UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA
DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

Máster en Ingeniería de Computadores y Redes

Trabajo Fin de Máster

Instalación, configuración y evaluación de un servidor web de alta disponibilidad con equilibrado de carga

Autor: *Manuel López Pérez*

Directores: *Pedro Juan López Rodríguez*

María Elvira Baydal Cardona

2015/2016

Resumen

En el presente documento se explican las bases del montaje de un servidor web de alta disponibilidad con equilibrado de carga. El servidor se implementará mediante un clúster de máquinas virtuales en el cual se instalará y configurará todo lo necesario para ofrecer la alta disponibilidad y el equilibrado de la carga en los nodos servidores. Para finalizar el trabajo se realizará una verificación del funcionamiento del servidor configurado. El trabajo se centra en la instalación, configuración y evaluación de un servidor web de alta disponibilidad.

Palabras clave: Linux, clúster, servidor web, alta disponibilidad, Ubuntu 14.04, equilibrado de carga.

Abstract

This document explains the basics to implement a high availability web server with load balancing are explained. The server will be based on a cluster of virtual machines. In this cluster, we will install and configure everything necessary to provide high availability and load balancing on the server nodes. In addition, several tests will be performed to check the behavior of the server. This work focuses on installation, configuration and evaluation of a high availability web server.

Keywords: Linux, cluster, web server, high availability, Ubuntu 14.04, load balancing.

Índice de contenidos

Índice de figuras

1. Introducción

1.1. Motivación

En los últimos años el crecimiento en la demanda de los servicios de Internet ha provocado que en muchos casos un único servidor web sea incapaz de atender el volumen de peticiones de los clientes. Esto ha motivado un cambio en la arquitectura de los servidores, que con frecuencia han pasado a implementarse mediante clústeres de computadores como una buena alternativa coste/prestaciones.

Los sistemas informáticos de las entidades públicas y privadas son una pieza imprescindible para su correcto funcionamiento. Cada vez más empresas y organismos públicos confían procesos de negocio críticos a sistemas informáticos para mejorar la productividad y disponer de esta información crítica en un tiempo mínimo. Por esa razón se debe dotar al sistema de alta disponibilidad.

El acceso al clúster suele realizarse a través de un nodo maestro o director, que se encarga de repartir las peticiones de los clientes entre un conjunto de servidores que son los que realmente implementan el servicio demandado. Los clientes ven este clúster como si fuera un único solo servidor.

1.2. Objetivo

El objetivo fundamental que persigue este Trabajo Fin de Máster (en adelante TFM) es la instalación, configuración y evaluación de un servidor web basado en un clúster de computadores que recibirá peticiones tanto HTTP como HTTPS.

El sistema estará formado por 8 nodos distribuidos de la siguiente manera:

- ***master* y *standby*:** nodos maestros. Están duplicados para conseguir alta disponibilidad, componiendo un servicio activo/pasivo. Estos nodos forman el *front-end* del sistema. Hacia el exterior (Internet), ofrecen el servicio web en el puerto 80 y en el 443 de una dirección IP virtual, asociada al nodo que en cada momento esté activo. Además de las funciones propias del nodo maestro de un clúster, también se ocupará de distribuir la carga, repartiendo las peticiones HTTP y HTTPS recibidas entre los servidores reales del *back-end*.

- **server1, server2 y server3:** servidores web reales. Estos nodos forman el *back-end* del sistema. Tienen instalado un servidor apache que atiende las peticiones que les deriva el nodo maestro activo.

- **NFS:** servidor de almacenamiento. Es un servidor NFS. Las páginas web proporcionadas por los servidores reales están almacenadas aquí.

- **bdserver1 y bdserver2:** nodos de bases de datos replicados en forma de maestro/esclavo que ofrecen acceso a los servidores web.

El esquema del clúster quedará como en la siguiente figura:

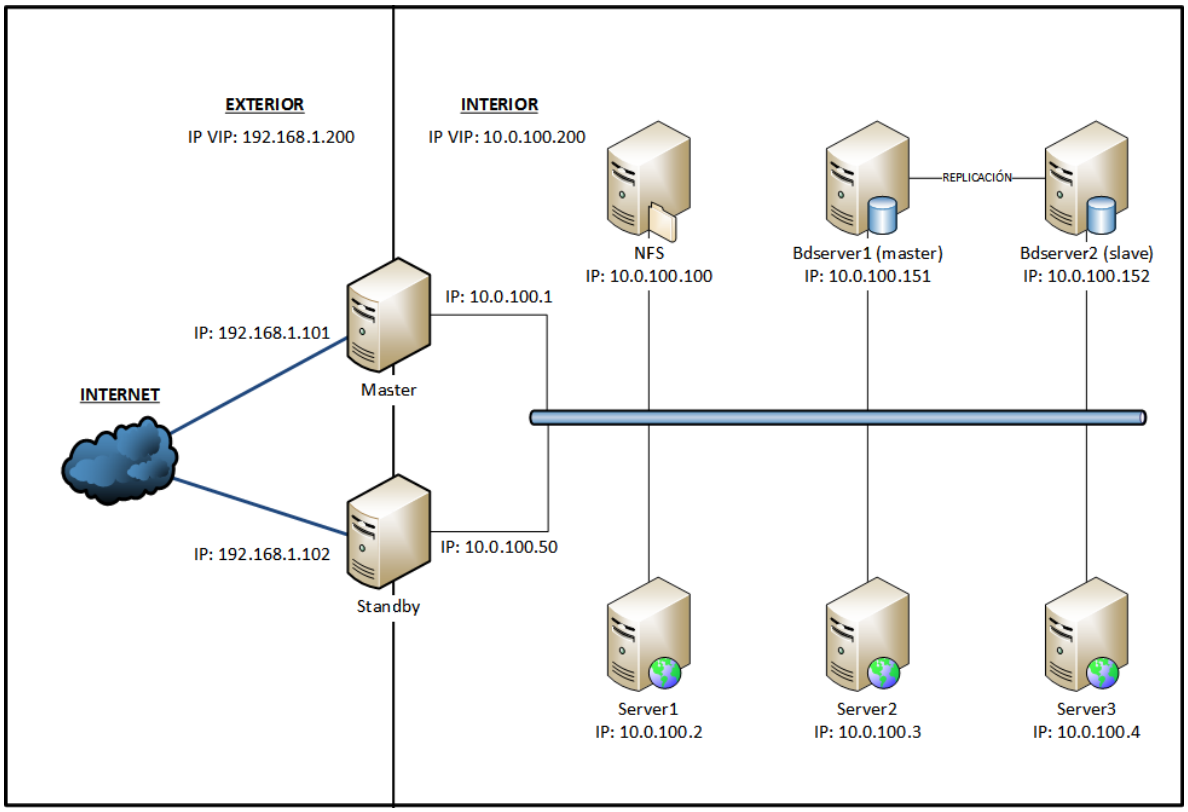

Figura 1. Esquema final del clúster.

Como podemos observar en la figura anterior, el acceso de las peticiones se hace a través de una IP VIP (Virtual IP, en adelante VIP). Posteriormente, el nodo maestro será el encargado de distribuir las peticiones entre los nodos servidores web reales. El clúster estará formado por dos redes:

- La **red interna** corresponde a la red del clúster. A través de esta red se comunican todos los nodos del clúster: los nodos maestros y los nodos servidores web, la base de datos y el nodo de almacenamiento. Los nodos maestros también se comunican entre sí para monitorizar su funcionamiento.

- La **red externa** permite el acceso de las máquinas del clúster a Internet.

1.3. Materiales utilizados

El clúster se ha desplegado utilizando un entorno de virtualización. Si bien un clúster real estaría compuesto de máquinas físicas, un entorno virtualizado ofrece una gran flexibilidad para la realización de pruebas de una forma muy económica. En concreto, se utilizará una infraestructura virtual basada en hypervisores de Oracle VM VirtualBox (versión 5.0.20).

El sistema operativo usado para la instalación de los nodos que constituyen el clúster de este trabajo será de la familia Ubuntu, en concreto, se instalará *Ubuntu Server 14.04.4 LTS (Long term Support)*, versión estable y gratuita.

Las máquinas virtuales se lanzarán desde un portátil con el programa VirtualBox. El host anfitrión es un portátil que tiene las siguientes características:

- **SO:** Windows 10 Home (64 bits)
- **RAM:** 8 GB a 1333 MHz
- **Disco duro:** 500 GB
- **Microprocesador:** Intel i3-3110M a 2.40 GHz

Estas máquinas se ejecutan en el portátil debido a que no se dispone del hardware para el montaje de un clúster real. Por ese motivo, debido a la limitación de recursos, el clúster implementado incluirá un número reducido de nodos servidores. Este clúster virtual se podría luego migrar a un clúster real con un mayor número de nodos servidores sin demasiado esfuerzo.

2. Clúster

Un clúster de computadores es un conjunto de varios servidores que se configuran e instalan para trabajar como si fuesen uno solo. Es decir, está formado como un conjunto de ordenadores que se unen mediante una red de alta velocidad, de tal forma que exteriormente el conjunto se ve como un único ordenador, mucho más potente que los ordenadores comunes. Estos sistemas han evolucionado mucho desde su primera aparición. En la actualidad se pueden construir distintos tipos de clústeres.

Otra de sus principales ventajas es que no es necesario que los equipos que lo integren sean iguales a nivel hardware ni que dispongan del mismo sistema operativo, lo que permite reciclar equipos que se encontraban anticuados o en desuso y rentabilizar su uso mediante un clúster de servidores. Es decir, los clústeres pueden ser heterogéneos tanto a nivel hardware como a nivel software.

Figura 2. Ejemplo de clúster.

Clasificación de los clústeres:

Podemos realizar la clasificación de los clústeres en función de varios criterios, pero todos ellos relacionados con los servicios que ya hemos mencionado. Atendiendo a estas características hablamos de tres tipos de clústeres:

- **Clústeres de alta productividad (HT o *High Throughput*).** En estos sistemas el objetivo central de diseño es la ejecución de múltiples tareas.

- **Clústeres de alta disponibilidad (HA o *High Availability*).** Con estos clústeres se busca dotar de disponibilidad y confiabilidad a los servicios que ofrecen. Para ello se utiliza hardware replicado, de modo que al no tener un único punto de fallo se garantiza la disponibilidad del sistema. Por otra parte, incorporan software de detección y recuperación ante fallos, con objeto de hacer confiable el sistema para su uso.

- **Clústeres de alto rendimiento (HP o *High Performance Clúster*).** Este tipo de sistemas ejecutan tareas que requieren de una gran capacidad de cálculo o del uso de grandes cantidades de memoria. La red de interconexión entre servidores ofrece muy altas prestaciones, por lo que son adecuadas para la ejecución de aplicaciones paralelas.

Estos tipos de clústeres se pueden combinar para que las empresas o centros científicos puedan satisfacer sus necesidades. Un ejemplo de esto, son los clústeres de infraestructuras comerciales que conjugan la alta disponibilidad con la alta eficiencia.

Componentes de los clústeres:

Para que un clúster funcione necesita de una serie de componentes, los cuales como hemos mencionado con anterioridad pueden tener diversos orígenes. Entre estos componentes citaremos:

- **Nodos:** es el nombre genérico que se dará a los computadores que forman el clúster. Generalmente son máquinas preparadas para ser montadas en rack, de forma que se facilita construcción compacta del clúster.

- **Sistema operativo:** podemos utilizar cualquier sistema operativo que sea multiproceso y multiusuario.

- **Red de interconexión:** es necesario que los distintos nodos de nuestra red estén conectados entre sí. Para ello podemos utilizar una red Ethernet o incluso otros sistemas de alta velocidad, como Infiniband.

- **Middleware:** es el nombre que recibe el software que se encuentra entre el sistema operativo y las aplicaciones. Su objetivo es que el usuario del clúster tenga la sensación de estar frente a un único superordenador ya que provee de una interfaz única de acceso al sistema. Mediante este software se consigue optimizar el uso del sistema y realizar operaciones de equilibrado de carga, tolerancia de fallos, etc.

- **Sistema de almacenamiento:** cuando trabajamos con clústeres podemos hacer uso de un sistema de almacenamiento interno de los equipos o bien recurrir a sistemas de almacenamiento más complejos, que proporcionarán una mayor eficiencia y disponibilidad de los datos, como son los dispositivos NAS (*Network Attached Storage*) o las redes SAN (*Storage Area Network*).

Todos estos sistemas utilizan protocolos de comunicación y servicios para el funcionamiento e interconexión de todos los nodos del clúster.

3. Instalación y configuración

3.1. Características de los nodos

Los nodos del clúster estarán constituidos de la siguiente forman:

- **Nodos *master* y *standby*:**
  - SO: Linux Ubuntu Server (64 bits)
  - RAM: 512 MB
  - Disco duro: 8 GB
  - Interfaces de red:
    - Adaptador 1 conectado a la Red _NAT.
    - Adaptador 2 conectado a la red interna.

- **Nodos *server1*, *server2* y *server3*:**
  - SO: Linux Ubuntu Server (64 bits)
  - RAM: 256 MB
  - Disco duro: 8 GB
  - Interfaz de red:
    - Adaptador 1 conectado a la red interna.

- **Nodo NFS:**
  - SO: Linux Ubuntu Server (64 bits)
  - RAM: 256 MB
  - Disco duro: 10 GB
  - Interfaz:
    - Adaptador 1 conectado a la red interna.

- **Nodos bdserver1 y bdserver2:**
  - SO: Linux Ubuntu Server (64 bits)
  - RAM: 512 MB
  - Disco duro: 10 GB
  - Interfaz de red:
    - Adaptador 1 conectado a la red interna.

Como podemos observar los nodos no tienen asignados muchos recursos debido a que se ejecutan de manera virtualizada en el portátil. Si se implementaran en un clúster real tendrían más recursos asignados.

3.2. Servicios utilizados

Los servicios que tendrán los nodos del clúster para ofrecer el servicio web son los siguientes:

Master y Standby:

- **Servidor DHCP:** el servidor asignará las direcciones IP dinámicamente a los servidores web del clúster.

- **Servidor PXE:** es un entorno para arrancar e instalar la imagen de un sistema operativo en los ordenadores a través de una red.

- **HAProxy:** es un servidor proxy de alta disponibilidad para servicios TCP y HTTP. Su uso más común es para mejorar el rendimiento y la fiabilidad de un entorno servidor mediante la distribución de la carga de trabajo a través de múltiples servidores (por ejemplo en aplicaciones de web, correo electrónico, base de datos). HAProxy ofrece las siguientes características:

    - Proxy TCP: puede aceptar conexiones TCP a través de un socket a la escucha, conectarse a otro servidor y permitir que la información fluya entre los dos servidores.
    - Proxy inverso HTTP: se presenta como un servidor que recibe peticiones HTTP a través de conexiones aceptadas y pasa las solicitudes a los servidores.
    - SSL *terminator/initiator/offloader*: para aligerar el trabajo de los nodos servidores.
    - Normalizador TCP: sirve para proteger las pilas TCP de los ataques y optimizar los parámetros de conexión con los clientes.
    - Normalizador HTTP: sirve para que solo las solicitudes válidas pasen a través del servidor evitando que se produzcan ataques basados en protocolo HTTP.
    - Herramienta de fijación HTTP: sirve para modificar/fijar/añadir/quitar/reescribir la dirección URL o el encabezado de cualquier petición o respuesta.
    - Encaminamiento basado en el contenido: permite considerar cualquier elemento de la solicitud para decidir a qué servidor se le remitirá la información o la conexión.
    - Equilibrador de carga: equilibra la carga de las peticiones, tanto TCP como HTTP, entre los nodos servidores.
    - Regulador del tráfico: protege a los nodos servidores contra sobrecargas, ajusta las prioridades de tráfico basado en el contenido e incluso transmite dicha información a las capas inferiores y a los componentes de la red exterior mediante el marcado de los paquetes.
    - Protección contra ataques de denegación de servicio (DDoS) y abuso de servicio: mantiene una gran cantidad de información por dirección IP, URL, cookies, etc. para detectar posibles ataques y actuar en consecuencia.
    - Punto de observación para solucionar problemas de red: debido a la precisión de la información almacenada en los registros, se utiliza a menudo para detectar y resolver problemas relacionados con la red.
    - HTTP *compression offloader*: puede comprimir las respuestas que no se comprimieron por el servidor, reduciendo así el tiempo de carga de la web para los

clientes con conectividad pobre o con alta latencia, como por ejemplo las redes móviles.

