第 19 章

负载均衡、高可用的 MySQL 集群

本章我们来介绍 MySQL 数据库服务器的集群。

网上基于 Debian 或者 Ubuntu 的 MySQL 数据库集群教程，几乎都是下载 MySQL 源代码，自行编译、安装；这样做的坏处前面已经介绍过，就是安全更新很麻烦。所以，在条件允许的情况下，要尽量使用 Ubuntu 官方的软件包。

Ubuntu 带的 MySQL 服务器软件包，已经包含了 MySQL 集群所需的 3 个组件，所以我们根本不需要去从源代码编译安装 MySQL。这 3 个组件分别是：

- MySQL
- MySQL NDB
- MySQL NDB Management

MySQL 使用 NDB 来实现集群。NDB 是一种“内存中”的存储引擎，可用性高、数据一致性好。

19.1 MySQL 集群架构介绍

19.1.1 架构图

图 19.1 来自于 MySQL 官方网站。通过该架构图，你就很容易理解 MySQL 集群是怎么工作的了。数据保存在存储节点（Data Nodes）中，也就是图中的 ndbd 节点。SQL 语句在 SQL 节点上执行，也就是图中的 mysqld 节点。集群的管理者是右下角的 NDB 管理服务器，也就是 ndb_mgmd 节点。

关于 MySQL 集群的 3 个主要组成部分，我们再来介绍一下。

（1）负载均衡节点（mysql）

- 负载均衡节点（也叫 SQL 节点）是用来**访问**集群数据的。相关的软件，就是我们平时所使用的 MySQL 数据库软件；也就是由/etc/init.d/mysql 脚本来管理的那个服务。

（2）存储节点（ndbd）

- 数据存储节点是用来**保存**集群数据的，其服务的启停是由脚本/etc/init.d/mysql-ndb 来管理的。

（3）管理节点（ndbd-mgm）

- 管理节点是用来管理集群内其他节点的，比如提供配置信息、启动或停止节点、执行备份等。其服务的启停是由/etc/init.d/mysql-ndb-mgm 脚本来管理的。由于这类节点是管理者，所以管理节点必须首先启动，然后其他两类节点再启动。

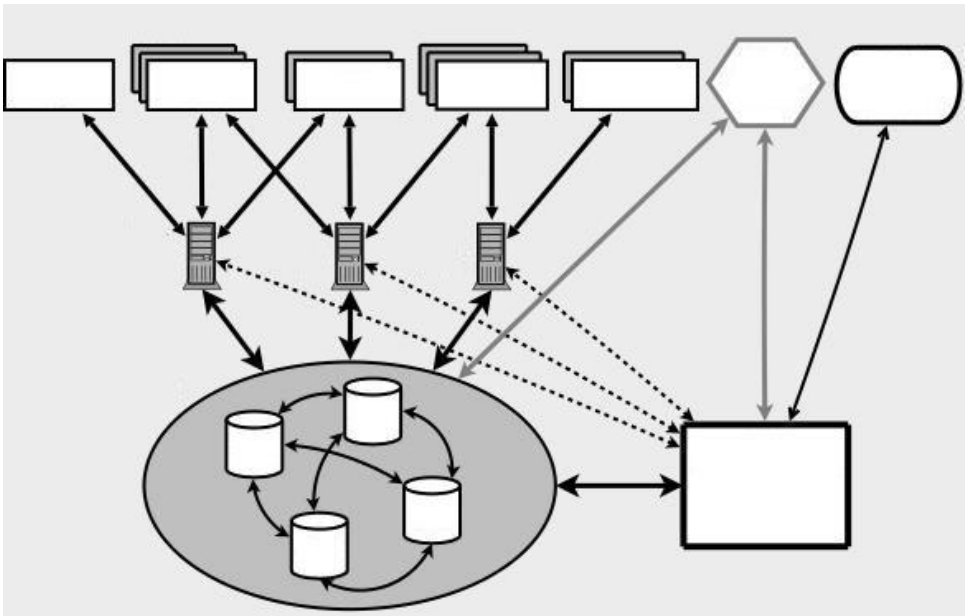

图 19.1　MySQL 集群架构

19.1.2　本例中的服务器

本例中，我们将使用 5 台 Ubuntu 服务器，它们的作用和 IP 配置如下：

- mysql-mgm.mytest.com：192.168.1.10，管理节点
- mysql-lb1.mytest.com：192.168.1.11，负载均衡节点 1
- mysql-lb2.mytest.com：192.168.1.12，负载均衡节点 2
- mysql-data1.mytest.com：192.168.1.13，数据节点 1
- mysql-data2.mytest.com：192.168.1.14，数据节点 2

