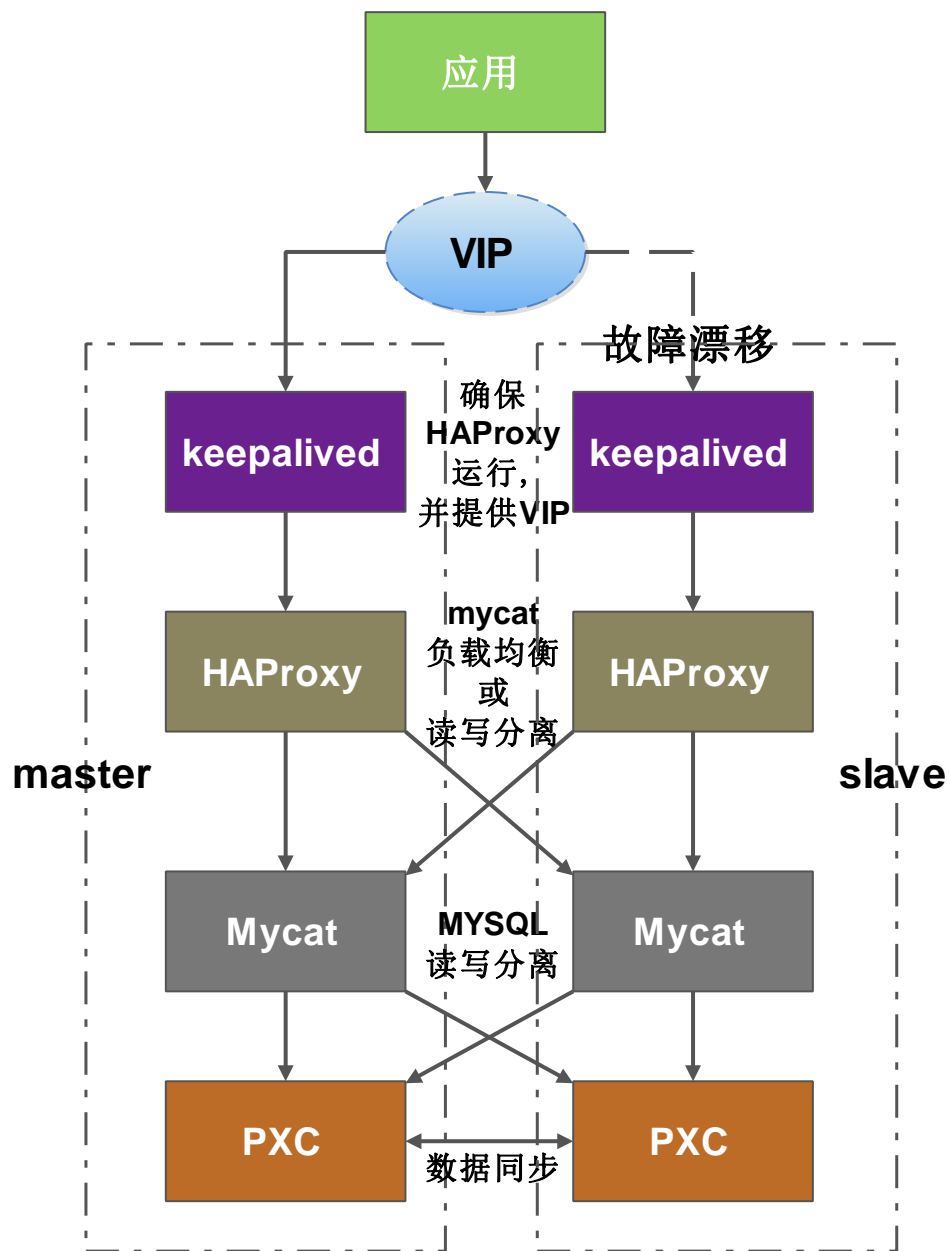


目录

1. 集群架构图.....	2
2. Percona XtraDB Cluster 集群	2
2.1 docker 镜像集群	2
3. Mycat 集群	4
3.1 下载	4
3.2 解压缩	4
3.3 修改 server.xml 配置文件	4
3.4 修改 schema.xml 配置文件	5
3.5 启动 mycat	11
3.6 连接 mycat 测试	11
3.7 启动另一个 mycat	11
4. Haproxy 集群	11
4.1 创建 haproxy.cfg 配置文件	11
4.2 启动 HAProxy 镜像	12
4.3 访问监控页面	13
4.4 测试	13
5. keepalived 集群	13
5.1 安装	13
5.2 解压	13
5.3 安装	13
5.4 复制或者连接文件到指定目录	14
5.5 创建启动项	14
5.6 修改 keepalived.conf 配置文件	14
5.7 设置更新虚拟服务器（VIP）地址的 arp 记录到网关脚本	16
5.8 设置 HAProxy 服务监控脚本	17
5.9 内核优化	18
5.10 启动	19
5.11 安装配置并启动另一台机器的 keepalived	19
5.12 测试	19
6. 问题和解决方案	19
6.1 Percona XtraDB Cluster 问题	20
6.2 keepalived 安装问题（返回安装流程）	20

