MYSQL 高可用方案探究

文件状态 [ ] 草稿 [√] 正式发布 [ ] 正在修改	名称	mysql 高可用方案探究
	作者	飞鸿无痕
	版本	V1.0
	日期	2012-09-04
	博客地址	http://blog.chinaunix.net/uid/20639775.html

1 前言

Mysql 高可用一直是 mysql 业界不断讨论的热点问题，其中涉及的东西比较多，可供选择的方案也相当多，面对这么多的方案，我们应该如何选择适合自己公司的 mysql 高可用方案呢，我觉得首先我们需要了解的自己公司的业务，了解在线系统中那些东西会影响高可用，以及了解各个高可用方案比较适合哪些场景，通过这些比对应该不难找出适合自己公司的高可用 mysql 方案。

经常有网友问 mysql 高可用如何实现，希望得到一些能实际使用的可验证的高可用方案。所以花了些时间对 mysql 高可用的几种常用方式做一下总结，及写出详细的配置方案，方便网友学习以及验证，希望对大家学习 mysql 高可用有所帮助。这也是本文档的目的所在

由于本人经验和水平有限，有不对之处烦请指出，多交流，互相帮助，共同进步。下面的几种高可用方案在 Centos 5.5 64bit /mysql 5.1.63 环境测试通过。

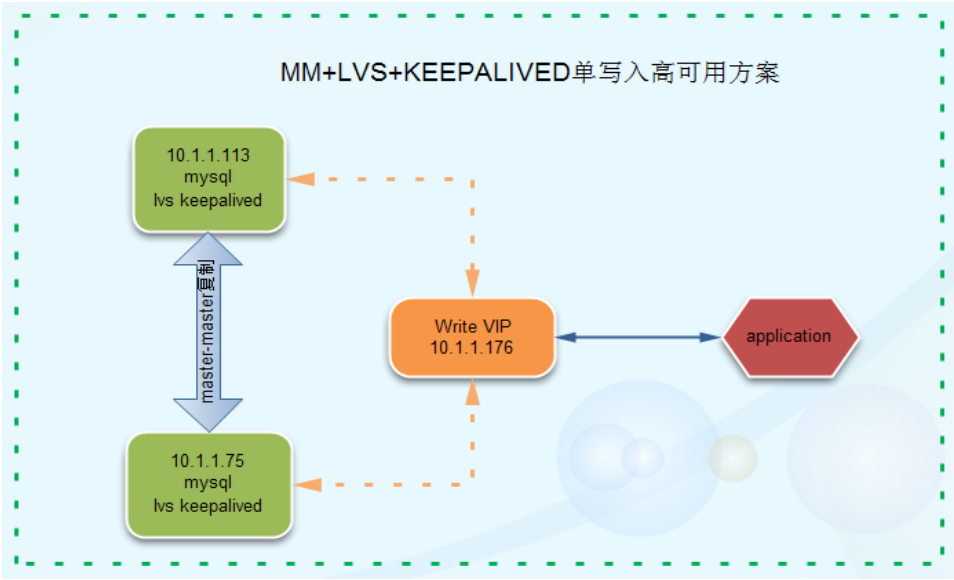
目前 Mysql 的高可用主要有 Lvs+Keepalived、Heartbeat、MMM、mysql cluster 三种方式，由于时间关系这里不对 mysql cluster 做介绍，有兴趣的可以访问 http://blog.chinaunix.net/uid-20639775-id-201960.html，下面就逐一地来详细介绍其他几种高可用方案。

2 Lvs+Keepalived+Mysql 单点写入主主同步高可用方案

2.1 方案简介

Lvs+keepalived 作为目前比较流行的高可用解决方案，lvs 提供负载均衡，keepalived 作为故障转移，提高系统的可用性。但是一般的 mysql 高可用为了实现 mysql 数据的一致性，一般都是采用单点写入，本方案采用 keepalived 中的 sorry_server 来实现写入数据库为单点的需求。本方案实现的功能是当网络有问题、mysql 有问题、服务器宕机、keepalived 服务停止后,服务器能自动跳转到备用机，当主服务器服务启动起来后会自动切换回来。

2.2 方案架构图



2.3 方案优缺点

优点：

- ✓ 安装配置简单，实现方便，高可用效率好，可以根据服务与系统的可用性多方面进行切换。
- ✓ 可以将写 VIP 和读 VIP 分别进行设置，为读写分离做准备。
- ✓ 扩展不是很方便。
- ✓ 可以在后面添加多个从服务器，并做到负载均衡。

缺点：

- ✓ 在启动或者恢复后会立即替换掉定义的 sorry_server，因此如果要实现指定条件替换或者不替换需要通过其他方式实现，比如：临时更改 mysql 的端口等。
- ✓ 切换需要 1s 左右的时间。

2.4 方案实战

2.4.1 适用场景

这个方案适用于只有两台数据库服务器并且还没有实现数据库的读写分离的情况，读和写都配置 VIP。这个方案能够便于单台数据库的管理维护以及切换工作。比如进行大表的表结构更改、数据库的升级等都是非常方便的。

2.4.2 实战环境介绍

服务器名	IP	VIP	系统	Mysql
Master	10.1.1.113	10.1.1.176	Centos 5.5 64bit	5.1.63
Backup	10.1.1.75	10.1.1.176	Centos 5.5 64bit	5.1.63

2.4.3 Mysql 的安装和配置

Mysql 的安装和配置相对来讲非常简单，这里就不做介绍，有兴趣的朋友可以查看我博客中关于 mysql 5.1.63 版本自动安装的文章 http://blog.chinaunix.net/uid-20639775-id-3168737.html

2.4.4 Mysql 的主主同步配置

Mysql 的主主同步这里也不做介绍了，有兴趣的话可以看一下我博文中关于 mysql 主从同步管理的介绍，主主同步和主从同步差不多，只是互为主从而已，链接如下：
http://blog.chinaunix.net/uid-20639775-id-3254611.html

2.4.5 Lvs 的安装

在 master、backup 服务器都进行安装：

```
wget http://www.linuxvirtualserver.org/software/kernel-2.6/ipvsadm-1.24.tar.gz
ln -s /usr/src/kernels/2.6.18-164.el5-i686/ /usr/src/linux
```

```
tar  zxvf  ipvsadm-1.24.tar.gz
cd  ipvsadm-1.24
make  &&  make  install
```

2.4.6 Keepalived 的安装

在 master、backup 服务器都进行安装：

```
wget  http://www.keepalived.org/software/keepalived-1.1.19.tar.gz
tar  zxvf  keepalived-1.1.19.tar.gz
cd  keepalived-1.1.19
./configure  --prefix=/usr/local/keepalived
make
make  install
cp  /usr/local/keepalived/sbin/keepalived  /usr/sbin/
cp  /usr/local/keepalived/etc/sysconfig/keepalived  /etc/sysconfig/
cp  /usr/local/keepalived/etc/rc.d/init.d/keepalived  /etc/init.d/
mkdir  /etc/keepalived
```