此外，我们需要有一个虚拟 IP 地址，作为整个集群对外的一个 IP 地址；各种应用程序都通过该 IP 地址来访问 MySQL。

请你先设置好以上服务器的 hostname 和 IP 地址。你可以在 5 台物理服务器上做实验，也可以在虚拟机里面做。

下面我们分别来安装、配置这些服务器。

19.2　管理节点（MGM）的安装及配置

Ubuntu 8.04 里面的 MySQL 版本是 5.0.51a。在软件包 mysql-server-5.0 中，不仅携带

了集群服务器管理程序（ndb_mgmd），还携带了集群管理客户端（ndb_mgm）。客户端 ndb_mgm 可以用来显示集群的状态，让你了解集群的工作情况。

本节我们来安装、配置 mysql-mgm.mytest.com（192.168.1.10）。

19.2.1　安装 MySQL

在 mysql-mgm.mytest.com 服务器上，安装 MySQL 服务器：

```
$ sudo apt-get update install mysql-server
```

安装时，安装程序会要求你设置 MySQL 的 root 口令。请设置好，并记住该口令。

19.2.2　配置 ndb_mgmd.cnf

现在我们来创建 MySQL 的集群配置文件，该文件路径为/etc/mysql/ndb_mgmd.cnf：

```
$ sudo nano /etc/mysql/ndb_mgmd.cnf
```

> **警告**　文件名必须是 ndb_mgmd.cnf，否则服务无法启动。

输入如下内容：

```
[NDBD DEFAULT]
NoOfReplicas=2

[MYSQLD DEFAULT]
[NDB_MGMD DEFAULT]
[TCP DEFAULT]

[NDB_MGMD]    # 管理节点
HostName=192.168.1.10    # 本机（管理节点）的 IP 地址

[NDBD]    # 存储节点 1
HostName=192.168.1.13
DataDir=/var/lib/mysql-cluster
BackupDataDir=/var/lib/mysql-cluster/backup

[NDBD]    # 存储节点 2
HostName=192.168.1.14
DataDir=/var/lib/mysql-cluster
BackupDataDir=/var/lib/mysql-cluster/backup

# 有几个存储节点，就写几行[MYSQLD]
[MYSQLD]
[MYSQLD]
```

文件保存后，即可启动 MySQL 的 mgm 服务了：

```
$ sudo /etc/init.d/mysql-ndb-mgm start
```

以后，如果要停止 MySQL 集群，只需要停止该 mgm 服务即可。

19.3 存储节点（NDB）的安装及配置

本节我们来安装、配置 mysql-data1.mytest.com 和 mysql-data2.mytest.com。

19.3.1 安装 MySQL

在 mysql-data1 和 mysql-data2 两台服务器上，分别安装 MySQL 服务器：

```
$ sudo apt-get update install mysql-server
```

安装时，安装程序会要求你设置 MySQL 的 root 口令。请设置好，并记住该口令。

程序安装完成后，请先停止 MySQL 服务：

```
$ sudo /etc/init.d/mysql stop
```

19.3.2 配置 my.cnf

在 mysql-data1 和 mysql-data2 两台服务器上，要执行相同的操作。

首先，备份原有的/etc/mysql/my.cnf：

```
$ sudo mv /etc/mysql/my.cnf /etc/mysql/my.cnf-back
```

然后，编辑一个新的 my.cnf：

```
$ sudo nano /etc/mysql/my.cnf
```

内容如下：

```
[client]
socket = /var/run/mysqld/mysqld.sock
port    = 3306

[mysqld]
ndbcluster
ndb-connectstring=192.168.1.10   # 管理节点的 IP 地址
default-storage-engine=NDBCLUSTER

[mysql_cluster]
ndb-connectstring=192.168.1.10   # 管理节点的 IP 地址
```

