Heartbeat2 Xen cluster with drbd8 and OCFS2

Posted by AtulAthavale on Tue 29 Jan 2008 at 13:04

The idea behind the whole set-up is to get a High availability two node Cluster with redundant data. The two identical Servers are installed with Xen hypervisor and almost same configuration as Cluster nodes. The configuration and image files of Xen virtual machines are stored on drbd device for redundancy. Drbd8 and OCFS2 allows simultaneous mounting on both nodes, which is required for live migration of xen virtual machines.

This Article describes Heartbeat2 Xen cluster Using Ubuntu (7.10) OS, drbd8 and OCFS2 (Ver. 1.39) File system. Although here Ubuntu is used it can be done in almost same way with Debian

Setup

OS Installation

Install two Computers with standard minimal Ubuntu Server (7.10) OS. After standard installation is done, we go ahead installing required packets.

Disc Partition

On both computers we partition the disc in three partitions and use as follows /dev/sda1 as /root /dev/sda2 as swap /dev/sda3 as drbd8 (just leave it as it is at the time of installation)

Network Configuration

Node	Hostname	IP-Address
Node1	node1	192.168.0.128
Node2	node2	192.168.0.129

Xen system

http://en.wikipedia.org/wiki/Xen We start with installing Xen Hypervisor and boot with Xen-kernel.

```
sudo apt-get install ubuntu-xen-server
```

Answer yes for additional software. Reboot the system with Xen hypervisor

OCFS2

http://oss.oracle.com/projects/ocfs2/ OCFS2 is a Cluster File System which allows simultaneous access from many nodes. We will set this on our drbd device to access it from both nodes simultaneously. While configuring OCFS2 we provide the information about nodes, which will access the file system later. Every Node that has a OCFS2 file system mounted, must regularly write into a meta-data of file system, letting the other nodes know that node is still alive.

Installation

```
sudo apt-get install ocfs2-tools ocfs2console
```

Configuration

Edit /etc/ocfs2/cluster.conf as follows

```
#/etc/ocfs2/cluster.conf

node:

        ip_port = 7777

        ip_address = 192.168.0.128

        number = 0

        name = node1

        cluster = ocfs2

node:

        ip_port = 7777

        ip_address = 192.168.0.129

        number = 1

        name = node2

        cluster = ocfs2

cluster:

        node_count = 2

        name = ocfs2
```

reconfigure ocfs2 with following command with their default values

```
sudo dpkg-reconfigure o2cb

sudo /etc/init.d/o2cb restart

sudo /etc/init.d/ocfs2 restart
```

drbd8

Installation

http://en.wikipedia.org/wiki/Drbd

The advantage of drbd8 over drbd7 is: It allows the drbd resource to be “master” on both nodes and so can be mounted read-write. We will build drbd8 modules and load it in kernel. For that we need packages “build-essential” and “kernel-headers-xen”

```
sudo apt-get install drbd8-utils drbd8-module-source drbd8-source  build-essential linux-headers

sudo sudo m-a a-i drbd8-module-source

sudo update-modules
sudo modprobe drbd
```

This builds the drbd module kernel/drivers/block/drbd.ko against the current running kernel. A default configuration file is installed as /etc/drbd.conf

Configuration

Edit the /etc/drbd.conf

```
#/etc/drbd.conf

global {

    usage-count yes;

}

common {

  syncer { rate 10M; }

}

resource r0 {

  protocol C;

  handlers {

    pri-on-incon-degr "echo o > /proc/sysrq-trigger ; halt -f";

    pri-lost-after-sb "echo o > /proc/sysrq-trigger ; halt -f";

    local-io-error "echo o > /proc/sysrq-trigger ; halt -f";

    outdate-peer "/usr/sbin/drbd-peer-outdater";

  }

  startup {

  }

  disk {

    on-io-error   detach;

  }

  net {

    allow-two-primaries;

    after-sb-0pri disconnect;

    after-sb-1pri disconnect;

    after-sb-2pri disconnect;

    rr-conflict disconnect;

  }
  syncer {

    rate 10M;

    al-extents 257;

  }

  on node1 {

    device     /dev/drbd0;

    disk       /dev/sda3;

    address    192.168.0.128:7788;

    flexible-meta-disk  internal;

  }

  on node2 {

    device    /dev/drbd0;

    disk      /dev/sda3;

    address   192.168.0.129:7788;

    meta-disk internal;

  }

}
```

“ allow-two-primaries” option in net section of drbd.conf allows the resource to be mounted as “master” on both nodes. Copy the /etc/drbd.conf to node2 and restart drbd on both nodes with following command.

```
sudo /etc/init.d/drbd restart
```