2.4.7 Keepalived 的配置

2.4.7.1 Master 的 keepalived 的配置

Master 和 backup 不一样的地方已经标记为红色

vim /etc/keepalived/keepalived.conf

```
global_defs {

notification_email {
                        zhangxy@test.com
            }
            notification_email_from jiankong@test.com
            smtp_server mail.test.com
            smtp_connect_timeout 30
            router_id LVS1
}

vrrp_sync_group test {
group {
            loadbalance
}
}

vrrp_instance loadbalance {
            state MASTER
            interface eth0
            lvs_sync_daemon_inteface eth0
            virtual_router_id 51
```

```
        priority 180
        advert_int 1

authentication {
        auth_type PASS
        auth_pass 1111
}

virtual_ipaddress {
        10.1.1.176 dev eth0 label eth0:1
}
}

virtual_server 10.1.1.176 3306 {
        delay_loop 6
        lb_algo rr
        lb_kind DR
        persistence_timeout 20
        protocol TCP
        sorry_server 10.1.1.75 3306
        real_server 10.1.1.113 3306 {
        weight 3
        TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
        connect_port 3306
}
}
}
```

2.4.7.2 Backup 的 keepalived 的配置

Master 和 backup 不一样的地方已经标记为红色

vim /etc/keepalived/keepalived.conf

```
global_defs {

notification_email {
                zhangxy@test.com
        }
        notification_email_from jiankong@test.com
        smtp_server mail.test.com
        smtp_connect_timeout 30
        router_id LVS1
}
```

```
vrrp_sync_group test {
group {
        loadbalance
}
}

vrrp_instance loadbalance {
        state BACKUP
        interface eth0
        lvs_sync_daemon_inteface eth0
        virtual_router_id 51
        priority 150
        advert_int 1

authentication {
        auth_type PASS
        auth_pass 1111
}

virtual_ipaddress {
        10.1.1.176 dev eth0 label eth0:1
}
}

virtual_server 10.1.1.176 3306 {
        delay_loop 6
        lb_algo rr
        lb_kind DR
        persistence_timeout 20
        protocol TCP
        sorry_server 10.1.1.75 3306
        real_server 10.1.1.113 3306 {
        weight 3
        TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
        connect_port 3306
}
}
}
```

2.4.8 Master 和 backup 的 realserver 的配置

对于 realserver 的配置 master 和 backup 是一致的，脚本内容如下：

vim /etc/rc.d/init.d/realserver.sh

```
#!/bin/bash
# description: Config realserver lo and apply noarp

SNS_VIP=10.1.1.176
/etc/rc.d/init.d/functions
case "$1" in

start)
        ifconfig lo:0 $SNS_VIP netmask 255.255.255.255 broadcast $SNS_VIP
        /sbin/route add -host $SNS_VIP dev lo:0
        echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
        echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
        echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
        echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
        sysctl -p >/dev/null 2>&1
        echo "RealServer Start OK"
        ;;

stop)
        ifconfig lo:0 down
        route del $SNS_VIP >/dev/null 2>&1
        echo "0" >/proc/sys/net/ipv4/conf/lo/arp_ignore
        echo "0" >/proc/sys/net/ipv4/conf/lo/arp_announce
        echo "0" >/proc/sys/net/ipv4/conf/all/arp_ignore
        echo "0" >/proc/sys/net/ipv4/conf/all/arp_announce
        echo "RealServer Stoped"
        ;;

*)

        echo "Usage: $0 {start|stop}"
        exit 1
esac
exit 0
```

2.4.9 Master 和 backup 的启动

启动 master 和 backup 的 mysql 以后，再在 master 和 backup 执行如下命令启动 keepalived 和 realserver 脚本：

```
/etc/rc.d/init.d/realserver.sh start
/etc/rc.d/init.d/keepalived start
```

并将 keepalived 和 realserver 的启动脚本加入到 rc.local 自启动中：

```
echo “/etc/rc.d/init.d/realserver.sh start” >> /etc/rc.local
```

```
echo “/etc/rc.d/init.d/keepalived start” >> /etc/rc.local
```

2.4.10 高可用方案测试

方案搭建好以后就要进行全方位的可靠性测试了，看看是否达到了我们的预期效果，大致测试步骤如下：

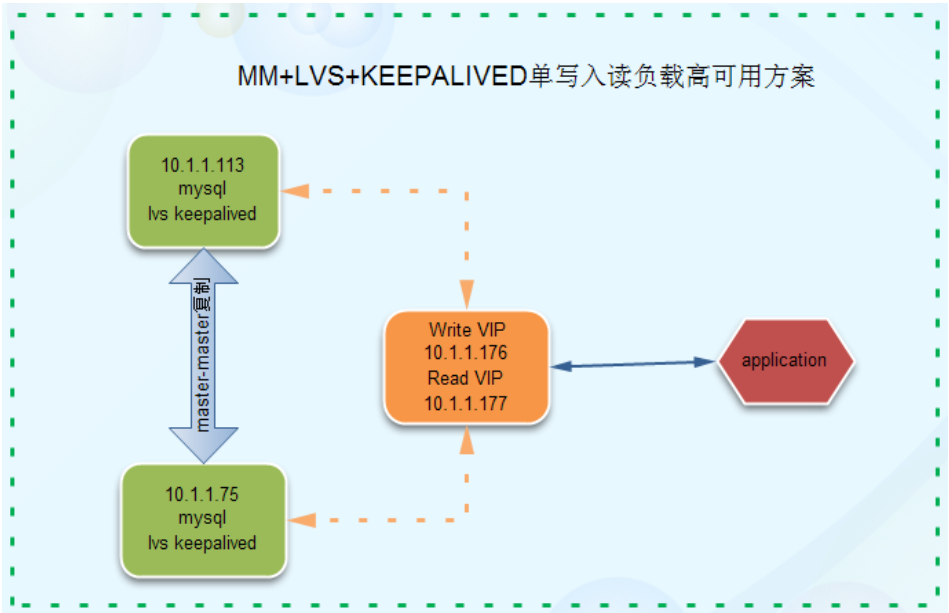
- ✓ 停掉 master 上的 mysql，看看能否自动切换到 sorry_server，使用如下命令查看：ipvsadm –ln。
- ✓ 停掉 master 上的 keepalived，看写 VIP 是否会迁移到 backup。
- ✓ 启动 master 上的 mysql，看是否能切换回 master。
- ✓ 启动 master 上的 keepalived，看 VIP 是否会迁移回 master 上。
- ✓ 重启 master 的系统，看看切换过程是否 OK

3 Lvs+Keepalived+Mysql 单点写入读负载均衡主主同步高可用方案

3.1 方案简介

Lvs+keepalived 作为目前比较流行的高可用解决方案，lvs 提供负载均衡，keepalived 作为故障转移，提高系统的可用性。但是一般的 mysql 高可用为了实现 mysql 数据的一致性，一般都是采用单点写入，本方案采用 keepalived 中的 sorry_server 来实现写入数据库为单点的需求，读负载均衡通过 lvs 实现，读能自由的实现负载均衡和故障切换。本方案实现的功能是当网络有问题、mysql 有问题、服务器宕机、keepalived 服务停止后,服务器能自动跳转到备用机，当主服务器服务启动起来后会自动切换回来。

