OpenStack

High Availability of nova-network Component

Version 1.0

25/04/2011

NTT DATA AgileNet L.L.C.

*Copyright © NTT DATA AgileNet L.L.C.*

－　History　－

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Page** | **Content** |
| 1.0 | 26/04/2011 | 1 | Initial version |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Assumptions 4](#_Toc291625960)

[2. High Availability of nova-network node 4](#_Toc291625961)

[3. Design Approaches 4](#_Toc291625962)

[3.1 Active/Standby using Linux-HA 4](#_Toc291625963)

[3.1.1 Detecting failover criteria 4](#_Toc291625964)

[3.1.2 Cluster configuration 5](#_Toc291625965)

[3.1.3 Current Release 5](#_Toc291625966)

[3.1.4 Installing pre-built packages 5](#_Toc291625967)

[3.1.5 Configure Heartbeat 6](#_Toc291625968)

[3.1.6 Nova-network Configuration 10](#_Toc291625969)

[3.1.7 Test HA for the network server 10](#_Toc291625970)

[3.1.8 Disconnection Time 11](#_Toc291625971)

[3.2 Active/Standby and Active/Active using Keepalived 14](#_Toc291625972)

# Assumptions

* Both primary and standby network servers in the HA pool should be running on the same network.

# High Availability of nova-network node

There are many open source software’s available to achieve high availability of critical applications.

Here we will be focusing mainly on following HA open source software’s:-

* Linux-HA
* Keepalived

# Design Approaches

## Active/Standby using Linux-HA

In this approach, we are trying to switch over from active network server to standby network server using heartbeat as a HA software.

### Detecting failover criteria

Failover will be detected by the standby network server in following cases

* heartbeat service is down
* network failure(ethernet interface on which the udp packets are sent by the heartbeats is down i.e. eth0, eth1 and so on)
* primary network server is powered off
* nova-network service is not running on primary network server
* syslog has reported any fatal error

### Cluster configuration

The test cluster configuration is shown in below figure 2. The setup consists of a pair of network server’s ubuntu-network-server-01 which will act as a primary and ubuntu-network –server-02 as a standby server. At any given point of time, only one of the network servers will be active and another server will be in dormant or passive state.

Figure : Heartbeat cluster configuration

### Current Release

The current stable release for Heartbeat is 3.0.3, released on Apr 14, 2010.

### Installing pre-built packages

Installing heartbeat package on Ubuntu is straightforward process. Assuming you have the correct packages repositories configured for APT, install the heartbeat package with the following command:-

|  |
| --- |
| *$aptitude install heartbeat* |

### Configure Heartbeat

You must configure three files to get heartbeat to work: authkeys, ha.cf and haresources.

##### /etc/ha.d/ha.cf - the global cluster configuration

##### /etc/ha.d/authkeys - a file containing keys for mutual node authentication

##### /etc/ha.d/haresources - This file specifies the services for the cluster and who the default owner is.

##### Ha.cf

This file will be placed in the /etc/ha.d directory that is created after installation of heartbeat. It tells heartbeat what types of media paths to use and how to configure them. This file also defines the nodes in the cluster and the interfaces that heartbeat uses to verify whether or not a system is up. To understand the meaning of each directive used please read the information at the website <http://www.linux-ha.org/wiki/Ha.cf>

Table 1 ha.cf file on the primary network server

|  |
| --- |
| logfacility daemon  #set heartbeat keep-alive interval  keepalive 250ms  #set failure (death) detection time  deadtime 4  #set late heartbeat warning time  warntime 3  #set initial deadtime detection interval  initdead 8  udpport 694  #configures unicast Heartbeat communication  ucast eth0 192.168.0.9# The Private IP address of your SLAVE server.  auto\_failback off  node ubuntu-openstack-network-server-01 # The hostname of your MASTER Server.  node ubuntu-openstack-network-server-02 # The hostname of your SLAVE Server.  respawn hacluster /usr/lib/heartbeat/ipfail  use\_logd yes |

Table 2 ha.cf file on the standby network server

|  |
| --- |
| logfacility daemon  keepalive 250ms  deadtime 4  warntime 3  initdead 8  udpport 694  ucast eth0 192.168.0.1# The Private IP address of your MASTER server.  auto\_failback off  node ubuntu-openstack-network-server-01 # The hostname of your MASTER Server.  # The hostname of your SLAVE Server.  node ubuntu-openstack-network-server-02  respawn hacluster /usr/lib/heartbeat/ipfail  use\_logd yesuse\_logd yes |

##### The authkeys file

This file determines your authentication keys for the cluster; the keys must be the same on both the servers. You can choose from three authentication schemes: crc, md5, or sha1. If your heartbeat runs over a secure network, such as the crossover cable, you will want to use crc. This is the cheapest method from a resources perspective. If the network is insecure, but you are concerned about minimizing CPU resources, use md5. Finally, if you want the best authentication without regard for CPU resources, use sha1, as it’s hardest to crack.