- **Keepalived:** nos ofrece una solución de alta disponibilidad mediante el uso del protocolo VRRP. Dicho protocolo permite compartir una dirección IP virtual (VIP) entre dos máquinas. Inicialmente, la dirección VIP estará asignada al nodo master. En caso de caída de este nodo, el paquete keepalived configurará la dirección VIP en el nodo *standby*, asegurando así un funcionamiento ininterrumpido.

Server1, Server2 y Server3:

- **Apache:** es un servidor web HTTP de código abierto que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.
- **PHP:** paquete para la ejecución de código php en el servidor web.
- **Wordpress:** es un sistema de gestión de contenidos enfocado a la creación de cualquier tipo de sitio web.
- **HyperDB:** es un plugin de expansión para wordpress que le permite conectarse a varios servidores de bases de datos.
- **Mysql-client:** es el programa cliente de Mysql que permite conectarse a bases de datos mysql-server tanto local como remotamente.

NFS:

- **Servicio NFS:** es un protocolo que permite acceso remoto a un sistema de archivos a través de la red.

Bdserver1 y Bdserver2:

- **Mysql-server:** es el programa servidor de Mysql que permite crear y ofrecer remotamente las bases de datos que contiene.

Todos los nodos tendrán instalado el servidor SSH para permitir el acceso remoto.

3.3. Implementación del clúster

En este apartado del TFM procederemos con la instalación de los sistemas y sus servicios, configurándolos en cada uno de los nodos que compondrán el clúster. La estrategia a seguir será crear una imagen única de la instalación básica que queramos que tengan dichos nodos y clonarla o replicarla en tantos nodos servidores web como tengamos (en nuestro caso tendremos sólo 3 nodos servidores web). El proceso a seguir sería instalar el sistema en el nodo "Master" tal y como queremos que quede en los servidores, crearemos una imagen del sistema generando así el nodo "modelo" a clonar. No se hará mucho hincapié en la instalación de los sistemas del clúster.

En todas las maquinas una vez instalado el sistema se habilitará el usuario "root", con la misma contraseña en todos los sistemas, mediante la siguiente orden:

sudo passwd root

Una vez habilitado este usuario en los sistemas, accederemos con él. Con este usuario podremos hacer las instalaciones y configuraciones necesarias sin tener que utilizar el prefijo sudo delante de la orden a ejecutar, facilitando así el proceso.

En la aplicación VirtualBox procederemos a crear las máquinas virtuales master, NFS, server1, server2 y server3 tal y como las hemos descrito en el apartado 3.1. Seguidamente crearemos una red NAT (Red_NAT1) que será utilizada para la IP VIP y para permitir el acceso a internet de los nodos master y standby. En dicha red también crearemos los reenvíos de puertos asociados a nuestra IP VIP para que los servicios que ofrecen sean accesibles desde el exterior.

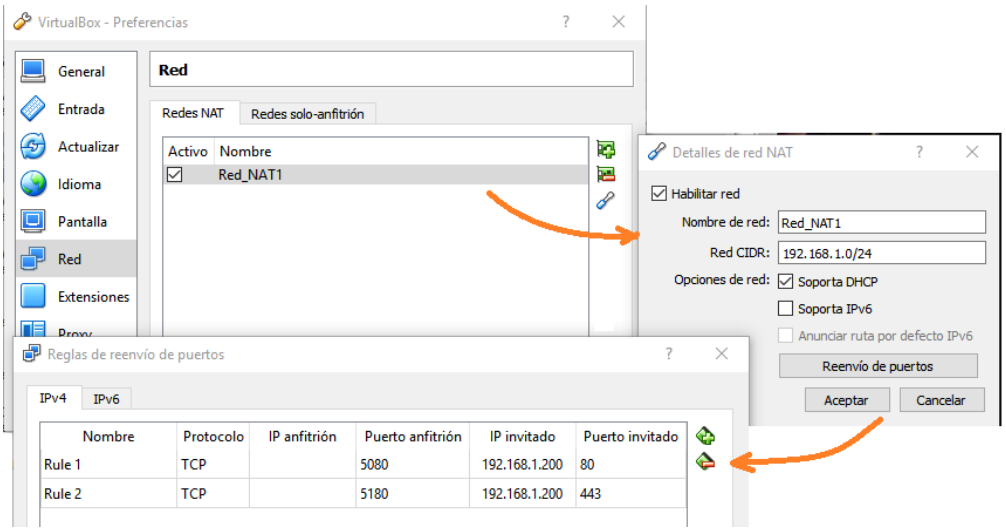

Figura 3. Configuración de IP VIP.

- **Regla1 (para HTTP):**
  - Protocolo: TCP
  - IP anfitrión: la que tenga el equipo local
  - Puerto anfitrión: 5080
  - IP invitado: 192.168.1.200
  - Puerto invitado: 80

- **Regla2 (para HTTPS):**
  - Protocolo: TCP
  - IP anfitrión: la que tenga el equipo local
  - Puerto anfitrión: 5180
  - IP invitado: 192.168.1.200
  - Puerto invitado: 443

3.3.1. Instalación del nodo NFS e instalación y configuración del servicio NFS

En esta sección se describe la instalación del servidor de almacenamiento. Los parámetros de la instalación son los siguientes:

- **Nombre de la maquina:** nas
- **Particiones primarias:**
  - Sistema: 4 GB, punto de montaje "/", formato ext4
  - Almacenamiento: 6 GB, punto de montaje "/home", formato ext4
  - SWAP: espacio restante, formato "área de intercambio"
- **Configuración automática de la red:** Sin proxy.
- **Quitamos las actualizaciones automáticas.**
- **Instalación del cargador de arranque.**

Con el sistema ya instalado procederemos con la configuración. Instalamos el paquete para que realice las tareas de servidor NFS mediante la siguiente orden:

apt-get update
apt-get install rpcbind nfs-kernel-server

Ahora procederemos instalando el servidor SSH para poder acceder al sistema desde, por ejemplo, el nodo master. Ejecutaremos el siguiente comando:

apt-get install openssh-server

Una vez instalado configuramos el servidor SSH para que acepte conexiones remotas a la cuenta root. Para ello editaremos el archivo de configuración modificando la línea con la opción "PermitRootLogin" de la siguiente manera:

- /etc/ssh/sshd_config:

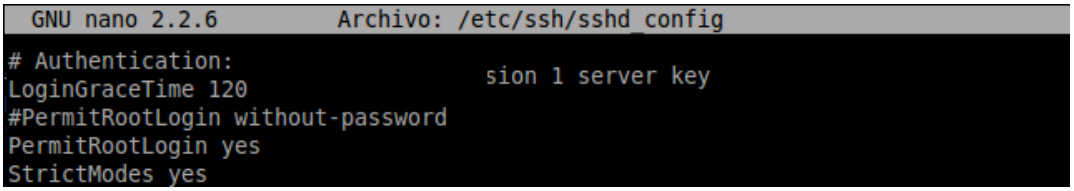

Figura 4. Archivo de configuración del servidor SSH.

Para que los cambios surtan efecto reiniciamos el servicio SSH ejecutando la siguiente orden:

service ssh restart

Como el servidor de almacenamiento tendrá una IP estática configuraremos la interfaz "eth0" con la siguiente configuración:

- /etc/network/interfaces:

```
  GNU nano 2.2.6        Archivo: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 10.0.100.100
        netmask 255.255.255.0
        gateway 10.0.100.1
        mtu 9000
        dns-nameservers 10.0.100.1
```

Figura 5. Archivo interfaces del servidor NFS.

Para que los cambios surtan efecto reiniciamos la interfaz de red "eth0" ejecutando las siguientes órdenes:

ifdown eth0 && ifup eth0

Vamos proceder con la configuración del servidor NFS para que exporte su directorio "/home" con la apariencia de "/nfs" para las máquinas internas de la red, en el cual tendrán permisos de escritura y lectura. Para ello añadiremos las siguientes líneas en los dos archivos que se indican a continuación:

- /etc/fstab:

```
  GNU nano 2.2.6              Archivo: /etc/fstab

# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=ef6cdd63-2172-487f-8fc5-88bcc5a14f56 /               ext4    errors=remoun$
# /home was on /dev/sda2 during installation
UUID=54f46749-6ea8-48cc-ac03-7d912e6b159b /home           ext4    defaults     $
# swap was on /dev/sda3 during installation
UUID=a17e9a06-882b-4ce2-8f1d-7b3db6ad0671 none            swap    sw           $
/home   /nfs    none    bind    0       0
```

Figura 6. Archivo fstab del servidor NFS.

- /etc/exports:

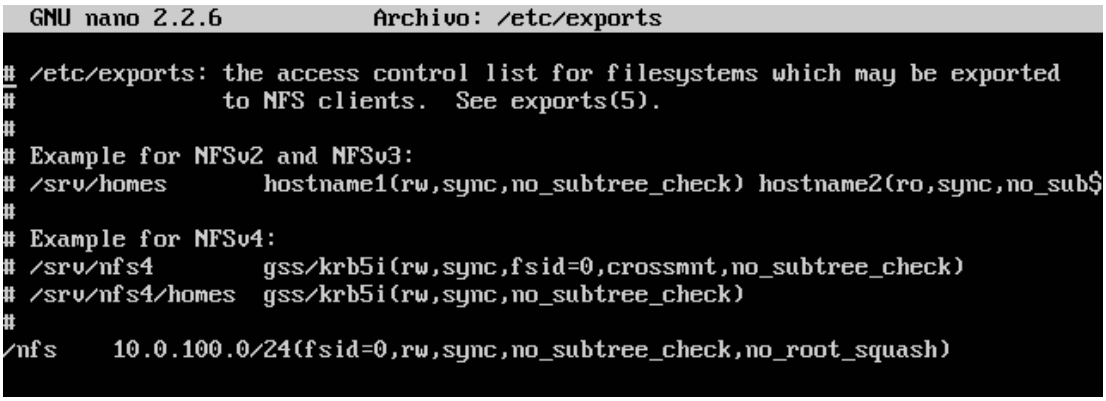

Figura 7. Archivo exports del servidor NFS.

Una vez modificado los archivos procedemos montando dicho directorio con las siguientes órdenes:

mkdir /nfs
mount /nfs

Para que los cambios surtan efecto reiniciamos el servidor NFS ejecutando la siguiente orden:

/etc/init.d/nfs-kernel-server restart

3.3.2. Instalación y configuración básica del nodo Master

Una vez instalado y configurado el nodo NFS procedemos con la instalación del nodo maestro. Los parámetros de la instalación son los siguientes:

- **Nombre de la máquina:** cluster1
- **Particiones primarias:**
  - Sistema: 8 GB, punto de montaje "/", formato ext4
  - SWAP: espacio restante, formato "área de intercambio"
- **Configuración automática de la red:** Sin proxy.
- **Quitamos las actualizaciones automáticas.**
- **Instalación del cargador de arranque.**

Se instalará y configurará el servidor SSH tal y como hemos hecho en el servidor de almacenamiento. Ya con el sistema instalado y con la configuración básica procederemos a prepararlo para clonarlo al resto de servidores. Procederemos instalando el cargador de arranque GRUBv2 lanzando las siguientes órdenes:

apt-get update
apt-get install grub2
grub-install /dev/sda

Configuraremos el paquete grub para que no busque automáticamente los sistemas instalados, indicando explícitamente el sistema a arrancar. Para ello desactivaremos la ejecución de algunos scripts ejecutando las siguientes órdenes:

chmod –x /etc/grub.d/10_linux
chmod –x /etc/grub.d/20_memtest86+
chmod –x /etc/grub.d/30_os-prober

También es necesario modificar el archivo "40_custom", indicando las opciones de inicio deseadas. En concreto, prepararemos un archivo que permitirá iniciar el sistema desde la única partición instalada, dicho archivo quedará de la siguiente forma:

- /etc/grub.d/40_custom:

```
  GNU nano 2.2.6            Archivo: /etc/grub.d/40_custom

#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
menuentry "Ubuntu-Part1" {
        set root=(hd0,1)
        linux /vmlinuz root=/dev/sda1
        initrd /initrd.img
}
```

Figura 8. Archivo de configuración 40_custom del grub.

Incrementaremos el tiempo de espera a 10 segundos antes del arranque del sistema, editando el archivo de configuración "grub" que se copiará en los servidores de web. Este incremento de tiempo permitirá que los servidores DHCP y NFS arranquen antes que los nodos servidores:

- /etc/default/grub:

```
  GNU nano 2.2.6            Archivo: /etc/default/grub                  Modificado

# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
```

Figura 9. Archivo de configuración del grub.

Vamos ahora a configurar la red del nodo master para que arranque por DHCP desde la interfaz "eth0", conectada al exterior, y con dirección IP fija en la interfaz "eth1", conectada a la red interna del clúster. Para ello modificaremos el archivo "interfaces" de la siguiente forma:

- /etc/network/interfaces:

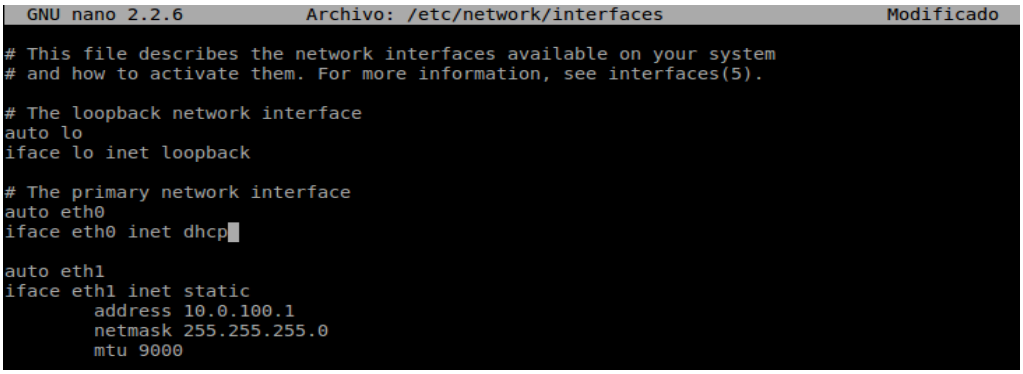

Figura 10. Primera configuración del archivo interface del servidor maestro.

