zookeeper万字雄文总结（深度）

原创 教你学懂大数据 教你学懂大数据 今天

收录于话题
#大数据

2个

教你学懂大数据
用最简单的案例，教你学懂最难的技术，关注我，学懂大数据不再难。
3篇原创内容

公众号

一、Zookeeper介绍

1.1 什么是zookeeper

Zookeeper是一个分布式的、高性能的、开源的分布式系统的协调（Coordination）服务，是Google的Chubby一个开源的实现，是Hadoop和Hbase的一个重要的组件。它是一个为分布式应用提供一致性服务的软件。

1.2 zookeeper应用场景

zookeeper是一个经典的分布式数据一致性解决方案，致力于为分布式应用提供一个高性能，高可用，且具有严格属性访问控制能力的分布式协调存储服务。

- 维护配置信息
- 分布式锁服务
- 集群管理
- 生成分布式唯一ID

1. 维护配置信息

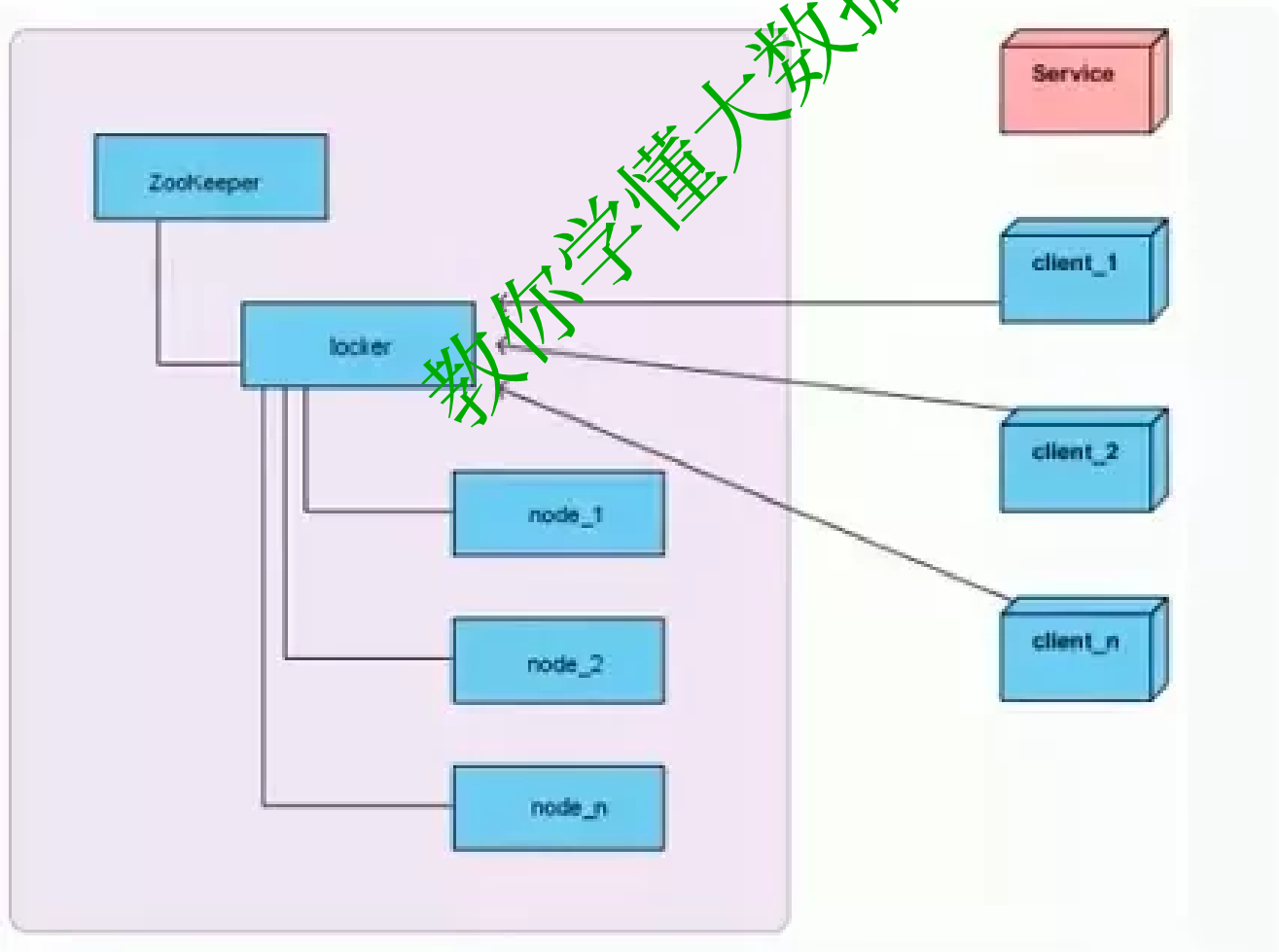
java编程经常会遇到配置项，比如数据库的url，schema，user和password等。通常这些配置项我们会放置在配置文件中，在将配置文件放置在服务器上当需要更改配置的时，需要去服务器上修改对应的配置信息文件。但是随着分布式系统的兴起，由于许多服务都