It should only be readable by root and follows this format:

|  |
| --- |
| auth <num>  <num> <algorithm> <secret>  num - is a simple key index, starting with 1. Usually, you will only have one key in your authkeys file.  algorithm - is the signature algorithm being used. You may use either md5 or sha1; the use of crc (a simple cyclic redundancy check, not secure) is not recommended.  secret - is the actual authentication key. |

Sample

|  |
| --- |
| auth 1  1 sha1 YourSecretPassPhrase |

##### Haresources

This file describes the resources that are managed by heartbeat. The resources are basically just start/stop scripts. Note that heartbeat will look for these scripts in /etc/ha.d/resource.d and /etc/rc.d/init.d directory in the same order.

haresources file:-

|  |
| --- |
| ubuntu-openstack-network-server-01 10.2.3.120/32/eth1 nova-network |

This file must be same on both the servers.

This line dictates that on startup of heartbeat service:-

* Have ubuntu-openstack-network-server-01 server the IP 10.2.3.120
* Start nova-network-script

This nova-network-script does following on startup:-

* killall dnsmasq and radvd processes if any
* clean all vlan and bridges
* Start nova-network service

On shutdown, heartbeat will:

* Stop nova-network-script

This nova-network-script does following during stop command:-

* Killall dnsmasq and radvd processes if any
* clean all vlan and bridges
* ip address del to unbind floating/public IP addresses from public interface
* Stop nova-network service
* Remove the Virtual IP Address 10.2.3.120

Table 3 : nova-network-script

|  |
| --- |
| #!/usr/bin/env bash  CMD=$1  echo $CMD  NL=`echo -ne '\015'`  function screen\_it {  screen -S nova -X screen -t $1  screen -S nova -p $1 -X stuff "$2$NL"  }  if [ "$CMD" == "start" ]; then  killall dnsmasq radvd  sudo /home/tpatil/nova/tools/clean-vlans  sudo /home/tpatil/nova/tools/setup\_iptables.sh clear all  screen -d -m -S nova -t nova  # nova api crashes if we start it with a regular screen command,  # so send the start command by forcing text into the window.  screen\_it network "/home/tpatil/nova/bin/nova-network --flagfile=/etc/nova/nova.conf"  exit 0  fi  if [ "$CMD" == "stop" ]; then  killall dnsmasq radvd  sudo /home/tpatil/nova/tools/clean-vlans  sudo /home/tpatil/nova/tools/setup\_iptables.sh clear all  #Disassociate all floating ip address  sudo ip addr del 10.2.3.104/32 dev eth1  screen -S nova -X quit  rm \*.pid\*  exit 0  fi |

### Nova-network Configuration

nova.conf

|  |
| --- |
| # routing source ip should be same on both primary and standby network server. This IP address will be #virtual ip address  --routing\_source\_ip=10.2.3.120  #host should be same on both primary and standby network server. All network messages #communication happens using rabbitmq on topic network.<hostname>. By default hostname is the #hostname of the server on which nova-network server is running if host flag is not provided. After #failover we don’t want hostname to differ from the primary network server so it should use host flag to #get rid of this problem. Most importantly, administrator will have to use the same host to create  #networks using nova-manage.  --host=somehostname |

### Test HA for the network server

To test the high availability of the network server:-

1. Start the heartbeat service on the primary and then on the standby network server using this command:

|  |
| --- |
| $service heartbeat start |

If it fails, look in /var/log/syslog to determine the reason and then correct it. After the heartbeat starts successfully, you should see a new network interface with the IP address that you configured in the haresources file. Once you’ve started heartbeat, take a peek at your log file on the primary and make sure that it is doing the IP takeover and then starting the nova-network service. Use the ps command to make sure that nova-network service is running on the primary server. Heartbeat will not start any nova-network service on the standby network server. This happens only after the primary fails.

1. Run new VM instance and associate floating ip address 10.2.3.104 to an instance.
2. Ping to 10.2.3.104 from your laptop. Ping should be successful.
3. Now simulate failover by simply stopping heartbeat on the primary network server using the command as shown below

|  |
| --- |
| $service heartbeat stop |

You will notice nova-network service will be running on the standby network server and nova-network service will be stopped on the primary server.

1. Verify that you can still ping to 10.2.3.104. If ping is successful that means you are successful in implementing HA of the network server.

