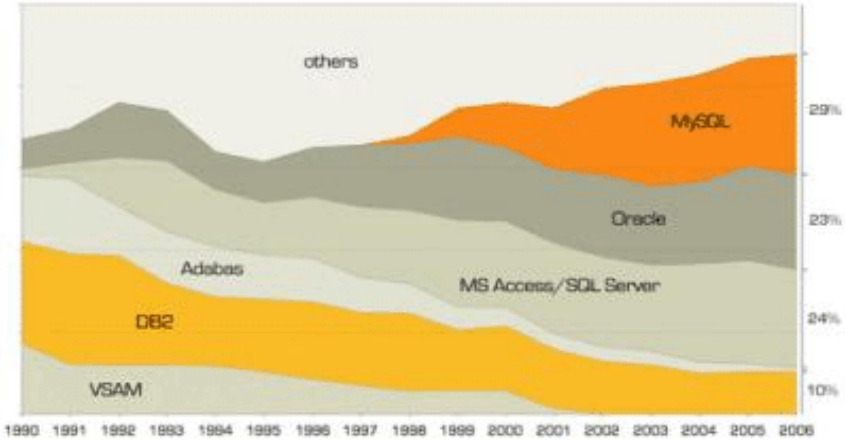
火龙果•整理
uml.org.cn

高可用性、负载均衡的 mysql 集群解决方案

- **mysql 的市场占有率**

  MySQL 是世界上最流行的开源数据库，已有1100多万的击活安装，每天超过五万的下载。MySQL 为全球开发者、DBA 和 IT 管理者在可靠性、性能、易用性方面提供了选择。

  第三方市场调查机构 Evans Data Corporation 调查显示，过去两年内在开发者使用的所有数据库中，MySQL 已经拥有了25%的市场占有率。开源已经成为当今 IT 结构中不可或缺的重要部分，而且开源的市场占有率将继续增加。如下图所示：



- **mysql 为什么受到如此的欢迎**

Sun 公司今天1月份花了10亿美元将 mysql 收购，准备进军开源和数据库。

数据库系统	Oracle	SQL Server	MySQL	DB2
是否免费	收费	收费	免费	收费
存储过程	支持	支持	支持	支持
视图	支持	支持	支持	支持
快照	支持	支持	不支持	支持
触发器	支持	支持	支持	支持
安全	强	中	中	强
复杂查询	强	中	弱	中

火龙果•整理
uml.org.cn

索引	丰富	一般	弱	中
数据类型	多	多	多	多
事务处理	强	强	弱	中

- **mysql 数据库系统的优缺点**

每个系统都有自身的不足和发展历程，mysql 也一样。

优点

1. 源码公开，免费
2. 跨平台
3. 为多种开发语言和包提供了 API
4. 支持多线程
5. 小巧、灵活、速度较快
6. 支持各种字符集
7. 提供各种连接、优化的工具包

缺点

1. 不完善，很多数据库特性不支持
2. 只适合中小型应用，对于大型应用，可以跟其他数据库互补；
3. 数据库系统数据量只能达到千万级别；

- **网络服务的需求**

随着 Internet 的飞速发展和对我们生活的深入影响，越来越多的个人在互联网上购物、娱乐、休闲、与人沟通、获取信息；越来越多的企业把他们与顾客和业务伙伴之间的联络搬到互联网上，通过网络来完成交易，建立与客户之间的联系。互联网的用户数和网络流量正以几何级数增长，这对网络服务的可伸缩性提出很高的要求。例如，比较热门的 Web 站点会因为被访问次数急剧增长而不能及时处理用户的请求，导致用户进行长时间的等待，大大降低了服务质量。另外，随着电子商务等关键性应用在网上运行，任何例外的服务中断都将造成不可估量的损失，服务的高可用性也越来越重要。所以，对用硬件和软件方法实现高可伸缩、高可用网络服务的需求不断增长，这种需求可以归结以下几点：

1) 可伸缩性（Scalability），当服务的负载增长时，系统能被扩展来满足需求，且不降低服务质量。
2) 高可用性（Availability），尽管部分硬件和软件会发生故障，整个系统的服务必须是每天24小时每星期7天可用的。
3) 可管理性（Manageability），整个系统可能在物理上很大，但应该容易管理。
4) 价格有效性（Cost-effectiveness），整个系统实现是经济的、易支付的。

单服务器显然不能处理不断增长的负载。这种服务器升级方法有下列不足：一是升级过程繁琐，机器切换会使服务暂时中断，并造成原有计算资源的浪费；二是越往高端的服务器，所花费的代价越大；三是一旦该服务器或应用软件失效，会导致整个服务的中断。

通过高性能网络或局域网互联的服务器集群正成为实现高可伸缩的、高可用网络服务的有效结构。这种松耦合结构比紧耦合的多处理器系统具有更好的伸缩性和性能价格比，组成

火龙果•整理
uml.org.cn

集群的PC服务器或RISC服务器和标准网络设备因为大规模生产，价格低，具有很高的性能价格比。但是，这里有很多挑战性的工作，如何在集群系统实现并行网络服务，它对外是透明的，它具有良好的可伸缩性和可用性。

针对上述需求，我们给出了基于IP层和基于内容请求分发的负载平衡调度解决方法，并在Linux内核中实现了这些方法，将一组服务器构成一个实现可伸缩的、高可用网络服务的服务器集群，我们称之为Linux虚拟服务器（Linux Virtual Server）。在LVS集群中，使得服务器集群的结构对客户是透明的，客户访问集群提供的网络服务就像访问一台高性能、高可用的服务器一样。客户程序不受服务器集群的影响不需作任何修改。系统的伸缩性通过在服务机群中透明地加入和删除一个节点来达到，通过检测节点或服务进程故障和正确地重置系统达到高可用性。

- **什么是mysql集群**

分为同步集群和异步集群。

同步集群（mysql cluster）
结构：**（data + sql + mgm节点）**
特点：
1) 内存级别的，对硬件要求较低，但是对内存要求较大。换算比例为：1：1.1；
2) 数据同时放在几台服务器上，冗余较好；
3) 速度一般；
4) 建表需要声明为engine=ndbcluster
5) 扩展性强；
6) 可以实现高可用性和负载均衡，实现对大型应用的支持；
7) 必须是特定的mysql版本，如：已经编译好的max版本；
8) 配置和管理方便，不会丢失数据；

异步集群（mysql replication）
结构：**（master + slave）**
特点：
1) 主从数据库异步数据；
2) 数据放在几台服务器上，冗余一般；
3) 速度较快；
4) 扩展性差；
5) 无法实现高可用性和负载均衡(只能在程序级别实现读写分离，减轻对主数据库的压力)；
6) 配置和管理较差，可能会丢失数据；

