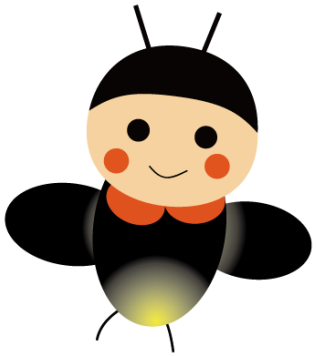


中华数据库
行业协会

2014中华数据库与运维安全大会

官方网址：www.zhdba.com

基于Drbd的数据库高可用

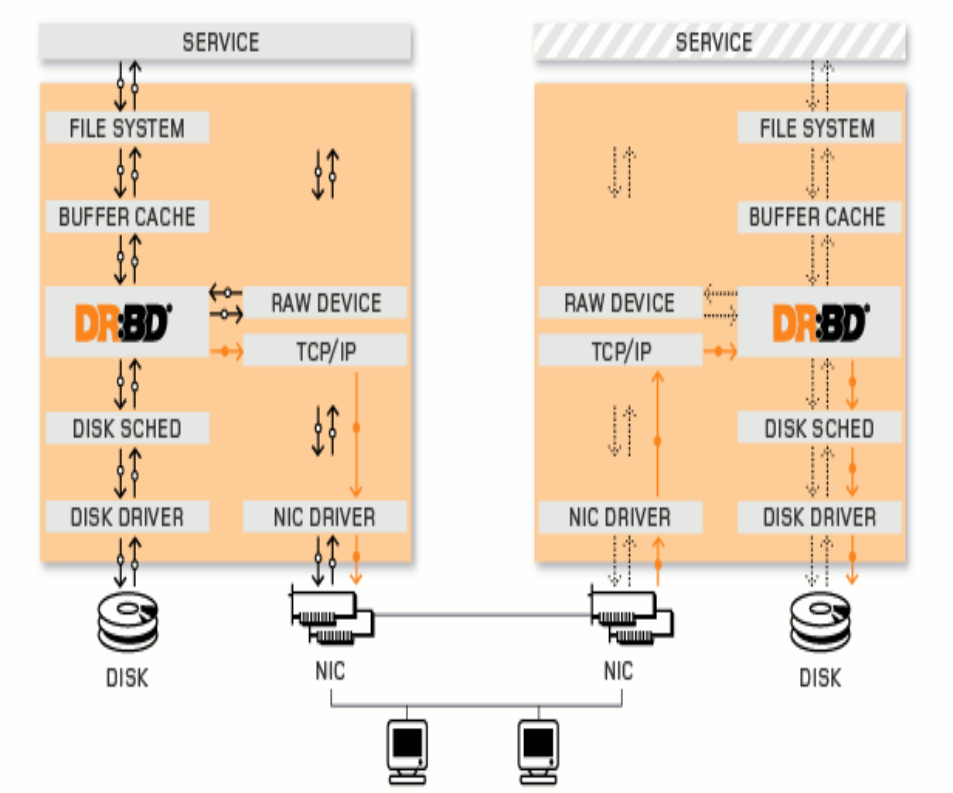
DRBD+COROSYNC+PACEMAKER+HAPROXY+KEEPALIVED实现HA和LB、并实现读写分离

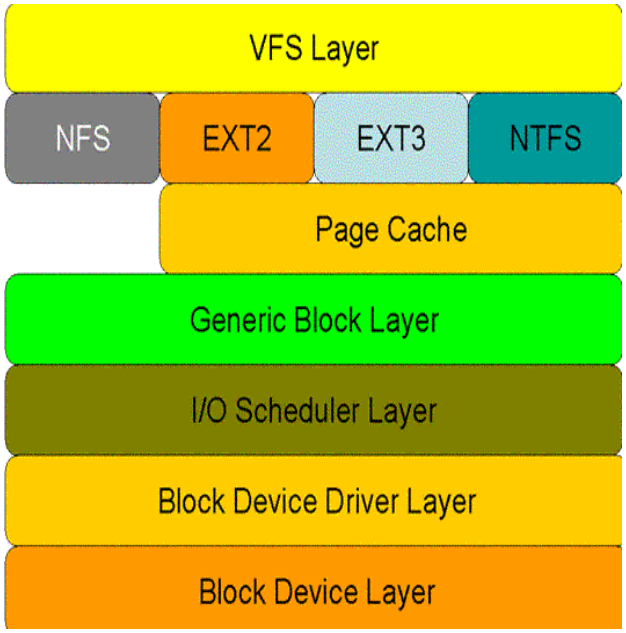
概要

- **一、DRBD介绍**
- 1、什么是DRBD
- 2、DRBD基础介绍以及特性
- 3、DRBD的复制模式
- 4、DRBD脑裂处理
- 5、DRBD调优
- **二、PACEMAKER和COROSYNC集群软件介绍**
- 1、 PACEMAKER重要参数配置
- 2、什么是colocation和order
- **三、基于drbd的高可用案列**
- 1、原理介绍以及适用场景
- 2、如何实现读写分离和负载均衡
- 3、故障转移如何实现以及如何避免单点故障
- 4、架构的优缺点以及如何改进

什么是DRBD?

