PARTE 1.- Configuración y ejecución de Apache

1. Introducción

En esta parte veremos una configuración simple del servidor web Apache. El objetivo no es aprender a configurar este servidor, para ello pueden remitirse a las prácticas de la asignatura *32558 – Laboratorio de Internet*. En esta parte se dejan disponibles todos los ficheros de configuración necesarios y sólo deberán entender el funcionamiento del servidor con estos parámetros, así como el comportamiento del script PHP que se empleará en la evaluación del sistema. El servidor web se ejecutará en una máquina virtual FreeBSD para el servidor preparada para los efectos de este ejercicio.

2. Virtualización

La virtualización de hardware permite en sistemas operativos modernos crear un entorno virtual en la máquina anfitrión ("host") que emula un ordenador genérico de forma que otro sistema operativo ("guest") puede ejecutarse simultáneamente. El sistema operativo guest ve el hardware tal y como se lo ofrece el sistema de virtualización. Estas soluciones permiten correr varias máquinas virtuales (VM = "Virtual Machine") simultáneamente y de forma independiente sobre el mismo ordenador (en función de sus recursos). Suelen permitir que los guest accedan a recursos hardware como son los lectores de CD/DVD, tarjeta de red, dispositivos de entrada y salida de audio, además de, evidentemente, disco duro, teclado, ratón y pantalla.

La virtualización se está volviendo muy común en el ámbito de servidores porque se puede trasladar la VM de una computadora a otra sin necesidad de reinstalar el sistema operativo guest. Esto es especialmente útil en los casos de fallos del hardware o de querer actualizar el mismo.

Emplearemos virtualización en esta práctica para crear la máquina servidor que ejecutará Apache. La máquina virtual vendrá ya preparada, con una instalación de FreeBSD 8[1], Apache 2.2 con PHP 5 y los ficheros de configuración necesarios.

3. Máquina virtual pre-configurada

Una máquina virtual contiene una instalación completa del un sistema operativo (S.O.). Esto puede ocupar más o menos espacio según el S.O. y los paquetes instalados. En este caso se ha creado una máquina que ocupa aproximadamente 1.5GB. Debido a su tamaño no se va a poder guardar en los directorios HOME de los grupos de prácticas y vamos a utilizar el disco duro local de cada máquina. En el directorio /opt/rss de cada máquina se ha dejado una copia de la máquina virtual y se ha creado un directorio /opt/rss/practica en el que tienen permiso de escritura. Haga una copia de la máquina virtual para tener permisos para modificarla:

```
$ cp -r /opt/rss/FreeBSD8_RSS /opt/rss/practica/VM_MIUSUARIO
```

A continuación lancen VMWare Workstation, bien desde el menú de aplicaciones ("System Tools") o desde terminal ("vmware"). Puede que le pida la licencia. Si es así:

```
$ mkdir .vmware
$ cp /etc/license-ws-70-e1-200904 $HOME/.vmware
```

y vuelva a lanzar el VMWare Workstation.

[1] http://www.freebsd.org/

Seleccione “Open” del menú o use el botón llamado “Open a virtual machine or team” (Ilustración 1). Navegue hasta abrir el fichero:

```
/opt/rss/practica/VM_MIUSUARIO/FreeBSD8RSS.vmx
```

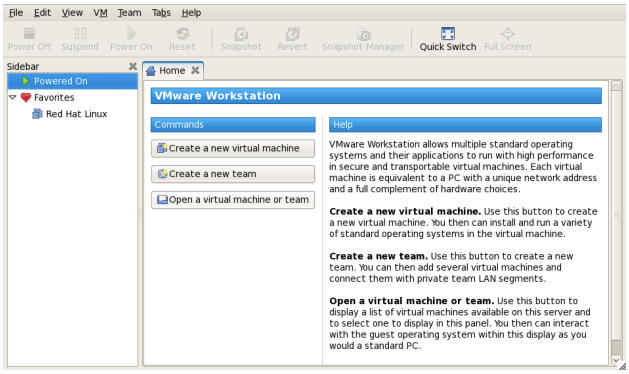

Ilustración 1 - Interfaz de VMWare Workstation

Arranque la VM (botón “Power on this virtual machine”). Si le pregunta si la ha movido o copiado seleccione que la ha copiado (“I copied it”, Ilustración 2).

