ZooKeeper集群安装配置和理论知识

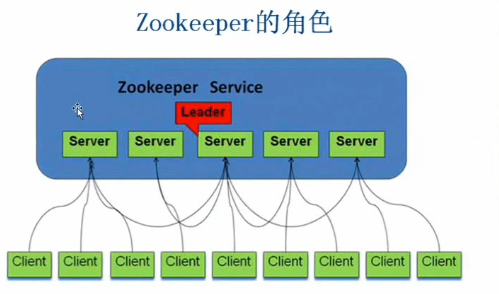
1. 简介：zookeeper是Google的Chubby的一个开源实现，是hadoop的分布式协调服务
2. Zookeeper(简称zk)包含一个简单的原语集，分布式应用程序可以给予它实现同步服务，配置维护和命名服务等
3. Zk的设计目标
   1. 简单化：通过共享体系的，命名空间进行协调，与文件系统相似，有一些数据寄存器组成，被称为Znode。Zk的数据是放在内存中的，zk可以达到高吞吐量、低延迟。

Zk能用在大型、分布式的系统。

严格的序列访问控制意味者复杂的控制源可以用在客户端上。

* 1. 健壮性：zk互相知道其他服务器存在。维护一个处于内存中的状态镜像，以及一个位于存储器中的交换日志和快照。只要大部分服务器可用，zk服务就可用。
  2. 有序性：zk为每次更新赋予一个版本号，全局有序。
  3. 速度优势:去读主要负载时尤其快，当读操作比写操作多时，性能会更好。
  4. Zk还有原子性、单系统镜像、可靠性和实效性特点。

1. Zk可以用来保证数据在zk集群之间的数据的事务性的一致 (一般数据在2M以下)



1. 如何搭建zk集群
   1. 前提：
      1. Zk服务器集群规模不小于3个节点，并且各个服务器之间时间一致
   2. 安装：

将zookeeper的压缩包上传到linux，通过winscp软件，暂定为/usr/local/下，解压zookeeper压缩包

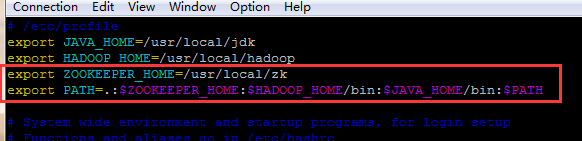
#tar –xzvf zookeeper-3.4.5.tar.gz

为zookeeper-3.4.5改为简单的名字，便于使用

#mv zookeeper-3.4.5 zk

添加环境变量

#vim /etc/profile



此处的PATH = .:$ZOOKEEPER\_HOME/bin:$HADOOP\_HOME/bin:$JAVVA\_HOME/bin:$PATH

使环境变量生效

#source /etc/profile

进入zookeeper的配置文件目录

#cd /usr/local/zk/conf

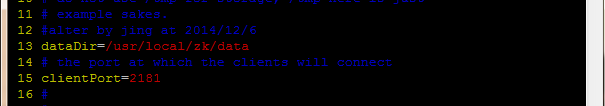
复制一份zoo\_sample.cfg 并更名为zoo.cfg

#cp zoo\_sample.cfg zoo.cfg

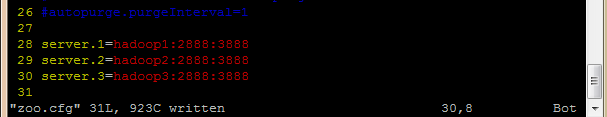
修改zookeeper的配置文件zoo.cfg

#vim zoo.cfg

修改第十三行的datadir路径为自己想要的路径



配置三个zk服务器



其中第一个端口用来集群成员的信息交换，第二个端口是在leader挂掉时专门用来进行选举leader所用。

创建文件夹data

#mkdir /usr/local/zk/data

在data目录下，创建文件myid,并写上对应的zoo.cfg中的标号

例如：server.1=hadoop1:2888:3888

则在hadoop1主机上的myid中添加1

在三台hadoop机器上同步部署zk的文件夹及环境变量

#scp -r /etc/profile hadoop2:/etc/

#scp -r /etc/profile hadoop3:/etc/

使得环境变量生效

#source /etc/profile

#scp -r /usr/local/zk/ hadoop2:/usr/local/

#scp -r /usr/local/zk/ hadoop3:/usr/local/

修改各个主机中的myid文件中的值，以便与zoo.cfg中的配置对应，使得zk能找到对应的机器

#echo 2 >/usr/local/zk/data/myid (机器hadoop2上)

#echo 3 >/usr/local/zk/data/myid (机器hadoop3上)

* 1. 启动zk集群服务

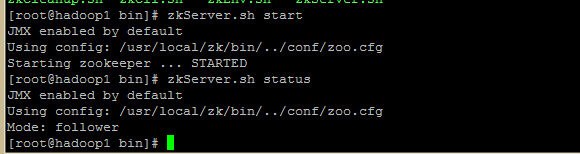
在每台机器上分别执行/usr/local/zk/bin/zkServer.sh文件