3.2 方案架构图



3.3 方案优缺点

优点：

- ✓ 实现方便，高可用效率好，可以根据服务与系统的可用性多方面进行切换。

- ✓ 可以将写 VIP 和读 VIP 分别进行设置，为读写分离做准备。
- ✓ 扩展很方便。可以在后面添加多个从服务器，并做到负载均衡。

缺点：

- ✓ 在启动或者恢复后会立即替换掉定义的 sorry_server，因此如果要实现指定条件替换或者不替换需要通过其他方式实现，比如：临时更改 mysql 的端口等。
- ✓ 安装配置比单写入稍微复杂，需要另外一个 VIP。管理比单写入复杂。
- ✓ 主切换后从需要手工切换。
- ✓ 切换需要 1s 左右的时间。

3.4 适用场景

这个方案适用于只有两台数据库服务器（后端有多个从服务器也是可以的，只是要手工切换从服务器比较麻烦，后面会介绍的 MMM 能将从服务器自动切换）并且还能实现数据库的读写分离的情况，这样 backup 机器也能用起来，提高系统资源的利用率，减少 master 端的负载。应用中读数据库配置读 VIP，写数据库配置写 VIP。这个方案也能够很方便的进行单台数据库的管理维护以及切换工作。比如进行大表的表结构更改、数据库的升级等都是非常方便的。

3.5 方案实战

3.5.1 实战环境介绍

服务器名	IP	VIP	系统	Mysql
Master	10.1.1.113	10.1.1.176 10.1.1.177	Centos 5.5 64bit	5.1.63
Backup	10.1.1.75	10.1.1.176 10.1.1.177	Centos 5.5 64bit	5.1.63

3.5.2 Mysql 的安装和配置

Mysql 的安装和配置相对来讲非常简单，这里就不做介绍，有兴趣的朋友可以查看我博客中关于 mysql 5.1.63 版本自动安装的文章 http://blog.chinaunix.net/uid-20639775-id-3168737.html

3.5.3 Mysql 的主主同步配置

Mysql 的主主同步这里也不做介绍了，有兴趣的话可以看一下我博文中关于 mysql 主从同步管理的介绍，主主同步和主从同步差不多，只是互为主从而已，链接如下：http://blog.chinaunix.net/uid-20639775-id-3254611.html

3.5.4 Lvs 的安装

在 master、backup 服务器都进行安装：

```
wget http://www.linuxvirtualserver.org/software/kernel-2.6/ipvsadm-1.24.tar.gz
ln -s /usr/src/kernels/2.6.18-164.el5-i686/ /usr/src/linux
```

```
tar zxvf ipvsadm-1.24.tar.gz
cd ipvsadm-1.24
make && make install
```

3.5.5 Keepalived 的安装

在 master、backup 服务器都进行安装：

```
wget http://www.keepalived.org/software/keepalived-1.1.19.tar.gz
tar zxvf keepalived-1.1.19.tar.gz
cd keepalived-1.1.19
./configure --prefix=/usr/local/keepalived
make
make install
cp /usr/local/keepalived/sbin/keepalived /usr/sbin/
cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/sysconfig/
cp /usr/local/keepalived/etc/rc.d/init.d/keepalived /etc/init.d/
mkdir /etc/keepalived
```

3.5.6 Keepalived 的配置

3.5.6.1 Master 的 keepalived 的配置

Master 和 backup 不一样的地方已经标记为红色

vim /etc/keepalived/keepalived.conf

```
global_defs {

notification_email {
                zhangxy@test.com
        }
        notification_email_from jiankong@test.com
        smtp_server mail.test.com
        smtp_connect_timeout 30
        router_id LVS1
}

vrrp_sync_group test {
group {
        loadbalance
}
}

vrrp_instance loadbalance {
        state MASTER
        interface eth0
        lvs_sync_daemon_inteface eth0
        virtual_router_id 51
```

```
        priority 180
        advert_int 1

authentication {
        auth_type PASS
        auth_pass 1111
}

virtual_ipaddress {
        10.1.1.176 dev eth0 label eth0:1
        10.1.1.177 dev eth0 label eth0:2
}
}

virtual_server 10.1.1.176 3306 {
        delay_loop 6
        lb_algo rr
        lb_kind DR
        persistence_timeout 20
        protocol TCP
        sorry_server 10.1.1.75 3306
        real_server 10.1.1.113 3306 {
        weight 3
        TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
        connect_port 3306
}
}
}

virtual_server 10.1.1.177 3306 {
        delay_loop 6
        lb_algo rr
        lb_kind DR
        #persistence_timeout 20
        protocol TCP
        real_server 10.1.1.113 3306 {
        weight 3
        TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
```

```
        connect_port 3306
}
}
        real_server 10.1.1.75 3306 {
        weight 3
        TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
        connect_port 3306
}
}
}
```

3.5.6.2 Backup 的 keepalived 的配置

Master 和 backup 不一样的地方已经标记为红色

vim /etc/keepalived/keepalived.conf

```
global_defs {

notification_email {
                zhangxy@test.com
        }
        notification_email_from jiankong@test.com
        smtp_server mail.test.com
        smtp_connect_timeout 30
        router_id LVS1
}

vrrp_sync_group test {
group {
        loadbalance
}
}

vrrp_instance loadbalance {
        state BACKUP
        interface eth0
        lvs_sync_daemon_inteface eth0
        virtual_router_id 51
        priority 150
        advert_int 1

authentication {
        auth_type PASS
```

```
        auth_pass 1111
}

virtual_ipaddress {
        10.1.1.176 dev eth0 label eth0:1
        10.1.1.177 dev eth0 label eth0:2
}
}

virtual_server 10.1.1.176 3306 {
        delay_loop 6
        lb_algo rr
        lb_kind DR
        persistence_timeout 20
        protocol TCP
        sorry_server 10.1.1.75 3306
        real_server 10.1.1.113 3306 {
        weight 3
        TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
        connect_port 3306
}
}
}

virtual_server 10.1.1.177 3306 {
        delay_loop 6
        lb_algo rr
        lb_kind DR
        persistence_timeout 20
        protocol TCP
        real_server 10.1.1.113 3306 {
        weight 3
        TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
        connect_port 3306
}
}
        real_server 10.1.1.75 3306 {
        weight 3
```

```
        TCP_CHECK {
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
        connect_port 3306
}
}
}
```