If you check the status it looks like this

```
suddo /etc/init.d/drbd status

drbd driver loaded OK; device status:

version: 8.0.3 (api:86/proto:86)

SVN Revision: 2881 build by root@node1, 2008-01-20 12:48:36

 0: cs:Connected st:Secondary/Secondary ds:UpToDate/UpToDate C r---

    ns:143004 nr:0 dw:0 dr:143004 al:0 bm:43 lo:0 pe:0 ua:0 ap:0

        resync: used:0/31 hits:8916 misses:22 starving:0 dirty:0 changed:22

        act_log: used:0/257 hits:0 misses:0 starving:0 dirty:0 changed:0
```

change the resource to “master” with following command on both nodes

```
 sudo drbdadm primary r0
```

and check the status again

```
sudo /etc/init.d/drbd status

drbd driver loaded OK; device status:

version: 8.0.3 (api:86/proto:86)

SVN Revision: 2881 build by root@node1, 2008-01-20 12:48:36

 0: cs:Connected st:Primary/Primary ds:UpToDate/UpToDate C r---

    ns:143004 nr:0 dw:0 dr:143004 al:0 bm:43 lo:0 pe:0 ua:0 ap:0

        resync: used:0/31 hits:8916 misses:22 starving:0 dirty:0 changed:22

        act_log: used:0/257 hits:0 misses:0 starving:0 dirty:0 changed:0
```

As you can see resource is “master” on both nodes Th drbd device is now accessible under /dev/drbd0

File system

We can now create a file system on /der/drbd0 by following command

```
sudo mkfs.ocfs2 /dev/drbd0
```

This can be mounted on both nodes simultaneously with

```
sudo mkdir /drbd0

sudo mount.ocfs2 /dev/drbd0 /drbd0
```

Now we have a common storage which will be synchronized with drbd on both nodes

Init script

We have to make sure that after reboot, the system will set drbd resources again to “master” and mount those on “/drbd0” before starting Heartbeat and Xen machines.

Edit /etc/init.d/mountdrbd.sh

```
#/etc/init.d/mountdrbd.sh

drbdadm primary r0

mount.ocfs2 /dev/drbd0 /mnt
```

make it executable and add symbolic link to this under /etc/rc3.d/S99mountdrbd.sh

```
sudo chmode +x /etc/init.d/mountdrbd.sh

sudo ln -s /etc/init.d/mountdrbd.sh /etc/rc3.d/S99mountdrbd.sh
```

Actually this step can be integrated also in Heartbeat by adding appropriate resources to the configuration. But as time being we will do this with script.

Heartbeat2

http://www.linux-ha.org/Heartbeat

Installation

Now we can install and setup Heartbeat 2

```
sudo apt-get install heartbeat-2 heartbeat-2-gui
```

Edit /etc/ha.d/ha.cf

```
#/etc/ha.d/ha.cf

crm on

bcast eth0

node node1 node2
```

and restart heartbeat2 with

```
sudo /etc/init.d/heartbeat restart
```

Configuration

In Heartbeat2 the configuration and status information of resources are stored in xml format in “/usr/lib/heartbeat/crm/cib.xml” file. Thy Syntax for this is very well explained by Alan Robertson in his tutorial at the linux.conf.au 2007. Which can be found at http://linux-ha.org/HeartbeatTutorials

This file can either edited directly as whole or manipulated in pieces using “cibadmin” tool. We will use this tool as it makes it much easier to manage the cluster. The required components we will save in xml files under /root/cluster

Initialaization

Edit file /root/cluster/bootstrap.xml

```
#/root/cluster/bootstrap.xml

<cluster_property_set id="bootstrap">

 <attributes>

  <nvpair id="bootstrap01" name="transition-idle-timeout" value="60"/>

  <nvpair id="bootstrap02" name="default-resource-stickiness" value="INFINITY"/>

  <nvpair id="bootstrap03" name="default-resource-failure-stickiness" value="-500"/>

  <nvpair id="bootstrap04" name="stonith-enabled" value="true"/>

  <nvpair id="bootstrap05" name="stonith-action" value="reboot"/>

  <nvpair id="bootstrap06" name="symmetric-cluster" value="true"/>

  <nvpair id="bootstrap07" name="no-quorum-policy" value="stop"/>

  <nvpair id="bootstrap08" name="stop-orphan-resources" value="true"/>
  <nvpair id="bootstrap09" name="stop-orphan-actions" value="true"/>

  <nvpair id="bootstrap10" name="is-managed-default" value="true"/>

 </attributes>

</cluster_property_set>
```

Load this file with following command

```
 sudo cibadmin -C crm_config -x /root/cluster/bootstrap.xml
```