Para que la configuración de las interfaces surja efecto reiniciaremos las interfaces lanzando las siguientes órdenes:

ifdown eth0 && ifup eth0
ifdown eth1 && ifup eth1

Ahora prepararemos el acceso a los servidores por SSH mediante clave pública. Generamos la clave y la copiamos sobre el archivo que contiene las claves autorizadas, que más tarde se copiarán sobre los nodos servidores, lanzando los siguientes comandos:

ssh-keygen
cp /root/.ssh/id_rsa.pub /root/.ssh/authorized.keys

También editaremos el archivo "hosts" añadiendo las parejas IP - nombre para la resolución de nombres. El nodo maestro y los nodos servidores tendrán el prefijo "cluster" (cluster1, cluster2,…), el nodo servidor de almacenamiento tendrá el nombre "nas", el servidor de backup del nodo maestro tendrá el nombre "standby" y los nodos servidores de bases de datos tendrán el prefijo "bdserver" (bdserver1, bdserver2,…). El archivo quedará de la siguiente forma:

- /etc/hosts:

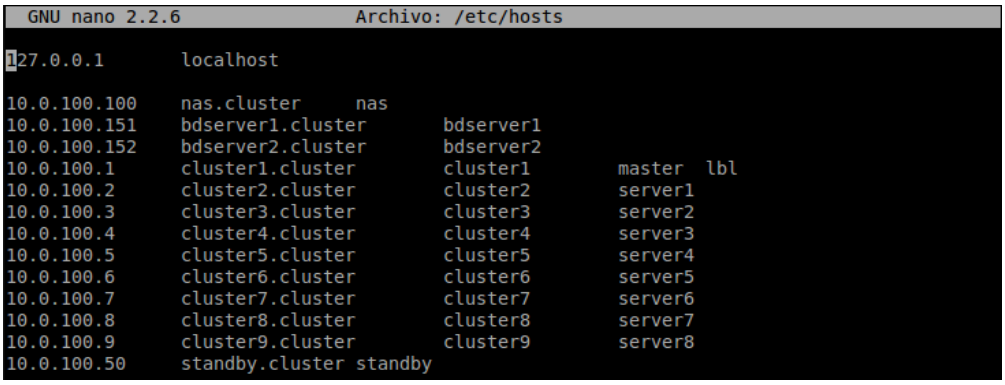

Figura 11. Archivo hosts en el nodo maestro.

Ya modificado el archivo hosts y la clave pública añadida, procederemos copiando los archivos al nodo NFS lanzando los siguientes comandos:

ssh nas "mkdir /root/.ssh"
scp /root/.ssh/id_rsa.pub nas: /root/.ssh/authorized.keys
scp /etc/hosts nas:/etc

Ahora instalamos el paquete cliente de NFS en el nodo maestro y posteriormente lo configuraremos para que monte automáticamente el directorio compartido por el servidor nas. Para ello ejecutaremos las siguientes órdenes:

apt-get install nfs-common
echo "nas:/nfs /nfs nfs auto,rsize=8192,wsize=8192 0 0" >> /etc/fstab
mkdir /nfs
mount /nfs

Una vez tenemos preparado el sistema, procederemos a copiarlo al servidor NFS para que los nodos servidores arranquen desde esté mediante PXE y posteriormente lo copien a su disco local. Para ello se han lanzado las siguientes órdenes:

mkdir /nfs/srv
cp –ax / /nfs/srv/

Explicación de las opciones de la orden cp utilizadas:

- **-a:** que equivale a las opciones -dR --preserve=all, para preservar los permisos del origen.
- **-x:** para que no cruce los límites del sistema de archivos montado.

Ahora haremos algunas modificaciones en los archivos que acabamos de copiar. Estos cambios los realizaremos desde el nodo maestro en los archivos del servidor NFS. En primer lugar como el sistema de archivos raíz de los nodos que arranquen por red estará ubicado en el servidor NFS, es necesario modificar el archivo fstab, sustituyendo su contenido por el siguiente:

- /nfs/srv/etc/fstab:

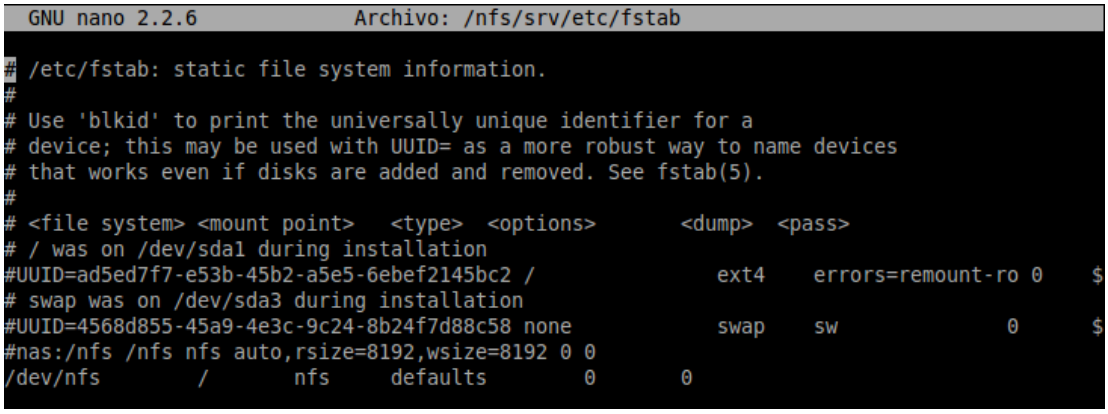

Figura 12. Archivo fstab para la ejecución de los sistemas por red.

Además, cuando los nodos arranquen por PXE, los interfaces de red ya estarán iniciados y configurados en el momento de montar el sistema raíz, por lo que no hay que hacer ninguna configuración adicional. Esto se consigue modificando el archivo "interfaces", eliminando la interfaz "eth1" y modificado la interfaz "eth0" para que adquiera la IP por DHCP.

- /nfs/srv/etc/network/interfaces:

```
  GNU nano 2.2.6         Archivo: /nfs/srv/etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
iface eth0 inet dhcp
```

Figura 13. Archivo interface para la ejecución de los sistemas por red.

3.3.3. Configuración del firewall e instalación de la interfaz gráfica del nodo Master

Como el nodo maestro debe permitir la conexión a la red exterior a los nodos servidores, debemos activar *ip-forwarding* y configurar correctamente el cortafuegos. Esto se puede conseguir añadiendo la siguiente secuencia de órdenes en el archivo "rc.local".

- /etc/rc.local:

```
  GNU nano 2.2.6               Archivo: /etc/rc.local

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
sysctl -w net.ipv4.ip_forward=1
iptables -P FORWARD ACCEPT
iptables --table nat -A POSTROUTING -o eth0 -j MASQUERADE

exit 0
```

Figura 14. Archivo rc.local que habilita el cortafuegos del servidor maestro y standby.

Y lo lanzaremos manualmente dándole permisos de ejecución ejecutando las siguientes órdenes:

chmod +x /etc/rc.local
./etc/rc.local

Para ayudarnos en la configuración de todos los nodos del clúster instalaremos un entorno gráfico para trabajar más cómodamente desde el nodo maestro. En este caso utilizaremos el entorno LXDE que consume pocos recursos, para ello también instalaremos un paquete con los drivers adecuados para la pantalla ofrecida por VirtualBox. Para la instalación se lanzarán las siguientes órdenes:

apt-get install virtualbox-guest-dkms
apt-get install lxde

Una vez instalado la interfaz gráfica quitamos la línea desplegable de usuarios de la pantalla de inicio editando el archivo "lxdm.conf" de la siguiente forma:

- /etc/lxdm/lxdm.conf

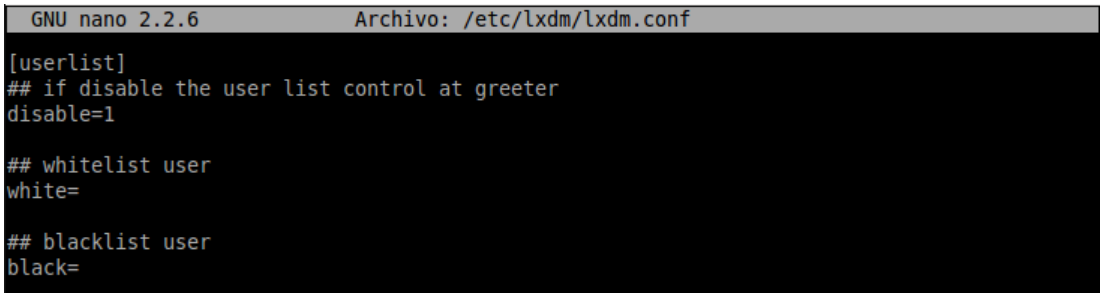

Figura 15. Archivo de configuración del entorno gráfico.

Reiniciamos el nodo maestro para que arranque con el entorno gráfico e iniciamos sesión como root, para proseguir con el TFM.

3.3.4. Configuración del servidor DHCP y la instalación del servicio PXE en el nodo Master

Como queremos que el nodo maestro se inicie antes que los nodos servidores tendremos que volver a modificar el archivo "grub" del maestro como se ha hecho con anterioridad, para que el sistema arranque más rápidamente que el de los nodos servidores, puesto que debe lanzar el servidor PXE.

Ahora instalaremos el servidor PXE en el nodo maestro, instalaremos el paquete que contiene el cargador a través de red y lo ubicamos en el directorio "/boot", lanzando las siguientes órdenes:

apt-get install syslinux
cp /usr/lib/syslinux/pxelinux.0 /boot/
mkdir /boot/pxelinux.cfg

También generaremos un archivo de configuración que indique la ubicación de los archivos que contienen el núcleo y el *ramdisk* inicial. Por otra parte una vez arrancados por red, los nodos montarán su sistema de archivos raíz en el servidor NFS.

- /boot/pxelinux.cfg/default

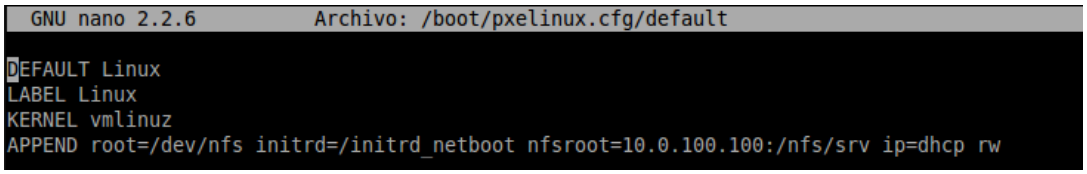

Figura 16. Archivo de configuración del PXE.

Seguidamente prepararemos un "*initial ramdisk*" que monte el sistema de archivos raíz en el servidor NFS. Para ello editaremos el archivo "initramfs.conf" modificando la línea "BOOT=" indicando que el inicio debe ser desde NFS. Previamente sacaremos una copia:

cp /etc/initramfs-tools/initramfs.conf /etc/initramfs-tools/initramfs.conf.m

Y el contenido del archivo debe ser:

- /etc/initramfs-tools/initramfs.conf:

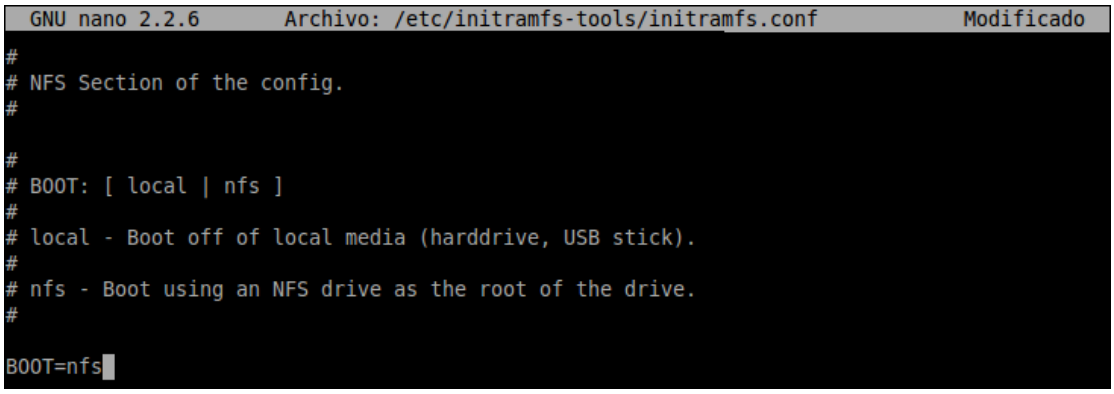

Figura 17. Archivo de configuración para el montage del sistema de archivos.

Seguidamente, generaremos el nuevo initial ramdisk con esta configuración y después restauraremos el archivo de configuración inicial lanzando la siguiente secuencia de órdenes:

/usr/sbin/mkinitramfs -o /boot/initrd_netboot
cp /etc/initramfs-tools/initramfs.conf.m /etc/initramfs-tools/initramfs.conf
ln -s /boot/vmlinuz-* /boot/vmlinuz
chmod 644 /boot/vmlinuz-*

Ahora prepararemos el servidor PXE utilizando la "dnsmasq", que incorpora un servidor DHCP, un servidor DNS y un servidor TFTP para el arranque por PXE. Instalaremos el paquete correspondiente y detendremos el servicio para configurarlo ejecutando las siguientes órdenes:

apt-get install dnsmasq
/etc/init.d/dnsmasq stop

Vamos a preparar dos configuraciones:

1. La primera, que utilizaremos en esta etapa de instalación del clúster, incorporará los servicios DNS, DHCP y PXE, mientras que
2. La segunda sólo incorporará DNS y DHCP.

Por lo tanto crearemos dos archivos de configuración:

- /etc/dnsmasq-dhcp-pxe.conf

```
 GNU nano 2.2.6          Archivo: /etc/dnsmasq-dhcp-pxe.conf

enable-tftp
tftp-root=/boot
dhcp-boot=/pxelinux.0
dhcp-range=10.0.100.2,10.0.100.50,infinite
log-dhcp
dhcp-option=26,9000
dhcp-option=3,10.0.100.1
dhcp-option=6,10.0.100.1
interface=eth1
read-ethers
```

Figura 18. Archivo de configuración dnsmasq que incorpora DNS+DHCP+PXE.

- /etc/dnsmasq-dhcp.conf

```
 GNU nano 2.2.6           Archivo: /etc/dnsmasq-dhcp.conf

dhcp-range=10.0.100.2,10.0.100.50,infinite
log-dhcp
dhcp-option=26,9000
dhcp-option=3,10.0.100.1
dhcp-option=6,10.0.100.1
interface=eth1
read-ethers
```

Figura 19. Archivo de configuración dnsmasq que incorpora DNS+DHCP.

La opción “read-ethers” nos permite tener control sobre las direcciones IP que proporciona el servidor DHCP a través del archivo “ethers”, que incluye las parejas MAC–IP para preasignar las direcciones IP. Dicho archivo quedará como se muestra a continuación:

- /etc/ethers:

```
 GNU nano 2.2.6              Archivo: /etc/ethers

08:00:27:01:01:02       10.0.100.2
08:00:27:01:01:03       10.0.100.3
08:00:27:01:01:04       10.0.100.4
08:00:27:01:01:05       10.0.100.5
08:00:27:01:01:06       10.0.100.6
08:00:27:01:01:07       10.0.100.7
08:00:27:01:01:08       10.0.100.8
08:00:27:01:01:09       10.0.100.9
```

Figura 20. Archivo ethers utilizado por dnsmasq.