3.5.7 Master 和 backup 的 realserver 的配置

对于 realserver 的配置 master 和 backup 是一致的，脚本内容如下：

vim /etc/rc.d/init.d/realserver.sh

```
#!/bin/bash
# description: Config realserver lo and apply noarp

SNS_VIP=10.1.1.176
SNS_VIP2=10.1.1.177
/etc/rc.d/init.d/functions
case "$1" in

start)
        ifconfig lo:0 $SNS_VIP netmask 255.255.255.255 broadcast $SNS_VIP
         ifconfig lo:1 $SNS_VIP2 netmask 255.255.255.255 broadcast $SNS_VIP2
        /sbin/route add -host $SNS_VIP dev lo:0
         /sbin/route add -host $SNS_VIP2 dev lo:1
        echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
        echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
        echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
        echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
        sysctl -p >/dev/null 2>&1
        echo "RealServer Start OK"
        ;;

stop)
        ifconfig lo:0 down
        ifconfig lo:1 down
        route del $SNS_VIP >/dev/null 2>&1
        route del $SNS_VIP2 >/dev/null 2>&1
        echo "0" >/proc/sys/net/ipv4/conf/lo/arp_ignore
        echo "0" >/proc/sys/net/ipv4/conf/lo/arp_announce
        echo "0" >/proc/sys/net/ipv4/conf/all/arp_ignore
        echo "0" >/proc/sys/net/ipv4/conf/all/arp_announce
        echo "RealServer Stoped"
        ;;
```

```
*)

        echo "Usage: $0 {start|stop}"
        exit 1
esac
exit 0
```

3.5.8 Master 和 backup 的启动

启动 master 和 backup 的 mysql 以后，再在 master 和 backup 执行如下命令启动 keepalived 和 realserver 脚本：

```
/etc/rc.d/init.d/realserver.sh start
/etc/rc.d/init.d/keepalived start
```

并将 keepalived 和 realserver 的启动脚本加入到 rc.local 自启动中：

```
echo “/etc/rc.d/init.d/realserver.sh start” >> /etc/rc.local
echo “/etc/rc.d/init.d/keepalived start” >> /etc/rc.local
```

3.5.9 高可用方案测试

方案搭建好以后就要进行全方位的可靠性测试了，看看是否达到了我们的预期效果，大致测试步骤如下：

- ✓ 停掉 master 上的 mysql，看看能写 IP 否自动切换到 sorry_server，看看读 IP 是否去掉了 master 的 mysql，使用如下命令查看：ipvsadm –ln。
- ✓ 停掉 master 上的 keepalived，看读写 VIP 是否会迁移到 backup 上。
- ✓ 启动 master 上的 mysql，看是否能切换回 master。
- ✓ 启动 master 上的 keepalived，看 VIP 是否会迁移回 master 上。
- ✓ 重启 master 的系统，看看切换过程是否 OK

4 Heartbeat 高可用 Mysql 主主同步方案

4.1 方案简介

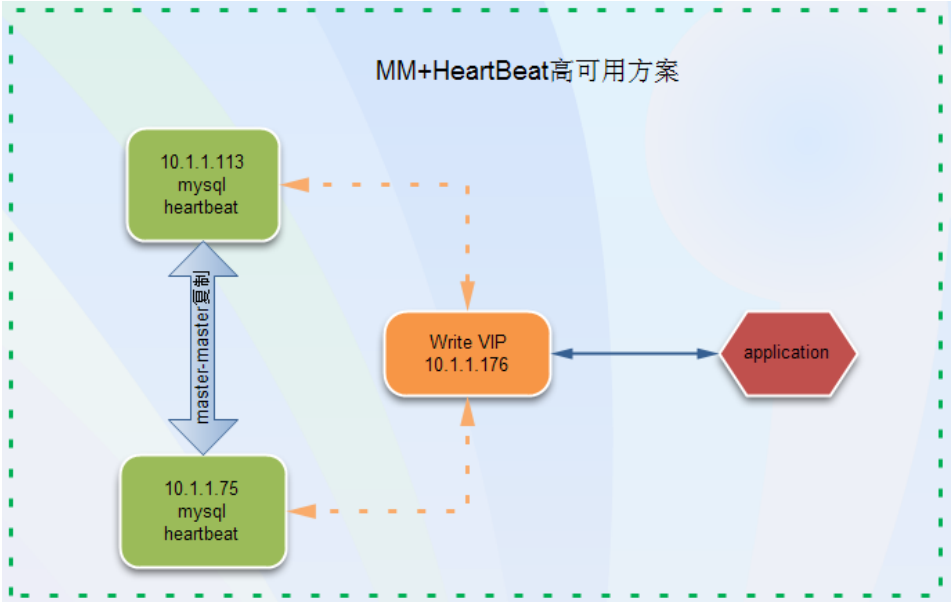
本方案使用 heartbeat+mysql 主主同步来实现 mysql 数据库的高可用， 当服务器或者 master 的 heartbeat 宕掉以后会自动切换到 backup 上，服务器或者 master 的 heartbeat 恢复以后可以自动切换回来，master 继续提供服务。

4.2 方案优缺点

- ➢ 优点：
  配置简单、，可配置主机恢复后是否切换回 master。不存在单点故障。
- ➢ 缺点：
  当 mysql 服务挂掉或者不可用的情况下不能进行自动切换，需要通过 crm 模式实现或者额外的脚本实现(比如 shell 脚本监测到 master 的 mysql 不可用就将主上的 heartbeat 停掉，这样就会切换到 backup 中去)。
  默认启动以及切换后的 backup 话 mysql 不会启动起来，因此这对于 mysql 复制是很不利的。因此需要做好监控，发生切换以后需要手动去启动。或者

mysql 之间不使用复制，而是用共享存储或者 DRBD，这样能解决这个问题。
不方便扩展。
可能会发生脑裂问题。

4.3 方案架构图



4.4 适用场景

该方案适合只有两台数据库的情况，访问量不大，不需要实现读写分离的情况。

4.5 方案实战

4.5.1 实战环境介绍

服务器名	IP	VIP	系统	Mysql
Master	10.1.1.113	10.1.1.176	Centos 5.5 64bit	5.1.63
Backup	10.1.1.75	10.1.1.176	Centos 5.5 64bit	5.1.63

4.5.2 Mysql 的安装和配置

Mysql 的安装和配置相对来讲非常简单，这里就不做介绍，有兴趣的朋友可以查看我博客中关于 mysql 5.1.63 版本自动安装的文章 http://blog.chinaunix.net/uid-20639775-id-3168737.html

4.5.3 Mysql 的主主同步配置

Mysql 的主主同步这里也不做介绍了，有兴趣的话可以看一下我博文中关于 mysql 主从同步管理的介绍，主主同步和主从同步差不多，只是互为主从而已，链接如下：http://blog.chinaunix.net/uid-20639775-id-3254611.html

4.5.4 Heardbeat 的安装

Master 和 backup 服务器都需要安装 heardheat 软件。下面两种安装方式任

选其一。

- Rpm 包的安装方式

```
yum -y install heartbeat-*
```

- 源代码编译安装方式

```
wget  http://www.ultramonkey.org/download/heartbeat/2.1.3/heartbeat-2.1.3.tar.gz
tar  xzvf  heartbeat-2.1.3.tar.gz
cd  heartbeat-2.1.3
./configure
 Make
 make  install
```

4.5.5 Heartbeat 的配置

Hearbeat 的配置主要包括三个配置文件，authkeys，ha.cf 和 haresources 的配置，下面就分别来看！

- Hosts 文件的配置

  需要在 hosts 文件中添加 master 和 backup 主机，加快节点间的通信 Master 和 backup 的 hosts 节点添加的内容一样，我的配置添加如下内容：

  vim /etc/hosts

```
#dbserver 和 puppet 是我的 master 和 backup 的主机名
10.1.1.113  dbserver
10.1.1.75  puppet
```