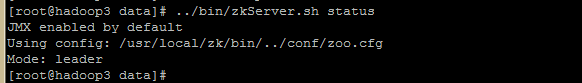
#/usr/local/zk/bin/zkServer.sh start

在这里三台zookeeper服务器会自动选举loader然后其他的都是follower

检验每个节点上的zookeeper的角色状态

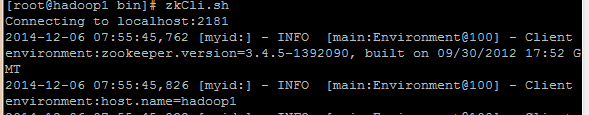
#/usr/local/zk/bin/zkServer.sh status

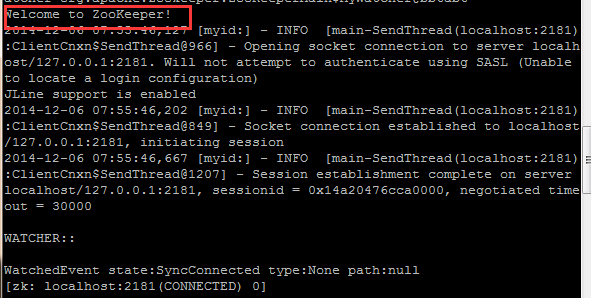




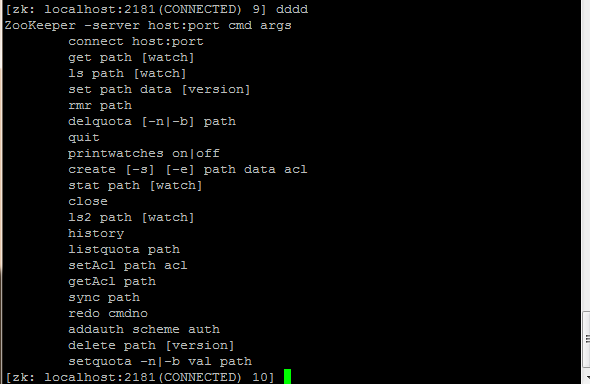
1. 使用zookeeper

在命令行中执行/usr/local/zk/bin/zkCli.sh



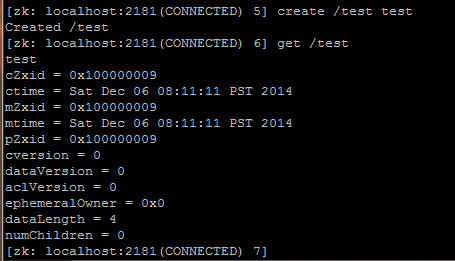


1. 随便输入什么，回车后可以查看提示信息



1. 试一试显示的命令

create /test test 创建一个路径/test，设置数据位test



登录另一台zk服务器的客户端，执行get /test命令，同样会看到如上的结果，则证明机器间的数据同步成功



1. 配置zookeeper

Zk是通过配置文件zoo.cfg控制，各个机器上的配置文件几乎是相同的，在集群部署时非常方便。

* 1. 最低配置：
     1. clientPort：监听客户端连接的端口
     2. dataDir：存储内存中数据库快照的位置
     3. tickTime：基本事件单元，毫秒单位，控制心跳和超时，默认为tickTime的两倍
  2. 高级配置：
     1. dataLogDir：事务日志写入” dataLogDir”指定的目录，而不是dataDir指定的目录。
     2. maxClientCnxns：限制连接到zk的客户端数量，限制并发连接的数量，通过IP区分客户端。设置为0或不设置会取消并发连接的控制。
     3. minSessionTimeout 和maxSessionTimeout

最小会话超时时间和最大会话超时时间，默认最小为tickTime的两倍。最大为20倍。

* 1. 集群配置：
     1. initLimit

允许follower连接并同步到leader的初始化连接时间，以tickTime的倍数表示，当时间超过tickTime的指定倍数时会失败。

* + 1. syncLimit：

leader和follower之间发送消息时请求和应答的时间长度。如果follower在设置的时间内不能和leader通信，那此follower将被丢弃。

1. Zookeeper特性

Zookeeper中指向节点的路径必须使用规范的绝对路径表示，并以斜线”/”分隔，zookeeper中不允许使用相对路径。

* 1. Znode：zk目录树中每个节点对应一个znode，每个znode维护者属性结构，包含版本号、时间戳等状态，跟linux的iNode节点作用类似。

Znode的主要特征：

* + 1. Watches：设置watch(监视器)，节点发生改变时，会触发watch对应操作，会向客户端发送且只一个通知，因为watch只能被触发一次。
    2. 数据访问：zk中每个节点存储的数据需要被原子性操作，每个节点都有ACL，限定了特定用户对目标节点可以执行的操作。
    3. 临时节点：节点分为临时节点和永久节点，节点类型在创建时确定，不能被改变。Zk临时节点的生命周期依赖创建他们的会话，会话结束临时节点结束。