- file system->buffer cache ->drbd->disk scheduler->disk drivers





DRBD基础介绍以及特性

- 基础介绍：
- 定义一个资源：

```
resource  clusterdb_res {
  device         /dev/drbd0;
  disk           /dev/sda3;
  meta-disk      internal;
  on mysql03 {
```

- 资源包括：resouce name, Volumes, DRBD device
- Drbd角色：primary<->secondary

特性：

- 支持end-to-end数据完整性验证
- 支持在线设备验证
- 磁盘IO错误处理策略
- 过期数据处理策略
- 暂停复制
- 脑裂通知和自动恢复

DRBD基础介绍以及特性(2)

- 支持end-to-end数据完整性验证

此特性针对在复制过程中由于网络传输原因导致的数据不一致。DRBD对每个要复制的块生成一个校验和，用来对peer端数据进行完整性校验，如果接收到的块的校验和与source端的校验和不一致，将会要求重传。此特性在我们的生产环境中出现了很大问题要慎用。

```
resource <resource>
  net {
    data-integrity-alg <algorithm>;
  }
  ...
}
```

- 支持在线设备验证（online device ）

如果我们不在传输过程中对数据进行校验，我们仍然可以采用在线设备验证的方式，原理同上，我们可以采用定时任务周期性的对 数据进行验证

- 磁盘IO错误处理策略

磁盘出现IO错误时候，我们应该采用何种策略呢？DRBD提供三种策略，分别是：detach 、pass_on 、call-local-io-error .detach的情况下,节点将会处于diskless状态，所有的对节点的读写将会从对端节点进行，这种情况下虽然性能有所下降，但是仍然可以提供服务，很明显在高可用的情况下，这个策略使我们的首选。

DRBD基础介绍以及特性(3)

- 过期数据处理策略

过期数据不是不一致性数据，只是说secondary不再与priamry同步数据了，secondary相当于是一个snapshot，这时候如果发生切换，那么可想而知，数据的一致性就会出现问题，我们需要通过某些策略来防止这种情况的发生：当出现过期数据的时候，drbd的连接状态将会由connect变为Wfconnection,这时候Pacemaker不会允许过期数据的节点提升为primary

```
resource <resource> {
  disk {
    fencing resource-only;
    ...
  }
  handlers {
    fence-peer "/usr/lib/drbd/crm-fence-peer.sh";
    after-resync-target "/usr/lib/drbd/crm-unfence-peer.sh";
    ...
  }
  ...
}
```

- 暂停复制

对于一些网络状态不好的情况，如果我们采用协议C进行复制，那么数据复制延时将会很严重，这时候我们可以采用暂停复制的策略，这样当网络状况不好的时候，primary端将会暂停复制，primary和secondary将会处于链式的不同步状态，当带宽变为可用的时候，复制将会继续进行。

DRBD复制模式

- 协议A

  Asynchronous replication protocol. Local write operations on the primary node are considered completed as soon as the local disk write has finished, and the replication packet has been placed in the local TCP send buffer

- 协议B

  Memory synchronous (semi-synchronous) replication protocol. Local write operations on the primary node are considered completed as soon as the local disk write has occurred, and the replication packet has reached the peer node

- 协议C

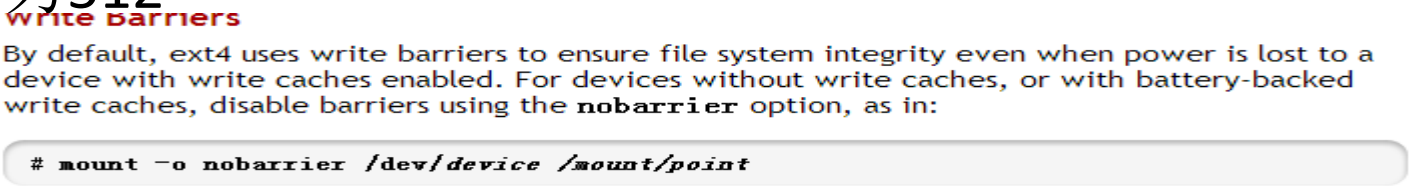
  Synchronous replication protocol. Local write operations on the primary node are considered completed only after both the local and the remote disk write have been confirmed

DRBD脑裂处理

- 自动处理：
- 1、Discarding modifications made on the younger primary
- --后切换成primary角色的节点数据将会被丢弃
- 2、Discarding modifications made on the older primary
- ---先切换成primary角色的节点数据将会被丢弃
- 3、Discarding modifications on the primary with fewer changes –哪个节点更改的数据少就丢弃
- 4、Graceful recovery from split brain if one host has had no intermediate changes
- --如果有节点根本没有更新数据，就直接恢复
- 手动处理

