# ****1 ZooKeeper概述****

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，是Google的Chubby一个开源实现，是Hadoop和Hbase的重要组件，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。

zookeeper是基于内存同步数据的，所以集群内的节点其内存中的数据结构是完全相同的，因此效率非常高。

ZooKeeper具备的几个特点决定了它能够用在大型的、分布式的系统当中:

顺序一致性

从同一个客户端发起的事务请求，最终将会严格按照其发起顺序被应用到zookeeper中

原子性

所有事物请求的处理结果在整个集群中所有机器上的应用情况是一致的，即，要么整个集群中所有机器都成功应用了某一事务，要么都没有应用，一定不会出现集群中部分机器应用了改事务，另外一部分没有应用的情况。

单一视图

无论客户端连接的是哪个zookeeper服务器，其看到的服务端数据模型都是一致的。

可靠性

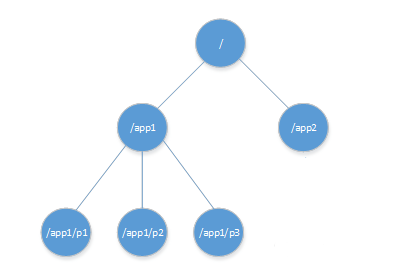
一旦服务端成功的应用了一个事务，并完成对客户端的响应，那么该事务所引起的服务端状态变更将会一直保留下来，除非有另一个事务又对其进行了改变。

实时性

zookeeper并不是一种强一致性，只能保证顺序一致性和最终一致性，只能称为达到了伪实时性。

# ****2 zookeeper 数据模型****

ZooKeeper拥有一个层次的命名空间，这个和标准的文件系统非常相似，如图:

[](http://s4.51cto.com/wyfs02/M00/88/77/wKiom1f4o3nCtsvDAAA8xYyMV0E583.png)

ZooKeeper采用树形层次结构，树中的每个节点被称为—Znode，Znode也可以拥有子节点。

**引用方式:**

Zonde通过路径引用，如同Unix中的文件路径。必须是绝对路径，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。

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。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。

# ****3节点类型****

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。

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。

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。

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。

监视器

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。

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**

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，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。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。

**(2) 版本号**

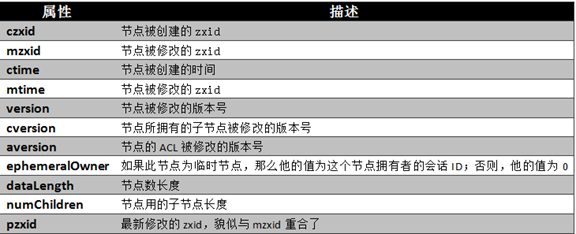
对节点的每一个操作都将致使这个节点的版本号增加。每个节点维护着三个版本号，他们分别为：

1、version：节点数据版本号

2、cversion：子节点版本号

3、aversion：节点所拥有的ACL版本号

ZooKeeper节点属性：

[](http://s5.51cto.com/wyfs02/M02/88/73/wKioL1f4rZzw2GjCAADUKpI2I-U152.png)

# ****4 ZooKeeper 安装配置****

ZooKeeper的工作模式有三种:单机模式、集群模式、伪集群模式。

单机模式：Zookeeper只运行在一台服务器上，适合测试环境；

伪集群模式：就是在一台物理机上运行多个Zookeeper 实例；

集群模式：Zookeeper运行于一个至少有三个节点以上集群中，适合生产环境;

Zookeeper通过复制来实现高可用性，只要集群中半数以上的节点处于可用状态，它就能够保证服务继续。为什么一定要超过半数呢？这跟Zookeeper的复制策略有关：zookeeper确保对znode 树的每一个修改都会被复制到集群中超过半数的节点上。

服务器角色有两种:

1、leader -主节点

2、follower -从节点

在一个集群中通常只有一个leader和多个follower，当leader下线时，follower能根据选举策略，选举一个新的leader保证集群的正常工作。

下面我们来构建一个单机模式的ZooKeeper节点:

可以去Apache的镜像站点下载zookeeper: http://www.apache.org/dyn/closer.cgi/zookeeper/

安装ZooKeeper之前首先需要安装JDK，在oracle官网可以下载到。（建议下载rpm包安装）

1、解压zookeeper

|  |  |
| --- | --- |
| 1  2 | tar xf zookeeper-3.4.9.tar.gz  cd zookeeper-3.4.9 |

2、创建配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5 | # conf/zoo.cfg    tickTime=2000  dataDir=/tmp/zookeeper  clientPort=2181 |