1. 集群架构图



2. Percona XtraDB Cluster 集群

2.1 docker 镜像集群

(1) 创建存放持久化数据的目录

```
mkdir mysql1  
mkdir mysql2  
chmod 777 mysql1  
chmod 777 mysql2
```

(2) 创建网络 pxc-net

```
docker network create -d bridge --subnet=10.10.1.0/24
--gateway=10.10.1.254 --attachable=true pxc-net
```

(3) 创建结点 1

```
docker run -d -p 3307:3306 -e MYSQL_ROOT_PASSWORD=654321
-e CLUSTER_NAME=PXC -e XTRABACKUP_PASSWORD=123456 -v
/data/mysql1:/var/lib/mysql --privileged --name=node1 --
net=pxc-net percona/percona-xtradb-cluster
```

注：mysql 的 root 用户密码为 654321，xtrabackup 用户密码为 123456，结点 1 的名称为 node1
复制时，注意--net=pxc-net 前的两个横线符号

(4) 进入结点 1 容器执行 bash 命令

```
docker exec -it node1 bash
```

(5) 进入 mysql

```
mysql -uroot -p
```

注：在容器内进入 mysql

(6) 授权

```
grant reload,lock tables,replication client,process on *.* to
'xtrabackup'@'localhost' IDENTIFIED BY '123456' WITH GRANT
OPTION;
flush privileges;
```

(7) 退出 mysql

```
exit
exit
```

(8) 创建结点 2

```
docker run -d -p 3308:3306 -e MYSQL_ROOT_PASSWORD=654321
-e CLUSTER_NAME=PXC -e XTRABACKUP_PASSWORD=123456 -e
CLUSTER_JOIN=node1 -v /data/mysql2:/var/lib/mysql --
privileged --name=node2 --net=pxc-net percona/percona-xtradb-
cluster
```

注：CLUSTER_JOIN 一定要填写结点 1 的容器名
复制时，注意--privileged 前的两个横线符号

(9) 测试

往其中一个数据库插入数据，另一个自动同步。

3. Mycat 集群

3.1 下载

下载地址: <http://www.mycat.io/>

或者运行命令: `wget http://dl.mycat.io/1.6.6.1/Mycat-server-1.6.6.1-release-20181031195535-linux.tar.gz`

注: 使用 mycat 需要先安装和配置 JDK

这里启动的两个 mycat 分别在两台机器上, 如果想要在一台机器上启动两个 mycat, 需要修改 `server.xml` 和 `wrapper.conf` 中的端口。

`server.xml`:

```
<property name="serverPort">8067</property> <property name="managerPort">9067</property>
<!-- <property name="id1.timeout">300000</property> <property name="binds">0.0.0.0</property>
<property name="frontwriteQueueSize">4096</property> <property name="processors">32</property> -->
<!-- 分布式事务开关, 0为不过滤分布式事务, 1为过滤分布式事务 (如果分布式事务内只涉及全局表, 则不过滤), 2为不过
```

`wrapper.conf`

```
wrapper.java.additional.6=-Dcom.sun.management.jmxremote
wrapper.java.additional.7=-Dcom.sun.management.jmxremote.port=1985
wrapper.java.additional.8=-Dcom.sun.management.jmxremote.authenticate=false
```

3.2 解压缩

`tar -xf Mycat-server-1.6.6.1-release-20181031195535-linux.tar.gz`

3.3 修改 `server.xml` 配置文件

`vi mycat/conf/server.xml`

```
<user name="root">
  <property name="password">123456</property>
  <property name="schemas">DB1</property>

  <!-- 表级 DML 权限设置 -->
  <!--
  <privileges check="false">
    <schema name="TESTDB" dml="0110" >
      <table name="tb01" dml="0000"></table>
      <table name="tb02" dml="1111"></table>
    </schema>
  </privileges>
  -->
</user>

<user name="user">
  <property name="password">123456</property>
  <property name="schemas">DB1</property>
  <property name="readOnly">true</property>
</user>
```

`<user name="root">` 为用户名

`<property name="password">` 为用户密码

`<property name="schemas">` 为 `server.xml` 同级目录下的 `schema.xml` 中定义的 `<schema name="cloudsystem" checkSQLschema="true" sqlMaxLimit="100">` 对应, 表现为 mycat 数据库名。

```

2.      <property name="serverPort">8066</property> <property name="managerPort">9066</
property>
3.      <property name="idleTimeout">300000</property> <property name="bindIp">0.0.0.0</p
roperty>
4.      <property name="frontWriteQueueSize">4096</property> <property name="processors">
32</property> -->
5.      <!--分布式事务开关，0 为不过滤分布式事务，1 为过滤分布式事务（如果分布式事务内
只涉及全局表，则不过滤），2 为不过滤分布式事务,但是记录分布式事务日志-->
6.      <property name="handleDistributedTransactions">0</property>