- **什么是负载均衡**

通过director，将用户的请求分发到real server服务器上，然后返回给用户。
负载均衡部署灵活、能够满足各种需求。

实现方式：

火龙果•整理
uml.org.cn

硬件：BIG/IP、Cisco、IBM（昂贵）

软件：LVS（免费）

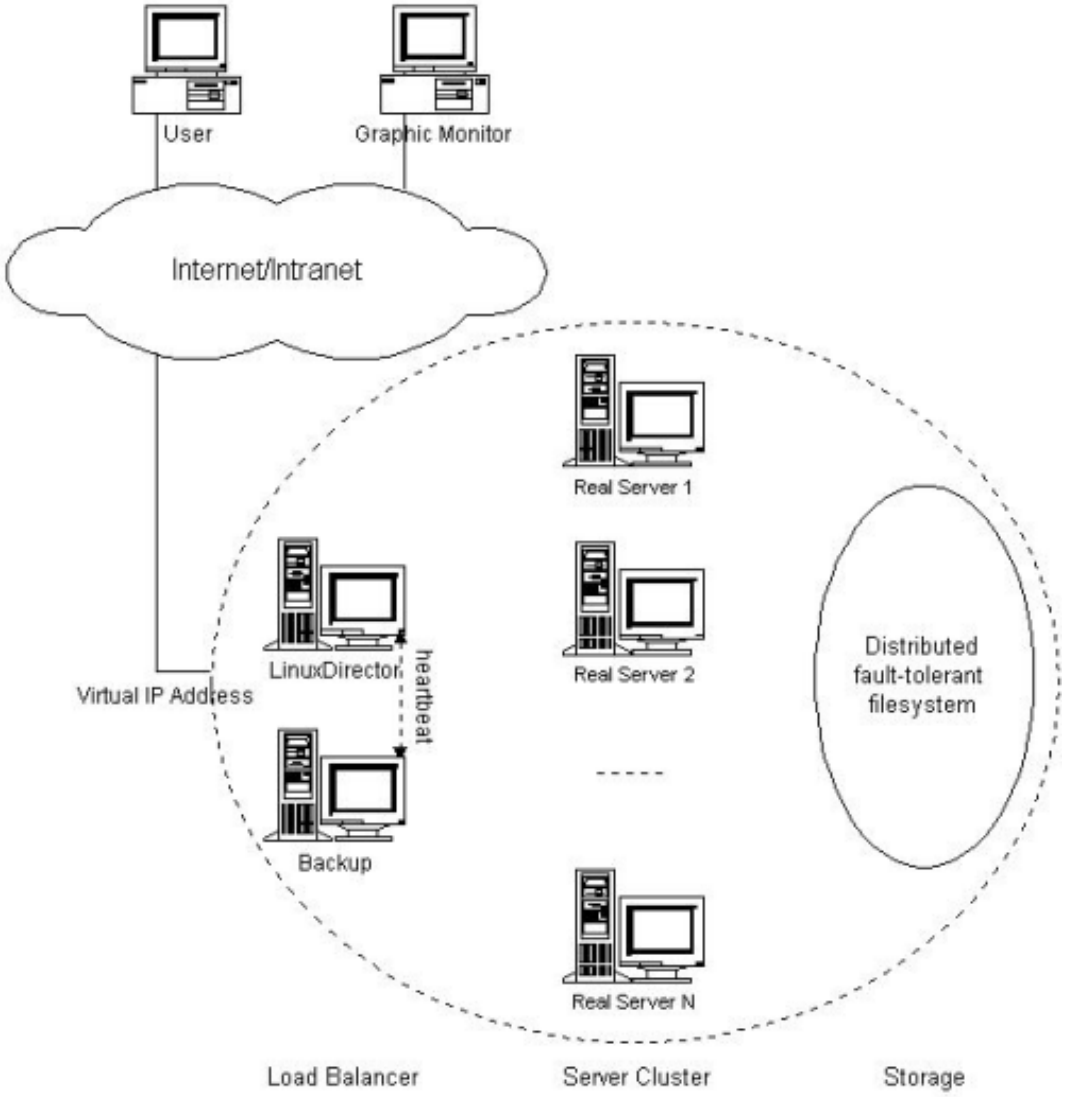
LVS 系统将用户的请求的数据包在数据层和网络层进行了封装和转发，由三种方式满足各种需求。

1) DR：直接路由
2) Tuning：tcp/ip 隧道
3) NAT：网络地址转换

需求：

免费的软件包
1) 2台低端的 director（active 和 standby）
2) 心跳线：连接2台 director，检测活动情况
3) 2台以上的 real servers

通用结构：



有兴趣的可以分别研究上面的三种 LVS 结构。

火龙果•整理
uml.org.cn

- **mysql 集群部署和实现方法**

1) 假设现在有4台服务器（mysql 官方推荐的最小配置）

服务器	开启的服务	角色
192.168.131.16	Mysqld	Mysql API
4	Ndb_mgmd	管理节点(master)
Ndb1	Heartbeat	Director(master)
192.168.131.26	Mysqld	Mysql API
Ndb2	Ndb_mgmd	管理节点(backup)
	Heartbeat	Director(standby)
192.168.131.77	Mysqld	Mysql API（realserver）
Sql1	Ndbd	存储节点
	Arptables	访问路由
192.168.131.10	Mysqld	Mysql API（realserver）
1	Ndbd	存储节点
Sql2	Arptables	访问路由

2）服务器安装配置和网络连接

(以下为所有服务器各操作一遍，共4遍)

安装：

将4台服务器安装 CentOS 5.2，选择下面的包：

Clustering

Storage Clustering

mysql 不需要安装，但 perl-mysql-xxx 的所有包需要安装

开发工具包和类库

sshd 服务

SElinux ==>disable

语言支持包不安装，默认美国英语

设定主机名：

Vi /etc/sysconfig/network

Hostname=xxx

:wq

检查主机名：

Uname -a

必须和上表中的一一对应。否则有问题。

Vi /etc/hosts

火龙果•整理
uml.org.cn

Ndb1 192.168.131.164
Ndb2 192.168.131.26
Sql1 192.168.131.77
Sql2 192.168.131.101

更新：
#rpm --import http://dries.ulyssis.org/rpm/RPM-GPG-KEY.dries.txt
#yum update -y && yum -y install lynx libawt xorg-x11-deprecated-libs nx freenx arptables_jf httpd-devel

下载：
Mysql cluster 版本（我下载的5.0.67社区版本）：
[root@ndb1 RHEL5]# ls -lh MySQL* | awk '{print $9}'
MySQL-client-community-5.0.67-0.rhel5.i386.rpm
MySQL-clusterextra-community-5.0.67-0.rhel5.i386.rpm
MySQL-clustermanagement-community-5.0.67-0.rhel5.i386.rpm
MySQL-clusterstorage-community-5.0.67-0.rhel5.i386.rpm
MySQL-clustertools-community-5.0.67-0.rhel5.i386.rpm
MySQL-devel-community-5.0.67-0.rhel5.i386.rpm
MySQL-server-community-5.0.67-0.rhel5.i386.rpm
MySQL-shared-community-5.0.67-0.rhel5.i386.rpm
MySQL-shared-compat-5.0.67-0.rhel4.i386.rpm
MySQL-shared-compat-5.0.67-0.rhel5.i386.rpm
MySQL-test-community-5.0.67-0.rhel5.i386.rpm
perl-HTML-Template-2.9-1.el5.rf.noarch.rpm
[root@ndb1 RHEL5]#

在服务器上安装以上包，在安装的过程中如果缺少包或者库，采用：
yum install xxxx 自行安装。

建立目录：
#mkdir /var/lib/mysql-cluster -p

以下分别操作：
安装 cluster 组件：
#Rpm -Uvh MySQL-xx-xx.rpm，根据不同，可以少安装部分组件。根据你需要而定。

163、26上，我安装了：
[root@ndb1 RHEL5]# rpm -aq | grep MySQL
MySQL-clusterstorage-community-5.0.67-0.rhel5
MySQL-clustertools-community-5.0.67-0.rhel5
MySQL-clustermanagement-community-5.0.67-0.rhel5

火龙果•整理
uml.org.cn

MySQL-shared-community-5.0.67-0.rhel5
perl-DBD-MySQL-3.0007-1.fc6
MySQL-server-community-5.0.67-0.rhel5
[root@ndb1 RHEL5]#

101、77上，我安装了：
[root@sql1 ~]# rpm -aq | grep MySQL
MySQL-clusterstorage-community-5.0.67-0.rhel4
MySQL-devel-community-5.0.67-0.rhel4
MySQL-server-community-5.0.67-0.rhel4
MySQL-client-community-5.0.67-0.rhel4
MySQL-shared-community-5.0.67-0.rhel4
[root@sql1 ~]#

以下在 ndb1（164）和 ndb2（26）上操作
```
[root@ndb1 ~]# vi /var/lib/mysql-cluster/config.ini
[NDBD DEFAULT]
NoOfReplicas=2
DataMemory=800M
IndexMemory=400M

[MYSQLD DEFAULT]

[NDB_MGMD DEFAULT]

[TCP DEFAULT]

# Section for the cluster management node
[NDB_MGMD]
# IP address of the management node (this system)
ID=1
HostName=192.168.131.164


[NDB_MGMD]
# IP address of the management node (this system)
ID=2
HostName=192.168.131.26

# Section for the storage nodes
[NDBD]
# IP address of the first storage node
HostName=192.168.131.77
```

