Servidores Linux Enterprise

En las siguientes líneas he documentado de forma muy corta, algunas de las cosas que veo como más intereantes, de cara a instalar servidores empresariales con Linux.

En todos los casos me he basado en instalaciones de Debian 4.0 (etch), aunque quizás no sea la mejor alternativa de Linux Enterprise. En todo caso varios de los conceptos e ideas se podrán aplicar a otras distribuciones.

Bondig de tarjetas de red

Bondig de tarjetas de red se utiliza para tener redundancia en las conexiones de red. Para ello lo que se hace es crear un interface de red virtual (generalmente bond0) y a este interface añadirle tarjetas de red físicas (por ejemplo eth0 y eth1).

De esta forma se hará la configuración de red sobre el nuevo interface virtual (por ejemplo asignar la dirección IP) y el sistema decidirá por qué interface física saldrán los datos. En caso de caída de una interface (por ejemplo por que se suelte un interface de red) se utilizará la otra que tenga conexión.

Para configurar bonding es necesario tener un módulo en el kernel (bonding.ko) y una utilidad para manejar los parametros del kernel.

En debian (etch 4.0 e incluso anteriores), el módulo del kernel viene precompilado y sólo es necesario instalar el paquete ifenslave (apt-get install ifenslave).

Los comandos para configurar el interface virtual son los siguientes:

```
modprobe bonding miimon=100 mode=1 downdelay=100 updelay=5000
ifconfig bond0 up
ifenslave -f bond0 eth0
ifenslave -f bond0 eth1
ifconfig bond0 192.168.1.5 netmask 255.255.255.0
```

El primer comando (el modprobe) carga el módulo de bonding configurando algunos parametros. Ver el significado de los parametros en fichero de documentacion del kernel, en caso de que esté instalado /usr/src/linux-source-2.6.18/Documentation/networking/bonding.txt).

El segundo comando (ifconfig bond0 up) crea el interface bond0, aquí podría haberse configurado directamente la IP también.

Los dos siguientes (los dos ifenslave) crean un interface virtual (bond0) con dos tarjetas de red físicas (eth0 y eth1)

Por último le configuramos la dirección de red al nuevo interface.

Configurar RAID por Software en Linux

Con el fin de mejorar tanto la disponibilidad, como la velocidad de acceso a los discos duros, lo que se suele hacer en cualquier empresa es instalar los discos en RAID. Básicamente existen tres tipos de raid que se suelen usar:

- RAID 0: no tiene redundancia de datos, junta varios discos con el fin de mejorar el rendimiento. Si tenemos 5 discos de 100GB, la cantidad de almacenamiento útil serán 5*100=500GB.
- RAID 1: tiene redundancia de datos, cada vez que se escribe en lugar de escribir en un disco se escribe la misma información en dos. Dado que toda la información está duplicada en ambos discos en caso de que un disco falle el otro tiene toda la información. Si tenemos dos discos de 100GB, la cantidad util de información serán 100GB (se perderá la mitad del almacenamiento).
- RAID 5: tiene redundancia de datos, debe tener 3 o más discos de la misma capacidad para hacer un raid, y en uno de los discos se escribe información de redundancia para poder seguir funcionando en caso de que falle un disco. Acepta el fallo de un disco. Si tenemos 5 discos de 100GB, la cantidad de almacenamiento útil serán 4*100=400GB (es decir siempre perderemos un disco de capacidad).

Otro tipo de raid que existe (aunque se usa menos, es el RAID 6, es la misma filosofía que el raid 5 pero con dos discos de redundancia, esto hace que se permita el fallo de dos discos).

Luego se suelen usar mucho también tipos de raids unidos, por ejemplo son típicos el RAID 1+0 (a veces llamado RAID 10) o el RAID 5+0 (a veces llamado RAID 50). En la página http://es.wikipedia.org/wiki/RAID se puede profundizar sobre los tipos de RAID y su funcionamiento.

Con el fin de configurar raid en linux, lo que se tiene es una serie de módulos en el kernel, y utilidades para gestionarlos (por ejemplo mdadm).

En debian (etch 4.0 e incluso anteriores), los módulos del kernel vienen preconfigurados (raid*.ko) y sólo es necesario instalar el paquete mdadm (apt-get install mdadm).