```

修改端口或分布式事务

需要复制，请用 word 文档打开，然后复制框内的内容

```

<!--
      <property          name="serverPort">8066</property>          <property
name="managerPort">9066</property>
      <property          name="idleTimeout">300000</property>      <property
name="bindIp">0.0.0.0</property>
      <property          name="frontWriteQueueSize">4096</property>  <property
name="processors">32</property> -->
      <!--分布式事务开关，0 为不过滤分布式事务，1 为过滤分布式事务（如果
分布式事务内只涉及全局表，则不过滤），2 为不过滤分布      式 事 务,但 是
记录分布式事务日志-->
      <property name="handleDistributedTransactions">0</property>

```

3.4 修改 schema.xml 配置文件

vi mycat/conf/schema.xml

```

1.  <?xml version="1.0"?>
2.  <!DOCTYPE mycat:schema SYSTEM "schema.dtd">
3.  <mycat:schema xmlns:mycat="http://io.mycat/">
4.
5.      <schema name="cloudsystem" checkSQLschema="true" sqlMaxLimit="100">
6.          <table name="art_article" primaryKey="ID" dataNode="dn1" />
7.          <table name="art_article_sku" primaryKey="ID" dataNode="dn1" />
8.          <table name="art_article_sku_attament" primaryKey="ID" dataNode="
dn1" />
9.          <table name="art_article_sku_reject_log" primaryKey="ID" dataNode
="dn1" />
10.         <table name="art_category" primaryKey="ID" dataNode="dn1" />
11.         <table name="art_damage" primaryKey="ID" dataNode="dn1" />
12.         <table name="art_damage_attament" primaryKey="ID" dataNode="dn1"
/>

```

```
13.      <table name="cms_advertisement" primaryKey="ID" dataNode="dn1" />
14.      <table name="cms_article" primaryKey="ID" dataNode="dn1" />
15.      <table name="cms_article_info" primaryKey="ID" dataNode="dn1" />
16.      <table name="cms_article_like" primaryKey="ID" dataNode="dn1" />
17.      <table name="cms_attament" primaryKey="ID" dataNode="dn1" />
18.      <table name="cms_category" primaryKey="ID" dataNode="dn1" />
19.      <table name="gue_category" primaryKey="ID" dataNode="dn1" />
20.      <table name="gue_member_option" primaryKey="ID" dataNode="dn1" />
21.      <table name="gue_option" primaryKey="ID" dataNode="dn1" />
22.      <table name="gue_question" primaryKey="ID" dataNode="dn1" />
23.      <table name="mem_account" primaryKey="ID" dataNode="dn1" />
24.      <table name="mem_account_log" primaryKey="ID" dataNode="dn1" />
25.      <table name="mem_address" primaryKey="ID" dataNode="dn1" />
26.      <table name="mem_attament" primaryKey="ID" dataNode="dn1" />
27.      <table name="mem_check" primaryKey="ID" dataNode="dn1" />
28.      <table name="mem_check_config" primaryKey="ID" dataNode="dn1" />
29.      <table name="mem_coupon" primaryKey="ID" dataNode="dn1" />
30.      <table name="mem_coupon_member" primaryKey="ID" dataNode="dn1" />
31.      <table name="mem_dept" primaryKey="ID" dataNode="dn1" />
32.      <table name="mem_exception_job" primaryKey="ID" dataNode="dn1" />
33.      <table name="mem_flow" primaryKey="ID" dataNode="dn1" />
34.      <table name="mem_goods_member" primaryKey="ID" dataNode="dn1" />
35.      <table name="mem_job" primaryKey="ID" dataNode="dn1" />
36.      <table name="mem_job_record" primaryKey="ID" dataNode="dn1" />
37.      <table name="mem_job_step" primaryKey="ID" dataNode="dn1" />
38.      <table name="mem_member" primaryKey="ID" dataNode="dn1" />
39.      <table name="mem_member_article" primaryKey="ID" dataNode="dn1" /
    >
40.      <table name="mem_member_role" primaryKey="ID" dataNode="dn1" />
41.      <table name="mem_point" primaryKey="ID" dataNode="dn1" />
42.      <table name="mem_point_log" primaryKey="ID" dataNode="dn1" />
43.      <table name="mem_role" primaryKey="ID" dataNode="dn1" />
44.      <table name="pro_attament" primaryKey="ID" dataNode="dn1" />
45.      <table name="pro_category" primaryKey="ID" dataNode="dn1" />
46.      <table name="pro_goods" primaryKey="ID" dataNode="dn1" />
47.      <table name="pro_goods_info" primaryKey="ID" dataNode="dn1" />
```

```

48.      <table name="pro_goods_property" primaryKey="ID" dataNode="dn1" /
>
49.      <table name="pro_goods_sku" primaryKey="ID" dataNode="dn1" />
50.      <table name="pro_order" primaryKey="ID" dataNode="dn1" />
51.      <table name="pro_order_refund" primaryKey="ID" dataNode="dn1" />

52.      <table name="pro_property" primaryKey="ID" dataNode="dn1" />
53.      <table name="pro_shop" primaryKey="ID" dataNode="dn1" />
54.      <table name="system_attament" primaryKey="ID" dataNode="dn1" />
55.      <table name="system_dept" primaryKey="ID" dataNode="dn1" />
56.      <table name="system_log" primaryKey="ID" dataNode="dn1" />
57.      <table name="system_resource" primaryKey="ID" dataNode="dn1" />
58.      <table name="system_role" primaryKey="ID" dataNode="dn1" />
59.      <table name="system_role_resource" primaryKey="ID" dataNode="dn1"
/>
60.      <table name="system_user" primaryKey="ID" dataNode="dn1" />
61.      <table name="system_user_role" primaryKey="ID" dataNode="dn1" />

62.  </schema>
63.  <dataNode name="dn1" dataHost="localhost1" database="cloudsystem" />

64.  <dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"
65.      writeType="0" dbType="mysql" dbDriver="native" switchType="
3" slaveThreshold="100">
66.      <heartbeat>show status like 'wsrep%</heartbeat>
67.
68.      <writeHost host="hostM1" url="123.207.77.109:3307" user="root"
69.          password="654321" />
70.      <writeHost host="hostM2" url="123.207.77.109:3308" user="root"
71.          password="654321" />
72.
73.  </dataHost>
74. </mycat:schema>

