

Zookeeper学习

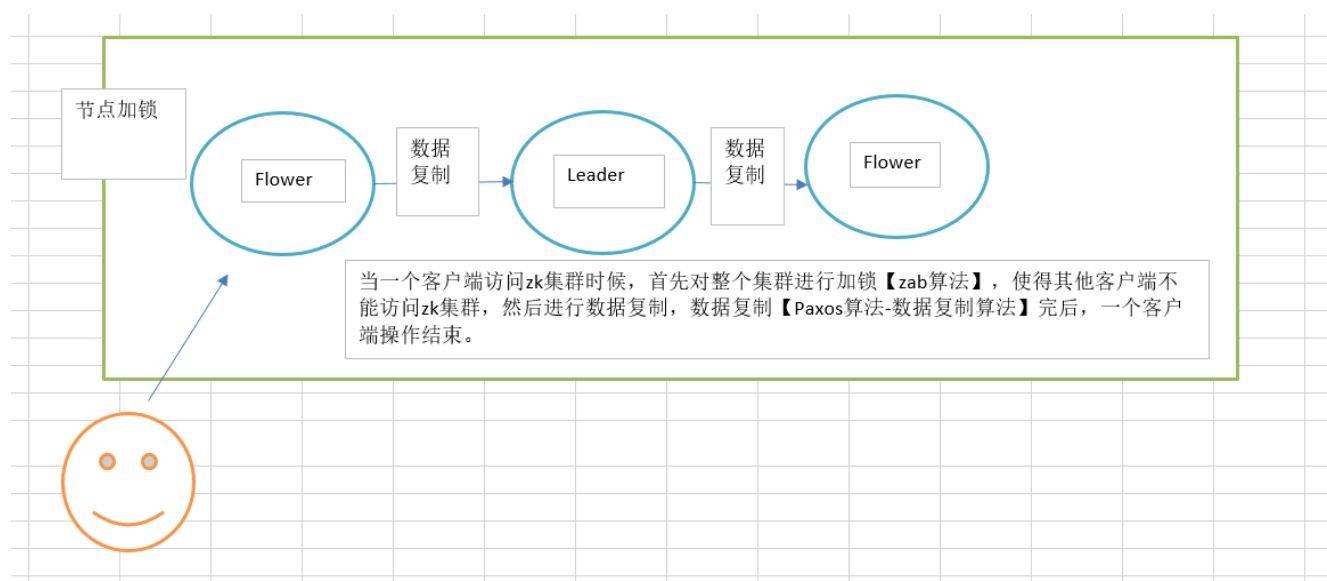
- 1 Zookeeper简介
- 2 搭建zk环境
- 3 Java操作zk的demo
- 4实际场景应用
- 5 zkClient 和 Curator框架简介
- 6 eclipse插件

1 Zookeeper简介

zookeeper是高效的分布式协调服务。它暴露了一些公用服务，比如命名/配置/管理/同步控制/群组服务。可以用zk

实现集群管理，leader选举，分布式锁，分布式队列。不适合做分布式事务。

zk不适合存储大量数据，而且不如mq。它基于zab算法【原子广播协议】和paxos的实现，该框架可以很好的保证分布式环境数据的一致性，是解决分布式一致性问题的利器。



zk特性介绍：

顺序一致性：从一个客户端发起事务请求，最终会严格地按照其发起顺序被应用到zk中。

原子性：所有事务请求的处理结果在整个集群所有机器上的应用情况是一致的，也就是说

要么整个集群所有的机器都成功应用了某一事务，要么都没有应用，一定不会

出现部分机器执行了该事物，而另一半机器没有应用的情况。

单一视图：无论客户端连接的是哪一个zk服务器，看到的数据模型都是一致的，也就是说

所有客户端看zk机器，其实都是一个服务器。

可靠性：一旦服务器成功地应用了某一事务，并完成对客户端的响应，那么该事务所引起的

服务器端状态，会被一致保留下来，除非有另一个事务对其更改。

2 搭建zk环境

一般建议奇数个节点，有利于选举算法的执行。

本机测试可以搭建一个节点，一个虚拟机。

也可以一个虚拟机搭建三个节点，伪集群。

也可以三个三个虚拟机搭建三个节点。

以下选择的三虚拟机三节点。

```
tar -zxvf zookeeper-3.4.5.tar.gz -C /usr/local/  
mv zookeeper-3.4.5/ zookeeper  
vim /etc/profile 【修改环境变量】 配置  
source /etc/profile  
cd zookeeper/conf  
mv zoo_sample.cfg zoo.cfg  
vim zoo.cfg  
mkdir data  
vim myid  
zkServer.sh start  
zkServer.sh status
```