DRBD调优

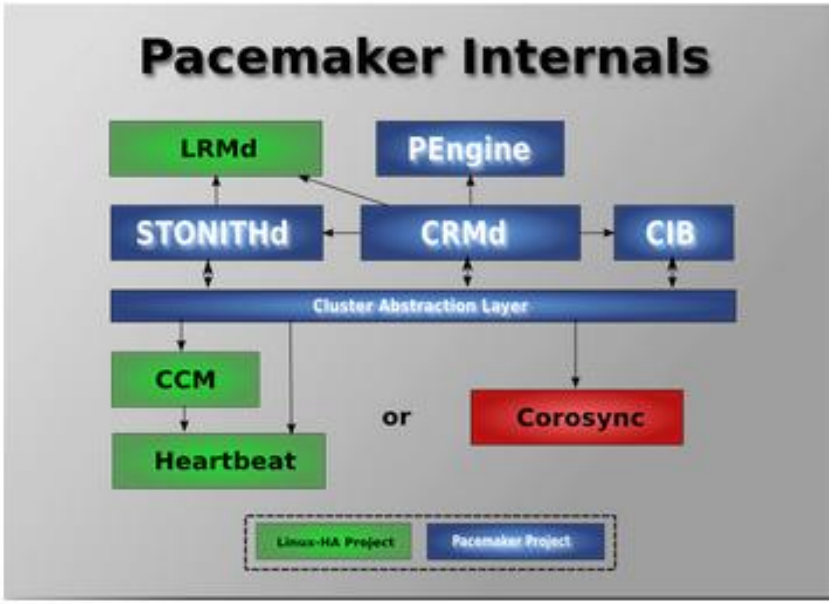
- Max-buffers：DRBD为写磁盘分配的缓冲区数量，默认是2048，对于RAID阵列建议调整为8000
- Max-epoch-size：默认2048，建议修改为8000
- sndbuf-size:tcp发送缓冲去，默认是128，千兆网卡可以设置为512

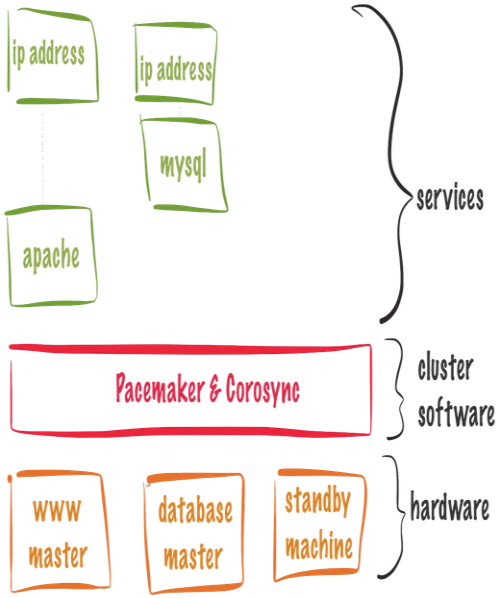


- 如果raid卡提供写保护，我们可以关闭，这样可以提高性能
- disk-barrier no; disk-flushes no;

PACEMAKER和COROSYNC介绍

- 什么是pacemaker:
- 通过corosync或者heartbeat提供的资源信息，pacemaker通过resource agent来管理资源，比如start、stop





PACEMAKER重要参数配置

- Pacemaker集群选项：
- no-quorum-policy：
- What to do when the cluster does not have quorum. Allowed values:
- * ignore - continue all resource management
- * stop - stop all resources in the affected cluster partition
- 当集群已经达不到法定票数的时候怎么办呢？对于我们的drbd来说就只有两个节点，如果其中的一个节点（secondary）出现问题，那么就只剩下primary了，这时候就只有1票了，已经不能形成法定票数，如果no-quorum-policy采用默认值，那么此时primary节点将不能提供服务了，这肯定不是我们希望看到的，所以我们要修改此参数值为ignore，这样即使只有一个节点仍然可以提供服务
- stonith-enabled:
- 我们不使用stonith设备，split brain问题我们通过其他方式来解决

PACEMAKER重要参数配置

- Pacemaker资源选项

在DRBD中有两个节点A、B，如果A节点出现问题了，这时候资源会switch到B节点，那么当A节点重新加入集群后，是否希望资源再切换回A节点呢？，由下面的参数来控制：

resource-stickiness	Calculated	How much does the resource prefer to stay where it is? Defaults to the value of resource-stickiness in the rsc_defaults section