- Authkerys 的配置

  这个文件用来配置密码认证方式,支持 3 种认证方式,crc,md5 和 sha1，从左到右安全性越来越高，消耗的资源也越多。因此如果 heartbeat 运行在安全的网路之上，比如私网，那么可以将验证方式设置成 crc，master 和 backup 的 authkeys 配置一样。我的 authkeys 文件配置如下：

  vim /etc/ha.d/authkeys

```
auth 1
1 crc
```

- ha.cf 的配置

  master 的 ha.cf 的配置

  vim /etc/ha.d/ha.cf

```
logfile /var/log/ha-log
logfacility     local0
keepalive 2
deadtime 30
warntime 10
initdead 60
udpport 694
ucast eth0 10.1.1.75
auto_failback on
```

```
node    dbserver
node    puppet
ping 10.1.1.1
respawn hacluster /usr/lib64/heartbeat/ipfail
```

backup 的 ha.cf 的配置

vim /etc/ha.d/ha.cf

```
logfile /var/log/ha-log
logfacility     local0
keepalive 2
deadtime 30
warntime 10
initdead 60
udpport 694
ucast eth0 10.1.1.113
auto_failback on
node    dbserver
node    puppet
ping 10.1.1.1
respawn hacluster /usr/lib64/heartbeat/ipfail
```

- haresources 的配置

  haresources 用来设置 master 的主机名、虚拟 IP、服务以及磁盘挂载等，master 和 backup 的配置是一样的，下面的 mysqld 需要做成服务，放在/etc/rc.d/init.d/目录下，配置配置如下：

  vim /etc/ha.d/haresources

```
dbserver  10.1.1.176/32/eth0 mysqld
```

4.5.6 Heartbeat 的启动

在启动 master 和 backup 上的 mysqld 启动以后，启动 master 和 backup 的 keepalived：

```
/etc/rc.d/init.d/heartbeat start
```

并将此启动语句加入到 master 和 backup 的/etc/rc.local 中去

4.5.7 方案测试

环境搭建好以后，就需要进行周密的测试，看是否实现了预期的功能：

- 停掉 master 上的 mysqld，看看是否切换(因为 heartheat 不检查服务的可用性，因此需要你通过而外的脚本来实现，方法前面已经描述)。
- 停掉 master 的 heartheat 看看是否能正常切换。
- 停掉 master 的网络或者直接将 master 系统 shutdown，看看能否正常切换。
- 启动 master 的 heartbeat 看看是否能正常切换回来。
- 重新启动 master 看看能否切换过程是否 OK。

4.5.8 监控方案

因为 heartbeat 不监控资源的可用性以及切换后会将资源停止，所以需要加强对资源和 heartbeat 的监控，推荐采用 nagios 软件来进行可用性的监控。

5 Heartbeat+DRBD+mysql 高可用方案

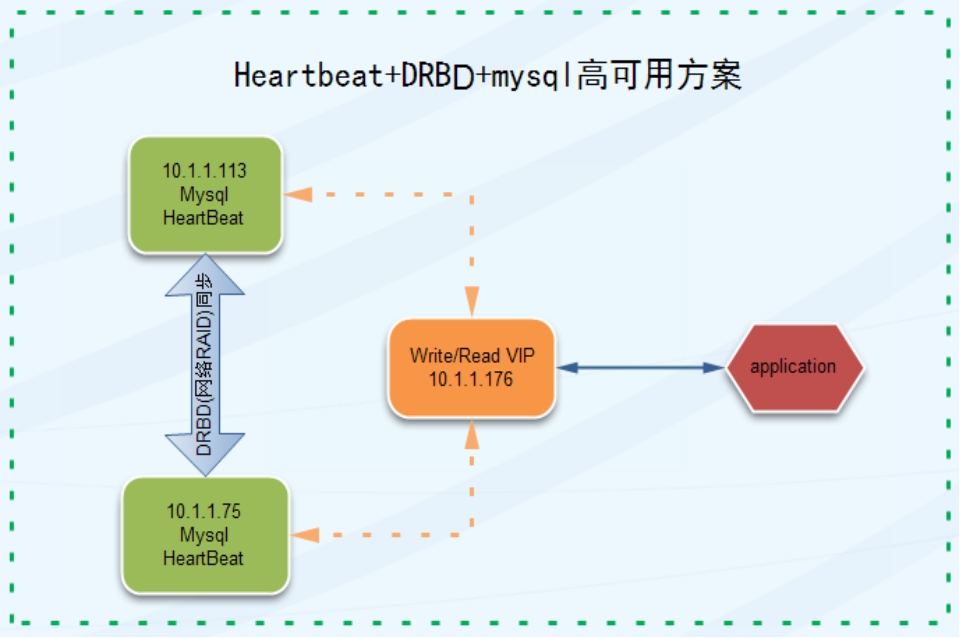
5.1 方案简介

本方案采用 Heartbeat 双机热备软件来保证数据库的高稳定性和连续性，数据的一致性由 DRBD 这个工具来保证。默认情况下只有一台 mysql 在工作，当主 mysql 服务器出现问题后，系统将自动切换到备机上继续提供服务，当主数据库修复完毕，又将服务切回继续由主 mysql 提供服务。

5.2 方案优缺点

优点：安全性高、稳定性高、可用性高，出现故障自动切换，
缺点：只有一台服务器提供服务，成本相对较高。不方便扩展。可能会发生脑裂。

5.3 方案架构图



5.4 方案适用场景

本方案适用于数据库访问量不太大，短期内访问量增长不会太快，对数据库可用性要求非常高的场景。

5.5 方案实战

5.5.1 实战环境介绍

主机名	Ip	系统	DRBD 磁盘	Heartbeat 版本
dbserver1	10.1.1.113	Centos 5.5 64bit	/dev/sdb1	2.1.3
dbserver2	10.1.1.108	Centos 5.5 64bit	/dev/sdb1	2.1.3

5.5.2 DRBD 的安装

官网的说法，如果系统内核（linux）版本低于 2.6.33，在安装软件之前需要加载 DRBD 模块，我的内核版本是 2.6.18 的，安装后会自动加载 drbd 模块。安装命令如下：

```
yum install -y drbd83 kmod-drbd83
```

安装后使用 lsmod | grep drbd 命令查看是否加载 drbd 模块，如果没有加载需要手动运行命令加载，命令如下：

```
insmod drbd/drbd.ko 或者 modprobe drbd
```

5.5.3 DRBD 的配置

配置之前需要先使用 fdisk 对/dev/sdb 进行分区。

对于 DRBD 的配置，只需要配置/etc/drbd.conf 和 hosts 文件即可，dbserver1 和 dbserver2 的 hosts 添加的内容如下：

```
10.1.1.113 dbserver1
10.1.1.108 dbserver2
```

