

mini-HOWTO DE CLUSTERING EN LINUX

Manual del proyecto "paKon" por César Ávila y Miguel Veguillas

OBJETIVO: Describir paso a paso la instalación y configuración de un *cluster openmosix* sobre un sistema *Linux Debian 3.0* con *nodos diskless*.

MATERIAL: A parte de la documentación aquí descrita, se ofrecen los ficheros de configuración, los programas y enlaces a URLs de interés.

INDICE:

- 1.- Instalación del sistema operativo mínimo y preparación de los nodos diskless.
- 2.- Preparando APT
- 3.- Instalando los servicios
 - ...3.1.- Introducción
 - ...3.2.- DHCP
 - ...3.3.- TFTP-hpa
 - ...3.4.- NFS
- 4.- Configurando los clientes
- 5.- Nodos diskless
- 6.- Compilación distribuida
- 7.- Instalando OpenMosix
 - ...7.1.- Parche para el kernel
 - ...7.2.- OpenMosixTools
 - ...7.3.- OpenMosixView
- 8.- Apéndice

Desarrollo:

1.- Instalación del Sistema Operativo Mínimo

Lo primero que necesitaremos será tener un PC que usaremos de "maestro" en el que instalar nuestro Linux, en este caso, Debian Woody 3.0

Realizaremos una instalación minimalista, es decir, sin entorno gráfico y con el menor número de paquetes posibles. Para información acerca de la instalación de Debian ver documentación en www.debian.org

Los archivos de esta instalación serán los que utilicen los clientes o "nodos" "diskless" (PC's sin disco duro) para arrancar y por lo tanto no tienen que ver con el tipo de instalación que vayamos a realizar en el pc maestro.

Creamos los siguientes directorios "/tftpboot/192.168.0.0"; la semántica de los nombres se entenderá más adelante y a continuación copiamos ahí los directorios bin, boot, dev, etc, home, initrd, lib, opt, root, sbin, tmp, usr, var a la carpeta anteriormente creada.

```
cp -a /bin /boot /dev /etc /home /initrd /lib /opt /root /sbin /tmp /usr /var --target-directory=/tftpboot/192.168.0.0
```

Naturalmente no son necesarios TODOS los directorios antes mencionados para que los nodos diskless arranquen, ni siquiera el contenido completo de cada uno de ellos, cada uno puede-debe personalizar el

contenido de los archivos de sistema de sus clientes añadiendo o suprimiendo contenido pero se trata de hacer una instalación lo más genérica posible y explicarlo de la forma más clara.

Ahora hay que crear las siguientes carpetas vacías:

```
mkdir /proc /mnt /cdrom /floppy
```

Nótese que los directorios pueden variar de una distribución a otra e incluso de una instalación a otra.

Con eso habremos creado una copia de nuestro sistema en el directorio /tftpboot/192.168.0.0 y será ahí donde modificaremos posteriormente la configuración que deben tener los nodos diskless como por ejemplo, servicios en el arranque, configuración de la red, etc.

2.- Preparando APT

Ahora ya podemos empezar a diferenciar los archivos de nuestro maestro de los de nuestros clientes. Para ello empezaremos a instalar los paquetes correspondientes a los servicios DHCP, TFTPd-hpa, etc., utilizando la potente herramienta APT.

Lo primero es obtener un fichero de "sources". Este es nuestro fichero [sources.list](#).

A continuación actualizaremos la base de datos de los paquetes accesibles con `apt-get update`

3.- Instalando servicios

3.1.- Introducción

Para realizar el cluster con openmosix, sólo se necesita que cada máquina tenga un kernel con el parche de openmosix y las openmosix-tools. Pero si además queremos hacer un cluster con nodos diskless, necesitaremos instalar varios servicios como son el servidor DHCP, el TFTP y el NFS.

El servidor DHCP asigna una IP a cada nodo en el arranque de la máquina y le dice de dónde tiene que cojer el fichero "[pxelinux.0](#)".

Con ese fichero se lee el fichero "[default](#)" que está en /tftpboot/pxelinux.cfg (directorio que hay que crear) y se especifica de dónde se carga el kernel y con qué parámetros. Nosotros usamos un kernel 2.4.21 con el parche de openmosix "[mosixkernel](#)". A continuación se inicia la transferencia del kernel a través del servidor TFTP, cuando se termina de cargar se ejecuta (tiene que tener compilada la opción de -kernel autoconfiguración-) y entonces carga a través de NFS, el sistema de ficheros que hay en /tftpbot/192.168.0.0

3.2.- DHCP

Lo primero es instalar el servicio: `apt-get install dhcp3-server`

A continuación hay que configurarlo. Para una amplia explicación sobre su funcionamiento ver documentación: `man dhcpd.conf`

De todas formas aquí está nuestro fichero de configuración [dhcpd.conf](#).

Este fichero de configuración está preparado para los nodos diskless, que arrancan por PXE gracias a los servicios DHCP, TFTP y NFS.