Para poder optar entre una y otra configuración para cuando necesitemos instalar otros servidores o no, se han creado los dos siguientes *scripts*:

- /root/start-dhcp-pxe (ANEXO 1): Inicia el modo DHCP y PXE
- /root/start-dhcp (ANEXO 2): Inicia el modo DHCP

A estos archivos les damos permisos de ejecución y como tenemos que instalar los nodos servidores, iniciaremos en el modo DHCP+PXE:

chmod +x start-dhcp-pxe
chmod +x start-dhcp
./start-dhcp-pxe

3.3.5. Instalación de los nodos servidores y su configuración básica

Para facilitar instalación y configuración de los nodos servidores hemos creado los siguientes *scripts*:

- /root/psh.sh (ANEXO 3): envía una orden a todos los nodos servidores.
- /root/ppsh.sh (ANEXO 4): envía una orden a todos los nodos servidores, los nodos la ejecutan simultáneamente.
- /root/pcp.sh (ANEXO 5): copia un archivo en todos los nodos servidores.

Les daremos los permisos de ejecución a los *scripts* anteriores:

chmod +x /root/psh.sh
chmod +x /root/ppsh.sh
chmod +x /root/pcp.sh

Iniciaremos los servidores a clonar a través de sistema VirtualBox, como los hemos configurado para que arranquen por red como primera opción y van a encontrar un servidor PXE activo, veremos cómo obtienen una dirección IP, inician el cargador de arranque, el núcleo, etc. Cuando se inicien, los servidores tendrán en marcha un sistema cuya raíz está compartida por todos, que reside en el servidor NFS. Además, cada uno de ellos habrá obtenido la dirección IP que habíamos previsto. Seguidamente procederemos al particionado de los discos de los servidores. Como los discos de los servidores son idénticos al del maestro, vamos a volcar la tabla de particiones del nodo maestro en un archivo y lo propagaremos a los servidores. Para obtener una copia de la tabla de particiones del nodo maestro, teclearemos:

sfdisk -d /dev/sda > /nfs/srv/root/sda.out

Ahora tendremos que crear la tabla de particiones en los servidores. Para ello utilizaremos el *script* “psh.sh”:

./psh.sh "sfdisk -f /dev/sda < /root/sda.out"

Una vez creada la tabla procederemos formateando las particiones de arranque y del área de intercambio. Para ello utilizaremos el *script* “ppsh.sh” que ejecuta la orden en todos los nodos simultáneamente:

./ppsh.sh "mkfs –t ext4 /dev/sda1"
./ppsh.sh "mkswap /dev/sda3"
./ppsh.sh "swapon /dev/sda3"

Formateadas las particiones, procederemos con la clonación del sistema a los servidores. Cada servidor montará su disco local y copiará el sistema desde el raíz ubicado en el servidor NFS. Lanzaremos la siguiente serie de órdenes:

./ppsh.sh "mkdir /mnt/sda1"
./ppsh.sh "mount /dev/sda1 /mnt/sda1"
./ppsh.sh "cp -ax / /mnt/sda1"

Ahora realizamos algunos cambios en los archivos que acabamos de copiar. Para ello debemos crear en cada nodo servidor un archivo "fstab" que indique que la raíz del sistema se encontrará en el disco local, con una partición de swap y un servidor NFS en el clúster. El contenido de este archivo en cada servidor debería ser el siguiente:

- /etc/fstab:

```
 GNU nano 2.2.6                Archivo: /root/fstab

/dev/sda1       /       ext4    defaults        0       0
/dev/sda3       none    swap    auto,defaults   0       0
nas:/home       /nfs    nfs     auto,defaults   0       0
```

Figura 21. Archivo fstab de los servidores web.

Para ganar tiempo generamos el archivo en el nodo maestro y posteriormente lo difundiremos a los servidores haciendo uso del script "pcp.sh" que copia un archivo a todos los nodos del clúster:

./pcp.sh /root/fstab /mnt/sda1/etc

Adicionalmente, hay que configurar correctamente el interfaz de red de los servidores para que se inicie automáticamente y obtenga su dirección IP por DHCP. Para ello, el archivo de configuración de cada servidor debería tener el siguiente contenido:

- /etc/network/interfaces:

```
 GNU nano 2.2.6          Archivo: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
iface eth0 inet dhcp
```

Figura 22. Archivo interfaces de los servidores web.

Podemos hacer como en el caso anterior, generar el archivo en el nodo maestro y posteriormente difundirlo a los servidores:

./pcp.sh /root/interfaces /mnt/sda1/etc/network/interfaces

Seguidamente procedemos instalando el cargador de arranque en los servidores. En cada uno de los servidores hay que ejecutar la orden "grub-install":

./pcp.sh "grub-install --root-directory=/mnt/sda1 /dev/sda"

Tendremos que regenerar correctamente los archivos de configuración. Para ello, montaremos los directorios necesarios en la partición que estamos instalando:

./psh.sh "for i in /dev /dev/pts /proc /sys; do mount -B \$i /mnt/sda1\$i; done"

Ahora ejecutaremos las órdenes que consolidan la configuración, con la ayuda de chroot, que permite cambiar el punto de montaje del sistema de archivos raíz:

./psh.sh "chroot /mnt/sda1 dpkg-reconfigure grub2"
./psh.sh "chroot /mnt/sda1 update-grub"

Seguidamente modificaremos el nombre de los nodos servidores debido a que, al ser una copia del nodo maestro, tienen su mismo nombre. En cada nodo hay que generar el archivo "hostname" correspondiente. Desde el nodo maestro lanzaremos las siguientes órdenes:

ssh cluster2 "echo cluster2 > /mnt/sda1/etc/hostname"
ssh cluster3 "echo cluster3 > /mnt/sda1/etc/hostname"
ssh cluster4 "echo cluster4 > /mnt/sda1/etc/hostname"

Una vez terminada la instalación del sistema en los nodos servidores lanzaremos la utilidad "dnsmasq" en modo DHCP y reiniciaremos los servidores ejecutando la siguiente serie de comandos:

./start-dhcp
./psh.sh "umount /mnt/sda1"
./psh.sh reboot

Una vez se inicien los nodos servidores vamos a asignarle una IP estática en la interfaz "eth0" del nodo maestro y la reiniciamos para que surtan efecto los cambios realizados como se ha hecho con anterioridad.

- /etc/network/interfaces:

```
  GNU nano 2.2.6          Archivo: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.1.101
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 192.168.1.1
        mtu 9000

auto eth1
iface eth1 inet static
        address 10.0.100.1
        netmask 255.255.255.0
        mtu 9000
```

Figura 23. Configuración final del archivo interfaces del servidor maestro.

3.3.6. Instalación y configuración de los servicios Apache y PHP en los nodos servidores.

Una vez instalados los sistemas en los nodos servidores procederemos instalando el paquete "apache" y "php" en los mismos, lanzando la siguiente orden:

./psh.sh "apt-get update && apt-get install apache2"
./psh.sh "apt-get install php5"

Seguidamente modificaremos la configuración de los servidores para que la localización de los contenidos que sirven todos ellos esté ubicada en un espacio de almacenamiento compartido. En nuestro caso será /nfs/www. Para ello, hay que modificar la configuración en los ficheros indicados a continuación:

- /etc/apache2/sites-enabled/000-default.conf:

```
  GNU nano 2.2.6 Archivo: ...apache2/sites-enabled/000-default.conf

<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port t$
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /nfs/www
```

Figura 24. Archivo de configuración para el host virtual.

- /etc/apache2/apache2.conf:

```
  GNU nano 2.2.6       Archivo: /etc/apache2/apache2.conf

        AllowOverride None
        Require all denied
</Directory>

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /nfs/www/>
        Options Includes
        AllowOverride All
        Require all granted
</Directory>
```

Figura 25. Archivo de configuración del Apache.

Creamos el directorio en donde se ubicarán las páginas web con la siguiente orden:

mkdir /nfs/www

Generamos una sencilla página "index.php[1]" (ANEXO 6) que nos mostrará en cada momento qué servidor es el que nos está contestando. Finalmente reiniciamos el servicio de apache:

service apache2 restart

3.3.7. Instalación y configuración del servicio HAProxy en el nodo Master

Una vez configurado los nodos servidores como servidores web, procederemos instalando "HAProxy" en el nodo maestro para realizar el equilibrado de carga, lanzando las siguientes órdenes:

apt-get update
apt-get install haproxy

Antes de configurar el paquete "HAProxy", como nuestro clúster también responde peticiones HTTPS debemos generar un certificado. Para este fin utilizaremos el "*SSL Terminator*". En primer lugar vamos a crear un certificado autofirmado para "*.xip.io", que nos permite utilizar el mismo certificado aunque la IP de nuestro servidor cambie. Para crear el certificado lanzaremos la siguiente serie de órdenes:

openssl genrsa -out /etc/ssl/xip.io/xip.io.key 1024
openssl req -new -key /etc/ssl/xip.io/xip.io.key -out /etc/ssl/xip.io/xip.io.csr
openssl x509 -req -days 365 -in /etc/ssl/xip.io/xip.io.csr -signkey /etc/ssl/xip.io/xip.io.key -out /etc/ssl/xip.io/xip.io.crt

Los archivos creados son los siguientes:

- xip.io.csr: certificado
- xip.io.key: clave del certificado
- xip.io.crt: autoridades de certificación

En segundo lugar, una vez creados los certificados, necesitamos crear un archivo "*.pem" que contenga los anteriores archivos. Esto se consigue lanzando la siguiente orden:

cat /etc/ssl/xip.io/xip.io.crt /etc/ssl/xip.io/xip.io.key | sudo tee /etc/ssl/xip.io/xip.io.pem

Una vez creado el archivo de certificación procederemos configurando el "HAProxy" modificando el archivo "haproxy.cfg". En nuestro caso la configuración por defecto nos sirve. Añadiremos dos conexiones, una parte externa (*front-end*) y otra parte interna (*back-end*).

[1] Código obtenido de la asignatura del Máster de Ingeniería de Computadores y Redes: Configuración, Administración y Utilización de Clusters de PCs. Dpto. de Informática de Sistemas y Computadores (DISCA, Universidad Politécnica de Valencia). Valencia.

- /etc/haproxy/haproxy.cfg:

```
 GNU nano 2.2.6          Archivo: /etc/haproxy/haproxy.cfg

frontend http_front
        bind *:80
        bind *:443 ssl crt /etc/ssl/xip.io/xip.io.pem
        default_backend http_back

backend http_back
        mode http
        balance roundrobin
        option httpclose
        option forwardfor
        option httpchk HEAD / HTTP/1.1\r\nHost:localhost
        cookie JSESSIONID prefix
        server cluster2 10.0.100.2:80 cookie A check
        server cluster3 10.0.100.3:80 cookie A check
        server cluster4 10.0.100.4:80 cookie A check
        http-request set-header X-Forwarded-Port %[dst_port]
        http-request add-header X-Forwarded-Proto https if { ssl_fc }
```

Figura 26. Archivo de configuración de HAProxy.

Explicación de la sección externa (*front-end*):

- **frontend http_front:** especifica el nombre de la conexión.
- **bind *:80:** Recibe peticiones HTTP
- **bind *:443 ssl crt...:** Recibe peticiones HTTPS con certificado.
- **default_backend http-back:** deriva las peticiones entrantes al *back-end* del HAProxy.

Explicación de la sección interna (*back-end*):

- **backend http-back:** especifica el nombre de la conexión.
- **mode http:** HAProxy funciona en modo HTTP
- **balance roundrobin:** realiza un equilibrado "*round robin*". Reparte las conexiones de manera equitativa y siguiendo un orden.
- **option httpclose:** cierra las conexiones persistentes.
- **option forwardfor:** incluye la dirección IP del cliente en la cabecera enviada a los nodos servidores.
- **option httpchk HEAD / HTTP/1.1\r\nHost:localhost:** se utiliza para probar si los servidores web siguen respondiendo.
- **cookie JSESSIONID prefix:** añade a la *cookie* existente "JSESSIONID" el id de la sesión.
- **server cluster…cookie A check:** especifica un servidor *back-end* llamado cluster* y su IP privada. La persistencia se consigue mediantes cookies. La opción "check" sirve para comprobar si el servidor está en marcha atendiendo peticiones.
- **http-request set-header X-Forwarded-Port %[dst_port]:** se utiliza para que los nodos servidores sepan a qué puerto va dirigida la petición.
- **http-request add-header X-Forwarded-Proto https if { ssl_fc }:** se utiliza para que los nodos servidores sepan el protocolo que utiliza el cliente al conectarse.

Se ha configurado de esta manera para que nuestro servidor web acepte tanto peticiones entrantes HTTP como HTTPS escuchando los puertos 80 y 443. Se ha configurado para que las conexiones no sean persistentes y con balanceo "*round robin*". Si se implementara en un escenario real, el balanceo idóneo sería "*leastconn*" el cual selecciona el servidor web con menos carga en ese momento. Para que los cambios surtan efecto tendremos que reiniciar el servicio lanzando la siguiente orden:

/etc/init.d/haproxy restart

Posteriormente podemos comprobar la nueva página abriendo en el navegador la URL http://localhost:5080/index.php o https://localhost:5180/index.php. Recuérdese que los puertos locales 5080 y 5180 están redirigidos a las direcciones VIP de nuestro clúster.

3.3.8. Instalación y configuración de los nodos servidores de bases de datos

Una vez configurado el "HAProxy" procedemos con la instalación y configuración del servidor de base de datos "bdserver1". Los parámetros de la instalación del sistema son los siguientes:

- **Nombre de la maquina:** bdserver1
- **Particiones primarias:**
  - Sistema: 10 GB, punto de montaje "/", formato ext4
  - SWAP: espacio restante, formato "área de intercambio"
- **Configuración automática de la red:** Sin proxy.
- **Quitamos las actualizaciones automáticas.**
- **Instalación del cargador de arranque.**

Con el sistema ya instalado empezaremos con la instalación del servidor de bases de datos de Mysql lanzando las siguientes órdenes:

apt-get update
apt-get install mysql-server

Una vez instalado el servidor de Mysql lo configuramos para que se pueda acceder a las bases de datos remotamente. Para ello modificamos el archivo "my.cnf" cambiando la línea "bind-address" poniendo la dirección IP del servidor, en este caso la dirección IP de la red interna.

- /etc/mysql/my.cnf

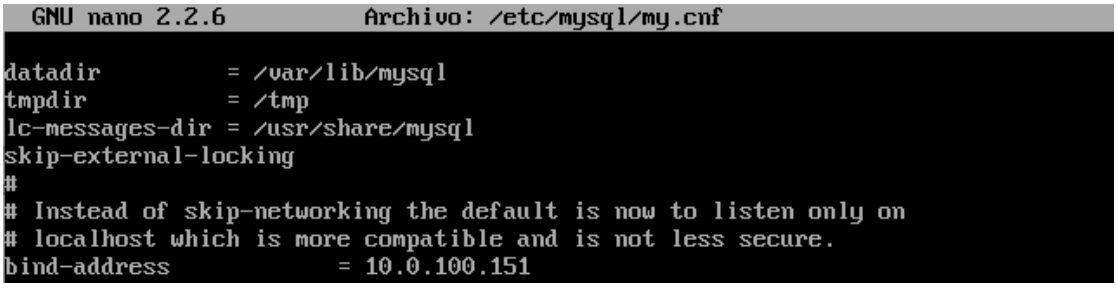

Figura 27. Primera configuración del archivo my.cnf del servidor bdserver1.