文件保存后，就可以启动 ndb 服务了：

```
$ sudo /etc/init.d/mysql-ndb start-initial
```

提示 平时启动 ndb 服务时，用/etc/init.d/mysql-ndb start 即可。在下列情况下，要使用/etc/init.d/mysql-ndb start-initial（“初始化”启动）：

- 第一次启动 ndb 服务；
- 管理节点更改配置后。

如果你在启动 ndb 服务时遇到下列错误：

```
 * Starting MySQL NDB Data Node ndbd                                  error=2350
2009-02-15 22:20:55 [ndbd] INFO     -- Error handler restarting system
2009-02-15 22:20:55 [ndbd] INFO     -- Error handler shutdown completed - exiting
sphase=0
exit=-1
```

则说明 ndb 服务已经启动了。使用下面的命令可以查看 ndb 服务是否已经启动：

```
$ ps aux|grep ndb|grep -v grep
```

现在，可以启动 mysql 服务了：

```
$ sudo /etc/init.d/mysql start
```

19.4　阶段测试

到目前为止，存储的集群已经做好了（负载均衡还没有做）。
现在，我们来进行一些必要的测试。

19.4.1　集群连接状态测试

首先，我们回到管理节点（mysql-mgm.mtest.com）上。在该服务器上，执行下面的命令：

```
$ ndb_mgm
```

命令执行后，就会进入 MGM 的客户端界面，并显示 mgm 提示符：

```
-- NDB Cluster -- Management Client --
ndb_mgm>
```

在该提示符下，输入 show 命令，来查看当前连接状态：

```
ndb_mgm> show
```

在正常情况下，应该显示如下内容：

```
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.1.13  (Version: 5.0.51, Nodegroup: 0)
id=3    @192.168.1.14  (Version: 5.0.51, Nodegroup: 0, Master)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.1.10  (Version: 5.0.51)
```

```
[mysqld(API)]   2 node(s)
id=4    @192.168.1.13  (Version: 5.0.51)
id=5    @192.168.1.14  (Version: 5.0.51)
```

从上述内容我们可以看到，两个 NDB 节点 192.168.1.13 和 192.168.1.14 都已经连接到管理节点上来了。配置成功！

执行 quit 或者 exit 命令，退出 MGM 客户端：

```
ndb_mgm> quit
```

19.4.2 测试

现在，让我们来看看集群在数据存储方面是否正常。我们将分别在两个存储节点上进行数据操作。

1. 数据同步测试

首先，我们在节点 mysql-data1.mytest.com 上，创建一个数据库，并插入一行数据。

```
$ mysql -u root -p
```

输入密码后，就进入了 MySQL 客户端的命令行界面。在该界面中，输入如下命令：

```
mysql> CREATE DATABASE clustertest;
Query OK, 1 row affected (0.24 sec)

mysql> USE clustertest;
Database changed

mysql> CREATE TABLE testtable (Count INT) ENGINE=NDBCLUSTER;
Query OK, 0 rows affected (0.24 sec)

mysql> INSERT INTO testtable () VALUES (1);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM testtable;
+-------+
| Count |
+-------+
|     1 |
+-------+
1 row in set (0.00 sec)
```

上述命令创建了一个叫做 clustertest 的数据库，在该数据库中创建了一个名为 testtable 的表，该表的类型为 NDBCLUSTER，也就是采用 NDBCLUSTER 作为数据库引擎。该表中有一个字段叫做 Count。我们还向该表插入了一条数据，其 Count 字段的值为 1。

现在，我们再到 mysql-data2.mytest.com 节点上，创建一个同名的数据库。我们仅需要

创建数据库，然后，该数据库的所有数据都会自动复制过来。

```
$ mysql -u root -p
```

输入密码后，就进入了 MySQL 客户端的命令行界面。在该界面中，输入如下命令：

```
mysql> CREATE DATABASE clustertest;
Query OK, 1 row affected (0.24 sec)

mysql> USE clustertest;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

mysql> SELECT * FROM testtable;
+-------+
| Count |
+-------+
|     1 |
+-------+
1 row in set (0.03 sec)
```