火龙果•整理
uml.org.cn

```
DataDir= /var/lib/mysql-cluster

[NDBD]
# IP address of the second storage node
HostName=192.168.131.101
DataDir=/var/lib/mysql-cluster

# one [MYSQLD] per storage node
[MYSQLD]
[MYSQLD]
[MYSQLD]
[MYSQLD]
[MYSQLD]
[MYSQLD]
[MYSQLD]
:wq
```

以下在mysql API上操作（这里，我设定了7个API，以后可以随时加入）

Mysqld API的配置文件：

```
Vi /etc/my.cnf
[root@ndb1 ~]# cat /etc/my.cnf
[mysqld]
ndbcluster
ndb-connectstring = "host=192.168.131.164,host=192.168.131.26"

[ndb_mgm]
connect-string = "host=192.168.131.164,host=192.168.131.26"

[ndbd]
connect-string = "host=192.168.131.164,host=192.168.131.26"
:wq
```

分别启动ndb_mgmd/ndbd/mysqld

```
164/26:
ndb_mgmd -f /var/lib/mysql-cluster/config.ini

77/101:
Ndbd --initial

164/26/77/101:
/etc/rc.d/init.d/mysql start
```

在管理节点ndb1(164)和ndb2(26)上查看各节点的情况：

火龙果•整理
uml.org.cn

```
[root@ndb1 ~]# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: 192.168.131.164:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=3    @192.168.131.77  (Version: 5.0.67, Nodegroup: 0, Master)
id=4    @192.168.131.101  (Version: 5.0.67, Nodegroup: 0)

[ndb_mgmd(MGM)] 2 node(s)
id=1    @192.168.131.164  (Version: 5.0.67)
id=2    @192.168.131.26  (Version: 5.0.67)

[mysqld(API)]   7 node(s)
id=5    @192.168.131.101  (Version: 5.0.67)
id=6    @192.168.131.26  (Version: 5.0.67)
id=7    @192.168.131.164  (Version: 5.0.67)
id=8    @192.168.131.77  (Version: 5.0.67)
id=9 (not connected, accepting connect from any host)
id=10 (not connected, accepting connect from any host)
id=11 (not connected, accepting connect from any host)

ndb_mgm>
```

以上说明一切正常。

将服务增加到开机启动服务项中：

164/26:

```
echo 'ndb_mgmd -f /var/lib/mysql-cluster/config.ini' > /etc/rc.d/init.d/ndb_mgmd
chmod 755 /etc/rc.d/init.d/ndb_mgmd
```

77/101:

```
Echo 'ndbd' > /etc/rc.d/init.d/ndbd
Chmod 755 /etc/rc.d/init.d/ndbd
Chkconfig --level 2345 ndbd on
```

OK，到此 mysql cluster 配置完成。

强调：

1）由于数据放在内存中，需要在 ndb 节点上加大内存的数量。按照1：1.1的比例，如果数据量达到3.6GB，需要4GB 的内存。

2）由于 NDB 和 mysqld（API）都很耗费内存，所以建议将 NDB 放在164和26上。可能启动的时候会有警告，但是没关系的。

火龙果•整理
uml.org.cn

查看数据和内存情况：

77：

```
[root@sql2 ~]# top
top - 16:39:36 up  1:59,  1 user,  load average: 1.37, 0.76, 0.60
Tasks:  80 total,   2 running,  78 sleeping,   0 stopped,   0 zombie
Cpu(s):  4.0%us,  4.0%sy,  0.0%ni, 87.3%id,  2.9%wa,  0.2%hi,  1.5%si,  0.0%st
Mem:   2075600k total,  2005868k used,    69732k free,    68256k buffers
Swap:  2031608k total,        0k used,  2031608k free,  1400812k cached

PID  USER      PR   NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+   COMMAND
2306 mysql     25   0  119m   15m   3952 S   22   0.8       10:20.94 mysqld
23791 root      15   0  1587m  484m  31m R   20   23.9      9:34.97 ndbd
```

由于77只有2GB的内存，而在config.ini中，把1.2GB的内存分配给了NDB，所以，加上mysqld用掉的，2GB的内存似乎已经所剩无几了。

查看77上的数据大小：

```
[root@sql2 ~]# cd /var/lib/mysql-cluster/ndb_4_fs/
[root@sql2 ndb_4_fs]# du -lh
1.3GB
```

连接API创建数据库：

由于上面4台都做为mysqld 的API，所以创建数据库的时候，都需要创建一遍。

以下操作在4台API上都需要操作：

```
# Mysql -uroot -pxxxxxxxxxxxx -A
Mysql> create database testdatabase;
Mysql> grant all on *.testdatabase to root@'192.168.131.%' identified by 'xxxxxxxxxxxxxx';
Mysql> flush privileges;
Mysql> create table test(int (1));
Mysql> insert into test(1);
Mysql> quit;
```

以上做完以后，可以通过任意一台API上创建表，并写数据到表中，其他数据库都会同步写入。

分别连接每台服务器进行检查：

```
# Mysql -uroot -pxxxxxxxxxxxx -A
Mysql> use testdatabase;
Mysql> select * from test;
```

如果输出结果完全相同，表明mysql cluster已经可以正常工作了。

火龙果•整理
uml.org.cn

在2台 API 上设置 LVS

Mysql cluster 做好以后，数据库分别建立同名的数据库以后，权限分配好，然后只要在一台上写入数据，其他的 NDB 就存储了相同的数据。

用程序连接任意一台 API 写数据，如果程序中未设置 API 的选择和判断，只使用了其中一个 API，一旦 API 当机，则无法写入数据，必须修改程序。即便做了 API 的判断和选择，因为没有实现负载均衡，服务器的性能没有充分利用。高可用性也没有达到目标。所以，我们现在在2台 API 之间做 LVS。

LVS 采用 ultramonkey（http://www.ultramonkey.org）

首先在 NDB1（164）和 NDB2（26）上下载 heartbeat 的软件包：
下载所有的 rpm 包：
Cd /usr/local/src
Mkdir heartbeat
Cd heartbeat
#Wget xxx.xxx.rpm

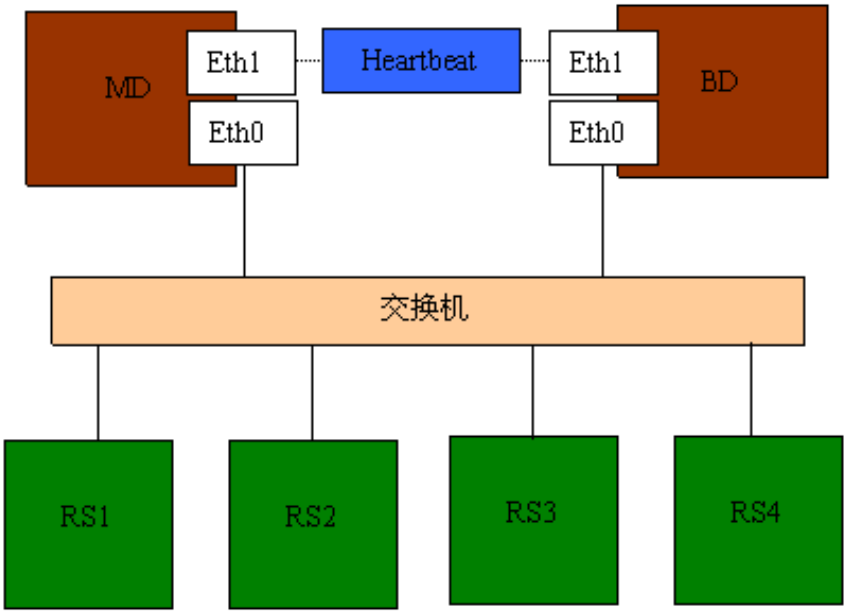
我下载了如下的软件包：

```
[root@ndb1 heartbeat]# ls -lh *.rpm | awk '{print $9}';
arptables-noarp-addr-0.99.2-1.rh.el.um.1.noarch.rpm
heartbeat-1.2.3.cvs.20050927-1.rh.el.um.4.i386.rpm
heartbeat-ldirectord-1.2.3.cvs.20050927-1.rh.el.um.4.i386.rpm
heartbeat-pils-1.2.3.cvs.20050927-1.rh.el.um.4.i386.rpm
heartbeat-stonith-1.2.3.cvs.20050927-1.rh.el.um.4.i386.rpm
ipvsadm-1.21-1.rh.el.1.um.1.i386.rpm
libnet-1.1.2.1-1.rh.el.um.1.i386.rpm
perl-Authen-SASL-2.08-1.rh.el.um.1.noarch.rpm
perl-Convert-ASN1-0.18-1.rh.el.um.1.noarch.rpm
perl-IO-Socket-SSL-0.96-1.rh.el.um.1.noarch.rpm
perl-ldap-0.3202-1.rh.el.um.1.noarch.rpm
perl-Mail-IMAPClient-2.2.9-1.rh.el.um.1.noarch.rpm
perl-Net-SSLeay-1.25-1.rh.el.um.1.i386.rpm
perl-Parse-RecDescent-1.94-1.el5.rf.noarch.rpm
perl-Parse-RecDescent-1.94-1.rh.el.um.1.noarch.rpm
perl-XML-NamespaceSupport-1.08-1.rh.el.um.1.noarch.rpm
perl-XML-SAX-0.12-1.rh.el.um.1.noarch.rpm
[root@ndb1 heartbeat]#
```