Para que los cambios surtan efecto tendremos que reiniciar el servicio lanzando la siguiente orden:

service mysql restart

Ahora vamos a crear la base de datos y los usuarios necesarios para que el sistema de Wordpress se pueda conectar a la misma. Procedemos accediendo a "Mysql":

mysql -u root –p

Creamos la base de datos:

CREATE DATABASE wordpress;

Creamos el usuario local y le damos todos privilegios:

CREATE USER 'wordpressuser'@'localhost' IDENTIFIED BY 'wordpress';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpressuser'@'localhost';

Creamos los usuarios que accederán remotamente (dos por servidor, uno por nombre de servidor y otro por IP) y les daremos todos los permisos (una vez esté todo el sistema instalado sería conveniente que estos usuarios solo tengan permisos de lectura, escritura, modificación y borrado sobre la base de datos anteriormente creada)

CREATE USER 'wordpressuser'@'claster2.cluster' IDENTIFIED BY 'wordpress';
CREATE USER 'wordpressuser'@'10.0.100.2' IDENTIFIED BY 'wordpress';
CREATE USER 'wordpressuser'@'claster3.cluster' IDENTIFIED BY 'wordpress';
CREATE USER 'wordpressuser'@'10.0.100.3' IDENTIFIED BY 'wordpress';
CREATE USER 'wordpressuser'@'claster4.cluster' IDENTIFIED BY 'wordpress';
CREATE USER 'wordpressuser'@'10.0.100.4' IDENTIFIED BY 'wordpress';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpressuser'@'claster2.cluster';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpressuser'@'10.0.100.2';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpressuser'@'claster3.cluster';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpressuser'@'10.0.100.3';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpressuser'@'claster4.cluster';
GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpressuser'@'10.0.100.4';

Para que todos los permisos se estabilicen y sean persistentes, ejecutaremos la siguiente orden:

FLUSH PRIVILEGES;

Una vez creados estos usuarios podemos probar estos usuarios con la siguiente orden en los servidores:

mysql -u wordpressuser -h "IP del servidor" -p

Previamente habrá que instalar el "mysql-client" en estos servidores ejecutando la siguiente orden desde el nodo maestro:

./psh.sh "apt-get install mysql-client"

Ahora procederemos con la duplicación de los servidores de bases de datos en formato mastro/esclavo replicados. Instalamos el sistema del servidor "bdserver2" igual que el servidor "bdserver1" y también instalamos el "mysql-server". Ahora debemos ir al servidor "bdserver1", donde modificamos el archivo "my.cnf" descomentando las líneas "server-id=1" y "log_bin=/var/log/mysql-mysql-bin.log". El archivo se quedará como en la figura siguiente:

- /etc/mysql/my.cnf:

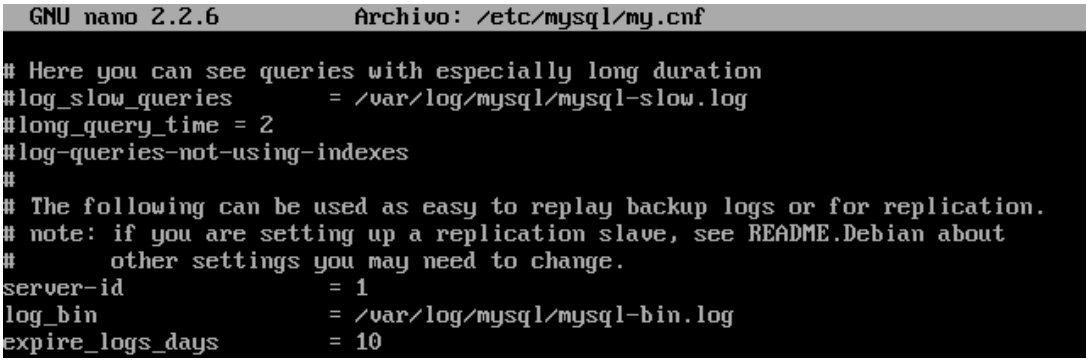

Figura 28. Configuración final del archivo my.cnf del servidor bdserver1.

Reiniciamos el servicio de mysql del "bdserver1". Ahora procedemos creando al usuario que utilizará el "bdserver2" para mantener la información sincronizada. Esto se realizará de la siguiente forma:

mysql -u root –p
CREATE USER 'replica'@'%' IDENTIFIED BY 'replica';
GRANT REPLICATION SLAVE ON *.* TO 'replica'@'%';

Seguidamente tenemos que bloquear la base de datos para poder realizar la copia. Se realizará de la siguiente forma:

FLUSH TABLES WITH READ LOCK;
SET GLOBAL read_only = ON;
EXIT

Posteriormente se hará la copia de seguridad de la base de datos y la copiaremos en el "bdserver2":

mysqldump --lock-all-tables -u root -p --all-databases > masterdump.sql
scp masterdump.sql bdserver2:/root

Una vez realizado esto volveremos a desbloquear la base de datos de la siguiente manera:

mysql -u root -p
SET GLOBAL read_only = OFF;
UNLOCK TABLES;

Como necesitaremos conocer varios parámetros del server "bdserver1" para configurar el "bdserver2" ejecutaremos la siguiente orden en el servidor de mysql:

SHOW MASTER STATUS;

Ahora vamos a configurar el servidor "bdserver2" que hará de esclavo. Procedemos importando la copia de seguridad hecha a la base de datos y copiada en este servidor. Dicha copia se realizará de la siguiente forma:

mysql -u root -p < /tmp/masterdump.sql

Seguidamente procederemos modificando en el archivo "my.cnf" los parámetros "bind-address" y "server-id" del servidor esclavo tal y como se muestra en la siguiente figura:

- /etc/mysql/my.cnf:

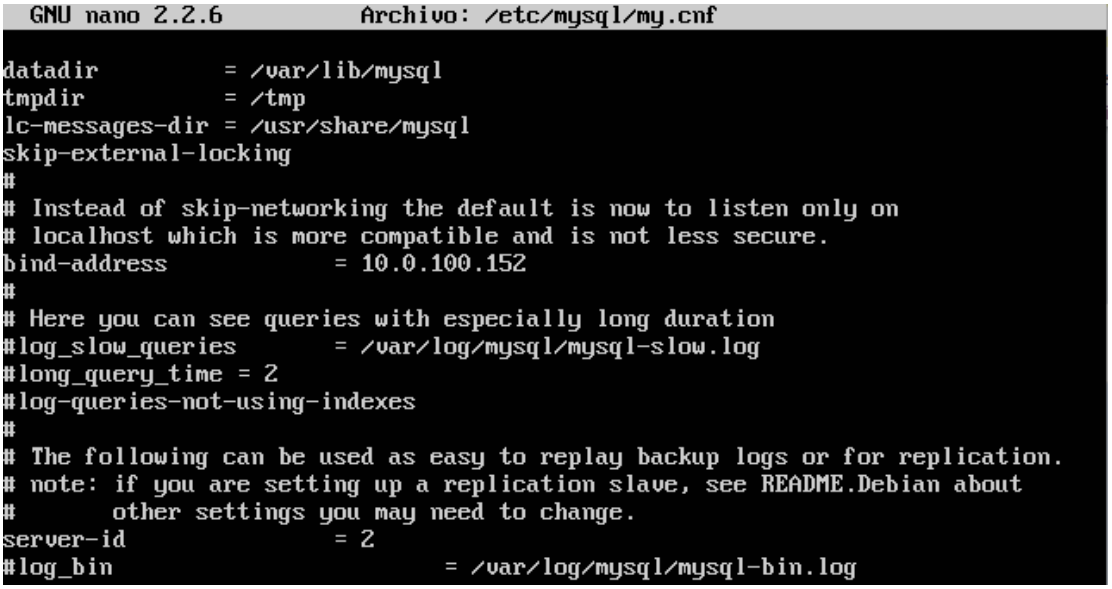

Figura 29. Archivo de configuración my.cnf del servidor bdserver2.

Reiniciamos el servicio de mysql para que surtan efecto los cambios. Vamos a conectar el servidor esclavo con el maestro de la siguiente forma:

mysql -u root –p
CHANGE MASTER TO
MASTER_HOST='10.0.100.151',
MASTER_USER='replica',
MASTER_PASSWORD='replica',
MASTER_LOG_FILE='mysql-bin.000001', #Información sacada del maestro
MASTER_LOG_POS=408; #Información sacada del maestro
START SLAVE;

Con esto ya tenemos los servidores de bases de datos conectado en forma maestro/esclavo. Para evitar que el esclavo tenga información que no esté en el maestro, se configurará el servidor esclavo para que solo reciba peticiones de lectura. Para ello revocaremos los permisos de creación, modificación y borrado en los usuarios del servidor esclavo, esto se realizará de la siguiente manera:

REVOKE INSERT, UPDATE, DELETE ON wordpress.* FROM 'wordpressuser'@'cluster2.cluster';
REVOKE INSERT, UPDATE, DELETE ON wordpress.* FROM 'wordpressuser'@'10.0.100.2';
REVOKE INSERT, UPDATE, DELETE ON wordpress.* FROM 'wordpressuser'@'cluster3.cluster';

```
REVOKE INSERT, UPDATE, DELETE ON wordpress.* FROM 'wordpressuser'@'10.0.100.3';
REVOKE INSERT, UPDATE, DELETE ON wordpress.* FROM 'wordpressuser'@'cluster4.cluster';
REVOKE INSERT, UPDATE, DELETE ON wordpress.* FROM 'wordpressuser'@'10.0.100.4';
FLUSH PRIVILEGES;
```

Con esto ya tendremos terminada la instalación y configuración de los servidores de bases de datos.

3.3.9. Instalación y configuración del sistema Wordpress y del plugin HyperBD en los nodos servidores

Una vez configurado los nodos servidores web y los nodos de bases de datos procederemos instalando la plataforma de Wordpress que se conectará a los servidores de bases de datos. En este caso lo haremos todo en el "servidor1". Descargaremos la aplicación, la descomprimimos y copiamos todos los archivos al directorio "/nfs/www/blog":

```
wget http://wordpress.org/latest.tar.gz
tar xzvf latest.tar.gz
mkdir -p /nfs/www/blog
cp -r /wordpress/* /nfs/www/blog
```

Copiamos la configuración ejemplo que viene con Wordpress en el archivo de configuración "wp-config.php".

```
cp /nfs/www/blog/wp-config-sample.php /nfs/www/blog/wp-config.php
```

Modificamos dicho archivo de configuración con los parámetros de conexión con el servidor de base de datos.

- /nfs/www/blog/wp-config.php

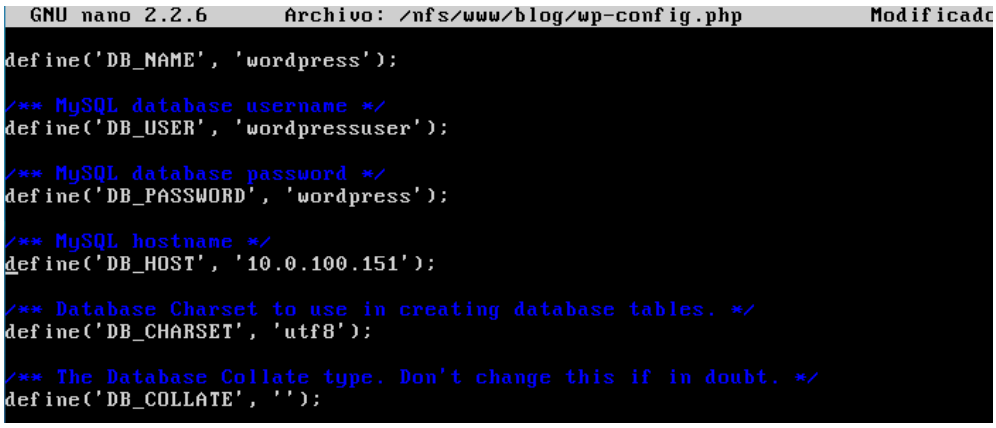

Figura 30. Archivo de configuración del Wordpress.

Le damos permisos al usuario del servidor web con las siguientes órdenes:

```
chown -R www-data:www-data /nfs/www/blog/*
chmod -R g+rw /nfs/www/blog/
```

Como tenemos el directorio compartido entre los nodos servidores web no tendremos que realizar ningún cambio adicional. En el caso que deseemos tener el directorio independiente en cada nodo servidor web tendremos que tener sincronizados dichos directorios. Dicha sincronización la podemos realizar tal y como se explica a continuación.

En todos los nodos servidores crearemos la carpeta "blog" y le volveremos a dar los permisos. Para mantener sincronizados los directorios "/nfs/www/blog" se creará un script "sincro.sh" (ANEXO 7) que comprobará si ha habido alguna modificación y con la herramienta rsync copiara los archivos en el resto de servidores. Dicho script se ejecutara automáticamente cada 10 segundos con la herramienta "crontab". El script lo crearemos en el nodo maestro y lo distribuiremos al resto de servidores con el script "./pcp.sh", donde le daremos permiso de ejecución. El comando que sincronizará los directorios será el siguiente:

rsync -rvuc --delete /nfs/www/blog/ cluster$i:/nfs/www/blog/

Explicación de las funciones de rsync utilizadas:

- **-r:** significa que hará el copiado recursivo de carpetas, es decir, creará toda la estructura de las carpetas que hay dentro de carpeta_origen en carpeta_destino, y copiará todos los archivos que es encuentren dentro de estas.
- **-v:** significa que la información imprimida durante la ejecución del programa será mucho más detallada, podemos utilizar esto para ver el progreso del comando.
- **-u:** significa que esto actualizará los contenidos de la carpeta destino en base a la carpeta de origen.
- **-c:** significa que realizará un checksum en los archivos a ser transferidos. Esto permitirá que se ignore cualquier archivo para el que el checksum coincida.
- **--delete:** esto usado en conjunto con el parámetro previo -u que actualiza los archivos modificados nos permite mantener dos carpetas en sincronía ahorrando ancho de banda.

La herramienta "crontab" se lanzara de la siguiente forma:

crontab -e

Y la configuramos de la siguiente forma:

```
  GNU nano 2.2.6      Archivo: /tmp/crontab.kJVmpU/crontab

#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
*/1 * * * * ./sincro
*/1 * * * * sleep 10 && ./sincro
*/1 * * * * sleep 20 && ./sincro
*/1 * * * * sleep 30 && ./sincro
*/1 * * * * sleep 40 && ./sincro
*/1 * * * * sleep 50 && ./sincro
```

Figura 31. Archivo de configuración de Crontab.

Ahora tenemos que instalar en los nodos servidores web el *plugin* "HyperDB" de Wordpress para que puedan conectarse a ambos servidores de bases de datos. Procedemos instalando el *plugin* de la siguiente manera:

wget http://downloads.wordpress.org/plugin/hyperdb.zip
apt-get install zip
unzip hyperdb.zip
cp /hyperdb/db-config.php /nfs/www/blog
cp /hyperdb/db.php /nfs/www/blog/wp-content/

Una vez instalado modificaremos el archivo "db-config.php" para conectarse con el servidor "bdserver2" modificando la segunda instancia "wpdb-> add_database (array…" como aparece en la siguiente figura donde en "host" pondremos "DB_SLAVE":

- /nfs/www/blog/ db-config.php:

```
  GNU nano 2.2.6      Archivo: /nfs/www/blog/db-config.php

        'name'     => DB_NAME,
));

/**
 * This adds the same server again, only this time it is configured as a slave.
 * The last three parameters are set to the defaults but are shown for clarity.
 */
$wpdb->add_database(array(
        'host'     => DB_SLAVE,     // If port is other than 3306, use host:por$
        'user'     => DB_USER,
        'password' => DB_PASSWORD,
        'name'     => DB_NAME,
        'write'    => 0,
        'read'     => 1,
        'dataset'  => 'global',
        'timeout'  => 0.2,
));
```

Figura 32. Archivo de configuración del HyperDB.