我的/etc/drbd.conf 文件的内容如下(dbserver1 和 dbserver2 的配置一样)：

```
global { usage-count yes; }
common { syncer { rate 10M; } }
resource r0 {
        protocol C;
        startup {
        }
        disk {
                on-io-error detach;
                #size 1G;
        }
        net {
        }
        on dbserver1 {
                device /dev/drbd0;
                disk /dev/sdb1;
                address 10.1.1.113:7888;
                meta-disk internal;
        }
        on dbserver2 {
                device /dev/drbd0;
                disk /dev/sdb1;
                address 10.1.1.108:7888;
                meta-disk internal;
        }
}
```

5.5.4 DRBD 的管理维护

- 创建 DRBD 资源

  配置好 drbd 以后，就需要使用命令在 dbserver1 和 dbserver2 上创建配置

的 drbd 资源，使用如下命令：

```
drbdadm create-md r0     # r0 为配置文件中定义的资源名
```

- DRBD 的启动和停止

```
/etc/rc.d/init.d/drbd start       #启动 drbd
/etc/rc.d/init.d/drbd stop        #停止 drbd
/etc/rc.d/init.d/drbd restart     #重启 drbd
```

- 查看 DRBD 状态

```
watch -n 1 cat /proc/drbd
/etc/init.d/drbd status
```

以上两种方式都可以查看 drbd 的状态

- 设置当前节点为主节点，并进行格式化和挂载

```
drbdadm -- --overwrite-data-of-peer primary all
mkfs.ext3 /dev/drbd0
mkdir /drbd
mount /dev/drbd0 /drbd
```

挂在后可以往/drbd 目录写入一些测试数据，看看是否能同步到从节点上。

- 迁移 mysql 的数据到 drdb(假设你的 mysql 已经在该服务器上安装好了)

DRBD 已经安装并且能正常同步了，那么我们接下来要做的就是将本机的已安装的 mysql 的数据迁移到 drbd 上，这里为了简单就不考虑新安装数据库的情形了，因为新安装的时候只需要将数据目录指定到 drbd 中并将 my.cnf 配置文件放到 drbd 中即可。具体的数据迁移步骤如下：

a) 关闭 dbserver1 和 dbserver2 的 mysql，/etc/rc.d/init.d/mysqld stop
b) 在 dbserver1 上创建存放数据库数据的目录：mkdir /drbd/dbdata
c) 在 dbserver1 将配置文件放到 drbd 目录中：
   mv /etc/my.cnf /drbd/dbdata/
   删除 dbserver2 上的/etc/my.cnf，rm -f /etc/my.cnf
   在 dbserver1 和 dbserver2 上执行如下命令创建软链接。
   ln -s /drbd/dbdata/my.cnf /etc/my.cnf
d) 修改/etc/my.cnf 的数据目录指向/drbd/dbdata
e) 将原来的 mysql 数据文件移动到/drbd/dbdata
f) 执行 chown -R mysql.mysql /drbd/dbdata
g) 启动 mysql

- 手工切换 DRBD

在没有安装配置 drbd 之前，drbd 是不能自动切换的，我们可以写出过程来加深对 drbd 的理解，也更能明白 heartbeat 的工作流程，下面是手工切换的步骤：

a) 在 dbserver1 上停止 mysql，/etc/rc.d/init.d/mysqld stop。
b) 在 dbserver1 上执行 umount /dev/drbd0。
c) 在 dbserver1 上执行 drbdadm secondary all 切换到从模式。当两个节点都是 secondary 模式的时候才可以将 dbserver2 设置成 primary。
d) 在 dbserver2 上执行 drbdadm primary all，将 dbserver2 提升为主模式，并观察/proc/drbd 是否正常。
e) 在 dbserver2 上执行 mount /dev/drbd0 /drbd 挂在文件系统。

f) 启动 dbserver2 的 mysql，/etc/rc.d/init.d/mysqld start。

注意：dbserver1 和 dbserver2 上的 mysql 用户的 uid 和 gid 要一样。不然切换后会导致 mysql 数据目录的属主不正确而启动失败。

- 主从切换

主切换成从，需要先卸载文件系统，再执行降级为从的命令：

```
umount /dev/drbd0
drbdadm secondary all
```

从切换成主，要先执行升级成主的命令然后挂在文件系统：

```
drbdadm  primary all  如果不成功 drbdsetup /dev/drbd0 primary -o
mount /dev/drbd0 /drbd/
```

- DRBD 脑裂后的处理

当 DRBD 出现脑裂后，会导致 drbd 两边的磁盘不一致，处理方法如下：

在确定要作为从的节点上切换成 secondary，并放弃该资源的数据：

```
drbdadm secondary r0
drbdadm -- --discard-my-data connect r0
```

在要作为 primary 的节点重新连接 secondary（如果这个节点当前的连接状态为 WFConnection 的话，可以省略），使用如下命令连接：

```
drbdadm connect r0
```

5.5.5 Heartbeat 的安装

在 DRBD 调试没有问题之后，就可以开始安装和配置 heartbeat 了，Master 和 backup 服务器都需要安装 heardheat 软件。下面两种安装方式任选其一。

- Rpm 包的安装方式

```
yum -y install heartbeat-*
```

- 源代码编译安装方式

```
wget  http://www.ultramonkey.org/download/heartbeat/2.1.3/heartbeat-2.1.3.tar.gz
tar  xzvf  heartbeat-2.1.3.tar.gz
cd  heartbeat-2.1.3
./configure
 Make
 make  install
```

5.5.6 Heartbeat 的配置

Hearbeat 的配置主要包括三个配置文件，authkeys，ha.cf 和 haresources 的配置，下面就分别来看！

- Hosts 文件的配置

需要在 hosts 文件中添加 master 和 backup 主机，加快节点间的通信

Master 和 backup 的 hosts 节点添加的内容一样，我的配置添加如下内容：

vim /etc/hosts

```
# dbserver1 和 dbserver2 是我的 master 和 backup 的主机名
```

```
10.1.1.113 dbserver1
10.1.1.108 dbserver2
```

- Authkerys 的配置

  这个文件用来配置密码认证方式,支持3种认证方式,crc,md5和sha1，从左到右安全性越来越高，消耗的资源也越多。因此如果 heartbeat 运行在安全的网路之上，比如私网，那么可以将验证方式设置成 crc，master 和 backup 的 authkeys 配置一样。我的 authkeys 文件配置如下：

  vim /etc/ha.d/authkeys

```
auth 1
1 crc
```

- ha.cf 的配置

  master(dbserver1)的 ha.cf 的配置

  vim /etc/ha.d/ha.cf

```
logfile /var/log/ha-log
logfacility     local0
keepalive 2
deadtime 30
warntime 10
initdead 60
udpport 694
ucast eth0 10.1.1.108
auto_failback on
node    dbserver1
node    dbserver2
ping 10.1.1.1
respawn hacluster /usr/lib64/heartbeat/ipfail
```

  backup(dbserver2)的 ha.cf 的配置

  vim /etc/ha.d/ha.cf

```
logfile /var/log/ha-log
logfacility     local0
keepalive 2
deadtime 30
warntime 10
initdead 60
udpport 694
ucast eth0 10.1.1.113
auto_failback on
node    dbserver1
node    dbserver2
ping 10.1.1.1
respawn hacluster /usr/lib64/heartbeat/ipfail
```