Heartbeat 中包含以下几部分：

1) Master Director（分发器）-- MD
2) Backup Director（备份分发器）-- BD
3) Real server （真实服务器，可以有2个以上）--RS

火龙果•整理
uml.org.cn



IP 设置并确认：

MD:
Eth0:192.168.131.164/24/GW:192.168.131.1
Eth1:10.9.30.1/24

MD:
Eth0:192.168.131.26/24/GW:192.168.131.1
Eth1:10.9.30.2

VIP:192.168.131.105/24/GW:192.168.131.1 -- 用户访问的统一虚拟 IP

RS1:192.168.131.101/24/GW:192.168.131.1
RS2:192.168.131.77/24/GW:192.168.131.1
。。。
等等

以下操作在所有服务器上执行：

主机名确认：

分别执行：
#uname -a
主机名对应表中所列。

在 MD 和 BD 修改 IP 转发：

#vi modprobe.sh
modprobe ip_vs_dh
modprobe ip_vs_ftp
modprobe ip_vs

火龙果•整理
uml.org.cn

```
modprobe ip_vs_lblc
modprobe ip_vs_lblcr
modprobe ip_vs_lc
modprobe ip_vs_nq
modprobe ip_vs_rr
modprobe ip_vs_sed
modprobe ip_vs_sh
modprobe ip_vs_wlc
modprobe ip_vs_wrr
:wq
```

```
#chmod 755 modprobe.sh

# sh modprobe.sh

# vi /etc/modules

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

:wq


#Vi

net.ipv4.ip_forward = 0
```

改为：

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

火龙果•整理
uml.org.cn

使修改生效：

/sbin/sysctl -p

在 MD 和 BD 上安装 heartbeat 软件包

#Rpm -Uvh perl-xx-xx-xx.rpm

#Yum install heartbeat

#Rpm -Uvh arptables-noarp-addr-0.99.2-1.rh.el.um.1.noarch.rpm

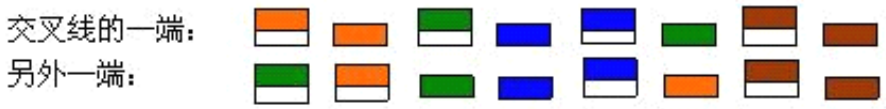
#rpm -Uvh perl-Mail-POP3Client-2.17-1.el5.centos.noarch.rpm

缺少 perl 包，就使用 yum install perl-xx-xx

#Perl -CPAN -e shell

这样安装的 perl 包不知道为何不好使？奇怪

这里 VIP 实际上是绑定在2台 director 上。所以 director 之间需要做心跳处理。心跳线使用 eth1口，用交叉线连接起来。



这样可以避免影响其他服务器。

配置 heartbeat

Heartbeat 有3个配置文件：

Ha.cf

Authkeys

Haresources

火龙果•整理
uml.org.cn

ldirectord进程的配置文件

Ldirectord.cf

一共需要配置4个配置文件。

```
#vi ha.cf
logfacility  local0
bcast        eth1
mcast eth1 225.0.0.1 694 1 0
auto_failback off
node        ndb1
node        ndb2
respawn hacluster /usr/lib/heartbeat/ipfail
apiauth ipfail gid=haclient uid=hacluster

:wq

# vi authkeys
auth 3
3 md5 514a49f83820e34c877ff48770e48ea7

:wq

# vi haresources
ndb1        \
        ldirectord::ldirectord.cf \
        LVSSyncDaemonSwap::master \
```

火龙果•整理
uml.org.cn

IPaddr2::192.168.131.105/24/eth0/192.168.131.255

Ndb2上需要将主机名更改一下。

:wq

设置属性并使 heartbeat 开机启动

chmod 600 /etc/ha.d/authkeys

#/sbin/chkconfig --level 2345 heartbeat on

#/sbin/chkconfig --del ldirectord

启动 heartbeat：

/etc/init.d/ldirectord stop

/etc/init.d/heartbeat start

在 MD 和 BD 上检查 VIP 是否生效：

ip addr sh eth0

[root@ndb1 ha.d]# ip addr sh eth0

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000

link/ether 00:30:48:28:c6:85 brd ff:ff:ff:ff:ff:ff

inet 192.168.131.164/24 brd 192.168.131.255 scope global eth0

inet 192.168.131.105/24 brd 192.168.131.255 scope global secondary eth0

inet6 fe80::230:48ff:fe28:c685/64 scope link

valid_lft forever preferred_lft forever

[root@ndb1 ha.d]#

[root@ndb2 ~]# ip addr sh eth0

火龙果•整理
uml.org.cn

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:30:48:28:c4:af brd ff:ff:ff:ff:ff:ff
    inet 192.168.131.26/24 brd 192.168.131.255 scope global eth0
    inet6 fe80::230:48ff:fe28:c4af/64 scope link
        valid_lft forever preferred_lft forever
[root@ndb2 ~]#
```

现在在 MD（164）上已经生效了。

检查 ldirectored 进程

```
[root@ndb1 ha.d]# /usr/sbin/ldirectord ldirectord.cf status
ldirectord for /etc/ha.d/ldirectord.cf is running with pid: 5596
[root@ndb1 ha.d]#
[root@ndb2 ~]# /usr/sbin/ldirectord ldirectord.cf status
ldirectord is stopped for /etc/ha.d/ldirectord.cf
[root@ndb2 ~]#
```

VIP 生效的 director 应该是 running 状态，standby 应该是 stop 状态。

利用 ipvs 检查包转发是否生效

```
[root@ndb1 ha.d]# /sbin/ipvsadm -L -n
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.131.105:3306 wrr
```

火龙果•整理
uml.org.cn

```
  -> 192.168.131.77:3306          Route    1      3          3034
  -> 192.168.131.101:3306         Route    1      3          3038
[root@ndb1 ha.d]#
```

```
[root@ndb2 ~]# /sbin/ipvsadm -L -n
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
[root@ndb2 ~]#
```

在 MB 上已经生效了。

在 MD 和 BD 上检查 LVSSyncDaemonSwap 的状态：

```
[root@ndb1 ha.d]# /etc/ha.d/resource.d/LVSSyncDaemonSwap master status
master running
(ipvs_syncmaster pid: 5689)
[root@ndb1 ha.d]#
[root@ndb2 ~]# /etc/ha.d/resource.d/LVSSyncDaemonSwap master status
master stopped
(ipvs_syncbackup pid: 5493)
[root@ndb2 ~]#
```

同样，standby 的处于 stopped 状态。

火龙果•整理
uml.org.cn

以下在 RS 服务器上执行：

ARP 转发限制

MD 或者 BD 采用 ARP 欺骗将 ARP 包转发给下面的 realserver。为了转发成功，需要做 ARP 限制。