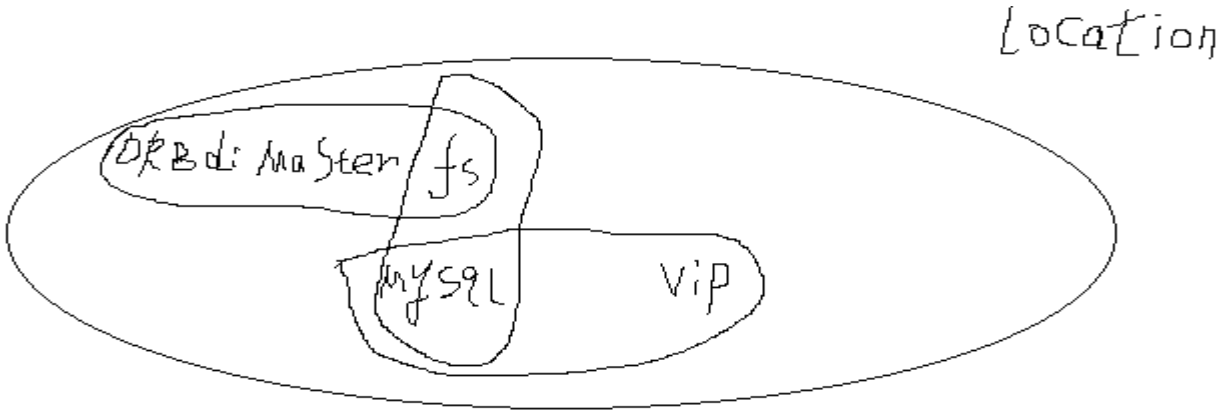
- 如果我们不希望资源来回切换可以设置这个值为100，值越大表明越希望停留在当前节点

什么是COLOCATION和ORDER

- 在DRBD中，资源之间是有些约束条件的，这些约束条件我们分为两类：colocation和order
- Colocation:资源之间的位置关系
- Order: 资源之间的启动，停止等的顺序关系
- DRBD中需要定义的colocation和order为：
- 1、DRBD和文件之间的依赖关系：
- colocation myfs_with_ms_mysqldrbd inf: p_fs_mysql ms_drbd_mysql:Master
- order mysqldrbd_before_myfs mandatory: ms_drbd_mysql:promote p_fs_mysql:start
- 2、文件系统和MYSQL的依赖关系：
- colocation mysql_with_myfs inf: p_mysql p_fs_mysql
- order myfs_before_mysql mandatory: p_fs_mysql:start p_mysql:start
- 3、vip和mysql的依赖关系：
- colocation myvip_with_mysql inf: p_ip_mysql p_mysql
- order myvip_before_mysql mandatory: p_mysql p_ip_mysql

什么是COLOCATION和ORDER(2)

- LOCATION:



- Order:

  drbd:promote→ fs:start→mysql:start→vip

基于drbd的高可用案列

- 原理介绍：
- 根据CAP理论，我们的架构实现CA
- 一致性:

DRBD通过协议 C保证数据的一致性,任何数据必须 成功写入（write-back仍然有可能丢失数据）对端磁盘本地才能开始写入

- 可用性:

通过pacemaker监控整个集群状态在节点出现问题的时候，资源会重新分配并按照定义的规则来启动，不需要人工参与，保证了数据库的高可用

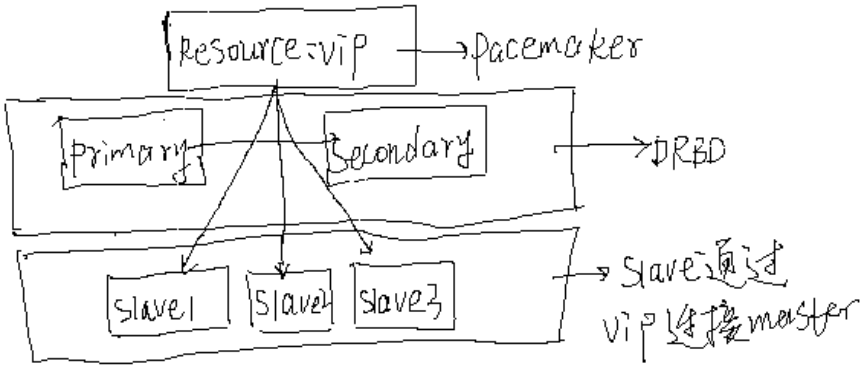
基于drbd的高可用案列

- **适用场景：**
- 对数据的一致性要求较高
- 能接受5-10S的切换时间，切换过程中SLAVE不受影响对带宽稳定性要求较高
- 数据库参数：innodb-flush-log-at-trx-commit = 1、 sync-binlog = 1，数据库将不能使用组提交
- 并不适应于并发在3000以上的业务环境