看到了吧，数据已经从 mysql-data1.mytest.com 复制到 mysql-data2.mytest.com 这个节点上了。

现在，我们在 mysql-data2.mytest.com 上，再向数据库插入一条数据：

```
mysql> INSERT INTO testtable () VALUES (2);
Query OK, 1 row affected (0.23 sec)
mysql> quit
Bye
```

数据插入后，我们回到 mysql-data1.mytest.com 上，看看数据是否同步过去了：

```
mysql> SELECT * FROM testtable;
+-------+
| Count |
+-------+
|     2 |
|     1 |
+-------+
2 rows in set (0.00 sec)

mysql> quit
Bye
```

很好！在 mysql-data1.mytest.com 上也能查到该数据。

也就是说，两个 NDB 节点，始终会保持互相同步，保持相同的数据。

2．故障模拟测试

假如两个 NDB 节点当中的一个发生故障，会怎么样呢？另一个节点上面的数据能够查询得到吗？现在我们来测试一下。

首先，我们把 mysql-data1.mytest.com 节点上的 ndb 服务停掉，来看看在另外一个节点 mysql-data2.mytest.com 上还能不能查到完整的数据。

在 mysql-data1.mytest.com 上执行命令：

```
$ sudo /etc/init.d/mysql-ndb stop
```

然后，到管理节点 mysql-mgm.mytest.com 上确认一下集群的连接状态：

```
$ ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2 (not connected, accepting connect from 192.168.1.13)
id=3    @192.168.1.14  (Version: 5.0.51, Nodegroup: 0, Master)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.1.10  (Version: 5.0.51)

[mysqld(API)]   2 node(s)
id=4    @192.168.1.13  (Version: 5.0.51)
id=5    @192.168.1.14  (Version: 5.0.51)
```

可以看到，192.168.1.13 这个 NDB 节点（mysql-data1.mytest.com）确实已经断开连接了。输入 quit 或者 exit 命令退出 MGM 界面：

```
ndb_mgm> quit
```

现在，我们在 mysql-data2.mytest.com 上看看是否还能查询数据：

```
$ mysql -u root -p
```

输入密码后，就进入了 MySQL 客户端的命令行界面。在该界面中，输入如下命令：

```
mysql> USE clustertest;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

mysql> SELECT * FROM testtable;
+--------+
| Count |
```

```
+-------+
|     2 |
|     1 |
+-------+
2 row in set (0.03 sec)
```

很好，在 mysql-data1.mytest.com 节点发生故障时，在 mysql-data2.mytest.com 节点上仍然可以工作。

我们再向 mysql-data2.mytest.com 的数据库中插入一条数据：

```
mysql> INSERT INTO testtable () VALUES (3);
Query OK, 1 row affected (0.89 sec)

mysql> SELECT * FROM testtable;
+-------+
| Count |
+-------+
|     1 |
|     2 |
|     3 |
+-------+
3 rows in set (0.00 sec)

mysql> quit
Bye
```

现在，我们重新启动 mysql-data1.mytest.com，看看刚才插入的数据能否查到：

```
$ sudo /etc/init.d/mysql-ndb start
$ mysql -u root -p
```

输入密码后，就进入了 MySQL 客户端的命令行界面。在该界面中，输入如下命令：

```
mysql> USE clustertest;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM testtable;
+-------+
| Count |
+-------+
|     2 |
|     3 |
|     1 |
+-------+
3 rows in set (0.00 sec)

mysql> quit
Bye
```

非常好！这说明，如果有 NDB 节点发生故障，故障期间在其他 NDB 节点上新插入的

数据，在故障节点恢复后，仍然可以成功地进行数据同步。

19.5 实现负载均衡

到目前为止，“集群”本身已经安装好了。不过，这个集群还没有一个统一的对外 IP 地址；所以，现在要使用该集群的话，你只能配置一部分程序使用 mysql-data1.mytest.com 作为数据库服务器，另外一部分程序则使用 mysql-data2.mytest.com。这样做的坏处是，两台数据库服务器的工作量很难做到“平衡”。而且，最大的问题是：万一某台数据库服务器发生故障，会导致所有使用它的程序停止工作。

