MySQL高可用及读写分离

课程目标

- 理解keepalived软件的工作原理
- 能够通过keepalived实现mysql的高可用配置
- 了解常见的mysql读写分离中间件
- 掌握mysql的中间件mycat进行读写分离

一、keepalived介绍

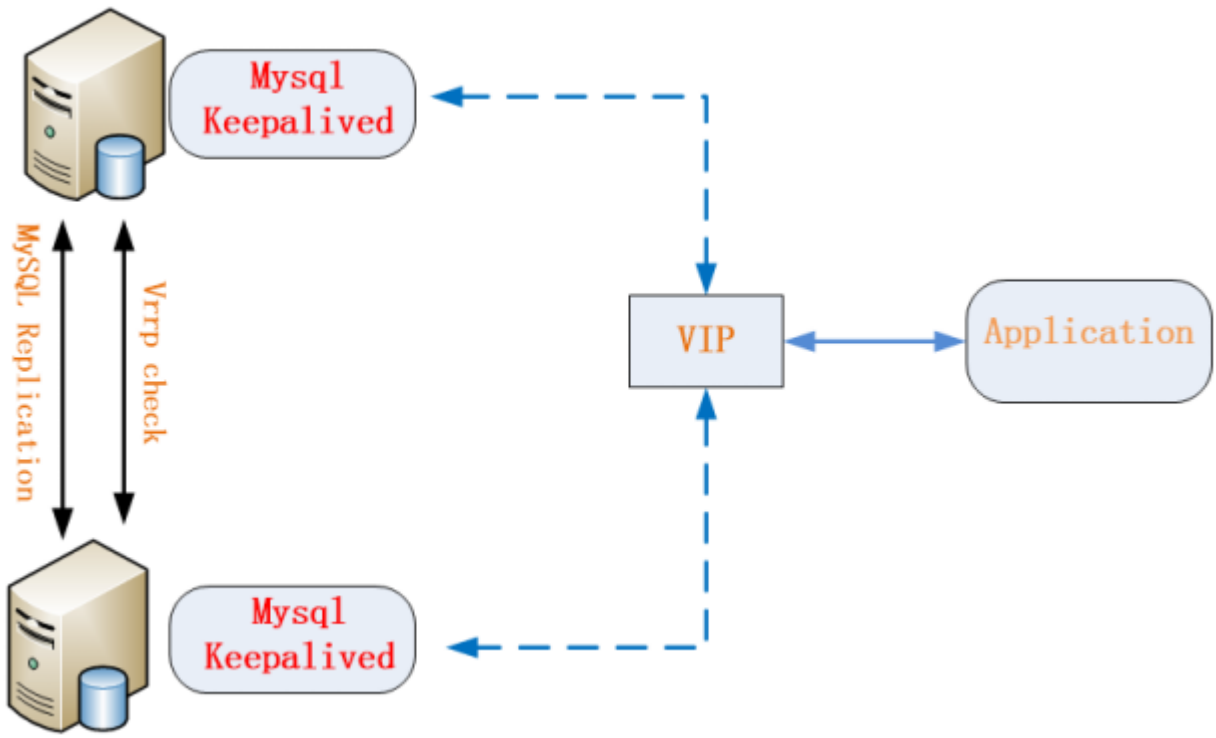
- **keepalived是什么？**

  keepalived是集群管理中保证集群高可用的一个服务软件,其功能类似heartbeat，用来防止单点故障。

- **keepalived工作原理**

  1. keepalived是以**VRRP协议**为实现基础的，VRRP全称Virtual Router Redundancy Protocol，即虚拟路由冗余协议。
  2. 虚拟路由冗余协议，可以认为是实现路由器高可用的协议，即将N台提供相同功能的路由器组成一个路由器组，这个组里面有一个master和多个backup，master上面有一个对外提供服务的vip（该路由器所在局域网内其他机器的默认路由为该vip），master会发组播，当backup收不到vrrp包时就认为master宕掉了，这时就需要根据VRRP的优先级来选举一个backup当master。这样的话就可以保证路由器的高可用了。
  3. keepalived主要有三个模块，分别是**core、check和vrrp**。core模块为keepalived的核心，负责主进程的启动、维护以及全局配置文件的加载和解析。check负责健康检查，包括常见的各种检查方式。vrrp模块是来实现VRRP协议的。