需要使用到该配置文件，因此有必须保证该配置服务的高可用性和各台服务器上配置数据的一致性。通常会将配置文件部署在一个集群上，然而一个集群动辄上千台服务器，此时如果在一台一台服务器逐个的修改配置文件将是非常繁琐的一个操作。因此就需要一种服务，能够高效快速且可靠的完成配置项的等待操作，并能够保证各个配置项在每一台服务器上的数据一致性。

zookeeper就可以提供这样一种服务，其使用Zab这种一致性协议来保证一致性。现在有很多开源项目使用zookeeper来维护配置，比如hhase中，客户端就是连接一个zookeeper，获得必要hbase集群的配置信息然后才可以进一步操作。还有开源的消息队列kafka中，也是用zookeeper来维护broker的信息。

2. 分布式锁服务

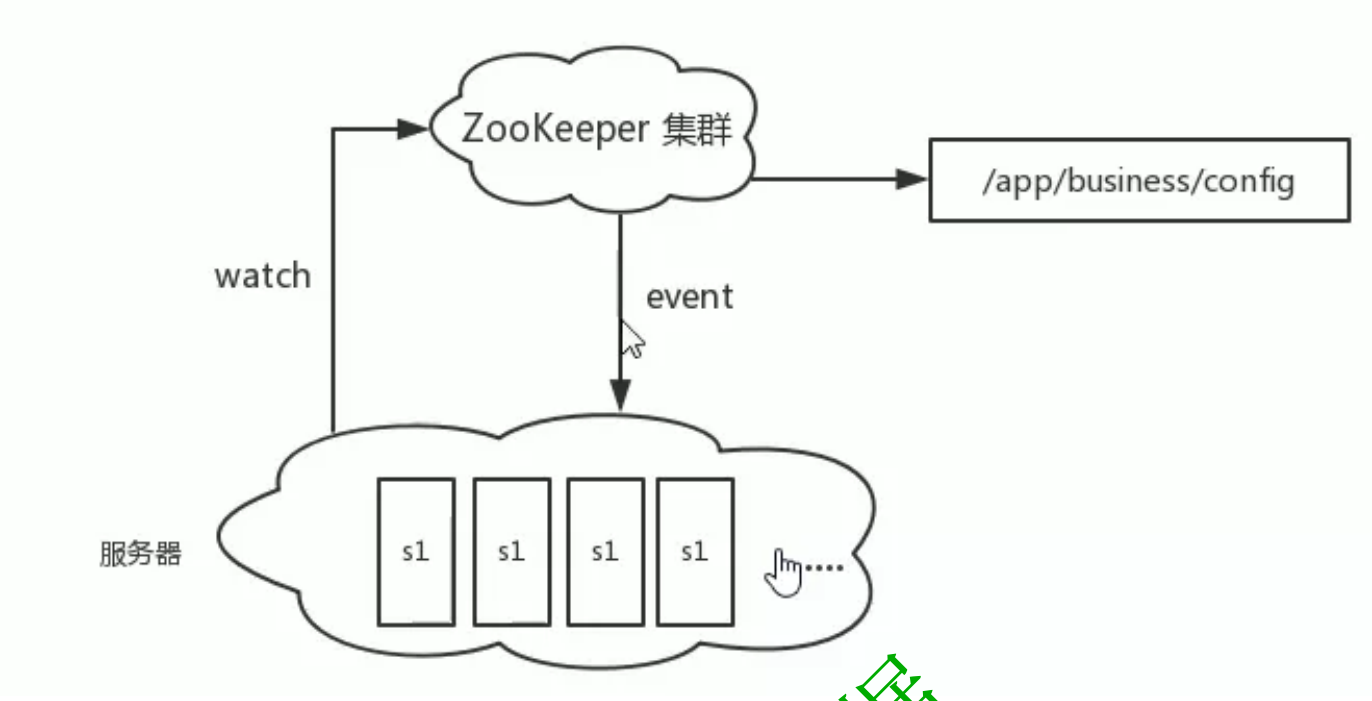
一个集群是一个分布式系统，由多台服务器组成。为了提高并发度和可靠性，多台服务器运行着同一种服务。当多个服务在运行时就需要协调各服务的进度，有时候需要保证当某个服务在进行某个操作时，其他的服务都不能进行该操作，即对该操作进行加锁，如果当前机器挂掉后，并释放fail over到其他的机器继续执行该服务。