解决该问题的方法，就是在数据库服务器的前面，放置“负载均衡”服务器。负载均衡服务器使用一个虚拟 IP 地址连接两台数据库服务器，所有应用程序都使用该虚拟 IP 地址作为数据库服务器地址。这样一来，即便某台数据库服务器宕机，也不会影响应用程序，只要有一台数据库服务器能正常工作，整个系统就不会停止运转。

不过，新的问题又来了。万一负载均衡服务器本身发生故障怎么办？那样岂不是整个系统就瘫痪了。是的，为了防止发生这种问题，我们需要配置两个负载均衡节点（它们分别是 mysql-lb1.mytest.com 和 mysql-lb2.mytest.com）；它们以“主/从”模式配合工作，平时只有“主服务器”在工作，“从服务器”在待命；一旦“主服务器”宕机，“从服务器”马上进入工作状态。

要实现负载均衡，所使用的主要软件有 heartbeat 和 ldirectord，下面我们来分别介绍。

19.5.1 ldirectord+heartbeat 介绍

1．ldirectord 的作用

在 Apache 服务器的前端，我们要放置一台服务器专门来做负载调度的任务（为了称呼简单和便于理解，我们将这样的负载调度服务器简称为“导演”），用来把访问需求分发给两台 Apache 服务器。这个“导演”的任务，正是由 ldirectord 来完成的。

“ldirectord”（Linux Director Daemon）可以对服务和物理服务器进行监测，被广泛地用于 http 和 https 等服务。它是专门为 LVS（Linux Virtual Server）监控而编写的，不仅能从 heartbeat 的配置文件/etc/ha.d/xxx.cf 中读取所有有关 IPVS（IP Virtul Server）路由表配置的信息，还可以方便地被 heartbeat 管理（比如由 heartbeat 来启动和停止 ldirectord 服务）。

2．heartbeat 是怎么工作的

因为我们要提供“高可用性”，所以要考虑到“导演”突然罢工的情形；因此我们要安排两个导演，也就是要有两个调度服务器节点。这两个节点的地位不同，其中一个是主节点，另外一个是辅节点（可以看成是一个“主导演”和一个“副导演”）。这两个节点正是用 heartbeat 来互相监测对方的。

heartbeat 可以通过以太网（或者串行接口）来监控节点的“健康”状况。如果有多个 heartbeat 节点（heartbeat 2.0 及后续版本已经能够支持两个以上节点），我们既可以使用串

行线又可以使用以太网连接它们，这样将大大提高系统的可用性。

heartbeat 的核心功能有两个部分：心跳监测和资源接管。通过心跳监测，节点之间相互“打招呼”（发送报文）来告诉对方自己当前的状态；如果在指定的时间内没“听”到对方“打招呼”（没收到报文），那么就认为对方罢工了，这时 heartbeat 会自动启动资源接管模块，运行相关的 shell 脚本来接管运行在对方主机上的资源或者服务。

19.5.2　让内核支持 IPVS

首先，我们必须让 mysql-lb1.mytest.com 和 mysql-lb2.mytest.com 两台服务器的内核支持 IPVS（IP Virtual Server）。通过 IPVS，我们可以在 Linux 内核级别上实现传输层的负载均衡。

在 mysql-lb1.mytest.com 和 mysql-lb2.mytest.com 上，执行下面的命令：

```
$ sudo modprobe ip_vs_dh
$ sudo modprobe ip_vs_ftp
$ sudo modprobe ip_vs
$ sudo modprobe ip_vs_lblc
$ sudo modprobe ip_vs_lblcr
$ sudo modprobe ip_vs_lc
$ sudo modprobe ip_vs_nq
$ sudo modprobe ip_vs_rr
$ sudo modprobe ip_vs_sed
$ sudo modprobe ip_vs_sh
$ sudo modprobe ip_vs_wlc
$ sudo modprobe ip_vs_wrr
```

然后，我们还需要修改/etc/modules，添加这些模块，以便系统在重新启动时能够自动加载它们。在 mysql-lb1.mytest.com 和 mysql-lb2.mytest.com 上，执行下面的命令：

```
$ sudo nano /etc/modules
```

在文件末尾，添加下列内容：

```
ip vs dh
ip_vs_ftp
ip_vs
ip vs lblc
ip_vs_lblcr
ip_vs_lc
ip_vs_nq
ip_vs_rr
ip_vs_sed
ip_vs_sh
ip_vs_wlc
ip_vs_wrr
```