如何实现读写分离和负载均衡

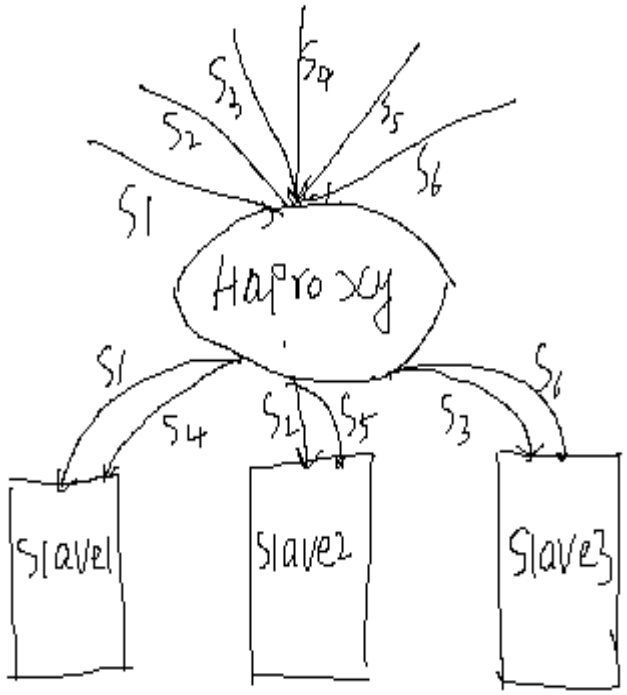
- 读写分离：

由于我们在DRBD上采用协议C来进行复制，所以任何时刻两个节点的数据都是一直的，我们通过在DRBD的基础上配置Master-Slave来实现读写分离

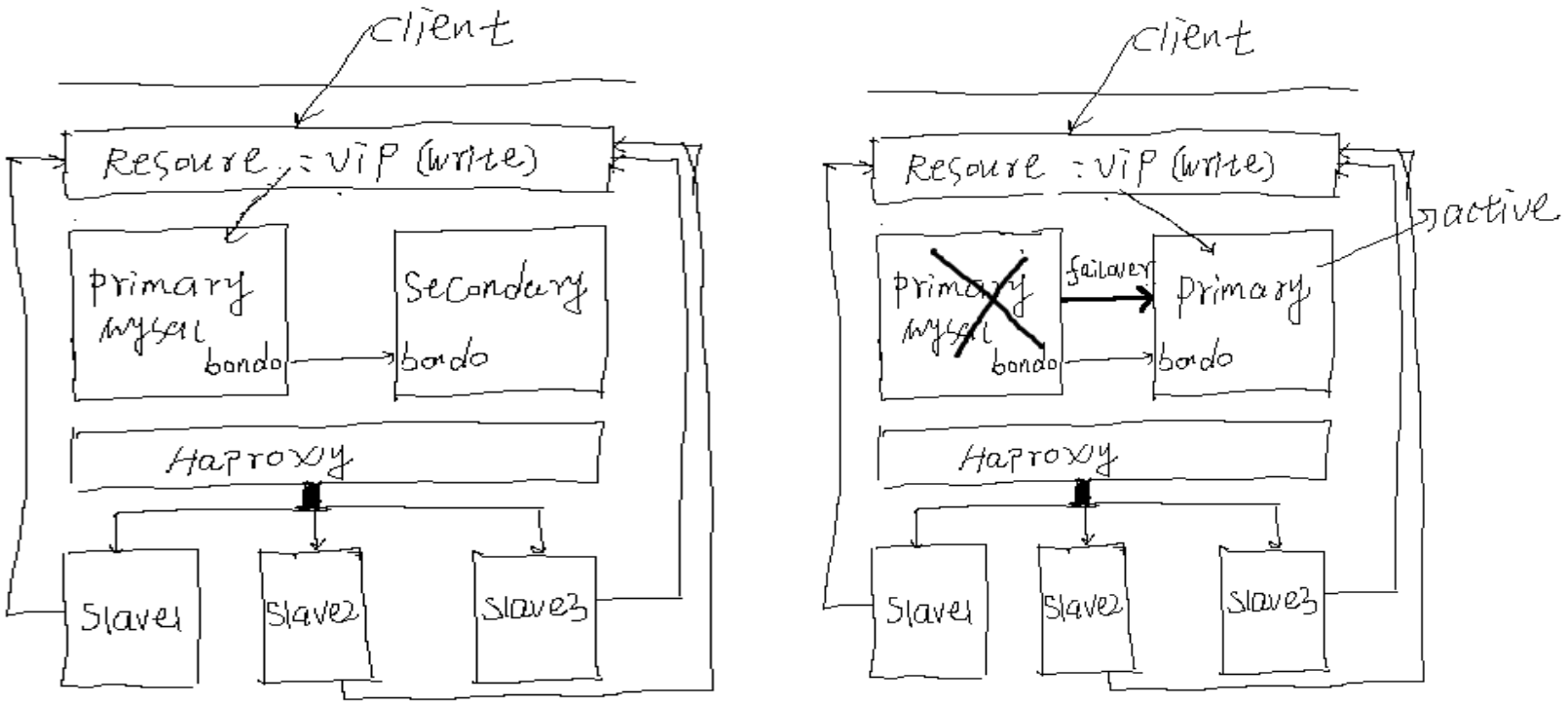


如何实现读写分离和负载均衡

- 负载均衡



如何实现读写分离和负载均衡



故障转移如何实现以及如何避免单点故障

- Pacemaker是集群管理软件
- PACEMAKER会根据配置规则启动和关闭MYSQL以及进行自动的切换
- 启动顺序:drbd->fs->mysql->ip
- Pacemaker会监控MYSQL的状态，出现问题是会实现自动切换
- primitive p_mysql lsb:mysqld \
- op start timeout="120s" interval="0" \
- op stop timeout="120s" interval="0" \
- op monitor interval="20" timeout="20" on-fail="restart"

架构的优缺点以及如何改进

在这个集群架构中我们没有有效的脑裂处理机制，我们没使用专门的fence设备，这样可能在出现脑裂时数据不一致，我们采用了下面的方法来尽量减少脑裂的产生，但是这并没有实际上解决问题，我们会在后续考虑其他方案。

- 我们采用下面的方式来减少脑裂发生的概率：

(1)、DRBD：手动处理脑裂，并且在发生脑裂的时候disconnect

(2)、Server:通过使用单独的网卡，使用双网卡绑定,通过直连线进行心跳检测

(3)、corosync之间的心跳检测和drbd之间的数据传输采用不同的网卡，这样即使corosync之间的心跳出现问题，只要DRBD之间的网络连接正常，仍然能够保证数据的一致性,不过这时候失效的节点会一直重启服务

架构的优缺点以及如何改进

- 改进方法：
- 增加slave节点到集群中，这样就存在至少三个节点，即使其中的任何一个节点失效，集群仍然可以提供服务，但是这时候要考虑的问题是，所有资源:DRBD\FS\MYSQL\VIP都不能运行在SLAVE节点上，SLAVE节点只是为了防止脑裂发生，我们使用location来限制资源是否可以运行在某个节点上：
- location p_drbd_mysql_loc p_drbd_mysql -inf: mysql02
- 其中mysql02（是slave服务器的）新增加的节点，由于我们已经指定了资源的colocation，所以其他所有的资源都不会在mysql02这个节点上运行了

案例—Drbd配置详解

```
[root@mysql03 drbd.d]# vim global_common.conf

global {
        usage-count yes;
}

common  {
        protocol C;

        handlers {