3. 集群管理

一个集群优势会因为各种软硬件故障或者网络故障，出现某种服务器挂掉而被移除集群，而某些服务器加入到集群中的情况，zookeeper会将这些服务器加入/移出的情况下通知

给集群汇总的其他正常工作的服务器，以及时调用存储和计算等任务的分配和执行等。此外zookeeper还会对故障的服务器做出诊断并尝试修复。



4. 生成分布式唯一ID

在过去的单库单表型系统中，通常可以使用数据库字段自带的auto_increment属性来自动为每条记录生成一个唯一的ID。但是分库分表后，就无法再依靠数据库的auto_increatment属性来唯一标识一条记录了，此时我们就可以用zookeeper在分布式环境下生成全局唯一ID。做法如下：每个生成一个新ID时，创建一个持久顺序节点，创建操作返回的节点序号，及微信ID，然后把比自己节点小的删除即可。

1.3 zookeeper的设计目标

zookeeper致力于为分布式应用提供一个高性能，高可用，具有严格顺序访问控制能力的分布式协调服务。

1. 高性能

zookeeper将全量数据存储在内存中，并直接发起与客户端的所有非事务请求，尤其适合用于以读为主的应用场景。

2. 高可用

zookeeper一般以集群的方式对外提供服务，一般3-5台机器就可以组成一个可用的zookeeper集群，每一台机器都会在内存中维护当前的服务器状态，并且每台机器之间都相互保持着通信。只要集群中超过一旦的机器都在工作，那么这个集群就能够正常对外服务；