二、keepalived实现高可用



三、keepalived实现mysql高可用配置

1. 安装keepalived

```
说明:
关闭防火墙和selinux

在mysql互主的基础上配置keepalived(两台mysql都要安装)
# tar -xvf keepalived-1.2.24.tar.gz -C /usr/local/src/
# cd /usr/local/src/keepalived-1.2.24/
#./configure --prefix=/ --mandir=/usr/local/share/man/
# make
# make install
```

2. 了解配置文件

```
# man 5 keepalived.conf
[root@mysql01 ~]# cat /etc/keepalived/keepalived.conf
! Configuration File for keepalived

global_defs {                    #全局定义主要设置 keepalived 的通知机制和标识
   notification_email {
        root@localhost          #邮件地址可以多个，每行一个
   }
   notification_email_from      keepalived@localhost     #邮件的发件人
   smtp_server 127.0.0.1        #邮件服务的地址（本地）smtp协议
   smtp_connect_timeout 30     #连接smtp超时时间
   router_id test               #标识号，路由名字
}

vrrp_instance VI_1 {             定义vrrp实例组
    state BACKUP                  当前主机所扮演的模式master|backup
    interface eth0                实例绑定的网卡
    virtual_router_id 51   VRID   虚拟路由标识 00-00-5e-00-01-{VRID}
    priority 100                  优先级高为master，master 至少要高于 backup
    nopreempt                    设置不抢占，重新启动不夺回master角色（高的优先级加）
    advert_int 1                 检查间隔，默认为1秒
    authentication {             认证设置
        auth_type PASS           认证类型|验证:主备之间做身份验证   主备之间一定一致
        auth_pass 1111
    }
    virtual_ipaddress {     虚拟ip地址
        192.168.100.200
    }
}
virtual_server 192.168.100.200 3306 {     虚拟服务器所监听的端口（mysql端口）
    delay_loop 1                        服务轮询的时间间隔
    lb_algo wrr                         负载均衡的算法
    lb_kind DR                        负载均衡调度规则
    persistence_timeout 50              会话保持时间（秒为单位），即以用户在50秒内被分配到同一个后端
realserver
```

```
    protocol TCP                          健康检查的协议 TCP还是UDP
    real_server 192.168.1.100 3306 {      实际mysql的真实ip
        weight 1                    权重：0表示失效(不转发请求直到恢复正常)，默认是1
   notify_down                    /usr/local/etc/keepalived/mysql.sh      检查服务器失败(down)后，
要执行的脚本
        TCP_CHECK {                 下面是常用的健康检查方式
            connect_timeout 10          连接超时时间，单位秒
         bingto 192.168.1.100         健康检查的IP地址
            nb_get_retry 3          重连次数
            delay_before_retry 3        延迟时间3秒
            connect_port 3306           连接端口3306
        }
    }
```

扩展：
负载均衡器常见的调度算法：
1.轮询调度（Round-Robin,RR）
最简单的调度算法，LB按照顺序将请求依次转发给后端的RS，并没有考量后端RS的状态（处理速度以及响应时间）。大部分情况下，RS的性能状态都是各不一致的，这种算法显然无法满足合理利用资源的要求。

2.带权重的轮询调度（Weighted Round-Robin,WRR）
在轮询算法的基础上加上权重设置，权重越高的RS被分配到的请求越多。适用于按照服务器性能高低，配置不同的权重，以达到合理的资源利用。