```
#/etc/init.d/arptables_jf stop

#/usr/sbin/arptables-noarp-addr 192.168.6.240 start

#/etc/init.d/arptables_jf save

#/sbin/chkconfig --level 2345 arptables_jf on

#/etc/init.d/arptables_jf start
```

查看限制链表

```
[root@sql2 mysql-cluster]# /sbin/arptables -L -v -n

Chain IN (policy ACCEPT 29243 packets, 819K bytes)

 pkts bytes target     in     out     source-ip            destination-ip
source-hw          destination-hw     hlen   op         hrd        pro

   54  1512 DROP       *      *       0.0.0.0/0            192.168.131.105
00/00              00/00              any    0000/0000  0000/0000  0000/0000


Chain OUT (policy ACCEPT 3931 packets, 110K bytes)

 pkts bytes target     in     out     source-ip            destination-ip
source-hw          destination-hw     hlen   op         hrd        pro

    0     0 mangle     *      eth0    192.168.131.105      0.0.0.0/0
00/00              00/00              any    0000/0000  0000/0000  0000/0000
--mangle-ip-s 192.168.131.101
```

火龙果•整理
uml.org.cn

```
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     in     out     source-ip            destination-ip
source-hw          destination-hw     hlen   op         hrd        pro
[root@sql2 mysql-cluster]#


[root@sql1 ~]# /sbin/arptables -L -v -n
Chain IN (policy ACCEPT 29375 packets, 823K bytes)
 pkts bytes target     in     out     source-ip            destination-ip
source-hw          destination-hw     hlen   op         hrd        pro
   54  1512 DROP       *      *       0.0.0.0/0            192.168.131.105
00/00              00/00              any    0000/0000  0000/0000  0000/0000


Chain OUT (policy ACCEPT 3903 packets, 109K bytes)
 pkts bytes target     in     out     source-ip            destination-ip
source-hw          destination-hw     hlen   op         hrd        pro
    0     0 mangle     *      eth0    192.168.131.105      0.0.0.0/0
00/00              00/00              any    0000/0000  0000/0000  0000/0000
--mangle-ip-s 192.168.131.77


Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     in     out     source-ip            destination-ip
source-hw          destination-hw     hlen   op         hrd        pro
[root@sql1 ~]#
```

火龙果•整理
uml.org.cn

这样，由 MD 或者 BD 转发过来的 ARP 包就被链表控制了。

设置如何接收 ARP 包

以下在所有 RS 上执行

```
# cp /etc/sysconfig/network-scripts/ifcfg-lo
/etc/sysconfig/network-scripts/ifcfg-lo:0
#Vi /etc/sysconfig/network-scripts/ifcfg-lo\:0
DEVICE=lo:0
IPADDR=192.168.131.105
NETMASK=255.255.255.255
NETWORK=192.168.131.0
BROADCAST=192.168.131.255
ONBOOT=yes
NAME=loopback
:wq
#/sbin/ifup lo
```

查看 lo:0

```
[root@sql1 ~]# ip addr sh lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet 192.168.131.105/32 brd 192.168.131.255 scope global lo:0
```

火龙果•整理
uml.org.cn

```
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
[root@sql1 ~]#


[root@sql2 mysql-cluster]# ip addr sh lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet 192.168.131.105/32 brd 192.168.131.255 scope global lo:0
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
[root@sql2 mysql-cluster]#
```

重新启动服务器

以下在所有服务器上执行（请确认 ip，服务器上没有 running 任何正在使用的服务）

```
reboot
```

启动 mysql cluster：

顺序：

ndb_mgmd -- 164/26

Ndbd -- 101/77

Mysqld -- 所有

检查服务是否正常

火龙果•整理
uml.org.cn

以下在 ndb 上执行

#ndb_mgm

```
[root@ndb1 ha.d]# ndb_mgm

-- NDB Cluster -- Management Client --

ndb_mgm> show

Connected to Management Server at: 192.168.131.164:1186

Cluster Configuration

---------------------

[ndbd(NDB)]     2 node(s)

id=3    @192.168.131.77  (Version: 5.0.67, Nodegroup: 0, Master)

id=4    @192.168.131.101  (Version: 5.0.67, Nodegroup: 0)


[ndb_mgmd(MGM)] 2 node(s)

id=1    @192.168.131.164  (Version: 5.0.67)

id=2    @192.168.131.26  (Version: 5.0.67)


[mysqld(API)]   7 node(s)

id=5    @192.168.131.101  (Version: 5.0.67)

id=6    @192.168.131.26  (Version: 5.0.67)

id=7    @192.168.131.164  (Version: 5.0.67)

id=8    @192.168.131.77  (Version: 5.0.67)

id=9 (not connected, accepting connect from any host)