Para crear un disco en RAID lo que tenemos que hacer es crear particiones, ponerlas de tipo FD y ejecutar el comando mdadm --create. Por ejemplo para crear un raid con 4 discos podríamos ejecutar

```
mdadm --create /dev/md0 --level=5 --raid-devices=4 /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1
```

Para ver el estado de los raid se puede ejecutar cat /proc/mdstat

Para que se arranque automáticamente configurar /etc/mdadm/mdadm.conf, en el ejemplo anterior:

```
cat >>/etc/mdadm/mdadm.conf >>EOF
#DEVICE /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1
ARRAY /dev/md0 name=nombre level=5 num-devices=4 devices=/dev/sdb1,/dev/sdc1,/dev/sdd1,/dev/sde1
EOF
```

Si se necesita copiar una tabla de partición a otro disco se puede ejecutar:

```
sfdisk -d /dev/sda | sfdisk /dev/sdb
sfdisk -d /dev/sda | sfdisk /dev/sdc
```

Logical Volume Manager (LVM2)

LVM2 es un sistema para gestionar espacios de discos.

En LVM2 hay tres conceptos a manejar, PV (volumen físico), VG (Grupo de volúmenes) y LV (volumen lógico). Más información en: http://es.wikipedia.org/wiki/Logical_Volume_Manager

Para gestionar los volumenes LVM2 se utiliza el comando lvm2, para ello instalar el con apt-get install lvm2

Como ejemplo en el raid anterior vamos a crear un par de volúmenes:

```
pvcreate /dev/md0
vgcreate nombrevolumen /dev/md0
lvcreate -L +100G -m data nombrevolumen
mkfs.ext3 /dev/nombrevolumen/data
lvcreate -L +200G -m data2 nombrevolumen
mkfs.ext3 /dev/nombrevolumen/data2
```

Se pueden ampliar despues con los comandos: vgextend y lvextend

Ejemplo para ampliar el data, creado anteriormente, sumandole 50G, se hace lo siguiente:

```
lvextend -L +50G /dev/nombrevolumen/data
resize2fs /dev/nombrevolumen/data
```

Snapshot

Una vez que se tiene instalado discos con LVM, y suponiendo que haya espacio disponible, hay una funcionalidad que se puede hacer con los discos que son los snapshot.

Un Snapshot es una "foto" de los contenidos del disco duro en un momento dado.

Un snapshot puede ser una forma de hacer un backup inmediato (parar servicios, desmontar discos, hacer el snapshot y volver a montar discos y servicios)

Para hacer un snapshot lo que se hace es crear un nuevo LV:

```
lvcreate -L 20G --name snap_data1 --snapshot /dev/nombrevolumen/data
mount /dev/nombrevolumen/snap_data1 /mnt/tmp
```

Para crear un snapshot no es necesario tener todo el espacio del disco a congelar, por que lo que se guardan son los cambios que se hacen sobre el disco. Esto hace que por cada acceso de escritura a disco se escriba en el snapshot

Para borrar el snapshot cuando ya no interese se ejecuta lvremove -f /dev/nombrevolumen/snap_data1

iSCSI

Instalar el iSCSI Enterprise Target en máquina servidora de discos.

Bajar el paquete iscsitarget-0.4.15.tar.gz y descomprimirlo en algún directorio (por ejemplo en /usr/src)

Bajar algunas cosas que se necesitan para compilarlo:

```
apt-get install build-essential linux-headers-2.6.18-4-686 libssl-dev
```

Compilar el iSCSI Enterprise Target

```
cd /usr/src/iscsitarget-0.4.15
make
make install
```

Crear configuracion del ietd.conf

```
cat >/etc/ietd.conf >>EOF
Target iqn.2007-10.local.prueba:sanprueba.1G
#       IncomingUser usuario password
        Lun 0 Path=/dev/disco,Type=fileio
EOF
```

Arrancar el iSCSI Target

```
/etc/init.d/iscsi-target start
```

Instalación de iSCSI Initiator

Ir a la máquina que va a ser cliente e instalar el paquete open-iscsi

```
apt-get install open-iscsi
```

descubrir el disco

```
iscsiadm -m discovery -t st -p 192.168.153.134
```

Añadirlo a la máquina

```
iscsiadm -m node -T iqn.2007-10.local.prueba:sanprueba.1G -p 192.168.153.134:3260 --login
```

En mi caso me crea el disco /dev/sdb (puedes ver los discos de la maquina con cat /proc/partitions)

Si cuando se arranque se quiere tener el disco será necesario cambiar el fichero /etc/iscsi/nodes// file y poner los siguientes valores en las líneas:

```
node.startup = automatic
[...]
node.conn[0].startup = automatic
```