3.最小连接调度（Least-Connection，LC）
把新的请求分配给连接数最少的RS。连接数少说明服务器空闲。

4.带权重的最小连接调度（Weight Least-Connection，WLC）
在最小连接算法的基础上加上权重设置，这样可以人为地控制请求分配。

5. 其他

负载均衡器的三种模式：
1）Virtual Server via NAT（VS-NAT）：
用地址翻译实现虚拟服务器。地址转换器有能被外界访问到的合法IP地址，它修改来自专有网络的流出包的地址。外界看起来包是来自地址转换器本身，当外界包送到转换器时，它能判断出应该将包送到内部网的哪个节点。优点是节省IP地址，能对内部进行伪装；缺点是效率低，因为返回给请求方的流量经过转换器。
2）Virtual Server via IP Tunneling （VS-TUN）：
用IP隧道技术实现虚拟服务器。这种方式是在集群的节点不在同一个网段时可用的转发机制，是将IP包封装在其他网络流量中的方法。为了安全的考虑，应该使用隧道技术中的VPN，也可使用租用专线。集群所能提供的服务是基于TCP/IP的Web服务、Mail服务、News服务、DNS服务、Proxy服务器等等.
3）Virtual Server via Direct Routing（VS-DR）：
用直接路由技术实现虚拟服务器。当参与集群的计算机和作为控制管理的计算机在同一个网段时可以用此法，控制管理的计算机接收到请求包时直接送到参与集群的节点。优点是返回给客户的流量不经过控制主机，速度快开销少。

3. 根据需求修改配置文件

需求：当master1挂掉后虚拟IP立马切换到master2上，反之亦然。
master1配置：

```
! Configuration File for keepalived

global_defs {
   notification_email {
     root@localhost.localdomain
   }
   notification_email_from root@localhost.localdomain
   smtp_server 127.0.0.1
   smtp_connect_timeout 30
   router_id master1.misshou.cc
}
vrrp_instance VI_1 {
    state MASTER
    interface ens37
    virtual_router_id 100
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        10.1.1.200
    }
}

virtual_server 10.1.1.200 3307 {
    delay_loop 1
    lb_algo wrr
    lb_kind DR
    persistence_timeout 5
    protocol TCP
    real_server 10.1.1.10 3307 {
        weight 1
   notify_down /etc/keepalived/mysql.sh
        TCP_CHECK {
            connect_timeout 1
            nb_get_retry 3
            delay_before_retry 3
            connect_port 3307
        }
    }
}


master2配置:

! Configuration File for keepalived

global_defs {
   notification_email {
     root@localhost.localdomain
```

```
   }
   notification_email_from root@localhost.localdomain
   smtp_server 127.0.0.1
   smtp_connect_timeout 30
   router_id master2.misshou.cc
}
vrrp_instance VI_1 {
    state BACKUP
    interface ens33
    virtual_router_id 100
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        10.1.1.200
    }
}

virtual_server 10.1.1.200 3307 {
    delay_loop 1
    lb_algo wrr
    lb_kind DR
    persistence_timeout 5
    protocol TCP
    real_server 10.1.1.20 3307 {
        weight 1
   notify_down /etc/keepalived/mysql.sh
        TCP_CHECK {
            connect_timeout 1
            nb_get_retry 3
            delay_before_retry 3
            connect_port 3307
        }
    }
}
```

4. 启动服务测试验证

```
启动keepalive服务:
[root@master1 keepalived]# systemctl start keepalived.service
[root@master2 keepalived]# systemctl start keepalived.service
查看vip:
[root@master1 keepalived]# ip a
...
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:4c:a3:0e brd ff:ff:ff:ff:ff:ff
    inet 10.1.1.10/24 brd 10.1.1.255 scope global ens37
       valid_lft forever preferred_lft forever
```

