许拥有子节点**。

② 永久节点：该节点的生命周期不依赖于会话，并且只有在客户端显示执行删除操作的时候，他们才能被删除。

1. 顺序节点

当创建Znode的时候，用户可以请求在ZooKeeper的路径结尾添加一个递增的计数。这个计数**对于此节点的父节点来说是唯一的**，它的格式为"%10d"(10位数字，没有数值的数位用0补充，例如"0000000001")。当计数值大于232-1时，计数器将溢出。

1. 观察

客户端可以在节点上设置watch，我们称之为监视器。当节点状态发生改变时(Znode的增、删、改)将会触发watch所对应的操作。当watch被触发时，ZooKeeper将会向客户端发送且仅发送一条通知，因为watch只能被触发一次，这样可以减少网络流量。

1. Zookeeper中的时间

ZooKeeper有多种记录时间的形式，其中包含以下几个主要属性：

1. Zxid -- 对于每个操作是唯一的，有序的。

致使ZooKeeper节点状态改变的每一个操作都将使节点接收到一个Zxid格式的时间戳，并且这个时间戳全局有序。也就是说，每个对节点的改变都将产生一个唯一的Zxid。如果Zxid1的值小于Zxid2的值，那么Zxid1所对应的事件发生在Zxid2所对应的事件之前。

1. 实际上，ZooKeeper的每个节点维护者三个Zxid值，为别为：cZxid、mZxid、pZxid。

① cZxid： 是节点的创建时间所对应的Zxid格式时间戳。

② mZxid：是节点的修改时间所对应的Zxid格式时间戳。

实现中Zxid是一个64为的数字，它高32位是epoch用来标识leader关系是否改变，每次一个leader被选出来，它都会有一个 新的epoch。低32位是个递增计数。

1. 版本号 -- 对节点的每一个操作都将致使这个节点的版本号增加。

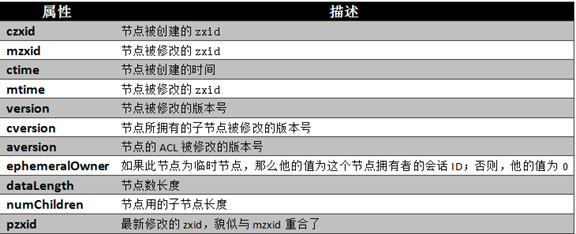
每个节点维护着三个版本号，他们分别为：

①version：节点数据版本号。

②cversion：子节点版本号。

③ aversion：节点所拥有的ACL版本号。

1. ZooKeeper的节点属性



三、原语 -- Zookeeper服务中操作 -- 有9个基本操作 --



* 注意事项：
  1. 更新ZooKeeper操作是**有限制的**。delete或setData必须明确要更新的Znode的版本号，我们可以调用exists找到。如果版本号不匹配，更新将会失败。
  2. 更新ZooKeeper操作是**非阻塞式的**。因此客户端如果失去了一个更新(由于另一个进程在同时更新这个Znode)，他可以在不阻塞其他进程执行的情况下，选择重新尝试或进行其他操作。
  3. 尽管ZooKeeper可以被看做是一个文件系统，但是处于便利，摒弃了一些文件系统地操作原语。因为文件非常的小并且使整体读写的，所以不需要打开、关闭或是寻地的操作。

四、Watch触发器

1. watch概述 -- 读操作、一次性、异步

ZooKeeper可以为**所有的读操作**设置watch，这些读操作包括：exists()、getChildren()及getData()。watch事件是**一次性的触发器**，当watch的对象状态发生改变时，将会触发此对象上watch所对应的事件。watch事件将被**异步地发送给客户端**，并且ZooKeeper为watch机制提供了有序的一致性保证。理论上，客户端接收watch事件的时间要快于其看到watch对象状态变化的时间。

1. watch类型
2. ZooKeeper所管理的watch可以分为两类：

① 数据watch(data watches)：getData和exists负责设置数据watch

② 孩子watch(child watches)：getChildren负责设置孩子watch

1. 可以通过操作返回的数据来设置不同的watch：

① getData和exists：返回关于节点的数据信息

② getChildren：返回孩子列表

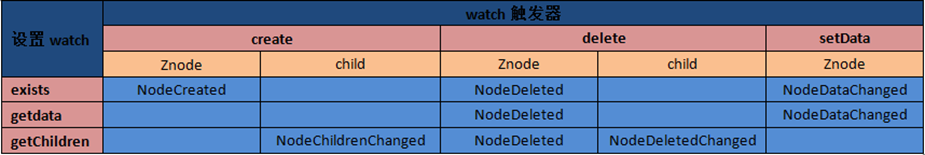
* 因此

① 一个成功的setData操作将触发Znode的数据watch

② 一个成功的create操作将触发Znode的数据watch以及孩子watch

③ 一个成功的delete操作将触发Znode的数据watch以及孩子watch

1. watch注册及触发



① exists操作上的watch，在被监视的Znode创建、删除或数据更新时被触发。

② getData操作上的watch，在被监视的Znode删除或数据更新时被触发。在被创建时不能被触发，因为只有Znode一定存在，getData操作才会成功。

③ getChildren操作上的watch，在被监视的Znode的子节点创建或删除，或是这个Znode自身被删除时被触发。可以通过查看watch事件类型来区分是Znode，还是他的子节点被删除：NodeDelete表示Znode被删除，NodeDeletedChanged表示子节点被删除。数据更新时不触发。

* Watch由客户端所连接的ZooKeeper服务器在本地维护，因此watch可以非常容易地设置、管理和分派。当客户端连接到一个新的服务器 时，任何的会话事件都将可能触发watch。另外，当从服务器断开连接的时候，watch将不会被接收。但是，当一个客户端重新建立连接的时候，任何先前注册过的watch都会被重新注册。

1. 注意事项
2. Zookeeper的watch实际上要处理两类事件：

① 连接状态事件(type=None, path=null)

这类事件不需要注册，也不需要我们连续触发，我们只要处理就行了。

② 节点事件

节点的建立，删除，数据的修改。它是one time trigger，我们需要不停的注册触发，还可能发生事件丢失的情况。

上面2类事件都在Watch中处理，也就是重载的process(Event event)

1. 节点事件的触发，通过函数exists，getData或getChildren来处理这类函数，有双重作用：
2. 注册触发事件
3. 函数本身的功能

函数的本身的功能又可以用异步的回调函数来实现,重载processResult()过程中处理函数本身的的功能。

zookeeper的安装配置：http://www.cnblogs.com/wuxl360/p/5817489.html