Acceso autentificado

En servidor descomentar la linea que dice IncomingUser... en /etc/ietd.conf y reiniciar el servicio

En el cliente editar el fichero /etc/iscsi/nodes// file y poner las líneas:

```
node.session.auth.authmethod = CHAP
node.session.auth.username = usuario
node.session.auth.password = password
```

Reiniciar el servicio (suponiendo que se haya puesto el arranque del cliente en automático se autoconectará al disco)

Acceso de dos máquinas simultaneamente (GFS)

```
apt-get install gfs-tools redhat-cluster-modules-2.6.18-4-686

mkfs.gfs /dev/sdb1 -p lock_dlm -t cluster:gfs -j 2
```

Multipathing

Con esto podríamos acceder a un mismo disco por dos servidores distintos, por ejemplo tener un cluster de servidores y blablabla

Replicación de discos (DRBD)

DRBD es una utilidad que nos va a permitir tener una copia del disco duro pero a través de red (como si fuese un RAID-1 pero en otra máquina). DRBD va a imponer la limitación de que en cada momento sólo una máquina va a poder acceder al disco, el resto recibirán las réplicas, pero no tendrán acceso al disco.

Para instalar el DRBD será necesario ejecutar:

```
apt-get install drbd0.7-utils drbd0.7-module-source
apt-get install linux-headers-2.6.18-4-686
cd /usr/src/linux-source-2.6.18
tar xvfz ../drbd0.7.tar.gz
cd modules/drbd/drbd
make
make install
```

Para configurarlo modificar el fichero /etc/drbd.conf según el siguiente fichero:

```
cat >/etc/drbd.conf >>EOF
resource r0 {
  protocol C;
  incon-degr-cmd "echo '!DRBD! rpi on incon-degr' | wall ; sleep 60; halt -f";
  startup {
    # wfc-timeout 300;
    degr-wfc-timeout 120;
  }
  disk {
    on-io-error detach;
  }
  net {
    # timeout 60;
    # connect-int 10;
    # ping-int 10;
    # max-buffers 2048;
    # max-epoch-size 2048;
  }
  syncer {
    rate 10M;
    group 1;
    al-extents 257;
  }
```

```
  on server1 {
    device /dev/drbd0;
    disk /dev/nombrevolumen/data;
    address 192.168.1.1:7788;
    meta-disk internal;
  }
  on server2 {
    device /dev/drbd0;
    disk /dev/nombrevolumen/data;
    address 192.168.1.2:7788;
    meta-disk internal;
  }
}
EOF
```

Una vez instalado y configurado en ambos servidores es necesario arrancar en ambos servidores el "/etc/init.d/drbd start". Después forzar una primera replicación y formateado del disco. Para ello ejecutar:

Después en uno de los servidores forzar la replicación: drbdsetup /dev/drbd0 primary --do-what-I-say

replicacion podras ver la evolucion con:

```
Debian40:/# cat /proc/drbd
version: 0.7.21 (api:79/proto:74)
SVN Revision: 2326 build by root@debian40,2007-10-24 18:12:46
 0: cs:SyncSource st:Promary/Secondary ld:Consistent
    ns:1859760 nr:0 dw:57576 dr:1805708 al:23 bm:349 lo:3 pe:428 ua:847 ap:2
        [==================>.] sync'ed: 92.1% (158548/1966080)K
        finish: 0:00:20 speed: 7,574 (5,076) K/sec
```

Cuando acabe formatear el disco con: mke2fs -j /dev/drbd0

Cluster (heartbeat)

Un cluster nos permite tener una aplicación configurada para ejecutarse en dos máquinas, de forma que normalmente se ejecute en una, y en caso de que esa caiga se levante en la otra.

Como ejemplo supongamos que tenemos un servidor web, o un servidor de base de datos... necesitamos que este servicio tenga alta disponibilidad, para ello lo configuramos en un cluster. De esta forma se creará una especie de máquina virtual (una IP virtual) que contendrá el servicio. El servicio además deberá de tener un almacenamiento compartido (iscsi, drbd, un disco NFS o algo).