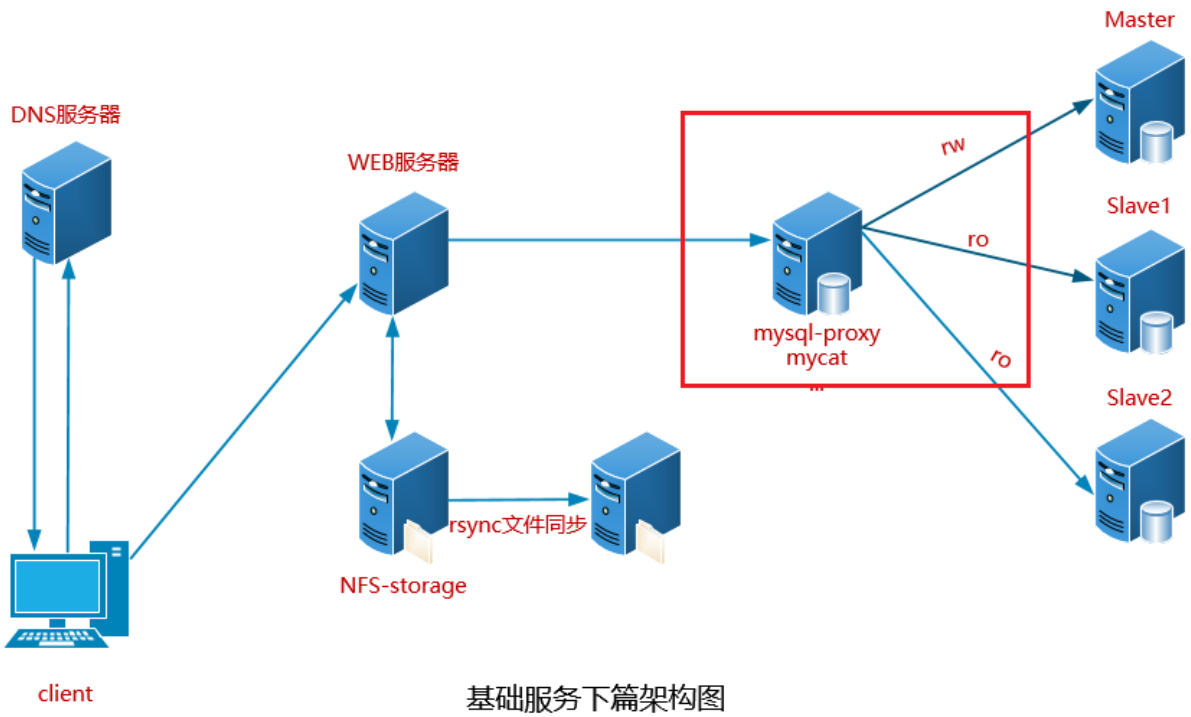
```
    inet 10.1.1.200/32 scope global ens37
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe4c:a30e/64 scope link
       valid_lft forever preferred_lft forever
```

模拟mysql故障，检查vip的漂移情况：
停止master1上的mysql服务

如果遇到问题：ipvs: Protocol not available
原因：ip_vs模块系统默认没有自动加载
解决：

```
modprobe -l|grep ip_vs
modprobe ip_vs
modprobe ip_vs_wrr
```

四、mycat实现读写分离



基础服务下篇架构图

1. 软件安装

1.1 jdk环境准备

```
1. 创建相应的目录
# mkdir /usr/local/java

2. 解压jdk软件包
# tar -xf jdk-10.0.1_linux-x64_bin.tar.gz -C /usr/local/java/
```

```
# ls /usr/local/java/jdk-10.0.1/
bin  conf  include  jmods  legal  lib  README.html  release
3. 配置环境变量
# vim /etc/profile
export JAVA_HOME=/usr/local/java/jdk-10.0.1
export JAVA_BIN=/usr/local/java/jdk-10.0.1/bin
export PATH=$PATH:/usr/local/java/jdk-10.0.1/bin
export CLASSPATH=./:/usr/local/java/jdk-10.0.1/lib:/usr/local/java/jdk-10.0.1/jre/lib
MYCAT_HOME=/usr/local/mycat/bin
export MYCAT_HOME

4. 重新读取环境变量文件
# source /etc/profile
或者
# . /etc/profile

5. 检查环境是否准备ok
# java -version      出现以下信息说明ok
java 10.0.1 2018-04-17
Java(TM) SE Runtime Environment 18.3 (build 10.0.1+10)
Java HotSpot(TM) 64-Bit Server VM 18.3 (build 10.0.1+10, mixed mode)
或者
# javac     能看到帮助提示就ok
```