- haresources 的配置

  haresources 用来设置 master 的主机名、虚拟 IP、服务以及磁盘挂载等，master 和 backup 的配置是一样的，下面的 mysqld 需要做成服务，放在

/etc/rc.d/init.d/目录下，配置配置如下：

vim /etc/ha.d/haresources

```
dbserver1       IPaddr::10.1.1.176/24/eth0:1       drbddisk::r0
Filesystem::/dev/drbd0::/drbd::ext3 mysqld
```

5.5.7 Heartbeat 的管理

配置好 heartbeat 之后，需要将 mysql 从自启动服务器中去掉，因为主 heartbeat 启动的时候会挂载 drdb 文件系统以及启动 mysql，切换的时候会将主上的 mysql 停止并卸载文件系统，从上会挂载文件系统，并启动 mysql。因此需要做如下操作：

```
chkconfig mysqld off
chkconfig --add heartbeat
chkconfig heartbeat on
```

或者将 heartbeat 的启动命令你写到/etc/rc.local 启动文件中去。

5.5.8 Heartbeat+DRBD 测试

环境搭建好以后，就需要进行周密的测试，看是否实现了预期的功能：

- 停掉 master 上的 mysqld，看看是否切换(因为 heartheat 不检查服务的可用性，因此需要你通过而外的脚本来实现，方法前面已经描述)。
- 停掉 master 的 heartheat 看看是否能正常切换。
- 停掉 master 的网络或者直接将 master 系统 shutdown，看看能否正常切换。
- 启动 master 的 heartbeat 看看是否能正常切换回来。
- 重新启动 master 看看能否切换过程是否 OK。

注意：这里说的切换是不是已经将 mysql 停掉、是否卸载了文件系统等等。

5.5.9 Heartbeat+DRBD 监控

由于 heartbeat 不能监控 mysql 的可用性，因此需要通过其他的方式来实现，对于 mysql 可用性的监控是必须的，如果发生切换，需要第一时间知道是什么原因导致的切换，使用 nagios 能很好的监控那些网络、mysql、系统等的可用性。

6 MMM 高可用 mysql 方案

6.1 方案简介

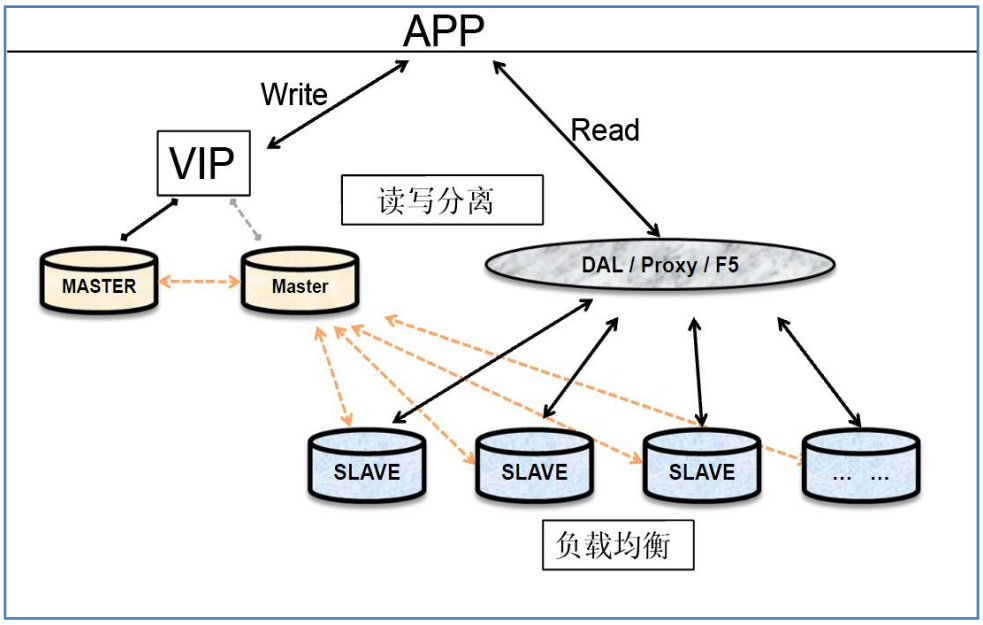
MMM 即 Master-Master Replication Manager for MySQL（mysql 主主复制管理器）关于 mysql 主主复制配置的监控、故障转移和管理的一套可伸缩的脚本套件（在任何时候只有一个节点可以被写入），这个套件也能对居于标准的主从配置的任意数量的从服务器进行读负载均衡，所以你可以用它来在一组居于复制的服务器启动虚拟 ip，除此之外，它还有实现数据备份、节点之间重新同步功能的脚本。MySQL 本身没有提供 replication failover 的解决方案，通过 MMM 方案能实现服务器的故障转移，从而实现 mysql 的高可用。MMM 不仅能提供浮动 IP 的功能，更可贵的是如果当前的主服务器挂掉后，会将你后端的从服务器自动转向新的主服务器进行同步复制，不用手工更改同步配置。这个方案是目前比较成熟的解决方案。

6.2 方案优缺点

优点：安全性、稳定性高，可扩展性好，高可用，当主服务器挂掉以后，另一个主立即接管，其他的从服务器能自动切换，不用人工干预。

缺点：至少三个节点，对主机的数量有要求，需要实现读写分离，对程序来说是个挑战。

6.3 方案架构图



6.4 适用场景

MMM 的适用场景为数据库访问量大，业务增长快，并且能实现读写分离的场景。

6.5 方案实战

6.5.1 实战环境介绍

实战环境服务器列表：

服务器	主机名	ip 地址	Serverid	系统	Mysql
Monitor	Mon	10.1.1.111	--	Centos 5.5 64bit	--
Master1	db1	10.1.1.113	1	Centos 5.5 64bit	5.1.63
Master2	db2	10.1.1.75	2	Centos 5.5 64bit	5.1.63
Slave1	db3	10.1.1.108	3	Centos 5.5 64bit	5.1.63

实战环境虚拟 IP 列表：

VIP	Role	description
10.1.1.176	Read	应用配置的读取 IP，也可以在前端加 lvs 等，做负载均衡 。三台数据库每台一个浮动 VIP
10.1.1.177	Read	
10.1.1.178	Read	
10.1.1.179	Write	应用配置的写入的 VIP，单点写入。

6.5.2 MMM 的安装

在安装 mmm 之前要安装 epel 包，因为 Centos 的默认源中没有 mmm 的安装包，Epel 是企业版 Linux 附加软件包(Extra Packages for Enterprise Linux)的缩写，是一个由特别兴趣小组创建、维护并管理的，针对红帽企业版 Linux(RHEL)及其衍生发行版(比如 CentOS、Scientific Linux)的一个高质量附加软件包项目。