此外，我们还需要启用内核的包转发功能。在 mysql-lb1.mytest.com 和 mysql-lb2.mytest.com 上，执行下面的命令：

```
$ sudo nano /etc/sysctl.conf
```

将 net.ipv4.ip_forward 所在行的注释去掉：

```
net.ipv4.ip_forward = 1
```

文件保存后，运行命令使之立即生效：

```
$ sudo sysctl -p
```

19.5.3 安装 heartbeat、ldirectord 等软件

下面我们来安装 heartbeat、ldirectord，以及要用到的其他软件。在 mysql-lb1.mytest.com 和 mysql-lb2.mytest.com 上，执行下面的命令：

```
$ sudo apt-get install heartbeat ldirectord
$ sudo apt-get install libdbi-perl libdbd-mysql-perl libmysqlclient15-dev
```

19.5.4 配置 heartbeat

heartbeat 是用来检测两台负载均衡服务器的心跳的。

首先，我们需要调整两个负载均衡节点上的/etc/hosts 文件，将主机名写进去：

```
$ sudo nano /etc/hosts
127.0.0.1       localhost
192.168.1.11        mysql-lb1.mytest.com    mysql-lb1
192.168.1.12        mysql-lb2.mytest.com    mysql-lb2
```

好，现在来配置 heartbeat。

我们需要为 heartbeat 创建 3 个配置文件（在两个节点 mysql-lb1.mytest.com 和 mysql-lb2.mytest.com 上的操作完全相同）。

第一个文件是/etc/ha.d/ha.cf。我们来创建它：

```
$ sudo nano /etc/ha.d/ha.cf
logfacility       local0
bcast       eth0
mcast eth0 225.0.0.1 694 1 0
auto_failback off
node       mysql-lb1
node       mysql-lb2
respawn hacluster /usr/lib/heartbeat/ipfail
apiauth ipfail gid=haclient uid=hacluster
```

> **警告** node 名称（这里的 mysql-lb1 和 mysql-lb2）必须和 uname -n 命令的输出结果一致。

第二个文件是/etc/ha.d/haresources，用来设置虚拟 IP 地址。我们来创建它：

```
$ suod nano /etc/ha.d/haresources
mysql-lb1       \
```

```
        ldirectord::ldirectord.cf \
        LVSSyncDaemonSwap::master \
        IPaddr2::192.168.1.15/24/eth0/192.168.1.255
```

上述配置中的 node 名称，既可以设置为 mysql-lb1，也可以设置为 mysql-lb2；但是两台服务器的设置要相同。

第三个文件是/etc/ha.d/authkeys，用于认证。我们来创建它：

```
$ sudo nano /etc/ha.d/authkeys
```

```
auth 3
3 md5 A46fsdgCH
```

其中 A46fsdgCH 是 heartbeat 用于在 mysql-lb1 和 mysql-lb2 之间认证的密码，请你设置为自己的字符串。

为了安全，我们要设置该文件的权限，使 root 以外的用户无法访问：

```
$ sudo chmod 600 /etc/ha.d/authkeys
```

19.5.5　配置 ldirectord

ldirectord 是用来实现两台数据库服务器的负载均衡的。

以下操作，在 mysql-lb1.mytest.com 和 mysql-lb2.mytest.com 上完全相同。

首先，我们来定义/etc/ha.d/ldirectord.cf：

```
$ sudo nano /etc/ha.d/ldirectord.cf
```

```
# Global Directives
checktimeout=10
checkinterval=2
autoreload=no
logfile="local0"
quiescent=yes
virtual = 192.168.1.15:3306
        service = mysql
        real = 192.168.1.13:3306 gate
        real = 192.168.1.14:3306 gate
        checktype = negotiate
        login = "ldirector"
        passwd = "ldirectorpassword"
        database = "ldirectordb"
        request = "SELECT * FROM connectioncheck"
        scheduler = wrr
```

在上面配置中，我们定义了虚拟 IP 地址为 192.168.1.15，端口为 3306（MySQL 的默认端口）；两台真实服务器的 IP 地址分别为 192.168.1.13 和 192.168.1.14，端口都是 3306。还定义了一个用户 ldirector，密码为 ldirectorpassword，一个数据库 ldirectordb，一个 SQL 语句。ldirectord 将用这些信息来检测两台数据库服务器是否正常。稍后，我们会创建该数据库和用户。