1.2 mycat软件安装

```
1. 解压到指定的目录
# tar -xf Mycat-server-1.6.5-release-20180122220033-linux.tar.gz -C /usr/local/
# ls /usr/local/mycat/
bin  catlet  conf  lib  logs  version.txt

bin：程序存放位置
conf：配置文件位置
logs：日志位置
lib：存放mycat依赖的jar文件

2. 配置文件说明
schema.xml  //逻辑库定义和表以及分片、读写分离定义的配置文件
server.xml  //Mycat服务器参数调整和用户授权的配置文件

server.xml文件定义说明：

#用户名、密码、逻辑库名需要自定义
<user name="root" defaultAccount="true">  /
<!--root表示前端程序或客户端通过mycat中间件登录后台数据库的用户，自定义-->
                <property name="password">123456</property>
                <!--123456表示前端程序或客户端通过mycat中间件登录后台数据库的用户密码，自定义-->
                <property name="schemas">TESTDB</property>
            <!--TESTDB表示前端程序或客户端通过mycat中间件登录后台数据库的具体逻辑库，自定义-->

 </user>
```

schema.xml文件定义说明：
schema标签：
用于定义MyCat实例中的逻辑库，MyCat可以有多个逻辑库，每个逻辑库都有自己的相关配置。可以使用schema标签来划分这些不同的逻辑库。如果不配置 schema 标签，所有的表配置，会属于同一个默认的逻辑库。

dataNode：该属性用于绑定逻辑库到某个具体的 database上
table标签：定义了MyCat中的逻辑表，所有需要拆分的表都需要在这个标签中定义。

2. 读写分离配置

根据需求进行配置，客户端可以通过mycat用户密码123可以访问后台的db01和db02两个数据库

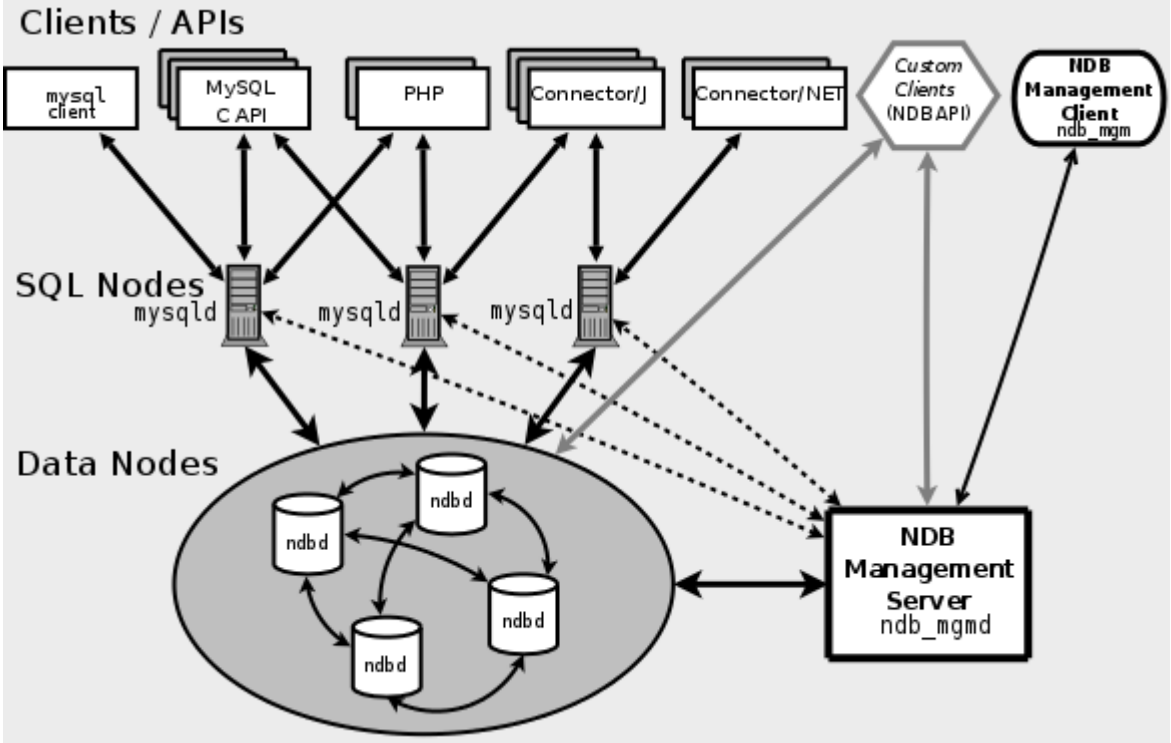
```
# vim server.xml
注意：该文件中出现的逻辑库名必须和后面schema.xml文件里定义
....
<user name="mycat" defaultAccount="true">
      <property name="password">123</property>
      <property name="schemas">DBB</property>
   </user>
....


# vim schema.xml
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">
        <schema name="DBB" checkSQLschema="false" sqlMaxLimit="100" dataNode="dn_test">
</schema>
        <dataNode name="dn_test" dataHost="dh_01" database="db01" />
        <dataHost name="dh_01" maxCon="1000" minCon="10" balance="1"
                          writeType="0" dbType="mysql" dbDriver="native" switchType="1"
 slaveThreshold="100">
                <heartbeat>select user()</heartbeat>
                <!-- can have multi write hosts -->
                <writeHost host="hostM1" url="10.1.1.20:3307" user="root"
                                    password="123">
                        <!-- can have multi read hosts -->
                        <readHost host="hostS1" url="10.1.1.30:3307" user="root" password="123"
/>
                </writeHost>
        </dataHost>
</mycat:schema>
```