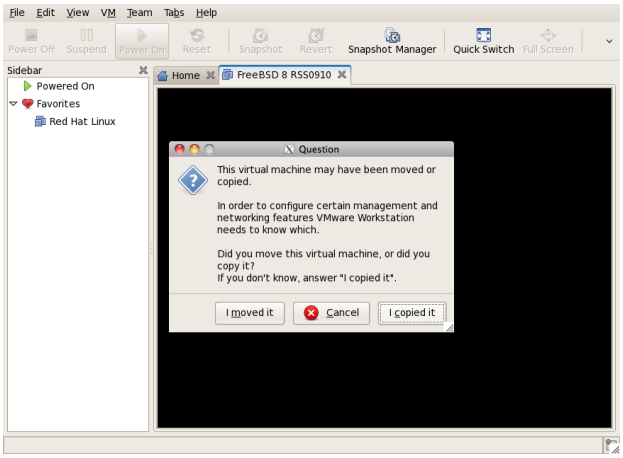

Ilustración 2 - Arrancando la máquina virtual la primera vez

Con eso debería arrancar el guest FreeBSD, mostrando su salida de consola en la ventana. Pulsen en la misma para poder escribir y para poder volver a usar el ratón en el host pulsen Ctr+Alt como dice el margen inferior izquierdo de la ventana de VMWare. Esta máquina tiene creados los usuarios “root”, “guest” y “tlm”, todos sin contraseña. Entre con el usuario **guest** que tiene preparado el directorio dir_httpd en su *HOME*, con todos los ficheros necesarios para lanzar Apache.

4. Ficheros necesarios

Normalmente Apache emplea ficheros de configuración que se encuentren en directorios que fueron especificados en su instalación y que suelen requerir privilegios de superusuarios para ser modificados. Sin embargo, también se le puede indicar la localización de los ficheros al lanzarlo. Vamos a emplear esta segunda opción, lo cual nos permitirá tener los ficheros de configuración en el HOME de un usuario normal (en este caso su usuario de prácticas).

Los ficheros preparados para este ejercicio se encuentran en */home/guest/dir_httpd/*.

El directorio contiene la siguiente estructura de ficheros:

```
$ tree dir_httpd
dir_httpd
```

```
|-- htdocs
|   |-- phpinfo.php
|   |-- prueba.html
|   `-- randwait.php
`-- conf
    |-- httpd.conf
    |-- logs
    `-- run

4 directories, 4 files
```

El fichero *httpd.conf* contiene una configuración mínima de Apache, con las siguientes directivas:

```
# Run params
ServerRoot /usr/local
PidFile /home/VM_USUARIO/dir_httpd/conf/run/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15
UseCanonicalName Off
ErrorLog /home/VM_USUARIO/dir_httpd/conf/logs/error_log
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /home/VM_USUARIO/dir_httpd/conf/logs/access_log common
AcceptFilter http none

ServerName 127.0.0.1

# Port
Listen 8087

# User & Group
User VM_USUARIO
Group VM_USUARIO

# Modules
LoadModule authz_host_module libexec/apache22/mod_authz_host.so
LoadModule mime_module libexec/apache22/mod_mime.so
LoadModule mime_magic_module libexec/apache22/mod_mime_magic.so
LoadModule autoindex_module libexec/apache22/mod_autoindex.so
LoadModule dir_module libexec/apache22/mod_dir.so
LoadModule php5_module libexec/apache22/libphp5.so
LoadModule env_module libexec/apache22/mod_env.so
LoadModule log_config_module libexec/apache22/mod_log_config.so

# prefork MPM
# StartServers: number of server processes to start
# MinSpareServers: minimum number of server processes which are kept spare
# MaxSpareServers: maximum number of server processes which are kept spare
# MaxClients: maximum number of server processes allowed to start
# MaxRequestsPerChild: maximum number of requests a server process serves
<IfModule mpm_prefork_module>
    StartServers          1
    MinSpareServers       1
    MaxSpareServers       1
    MaxClients            1
    MaxRequestsPerChild   0
    LockFile /home/VM_USUARIO/dir_httpd/conf/logs/accept.lock
</IfModule>

# Maximum length of the queue of pending connections
ListenBacklog 1

# This forces the request to be treated as a HTTP/1.0 request even if it was in a later
dialect
SetEnv downgrade-1.0

# Documents directory
DocumentRoot /home/VM_USUARIO/dir_httpd/htdocs
<Directory />
      Options FollowSymLinks
```

```
        AllowOverride All
</Directory>
<Directory /home/VM_USUARIO/dir_httpd/htdocs>
        Options FollowSymLinks
        AllowOverride All
        Order allow,deny
        Allow from all
</Directory>

# Types
DefaultType text/plain
TypesConfig etc/apache22/mime.types
<IfModule mod_mime_magic.c>
        MIMEMagicFile etc/apache22/magic
</IfModule>
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
AddType application/x-httpd-php .php
```

En este fichero deberá sustituir la cadena *VM_USUARIO* con el nombre del usuario de la máquina virtual, de forma que las rutas a los ficheros sean correctas y el usuario con el que vaya a correr Apache sea el que está usando.