En el momento que se detecte la caida de un nodo, la IP, el almacenamiento (si es necesario) y el servicio se arrancará en el otro

Para hacer una configuración de Heartbeat v2, los ficheros que hay que configurar son los siguientes:

```
cat >/etc/heartbeat/ha.cf <<EOF
logfacility daemon
keepalive 2
deadtime 30
warntime 10
ucast eth0 192.168.1.2
#ucast eth1 192.168.2.2
auto_failback on
watchdog /dev/watchdog
#nombres nodos
node server1
node server2

crm yes

ping 192.168.1.1
respawn hacluster /usr/lib/heartbeat/ipfail

apiauth ipfail uid=hacluster
apiauth ccm uid=hacluster
apiauth ping uid=hacluster
apiauth default uid=hacluster

respawn hacluster /usr/lib/heartbeat/lrmd
EOF


cat >/etc/heartbeat/authkeys <<EOF
# IVAN: usar seguridad sha1
auth 2
#1 crc
# IVAN: clave para usar en cada maquina
2 sha1 ha_password
#3 md5 Hello!
EOF
chmod 600 /etc/heartbeat/authkeys

cat >/var/lib/heartbeat/crm/cib.xml <<EOF
 <cib id="cib">
   <configuration id="configuration">
     <crm_config id="crm_config">
       <cluster_property_set id="default">
         <attributes id="attributes">
```

```
          <nvpair id="symmetric_cluster" name="symmetric_cluster" value="true"/>
          <nvpair id="no_quorum_policy" name="no_quorum_policy" value="stop"/>
          <nvpair id="default_resource_stickiness" name="default_resource_stickiness" value="0"/>
          <nvpair id="stonith_enabled" name="stonith_enabled" value="false"/>
          <nvpair id="stop_orphan_resources" name="stop_orphan_resources" value="false"/>
          <nvpair id="stop_orphan_actions" name="stop_orphan_actions" value="true"/>
          <nvpair id="remove_after_stop" name="remove_after_stop" value="false"/>
          <nvpair id="short_resource_names" name="short_resource_names" value="true"/>
          <nvpair id="transition_idle_timeout" name="transition_idle_timeout" value="200s"/>
          <nvpair id="is_managed_default" name="is_managed_default" value="true"/>
          <nvpair id="suppress_cib_writes" name="suppress_cib_writes" value="false"/>
        </attributes>
      </cluster_property_set>
    </crm_config>
    <nodes id="nodes">
      <node id="00000000-0000-0000-0000-000000000001" uname="server1"/>
      <node id="00000000-0000-0000-0000-000000000002" uname="server2"/>
    </nodes>
    <resources id="resources">
      <group id="dnsdhcp_group">
        <primitive id="ip_resource" class="ocf" type="IPaddr" provider="heartbeat">
          <operations id="dnsdhcp_group_operations">
            <op id="monitorip1" name="monitor" interval="60" timeout="5"/>
          </operations>
          <instance_attributes id="instanceatributes">
            <attributes>
              <nvpair name="ip" value="192.168.153.136" id="ip_resource_par1"/>
              <nvpair name="netmask" value="24" id="ip_resource_par2"/>
              <nvpair name="nic" value="eth0" id="ip_resource_par3"/>
            </attributes>
          </instance_attributes>
        </primitive>
      </group>
    </resources>
    <constraints id="constraints">
      <rsc_location id="run_dnsdhcp_group" rsc="dnsdhcp_group">
        <rule id="pref_run_dnsdhcp_group" score="100">
          <expression attribute="#uname" operation="eq" value="debian40" id="ad72d73d-4abd-4426-ba1a-1e4dfe7d156a"/>
        </rule>
      </rsc_location>
    </constraints>
  </configuration>
  <status>
  </status>
</cib>


EOF
chown hacluster.haclient cib.xml
```

El servicio se arrancará con /etc/init.d/heartbeat start, y dejara su log en /var/log/syslog

Para poder monitorizar el cluster en un momento dado se usará el programa crm_mon

Para cambiar la configuración se puede usar el comando cibadmin (parece demasiado coñazo y yo no lo he probado)

También tiene una superherramienta gráfica de visualización del cluster /usr/lib/heartbeat/haclient.py, pero para ejecutarla necesita que antes instalemos el módulo phyton-gtk2 (la aplicacion corre en X windows, por lo tanto ejecutarla desde otra maquina que tenga xwindows con el comando "ssh -X ip /usr/lib/heartbeat/haclient.py"). Tiene buena pinta pero no la he conseguido ejecutar.

Links

- Instalando un target iSCSI
- Homepage de "iSCSI Enterprise Target Project"
- Configuracion de iSCSI en Linux

Ivan Ricondo (ubanov@hotmail.com) 2007