ActiveMQ 高可用集群安装、配置、高可用测试
（ZooKeeper + LevelDB）

从 ActiveMQ 5.9 开始，ActiveMQ 的集群实现方式取消了传统的 Master-Slave 方式，增加了基于 ZooKeeper + LevelDB 的 Master-Slave 实现方式，其他两种方式目录共享和数据库共享依然存在。

三种集群方式的对比：

(1)基于共享文件系统（KahaDB，默认）：

```
<persistenceAdapter>
    <kahaDB directory="${activemq.data}/kahadb"/>
</persistenceAdapter>
```

(2)基于 JDBC：

```
<bean id="mysql-ds" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
    <property name="url" value="jdbc:mysql://localhost:3306/amq?relaxAutoCommit=true"/>
    <property name="username" value="root"/>
    <property name="password" value="root"/>
    <property name="maxActive" value="20"/>
    <property name="poolPreparedStatements" value="true"/>
</bean>
<persistenceAdapter>
  <jdbcPersistenceAdapter dataDirectory="${activemq.data}" dataSource="#mysql-ds"
        createTablesOnStartup="false"/>
</persistenceAdapter>
```

(3)基于可复制的 LevelDB（本教程采用这种集群方式）：

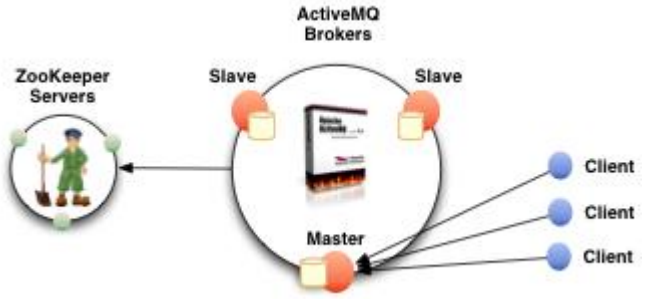
LevelDB 是 Google 开发的一套用于持久化数据的高性能类库。LevelDB 并不是一种服务,用户需要自行实现 Server。是单进程的服务，能够处理十亿级别规模 Key-Value 型数据，占用内存小。

```
<persistenceAdapter>
    <replicatedLevelDB
        directory="${activemq.data}/leveldb"
        replicas="3"
        bind="tcp://0.0.0.0:62621"
        zkAddress="localhost:2181,localhost:2182,localhost:2183"
        hostname="localhost"
        zkPath="/activemq/leveldb-stores"
    />
 </persistenceAdapter>
```

本节课程主要讲解基于 ZooKeeper 和 LevelDB 搭建 ActiveMQ 集群。集群仅提供主备方式的高可用集群功能，避免单点故障，没有负载均衡功能。

官方文档：http://activemq.apache.org/replicated-leveldb-store.html

集群原理图：



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。如果 Master 因故障而不能提供服务，ZooKeeper 会从 Slave 中选举出一个 Broker 充当 Master。

Slave 连接 Master 并同步他们的存储状态，Slave 不接受客户端连接。所有的存储操作都将被复制到连接至 Master 的 Slaves。如果 Master 宕了，得到了最新更新的 Slave 会成为 Master。故障节点在恢复后会重新加入到集群中并连接 Master 进入 Slave 模式。

所有需要同步的 disk 的消息操作都将等待存储状态被复制到其他法定节点的操作完成才能完成。所以，如果你配置了 replicas=3，那么法定大小是(3/2)+1=2。Master 将会存储并更新然后等待 (2-1)=1 个 Slave 存储和更新完成，才汇报 success。至于为什么是 2-1，熟悉 Zookeeper 的应该知道，有一个 node 要作为观擦者存在。当一个新的 Master 被选中，你需要至少保障一个法定 node 在线以能够找到拥有最新状态的 node。这个 node 可以成为新的 Master。因此，推荐运行至少 3 个 replica nodes，以防止一个 node 失败了，服务中断。（原理与 ZooKeeper 集群的高可用实现方式类似）