Para activar el servicio: `/etc/init.d/dhcp3-server restart`

3.3.- TFTP-hpa

Para instalarlo: `apt-get install tftp-hpa`

Hay dos maneras de ejecutarlo: a través del servidor inetd o como demonio aislado. A nosotros nos ha dado mejor resultado ejecutarlo como demonio aislado, pues había veces que por inetd no se activaba aunque dependerá de la configuración de cada uno.

Ejecutando el demonio en "stand-alone" no hay fichero de configuración simplemente añadir en el arranque del sistema la siguiente línea:

```
in.tftpd -ls /tftpboot
```

también hay que tener en cuenta que hay que quitar (o comentar) de `/etc/services` la línea correspondiente al puerto 69, sino el demonio in.tftpd no carga.

```
tftp 69/udp # Tftpd daemon
```

Ejecutándolo por inetd hace falta cargar el demonio portmap:

```
/etc/inet.d/portmap restart
```

3.4.- NFS

Normalmente, se instala siempre, pero sino es así, realizaremos `apt-get install nfs-common nfs-kernel-server`

Para configurar este servicio utilizaremos el fichero `/etc/exports`. En él se especifica qué directorios queremos exportar por NFS y a quién le damos permiso.

También hay que modificar el fichero `/etc/hosts.allow`. para permitir el acceso al sistema a ciertas máquinas a través de ciertos servicios.

4.- Configurando los clientes

En el directorio `/tftpboot/192.168.0.0` que antes copiamos la instalación mínima, se guardan los ficheros de configuración (y del sistema en general) de los nodos clientes. Por lo tanto, es ahí donde tendremos que configurar los clientes.

El primer fichero importante es `/tftpboot/192.168.0.0/etc/fstab` que indica el sistema de archivos que montan los clientes.

El segundo es la configuración de red de los clientes. Para que sea genérica, van por DHCP. El fichero está en `/tftpboot/192.168.0.0/etc/network/interfaces`

A continuación habría que "limpiar" los scripts de arranque, es decir, desactivar los servicios que no queramos tener en los clientes; un ejemplo claro es el servidor de DHCP que sólo debe estar en el maestro. Estos script están almacenados en `/tftpboot/192.168.0.0/etc/rcX.d` Siendo X el runlevel escogido por el administrador para el arranque.

Un servicio que resultó problemático fue `/etc/rcS.d/S39ifupdown` y es que temporalmente desactiva la tarjeta de red, y eso para un nodo diskless, que coje toda la información de la tarjeta de red es fatídico. Ese es uno de los servicios candidatos a eliminar :-)

5.- Nodos diskless

Siguiendo los pasos de los cuatro puntos anteriores ya dispondríamos de un pc maestro con la configuración necesaria para que arrancaran tantos nodos diskless, es decir, sin disco duro, como quisieramos.

Hasta ahora simplemente podrían ser varios ordenadores en red, ahora es el momento de empezar a desarrollar el cluster.

6.- *Compilación distribuída: distcc*

Aunque esto no tiene que ver con el cluster, nos vendrá muy bien para aprovechar el trabajo realizado hasta ahora, ya que con distcc repartiremos cualquier compilación (kernel, kde, etc.) entre todos los nodos de nuestro cluster disminuyendo así el tiempo de compilación.

Lo primero que necesitamos es bajarnos el paquete o bien de la web: distcc.samba.org o bien de este servidor distcc-2.8.tar.bz2.

Paso1: Descomprimos el fichero desde un directorio cualquiera.... por ejemplo desde el home de root

```
bunzip2 distcc-2.8.tar.bz2
tar -xvf distcc-2.8.tar
```

Paso 2: Tendremos en /root un directorio que se llama distcc-2.8 Ahora tendremos que ejecutar

```
cd distcc-2.8
./configure
make
make install
```

Paso 3: Con esto ya tendremos distcc instalado en el sistema, pero aún no se está ejecutando. Hay dos maneras de ejecutarlo, como comando local: distcc gcc -o hola.o -c hola.c o como demonio. La que mejor resultado da es la de demonio. Para instalarlo como servicio haremos:

```
cp ESTE-SCRIPT /etc/init.d/distccd
ln -s /etc/init.d/distcd /etc/rcX.d/S99distccd
```

Paso 4: Pero el demonio se ejecuta con el UID y GID del usuario distcc que si no está creado en la máquina tendremos que crear.

```
adduser --no-create-home --disabled-login distcc
```

Paso 5: Hay que repetir los pasos 1,2,3 y 4 en todas las máquinas que queramos que tengan posibilidad de realizar o ayudar a las compilaciones distribuidas, es decir en nuestro caso, en todos los nodos clientes.