安装 epel：

```
wget ftp://ftp.sunet.se/pub/Linux/distributions/yellowdog/yum/6.2/extras/RPMS/epel-release-5-3.noarch.rpm
rpm -Uvh epel-release-5-3.noarch.rpm
```

安装 monitor 的安装：

```
yum -y install mysql-mmm*
```

各个 DB 上只需要安装 mysql-mmm-agent

```
yum -y install mysql-mmm-agent
```

6.5.3 MMM 的配置

6.5.3.1 配置之前的准备

前提是要配置好 master1 和 master2 的主主同步，master1 和 slave1 的主从同步，限于篇幅这里我就不做介绍了。

在配置 mmm 之前首先要在 mysql 中创建除复制帐号之外的另外两个帐号，首先来介绍 monitor user 帐号，这个帐号是 monitor 服务器用来对 mysql 服务器做健康检查的，其次就是 agent user，这个帐号是 mmm agent(mmm 代理)用来变成只读模式和同步 master 等，下面是创建这两个帐号的语句：

```
GRANT REPLICATION CLIENT ON *.* TO 'mmm_monitor'@'10.1.1.%' IDENTIFIED BY 'monitor_password';
GRANT SUPER, REPLICATION CLIENT, PROCESS ON *.* TO 'mmm_agent'@'10.1.1.%' IDENTIFIED BY 'agent_password';
flush privileges;
```

这两个语句在每个 mysql 中都要执行一下。

6.5.3.2 Monitor 服务器的配置

MMM 的配置文件在/etc/mysql-mmm 目录下，monitor 需要配置的文件有 mmm_common.conf、mmm_mon.conf 两个文件。mmm_common.conf 文件在 mmm 的各个节点都是一样的，因此配置好以后 copy 到各个 DB 节点即可。

我 mmm_common.conf 的配置如下：

vim /etc/mysql-mmm/mmm_common.conf

```
active_master_role      writer

<host default>
    cluster_interface       eth0
```

```
    pid_path                    /var/run/mysql-mmm/mmm_agentd.pid
    bin_path                    /usr/libexec/mysql-mmm/
    replication_user            replication
    replication_password        123456
    agent_user                  mmm_agent
    agent_password              agent_password
</host>

<host db1>
    ip          10.1.1.113      #这个 IP 尤其注意是 db1 的 IP
    mode        master
    peer        db2
</host>

<host db2>
    ip          10.1.1.75       #这个 IP 尤其注意是 db2 的 IP
    mode        master
    peer        db1
</host>

<host db3>
    ip          10.1.1.108
    mode        slave
</host>

<role writer>
    hosts       db1, db2
    ips         10.1.1.179
    mode        exclusive
</role>

<role reader>
    hosts       db1, db2, db3
    ips         10.1.1.176, 10.1.1.177, 10.1.1.178
    mode        balanced
</role>
```

我的 mmm_mon.conf 配置如下：

vim /etc/mysql-mmm/mmm_mon.conf

```
include mmm_common.conf

<monitor>
    ip                  127.0.0.1
    pid_path            /var/run/mysql-mmm/mmm_mond.pid
    bin_path            /usr/libexec/mysql-mmm
```

```
    status_path             /var/lib/mysql-mmm/mmm_mond.status
    ping_ips                10.1.1.1, 10.1.1.113, 10.1.1.75, 10.1.1.108
  # ping_ips 监控了网关 IP 和其他的 DB 节点 IP
    auto_set_online         60
</monitor>

<host default>
    monitor_user            mmm_monitor
    monitor_password        monitor_password
</host>

debug 0
```

6.5.3.3 各个 DB 服务器的配置

各个 DB 服务器要配置的东西比较不多，主要有 mmm_common.conf、mmm_agent.conf 和/etc/default/mysql-mmm-agent 文件。

- db1 的配置

  mmm_common.conf 文件的配置和前面 monitor 的一样，直接 copy 过来即可使用

  mmm_agent.conf 的配置：

  vim /etc/mysql-mmm/mmm_agent.conf

  ```
  include mmm_common.conf
  this db1
  ```

  mysql-mmm-agent 的配置：

  vim /etc/default/mysql-mmm-agent

  ```
  ENABLED=1
  ```

- db2 的配置

  mmm_common.conf 文件的配置和前面 monitor 的一样，直接 copy 过来即可使用

  mmm_agent.conf 的配置：

  vim /etc/mysql-mmm/mmm_agent.conf

  ```
  include mmm_common.conf
  this db2
  ```

  mysql-mmm-agent 的配置：

  vim /etc/default/mysql-mmm-agent

  ```
  ENABLED=1
  ```

- db3 的配置

  mmm_common.conf 文件的配置和前面 monitor 的一样，直接 copy 过来即可使用

  mmm_agent.conf 的配置：

  vim /etc/mysql-mmm/mmm_agent.conf

  ```
  include mmm_common.conf
  ```

```
this db3
```

mysql-mmm-agent 的配置：

vim /etc/default/mysql-mmm-agent

```
ENABLED=1
```

6.5.4 MMM 的管理

6.5.4.1 MMM 的启动和停止

- MMM 的启动

  启动 mmm agent

  ```
  /etc/init.d/mysql-mmm-agent start
  ```

  将 agent 的启动命令写入到三个 DB 的 rc.local 文件中

  启动 mmm monitor

  ```
  /etc/init.d/mysql-mmm-monitor start
  ```

  将 monitor 的启动命令写入到 monitor 服务器的 rc.local 文件中

- MMM 的停止

  停止 mmm agent

  ```
  /etc/init.d/mysql-mmm-agent stop
  ```

  停止 mmm monitor

  ```
  /etc/init.d/mysql-mmm-monitor stop
  ```

6.5.4.2 MMM 的基本管理

查看集群的状态

```
mmm_control show
```

将 db1 设置成 online 状态

```
mmm_control set_online db1
```

更多管理命令请使用 mmm_control help 查看，或者参考：

http://blog.chinaunix.net/uid-20639775-id-154606.html

6.5.5 MMM 架构的测试

环境搭建好以后，就需要进行周密的测试，看是否实现了预期的功能：

- 停掉 master1 后在 monitor 端使用 mmm_control show 看是否能切换。看看 slave1 是否能正确切换同步。
- 启动 master1 后在 monitor 端使用 mmm_control show 看是否能切换。
- 停掉 master2 后看看能否正确切换。

6.5.6 MMM 架构的监控

为实现高可用，系统的各个层面都需要全面的监控起来，比如 agent 进程的监控，monitor 进程的监控，mysql 可用性的监控，数据库同步的监控等，推荐使用 nagios 对以上资源进行监控，第一时间发现问题，第一时间处理。

7 参考文献

http://www.xifenfei.com/1465.html
http://blog.chinaunix.net/uid-20639775-id-154605.html
http://blog.chinaunix.net/uid-20639775-id-154604.html
http://www.eit.name/blog/read.php?472
http://blog.chinaunix.net/uid-20639775-id-3284747.html
http://blog.chinaunix.net/uid-20639775-id-3088302.html
http://blog.chinaunix.net/uid-20639775-id-154518.html
《高性能 MySQL(第 2 版)》