1、ActiveMQ 集群部署规划：

环境：CentOS 6.6 x64 、 JDK7

版本：ActiveMQ 5.11.1

ZooKeeper 集群环境：192.168.1.81:2181,192.168.1.82:2182,192.168.1.83:2183

（ZooKeeper 集群部署请参考《高可用架构篇--第 01 节--ZooKeeper 集群的安装、配置、高可用测试》）

主机	集群端口	消息端口	管控台端口	节点安装目录
192.168.1.81	62621	51511	8161	/home/wusc/activemq/node-01
192.168.1.82	62622	51512	8162	/home/wusc/activemq/node-02
192.168.1.83	62623	51513	8163	/home/wusc/activemq/node-03

2、防火墙打开对应的端口

3、分别在三台主机中创建/home/wusc/activemq 目录

```
$ mkdir /home/wusc/activemq
```

上传 apache-activemq-5.11.1-bin.tar.gz 到/home/wusc/activemq 目录

4、解压并按节点命名

```
$ cd /home/wusc/activemq
$ tar -xvf apache-activemq-5.11.1-bin.tar.gz
$ mv apache-activemq-5.11.1 node-0X   #(X 代表节点号 1、2、3，下同)
```

5、修改管理控制台端口（默认为 8161）可在 conf/jetty.xml 中修改，如下：

Node-01 管控台端口：

```
<bean id="jettyPort" class="org.apache.activemq.web.WebConsolePort" init-method="start">
    <!-- the default port number for the web console -->
    <property name="host" value="0.0.0.0"/>
    <property name="port" value="8161"/>
</bean>
```

Node-02 管控台端口：

```
<bean id="jettyPort" class="org.apache.activemq.web.WebConsolePort" init-method="start">
    <!-- the default port number for the web console -->
    <property name="host" value="0.0.0.0"/>
    <property name="port" value="8162"/>
</bean>
```

Node-03 管控台端口：

```
<bean id="jettyPort" class="org.apache.activemq.web.WebConsolePort" init-method="start">
    <!-- the default port number for the web console -->
    <property name="host" value="0.0.0.0"/>
    <property name="port" value="8163"/>
</bean>
```

6、集群配置：

在 3 个 ActiveMQ 节点中配置 conf/activemq.xml 中的持久化适配器。修改其中 bind、zkAddress、hostname 和 zkPath。注意：每个 ActiveMQ 的 BrokerName 必须相同，否则不能加入集群。

Node-01 中的持久化配置：

```
<broker xmlns="http://activemq.apache.org/schema/core" brokerName="DubboEdu" dataDirectory="${activemq.data}">
  <persistenceAdapter>
    <!-- kahaDB directory="${activemq.data}/kahadb"/ -->
    <replicatedLevelDB
        directory="${activemq.data}/leveldb"
        replicas="3"
        bind="tcp://0.0.0.0:62621"
        zkAddress="192.168.1.81:2181,192.168.1.82:2182,192.168.1.83:2183"
        hostname="edu-zk-01"
        zkPath="/activemq/leveldb-stores"
    />
  </persistenceAdapter>
</broker>
```

Node-02 中的持久化配置：

```
<broker xmlns="http://activemq.apache.org/schema/core" brokerName="DubboEdu" dataDirectory="${activemq.data}">
  <persistenceAdapter>
      <!-- kahaDB directory="${activemq.data}/kahadb"/ -->
      <replicatedLevelDB
            directory="${activemq.data}/leveldb"
            replicas="3"
            bind="tcp://0.0.0.0:62622"
            zkAddress="192.168.1.81:2181,192.168.1.82:2182,192.168.1.83:2183"
            hostname="edu-zk-02"
            zkPath="/activemq/leveldb-stores"
       />
  </persistenceAdapter>
</broker>
```