3. 读写分离测试

```
1. 启动mycat
# chmod +x /usr/local/mycat/bin/mycat
# /usr/local/mycat/bin/mycat start
2. 检查是否起来
[root@mycat ~]# netstat -nltp|grep 8066
tcp6       0      0 :::8066                 :::*                    LISTEN      14029/java

[root@mycat ~]# netstat -nltp|grep 9066
tcp6       0      0 :::9066                 :::*                    LISTEN      14029/java

说明:
8066端口是mycat的代理端口，用于客户连接后台mysql数据库
9066端口是mycat的管理端口，用于管理监控mysql数据库

# /usr/local/mysql/bin/mysql -h 10.1.1.10 -P9066 -umycat -p123
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.29-mycat-1.6.5-release-20180122220033 MyCat Server (monitor)
...
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show @@help;     查看帮助
mysql> reload @@config;
该命令用于更新配置文件，例如更新schema.xml文件后在命令行窗口输入该命令，可不用重启即进行配置文件更新。

mysql> show @@database;    //该命令用于显示MyCAT的数据库的列表，对应schema.xml配置文件的schema子节点
+----------+
| DATABASE |
+----------+
| TESTDB1  |
| TESTDB2  |
+----------+
2 rows in set (0.01 sec)
mysql> show @@datanode;
该命令用于显示MyCAT的数据节点的列表，对应schema.xml配置文件的dataNode节点
其中，“NAME”表示dataNode的名称；“dataHost”表示对应dataHost属性的值，即数据主机；“ACTIVE”表示活跃连接数；“IDLE”表示闲置连接数；“SIZE”对应总连接数量。

mysql> show @@version;     //该命令用于获取MyCAT的版本
+-----------------------------------------+
| VERSION                                 |
+-----------------------------------------+
| 5.6.29-mycat-1.6.5-release-20180122220033 |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> show @@sql.slow;    //该命令用于查询运行缓慢的SQL语句
mysql> Show @@heartbeat;   //当前后端物理库的心跳检测情况,RS_CODE为1表示心跳正常
```

```
负载均衡类型有 3 种:
1. balance="0", master可读可写，slave不使用，也就是不启动读写分离
2. balance="1",master可读可写，slave只读；stand by writeHost 参与 select 查询做负载均衡
3. balance="2",master可读可写，slave只读；所有读操作都随机的在 master和slave上分配。
4. balance="3",master只写，slave只读；1.4版本以后有

switchType 属性
-1 表示不自动切换
1 默认值,自动切换
2 基于 MySQL 主从同步的状态决定是否切换

更多内容请查看《Mycat 权威指南》
```