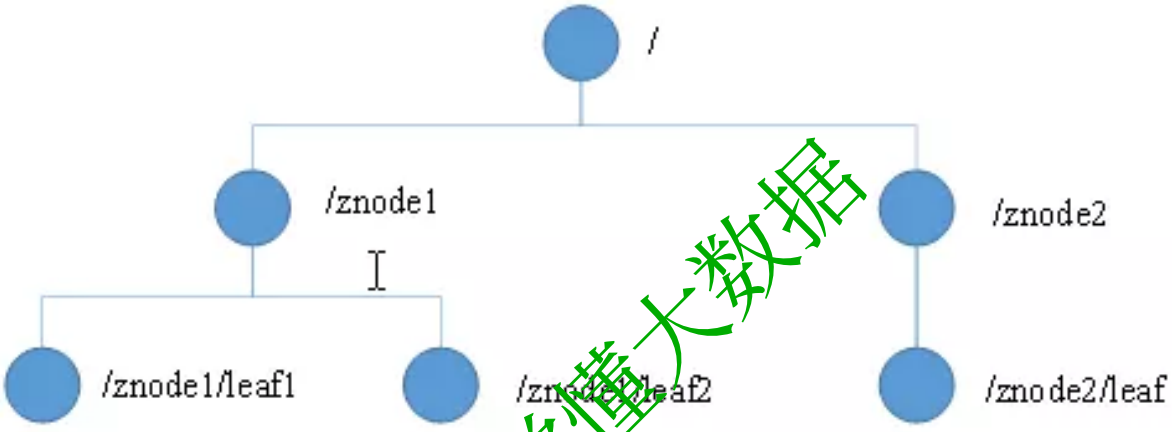
3. 严格访问数据

对于客户端的每一个更新请求，Zookeeper都会分配一个全局唯一的递增编号，这个编号反映了所有事物操作的先后顺序。

二、Zookeeper的数据模型

2.1. zookeeper数据结构

Zookeeper数据模型的结构与Unix文件系统很类似，整体上可以看作是一颗树，每一个节点称做一个ZNode。每一个Znode默认能够存储1MB的数据，每个ZNode都可以通过其路径唯一标识。



如何来描述一个ZNode呢？一个znode大体上分为3部分：

- 节点的数据：即znode data(节点path，节点data的关系)就像是java map中(key，value)的关系
- 节点的子节点children
- 节点的状态stat：用来描述当前节点的创建，修改记录，包括cZxid、ctime等。

2.2 zookeeper节点类型

zookeeper中的节点有两种类型，一种是临时节点和永久节点。节点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他们的会话。一旦会话（Session）结束，临时节点将会被自动删除，当然可以手动的进行删除。虽然每个临时的ZNode都会绑定到一个客户端回话，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。

- 持久化节点：该节点的生命周期不依赖于花花，并且只有在客户点显示执行删除操作的时候，他们才能被删除。

2.3 zookeeper 单机安装

当前测试系统环境centos7.3

1. 在centos中使用root用户创建zookeeper用户，用户名：zookeeper 密码：zookeeper

```
useradd zookeeper
passwd zookeeper
```

2. zookeeper 底层依赖jdk，zookeeper用户登录后，根目录下先进行jdk的安装，jdk使用jdk-11.0.5

```
//解压jdk
tar -xzvf jdk-11.0.5.tar.gz
```

3. 配置jdk环境量

```
//打开/etc/profile文件
# vim /etc/profile
export JAVA_HOME=/root/apps/jdk-11.0.5
export PATH=$PATH:$JAVA_HOME/bin
//执行profile
#source /etc/profile
```

4. zookeeper使用zookeeper-3.4.10.tar.gz,上传并解压

```
//解压zookeeper
tar -xzvf zookeeper-3.4.10.tar.gz
```

5. 为zookeeper准备配置文件

```
//进入conf目录
cd /root/apps/zookeeper/zookeeper-3.4.14/conf
//复制配置文件
cp zoo_sample.cfg zoo.cfg
//zookeeper根目录下新建data目录
mkdir data
// vi 修改配置文件中的dataDir
```

```
// 此路径用于存储zookeeper中数据的内存快照，及事务日志文件
dataDir=/root/apps/zookeeper/zookeeper-3.4.14/data
```