Node-03 中的持久化配置：

```
<broker xmlns="http://activemq.apache.org/schema/core" brokerName="DubboEdu" dataDirectory="${activemq.data}">
  <persistenceAdapter>
      <!-- kahaDB directory="${activemq.data}/kahadb"/ -->
      <replicatedLevelDB
            directory="${activemq.data}/leveldb"
            replicas="3"
            bind="tcp://0.0.0.0:62623"
            zkAddress="192.168.1.81:2181,192.168.1.82:2182,192.168.1.83:2183"
            hostname="edu-zk-03"
            zkPath="/activemq/leveldb-stores"
       />
  </persistenceAdapter>
</broker>
```

修改各节点的消息端口（注意，避免端口冲突）：

Node-01 中的消息端口配置：

```
<transportConnectors>
   <!-- DOS protection, limit concurrent connections to 1000 and frame size to 100MB -->
   <transportConnector name="openwire" uri="tcp://0.0.0.0:51511?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
   <transportConnector name="amqp" uri="amqp://0.0.0.0:5672?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
   <transportConnector name="stomp" uri="stomp://0.0.0.0:61613?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
   <transportConnector name="mqtt" uri="mqtt://0.0.0.0:1883?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
   <transportConnector name="ws" uri="ws://0.0.0.0:61614?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
</transportConnectors>
```

Node-02 中的消息端口配置：

```
<transportConnectors>
   <!-- DOS protection, limit concurrent connections to 1000 and frame size to 100MB -->
   <transportConnector name="openwire" uri="tcp://0.0.0.0:51512?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
   <transportConnector name="amqp" uri="amqp://0.0.0.0:5672?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
```

```
    <transportConnector name="stomp" uri="stomp://0.0.0.0:61613?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
    <transportConnector name="mqtt" uri="mqtt://0.0.0.0:1883?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
    <transportConnector name="ws" uri="ws://0.0.0.0:61614?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
</transportConnectors>
```

Node-03 中的消息端口配置：

```
<transportConnectors>
    <!-- DOS protection, limit concurrent connections to 1000 and frame size to 100MB -->
    <transportConnector name="openwire" uri="tcp://0.0.0.0:51513?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
    <transportConnector name="amqp" uri="amqp://0.0.0.0:5672?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
    <transportConnector name="stomp" uri="stomp://0.0.0.0:61613?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
    <transportConnector name="mqtt" uri="mqtt://0.0.0.0:1883?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
    <transportConnector name="ws" uri="ws://0.0.0.0:61614?maximumConnections=1000&amp;wireFormat.maxFrameSize=104857600"/>
</transportConnectors>
```

7、按顺序启动 3 个 ActiveMQ 节点：

```
$ /home/wusc/activemq/node-01/bin/activemq start
$ /home/wusc/activemq/node-02/bin/activemq start
$ /home/wusc/activemq/node-03/bin/activemq start
```