Paso 6: Hay que exportar una variable llamada DISTCC_HOSTS desde el nodo que va a iniciar la compilación distribuída, en nuestro caso sólo será el maestro pero alguien puede querer hacerlo también en los nodos clientes en cuyo caso tendrá que repetir en ellos lo siguiente:

Hay que hacer una lista con los nombre de equipos que se prestan voluntarios para realizar compilaciones distribuidas. Esta lista puede ser por ips o por nombres siempre y cuando hayan sido previamente definidos en /etc/hosts y guardar la lista en la variable DISTCC_HOSTS y exportarla. Además, para que se guarden los cambios de un reinicio a otro, utilizaremos el fichero /etc/profile

```
echo "export DISTCC_HOSTS='localhost nombre1 nombre2 ip3 ip4'" >> /etc/profile
```

nombre1 y nombre2 son los nombre de los equipos clientes 1 y 2. Esos nombres también están definidos en /etc/hosts

ip3 e ip4 son las ips de los equipos 3 y 4 donde también está instalado distcc

Usandolo: Ya sólo queda aprovechar las ventajas de la compilación distribuida por ejemplo:

```
cd /usr/src/linux/ && make dep modules modules_install bzImage -j<NºdeClientesAutilizar>  
CC=distcc
```

7.- Instalando Openmosix

Se trata de migrar procesos de una máquina a otra para disminuir el tiempo de ejecución de los programas. Eso se consigue gracias al parche para el kernel y a las OpenMosixTools disponibles en la web de [Openmosix](#) o desde aquí: [patch kernel for openMosix-2.4.21-1.bz2](#) [OpenMosixTools](#).

7.1.- Parche para el kernel

Una vez descargado el parche para la versión de nuestro kernel (aquí está la 2.4.21) ejecutamos:

```
mv /root/patch kernel for openMosix-2.4.21-1.bz2 /usr/src/linux-2.4.21  
cd /usr/src/linux-2.4.21  
bunzip2 patch kernel for openMosix-2.4.21-1.bz2  
cat patch kernel for openMosix-2.4.21-1 | patch -Np1
```

Este kernel le utilizaremos para arrancar tanto el maestro como los nodos diskless. Hay que copiarlo en /boot para el maestro y en /tftpboot/pxelinux.cfg/mosixkernel.

7.2.- OpenMosixTools

Una vez descargado el paquete adecuado para nuestra versión del kernel ejecutamos en el maestro y en uno de los clientes arrancados:

```
dpkg -i openmosix-tools-0.3.4-1_i386.deb
```

A continuación hay que comprobar que el servicio en /etc/init.d/openmosix está cargado en el arranque del maestro y en el del cliente, es decir,

que hay un enlace simbólico en /etc/rcX.d/S99openmosix a /etc/inid.d/openmosix ;sino lo hay, ejecutamos

```
ln -s /etc/inid.d/openmosix /etc/rcX.d/S99openmosix
```

y otro en /tftpboot/192.168.0.0/etc/rcX.d/S99openmosix a ../init.d/openmosix ; sino lo hay ejecutamos

```
ln -s ../init.d/openmosix /tftpboot/192.168.0.0/etc/rcX.d/S99openmosix (esta se ejecuta desde /tftpboot/192.168.0.0/etc/rcX.d/)
```

7.3.- OpenMosixView

Con este paquete podremos configurar y monitorizar nuestro cluster openmosix. Permite asignar prioridades a cada nodo según sus recursos libres, realiza gráficas de estadísticas, monitoriza el estado de los procesos en tiempo real; en definitiva una buena herramienta de administración para nuestro cluster.

Lo primero que tenemos que hacer es bajarnos el paquete: o bien de la web oficial www.openmosixview.com o de aquí: [openmosixview](#).

El segundo es un rpm para Red Hat 7.2 por lo que necesitaremos la herramienta "alien" para convertir el

rpm a un .deb y poder así instalarlo sin necesidad de tener que compilar.

```
apt-get install alien
alien -d openmosixview-1.5-redhat72.i386.rpm
dpkg -i openmosixview_1.5-1_i386.deb
```

Así ya tenemos instalado el paquete, pero para su correcto funcionamiento hace falta habilitar conexión ssh entre los nodos sin contraseña. Par ello ejecutamos:

```
ssh-keygen -t dsa -N "" -f /root/.ssh/id_dsa
cp -a /root/.ssh "/tftpboot/192.168.0.0/root/"
cp /root/.ssh/id_dsa.pub "/tftpboot/192.168.0.0/root/.ssh/authorized_keys"
cp /root/.ssh/id_dsa.pub "/root/.ssh/authorized_keys"
chmod 644 "/tftpboot/192.168.0.0/root/.ssh/authorized_keys"
chmod 644 "/root/.ssh/authorized_keys"
/etc/init.d/ssh start
echo -n "*" ">> "/tftpboot/192.168.0.0/root/.ssh/known_hosts"
cat /etc/ssh/ssh_host_rsa_key.pub >> "/tftpboot/192.168.0.0/root/.ssh/known_hosts"
echo -n "*" ">> "/root/.ssh/known_hosts"
cat /etc/ssh/ssh_host_rsa_key.pub >> "/root/.ssh/known_hosts"
cp -a /etc/ssh "/tftpboot/192.168.0.0/root/"
```

Sobre un servidor X ejecutamos openmosixview y ya podemos empezar a administrar nuestro cluster.

8.- Apéndice

Esta es la versión 0.1 de este mini-howto, estoy seguro que habrá erratas en alguna sintaxis pero creo que las ideas de los que hay que hacer están claras. No obstante, se ampliará la documentación aquí descrita con más ejemplos y algunas correcciones.

Nota por César: Dedicado a mi novia, la chica que más quiero... Cris.