配置文件

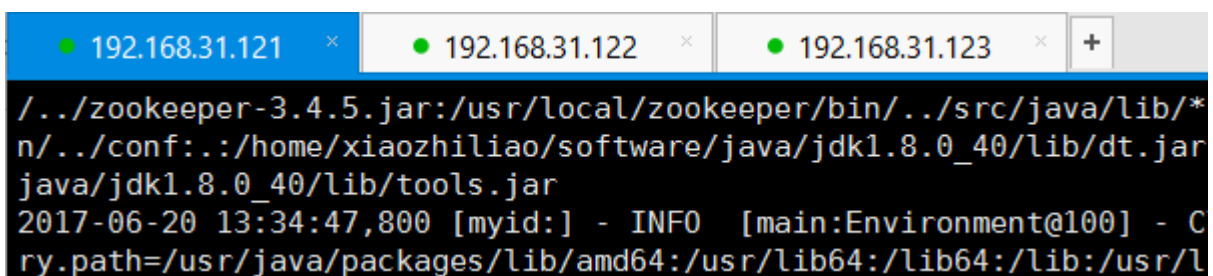
› 展开

原码

```
# 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=/usr/local/zookeeper/data  
# the port at which the clients will connect  
clientPort=2181  
#  
# 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  
server.0=192.168.31.121:2888:3888  
server.1=192.168.31.122:2888:3888  
server.2=192.168.31.123:2888:3888
```

进入zk客户端：

zkCli.sh



```
./zookeeper-3.4.5.jar:/usr/local/zookeeper/bin/../../src/java/lib/*  
n/../../conf:../home/xiaozhiliai/software/java/jdk1.8.0_40/lib/dt.jar  
java/jdk1.8.0_40/lib/tools.jar  
2017-06-20 13:34:47,800 [myid:] - INFO [main:Environment@100] - C  
ry.path=/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/l
```

```
WatchedEvent state:SyncConnected type:None path:null  
[zk: localhost:2181(CONNECTED) 0] █
```

zkShell命令:

```
操作zookeeper (shell)  
zkCli.sh 进入zookeeper客户端  
根据提示命令进行操作  
查找: ls / ls /zookeeper  
创建并赋值: create /bhz hadoop  
获取: get /bhz  
设值: set /bhz baihezhua  
可以看到zookeeper集群的数据一致性  
rmr /path 递归删除节点  
delete /path/child 删除指定某个节点  
创建节点有两种类型: 短暂 (ephemeral) 持久 (persistent)
```

3 Java操作zk的demo

demo

展开

原码

```
public class ZkClientBase {

    /** zookeeper地址 */
    static final String CONNECT_ADDR =
"192.168.31.121:2181, 192.168.31.122:2181:2181, 192.168.31.123:2181";
    /** session超时时间 */
    static final int SESSION_OUTTIME = 5000;//ms

    public static void main(String[] args) throws Exception {
        ZkClient zkc = new ZkClient(new ZkConnection(CONNECT_ADDR), 5000);
        //1. create and delete方法
        zkc.createEphemeral("/temp");
        // 递归创建c1,
        // zkc.createPersistent("/super/c1",new String[]{"dd","ddd"}, true); //每个节点没发给出value
        zkc.createPersistent("/super/c1", true);
        Thread.sleep(10000);
        zkc.delete("/temp");
        //递归删除
        zkc.deleteRecursive("/super");

        //2. 设置path和data 并且读取子节点和每个节点的内容
        zkc.createPersistent("/super", "1234");
        zkc.createPersistent("/super/c1", "c1内容");
        zkc.createPersistent("/super/c2", "c2内容");
        List<String> list = zkc.getChildren("/super"); //zkClient最大特点, 和watch没关系
        for (String p : list) {
            System.out.println(p);
            String rp = "/super/" + p;
            String data = zkc.readData(rp);
            System.out.println("节点为: " + rp + ", 内容为: " + data);
        }

        //3. 更新和判断节点是否存在
        zkc.writeData("/super/c1", "新内容");
        // System.out.println(zkc.readData("/super/c1"));
        System.out.println(zkc.exists("/super/c1"));

        //4. 递归删除/super内容
        zkc.deleteRecursive("/super");
    }
}
```

4实际场景应用

- 1配置管理。
- 2 集群管理。
- 3 发布和订阅。
- 4 数据库切换。
- 5 分布式日志收集。

6 分布式锁，分布式队列。

5 zkClient 和 Curator 框架简介

zkClient 和 Curator 是对 zk api 的二次封装。解决了重复注册事件监听的问题，并且提供了比原生 api 更加强大的功能。

6 eclipse 插件

与 eclipse 集成的管理 zookeeper 工具：

zookeeperBrowser

<http://www.massedynamic.org/eclipse/updates/>