```

注：

<schema name="DB1">与 server.xml 对应

<table name="test">为 mycat 的表名

<dataNode name="pxc">为数据库分片

<dataHost name="datahost1">直接定义了具体的数据库实例、读写分离配置和心跳语句，balance="1" 代表开启读写分离，默认情况第一个参与写，剩下参与读，switchType="3" 基于 mysql galary cluster 的切换机制，心跳语句为 show status like 'wsrep%';

数据库分片：将存放在同一个数据库中的数据分散存放到多个数据库（主

机)上面,以达到分散单台设备负载的效果。

- ① 按照不同的表(或者 **Schema**)来切分到不同的数据库(主机)之上,这种切可以称之为数据的垂直(纵向)切分;
- ② 根据表中的数据的逻辑关系,将同一个表中的数据按照某种条件拆分到多台数据库(主机)上面,这种切分称之为数据的水平(横向)切分。

参考: https://blog.csdn.net/qq_35152037/article/details/79681762

需要复制, 请用 **word** 文档打开, 然后复制框内的内容

```
<?xml version="1.0"?>
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">
<mycat:schema xmlns:mycat="http://io.mycat/">

    <schema name="cloudsystem" checkSQLschema="true" sqlMaxLimit="100">
        <table name="art_article" primaryKey="ID" dataNode="dn1" />
        <table name="art_article_sku" primaryKey="ID" dataNode="dn1" />
        <table          name="art_article_sku_attament"          primaryKey="ID"
dataNode="dn1" />
        <table          name="art_article_sku_reject_log"          primaryKey="ID"
dataNode="dn1" />
        <table name="art_category" primaryKey="ID" dataNode="dn1" />
        <table name="art_damage" primaryKey="ID" dataNode="dn1" />
        <table name="art_damage_attament" primaryKey="ID" dataNode="dn1" />
        <table name="cms_advertisement" primaryKey="ID" dataNode="dn1" />
        <table name="cms_article" primaryKey="ID" dataNode="dn1" />
        <table name="cms_article_info" primaryKey="ID" dataNode="dn1" />
        <table name="cms_article_like" primaryKey="ID" dataNode="dn1" />
        <table name="cms_attament" primaryKey="ID" dataNode="dn1" />
        <table name="cms_category" primaryKey="ID" dataNode="dn1" />
        <table name="gue_category" primaryKey="ID" dataNode="dn1" />
        <table name="gue_member_option" primaryKey="ID" dataNode="dn1" />
        <table name="gue_option" primaryKey="ID" dataNode="dn1" />
        <table name="gue_question" primaryKey="ID" dataNode="dn1" />
        <table name="mem_account" primaryKey="ID" dataNode="dn1" />
        <table name="mem_account_log" primaryKey="ID" dataNode="dn1" />
        <table name="mem_address" primaryKey="ID" dataNode="dn1" />
        <table name="mem_attament" primaryKey="ID" dataNode="dn1" />
        <table name="mem_check" primaryKey="ID" dataNode="dn1" />
        <table name="mem_check_config" primaryKey="ID" dataNode="dn1" />
        <table name="mem_coupon" primaryKey="ID" dataNode="dn1" />
        <table name="mem_coupon_member" primaryKey="ID" dataNode="dn1"
/>

        <table name="mem_dept" primaryKey="ID" dataNode="dn1" />
        <table name="mem_exception_job" primaryKey="ID" dataNode="dn1" />
        <table name="mem_flow" primaryKey="ID" dataNode="dn1" />
        <table name="mem_goods_member" primaryKey="ID" dataNode="dn1"
/>
    </schema>
</mycat:schema>
```

```

        <table name="mem_job" primaryKey="ID" dataNode="dn1" />
        <table name="mem_job_record" primaryKey="ID" dataNode="dn1" />
        <table name="mem_job_step" primaryKey="ID" dataNode="dn1" />