Note: if you start the heartbeat service on the primary again, then the primary network server will go in the passive state and standby will still be in the active. If you want primary server to become active immediately even if standby is active, then you need to configure auto\_failback directive to on.

### Disconnection Time

The disconnection time will vary depending upon the settings you have made in the ha.cf. You will need to fine tune these settings to achieve best possible failover time. For better results please read FAQ provided at <http://linux-ha.org/wiki/FAQ#Heavy_Load>. The failover time is depended on following directives:

deadtime, in ha.cf above I have set it to 4 seconds. That means if you stop the primary server then the failover will take place after 4 seconds and you shouldn’t get ping response for 4 seconds. To find out the actual network disconnection time, I have taken 3 samples of tcpdump (tcpdump -i eth1 -n icmp) with different deadtime settings and followed Section [Test HA for the network server](#_Test_HA_for) to simulate failover.

Table 4 : Disconnection Time

|  |  |  |  |
| --- | --- | --- | --- |
| Dead time (in Seconds) | Ping Reply Stop | Ping Reply Response | Time difference (in Seconds) |
| 4 | 15:20:28 | 15:20:56 | 28 |
| 4 | 15:21:47 | 15:22:14 | 27 |
| 4 | 15:23:46 | 15:24:26 | 40 |
| 3 | 16:23:07 | 16:23:54 | 47 |
| 3 | 16:25:16 | 16:26:03 | 47 |
| 3 | 16:26:59 | 16:27:46 | 47 |
| 2 | 16:40:54 | 16:41:19 | 25 |
| 2 | 16:41:56 | 16:42:20 | 24 |
| 2 | 16:43:02 | 16:43:26 | 24 |

Figure 3 : Disconnection time

#### Analysis

Based on the above disconnection time results, it looks like the disconnection time is not depended on deadtime setting directly. The failover happens exactly after 4/3/2 seconds but ARP table in the neighboring machines are not updated with the new MAC address of the standby network server. It takes lot of time to update the ARP table if we don’t broadcasts a gratuitous ARP reply to inform the neighboring machines about the change of MAC address for the IP.

I tried broadcasting a gratuitous ARPs to router (in our setup 10.2.3.1 is the router) just before starting the nova network service and recorded the disconnection time again.

ARP command

arping -b -c 1 -s <floating Ip address> -I <public interface> <router ip address>

$arping -b -c 1 -s 10.2.3.104 -I eth1 10.2.3.1

We may need to send gratuitous ARPs for following ips

- Routing source ip

- Floating ips

- Gateway address of each vlan(which is assigned to each bridge)

Note: I think we should make changes to the nova-network source code to address all of the 3 points above. Even if we modify the source code to send the ARP messages, there will be a down time of few seconds. However, this modification sound reasonable than the current implementation in terms of HA.

Table 5: Disconnection Time

|  |  |  |  |
| --- | --- | --- | --- |
| Dead time (in Seconds) | Ping Reply Stop | Ping Reply Response | Time difference (in Seconds) |
| 4 | 13:54:22 | 13:54:26 | 4 |
| 4 | 13:55:04 | 13:55:08 | 4 |
| 4 | 13:55:43 | 13:55:47 | 4 |
| 3 | 14:06:33 | 14:06:37 | 4 |
| 3 | 14:07:23 | 14:07:27 | 4 |
| 3 | 14:08:19 | 14:08:24 | 5 |
| 2 | 14:11:39 | 14:11:44 | 5 |
| 2 | 14:12:12 | 14:12:16 | 4 |
| 2 | 14:12:38 | 14:12:43 | 5 |

Figure 4: Disconnection time

If you see by broadcasting a gratuitous ARPs, the disconnection time is reduced substantially to an average of 4-5 seconds.

Of course this disconnection time will depend on the number of factors like the number of vm instances, vlans, bridges etc. For POC, I have used 2 VM instances each associated with one floating ip address.

If we use primitive resource manager which is supported by heartbeat, then maximum 2 network servers will be allowed to be configured in the HA pool. But I think if we use CRM (Pacemaker), then 1: N network servers can be configured in the HA pool.

## Active/Standby and Active/Active using Keepalived

In order to achieve active/standby/active HA of the nova network server, we are trying out keepalived.

But we didn't finish the POC yet. We would like to discuss how to realize active/active HA to reduce the downtime to few milliseconds. I think if we use keepalived (virtual ip address using virtual mac address), we can completely eliminate the need to send the gratuitous ARPs. Keepalived have recently introduced VMAC support in the month of April 2011 and it is in the testing phase. So I think it will be too early to introduce in Openstack.