注:

OS:CentOS-7

JDK: jdk1.8.0\_131

Zookeeper: 3.4.10

ActiveMq: 5.14.5

测试使用场景:使用Zookeeper和ActiveMq配置MQ集群

# Zookeeper介绍

## 1.设计特性

1.最终一致性：client不论连接到哪个Server，展示给它都是同一个视图，这是zookeeper最重要的性能。

2 .可靠性：具有简单、健壮、良好的性能，如果消息m被到一台服务器接受，那么它将被所有的服务器接受。

3 .实时性：Zookeeper保证客户端将在一个时间间隔范围内获得服务器的更新信息，或者服务器失效的信息。但由于网络延时等原因，Zookeeper不能保证两个客户端能同时得到刚更新的数据，如果需要最新数据，应该在读数据之前调用sync()接口。

4 .等待无关（wait-free）：慢的或者失效的client不得干预快速的client的请求，使得每个client都能有效的等待。

5.原子性：更新只能成功或者失败，没有中间状态。

6 .顺序性：包括全局有序和偏序两种：全局有序是指如果在一台服务器上消息a在消息b前发布，则在所有Server上消息a都将在消息b前被发布；偏序是指如果一个消息b在消息a后被同一个发送者发布，a必将排在b前面。

## 2. Zookeeper 的典型应用场景

配置文件的管理、集群管理、分布式队列、同步锁、Leader 选举、队列管理等

# (一)Zookeeper安装

1.使用指令解压tar -zxvf /opt/zookeeper-3.4.10.tar.gz

2.cp conf/zoo\_sample.cfg conf/zoo.cfg 复制配置文件并修改配置文件

# 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=/opt/zookeeper-3.4.10/data

# the port at which the clients will connect

clientPort=2181

#server cinfig

server.0=192.168.28.101:2888:3888

server.1=192.168.28.102:2888:3888

server.2=192.168.28.103:2888:3888

3.在dataDir指定的目录下创建data文件夹并且创建myid文件

4.在myid文件上填写配置文件上server的id

5.启动集群bin/zkServer.sh start

注:

1.Zookeeper是基于JAVA开发的,需要JVM运行环境

2.Zookeeper集群过半的实例不可用导致整个集群不可用.

3.CentOS-7 默认的firewall是自动启动的,防火屏蔽了2181端口,需要开放端口或者关闭防火墙.

4.使用ZK客户端连接服务器的时候需要使用相同版本的ZK

# (二)Zookeeper客户端的使用

1.使用相同的Zookeeper版本zkCli.sh –server host:port 指令可以连接到对应的zk服务器端.

2.连接成功后可以使用help查看客户端的操作

[zk: 192.168.28.101:2181(CONNECTED) 0] help

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 --查看客户端历史操作

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

# (三)Znode stat structure

czxid:The zxid of the change that caused this znode to be created.

mzxid:The zxid of the change that last modified this znode.

pzxid:The zxid of the change that last modified children of this znode.

ctime:The time in milliseconds from epoch when this znode was created.

mtime:The time in milliseconds from epoch when this znode was last modified.

version:The number of changes to the data of this znode.

cversion:The number of changes to the children of this znode.

aversion:The number of changes to the ACL of this znode.

ephemeralOwner:The session id of the owner of this znode if the znode is an ephemeral node. If it is not an ephemeral node, it will be zero.

dataLength:The length of the data field of this znode.

numChildren:The number of children of this znode.

# (四)Zookeeper Java usage

## 1.在idea中安装Zookeeper插件可以在Idea中观察Zk集群的节点情况

## 2.使用原生的zk api

public static void main(String[] args) throws IOException, InterruptedException, KeeperException {  
 final ZooKeeper zooKeeper = new ZooKeeper(*connectString*, *sessionTimeout*, new Watcher() {  
 @Override  
 public void process(WatchedEvent event) {  
 Event.KeeperState state = event.getState();  
 Event.EventType type = event.getType();  
 if (state == Event.KeeperState.*SyncConnected* && type == Event.EventType.*None*){  
 System.*out*.println("connected zk server!");  
 *countDownLatch*.countDown();  
 }  
 }  
 });  
  
 *countDownLatch*.await();  
 System.*out*.println("run..." + zooKeeper);  
 byte[] data = zooKeeper.getData("/",false,null);  
 System.*out*.println(new String(data));  
 List<String> childrens = zooKeeper.getChildren("/",false);  
 for(String child : childrens){  
 System.*out*.println(child);  
 }  
 zooKeeper.close();  
}

## 3.curator框架.

Curator框架是apache的顶级项目

# (五)Zookeeper 使用实例构建ActiveMQ集群

## 1.解压ActiveMQ

tar -zxvf /opt/apache-activemq-5.14.5-bin.tar.gz -C /opt/

## 2.ActiveMQ配置

集群里面的Broker配置为相同的名字

<broker xmlns="http://activemq.apache.org/schema/core" brokerName="broker-cluster" dataDirectory="${activemq.data}">

将持久配置改为replicatedLevelDB,并且配置好Zookeeper地址,hostname要修改为对应实例的ip

<persistenceAdapter>

<!--<kahaDB directory="${activemq.data}/kahadb"/> -->

<replicatedLevelDB

directory="${activemq.data}/leveldb"

replicas="3"

bind="tcp://0.0.0.0:0"

zkAddress="192.168.28.101:2181,192.168.28.102:2181,192.168.28.103:2181"

hostname="localhost"

zkPath="/activemq/leveldb-stores"

/>

</persistenceAdapter>

## 3.先启动zk集群,然后再启动MQ

# (六)使用zookeeper实现分布式锁

## 1.实现步骤:

1.client调用create()方法创建“/root/lock\_”节点，注意节点类型是EPHEMERAL\_SEQUENTIAL

2.client调用getChildren("/root/lock\_",false)来获取所有已经创建的子节点，这里并不注册任何Watcher

3.客户端获取到所有子节点Path后，如果发现自己在步骤1中创建的节点是所有节点中最小的，那么就认为这个客户端获得了锁

4.如果在步骤3中，发现不是最小的，那么找到比自己小的那个节点，然后对其调用exist()方法注册事件监听

5.之后一旦这个被关注的节点移除，客户端会收到相应的通知，这个时候客户端需要再次调用getChildren("/root/lock\_",false)来确保自己是最小的节点，然后进入步骤3

## 2.Code