        <table name="mem_member" primaryKey="ID" dataNode="dn1" />
        <table name="mem_member_article" primaryKey="ID" dataNode="dn1"
/>

        <table name="mem_member_role" primaryKey="ID" dataNode="dn1" />
        <table name="mem_point" primaryKey="ID" dataNode="dn1" />
        <table name="mem_point_log" primaryKey="ID" dataNode="dn1" />
        <table name="mem_role" primaryKey="ID" dataNode="dn1" />
        <table name="pro_attament" primaryKey="ID" dataNode="dn1" />
        <table name="pro_category" primaryKey="ID" dataNode="dn1" />
        <table name="pro_goods" primaryKey="ID" dataNode="dn1" />
        <table name="pro_goods_info" primaryKey="ID" dataNode="dn1" />
        <table name="pro_goods_property" primaryKey="ID" dataNode="dn1" />
        <table name="pro_goods_sku" primaryKey="ID" dataNode="dn1" />
        <table name="pro_order" primaryKey="ID" dataNode="dn1" />
        <table name="pro_order_refund" primaryKey="ID" dataNode="dn1" />
        <table name="pro_property" primaryKey="ID" dataNode="dn1" />
        <table name="pro_shop" primaryKey="ID" dataNode="dn1" />
        <table name="system_attament" primaryKey="ID" dataNode="dn1" />
        <table name="system_dept" primaryKey="ID" dataNode="dn1" />
        <table name="system_log" primaryKey="ID" dataNode="dn1" />
        <table name="system_resource" primaryKey="ID" dataNode="dn1" />
        <table name="system_role" primaryKey="ID" dataNode="dn1" />
        <table name="system_role_resource" primaryKey="ID" dataNode="dn1"
/>

        <table name="system_user" primaryKey="ID" dataNode="dn1" />
        <table name="system_user_role" primaryKey="ID" dataNode="dn1" />
</schema>
<dataNode name="dn1" dataHost="localhost1" database="cloudsystem" />
<dataHost name="localhost1" maxCon="1000" minCon="10" balance="1"
        writeType="0" dbType="mysql" dbDriver="native" switchType="3"
slaveThreshold="100">
        <heartbeat>show status like 'wsrep%'/>heartbeat>

        <writeHost host="hostM1" url="123.207.77.109:3307" user="root"
                password="654321" />
        <writeHost host="hostM2" url="123.207.77.109:3308" user="root"
                password="654321" />
</dataHost>
</mycat:schema>

```

3.5 启动 mycat

启动脚本在 mycat/bin/下
cd mycat/bin/
./mycat start 启动
./mycat stop 关闭
./mycat restart 重启

3.6 连接 mycat 测试

使用 8066 端口、root 用户、123456 密码连接 mycat，插入数据，查看 3306、3307 数据库是否有数据。

3.7 启动另一个 mycat

复制 mycat 到另一台机器
scp -r -p 22 mycat root@192.168.9.214:/u06/root/pkg/

启动
/u06/root/pkg/mycat start

4. Haproxy 集群

4.1 创建 haproxy.cfg 配置文件

```
1. global
2. log 127.0.0.1 local0 notice
3. # user haproxy
4. # group haproxy
5.
6. defaults
7. log global
8. retries 2
9. timeout connect 3000
10. timeout server 5000
11. timeout client 5000
12.
13. listen mycat_service #192.168.9.201:7066
14. mode tcp
15. bind *:7066
16. # option mysql-check user haproxy_check
17. balance roundrobin
18. server mysql01 192.168.9.220:8066 check
19. server mysql02 192.168.9.214:8066 check
20.
21. listen stats #monitor
22. mode http
```

```
23. bind *:1081
24. stats refresh 3s
25. stats uri /
```