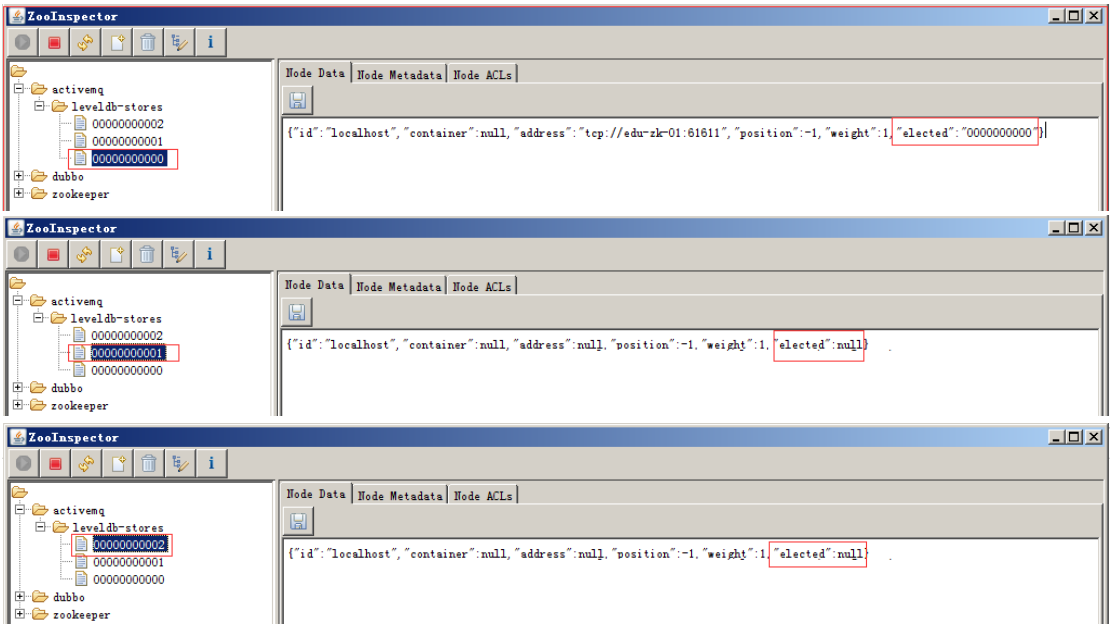
监听日志：

```
$ tail -f /home/wusc/activemq/node-01/data/activemq.log
$ tail -f /home/wusc/activemq/node-02/data/activemq.log
$ tail -f /home/wusc/activemq/node-03/data/activemq.log
```

8、集群的节点状态分析：

集群启动后对 ZooKeeper 数据的抓图，可以看到 ActiveMQ 的有 3 个节点，分别是 00000000000，00000000001，00000000002。

以下第一张图展现了 00000000000 的值，可以看到 elected 的值是不为空，说明这个节点是 Master，其他两个节点是 Slave。



9、集群可用性测试（配置和测试代码，请看视频）：

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。

failover:(tcp://192.168.1.81:51511,tcp://192.168.1.82:51512,tcp://192.168.1.83:51513)?randomize=false

10、集群高可用测试(请看视频)：

当一个 ActiveMQ 节点挂掉，或者一个 ZooKeeper 节点挂掉，ActiveMQ 服务依然正常运转。如果仅剩一个 ActiveMQ 节点，因为不能选举 Master，ActiveMQ 不能正常运转；同样的，如果 ZooKeeper 仅剩一个节点活动，不管 ActiveMQ 各节点是否存活，ActiveMQ 也不能正常提供服务。

（ActiveMQ 集群的高可用，依赖于 ZooKeeper 集群的高可用。）

11、设置开机启动：

```
# vi /etc/rc.local
su - wusc -c '/home/wusc/activemq/node-01/bin/activemq start'
su - wusc -c '/home/wusc/activemq/node-02/bin/activemq start'
su - wusc -c '/home/wusc/activemq/node-03/bin/activemq start'
```

12、配置优化(可选)：

updateURIsURL，通过 URL（或者本地路径）获取重连的 url，这样做具有良好的扩展性，因为客户端每次连接都是从 URL（或文件）中加载一次，所以可以随时从文件中更新 url 列表，做到动态添加 MQ 的备点。

failover:()?randomize=false&updateURIsURL=file:/home/wusc/activemq/urllist.txt

urllist.txt 中的地址通过英文逗号分隔，示例：

tcp://192.168.1.81:51511,tcp://192.168.1.82:51512,tcp://192.168.1.83:51513

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。

高可用+负载均衡实现介绍（下一节）：

Broker-Cluster 可以解实现载均衡，但当其中一个 Broker 突然宕掉的话，那么存在于该 Broker 上处于 Pending 状态的 message 将会丢失，无法达到高可用的目的。

Master-Slave 与 Broker-Cluster 相结合的部署。