

El servidor de web Apache: Introducción práctica

Apache 1.x y 2.0 alpha

Alvaro del Castillo San Félix
Desarrollador y admin de software libre
Barrapunto

acs@barrapunto.com
<http://barrapunto.com>

El servidor de web Apache: Introducción práctica: Apache 1.x y 2.0 alpha

by Alvaro del Castillo San Félix

Copyright © 2000 by Alvaro del Castillo San Félix bajo FDL

Prólogo

Este artículo describe el servidor de web Apache, su instalación y su configuración, en la versión 1.3.x y 2.0. La plataforma objetivo es GNU/Linux.

Table of Contents

1. El servidor de web Apache.....	4
Introducción	4
Características	4
El protocolo HTTP	5
Obteniendo el servidor	8
Configuración del servidor	10
El fichero de configuración httpd.conf	10
Documentación	13
Proyectos asociados	13
PHP	14
Apache-SSL.....	14
Conclusiones Apache	15
Apache 2.0	15
Referencias	18
A.	19

Chapter 1. El servidor de web Apache

Introducción

Lo primero de todo es decir que este artículo está basado en uno que escribí en Diciembre de 1997 y se publicó en Linux Actual. Me ha sorprendido lo poco que he tenido que tocarlo para actualizarlo y como Apache se ha consolidado más aún de lo que estaba por aquellas fechas. Si el lector encuentra alguna errata u información incorrecta puede estaré encantado de que me corrija.

He introducido un apartado dedicado a Apache 2.0 que sin duda será el más interesante para los lectores que lean este artículo en la actualidad.

El servidor web Apache es uno de los mayores triunfos del software libre. En Diciembre de 1997 tenía una cuota de mercado cercana al 45% y en la actualidad (Julio 2000) ya está por encima del 60%, según los estudios de Netcraft [3] que ya se han establecido como la referencia dentro del mercado de servidores web. Esta es la primera cifra que hace que cualquier responsable de la estrategia internet de una empresa tenga que tomar a Apache como el servidor de referencia.

Apache era inicialmente unos parches al servidor que de WWW de NCSA conocido como httpd (principios de 1995). Al igual que GNU/Linux, fue un proyecto que atrajo a mucha gente por el gran interés de su objetivo: lograr el servidor web más rápido, más eficiente y con mayor funcionalidad desde el enfoque del software libre. Y ha sido un objetivo que como veremos, se ha logrado. Sólo hacían sobras en Apache ciertos aspectos de rendimiento, fundamentalmente por no utilizar hebras de ejecución. Este aspecto se va a solucionar en el próximo Apache 2.0, un nuevo paso de Apache.

Con un enorme equipo de voluntarios a lo largo y ancho de toda la red, se han logrado estos objetivos, logrando batir a compañías comerciales de la talla de Microsoft y Netscape. Y no sólo se ha logrado batir a los servidores web de grandes compañías: se ha logrado atraer al todopoderoso IBM que está apoyando Apache tanto a nivel de soporte como de desarrollo. La incorporación de empresas al desarrollo de proyectos de software abierto es una característica que ha brotado en el año 2000. Otros ejemplos son Corel, Dell etc.

Características

Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los

nuevos protocolos (HTTP 1.1). Entre sus características destacan:

- Multiplataforma
- Es un servidor de web conforme al protocolo HTTP/1.1
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- Basado en hebras en la versión 2.0
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- Se desarrolla de forma abierta
- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.

El protocolo HTTP

El protocolo HTTP es el que da vida a Internet, y gracias al cual, los clientes y servidores se pueden comunicar.

El lector, si tiene experiencia en el campo de protocolos, puede pensar que esta es la parte más compleja del Web. Pues bien, este protocolo se diseñó con la sencillez en mente, por lo que es de lo más trivial.

