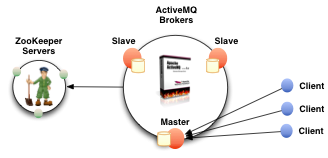
# ActiveMQ--4.基于LevelDB的高可用

ActiveMQ 5.9版本提供了基于LevelDB的高可用方式，包括数据的高可用和服务的高可用，一站式提供全套服务，很方便。

**服务高可用的原理：**使用zookeeper（集群）注册所有的ActiveMQ Broker。只有其中的一个Broker可以提供服务，被视为master，其他的Broker处于待机状态，被视为slave。当master由于死机等原因，不能提供服务，zookeeper会从slave中选举出一个Broker充当master。当原来的master Broker恢复继续提供服务的能力时，重新注册入zookeeper集群，作为slave待机。

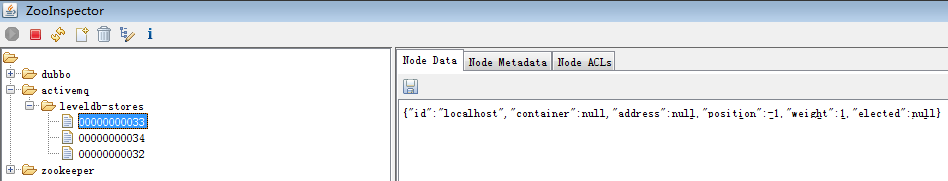


这时ActiveMQ的客户端只能访问master的Broker，其他处于slave的Broker不能访问。所以客户端连接Broker应该使用failover协议。

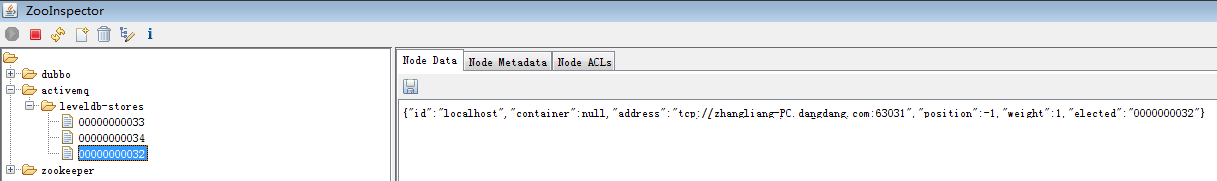
|  |  |
| --- | --- |
| 1 | failover:(tcp: //broker1:61616,tcp://broker2:61616,tcp://broker3:61616) |

这样即使当前的master Broker死机，zookeeper切换另一台机器为master，客户端也不需要重启和修改代码，完全透明。

下面是对zookeeper数据的抓图，可以看到activemq的有3个节点，分别是00000000033，00000000034，00000000032。这个图展现了00000000033的值，可以看到elected的值是null，表明这个节点是slave。



而master节点的数据内容如下图，可以看到elected的值是明确的写着00000000032，是被选举的master节点。



**消息数据高可用原理：**消息的操作会用同步的方式复制到所有的集群内的Broker中，一直阻塞到满足高可用，才会完成本次消息操作。满足消息高可用的条件是设置replicas值。举例说明，如果值设置为3，则高可用节点的法定数量 (3/2+1)=2个节点，消息会被master Broker存储到本地，并且至少存储到另一个远程的节点，必须2个节点都复制了消息，才算是消息安全存储。也就是说，如果replicas=3，说明当前是3个节点的集群。如果有1个节点不可用，不影响集群，可以继续提供服务。但是如果2个节点不可用，虽然还有1个节点是好的，但是由于可以提供服务的节点数量小于高可用的法定节点数量，则整个集群将不可用。

配置如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | < broker  xmlns = "<http://activemq.apache.org/schema/core>"  brokerName = "DemoBroker"  dataDirectory = "${activemq.data}" >       < persistenceAdapter >           < replicatedLevelDB             directory = "activemq-data"             replicas = "3"             bind = "<tcp://0.0.0.0:0>"             zkAddress = "zoo1.example.[org:2181,zoo2.example.org:2181,zoo3.example.org:2181](http://org:2181,zoo2.example.org:2181,zoo3.example.org:2181)"             zkPassword = "password"             zkPath = "/activemq/leveldb-stores"             hostname = "broker1.example.org"             />       </ persistenceAdapter >       ...其他配置  </ broker > |

**这里有一个配置点是需要注意的，我曾经在这里出错卡住过两小时。**就是每个集群的brokerName必须一致，否则就不会加到同一个集群里。用官方文档的原话就是：All the broker nodes that are part of the same replication set should have matching brokerName XML attributes.

下面是一个简单的性能测试结果，用以对比复制的LevelDB和单机LevelDB的性能差距

可以看出，发送消息性能差2-3倍，消费消息性能相差不多。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 发送1000条消息(毫秒) | 发送10000条消息(毫秒) | 消费1000条消息的时间(毫秒) | 消费10000条消息的时间(毫秒) |
| replicatedLevelDB(3节点) | 89138 | 611313 | 306 | 2628 |
| LevelDB | 34032 | 347712 | 220 | 2877 |

**最后，附上官方文档的一则警告，请使用者注意。replicatedLevelDB不支持延迟或者计划任务消息。这些消息存储在另外的LevelDB文件中，如果使用延迟或者计划任务消息，将不会复制到slave Broker上，不能实现消息的高可用。**