五、mysql-cluster（扩展）



- **集群成员：**
  - 管理节点(MGM)　　进程名**ndb_mgmd**
    数据节点(Data Nodes)　　进程名**ndbd**
    SQL节点(Sql Nodes)　　进程名**mysqld**
- **注意：**
  1. 所有节点均不安装和启动mysql-server软件包，仅安装集群软件包 mysql-cluster-gpl-7.4.12.tar.gz
  2. Mysql Cluster采用的是NDB存储引擎，因此在建表时必须指定ENGINE为ndbcluster，这是一种内存式的存储引擎，因此对内存要求很高

1. 环境准备

```
                10.1.1.2              管理节点   mgm
        10.1.1.2      10.1.1.3       SQL节点   sql1 sql1
        10.1.1.4      10.1.1.5       数据节点 data1 data2
```

2. 详细操作步骤

```
一、初始配置（所有节点）
IP、Iptables、SELinux、hostname解析
卸载已经安装了的mysql软件包

二、Installing a MySQL Cluster Binary Release on Linux
SQL nodes.
[root@sql1 ~]# useradd mysql
[root@sql1 ~]# tar -xf mysql-cluster-gpl-7.4.12-linux-glibc2.5-x86_64.tar.gz -C /usr/local/
[root@sql1 ~]# ln -sv /usr/local/mysql-cluster-gpl-7.4.12-linux-glibc2.5-x86_64/
/usr/local/mysql
[root@sql1 ~]# cd /usr/local/mysql
[root@sql1 ~]# chown -R mysql.mysql .
[root@sql1 mysql]# scripts/mysql_install_db --user=mysql
[root@sql1 mysql]# cp support-files/mysql.server /etc/rc.d/init.d/mysql-cluster
[root@sql1 mysql]# chmod +x /etc/rc.d/init.d/mysql-cluster
[root@sql1 mysql]# chkconfig --add mysql-cluster
[root@sql1 mysql]# chkconfig mysql-cluster on

Data nodes.
[root@data1 ~]# rsync -va sql1:/usr/local/mysql/bin/ndbd /usr/local/bin/
[root@data1 ~]# rsync -va sql1:/usr/local/mysql/bin/ndbmtd /usr/local/bin/
[root@data1 ~]# chmod a+x /usr/local/bin/ndb*

Management nodes.
[root@mgm ~]# rsync -va sql1:/usr/local/mysql/bin/ndb_mgm* /usr/local/bin/
[root@mgm ~]# chmod a+x /usr/local/bin/ndb_mgm*
[root@mgm ~]# mkdir -p /usr/local/mysql/mysql-cluster


三、Initial Configuration of MySQL Cluster
Configuring the data nodes and SQL nodes.
[root@sql1 mysql]# vim /etc/my.cnf
[mysqld]
# Options for mysqld process:
ndbcluster

[mysql_cluster]
# Options for MySQL Cluster processes:
ndb-connectstring=10.1.1.10       【管理节点】

[root@sql1 mysql]# rsync -va /etc/my.cnf sql2:/etc/
[root@sql1 mysql]# rsync -va /etc/my.cnf data1:/etc/
[root@sql1 mysql]# rsync -va /etc/my.cnf data2:/etc/

Configuring the management node.
[root@mgm ~]# cd /usr/local/mysql/mysql-cluster
```

```
[root@mgm mysql-cluster]# vim config.ini
[ndbd default]
# Options affecting ndbd processes on all data nodes:
NoOfReplicas=2          # Number of replicas
DataMemory=80M    # How much memory to allocate for data storage
IndexMemory=18M   # How much memory to allocate for index storage
                          # For DataMemory and IndexMemory, we have used the
                          # default values. Since the "world" database takes up
                          # only about 500KB, this should be more than enough for
                          # this example Cluster setup.

[tcp default]
# TCP/IP options:
portnumber=3306      # This the default; however, you can use any
                          # port that is free for all the hosts in the cluster
                          # Note: It is recommended that you do not specify the port
                          # number at all and simply allow the default value to be used
                          # instead

[ndb_mgmd]
# Management process options:
id=1
hostname=10.1.1.2         # Hostname or IP address of MGM node
datadir=/usr/local/mysql/mysql-cluster  # Directory for MGM node log files

[mysqld]
# SQL node options:
id=2
hostname=10.1.1.2

[mysqld]
# SQL node options:
id=3
hostname=10.1.1.3
[ndbd]
# Options for data node "data1":
id=4
hostname=10.1.1.4       # Hostname or IP address
datadir=/usr/local/mysql/data   # Directory for this data node's data files

[ndbd]
# Options for data node "data2":
id=5
hostname=10.1.1.5         # Hostname or IP address
datadir=/usr/local/mysql/data   # Directory for this data node's data files


四、Initial Startup of MySQL Cluster
1. Management nodes
启动管理节点:
[root@mgm ~]# ndb_mgmd -f /usr/local/mysql/mysql-cluster/config.ini

2. data node
```