Dicho parámetro "DB_SLAVE" tendremos que definirlo con la IP del "bdserver2" añadiendo la siguiente línea en el archivo "wp-config.php":

define('DB_SLAVE', '10.0.100.152');

Para terminar con la instalación y configuración de "HyperDB" tendremos que volver a actualizar los permisos de la siguiente manera:

chmod a-w /nfs/www/blog/wp-content/db.php
chown -R www-data:www-data /nfs/www/blog/

Finalmente, como php5 tiene deshabilitada la conexión remota con mysql debemos modificar el archivo "php.ini" descomentando las siguientes líneas en el apartado "Dynamic Extension":

- /etc/php5/apache2/php.ini:

```
  GNU nano 2.2.6       Archivo: /etc/php5/apache2/php.ini

; syntax:
;
;   extension=modulename.extension
;
; For example, on Windows:
;
extension=msql.dll
;
; ... or under UNIX:
;
extension=msql.so
;
; ... or with a path:
;
extension=/path/to/extension/msql.so
        'dataset'   => 'global',
        'timeout'   => 0.2,
));
```

Figura 33. Archivo de configuración php.ini del Apache.

Ahora vamos a terminar de configurar el sistema Wordpress accediendo a la URL http://localhost:5080/blog/wp-admin/ donde nos pedirá el nombre del sitio, usuario y contraseña del administrador y una cuenta de correo. Escribiremos esta información y seleccionaremos la opción "Instalar Wordpress".

Figura 34. Formulario de configuración de Wordpress.

Si todo ha salido bien nos aparecerá la ventana siguiente:

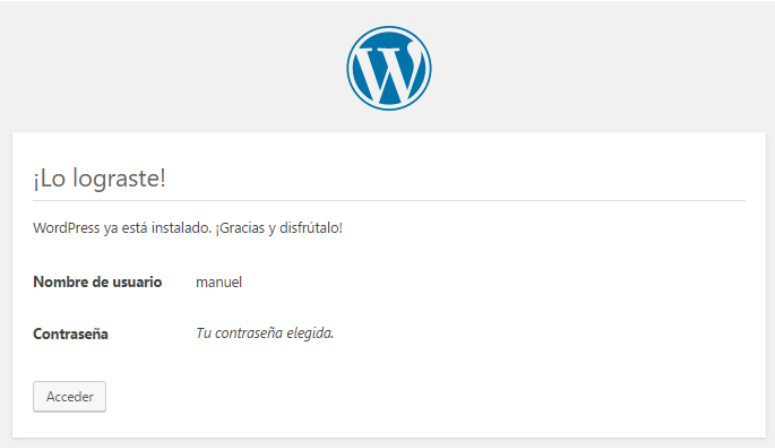

Figura 35. Ventana de confirmación de la instalación del Wordpress.

3.3.10. Instalación y configuración del servicio Keepalived en los nodos Master y Standby

Instalado y configurado el sistema Wordpress ya tendremos los servidores web configurados. Para añadir alta disponibilidad, procedemos clonando el servidor maestro para crear el servidor "standby" mediante VirtualBox, reiniciando la MAC. Editamos el archivo "interfaces" con las direcciones IP estáticas correspondientes a cada interfaz, tal y como se muestra en la siguiente figura. También modificaremos el archivo "hostname" con el nombre "standby".

- /etc/network/interfaces:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.1.102
        netmask 255.255.255.0
        gateway 192.168.1.1
        dns-nameservers 192.168.1.1
        mtu 9000

auto eth1
iface eth1 inet static
        address 10.0.100.50
        netmask 255.255.255.0
        mtu 9000
```

Figura 36. Archivo interfaces del servidor standby.

Una vez realizados estos cambios en el nodo standby procederemos instalando y configurando el servicio "keepalived" en ambos servidores, esto quiere decir que lo que se haga en uno se realizará en el otro. Vamos a instalar el "keepalived" de la siguiente manera:

apt-get install keepalived

Instalado el paquete en ambos servidores, lo configuraremos en el maestro creando el archivo de configuración "keepalived.conf".

- /etc/keepalived/keepalived.conf:
  - **Master:**

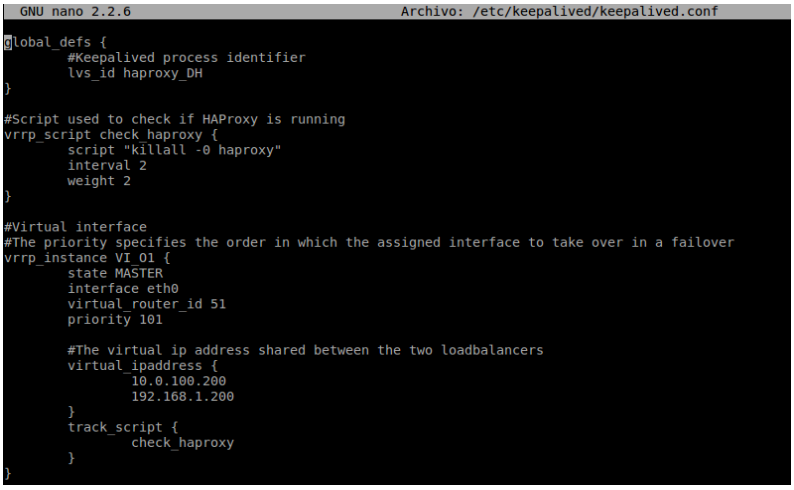

Figura 37. Archivo de configuración keepalived del servidor maestro.

  - **Standby:**

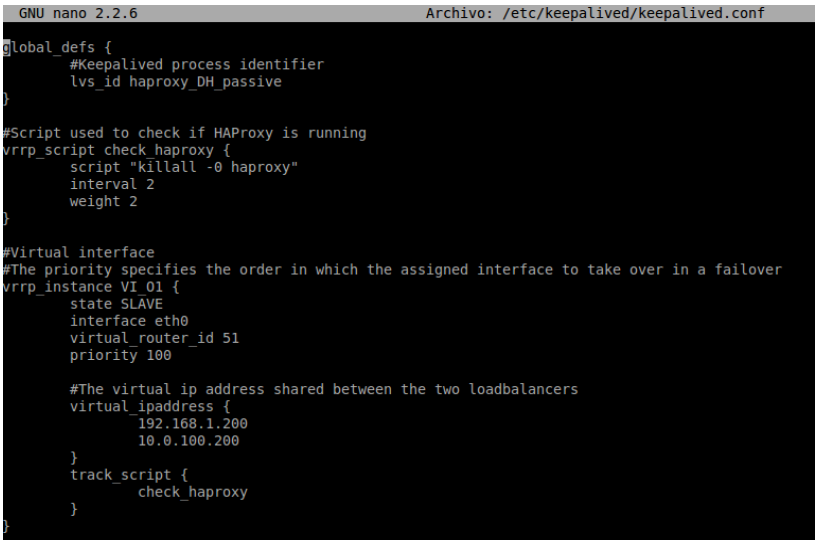

Figura 38. Archivo de configuración keepalived del servidor standby.

Como observamos en el archivo de configuración se creará un método "global_defs" para identificar a cada servidor. Una sección "vrrp_script check_haproy" que se utiliza para comprobar si el proceso local continua en ejecución. Otra sección "vrrp_intance VI_01" contiene la prioridad de cada nodo. En nuestro caso será prioritario el nodo maestro, también se configura un router virtual con las dos direcciones IP VIP (la externa y la interna). Para que los cambios surtan efecto, se reiniciará el servicio de keepalived en ambos servidores.

service keepalived restart

Con esto ya tendremos nuestro clúster montado y en funcionamiento, en caso que el nodo maestro caiga el nodo standby cogerá sus funciones.

4. Verificación del funcionamiento

En esta sección del TFM vamos a verificar el correcto funcionamiento del clúster, para comprobar que realiza adecuadamente las funciones configuradas.

En todas las pruebas realizadas, dado que los nodos están basados en máquinas virtuales, se simulará la caída del nodo desconectando la unidad de red. Se ha utilizado la página web sencilla "index.php" para comprobar la alta disponibilidad y el equilibrado de carga y la web de Wordpress para comprobar la conexión con los servidores de bases de datos.

Vamos a proceder con las pruebas. En la primera prueba comprobaremos si se garantiza la alta disponibilidad, desconectando la interfaz externa del nodo maestro (interfaz "eth0"). La prueba consiste en lanzar un par de peticiones, desconectar la interfaz de red y lanzar una nueva petición.

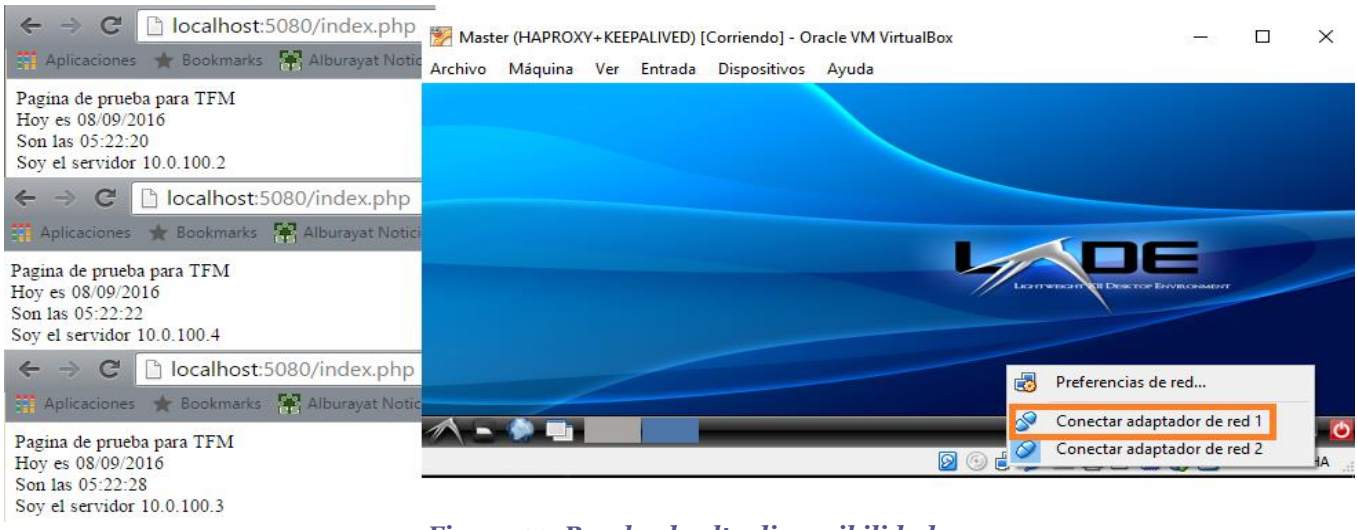

Figura 39. Prueba de alta disponibilidad.

Como vemos en la figura anterior el servidor responde satisfactoriamente las peticiones lanzadas. Aquí el nodo standby ha adquirido la labor de nodo maestro. Sin embargo, debe hacerse constar que el cambio no es instantáneo, la carga de la web falla un par de veces.

En la segunda prueba simularemos la caída de uno de los nodos servidores. Por ejemplo, el nodo server1 (cluster2) falla.

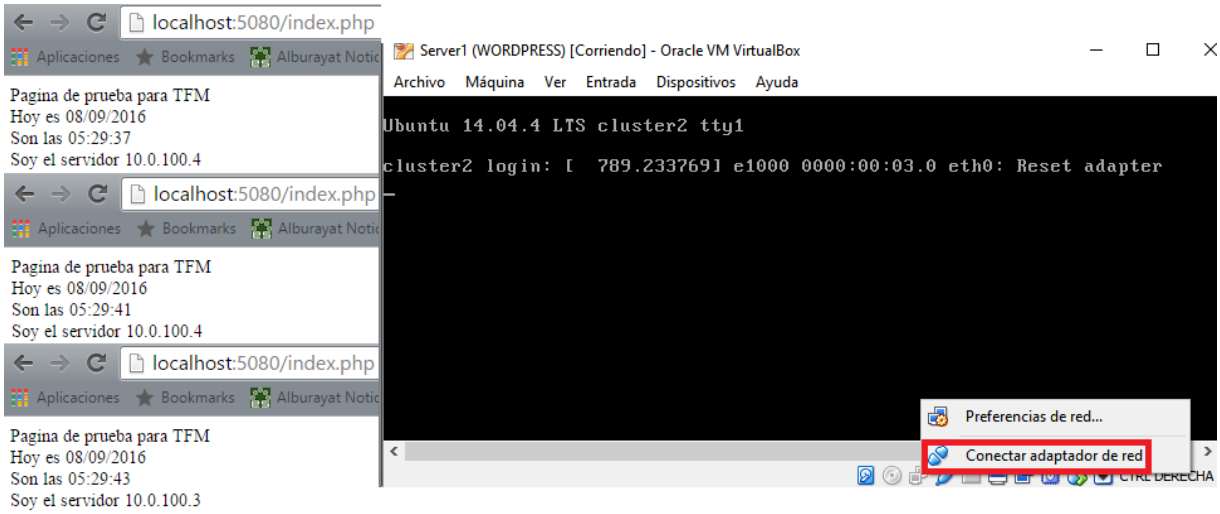

Figura 40. Prueba ante falla de un nodo servidor web.

Vemos en la figura anterior que la web se sigue sirviendo por parte de los nodos supervivientes: server2 y server3. Si cayera otro de los nodos servidores, las peticiones serían respondidas por el nodo que quedase.

En la última prueba comprobamos la conexión con los servidores de base de datos. En el equilibrado de carga vemos que cada vez que se conecta responde un nodo servidor distinto debido al equilibrado "round robin".

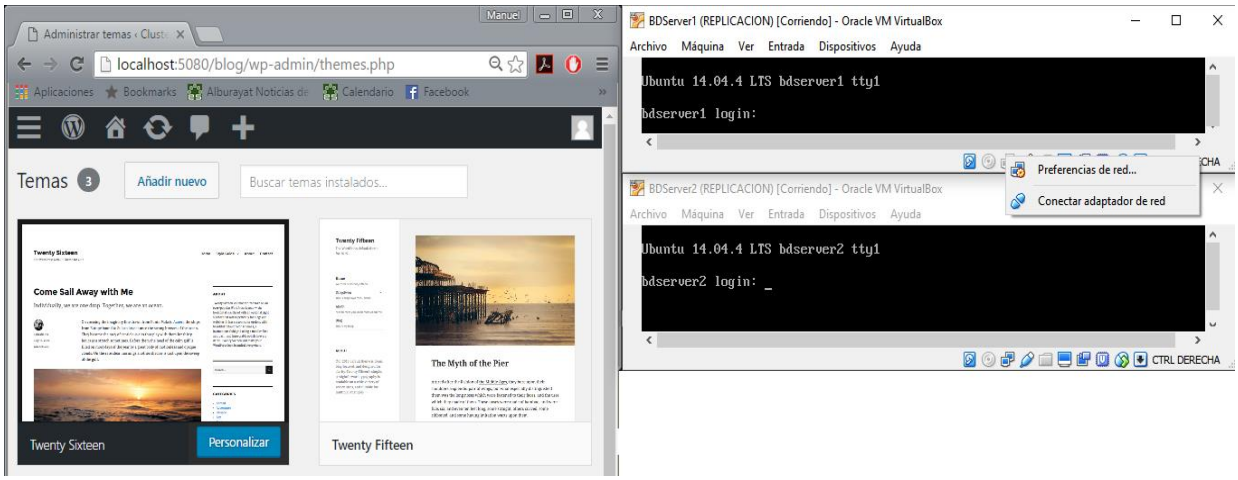

Figura 41. Prueba ante fallo de un nodo de base de datos.

Como vemos en la figura anterior podemos acceder a la base de datos, obteniendo la información que tenemos configurada. En la figura se ha desconectado el nodo maestro de la base de datos, con lo cual no se puede actualizar la configuración en el caso que hagamos algún cambio. Esto es debido a que el servidor esclavo solo puede recibir conexiones de lectura, cualquier otro tipo de conexión la rechaza. Por ese motivo, para acceder a la web de configuración de Wordpress necesitamos que esté activo el servidor maestro de base de datos, sino no se podrá acceder.

Tras estas sencillas pruebas, podemos concluir que el clúster realiza las acciones para las cuales ha sido configurado.

5. Evaluación

Para la evaluación del servidor web configurado utilizaremos los siguientes *benchmarks*:

- **ab (Apache Benchmark).**
- **Siege.**

En ambos *benchmarks*, el programa envía un número de peticiones al servidor web, según los parámetros indicados, obteniendo las correspondientes métricas sobre su comportamiento. También se permite indicar el nivel de concurrencia en las peticiones, simulando así la presencia de varios usuarios simultáneos. El paquete Siege nos añade la posibilidad de insertar un retardo entre los bloques de peticiones, así como solicitar, no una sino varias URL diferentes a lo largo del test, simulando la navegación de los usuarios por la web.

Ambos *benchmarks* pueden lanzarse con diversas opciones, que modifican la carga que se lanzará al sistema a evaluar. Las tablas siguientes muestran las opciones de configuración más habituales:

- **ab [-c concurrency] [-H custom-header] [-k] [-n requests] [-t timelimit] [-v verbosity] [http[s]://]hostname[:port]/path**
  - [-c concurrency] Número de usuarios concurrentes
  - [-n requests] Número total de peticiones servidas para completar la prueba
  - [-t timelimit] Tiempo total para completar la prueba
  - [-v verbosity] Permite aumentar el nivel de información volcada durante su realización. Por ejemplo, -v2 muestra las cabeceras http
  - [-H custom-header] Permite añadir campos a la cabecera. Por ejemplo, para solicitar compresión gzip se utilizaría –H “Accept-Encoding:gzip”
  - [-k] Solicita conexiones persistentes (http keepalive)
  - [http[s]://]hostname[:port]/path URL a evaluar

- **siege [-b] [-c NUM] [-d NUM] [-f FILE] [-g] [-i] [-r NUM] [-t NUM] [-v] [protocol://] host.domain.xxx [:port] [/path/file]**
  - [-c NUM] Número de usuarios concurrentes
  - [-r NUM] Número total de repeticiones de cada ráfaga de usuarios concurrentes
  - [-t NUM] Tiempo total para completar la prueba
  - [-d NUM] Inserta un retardo comprendido entre 0 y NUM s antes de lanzar la siguiente ráfaga de peticiones
  - [-b] Lanza la siguiente ráfaga de peticiones sin demora
  - [-f FILE] Lanza las peticiones a la lista de URL contenidas en el archivo FILE
  - [-i] Con la opción [-f FILE], selecciona aleatoriamente la URL
  - [-g] Muestra las cabeceras http (e incluso del contenido de la página, según se indique en el archivo de configuración /etc/siege/siegerc)
  - [-v] Muestra información acerca de la petición lanzada y la respuesta del servidor (retorno http y petición GET)
  - [http[s]://]hostname[:port]/path URL a evaluar

Hay que hacer notar que las pruebas las haremos sobre un entorno virtualizado y en un ordenador que tiene pocos recursos, por lo que esto puede interferir en los resultados que obtengamos. Pero en todo caso, pretendemos experimentar con herramientas que se utilizarían para evaluar un clúster formado por máquinas físicas.

Procederemos con la instalación en el nodo maestro de los dos *benchmarks* y la herramienta gráfica. Sería mejor instalarlos en una maquina externa para no interferir en la prueba, pero como no disponemos de suficientes recursos lo haremos sobre el nodo maestro. Para ello lanzaremos los siguientes comandos:

apt-get install apache2 siege gnuplot
apt-get install apache2-utils

5.1. ab (Apache Benchmark)

Empezamos las pruebas de carga con "ab". El resultado obtenido de cada prueba lo guardaremos en un archivo .csv (ejemplo: resultado.csv) para luego procesarlo con GNUPlot. Vamos a comprobar cómo se comporta nuestro clúster ante una serie de peticiones realizadas. Inicialmente lanzaremos una prueba de 100.000 peticiones en total, con un número de peticiones concurrentes igual al número total de procesadores que tenemos en los servidores web (en nuestro caso 1 procesador por servidor, 3 procesadores en total). Lanzaremos la siguiente orden:

- **ab –g /root/resultados1.csv –n 100000 –c 3 http://192.168.1.200/index.php**