                pri-on-incon-degr "/usr/lib/drbd/notify-pri-on-incon-
                pri-lost-after-sb "/usr/lib/drbd/notify-pri-lost-afte
                local-io-error "/usr/lib/drbd/notify-io-error.sh; /us
                fence-peer "/usr/lib/drbd/crm-fence-peer.sh";
                #split-brain "/usr/lib/drbd/notify-split-brain.sh roo
                #out-of-sync "/usr/lib/drbd/notify-out-of-sync.sh roo
                after-resync-target "/usr/lib/drbd/crm-unfence-peer.s
                 }

        startup {
        }

        disk {
                on-io-error detach;
                fencing resource-only;
        }

        net {
                cram-hmac-alg sha1;
                sndbuf-size 512k;
                shared-secret [redacted]
                #data-integrity-alg sha1;
                after-sb-0pri disconnect;
                after-sb-1pri disconnect;
                after-sb-2pri disconnect;
       }
       syncer {
                #on-no-data-accessible io-error;
                verify-alg sha1;
                rate 1000M;
       }
}
```

```
[root@mysql03 drbd.d]# vim clusterdb_res.res

resource  clusterdb_res {
  device         /dev/drbd0;
  disk           /dev/sda3;
  meta-disk      internal;
  on mysql03 {
    address      [redacted]
  }
  on mysql01 {
    address      [redacted]
  }
}
~
~
~
```