启动数据节点：

```
[root@data1 ~]# ndbd
2015-02-17 10:46:23 [ndbd] INFO     -- Angel connected to '10.1.1.2:1186'
2015-02-17 10:46:23 [ndbd] INFO     -- Angel allocated nodeid: 4

[root@data2 ~]# ndbd
2015-02-17 10:47:15 [ndbd] INFO     -- Angel connected to '10.1.1.2:1186'
2015-02-17 10:47:15 [ndbd] INFO     -- Angel allocated nodeid: 5

# echo "ndbd"  >> /etc/rc.local     //设置数据节点开机运行

3. SQL node
[root@sql1 mysql]# /etc/init.d/mysql-cluster start
Starting MySQL.....................................         [  OK  ]
[root@sql1 mysql]# chkconfig mysql-cluster on

[root@sql2 mysql]# /etc/init.d/mysql-cluster start
Starting MySQL.....................................         [  OK  ]
[root@sql2 mysql]# chkconfig mysql-cluster on

4. Management nodes
```

管理节点监控集群状态：

```
[root@mgm ~]# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: 10.1.1.2:1186
Cluster Configuration
---------------------
[ndbd(NDB)] 2 node(s)
id=4  @10.1.1.4  (mysql-5.6.31 ndb-7.4.12, Nodegroup: 0, *)
id=5  @10.1.1.5  (mysql-5.6.31 ndb-7.4.12, Nodegroup: 0)

[ndb_mgmd(MGM)]   1 node(s)
id=1  @10.1.1.2  (mysql-5.6.31 ndb-7.4.12)

[mysqld(API)]  2 node(s)
id=2  @10.1.1.2  (mysql-5.6.31 ndb-7.4.12)
id=3  @10.1.1.3  (mysql-5.6.31 ndb-7.4.12)
```

五、测试：

1. 在所有SQL节点建立授权用户

```
[root@sql1 ~]# mysql
mysql> grant all on *.* to root@'%' identified by '456';
Query OK, 0 rows affected (0.04 sec)
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

2. 从客户端连接测试

```
[root@client1 ~]# mysql -h 10.1.1.3 -uroot -p123
```

测试一：所有节点均正常，写入或查询

```
mysql> create table t1(id int)engine=ndbcluster;
```

```
mysql> alter table uplooking.t1 engine=ndbcluster;        //改变已创建表的存储引擎
```

测试二：停用一个SQL节点

```
[root@sql1 ~]#service mysql-cluster stop
```

测试三：停用一个数据节点

```
[root@mgm ~]# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm>4 stop
```