临时节点不能有子节点。永久节点不依赖会话，只能在客户端执行删除操作删除。

* + 1. 顺序节点：创建Znode时，用户可在请求zk路径结尾添加递增计数。
  1. Zookeeper中的时间
     1. Zxid:每一个操作都会使节点收到zxid格式的时间戳，全局有序。每个节点维护三个zxid：cZxid、mZxid、pZxid。
     2. 版本号：对节点每个操作会使该节点版本号增加，三个版本号：dataVersion(节点数据版本号)、cversion(子节点版本号)、aclVersion(节点所拥有的ACL版本号)
  2. Zookeeper watches：zk可以为所有的读操作设置watch，包括(exists()、getChildren()、getData()).watch是一次性触发器。Watch事件将被异步发送到客户端，并且zk为watch提供了有序的一致性保证。

Zookeeper的watch分为两类：数据watch和子watch。exists()和getData()负责设置数据watch，getChildren()负责设置子watch。Create()和delete()触发znode的数据watch和子watch

Watch由客户端所连接的zookeeper服务器在本地维护，非常容易设置、管理和分派。

当客户端连接新的服务器时，任何会话事件都可能触发watch，当从服务器断开连接时，watch不会被接收，但当客户端重新连接时，先前注册的watch会被重新注册。

* 1. Zookeeper ACL

Ids.*OPEN\_ACL\_UNSAFE 🡪对所有的ACL都完全开放，*

Ids.READ\_ACL\_UNSAFE 🡪对任何应用程序都只有读权限

Ids.CREATOR\_ALL\_ACL 🡪节点创建者的所有权限，创建者必须通过服务器认证。

* 1. zookeeper的一致性：顺序一致性(与被发送顺序一直)、原子性(要么成功要么失败)、单系统镜像(客户端连接到集群的任一服务器看到相同的zookeeper视图)、可靠性(1.客户端成功返回代码🡪成功，否则不知道操作是否生效；2.故障恢复时，任何客户端能看到的执行成功的更新操作将不会回滚)和实时性(特定时间内，客户端看到的系统是实时的，任何系统的改变将被客户端看到，或者被客户端侦测到)

1. zookeeper进行leader选举

核心思想：

1.首先创建EPHEMERAL目录节点，如”/election”

2.每个zookeeper服务器在此目录下创建一个SEQUENCE|EPHEMERAL类型节点”如/election/n\_”，

3.zookeeper将自动为每个zookeeper服务器分配一个比前面所分配的序号要大的序号，拥有最小编号的zookeeper服务器将成为leader。

为了能在leader发生意外时，整个系统能选出leader，需要所有的follower都监视leader所对应节点，当leader故障时，leader对应的临时节点将会被删除，会触发所有监视的follower的watch，从而进行选举leader操作。

缺点：这样的解决方案会导致”从众效应。”

实现：每个follower为follower集群中对应着比自己节点序号小的节点中x序号最大的节点设置一个watch，只有当follower所设置的watch被触发时，他才进行leader操作，一般将其设置为集群的下一个leader。这样很快，因为每一leader选举几乎只涉及单个leaderfollower的操作

1. zookeeper锁服务
   1. zookeeper中完全分布的锁是全局存在的。
   2. zookeeper的锁机制(实现加锁)
      1. zk调用create()，创建路径格式为”\_locknode/lock\_”的节点，此节点类型为sequence(连续)和ephemeral(临时)，创建节点为临时节点，所有节点连续编号🡪“lock-i”格式
      2. 在创建锁节点上调用getChildren()方法，以获取锁目录下的最小编号节点，并且不设置watch。
      3. 步骤2获取的节点是步骤1中客户端创建的节点，此客户端会获得该种类型的锁，然后退出操作。
      4. 客户端在锁目录上调用exists()方法,并设置watch来监视锁目录下序号相对自己次小的连续临时节点的状态。
      5. 如果监视节点状态发生变化，则跳转到步骤2，继续后续操作直到退出锁竞争。
      6. Zookeeper解锁简单，只需在步骤1中创建的临时节点删除即可。

PS：1.一个客户端解锁后，将只可能有一个客户端获得锁，因此每个临时的连续节点对应一个客户端，并且节点间没有重叠；2.在zookeeper锁机制中没有轮询和超时。

1. BooKeeper

副本功能。提供可靠的日志记录。BooKeeper为每份日志提供了分布式存储，并且采用了大多数概念，就是说只要集群中大多数机器可用，那么该日志一直有效。

BooKeeper包含四个角色：账本(服务器)，账户(Ledger,账本中存储的一系列记录)、客户端(BooKeeper Client,允许APP在系统上进行操作，包括创建账户，写账户)、元数据存储服务(metadata storage service,存储关于账户和版本的信息)