3、启动服务

|  |  |
| --- | --- |
| 1 | bin/zkServer.sh start |

**ZooKeeper伪集群模式**

ZooKeeper提供了伪集群模式，也就是可以在同一台机器上启动多个zookeeper服务，当手头机器不足时，又需要做实验，提供了很大的便利。

注: 在同一台机器上部署3个zookeeper server时，需要为每个server创建独立的配置文件，并且保证每个配置文件中的端口不能冲突，除了clientPort不同之外，另外，还要在dataDir所对应的目录中创建myid文件来指定对应的zookeeper server 实例。

需要注意的几个配置项:

clientPort端口：如果在1台机器上部署多个server，那么每台机器都要不同的 clientPort，比如 server1是2181,server2是2182，server3是2183

dataDir和dataLogDir：dataDir和dataLogDir也需要区分下，将数据文件和日志文件分开存放，同时每个server的这两变量所对应的路径都是不同的

server.X和myid： server.X 这个数字就是对应，data/myid中的数字。在3个server的myid文件中分别写入了0，1，2，那么每个server中的zoo.cfg都配 server.0 server.2,server.3就行了。因为在同一台机器上，后面连着的2个端口，3个server都不要一样，否则端口冲突

启动命令

|  |  |
| --- | --- |
| 1  2  3  4 | # 这时使用单机模式的启动命令是不行的，需要在指令后面跟上不同实例的配置文件名称。如下:  bin/zkServer.sh start zoo1.cfg  bin/zkServer.sh start zoo2.cfg  bin/zkServer.sh start zoo3.cfg |

# ****5 集群实战****

构建一个3节点的 ZooKeeper 集群

实验环境:

|  |  |
| --- | --- |
| 服务器IP | myid |
| 192.168.1.21 | 11 |
| 192.168.1.22 | 12 |
| 192.168.1.23 | 13 |

注:

(1) zk服务器集群规模不得小于3个节点

(2) 要求各服务器之间系统时间要保持一致。

为了实现高可用的ZooKeeper服务，应该把zk部署在一个集群中，每台机器只跑一个zk实例。配置方式和前面差不多，只是每台机器上的配置文件 conf/zoo.cfg 完全相同。

创建myid

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | # 为每台机器创建myid文件    # 192.168.1.21  echo 11 >/u01/zookeeper/zookeeper-3.4.9/data/myid    # 192.168.1.22  echo 12 >/u01/zookeeper/zookeeper-3.4.9/data/myid    # 192.168.1.23  echo 13 >/u01/zookeeper/zookeeper-3.4.9/data/myid |

创建配置文件

可以复制 zoo\_sample.cfg 到 zoo.cfg 以下为配置文件的参数设置:

改好配置文件后，同步到集群内的所有节点。

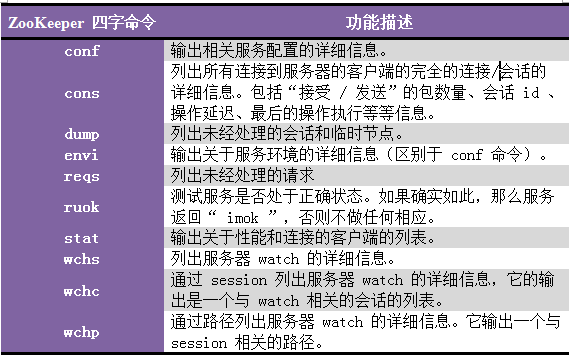
常用配置参数解析:

启动服务 & 查看节点的角色分配情况

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  0 | # 192.168.1.21  bin/zkServer.sh start  bin/zkServer.sh status  # ZooKeeper JMX enabled by default  # Using config: /u01/zookeeper/zookeeper-3.4.9/bin/../conf/zoo.cfg  # Mode: leader    # 192.168.1.22  bin/zkServer.sh start  bin/zkServer.sh status  # ZooKeeper JMX enabled by default  # Using config: /u01/zookeeper/zookeeper-3.4.9/bin/../conf/zoo.cfg  # Mode: follower    # 192.168.1.23  bin/zkServer.sh start  bin/zkServer.sh status  # ZooKeeper JMX enabled by default  # Using config: /u01/zookeeper/zookeeper-3.4.9/bin/../conf/zoo.cfg  # Mode: follower |

由此可见，集群已经正常运行;

# ****6 ZooKeeper的四字命令****

[](http://s1.51cto.com/wyfs02/M00/88/7C/wKioL1f57uLyirQMAAA_16Dojns720.png)