接下来，我们禁止 ldirectord 服务自动启动（我们用 heartbeat 来控制 ldirectord 服务）；

然后修改 heartbeat 服务的启动顺序，让其推后启动，以等待其他服务先行启动：

```
$ sudo update-rc.d -f ldirectord remove
$ sudo update-rc.d -f heartbeat remove
$ sudo update-rc.d heartbeat start 90 2 3 4 5 . stop 05 0 1 6 .
```

19.5.6 NDB 节点配置

1. 为 ldirector 创建数据库

现在，我们来创建 ldirectordb 数据库，并把该数据库的权限赋予 ldirector 用户。ldirector 将使用该数据库来检查两个数据库节点的运行状态。

在 mysql-data1.mytest.com 上，创建该数据库：

```
$ mysql -u root -p
```

在 MySQL 命令行界面中，输入如下命令：

```
mysql> GRANT ALL ON ldirectordb.* TO 'ldirector'@'%' IDENTIFIED BY 'ldirectorpassw
ord';
mysql> FLUSH PRIVILEGES;
mysql> CREATE DATABASE ldirectordb;
mysql> USE ldirectordb;
mysql> CREATE TABLE connectioncheck (Status INT) ENGINE=NDBCLUSTER;
mysql> INSERT INTO connectioncheck () VALUES (1);
mysql> quit
```

然后，在 mysql-data2.mytest.com 上创建该数据库，数据会自动从 mysql-data1 上复制过来：

```
$ mysql -u root -p
```

在 MySQL 命令行界面中，输入如下命令：

```
mysql> GRANT ALL ON ldirectordb.* TO 'ldirector'@'%' IDENTIFIED BY 'ldirectorpassw
ord';
mysql> FLUSH PRIVILEGES;
mysql> CREATE DATABASE ldirectordb;
mysql> quit
```

2. 设置 IP 路由

我们需要让两台数据库服务器 mysql-data1.mytest.com 和 mysql-data2.mytest.com 能够通过虚拟 IP 地址 192.168.1.15 连接起来。

在 mysql-data1 和 mysql-data2 上分别执行下列操作。

首先，安装 iproute 软件包：

```
$ sudo apt-get install iproute
```

修改/etc/sysctl.conf：

```
$ sudo nano /etc/sysctl.conf
```

加入如下内容：

```
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.eth0.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.eth0.arp_announce = 2
```

加入后，执行 sysctl -p 使之立即生效：

```
$ sudo sysctl -p
```

3．设置虚拟 IP 地址

在 mysql-data1 和 mysql-data2 上分别执行下列操作。

修改/etc/network/interfaces，添加虚拟 IP 地址：

```
$ sudo nano /etc/network/interfaces
```

在文件末尾添加如下内容：

```
auto lo:0
iface lo:0 inet static
  address 192.168.1.15
  netmask 255.255.255.255
  pre-up sysctl -p > /dev/null
```

然后启用它：

```
$ sudo ifup lo:0
```

19.5.7　测试

现在，我们来启动负载均衡服务。

在 mysql-lb1.mytest.com 和 mysql-lb2.mytest.com 上，执行下列命令：

```
$ sudo /etc/init.d/ldirectord stop
$ sudo /etc/init.d/heartbeat start
```

如果没有任何错误，则说明一切配置正确。

现在，重新启动 mysql-lb1 和 mysql-lb2：

```
$ sudo reboot
```

重启后，再做下面的测试。

1. ldirectord 状态检查

在 mysql-lb1 和 mysql-lb2 上，执行下面的命令：

```
$ ldirectord ldirectord.cf status
```

在 mysql-lb1 上，其输出应为：

```
ldirectord for /etc/ha.d/ldirectord.cf is running with pid: 4584
```