id=10 (not connected, accepting connect from any host)
```

火龙果•整理
uml.org.cn

id=11 (not connected, accepting connect from any host)

ndb_mgm>

一切正常。

检查 heartbeat 是否正常：

关闭 BD，在 MD 上查看日志：

[root@ndb1 ha.d]# tail -f /var/log/messages

Dec 17 19:42:21 ndb1 heartbeat: [5462]: info: Received shutdown notice from 'ndb2'.

Dec 17 19:42:21 ndb1 heartbeat: [5462]: info: Resources being acquired from ndb2.

Dec 17 19:42:21 ndb1 harc[7085]: info: Running /etc/ha.d/rc.d/status status

Dec 17 19:42:21 ndb1 mach_down[7118]: info: /usr/share/heartbeat/mach_down: nice_failback: foreign resources acquired

Dec 17 19:42:21 ndb1 mach_down[7118]: info: mach_down takeover complete for node ndb2.

Dec 17 19:42:21 ndb1 heartbeat: [5462]: info: mach_down takeover complete.

Dec 17 19:42:21 ndb1 ldirectord[7153]: Invoking ldirectord invoked as: /etc/ha.d/resource.d/ldirectord ldirectord.cf status

Dec 17 19:42:21 ndb1 ldirectord[7153]: ldirectord for /etc/ha.d/ldirectord.cf is running with pid: 5596

Dec 17 19:42:21 ndb1 ldirectord[7153]: Exiting from ldirectord status

Dec 17 19:42:21 ndb1 heartbeat: [7086]: info: Local Resource acquisition completed.

Dec 17 19:42:21 ndb1 harc[7175]: info: Running /etc/ha.d/rc.d/ip-request-resp ip-request-resp

火龙果•整理
uml.org.cn

Dec 17 19:42:21 ndb1 ip-request-resp[7175]: received ip-request-resp ldirectord::ldirectord.cf OK yes

Dec 17 19:42:21 ndb1 ResourceManager[7196]: info: Acquiring resource group: ndb1 ldirectord::ldirectord.cf LVSSyncDaemonSwap::master IPaddr2::192.168.131.105/24/eth0/192.168.131.255

Dec 17 19:42:22 ndb1 ldirectord[7223]: Invoking ldirectord invoked as: /etc/ha.d/resource.d/ldirectord ldirectord.cf status

Dec 17 19:42:22 ndb1 ldirectord[7223]: ldirectord for /etc/ha.d/ldirectord.cf is running with pid: 5596

Dec 17 19:42:22 ndb1 ldirectord[7223]: Exiting from ldirectord status

Dec 17 19:42:22 ndb1 ResourceManager[7196]: info: Running /etc/ha.d/resource.d/ldirectord ldirectord.cf start

Dec 17 19:42:23 ndb1 ldirectord[7245]: Invoking ldirectord invoked as: /etc/ha.d/resource.d/ldirectord ldirectord.cf start

Dec 17 19:42:23 ndb1 IPaddr2[7291]: INFO: Running OK

如果没有出现异常，表明一切正常。

破坏性试验

1) 检查 ndbd

关闭任意一台 ndbd 的进程，在 ndb_mgm 上查看是否失去连接。

如果失去连接，表示已经识别出来。

此时在数据库表中增加内容之后启动刚刚关闭的 ndbd，检查新写入的数据是否已经被同步过来。如果同步过来，一切正常。

火龙果•整理
uml.org.cn

2) 检查 heartbeat

关闭 MD，检查 BD 的反应：

[root@ndb2 ~]# tail -f /var/log/messages

Dec 17 19:47:22 ndb2 harc[6862]: info: Running /etc/ha.d/rc.d/status status

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: Comm_now_up(): updating status to active

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: Local status now set to: 'active'

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: Starting child client "/usr/lib/heartbeat/ipfail" (498,496)

Dec 17 19:47:23 ndb2 heartbeat: [6879]: info: Starting "/usr/lib/heartbeat/ipfail" as uid 498 gid 496 (pid 6879)

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: remote resource transition completed.

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: remote resource transition completed.

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: Local Resource acquisition completed. (none)

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: Initial resource acquisition complete (T_RESOURCES(them))

Dec 17 19:47:29 ndb2 ipfail: [6879]: info: Ping node count is balanced.

Dec 17 19:47:43 ndb2 heartbeat: [6852]: info: Received shutdown notice from 'ndb1'.

Dec 17 19:47:43 ndb2 heartbeat: [6852]: info: Resources being acquired from ndb1.

Dec 17 19:47:43 ndb2 heartbeat: [6884]: info: acquire all HA resources (standby).

Dec 17 19:47:43 ndb2 ResourceManager[6911]: info: Acquiring resource group: ndb2 ldirectord::ldirectord.cf LVSSyncDaemonSwap::master IPaddr2::192.168.131.105/24/eth0/192.168.131.255

火龙果•整理
uml.org.cn

Dec 17 19:47:43 ndb2 ldirectord[6957]: ldirectord is stopped for /etc/ha.d/ldirectord.cf

Dec 17 19:47:43 ndb2 ldirectord[6957]: Exiting with exit_status 3: Exiting from ldirectord status

Dec 17 19:47:43 ndb2 heartbeat: [6885]: info: Local Resource acquisition completed.

Dec 17 19:47:43 ndb2 ldirectord[6961]: ldirectord is stopped for /etc/ha.d/ldirectord.cf

Dec 17 19:47:43 ndb2 ldirectord[6961]: Exiting with exit_status 3: Exiting from ldirectord status

Dec 17 19:47:43 ndb2 ResourceManager[6911]: info: Running /etc/ha.d/resource.d/ldirectord ldirectord.cf start

Dec 17 19:47:44 ndb2 ldirectord[6986]: Starting Linux Director v1.77.2.32 as daemon

Dec 17 19:47:44 ndb2 ldirectord[6988]: Added virtual server: 192.168.131.105:3306

Dec 17 19:47:44 ndb2 ldirectord[6988]: Quiescent real server: 192.168.131.101:3306 mapped from 192.168.131.101:3306 (x 192.168.131.105:3306) (Weight set to 0)

Dec 17 19:47:44 ndb2 ldirectord[6988]: Quiescent real server: 192.168.131.77:3306 mapped from 192.168.131.77:3306 (x 192.168.131.105:3306) (Weight set to 0)

Dec 17 19:47:44 ndb2 ResourceManager[6911]: info: Running /etc/ha.d/resource.d/LVSSyncDaemonSwap master start

Dec 17 19:47:44 ndb2 kernel: IPVS: stopping sync thread 5493 ...

Dec 17 19:47:45 ndb2 kernel: IPVS: sync thread stopped!

Dec 17 19:47:45 ndb2 LVSSyncDaemonSwap[7050]: info: ipvs_syncbackup down

Dec 17 19:47:45 ndb2 kernel: IPVS: sync thread started: state = MASTER, mcast_ifn = eth0, syncid = 0

Dec 17 19:47:45 ndb2 LVSSyncDaemonSwap[7050]: info: ipvs_syncmaster up

火龙果•整理
uml.org.cn

Dec 17 19:47:45 ndb2 LVSSyncDaemonSwap[7050]: info: ipvs_syncmaster obtained

Dec 17 19:47:45 ndb2 IPaddr2[7102]: INFO: Resource is stopped

Dec 17 19:47:45 ndb2 ResourceManager[6911]: info: Running /etc/ha.d/resource.d/IPaddr2 192.168.131.105/24/eth0/192.168.131.255 start

Dec 17 19:47:45 ndb2 IPaddr2[7214]: INFO: ip -f inet addr add 192.168.131.105/24 brd 192.168.131.255 dev eth0

Dec 17 19:47:45 ndb2 avahi-daemon[2776]: Registering new address record for 192.168.131.105 on eth0.

Dec 17 19:47:45 ndb2 IPaddr2[7214]: INFO: ip link set eth0 up

Dec 17 19:47:45 ndb2 IPaddr2[7214]: INFO: /usr/lib/heartbeat/send_arp -i 200 -r 5 -p /var/run/heartbeat/rsctmp/send_arp/send_arp-192.168.131.105 eth0 192.168.131.105 auto not_used not_used

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 IPaddr2[7185]: INFO: Success

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 heartbeat: [6884]: info: all HA resource acquisition completed (standby).

Dec 17 19:47:45 ndb2 heartbeat: [6852]: info: Standby resource acquisition done [all].

Dec 17 19:47:45 ndb2 harc[7277]: info: Running /etc/ha.d/rc.d/status status

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 last message repeated 14 times

Dec 17 19:47:45 ndb2 mach_down[7293]: info: /usr/share/heartbeat/mach_down:

火龙果•整理
uml.org.cn

nice_failback: foreign resources acquired

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 mach_down[7293]: info: mach_down takeover complete for node ndb1.

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 heartbeat: [6852]: info: mach_down takeover complete.

Dec 17 19:47:45 ndb2 harc[7327]: info: Running /etc/ha.d/rc.d/ip-request-resp ip-request-resp

Dec 17 19:47:45 ndb2 ip-request-resp[7327]: received ip-request-resp ldirectord::ldirectord.cf OK yes

Dec 17 19:47:45 ndb2 ResourceManager[7348]: info: Acquiring resource group: ndb2 ldirectord::ldirectord.cf LVSSyncDaemonSwap::master IPaddr2::192.168.131.105/24/eth0/192.168.131.255

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:46 ndb2 last message repeated 3 times

Dec 17 19:47:46 ndb2 ldirectord[7375]: ldirectord for /etc/ha.d/ldirectord.cf is running with pid: 6988

Dec 17 19:47:46 ndb2 ldirectord[7375]: Exiting from ldirectord status