```
Document Path:          /
Document Length:        131 bytes

Concurrency Level:      3
Time taken for tests:   1112.550 seconds
Complete requests:      100000
Failed requests:        0
Total transferred:      34300000 bytes
HTML transferred:       13100000 bytes
Requests per second:    89.88 [#/sec] (mean)
Time per request:       33.376 [ms] (mean)
Time per request:       11.125 [ms] (mean, across all concurrent requests)
Transfer rate:          30.11 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   13 421.2      0   63132
Processing:     1   21 190.8      4   18014
Waiting:        0   21 190.8      4   18014
Total:          1   33 484.0      4   63137

Percentage of the requests served within a certain time (ms)
  50%      4
  66%      5
  75%      7
  80%      8
  90%     11
  95%     14
  98%     26
  99%   1005
 100%  63137 (longest request)
```

Figura 42. Resultado de prueba con ab. Página index.php, 100.000 peticiones, c=3.

Como observamos en los resultados, vemos que el tiempo de respuesta es de unos 33 ms y la tasa de peticiones atendidas es aproximadamente de 90 peticiones/s. Se trata de un valor relativamente bajo, y que se debe a la escasez de recursos del sistema, basado en máquinas virtuales.

En una segunda prueba veremos que con un número superior de usuarios concurrentes, los nodos servidores web se saturan. Para esta prueba enviaremos 4.000 peticiones en grupos de 100 conexiones simultáneas a nuestro sitio web. Utilizaremos la siguiente orden:

ab –g /root/resultados.csv –n 4000 –c 100 http://192.168.1.200/index.php

```
Server Software:        Apache/2.4.7
Server Hostname:        192.168.1.200
Server Port:            80

Document Path:          /
Document Length:        132 bytes

Concurrency Level:      100
Time taken for tests:   5.516 seconds
Complete requests:      4000
Failed requests:        1334
   (Connect: 0, Receive: 0, Length: 1334, Exceptions: 0)
Total transferred:      1339982 bytes
HTML transferred:       491982 bytes
Requests per second:    725.20 [#/sec] (mean)
Time per request:       137.894 [ms] (mean)
Time per request:       1.379 [ms] (mean, across all concurrent requests)
Transfer rate:          237.24 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   2.9      0      19
Processing:    10  136  24.2    136     297
Waiting:        9  135  23.8    136     297
Total:         22  137  23.7    138     315

Percentage of the requests served within a certain time (ms)
  50%    138
  66%    144
  75%    149
  80%    151
  90%    163
  95%    173
  98%    192
  99%    206
 100%    315 (longest request)
```

Figura 43. Resultado de prueba con ab. Página index.php, 4.000 peticiones, c=100.

Como observamos en los resultados hay 1.334 peticiones perdidas debido a que el clúster no puede responder tantas peticiones concurrentes tan seguidas. También vemos que la petición que más ha tardado, necesitó unos 315 ms. También observamos que el tiempo medio de respuesta, en comparación con las pruebas anteriores, es muy elevado, de 138 ms.

La gráfica siguiente muestra la distribución del tiempo de respuesta de las peticiones. Es normal que haya unas pocas peticiones con un tiempo de respuesta inferior o superior a la media.

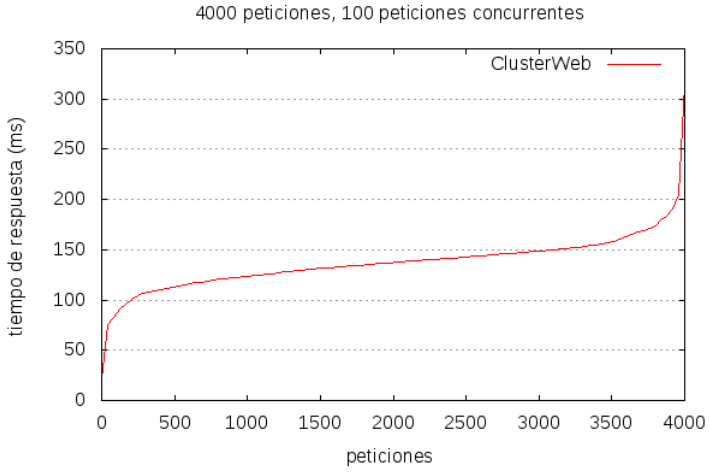

Figura 44. Resultado de prueba con ab. Página index.php, 4.000 peticiones, c=100.

En la siguiente prueba se utilizará una petición web que genere más carga, en este caso se ejecutara el archivo "pi.php[2]" (ANEXO 8) que calcula el valor de la constante PI. El código calcula la constante PI basado en la integración numérica y admite como parámetro el número de intervalos a considerar. Se han solicitado 1.000.000 de intervalos. Para ello lanzaremos las siguientes órdenes:

- **ab –g /root/resultadospi.csv –n 1000 –c 1** http://192.168.1.200/pi.php?n=1000000

```
Document Path:          /pi.php?n=1000000
Document Length:        34 bytes

Concurrency Level:      1
Time taken for tests:   260.571 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      222000 bytes
HTML transferred:       34000 bytes
Requests per second:    3.84 [#/sec] (mean)
Time per request:       260.571 [ms] (mean)
Time per request:       260.571 [ms] (mean, across all concurrent requests)
Transfer rate:          0.83 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       1
Processing:   243  260  20.8    256     489
Waiting:      243  260  20.8    256     489
Total:        243  260  20.8    256     489

Percentage of the requests served within a certain time (ms)
  50%    256
  66%    261
  75%    264
  80%    267
  90%    273
  95%    282
  98%    306
  99%    327
 100%    489 (longest request)
```

Figura 44. Resultado de prueba con ab. Página pi.php, 1.000 peticiones, c=1.

- **ab –g /root/resultadospi.csv –n 1000 –c 2** http://192.168.1.200/pi.php?n=1000000

```
Document Path:          /pi.php?n=1000000
Document Length:        34 bytes

Concurrency Level:      2
Time taken for tests:   159.527 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      222000 bytes
HTML transferred:       34000 bytes
Requests per second:    6.27 [#/sec] (mean)
Time per request:       319.054 [ms] (mean)
Time per request:       159.527 [ms] (mean, across all concurrent requests)
Transfer rate:          1.36 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   257  319  26.6    315     533
Waiting:      257  319  26.6    315     533
Total:        257  319  26.6    315     533

Percentage of the requests served within a certain time (ms)
  50%    315
  66%    322
  75%    327
  80%    331
  90%    341
  95%    352
  98%    383
  99%    467
 100%    533 (longest request)
```

Figura 45. Resultado de prueba con ab. Página pi.php, 1.000 peticiones, c=2.

[2] Código obtenido de la asignatura del Máster de Ingeniería de Computadores y Redes: "Configuración, Administración y Utilización de Clusters de PCs". Dpto. de Informática de Sistemas y Computadores (DISCA, Universidad Politécnica de Valencia). Valencia.

- **ab –g /root/resultadospi.csv –n 1000 –c 3 http://192.168.1.200/pi.php?n=1000000**

```
Document Path:          /pi.php?n=1000000
Document Length:        34 bytes

Concurrency Level:      3
Time taken for tests:   137.652 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      222000 bytes
HTML transferred:       34000 bytes
Requests per second:    7.26 [#/sec] (mean)
Time per request:       412.957 [ms] (mean)
Time per request:       137.652 [ms] (mean, across all concurrent requests)
Transfer rate:          1.57 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:   318  412  43.3    401     839
Waiting:      318  412  43.3    401     839
Total:        318  413  43.3    401     839

Percentage of the requests served within a certain time (ms)
  50%    401
  66%    411
  75%    420
  80%    426
  90%    467
  95%    518
  98%    536
  99%    548
 100%    839 (longest request)
```

Figura 46. Resultado de prueba con ab. Página pi.php, 1.000 peticiones, c=3.

Como observamos en los resultados de las 3 pruebas realizadas, vemos que el tiempo de respuesta está en torno a los 300 ms, aunque aumenta ligeramente para c=3. El número de peticiones por segundo aumenta con el número de peticiones concurrentes, aunque no linealmente, como cabía esperar.

En la última prueba que realizaremos con este *benchmark* veremos cómo nuestro sistema se satura. Para ello ejecutaremos la misma orden que en la prueba anterior pero con concurrencia 6.

ab –g /root/resultadospi.csv –n 1000 –c 6 http://192.168.1.200/pi.php?n=1000000

```
Document Path:          /pi.php?n=1000000
Document Length:        34 bytes

Concurrency Level:      6
Time taken for tests:   141.194 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      222000 bytes
HTML transferred:       34000 bytes
Requests per second:    7.08 [#/sec] (mean)
Time per request:       847.161 [ms] (mean)
Time per request:       141.194 [ms] (mean, across all concurrent requests)
Transfer rate:          1.54 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:   361  846 467.0    675    1992
Waiting:      361  846 467.0    674    1992
Total:        361  846 467.0    675    1992

Percentage of the requests served within a certain time (ms)
  50%    675
  66%    938
  75%   1271
  80%   1364
  90%   1597
  95%   1672
  98%   1755
  99%   1876
 100%   1992 (longest request)
```

Figura 47. Resultado de prueba con ab. Página pi.php, 1.000 peticiones, c=6.

Observamos en los resultados que el tiempo de respuesta se duplica claramente (alrededor de 850 ms) y el número de peticiones servidas por segundo no aumenta más. Esto se debe a que los servidores están saturados.

5.2. Siege

Ahora procedemos a la evaluación del sistema con la herramienta "Siege". Para ello haremos una primera prueba que será hacer el test durante 30 minutos donde cada 10 segundos se lanzarán grupos de 100 conexiones simultáneas a nuestro sitio web. Este test es diferente a los realizados con el Apache Benchmark. Comprobamos que si hay un retardo entre las peticiones concurrentes, nuestro servidor web puede responder sin rechazar ninguna petición. Lanzaremos la siguiente orden:

siege –c100 –d10 –t30M http://192.168.1.200/

```
** SIEGE 3.0.5
** Preparing 100 concurrent users for battle.
The server is now under siege...
Lifting the server siege...      done.