Puede editar los ficheros del VM desde la línea de comandos del VM mediante el programa de edición en línea vi o desde el host usando un programa tipo *gedit* y cargando el fichero via *sftp*. Por ejemplo, lance desde la línea de comandos la siguiente instrucción para abrir el fichero *httpd.conf*:

```
$ gedit sftp://VM_USUARIO@VM_IP/home/VM_USUARIO/dir_httpd/conf/httpd.conf &
```

En la web de documentación del servidor web pueden encontrar descrito el objetivo y modo de uso de cada una de sus directivas[1]. Sin embargo, a continuación resumiremos brevemente cómo emplear las pocas que se considerarán parámetros en este trabajo.

5. Configurando y lanzando Apache

Puede lanzar el servidor con la siguiente instrucción (donde de nuevo debe sustituir *VM_USUARIO* con el nombre del usuario de la máquina virtual):

```
$ apachectl -f /home/VM_USUARIO/dir_httpd/conf/httpd.conf -k start
```

Puede comprobar que el servidor está corriendo de la siguiente forma:

```
$ ps auxw | grep httpd.conf
VM_USUARIO     9760   0.0   0.1   22896   5588 ?          Ss    22:36    0:00 /usr/sbin/httpd -f
/home/VM_USUARIO/dir_httpd/conf/httpd.conf -k start
VM_USUARIO     9762   0.0   0.0   22896   2560 ?          S     22:36    0:00 /usr/sbin/httpd -f
/home/VM_USUARIO/dir_httpd/conf/httpd.conf -k start
```

Para detenerlo habría que hacer:

```
$ apachectl -f /home/VM_USUARIO/dir_httpd/conf/httpd.conf -k stop
```

6. Puerto TCP empleado por el servidor

Lanzaremos Apache en un puerto no reservado, de forma que no haya necesidad de privilegios de superusuario en las máquinas del laboratorio. Esto se controla mediante la directiva *Listen*. Como se puede ver, en el fichero que se ha ofrecido se indica el puerto 8087 para el servidor. Pueden comprobar el estado de los sockets TCP del servidor mediante:

```
$ netstat -an | grep 8087
tcp        0      0 :::8087                     :::*                        LISTEN
```

En este caso, vemos solamente el socket inicial del servidor en estado LISTEN (puede que

[1] http://httpd.apache.org/docs/2.2/

aparezca con su traducción en castellano, ESCUCHAR), esperando solicitudes de establecimiento de conexión (es decir, mensajes TCP con el flag SYN activo). Mientras el servidor tenga alguna conexión establecida con un cliente (normalmente un navegador web) se verá de esta forma:

```
$ netstat -an | grep 8087
tcp        0      0 :::8087                     :::*                        LISTEN
tcp        0      0 ::ffff:10.1.1.25:8087       ::ffff:10.1.1.11:49826      ESTABLISHED
```

(puede que ESTABLISHED aparezca con su traducción en castellano, ESTABLECIDO)

7. Directorio con las páginas/scripts servidos

Especificamos el directorio principal que contiene los ficheros ofrecidos por el servidor web mediante la directiva *DocumentRoot*, en este caso:

```
DocumentRoot /home/VM_USUARIO/dir_httpd/htdocs
```

Pueden lanzar un navegador web (Firefox) en el host (no en la máquina virtual) y acceder a una de las páginas de ejemplo que se les ha dejado con el siguiente URL:

http://VM_IP:8087/prueba.html

Si no obtienen el fichero *$HOME/dir_httpd/htdocs/prueba.html* revisen los pasos anteriores.

A continuación comprobamos el correcto funcionamiento del módulo PHP accediendo a este script:

http://VM_IP:8087/phpinfo.php

Si pueden ver el contenido del fichero *$HOME/dir_httpd/htdocs/phpinfo.php* entonces hay algo mal configurado, revisen los pasos anteriores. En caso de que el servidor esté correctamente configurado deberán obtener una larga página HTML con información sobre la configuración y submódulos incluidos en el módulo PHP del servidor.