复制 haproxy.cfg 到另一台机器

需要复制，请用 word 文档打开，然后复制框内的内容

```
global
log 127.0.0.1 local0 notice
# user haproxy
# group haproxy

defaults
log global
retries 2
timeout connect 3000
timeout server 5000
timeout client 5000

listen mycat_service #192.168.9.201:7066
mode tcp
bind *:7066
# option mysql-check user haproxy_check
balance roundrobin
server mysql01 192.168.9.220:8066 check
server mysql02 192.168.9.214:8066 check

listen stats #monitor
mode http
bind *:1081
stats refresh 3s
stats uri /
```

4.2 启动 HAProxy 镜像

master 机器启动容器:

```
docker run -d --name haproxy --net=host -v
/u06/root/pkg/mytest/haproxy/haproxy.cfg:/usr/local/etc/haproxy/haproxy.
cfg haproxy
```

slave 机器启动容器:

```
docker run -d --name haproxy --net=host -v
/u06/root/pkg/mytest/haproxy/haproxy.cfg:/usr/local/etc/haproxy/haproxy.
cfg haproxy
```

注：需要复制使用，请用 word 文档打开后复制

这里使用的是 HOST 宿舍机网络，原因是使用其他网络（bridge、overlay）可能会导致 HAProxy 无法访问 mycat

4.3 访问监控页面

http://ip1:1081/

http://ip2:1081/

查看 mycat 状态是否为 up

mycat_service																							
	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings						
	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status				
Frontend				0	0	-	0	0	2 000	0		0	0	0	0	0			OPEN				
mysql01	0	0	-	0	0		0	0	-	0	0	?	0	0		0	0	0	0	6m42s	UP		
mysql02	0	0	-	0	0		0	0	-	0	0	?	0	0		0	0	0	0	6m41s	DOWN		
Backend	0	0		0	0		0	0	200	0	0	?	0	0	0	0	0	0	0	6m42s	UP		

4.4 测试

使用 HAProxy 端口 3066 连接数据库，插入数据，查看 3307、3308 数据库是否有数据。

5. keepalived 集群

5.1 安装

官网下载：<http://www.keepalived.org/>

或执行命令 `wget http://www.keepalived.org/software/keepalived-2.0.10.tar.gz`

5.2 解压

`tar -xf keepalived-2.0.10.tar.gz`

5.3 安装

`cd keepalived-2.0.10`

`./configure --prefix=/usr/local/keepalived`

`make && make install`

执行 `configure` 遇到的问题：

① [no acceptable C compiler found in \\$PATH](#)

② [!!! OpenSSL is not properly installed on your system. !!!](#)

执行 `make && make install` 遇到的问题：

[WARNING: 'automake-1.15' is missing on your system.](#)

5.4 复制或者连接文件到指定目录

```
cp -p keepalived/etc/init.d/keepalived /etc/rc.d/init.d/

cp -p keepalived/etc/sysconfig/keepalived /etc/sysconfig

mkdir /etc/keepalived/

cp -p keepalived/etc/keepalived/keepalived.conf /etc/keepalived/

cp -p /usr/local/keepalived/sbin/keepalived /usr/sbin/
```

5.5 创建启动项

```
echo "/etc/init.d/keepalived start" >> /etc/rc.local
chmod 755 /etc/rc.d/init.d/keepalived
开机自启:
chkconfig keepalived on          #开机自启
```

5.6 修改 keepalived.conf 配置文件

```
vi /etc/keepalived/keepalived.conf
```

注: `vrp_script` 最好写在 `vrp_instance` 上方, 否则可能会出现
(Line 22) (VI_1) track script chk_haproxy not found, ignoring...

```
1. global_defs {
2.     notification_email {
3.         root@localhost
4.     }
5.     notification_email_from keepalived@localhost
6.     smtp_server 127.0.0.1      #发送错误发送邮件的地址
7.     smtp_connect_timeout 30
8.     router_id LVS_master      #主从可以一样, 也可以不一样
9. }
10.
11. vrrp_script chk_haproxy {
12.     script "/etc/keepalived/check_haproxy.sh"  #HAProxy 监控监本
13.     interval 20
14.     weight -5
15. }
16.
17. vrrp_instance VI_1 {
18.     state MASTER              #主节点上为 MASTER。从节点上为 BUCKUP
```

```
19.     interface ens33                #网络接口
20.     virtual_router_id 50           #此值主从必须一致
21.     priority 100                   #此值在 MASTER 上比 BUCKUP 大
22.     advert_int 1
23.     mcast_src_ip 192.168.9.220
24.     authentication {
25.         auth_type PASS
26.         auth_pass 1111
27.     }
28.     track_script {
29.         chk_haproxy
30.     }
31.     virtual_ipaddress {
32.         192.168.9.100 dev ens33     #Haproxy 提供的虚拟 ip,主从一样
33.     }
34.     notify_master "/etc/keepalived/clean_arp.sh 192.168.9.100" #主从一样
```