192.168.90.108 ×

```
-rw-rw-r--  1 2002 2002 2161 3月   7 2019 log4j.properties
-rw-r--r--  1 root root  953 4月  22 05:50 zoo.cfg
-rw-r--r--  1 root root 5013 4月  22 06:13 zookeeper.out
-rw-rw-r--  1 2002 2002  922 3月   7 2019 zoo_sample.cfg
[root@centos conf]# cat zoo.cfg
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=/tmp/zookeeper
dataDirLog=/tmp/zookeeper/log
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
#autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1

[root@centos conf]#
```


6. zookeeper启动命令

```
[root@centos ~]# systemctl stop firewalld #关闭防火墙的命令
[root@centos ~]# zkServer.sh start  #启动
[root@centos ~]# zkServer.sh stop   #关闭
[root@centos ~]# zkServer.sh status #查看运行状态

启动客户端：
[root@localhost ~]# zkCli.sh   //启动客户端
[zk: localhost:2181(CONNECTED) 0] ls /  #查看根节点命令
//查看zookeeper进程是否存在
[root@centos zookeeper-3.4.14]# ps -ef | grep zookeeper
```

三、zookeeper常用的Shell命令

3.1 新增节点

```
create [-s] [-e] path data   # 其中 -s 为有序节点， -e 临时节点
```

1. 创建持久化节点并写入数据：

```
create /hadoop  "123456"
```

2. 创建持久化有序节，此时创建的节点名为指定节点名+自增序号

```
//创建一个持久化节点
[zk: localhost:2181(CONNECTED) 1] create /hadoop "123456"
//通过get命令获取该节点的值
[zk: localhost:2181(CONNECTED) 2] get /hadoop
//创建一个有序的节点
[zk: localhost:2181(CONNECTED) 4] create -s /a "aa"
Created /a0000000001
//获取路径的值
[zk: localhost:2181(CONNECTED) 5] get /a0000000001
aa

//创建临时节点并获取值
[zk: localhost:2181(CONNECTED) 1] create -e /tmp "tmp"
Created /tmp
[zk: localhost:2181(CONNECTED) 2] get /tmp
tmp
```


3. 创建临时节点，临时节点会在回话过期后被删除;

```
[zk: localhost:2181(CONNECTED) 1] create -e /tmp "tmp"
Created /tmp
```

4. 创建临时有序节点，临时节点会在会话过期后被删除：

```
[zk: localhost:2181(CONNECTED) 1] create -s -e /tmp "tmp"
Created /tmp000000001
```

3.2 更新节点

```
//使用set命令来更新hadoop节点
[zk: localhost:2181(CONNECTED) 1] set /hadoop "345"
//根据版本号来更新节点
[zk: localhost:2181(CONNECTED) 1] set /hadoop "345" 2
```

也可以基于版本号来进行更改，此时类似于乐观锁机制，当你传入的数据版本号(dataVersion)和当前节点的数据版本号不符合时，zookeeper会拒绝本次修改：

3.3 删除节点

1. 删除节点的命令如下

```
delete path [version]
```

和更新节点数据一样，也可以传入版本号，当你传入的数据版本号(dataVersion)和当前节点的数据版本号不符合时，zookeeper不会执行删除操作。

```
//根据版本号来删除节点
[zk: localhost:2181(CONNECTED) 1] set /hadoop "345" 2
```

要想删除某个节点及其所有后代节点，可以使用递归删除，命令为 rmr path。

3.4 查看节点

```
get path
[zk: localhost:2181(CONNECTED) 1] get /hadoop
```

```
[zk: localhost:2181(CONNECTED) 0] create /hadoop2 "hadoop"
Created /hadoop2
[zk: localhost:2181(CONNECTED) 1] get /hadoop2
hadoop
cZxid = 0xb
ctime = Thu May 14 09:06:29 CST 2020
mZxid = 0xb
mtime = Thu May 14 09:06:29 CST 2020
pZxid = 0xb
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
```

```
dataLength = 6
numChildren = 0
[zk: localhost:2181(CONNECTED) 2]
```