案例—crm配置详解

```
node mysql01 \
        attributes standby="off"
node mysql03 \
        attributes standby="off"
primitive p_drbd_mysql ocf:linbit:drbd \
        params drbd_resource="clusterdb_res" \
        op monitor interval="29s" role="Master" \
        op monitor interval="31s" role="Slave"
primitive p_fs_mysql ocf:heartbeat:Filesystem \
        params device="/dev/drbd0" directory="/app/mysqldata" fstype="ext4" \
        op monitor interval="40" timeout="40" \
        op start timeout="60" interval="0" \
        op stop timeout="60" interval="0" \
        meta target-role="Started"
primitive p_ip_mysql ocf:heartbeat:IPaddr2 \
        params ip="[redacted]" nic="bond0" \
        op monitor interval="20" timeout="20"
primitive p_mysql lsb:mysqld \
        op start timeout="120s" interval="0" \
        op stop timeout="120s" interval="0" \
        op monitor interval="20" timeout="20" on-fail="restart" \
        meta target-role="Started"
ms ms_drbd_mysql p_drbd_mysql \
        meta master-max="1" master-node-max="1" clone-max="2" clone-node-max="1" notify="true" target-role="Started"
colocation myfs_with_ms_mysqldrbd inf: p_fs_mysql ms_drbd_mysql:Master
colocation mysql_with_myfs inf: p_mysql p_fs_mysql
colocation myvip_with_mysql inf: p_ip_mysql p_mysql
order myfs_before_mysql inf: p_fs_mysql:start p_mysql:start
order mysqldrbd_before_myfs inf: ms_drbd_mysql:promote p_fs_mysql:start
order myvip_before_mysql inf: p_mysql p_ip_mysql
property $id="cib-bootstrap-options" \
        expected-quorum-votes="2" \
        dc-version="1.1.10-14.el6_5.2-368c726" \
        cluster-infrastructure="classic openais (with plugin)" \
        stonith-enabled="false" \
        no-quorum-policy="ignore" \
        last-lrm-refresh="1393220267"
rsc_defaults $id="rsc-options" \
```

案例—Haproxy配置(1)

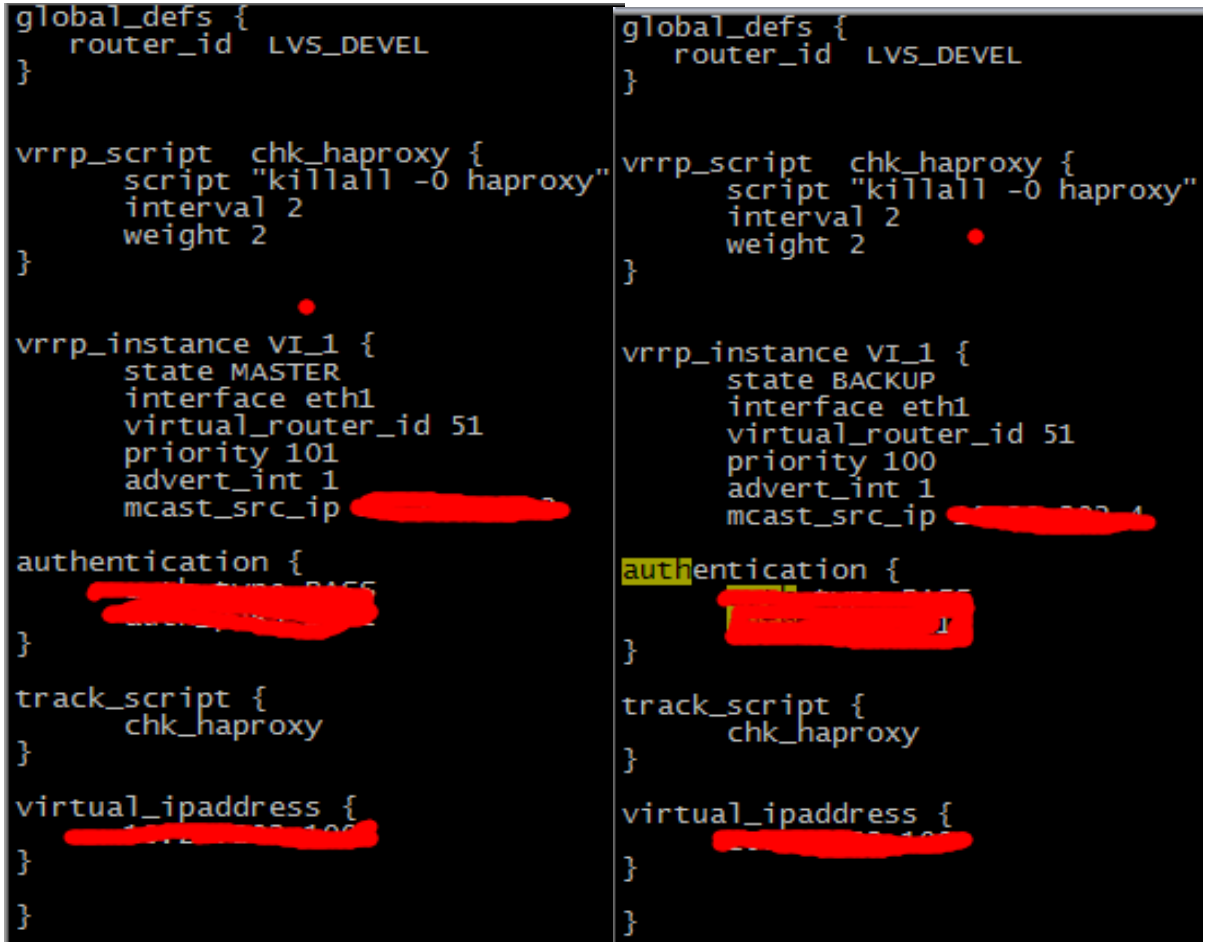
```
defaults
log global
mode http
option tcplog
option dontlognull
retries 3
option redispatch
maxconn 2000
contimeout 5000
clitimeout 50000
srvtimeout 50000


frontend stats-front
bind *:80
mode http
default_backend stats-back

frontend proxy-front
bind *:[redacted]
mode tcp
default_backend proxy-back

backend proxy-back
mode tcp
balance leastconn
option httpchk
server slave-1 [redacted] check port [redacted] inter 12000 rise 3 fall 5



backend stats-back
mode http
balance roundrobin
stats uri /haproxy/stats
stats auth [redacted]
```