Dec 17 19:47:46 ndb2 ResourceManager[7348]: info: Running /etc/ha.d/resource.d/ldirectord ldirectord.cf start

Dec 17 19:47:46 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:46 ndb2 last message repeated 6 times

Dec 17 19:47:46 ndb2 IPaddr2[7443]: INFO: Running OK

Dec 17 19:47:46 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

火龙果•整理
uml.org.cn

Dec 17 19:48:16 ndb2 last message repeated 289 times

Dec 17 19:48:16 ndb2 heartbeat: [6852]: WARN: node ndb1: is dead

Dec 17 19:48:16 ndb2 heartbeat: [6852]: info: Dead node ndb1 gave up resources.

Dec 17 19:48:16 ndb2 heartbeat: [6852]: info: Link ndb1:eth1 dead.

Dec 17 19:48:16 ndb2 ipfail: [6879]: info: Status update: Node ndb1 now has status dead

Dec 17 19:48:16 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:48:17 ndb2 last message repeated 8 times

Dec 17 19:48:17 ndb2 ipfail: [6879]: info: NS: We are dead. :<

Dec 17 19:48:17 ndb2 ipfail: [6879]: info: Link Status update: Link ndb1/eth1 now has status dead

Dec 17 19:48:17 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:48:17 ndb2 ipfail: [6879]: info: We are dead. :<

Dec 17 19:48:17 ndb2 ipfail: [6879]: info: Asking other side for ping node count.

Dec 17 19:48:18 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers[root@ndb2 ~]# tail -f /var/log/messages

Dec 17 19:47:22 ndb2 harc[6862]: info: Running /etc/ha.d/rc.d/status status

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: Comm_now_up(): updating status to active

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: Local status now set to: 'active'

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: Starting child client "/usr/lib/heartbeat/ipfail" (498,496)

Dec 17 19:47:23 ndb2 heartbeat: [6879]: info: Starting "/usr/lib/heartbeat/ipfail" as uid 498 gid 496 (pid 6879)

火龙果•整理
uml.org.cn

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: remote resource transition completed.

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: remote resource transition completed.

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: Local Resource acquisition completed. (none)

Dec 17 19:47:23 ndb2 heartbeat: [6852]: info: Initial resource acquisition complete (T_RESOURCES(them))

Dec 17 19:47:29 ndb2 ipfail: [6879]: info: Ping node count is balanced.

Dec 17 19:47:43 ndb2 heartbeat: [6852]: info: Received shutdown notice from 'ndb1'.

Dec 17 19:47:43 ndb2 heartbeat: [6852]: info: Resources being acquired from ndb1.

Dec 17 19:47:43 ndb2 heartbeat: [6884]: info: acquire all HA resources (standby).

Dec 17 19:47:43 ndb2 ResourceManager[6911]: info: Acquiring resource group: ndb2 ldirectord::ldirectord.cf LVSSyncDaemonSwap::master IPaddr2::192.168.131.105/24/eth0/192.168.131.255

Dec 17 19:47:43 ndb2 ldirectord[6957]: ldirectord is stopped for /etc/ha.d/ldirectord.cf

Dec 17 19:47:43 ndb2 ldirectord[6957]: Exiting with exit_status 3: Exiting from ldirectord status

Dec 17 19:47:43 ndb2 heartbeat: [6885]: info: Local Resource acquisition completed.

Dec 17 19:47:43 ndb2 ldirectord[6961]: ldirectord is stopped for /etc/ha.d/ldirectord.cf

Dec 17 19:47:43 ndb2 ldirectord[6961]: Exiting with exit_status 3: Exiting from ldirectord status

Dec 17 19:47:43 ndb2 ResourceManager[6911]: info: Running /etc/ha.d/resource.d/ldirectord ldirectord.cf start

Dec 17 19:47:44 ndb2 ldirectord[6986]: Starting Linux Director v1.77.2.32 as daemon

火龙果•整理
uml.org.cn

Dec 17 19:47:44 ndb2 ldirectord[6988]: Added virtual server: 192.168.131.105:3306

Dec 17 19:47:44 ndb2 ldirectord[6988]: Quiescent real server: 192.168.131.101:3306 mapped from 192.168.131.101:3306 (x 192.168.131.105:3306) (Weight set to 0)

Dec 17 19:47:44 ndb2 ldirectord[6988]: Quiescent real server: 192.168.131.77:3306 mapped from 192.168.131.77:3306 (x 192.168.131.105:3306) (Weight set to 0)

Dec 17 19:47:44 ndb2 ResourceManager[6911]: info: Running /etc/ha.d/resource.d/LVSSyncDaemonSwap master start

Dec 17 19:47:44 ndb2 kernel: IPVS: stopping sync thread 5493 ...

Dec 17 19:47:45 ndb2 kernel: IPVS: sync thread stopped!

Dec 17 19:47:45 ndb2 LVSSyncDaemonSwap[7050]: info: ipvs_syncbackup down

Dec 17 19:47:45 ndb2 kernel: IPVS: sync thread started: state = MASTER, mcast_ifn = eth0, syncid = 0

Dec 17 19:47:45 ndb2 LVSSyncDaemonSwap[7050]: info: ipvs_syncmaster up

Dec 17 19:47:45 ndb2 LVSSyncDaemonSwap[7050]: info: ipvs_syncmaster obtained

Dec 17 19:47:45 ndb2 IPaddr2[7102]: INFO: Resource is stopped

Dec 17 19:47:45 ndb2 ResourceManager[6911]: info: Running /etc/ha.d/resource.d/IPaddr2 192.168.131.105/24/eth0/192.168.131.255 start

Dec 17 19:47:45 ndb2 IPaddr2[7214]: INFO: ip -f inet addr add 192.168.131.105/24 brd 192.168.131.255 dev eth0

Dec 17 19:47:45 ndb2 avahi-daemon[2776]: Registering new address record for 192.168.131.105 on eth0.

Dec 17 19:47:45 ndb2 IPaddr2[7214]: INFO: ip link set eth0 up

Dec 17 19:47:45 ndb2 IPaddr2[7214]: INFO: /usr/lib/heartbeat/send_arp -i 200 -r 5 -p /var/run/heartbeat/rsctmp/send_arp/send_arp-192.168.131.105 eth0 192.168.131.105 auto not_used not_used

火龙果•整理
uml.org.cn

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 IPaddr2[7185]: INFO: Success

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 heartbeat: [6884]: info: all HA resource acquisition completed (standby).

Dec 17 19:47:45 ndb2 heartbeat: [6852]: info: Standby resource acquisition done [all].

Dec 17 19:47:45 ndb2 harc[7277]: info: Running /etc/ha.d/rc.d/status status

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 last message repeated 14 times

Dec 17 19:47:45 ndb2 mach_down[7293]: info: /usr/share/heartbeat/mach_down: nice_failback: foreign resources acquired

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 mach_down[7293]: info: mach_down takeover complete for node ndb1.

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:45 ndb2 heartbeat: [6852]: info: mach_down takeover complete.

Dec 17 19:47:45 ndb2 harc[7327]: info: Running /etc/ha.d/rc.d/ip-request-resp ip-request-resp

Dec 17 19:47:45 ndb2 ip-request-resp[7327]: received ip-request-resp ldirectord::ldirectord.cf OK yes

Dec 17 19:47:45 ndb2 ResourceManager[7348]: info: Acquiring resource group: ndb2 ldirectord::ldirectord.cf LVSSyncDaemonSwap::master

火龙果•整理
uml.org.cn

IPaddr2::192.168.131.105/24/eth0/192.168.131.255

Dec 17 19:47:45 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:46 ndb2 last message repeated 3 times

Dec 17 19:47:46 ndb2 ldirectord[7375]: ldirectord for /etc/ha.d/ldirectord.cf is running with pid: 6988

Dec 17 19:47:46 ndb2 ldirectord[7375]: Exiting from ldirectord status

Dec 17 19:47:46 ndb2 ResourceManager[7348]: info: Running /etc/ha.d/resource.d/ldirectord ldirectord.cf start

Dec 17 19:47:46 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:47:46 ndb2 last message repeated 6 times

Dec 17 19:47:46 ndb2 IPaddr2[7443]: INFO: Running OK

Dec 17 19:47:46 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:48:16 ndb2 last message repeated 289 times

Dec 17 19:48:16 ndb2 heartbeat: [6852]: WARN: node ndb1: is dead

Dec 17 19:48:16 ndb2 heartbeat: [6852]: info: Dead node ndb1 gave up resources.

Dec 17 19:48:16 ndb2 heartbeat: [6852]: info: Link ndb1:eth1 dead.

Dec 17 19:48:16 ndb2 ipfail: [6879]: info: Status update: Node ndb1 now has status dead

Dec 17 19:48:16 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

Dec 17 19:48:17 ndb2 last message repeated 8 times

Dec 17 19:48:17 ndb2 ipfail: [6879]: info: NS: We are dead. :<

Dec 17 19:48:17 ndb2 ipfail: [6879]: info: Link Status update: Link ndb1/eth1 now has status dead

Dec 17 19:48:17 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers

火龙果•整理
uml.org.cn