节点各个属性如下表。其中一个重要的概念是Zxid(ZooKeeper Transaction Id)，ZooKeeper节点的每一个更改都具唯一的Zxid，如果Zxid1小于Zxid2,则Zxid1的更改发生在Zxid2更改之前。

状态属性	节点说明
cZxid	数据节点创建时的事务ID
ctime	数据节点创建世的时间
mZxid	数据节点最后一个更新是的事务ID
mtime	数据节点最后一个跟新时的时间
pZxid	数据节点的子节点最后一个被修改时的事务ID
cversion	子节点的更改次数
dataVerion	节点数据的更改次数
aclVersion	节点ACL的更改次数
ephemeralOwner	如果节点是临时节点，则表示创建该节点的会话的SeeesionID;如果是持久节点，则该属性值为0
dataLength	数据内容的长度
numChildren	数据节点当前的子节点个数

3.5 查看节点状态

可以使用stat命令查看节点状态，它的返回值和get命令类似，但不会返回节点数据

```
//使用stat命令来查看节点状态
[zk: localhost:2181(CONNECTED) 0] stat /hadoop
cZxid = 0x4
ctime = Thu May 14 07:56:33 CST 2020
mZxid = 0x4
mtime = Thu May 14 07:56:33 CST 2020
```

```
pZxid = 0x4
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 6
numChildren = 0
[zk: localhost:2181(CONNECTED) 1]
```

3.6 查看节点列表

查看节点列表有ls path 和ls2 path 两个命令，后者是前者的增强。不仅可以查看指定路径下的所有节点，还可以查看当前几点的信息

```
[zk: localhost:2181(CONNECTED) 5] ls /
[a0000000001, hadoop, zookeeper, tmp, hadoop2]
[zk: localhost:2181(CONNECTED) 6] ls2 /
[a0000000001, hadoop, zookeeper, tmp, hadoop2]
cZxid = 0x0
ctime = Thu Jan 01 08:00:00 CST 1970
mZxid = 0x0
mtime = Thu Jan 01 08:00:00 CST 1970
pZxid = 0xb
cversion = 3
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 0
numChildren = 5
[zk: localhost:2181(CONNECTED) 7]
```


3.7 监听器get path [watch]

使用get path [watch] 注册的监听器能够在节点内容发生改变的时候，向客户点发出通知。需要注意的是zookeeper的触发器是一次性的(One-time trigger)，触发一次后就会立即失效。

```
[zk: localhost:2181(CONNECTED) 8] get /hadoop watch
123456
cZxid = 0x4
ctime = Thu May 14 07:56:33 CST 2020
mZxid = 0x4
mtime = Thu May 14 07:56:33 CST 2020
pZxid = 0x4
cversion = 0
dataVersion = 0
aclVersion = 0
```

```
ephemeralOwner = 0x0
dataLength = 6
numChildren = 0
[zk: localhost:2181(CONNECTED) 9]
WATCHER::

WatchedEvent state:SyncConnected type:NodeDataChanged path:/hadoop
```

3.8 监听器stat path [watch]

使用stat path [watch] 注册的监听器能够在节点抓哪个台发生改变的时候，向客户点发出通知。

```
[zk: localhost:2181(CONNECTED) 1] stat /hadoop watch
cZxid = 0x4
ctime = Thu May 14 07:56:33 CST 2020
mZxid = 0xf
mtime = Thu May 14 10:01:04 CST 2020
pZxid = 0x4
cversion = 0
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 6
numChildren = 0
[zk: localhost:2181(CONNECTED) 2]
WATCHER::

WatchedEvent state:SyncConnected type:NodeDataChanged path:/hadoop
```


3.9 监听器ls\ls2 path [watch]