This will initialize the Cluster with values set in xml file. (some how if it has alredy set you can use “sudo cibadmin -M crm_config -x /root/cluster/bootstrap.xml” to modify it with our new values)

Setting up STONITH device

STONITH prevents “split-brain-situation” (i.e. running Resource on both nodes unwontedly at same time) by fencing the other node. Details can be found out at http://www.linux-ha.org/STONITH We will use “stonth” over ssh to reboot the faulty machine

```
 sudo apt-get install stonith
```

Follow “ http://sial.org/howto/openssh/publickey-auth/” to setup public key authentication. In short just do following on both nodes

```
sudo ssh-keygen

--> save key under /root/.ssh/*

-->dont give any passphrase

scp /root/.ssh/id_rsa.pub node2:/root/.ssh/authorized_keys
```

Now check that you can log on from node1 to node2 per ssh without password asked and vice a versa Now check that stonith is working

```
sudo ssh -q -x -n -l root "node2" "ls -la"
```

you should get a file list from node2 Now we configure “stonith” device as Cluster resource. It will be a special cluster resource “Clone” which will run simultaneously on all nodes.

```
#/root/cluster/stonith.xml

<clone id="stonithclone" globally_unique="false">

 <instance_attributes id="stonithclone">

  <attributes>

   <nvpair id="stonithclone01" name="clone_node_max" value="1"/>

  </attributes>

 </instance_attributes>

 <primitive id="stonithclone" class="stonith" type="external/ssh" provider="heartbeat">

  <operations>

   <op name="monitor" interval="5s" timeout="20s" prereq="nothing" id="stonithclone-op01"/>
```

```
    <op name="start" timeout="20s" prereq="nothing" id="stonithclone-op02"/>

   </operations>

  <instance_attributes id="stonithclone">

   <attributes>

    <nvpair id="stonithclone01" name="hostlist" value="node1,node2"/>

   </attributes>

  </instance_attributes>

  </primitive>

 </clone>
```

Load this file with following command

```
sudo cibadmin -C -o resources -x /root/cluster/stonith.xml
```

Xen as cluster resource

Now we can add a Xen virtual machine as cluster resource.Lets say we have a Xen para visualized machine called vm01. The cofiguration and image files of vm01 we keep under /drbd0/xen/vm01/ as vm01.cfg and vm01-disk0.img respectively

Edit /root/cluster/vm01.xml

```
#/root/cluster/vm01.xml

<resources>

 <primitive id="vm01" class="ocf" type="Xen" provider="heartbeat">

  <operations>

   <op id="vm01-op01" name="monitor" interval="10s" timeout="60s" prereq="nothing"/>

   <op id="vm01-op02" name="start" timeout="60s" start_delay="0"/>

   <op id="vm01-op03" name="stop" timeout="300s"/>

 </operations>

 <instance_attributes id="vm01">

  <attributes>

   <nvpair id="vm01-attr01" name="xmfile" value="/drbd0/xen/vm01/vm01.cfg"/>

   <nvpair id="vm01-attr02" name="target_role" value="started"/>

  </attributes>

 </instance_attributes>

 <meta_attributes id="vm01-meta01">

  <attributes>
```

```
   <nvpair id="vm01-meta-attr01" name="allow_migrate" value="true"/>

  </attributes>

 </meta_attributes>

 </primitive>

</resources>
```

Load this file with following command

```
sudo cibadmin -C -o resources -x /root/cluster/vm01.xml
```

Monitoring Tool

With command “crm_mon” you can monitor the cluster including its nodes and resources

```
sudo crm_mon Refresh in 14s...

============

Last updated: Fri Jan 25 17:26:10 2008

Current DC: node2 (83972cf7-0b56-4299-8e42-69b3411377a7)

2 Nodes configured.

6 Resources configured.

============

Node: node2 (83972cf7-0b56-4299-8e42-69b3411377a7): online

Node: node1 (6bfd2aa7-b132-4104-913c-c34ef03a4dba): online

Clone Set: stonithclone

        stonithclone:0      (stonith:external/ssh): Started node1

        stonithclone:1      (stonith:external/ssh): Started node2

   vm01    (heartbeat::ocf:Xen):   Started node2
```

There is also a GUI available. For using it just set a password for user “hacluster” with following command and call “hb_gui”

```
sudo passwd hacluster

password

re type password

sudo hb_gui &
```

Managing Tool

The Cluster resources can be managed either with GUI or with crm_* commands. Please refer to “man” pages for details list of crm_* commands: crm_attribute, crm_failcount, crm_mon, crm_sh, crm_uuid, crm_diff, crm_master, crm_resource , crm_standby, crm_verify

I hope you find some fun trying it out. Gruß, atul.athavale [at] gmail [dot] com .