需要复制，请用 **word** 文档打开，然后复制框内的内容

```
global_defs {
    notification_email {
        root@localhost
    }
    notification_email_from keepalived@localhost
    smtp_server 127.0.0.1    #发送错误发送邮件的地址
    smtp_connect_timeout 30
    router_id LVS_master    #主从可以一样，也可以不一样
}

vrrp_script chk_haproxy {
    script "/etc/keepalived/check_haproxy.sh" #HAProxy 监控监本
    interval 20
    weight -5
}

vrrp_instance VI_1 {
    state MASTER            #主节点上为 MASTER。从节点上为 BUCKUP
    interface ens33         #网络接口
    virtual_router_id 50    #此值主从必须一致
    priority 100            #此值在 MASTER 上比 BUCKUP 大
    advert_int 1
    mcast_src_ip 192.168.9.220
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    track_script {
        chk_haproxy
    }
    virtual_ipaddress {
        192.168.9.100 dev ens33    #Haproxy 提供的虚拟 ip,主从一样
    }
    notify_master "/etc/keepalived/clean_arp.sh 192.168.9.100"    #主从一样
}
```

5.7 设置更新虚拟服务器（VIP）地址的 arp 记录到网关脚本

vi /etc/keepalived/clean_arp.sh

授权 chmod 777 /etc/keepalived/clean_arp.sh


```

1.  #!/bin/bash
2.  VIP=$1
3.  GATEWAY=192.168.9.1      #这个是本机的外网网卡网关地址
4.  /sbin/arping -I ens33 -c 5 -s $VIP $GATEWAY &>/dev/null

```

需要复制，请用 **word** 文档打开，然后复制框内的内容

```

#!/bin/bash
VIP=$1
GATEWAY=192.168.9.1      #这个是本机的外网网卡网关地址
/sbin/arping -I ens0 -c 5 -s $VIP $GATEWAY &>/dev/null

```

5.8 设置 HAproxy 服务监控脚本

vi /etc/keepalived/check_haproxy.sh

授权 chmod 777 /etc/keepalived/check_haproxy.sh

```

1.  #!/bin/bash
2.  A=`ps -C haproxy --no-header | wc -l`
3.  if [ $A -eq 0 ];then
4.      docker rm -f haproxy &
5.      docker run -d -p 7066:7066 -p 1081:1081 --name haproxy --net=pxc-net -
v /u06/root/pkg/mytest/haproxy/haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg haproxy
6.      sleep 3
7.      if [ `ps -C haproxy --no-header | wc -l` -eq 0 ];then
8.          /etc/init.d/keepalived stop
9.      fi

```

需要复制，请用 **word** 文档打开，然后复制框内的内容

```
#!/bin/bash
A=`ps -C haproxy --no-header | wc -l`
if [ $A -eq 0 ];then
    docker rm -f haproxy &
    docker rm -f haproxy1 &
    docker run -d -p 7066:7066 -p 1081:1081 --name haproxy --net=pxc-net -v
/u06/root/pkg/mytest/haproxy/haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg
haprox
y
    docker run -d -p 7067:7066 -p 1082:1081 --name haproxy1 --net=pxc-net -v
/u06/root/pkg/mytest/haproxy/haproxy1.cfg:/usr/local/etc/haproxy/haproxy.cfg
hapr
oxy
    sleep 3
    if [ `ps -C haproxy --no-header | wc -l` -eq 0 ];then
        /etc/init.d/keepalived stop
    fi
fi
```

5.9 内核优化

1. echo 1024 60999 > /proc/sys/net/ipv4/ip_local_port_range
2. echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
3. echo 4096 > /proc/sys/net/ipv4/tcp_max_syn_backlog
4. echo 262144 > /proc/sys/net/ipv4/tcp_max_tw_buckets
5. echo 262144 > /proc/sys/net/ipv4/tcp_max_orphans
6. echo 300 > /proc/sys/net/ipv4/tcp_keepalive_time
7. echo 1 > /proc/sys/net/ipv4/tcp_tw_recycle
8. echo 0 > /proc/sys/net/ipv4/tcp_timestamps
9. echo 0 > /proc/sys/net/ipv4/tcp_ecn
10. echo 1 > /proc/sys/net/ipv4/tcp_sack
11. echo 0 > /proc/sys/net/ipv4/tcp_dsack

需要复制，请用 word 文档打开，然后复制框内的内容

```
echo 1024 60999 > /proc/sys/net/ipv4/ip_local_port_range
echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
echo 4096 > /proc/sys/net/ipv4/tcp_max_syn_backlog
echo 262144 > /proc/sys/net/ipv4/tcp_max_tw_buckets
echo 262144 > /proc/sys/net/ipv4/tcp_max_orphans
echo 300 > /proc/sys/net/ipv4/tcp_keepalive_time
echo 1 > /proc/sys/net/ipv4/tcp_tw_recycle
echo 0 > /proc/sys/net/ipv4/tcp_timestamps
echo 0 > /proc/sys/net/ipv4/tcp_ecn
echo 1 > /proc/sys/net/ipv4/tcp_sack
echo 0 > /proc/sys/net/ipv4/tcp_dsack
```