Transactions:                  35756 hits
Availability:                 100.00 %
Elapsed time:                1799.25 secs
Data transferred:               4.18 MB
Response time:                  0.00 secs
Transaction rate:              19.87 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    0.07
Successful transactions:       35756
Failed transactions:               0
Longest transaction:            0.50
Shortest transaction:           0.00
```

Figura 48 Resultados de la 1º prueba del benchmark Siege.

Como observamos en los resultados del Siege, no se ha perdido ninguna respuesta en comparación con los resultados obtenidos con el Apache Benchmark. El servidor configurado procesa unas 20 peticiones por segundo.

A continuación, vamos a realizar una prueba similar a la realizada con el Apache Benchmark (cuyo resultado se muestra en la Figura 42) y la página index.php. Ejecutamos la siguiente orden:

siege –b –r100000 –c3 http://192.168.1.200/index.php

```
** Preparing 3 concurrent users for battle.
The server is now under siege..      done.

Transactions:                  300000 hits
Availability:                 100.00 %
Elapsed time:                3502.21 secs
Data transferred:              37.20 MB
Response time:                  0.03 secs
Transaction rate:              85.66 trans/sec
Throughput:                     0.01 MB/sec
Concurrency:                    2.99
Successful transactions:       300000
Failed transactions:                0
Longest transaction:           31.08
Shortest transaction:           0.00
```

Figura 49 Resultados de prueba del benchmark Siege. Página index.php, 100.000 peticiones, c=3.

Como observamos en los resultados, se han realizado un total de 300.000 peticiones en total, de tres en tres. El sistema ha podido procesar 85,66 transacciones por segundo y el tiempo respuesta ha sido de 0,03 segundos, resultados similares a los obtenidos en la figura 42.

Seguidamente, mostraremos los resultados obtenidos utilizando el fichero "pi.php". Esta prueba tardará aproximadamente unas 20 horas debido a la intensidad del cómputo solicitado a los nodos servidores y al elevado número de peticiones solicitadas. Lanzamos la siguiente orden:

siege –b –r100000 –c3 http://192.168.1.200/pi.php?n=1000000

```
** Preparing 3 concurrent users for battle.
The server is now under siege..      done.

Transactions:                 300000 hits
Availability:                 100.00 %
Elapsed time:               48134.40 secs
Data transferred:               9.73 MB
Response time:                  0.48 secs
Transaction rate:               6.23 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    3.00
Successful transactions:      300000
Failed transactions:               0
Longest transaction:            2.03
Shortest transaction:           0.28
```

Figura 50Resultados de prueba del benchmark Siege. Página pi.php, 300.000 peticiones, c=3.

Como observamos en los resultados, se han realizado un total de 300.000 peticiones en total, de tres en tres. El sistema ha podido procesar 6,23 transacciones por segundo, y el tiempo de respuesta ha sido de 0,48 segundos. Estos resultados son similares a los obtenidos con el Apache Benchmark mostrados en la figura 46.

Una última prueba que vamos a lanzar será simular como un usuario navega entre diferentes URLs. Para ello lanzaremos 300.000 peticiones entre varias webs, almacenadas en el fichero urls.txt.

- /root/urls.txt:

```
  GNU nano 2.2.6          Archivo: /root/uris.txt

http://192.168.1.200/index.php
http://192.168.1.200/pi.php?n=100000
https://192.168.1.200/index.php
https://192.168.1.200/pi.php?n=100000
```

Figura 51 Archivo urls.txt utilizado en la última prueba del benchmark Siege.

Lanzaremos la siguiente orden:

siege –b –r100000 –c3 –i –f /root/urls.txt

```
** Preparing 3 concurrent users for battle.
The server is now under siege..      done.

Transactions:                 300000 hits
Availability:                 100.00 %
Elapsed time:                4030.21 secs
Data transferred:              20.03 MB
Response time:                  0.04 secs
Transaction rate:              74.44 trans/sec
Throughput:                     0.00 MB/sec
Concurrency:                    2.99
Successful transactions:      300000
Failed transactions:               0
Longest transaction:            1.09
Shortest transaction:           0.00
```

Figura 52 Resultados de prueba del benchmark Siege. Página urls.txt, 300.000 peticiones, c=3.

Como observamos en los resultados, se han realizado un total de 300.000 peticiones en total. El sistema ha podido procesar 74,44 transacciones por segundo y el tiempo de respuesta ha sido de 0,04 segundos. Igualmente no se ha perdido ninguna respuesta.

6. Conclusiones

Como se ha observado en el TFM se pueden crear servidores web mediante clústeres como una buena alternativa coste/prestaciones. En nuestro caso se ha configurado un prototipo basado en máquinas virtuales para este fin.

Con la ayuda de servicios como HAProxy y Keepalived podemos ofrecer la alta disponibilidad y equilibrado de carga que se requieren hoy en día. HAProxy ofrece también protección contra ataques DDoS sin tener que instalar ningún nuevo servicio.

En los servidores web internos se ha elegido la plataforma Wordpress por su popularidad. Wordpress precisa de Mysql. Por ese motivo, se ha instalado un servidor de base de datos. Para reducir la sobrecarga en el servidor de base de datos se ha instalado un segundo servidor conectado en formato maestro/esclavo. En Wordpress tenemos el problema de que solo se conecta a un solo servidor Mysql y esto se ha solucionado instalándole el plugin HyperDB.

La utilización de *benchmarks* como Apache Benchmark y Siege nos permite evaluar las prestaciones del sistema configurado. Para perfeccionarlo o corregir posibles problemas.

7. ANEXOS

- ANEXO 1: dnsmasq-dhcp-pxe

Script para lanzar el dnsmasq con las funciones DNS+DHCP+PXE.

```
#!/bin/bash

echo "Iniciando dnsmasq en modo DHCP + PXE"
cp /etc/dnsmasq-dhcp-pxe.conf /etc/dnsmasq.conf
/etc/init.d/dnsmasq restart
```

- ANEXO 2: dnsmasq-dhcp

Script para lanzar el dnsmasq con las funciones DNS+DHCP.

```
#!/bin/bash

echo "Iniciando dnsmasq en modo DHCP"
cp /etc/dnsmasq-dhcp.conf /etc/dnsmasq.conf
/etc/init.d/dnsmasq restart
```

- ANEXO 3: psh.sh

Script para lanzar órdenes a todos los nodos secuencialmente.

```
#!/bin/bash

for i in 2 3 4;
do
   ssh cluster$i $1
   echo $i
done
```

- ANEXO 4: ppsh.sh

Script para lanzar órdenes a todos los nodos paralelamente.

```
#!/bin/bash

for i in 2 3 4;
do
   ssh cluster$i $1 &
   echo $i
done
```

- ANEXO 5: pcp.sh

Script para copiar un mismo archivo a todos los nodos en una ubicación determinada.

```
#!/bin/bash

for i in 2 3 4;
do
   scp $1 cluster$i:$2
   echo $i
done
```

- ANEXO 6: index.php

Página de ejemplo para comprobar el funcionamiento del HAProxy.

```
<html>
<body>
Página de prueba para TFM
<?php
   echo "<br>";
   echo "Hoy es " . date ("d/m/Y") . "<br>";
   echo "Son las " . date ("h:i:s") . "<br>";
   echo "Soy el servidor " . $_SERVER['SERVER_ADDR'] . "<br>";
?>
</body>
</html>
```

- ANEXO 7: sincro.sh

Script para sincronizar los directorios web en los servidores web.

```
#!/bin/bash

DIR_TO_CHECK='/nfs/www/blog'
OLD_STAT_FILE='/root/old_stat.txt'

if [ -e $OLD_STAT_FILE ]
then
   OLD_STAT=$(cat $OLD_STAT_FILE)
else
   OLD_STAT="nothing"
fi

NEW_STAT=$(stat -t $DIR_TO_CHECK)

if [ "$OLD_STAT" != "$NEW_STAT" ]
then
   for i in 2 4;
   do
          rsync -rvuc --delete /nfs/www/blog/
cluster$i:/nfs/www/blog/
   done
   echo $NEW_STAT > $OLD_STAT_FILE
fi
```

- ANEXO 8: pi.php

Página de ejemplo que calcula el número pi mediante integración numérica para la evaluación del clúster.

```
<html>
<body>
<?php
$start=microtime(true);

$area=0.0;

$n=$_GET["n"];
// $n=100000000;

for ($i=0; $i<$n; $i++)
   {
   $x=($i+0.5)/$n;
   $area=$area+4.0/(1.0+$x*$x);
   }
$result=$area/$n;

$end=microtime(true);
$exectime=$end-$start;

echo "<br>Calculo de PI<br><br>";
printf ("La cte. PI con n= %d es igual a %f<br>", $n, $result);
printf ("Tiempo de ejecucion= %.5f segundos<br>",$exectime);
printf ("<br>El servidor es %s<br>", $_SERVER['SERVER_ADDR']);
?>
</body>
</html>
```

8. Referencias bibliográficas

- ab - Apache HTTP server benchmarking tool [En línea]. Apache. [Consulta: de julio a agosto 2016]. Disponible en: https://httpd.apache.org/docs/2.4/programs/ab.html
- Apache Benchmark + GNUPlot: Medir y graficar el rendimiento de tu servidor web [En línea]. usemoslinux. [Consulta: de julio a agosto 2016]. Disponible en: http://blog.desdelinux.net/apache-benchmark-gnuplot-medir-rendimiento-de-servidor-web/
- Benchmarking and Load Testing with Siege [En línea]. Kevin Faustino, 29/11/2012. [Consulta: de julio a agosto 2016]. Disponible en: http://blog.remarkablelabs.com/2012/11/benchmarking-and-load-testing-with-siege
- Cómo detectar cambios en un directorio con Bash [En línea]. Jose Pablo, 21/11/2014. [Consulta: de julio a agosto 2016]. Disponible en: http://www.jpablo128.com/como-detectar-cambios-en-un-directorio-con-bash/
- Como ejecutar una tarea cada 10 segundos con crontab [En línea]. Osbeta, 15/06/2011. [Consulta: de julio a agosto 2016]. Disponible en: http://proyectosbeta.blogspot.com.es/2011/06/como-ejecutar-una-tarea-cada-10.html
- Cluster de servidores, ¿qué es y cómo funciona? [En línea]. Solingest. [Consulta: de marzo a julio 2016]. Disponible en: http://www.solingest.com/blog/cluster-de-servidores-que-es-y-como-funciona
- HAProxy [En línea]. HAProxy, 14/08/2016. [Consulta: de marzo a julio 2016]. Disponible en: http://www.haproxy.org
- How To Implement SSL Termination With HAProxy on Ubuntu 14.04 [En línea]. Mitchell Anicas, 10/07/2014. [Consulta: de marzo a julio 2016]. Disponible en: https://www.digitalocean.com/community/tutorials/how-to-implement-ssl-termination-with-haproxy-on-ubuntu-14-04
- How to Install HAProxy Load Balancer on Ubuntu [En línea]. UpCloud. [Consulta: de marzo a julio 2016]. Disponible en: https://www.upcloud.com/support/haproxy-load-balancer-ubuntu/
- How To Optimize WordPress Performance With MySQL Replication On Ubuntu 14.04 [En línea]. Mitchell Anicas, 21/05/2014. [Consulta: de julio a agosto 2016]. Disponible en: https://www.digitalocean.com/community/tutorials/how-to-optimize-wordpress-performance-with-mysql-replication-on-ubuntu-14-04
- How To Set Up a Remote Database to Optimize Site Performance with MySQL [En línea]. Justin Ellingwood, 17/04/2014. [Consulta: de julio a agosto 2016]. Disponible en: https://www.digitalocean.com/community/tutorials/how-to-set-up-a-remote-database-to-optimize-site-performance-with-mysql
- HOW TO SETUP HAPROXY WITH KEEPALIVED [En línea]. Dasun Hegoda, 18/06/2015. [Consulta: de marzo a julio 2016]. Disponible en: http://dasunhegoda.com/how-to-setup-haproxy-with-keepalived/833/
- Installing HAProxy on Ubuntu 14.04 [En línea]. Vultr Docs, 25/06/2015. [Consulta: de marzo a julio 2016]. Disponible en: https://www.vultr.com/docs/installing-and-configuring-haproxy-on-ubuntu-14-04
- Keepalived [En línea]. Keepalived, 11/07/2016. [Consulta: de marzo a julio 2016]. Disponible en: http://www.keepalived.org/

- Load Testing your Site with Siege [En línea]. William Hetherington, 21/07/2015. [Consulta: de julio a agosto 2016]. Disponible en: https://drupalize.me/blog/201507/load-testing-your-site-siege
- López Rodríguez, Pedro Juan; Baydal Cardona, María Elvira. Asignatura del Máster de Ingeniería de Computadores y Redes: *Configuración, Administración y Utilización de Clusters de PCs*. Dpto. de Informática de Sistemas y Computadores (DISCA, Universidad Politécnica de Valencia). Valencia.
- MANEJO DE ARCHIVOS Y DIRECTORIOS A TRAVES DE COMANDOS [En línea]. Investigación. [Consulta: de julio a agosto 2016]. Disponible en: http://www.investigacion.frc.utn.edu.ar/labsis/Publicaciones/apunte_linux/mmad.html
- RedHat [En línea]. RedHat, 2015. [Consulta: de marzo a julio 2016]. Disponible en: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Load_Balancer_Administration/ch-lvs-overview-VSA.html
- SEND USER TO THE SAME BACKEND FOR BOTH HTTP AND HTTPS [En línea]. BAPTISTE ASSMANN, 12/07/2011. [Consulta: de marzo a julio 2016]. Disponible en: http://blog.haproxy.com/2011/07/12/send-user-to-the-same-backend-for-both-http-and-https/
- Siege Manual [En línea]. Joe Dog, 30/01/2012. [Consulta: de julio a agosto 2016]. Disponible en: https://www.joedog.org/siege-manual/
- Siege! Benchmarking your web applications [En línea]. Ben Everard, 23/11/2011. [Consulta: de julio a agosto 2016]. Disponible en: https://beneverard.co.uk/blog/siege-benchmarking-your-web-applications/
- Sincronizando carpetas con rsync [En línea]. Juan, 10/11/2010. [Consulta: de julio a agosto 2016]. Disponible en: http://www.jveweb.net/archivo/2010/11/sincronizando-carpetas-con-rsync.html
- Using SSL Certificates with HAProxy [En línea]. Servers for Hackers, 29/06/2014. [Consulta: de marzo a julio 2016]. Disponible en: https://serversforhackers.com/using-ssl-certificates-with-haproxy
- Using SSL/HTTPS with HAProxy [En línea]. Sean McGary, 27/06/2014. [En línea]. [Consulta: de marzo a julio 2016]. Disponible en: https://seanmcgary.com/posts/using-sslhttps-with-haproxy
- VRRP (*Virtual Router Redundancy Protocol*). Request for Comments: 5798. [En línea]. S. Nadas, Ed. Ericsson. [Consulta: julio 2016]. Disponible en: https://tools.ietf.org/html/rfc5798
- Web Server Load-Balancing with HAProxy on Ubuntu 14.04 [En línea]. Sanchit Jain Rasiya. [Consulta: de marzo a julio 2016]. Disponible en: https://www.howtoforge.com/tutorial/ubuntu-load-balancer-haproxy/