8. Número de procesos y peticiones simultáneas

Apache (con el módulo MPM prefork[1]) emplea concurrencia de procesos para servir múltiples peticiones que se solapan en el tiempo. Es decir, lanza varios procesos (creados empleando fork()) y cada uno de ellos atiende a una petición hasta que termina de servirla, momento en que puede atender a otra. Así, si tiene 10 procesos podrá estar sirviendo hasta 10 páginas simultáneamente a diferentes clientes. Podemos controlar la cantidad de procesos que lanza Apache mediante las directivas *StartServers, MinSpareServers, MaxSpareServers y MaxClients*. No vamos a entrar en mucho detalle sobre el uso de cada directiva y se recomienda para ello la documentación de Apache. Hemos indicado en el fichero de configuración de Apache de ejemplo:

```
StartServers          1
MinSpareServers       1
MaxSpareServers       1
MaxClients            1
```

con lo que Apache solo aceptará servir una petición a la vez (*MaxClients* = 1). Podríamos incrementar el número de peticiones simultáneas que sirve incrementando el valor de *MaxClients*, lo cual haría que Apache lanzara más procesos para atender dichas peticiones. Si alcanza este valor máximo de clientes simultáneos, nuevas peticiones (hasta que se complete una de las solicitudes) quedarán a la espera en cola. El sistema operativo aceptará la conexión TCP solicitada por el cliente web pero el servidor no atenderá a dicha petición hasta tener libre

[1] http://httpd.apache.org/docs/2.0/en/mod/prefork.html

un servidor (de forma que mantenga el número de servidores ocupados como máximo en el valor de *MaxClients*). Se puede controlar el tamaño máximo de esta cola de espera mediante la directiva *ListenBacklog*[2]. En el fichero de configuración que se ha entregado se ha configurado:

```
ListenBacklog 1
```

lo cual quiere decir que como mucho se mantendrá una conexión TCP aceptada en cola a la espera de que quede un servidor libre. Peticiones (segmentos TCP con bit de SYN activo al puerto del servidor Web) que lleguen mientras haya una aceptada en cola no completarán la conexión TCP.

9. Versión de HTTP

Las peticiones del elemento “Petición HTTP" implementan la versión 1.1 de dicho protocolo. En este trabajo vamos a centrarnos en la versión 1.0 de HTTP, en la cual cada recurso que se solicita al servidor se hace en una conexión TCP independiente, cerrando el servidor la conexión al terminar de enviar el recurso. Dado que no hay forma (o no la hemos encontrado) de indicarle a JMeter que emplee la versión 1.0 del protocolo, se puede forzar esto en el servidor, y es lo que hemos hecho mediante la directiva:

```
SetEnv downgrade-1.0
```

10.randwait.php

Para este trabajo se ha preparado un pequeño script PHP el cual simulará el funcionamiento del servidor de aplicaciones de la empresa. Dicho servidor se supone que recibe peticiones de sus clientes, lleva a cabo la operación que se le ha solicitado, que le lleva un tiempo variable según la petición, y finalmente entrega una respuesta al cliente, cerrando a continuación la conexión. El script simula esto aceptando la petición de un navegador web, esperando durante un tiempo para simular el procesamiento de dicho servidor y finalizando a continuación la conexión tras entregar al navegador un pequeño texto diciendo que ha completado la operación.

El script acepta en la *query string* los siguientes argumentos:

- *distribution* : se indica aquí el tipo de distribución para la variable aleatoria del tiempo que tarda el servidor en procesar la petición. En estos momentos los valores posibles son:
    - *deterministic* : espera un tiempo predeterminado por el argumento *average*
    - *exponential* : espera un tiempo distribuido según una variable aleatoria exponencial de media el valor indicado en el argumento *average*
    - *uniform* : espera un tiempo distribuido según una variable aleatoria uniforme en [0, 2 x average]
- *average* : se indica con este argumento el tiempo medio que debe esperar el servidor a entregar el final de la respuesta. Está medido en microsegundos.

Ejemplos:

1. Esperar 10 segundos:

   http://VM_IP:8087/randwait.php?distribution=deterministic&average=10000000

[2] http://httpd.apache.org/docs/2.0/en/mod/mpm_common.html#listenbacklog

2. Esperar un tiempo aleatorio según una distribución exponencial de media 100ms

   http://VM_IP:8087/randwait.php?distribution=exponential&average=100000

   Familiarícese a continuación con el funcionamiento de este script PHP así como de las configuración de Apache en lo que a limitación de su concurrencia se refiere. Puede ver los accesos que recibe el servidor web en el fichero *access_log* que se ha configurado.