5.10 启动

```
service keepalived start    #启动
service keepalived stop    #关闭
service keepalived restart  #重启
```

5.11 安装配置并启动另一台机器的 keepalived

需要修改 keepalived.conf:

- state MASTER 改为 state BACKUP
- priority 100 改为 priority 99（数值小于 100）

5.12 测试

若 keepalived 启动正常，master 机器使用 ip addr 命令可以看到虚拟 ip:

```
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:31:98:5f brd ff:ff:ff:ff:ff:ff
    inet 192.168.9.214/24 brd 192.168.9.255 scope global noprefixroute dynamic ens33
        valid_lft 57775sec preferred_lft 57775sec
    inet 192.168.9.100/32 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::aa95:bb72:a351:36f0/64 scope link noprefixroute
```

使用虚拟 ip: 192.168.9.100 访问数据库，进行数据操作，观察数据是否正常；

关闭 master 的 keepalived，backup 机器使用 ip addr 命令可以看到虚拟 ip:

继续使用虚拟 ip 访问数据库，进行数据操作，观察数据是否正常。

6. 问题和解决方案

6.1 Percona XtraDB Cluster 问题

容器删除或停止后，重新启动失败，日志显示

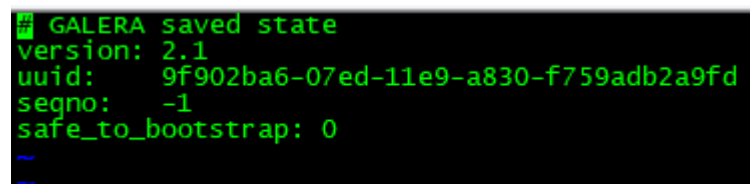
```
28 15772 [ERROR] WSREP: It may not be safe to bootstrap the cluster from this node.  
t the grastate.dat file manually and set safe_to_bootstrap to 1 .  
28 15772 [ERROR] WSREP: wsrep::connect(gcomm://192.168.9.136,192.168.9.143,192.168.9.144) failed: 1  
28 15772 [ERROR] Aborting  
  
28 15772 [Note] WSREP: Service disconnected.  
28 15772 [Note] WSREP: waiting to close threads.....  
33 15772 [Note] WSREP: Some threads may fail to exit.  
33 15772 [Note] Binlog end  
33 15772 [Note] /usr/sbin/mysqld: shutdown complete
```

解决方法:

修改创建的数据目录下的 grastate.dat 文件

vi /data/mysql1/grastate.dat

safe_to_bootstrap: 1



```
GALERA saved state  
version: 2.1  
uuid: 9f902ba6-07ed-11e9-a830-f759adb2a9fd  
seqno: -1  
safe_to_bootstrap: 0  
~  
~
```

6.2 keepalived 安装问题 ([返回安装流程](#))

① 执行 ./configure -prefix=/usr/local/keepalived 时报错

configure: **error**: in `/u06/root/pkg/mytest/keepalived/keepalived-2.0.10':

configure: **error**: no acceptable C compiler found in \$PATH

See `config.log' for more details

解决方法:

yum -y install gcc

② 执行 ./configure -prefix=/usr/local/keepalived 时报错

configure: error:

!!! OpenSSL is not properly installed on your system. !!!

!!! Can not include OpenSSL headers files. !!!

解决方法:

yum -y install openssl-devel

③ 执行 make && make install 时报错

WARNING: 'automake-1.15' is missing on your system.

You should only need it if you modified 'Makefile.am' or

'configure.ac' or m4 files included by 'configure.ac'.

The 'automake' program is part of the GNU Automake package:

```
<http://www.gnu.org/software/automake>
It also requires GNU Autoconf, GNU m4 and Perl in order to run:
<http://www.gnu.org/software/autoconf>
<http://www.gnu.org/software/m4/>
<http://www.perl.org/>
make: *** [Makefile.in] 错误 1
```

解决方法:

下载安装包

```
wget https://ftp.gnu.org/gnu/automake/automake-1.15.tar.gz
```

解压

```
tar -xf automake-1.15.tar.gz
```

安装

```
cd automake-1.15
```

```
./configure
```

```
make && make install
```

④ 安装 **automake** 时报错:

```
configure.ac:20: error: Autoconf version 2.65 or higher is required
```

下载安装包

```
wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.68.tar.gz
```

解压

```
tar xzf autoconf-2.68.tar.gz
```

安装

```
cd autoconf-2.68
```

```
./configure
```

```
make && make install
```