```
Dec 17 19:48:17 ndb2 ipfail: [6879]: info: We are dead. :<

Dec 17 19:48:17 ndb2 ipfail: [6879]: info: Asking other side for ping node count.

Dec 17 19:48:18 ndb2 kernel: IPVS: ip_vs_wrr_schedule(): no available servers
```

如果没有错误，表明 heartbeat 已经切换。

此时再次插入数据验证，如果还可以继续写入，表明配置完全成功。

Mysql cluster 的测试报告：

在192.168.8.48上部署测试脚本，让这台服务器表示一个客户端请求读写数据库。

测试脚本1：

```
[root@localhost mysql-cluster]# cat /data/pay.kingsoft.com/wwwroot/test.php

<?php



$link = mysql_connect('192.168.131.105', 'ldirector', 'xxxxxxxxx');

mysql_select_db('kingsoft',$link);

$sql = "insert into `preference`(`id`,`preferenceSerialNumber`,`username`,`preferenceTypeId`,`isExpired`,`isUsed`,`preferenceUsername`,`equalMoney`,`genDatetime`,`useDatetime`,`grantDatetime`,`expriedDatetime`) values ( NULL,'514a49f83820e34c877ff48770e48ea7','liujun','2','1','1','kingsoft','512.23','2008-12-03','2008-12-03','2008-12-03','2008-12-03')";

for($i = 0;$i < 100 ;$i++){

        mysql_query($sql);
```

火龙果•整理
uml.org.cn

```
}
mysql_close($link);
?>
```

测试脚本2：

```
[root@localhost mysql-cluster]# cat test.sh
#!/bin/sh
i=0;
j=0;

        while [ $i -lt  1000 ]
        do
                wget -q http://pay.kingsoft.sug/test.php;
                i=`expr $i + 1`;
        done

sleep 2;

find . -name "test.php.*" | xargs rm -rf ;

while [ $j -lt 1000 ]
do
        mysql  -uldirector  -pxxxxxxxxxxx   -h192.168.131.105  -e  "use  kingsoft;
```

火龙果•整理
uml.org.cn

```
insert into preference(preferenceSerialNumber,username,preferenceTypeId,preferenceUsername,equalMoney,genDatetime,useDatetime,grantDatetime,expriedDatetime) values('514a49f83820e34c877ff48770e48ea7','liujun2','3','liujun33333','33.8','2008-12-23 7:05:00','2008-12-23 7:15:00','2008-12-23 7:25:00','2008-12-23 7:35:00')";
        j=`expr $j + 1`;
done
sleep 3;
server=`mysql -uldirector -pxxxxxxxxxx  -h192.168.131.105 -e "use kingsoft;select count(*) from preference"`;
datetime=`date +%T`;
echo $datetime"----------"$server >> /tmp/mysql-cluster/mysql.log;


[root@localhost mysql-cluster]#
```

测试时间：

在192.168.8.48的cron中添加：

```
[root@localhost mysql-cluster]# crontab -e
*/3 * * * * sh /tmp/mysql-cluster/test.sh > /dev/null 2>&1
[root@localhost mysql-cluster]#
```

连续运行24小时。

测试结果：

火龙果•整理
uml.org.cn

```
#Cat mysql.log

14:31:54----------count(*) 21022

14:35:00----------count(*) 42634

14:37:57----------count(*) 63608

14:40:55----------count(*) 84708

14:43:55----------count(*) 105887

14:46:54----------count(*) 127045

14:49:58----------count(*) 148512

14:53:01----------count(*) 169795

14:56:27----------count(*) 190714

14:59:29----------count(*) 209921

15:02:03----------count(*) 231380

15:03:51----------count(*) 252231

15:05:12----------count(*) 269825

15:05:33----------count(*) 271824

15:08:05----------count(*) 291141

15:10:59----------count(*) 311836

15:14:00----------count(*) 332951

15:16:57----------count(*) 353841

15:19:59----------count(*) 374977

15:23:03----------count(*) 396181

15:26:01----------count(*) 417064

15:29:01----------count(*) 438098
```

火龙果•整理
uml.org.cn

15:32:03----------count(*) 459191

15:35:05----------count(*) 480229

15:38:05----------count(*) 501222

15:41:02----------count(*) 521868

15:43:59----------count(*) 542721

15:47:02----------count(*) 563841

16:00:32----------count(*) 698215

18:50:49----------count(*) 2105513

19:09:01----------count(*) 2105513

19:26:13----------count(*) 2105513

19:27:28----------count(*) 2105513

[root@localhost mysql-cluster]#

测试结果分析：

1）当逐渐增加负载，数据库的负载并不大，CPU 占用率为30%，而内存则由600MB 逐渐升至2GB，最终达到极限。

2）数据并发量大，并未引起数据库的异常，表明负载均衡已经解决了单台服务器负载太大的引起的瓶颈。

3）由于内存有限（2GB），当表中数据达到一定量以后，会出现表满现象。这种情况可以通过增加内存来解决。

4）mysql cluster 可以实现高可用性、负载均衡，并且通过优化参数使其进一步稳定服务。

5）可以采用6.3版本的 mysql cluster，来减小 NDBD 内存用量。

6）Mysql cluster 的性能一般，比 mysql replication 慢。

需要注意的问题：

火龙果•整理
uml.org.cn

1) 当 ndbd 第一次启动的时候或者 config.ini 更改的时候，需要加--initial 参数进行初始化。

2) 尽可能的不要人工干预系统，出现问题需要谨慎对待。

以下是一个外国人写的注意事项和优化：

1. Broken up into three parts. The MySQL servers sit separate from the NDB Storage Engine, which are storage nodes (NDB nodes). The third part is called a management server. The management server, oddly enough, isn’t required once the cluster is up and running unless you want to add another storage node.

2. Memory based storage engine. If you’re not using 5.1+ then you must have enough RAM in each storage node to store the data set. This means that if you have four machines with 4GB of RAM each you can store 8GB of data (16GB of total storage divided by two for two copies of the data set).

3. Storage nodes are static and pre-allocate resources on startup.

4. Supports transactions and row level locking.

5. Should be noted this is a storage engine so you can’t create MyISAM or InnoDB tables inside of a cluster.

6. Uses fixed sized records. This means if you have a varchar(255) and put a single byte into it that field is still using 255 bytes.

7. No foreign key constraint support.

8. Replication across nodes is syncronous across nodes. I assume this means that an INSERT happens once all of the nodes have completed the INSERT. This is different than regular replication which is asyncronous and introduces race conditions.

9. Tables are divided into fragments (one fragment for each storage node). Each storage node is responsible for each fragment. Each fragment also has a secondary

火龙果•整理
uml.org.cn

fragment, which is a copy of another node's primary fragment. This data distribution happens automatically.

10. NDB takes your primary key, creates a hash and then converts that to a fragment. So you'll have various rows on each different storage node.

11. A node group is a set of nodes that share the same fragment information. If you lose an entire node group you've lost half of the table and the cluster will not continue to operate. However, if one node in a node group fails there will still be enough data to keep the cluster up and running.

12. If a node fails and it's secondary counterpart takes over it will, essentially, have to perform the job of two nodes. Until a node has fully recovered it will not rejoin the cluster.

13. Backups are hot and non-locking. Each node writes its own set of backup files. No support for incremental backups.

14. Because it's memory based you could lose data on a system crash (as you might have transactions sitting in RAM when a crash occurs). The COMMIT command does not write changes to disk meaning that you could have data sitting in memory that's not on disk when a node crashes. This means the odd truth is that MySQL Clusters support syncronous replication, but are not atomic.

15. NDB nodes will checkpoint data to disk (data + logs), which are used for system recovery. They write two logs, the UNDO and REDO logs.

16. They recommend using TRUNCATE to delete all rows from a table.

17. Modification operations are distributed to both the primary and secondary fragments (obviously).

18. NDB will run on 64-bit machines. They recommend Dual CPU 64-bit machines. NDB is threaded. Application nodes (MySQL servers) can be whatever.

19. SCI offers 30-100% better performance over gigabit.

火龙果•整理
uml.org.cn

20. They actually recommend avoiding joins and to denormalize your schemas. Are you kidding me? He actually said “Performance for joins sucks.”

Overall, I’m underwhelmed by MySQL Clustering. You’re limited in storage with the RAM and you can’t optimize your schemas due to fixed field sizes. And any RDBMS “solution” that recommends you denormalize puts me off.

That being said the actual technology is pretty interesting and I suspect that in a few years we’ll see the clustering features in MySQL come into their own. As of now I suspect few people would be able to justify the sacrifices for the gains clustering allows.

(全文完)

参考资料：

http://www.ultramonkey.org/3/topologies/hc-ha-lb-eg.html
http://www.howtoforge.com/loadbalanced_mysql_cluster_debian
http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/LVS-HOWTO.install.html
http://dev.mysql.com/doc/refman/5.1/en/faqs-mysql-cluster.html#qandaitem-24-10-11
http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-multi-hardware-software-network.html
http://www.jedi.com/obiwan/technology/ultramonkey-rhel4.html
http://www.howtoforge.com/loadbalanced_mysql_cluster_debian_p8
http://zh.linuxvirtualserver.org/handbooks