客户端可以通过nc或telnet连接ZooKeeper Server提交指令

简单用例:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  2 | # 检查服务器状态是否正常    echo ruok | nc localhost 2181    imok    # 输出服务器配置信息    echo conf | nc localhost 2181    clientPort=2181  dataDir=/u01/zookeeper/zookeeper-3.4.9/data/version-2  dataLogDir=/u01/zookeeper/zookeeper-3.4.9/data/version-2  tickTime=2000  maxClientCnxns=60  minSessionTimeout=4000  maxSessionTimeout=40000  serverId=11  initLimit=10  syncLimit=5  electionAlg=3  electionPort=3888  quorumPort=2888  peerType=0    # 输出服务器状态信息    echo stat | nc localhost 2181    Zookeeper version: 3.4.9-1757313, built on 08/23/2016 06:50 GMT  Clients:   /127.0.0.1:60822[0](queued=0,recved=1,sent=0)    Latency min/avg/max: 0/1/1591  Received: 200338  Sent: 200389  Connections: 1  Outstanding: 0  Zxid: 0x200498db5  Mode: follower  Node count: 166 |

# ****7 ZooKeeper Client 简单操作****

9个基本操作指令:

[](http://s1.51cto.com/wyfs02/M00/88/7D/wKioL1f5-vaRFV-5AAC_vdJ_BPM006.png)

简单用例:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  6 | # 连接 ZooKeeper 服务器    bin/zkCli.sh -server localhost:2181    # 查看根下有哪些节点    [zk: localhost:2181(CONNECTED) 1] ls /    [controller\_epoch, controller, brokers, zookeeper, admin, isr\_change\_notification, consumers, config]      # 查看 brokers 下有哪些子节点    [zk: localhost:2181(CONNECTED) 4] ls /brokers    [ids, topics, seqid]      # 在根下创建一个 "tuchao" 节点，并设置数据为 "hello zookeeper"    [zk: localhost:2181(CONNECTED) 5] create /tuchao "hello zookeeper"    # 查看是否创建成功    [zk: localhost:2181(CONNECTED) 6] ls /    [controller\_epoch, controller, brokers, zookeeper, tuchao, admin, isr\_change\_notification, consumers, config]    # 查看 /tuchao 中的数据    [zk: localhost:2181(CONNECTED) 7] get /tuchao    hello zookeeper  cZxid = 0x20049ab24  ctime = Sun Oct 09 15:45:02 CST 2016  mZxid = 0x20049ab24  mtime = Sun Oct 09 15:45:02 CST 2016  pZxid = 0x20049ab24  cversion = 0  dataVersion = 0  aclVersion = 0  ephemeralOwner = 0x0  dataLength = 15  numChildren = 0    # 可以看到刚刚设置的数据。    # 修改 /tuchao 中的数据为 "Happy birthday"    [zk: localhost:2181(CONNECTED) 8] set /tuchao "Happy birthday"    cZxid = 0x20049ab24  ctime = Sun Oct 09 15:45:02 CST 2016  mZxid = 0x20049b2cc  mtime = Sun Oct 09 15:51:17 CST 2016  pZxid = 0x20049ab24  cversion = 0  dataVersion = 1   # 可以发现，当数据变更时，数据版本号增加了。  aclVersion = 0  ephemeralOwner = 0x0  dataLength = 14  numChildren = 0    # 再查看一次 /tuchao 中的数据    [zk: localhost:2181(CONNECTED) 9] get /tuchao    Happy birthday  cZxid = 0x20049ab24  ctime = Sun Oct 09 15:45:02 CST 2016  mZxid = 0x20049b2cc  mtime = Sun Oct 09 15:51:17 CST 2016  pZxid = 0x20049ab24  cversion = 0  dataVersion = 1  aclVersion = 0  ephemeralOwner = 0x0  dataLength = 14  numChildren = 0    # 给 /tuchao 创建一个子节点 /tuchao/abc1    [zk: localhost:2181(CONNECTED) 10] create /tuchao/abc1 "abc1 node"  Created /tuchao/abc1    [zk: localhost:2181(CONNECTED) 11] ls /tuchao  [abc1]    # 删除节点 /tuchao/abc1    [zk: localhost:2181(CONNECTED) 0] delete /tuchao/abc1  [zk: localhost:2181(CONNECTED) 1] ls /tuchao  []    # 查看命令帮助，输入 h    [zk: localhost:2181(CONNECTED) 2] h  ZooKeeper -server host:port cmd args      stat path [watch]      set path data [version]      ls path [watch]      delquota [-n|-b] path      ls2 path [watch]      setAcl path acl      setquota -n|-b val path      history      redo cmdno      printwatches on|off      delete path [version]      sync path      listquota path      rmr path      get path [watch]      create [-s] [-e] path data acl      addauth scheme auth      quit      getAcl path      close      connect host:port |