案例—Haproxy配置(2)

```
[root@mysql02 xinetd.d]# vim mysqlchk

# default: on
# description: mysqlchk
service mysqlchk
{
# this is a config for xinetd, place it in /etc/xinetd.d/
   flags = REUSE
   socket_type = stream
   port = 9200
   wait = no
   user = nobody
   server = /usr/local/bin/mysqlchk_status.sh
   log_on_failure += USERID
   only_from = 10.10.10.0
   # recommended to put the IPs that need
   # to connect exclusively (security purposes)
   per_source = UNLIMITED
}
```

```
SLAVE_STATUS=`/usr/local/bin/mysql --host=$MYSQL_HOST --port=$MYSQL_PORT --user=$MYSQL_USERNAME     -e "show slave status\G;"
|egrep 'Slave_IO_Running|Slave_SQL_Running:|Seconds_Behind_Master'|awk -F ':' '{print $2}' 2>/dev/null`

IO_THREAD_STATUS=`echo $SLAVE_STATUS|awk   '{print  $1}'`
SQL_THREA_STATUS=`echo $SLAVE_STATUS|awk   '{print  $2}'`
LAG=`echo $SLAVE_STATUS|awk   '{print  $3}'`



#
# Check the output. If it is not empty then everything is fine and we return
# something. Else, we just do not return anything.
#
if [  "$IO_THREAD_STATUS" = "Yes" -a  "$SQL_THREA_STATUS" = "Yes" -a "$LAG" -le  1800  ]
then
        echo -en "HTTP/1.1 200 OK\r\n"
        echo -en "Content-Type: text/plain\r\n"
        echo -en "Connection: close\r\n"
        echo -en "Content-Length: 40\r\n"
        echo -en "\r\n"
        echo -en "slave is ok .\r\n"
        exit 0
else
        echo -en "HTTP/1.1 503 Service Unavailable\r\n"
        echo -en "Content-Type: text/plain\r\n"
        echo -en "Connection: close\r\n"
        echo -en "Content-Length: 44\r\n"
        echo -en "\r\n"
        echo -en "slave  is not ok.\r\n"
        exit 1
fi
```

KEEPALIVED配置

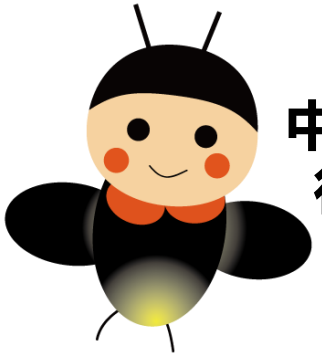


```
global_defs {
   router_id  LVS_DEVEL
}


vrrp_script  chk_haproxy {
      script "killall -0 haproxy"
      interval 2
      weight 2
}


vrrp_instance VI_1 {
      state MASTER
      interface eth1
      virtual_router_id 51
      priority 101
      advert_int 1
      mcast_src_ip [redacted]

authentication {
      [redacted]
}

track_script {
      chk_haproxy
}

virtual_ipaddress {
      [redacted]
}

}
```

```
global_defs {
   router_id  LVS_DEVEL
}


vrrp_script  chk_haproxy {
      script "killall -0 haproxy"
      interval 2
      weight 2
}


vrrp_instance VI_1 {
      state BACKUP
      interface eth1
      virtual_router_id 51
      priority 100
      advert_int 1
      mcast_src_ip [redacted]

authentication {
      [redacted]
}

track_script {
      chk_haproxy
}

virtual_ipaddress {
      [redacted]
}

}
```

2014年11月中华架构师大会预告

演讲主题	演讲嘉宾	公司名称	职位/职称
待定	朱超	360	中间件研发负责人
TFS技术架构及运维	张友东	阿里云	TFS研发负责人
待定	黄俊	国药集团	常务副总经理
golang实时消息推送架构实战	毛剑	金山网络	移动游戏技术经理
MyCAT之前世今生	吴治辉	惠普中国	系统架构师
雪球的架构实践	王栋	雪球财经	CTO
待定	刘建平	热璞科技	技术总监



中华数据库
行业协会

中华数据库行业协会

官方网站：www.zhdba.com

官方微信平台：zhdba2014

官方微博：中华数据库行业协会ZHDBA

技术交流QQ群：91596001