El funcionamiento básico es que el cliente establece una conexión TCP con el servidor, hace una petición, el servidor le responde y se cierra la conexión. Para que se haga una idea del lector de la sencillez, en la primera versión ampliamente utilizada del protocolo (1.0), el cliente solo podía invocar tres operaciones en el servidor: GET para pedir una página, HEAD para pedir la cabecera de una página y POST para enviar datos a una URL.

Siento un poco más estrictos, el funcionamiento del protocolo es:

- El cliente envía una petición al servidor. Dicha petición está compuesta por un método a invocar en el servidor (URI) y una versión del protocolo, seguida por un mensaje compatible con MIME con los parámetros de la petición, información del cliente, y un cuerpo opcional con más datos para el servidor. Un ejemplo es:

```
GET /index.html HTTP/1.0
Accept: text/plain
Accept: text/html
Accept: */*
```

User-Agent: Un Agente de Usuario Cualquiera

- El servidor responde con una línea de estado, incluyendo la versión del protocolo del mensaje y si la petición tuvo éxito o fracaso, con un código de resultado, seguido de un mensaje compatible con MIME con información del servidor, metainformación (datos a cerca de la información) de la entidad solicitada y un cuerpo opcional con la entidad solicitada. Un ejemplo es:

```
HTTP/1.0 200 OK
Server: MDMA/0.1
MIME-version: 1.0
Content-type: text/html
Last-Modified: Thu Jul 7 00:25:33 1994
Content-Length: 2003
<title>Página de web del IEEE de Madrid<title>
<hr>
....
<hr>
<h2> Proyectos desarrollados en Internet </h2>
<hr>
```

Pero como todo en el Web, este protocolo (versión 1.1) ya es mucho más potente que en su versión original, y como luego veremos, en total hay ya trece métodos diferentes, además de un conjunto de características nuevas como por ejemplo, el tiempo tras el cual el cliente debe volver a recargar la página.

Los creadores del HTTP 1.1 lo describen como: *"un protocolo de nivel de aplicación orientado a sistemas distribuidos, para la colaboración e hypermedia. Un protocolo genérico, sin estado, orientado a objetos y que puede ser utilizado para muchas aplicaciones, como servidores de nombres y sistemas de gestión de objetos distribuidos, a través de las extensiones de los métodos de petición. Una característica de este protocolo es la negociación de los tipos y representación de los datos, permitiendo que los sistemas no dependan del tipo de datos que se utilicen"*.

Los mismos creador de HTTP 1.0 son conscientes de las limitaciones de escalabilidad y rendimiento del protocolo, por lo que recomiendan que ningún otro servidor lo utilice, y que se utilice de forma única el HTTP 1.1

Los problemas principales que existen en la versión 1.0 son de rendimiento. Esto esta perfectamente documentado dentro de <http://www.w3.org>, y destacamos aquí las conclusiones principales a las que llegaron, pero recomendamos al lector interesado en protocolos que consulte dicho documento:

- Las conexiones del protocolo TCP son lentas de establecer (conexión en tres pasos y ajuste de ventanas de recepción de datos), y como por cada página y cada imagen que haya en la página, ha de establecerse una conexión nueva, la transmisión de datos está ralentizada por el establecimiento de la conexión TCP.
- Una conexión para transmitir 1Kbyte de datos tarda alrededor de 500 ms.

- Tras cerrar una conexión TCP, el puerto del servidor utilizado en dicha conexión, se queda en estado TIME_WAIT un tiempo recomendado de 240 segundos, por lo que un servidor que reciba muchas peticiones puede agotar todos los puertos TCP (que recordemos son 65535) y dejar al servidor sin posibilidad de enviar ningún tipo de dato. Esto supone un problema de escalabilidad muy importante.

Por lo tanto, a partir de este momento nos centraremos únicamente en el protocolo HTTP 1.1, ya que la versión 1.0 es historia, aunque una historia muy presente que nos puede saltar así que es mejor tenerlo presente.