使用 ls path [watch] 或者 ls2 path [watch] 注册的监听器能够监听该节点下所有子节点的增加和删除操作。

四、zookeeper权限控制

4.1 概述

zookeeper类似于文件系统，client可以创建节点，更新节点，删除节点，那么如何做到节点的权限的控制呢？zookeeper的access control list 访问控制列表可以连接到这一点。acl权限控制，使用scheme：id：permission来表示，主要涵盖3个方面：

- 权限模式(scheme)：授权的策略

- 授权对象(id)：授权的对象
- 权限(permission)：授予的权限 其特性如下：
- zookeeper的权限控制是基于每个znode节点的，需要对每个节点设置权限
- 每个znode支持设置多种权限控制方案和多个权限
- 子节点不会继承父节点的权限，客户点无权访问某个节点，但可能可以访问他的子节点

```
setAcl /test2 ip:192.168.60.130:rewda //将节点权限设置为Ip:19
2.168.60.130的客户端可以对节点进行增、删、改、查、权限管理
```

4.2 权限模式

采用何种方式授权

方案	描述
world	只有一个用户：anyone，代表登录zooleeper所有人（默认）
ip	对客户端使用ip地址认证
auth	使用已添加认证的用户认证
digest	使用"用户名：密码"方式认证

4.3 授权的对象

给谁授权 授权对象ID是指，权限赋予的实体例如：ip地址或用户。

4.4 授予的权限

授予什么权限

create、delete、read、writer、admin也就是增、删、改、查、管理权限，这5种权限简写为cdrwa、注意：这5种权限中，delete是指对子节点的删除权限，其他4种权限只()对自身节点的权限操作。

权限	ACL简写	描述
create	c	可以创建子节点
delete	d	可以删除子节点(仅下一级节点)
read	r	可以读取节点数据及显示子节点列表
write	w	可以设置节点数据
admin	a	可以设置节点访问控制列表权限

4.5 授权的相关命令

命令	使用方式	描述
getAcl	getAcl	读取ACL权限
setAcl	setAcl	设置ACL权限
addauth	addauth	添加认证用户

4.6 案例

- world授权模式：命令

```
setAcl <path> world:anyone:<acl>
setAcl /hadoop world:anyone:cdrwa
```

案例

```
[zk: localhost:2181(CONNECTED) 2] getAcl /hadoop
'world,'anyone    #world方式对所有用户进行授权
: cdrwa           #增、删、改、查、管理
[zk: localhost:2181(CONNECTED) 3]
```

- IP 授权模式 命令

```
#需要两台机器来进行连接 192.168.60.129  192.168.60.130
#使用 192.168.60.129  登录zookeeper
zkCli.sh -server 192.168.60.130
#使用本机  192.168.60.130 zookeeper
zkCli.sh -server 192.168.60.130
```

- Auth授权模式 命令

```
addauth digest <user>:<password> #添加认证用户
setAcl <path>auth:<user>:<acl>
```

案例

```
[zk: localhost:2181(CONNECTED) 0] create /hadoop3 "hadoop3"
Created /hadoop3
[zk: localhost:2181(CONNECTED) 1] getAcl /hadoop3
'world,'anyone
: cdrwa
[zk: localhost:2181(CONNECTED) 2] addauth digest tyx:123
[zk: localhost:2181(CONNECTED) 3] setAcl /hadoop3 auth:tyx:cdrwa
cZxid = 0x16
ctime = Thu May 14 20:29:34 CST 2020
mZxid = 0x16
mtime = Thu May 14 20:29:34 CST 2020
pZxid = 0x16
cversion = 0
dataVersion = 0
aclVersion = 1
ephemeralOwner = 0x0
dataLength = 7
numChildren = 0
[zk: localhost:2181(CONNECTED) 7] getAcl /hadoop3
'digest,'tyx:nSk0WYb+XoISHNhIQiQ1BGsZHjE=
: cdrwa
[zk: localhost:2181(CONNECTED) 8]
```