在 mysql-lb2 上，其输出应为：

```
ldirectord is stopped for /etc/ha.d/ldirectord.cf
```

2. 虚拟 IP 状态检查

在 mysql-lb1 和 mysql-lb2 上，执行下面的命令：

```
$ ip addr sh eth0
```

在 msyql-lb1（负载均衡“主服务器”）上，应该会列出虚拟 IP 地址 192.168.1.15。其输出结果应为：

```
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:16:3e:45:fc:f8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.11/24 brd 192.168.0.255 scope global eth0
    inet 192.168.1.15/24 brd 192.168.0.255 scope global secondary eth0
```

在 msyql-lb2（负载均衡“从服务器”）上，其输出结果应为：

```
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:16:3e:16:c1:4e brd ff:ff:ff:ff:ff:ff
inet 192.168.1.12/24 brd 192.168.0.255 scope global eth0
```

3. IPVS 状态检查

在 mysql-lb1 和 mysql-lb2 上，执行下面的命令：

```
$ sudo ipvsadm -L -n
```

在 mysql-lb1 上，其输出应为：

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.15:3306 wrr
  -> 192.168.1.13:3306           Route   1      0          0
  -> 192.168.1.14:3306           Route   1      0          0
```

在 mysql-lb2 上，其输出应为：

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
```

在 mysql-lb1 和 mysql-lb2 上，执行下面的命令：

```
$ sudo /etc/ha.d/resource.d/LVSSyncDaemonSwap master status
```

在 mysql-lb1 上，其输出应为：

```
master running
(ipvs_syncmaster pid: 4704)
```

在 mysql-lb2 上，其输出应为：

```
master stopped
(ipvs_syncbackup pid: 1440)
```

4．MySQL 测试

在 192.168.1.0 网络内的某台机器上，通过虚拟 IP 地址 192.168.1.15 连接我们的负载均衡服务器试试看：

```
$ mysql -h 192.168.1.15 -u ldirector -p
```

在正常情况下，你应该可以顺利连接到 MySQL。

5．故障模拟测试

现在，可以将负载均衡的“主服务器”停掉，看看“从服务器”能不能很快变成“主服务器”。

测试的方法有很多，我们使用简单的 ping 命令。在某台机器上，ping 负载均衡的虚拟 IP 地址 192.168.1.15，观察 ping 命令的输出结果；然后将 mysql-lb1 服务器上的 heartbeat 服务停掉，这时从 ping 命令的输出看，应该 ping 不到 192.168.1.15，几秒钟后，又能够重新 ping 到 192.168.1.15（说明 mysql-lb2 成功地接替 mysql-lb1，进入工作状态了）：

```
$ ping 192.168.1.15
[...]
64 bytes from 192.168.1.15: icmp_seq=22 ttl=64 time=0.416 ms
64 bytes from 192.168.1.15: icmp_seq=23 ttl=64 time=0.901 ms
64 bytes from 192.168.1.15: icmp_seq=24 ttl=64 time=217 ms
From 192.168.1.11: icmp_seq=25 Redirect Host(New nexthop: 192.168.1.15)
From 192.168.1.11 icmp_seq=26 Destination Host Unreachable
[...]
64 bytes from 192.168.1.15: icmp seq=50 ttl=64 time=1961 ms
64 bytes from 192.168.1.15: icmp_seq=51 ttl=64 time=975 ms
```

19.6　注意事项

19.6.1　数据库引擎问题

如果你要将现有的数据库转移到 MySQL 集群中，那么请将数据表的格式由 MyISAM 或 InnoDB 改为 NDBCLUSTER。转换的方法，请阅读 MySQL 的官方文档：

http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-multi-load-data-queries.html

否则，可能会遇到意想不到的问题，比如要添加 MySQL 用户，就必须在每个 NDB 节点上添加，等等。

19.6.2 内存问题

NDB 把所有数据存放在内存中，所以你需要为 NDB 节点配备较大的内存。

一般情况下，换算比例为 1:1.1，也就是说，如果你有 1GB 数据库，至少要配备 1.1GB 内存；如果你的数据库会增长到 8GB，就需要至少为服务器配备 8.8GB 内存。而且每个 NDB 节点都要做如此配置。

19.6.3 安全问题

该集群的 MGM 节点，其管理服务运行在 1186 端口上；NDB 节点的 MySQL 运行在 3306 端口上。

建议你使用防火墙将各个相关端口的访问权限做一下限制。比如 NDB 节点的 3306 端口，只允许 192.168.1.15 访问，等等。