- Digest 授权模式 命令

```
setAcl <path> digest:<user>:<password>:<acl>
```

这里的密码是经过SHA1及BASE64处理的密文，在SHEEL中可以通过命令计算：

```
echo -n <user>:<password> | openssl dgst -binary -sha1 | openssl base64
```

先来计算一个密文

```
echo -n tyx:123 | openssl dgst -binary -sha1 | openssl base64
#得到的密文
[root@centos zookeeper-3.4.14]# echo -n tyx:123 | openssl dgst -binary -sha1 | openssl base64
nSk0WYb+XoISHNhIQiQ1BGsZHjE=
```

案例：

```
[zk: localhost:2181(CONNECTED) 8] create /hadoop4 "hadoop4"
Created /hadoop4
[zk: localhost:2181(CONNECTED) 10] getAcl /hadoop4
'world,'anyone
: cdrwa
//使用digest进行授权
[zk: localhost:2181(CONNECTED) 11] setAcl /hadoop4 digest:tyx:nSk0WYb+XoISHNhIQiQ1BGsZHjE=:cdrwa

//该节点的权限
[zk: localhost:2181(CONNECTED) 13] getAcl /hadoop4
'digest,'tyx:nSk0WYb+XoISHNhIQiQ1BGsZHjE=
: cdrwa
//没有权限
[zk: localhost:2181(CONNECTED) 0] get /hadoop4
Authentication is not valid : /hadoop4
//添加授权用户
[zk: localhost:2181(CONNECTED) 1] addauth digest tyx:123
[zk: localhost:2181(CONNECTED) 2] get /hadoop4
hadoop4
cZxid = 0x19
ctime = Thu May 14 21:12:46 CST 2020
mZxid = 0x19
mtime = Thu May 14 21:12:46 CST 2020
pZxid = 0x19
cversion = 0
dataVersion = 0
aclVersion = 1
ephemeralOwner = 0x0
dataLength = 7
numChildren = 0
```


- 多种密室授权：同一个节点可以同时使用多种模式授权

```
[zk: localhost:2181(CONNECTED) 8] create /hadoop4 "hadoop4"
Created /hadoop4
//添加认证用户
[zk: localhost:2181(CONNECTED) 8] addauth digest itcast:123456
[zk: localhost:2181(CONNECTED) 8] setAcl /hadoop4 ip:192.168.60.129:cdrwa,auth:tyx:123:cdrwa,d
```

4.7 acl 超级管理员

zookeeper的权限管理模式有一种叫做super，该模式提供一个超级管理员，可以访问任何权限的节点

假设这个超级管理员是：super：admin，需要先为超级管理员生成密码的密文

```
echo -n super:admin | openssl dgst -binary -sha1 | openssl base64
```

那么打开zookeeper目录下的/bin/zkServer.sh 服务器脚本，找到如下一行：

```
nohup $JAVA "-Dzookeeper.log.dir=${ZOO_LOG_DIR}" "-
Dzookeeper.root.logger=${ZOO_LOG4J_PROP}"
```

这就是脚本中启动zookeeper的命令，默认只有以上两个配置项，我们需要加一个超管的配置项

```
"-Dzookeeper.DigestAuthenticationProvider.superDigest=super:xQJmxLMiHGwaqBvst5y6rkB6HQs="
```

那么修改以后这条完整命令变成了

java

```
nohup $JAVA "-Dzookeeper.log.dir=${ZOO_LOG_DIR}" "-
Dzookeeper.root.logger=${ZOO_LOG4J_PROP}" "-
Dzookeeper.DigestAuthenticationProvider.superDigest=super:xQJmxLMiHGwaqBvst5y6rkB6HQs="\
    -cp "$CLASSPATH" $JVMFLAGS $ZOOMAIN "$ZOOCFG" > "$_ZOO_DAEMON_OUT" 2>&1 < /dev/null &
```

之后启动zookeeper，输入如下的命令添加权限

```
addauth digest super:admin  #添加认证用户
```

让学习大数据不再难，谢谢大家。