Esta nueva versión de HTTP está recogida dentro de la RFC 2068 de Enero de 1997, la cual se puede obtener en [1]. Las principales características de esta nueva versión son:

- Conexiones persistentes: ya no se cierra la conexión tras el envío de cada parte de un documento, evitando la sobrecarga del establecimiento de conexiones TCP.
- Varias peticiones simultáneas: un cliente puede realizar varias peticiones utilizando una única conexión, sin esperar a la respuesta del servidor para cada una de ellas.
- Negociación del contenido: se asignan diferentes valores a las características de la comunicación, entre ellos cuanto se puede degradar la calidad de la conexión,
- Nuevos métodos: junto a GET, POST y HEAD aparecen los métodos DELETE para borrar un recurso del servidor asociado al URI de borrado, TRACE para ver que está recibiendo el servidor de lo que él envía, PUT para enviar datos a un recurso asociado a una URI, PATCH para aplicar correcciones en un recurso asociado a una URI, COPY para copiar unos recursos identificados por una URI en otro lugar determinada URI en uno destino determinado, MOVE para mover el recurso identificado por la URI a otro lugar, DELETE para borrar un recurso asociado a una URI, LINK para establecer enlaces entre diferentes recursos, UNLINK para quitar enlaces establecidos previamente por LINK, OPTIONS para que el cliente pueda obtener del servidor sus características, WRAPPED que permite unir varias peticiones y recubrirlas con algún tipo de filtrado (encriptación por ejemplo).
- Nuevo método de autenticación: en la RFC 2069 se describe un nuevo método de autenticación, en el cual las claves de acceso van encriptadas por la red, al contrario de lo que ocurre en HTTP 1.0. Esta RFC aún no se ha unificado con la RFC 2068 para formar la especificación de HTTP 1.1, pero se está en vías de ello.

Junto con estas mejoras, hay muchas ampliaciones en el intercambio de información entre el cliente y el servidor, lo cual ha hecho que la RFC 1954 (Mayo 1996) que describe HTTP 1.0 haya pasado de 60 páginas a 161 página en la RFC 2068 que describe HTTP 1.1. O que el número de autores haya pasado de cuatro a seis, entre ellos Tim Berners-Lee, el "inventor" del web.

El estado de las conexiones sigue sin poderse mantener, a menos que utilicemos mecanismos auxiliares como las cookies (RFC 2109) , pero si bien el no mantener el estado de la conexión tiene sus desventajas, también tiene una importancia fundamental en una red como Internet: algunas de las peticiones HTTP son idempotentes, es decir, que el resultado de invocar un número arbitrario de veces en

el servidor, tiene los mismo efectos. Los métodos GET, HEAD, PUT y DELETE tienen esta propiedad. Es importante esta propiedad ya que, para asegurarnos que algo ocurra, y que sea exactamente lo que nosotros queremos que ocurra, podemos invocar estos métodos repetidas veces, de forma que aunque una petición no llegue al servidor, pueda llegar alguna de las peticiones que repiten a esta primera. En la RFC 2068 también se habla de métodos seguros, que son aquellos que su invocación no conlleva ningún riesgo, es decir, que su invocación no tiene efectos laterales en el servidor. Estos métodos son GET y HEAD, que lo único que suponen es una petición de información.

Pero esta nueva versión (HTTP 1.1) es un puente hacia lo que en realidad se quiere imponer en Internet: HTTP-NG (HTTP Next Generation). Este nuevo protocolo pretende cubrir una gran cantidad de nuevas funcionalidades, entre la que destaca el comercio electrónico. Sus criterios de diseño han sido:

- **Simplicidad:** no se debe abandonar el criterio introducido en HTTP 1.0, que las cosas habituales sean sencillas, de forma que sea fácil implementar el protocolo
- **Rendimiento:** debe ser eficiente transmitiendo objetos en redes de comunicaciones
- **Asíncrono:** las peticiones desde los clientes han de poderse hacer en paralelo a través de una única conexión
- **Seguridad:** los objetos que se transmiten deben estar encriptados, sin forzar ninguna política de seguridad en particular.
- **Autenticación:** se debe poder autenticar a las dos partes de la conexión, así como a cualquier intermediario. Pagos en línea: el protocolo debe soportar la realización de pagos en línea
- **Servidores intermediarios:** se debe soportar la comunicación entre servidores, para el mantenimiento de cachés, espejos de datos e intermediarios de comunicación (proxys).
- **Visualización obligatoria:** se debe poder obligar al cliente a mostrar ciertos datos acerca del objeto que se transmite, como el autor del objeto, el copyright y la licencia.
- **Información de registro:** la información de registro (logs) ha de poder ser enviada entre diferentes servidores.
- **Requerimientos de red:** el protocolo debe trabajar de forma independiente de la capa de transporte de la que disponga, aunque debe funcionar especialmente bien con TCP, al ser el protocolo más utilizado en Internet.

Como puede observar el lector, los avances en HTTP 1.1 (y más aún en HTTPng), son muy importantes, y un servidor como Apache (cuya cuota de mercado es superior al 45%) en su versión 1.2 ya tiene incluido este nuevo protocolo. Y todos los clientes de WWW lo soportarán, dando una nueva vitalidad al día a día en Internet.

En Apache 3.0, que aún está en un futuro lejano, el principal y único objetivo a día de hoy es dar soporte a HTTP-NG. Si Apache implementa este protocolo de forma automática puede alcanzar al 60% del mercado, lo que le daría un impulso definitivo. Veremos si termina ocurriendo.

Obteniendo el servidor

Como es habitual en el mundo del software libre, el servidor de web Apache lo podemos obtener de su lugar original [1], o bien de espejos a lo largo y ancho de todo el mundo, entre ellos varios en España [5]. El fichero que nos deberemos bajar es "apache_1.3b3.tar.gz" en el que se incluyen las fuentes del servidor, que han sufrido una fuerte remodelación en esta nueva versión, así como la documentación del servidor de web

Instalación del servidor

Tras obtener las fuentes de Apache 1.3b3 pasamos a descomprimirlas. Para ello, lo más recomendable es copiar el fichero "apache_1.3b3.tar.gz" a un directorio en nuestra cuenta en el sistema Linux (no es recomendable hacerlo como superusuario) y desde allí, ejecutar la orden: `tar xvzf apache_1.3b3.tar.gz` Con ello obtendremos una estructura de directorios como la que sigue:

- `./apache_1.3b3/cgi-bin`: cgi-bin de utilidad genérica
- `/conf`: ficheros de configuración de apache
- `/docs`: documentación y FAQ de apache 1.3
- `/htdocs`: páginas html del servidor
- `/icons`: iconos que utiliza el servidor
- `/src`: fuentes de apache

En el directorio principal "apache_1.3b3" encontramos ya un "Makefile" y, en un sistema Debian 1.3.1 se puede ejecutar directamente `make` en este directorio, y el servidor de web es construido sin ningún problema. Con ello obtenemos el conocido como servidor de web estándar, el cual incluye una funcionalidad muy genérica, sin ningún tipo de opción avanzada.

Pero una de las características fundamentales de Apache es su flexibilidad a la hora de ser configurado. Dicha configuración está basada en módulos, que pueden ser añadidos o eliminados según nos interese su funcionalidad o no. Para consultar los módulos que están disponibles, lo mejor es ir a la documentación, en la que se describen todos los módulos, su funcionalidad y como se configuran sus diferentes opciones. En el servidor web estándar sólo se incluyen los módulos que aparecen definidos dentro del fichero

```
./apache_1.3b3/src/Configuration.tpl
```

Dicho fichero contiene información acerca de lo que se debe incluir dentro del servidor, y es recomendable leerlo. Es un fichero muy legible, en el que se muestra una gran variedad de módulos, con su descripción, y entre ellos sólo se activan los fundamentales. En la instalación estándar dicho fichero se copia al fichero `./apache_1.3b3/src/Configuration` que es utilizado por el script `./apache_1.3b3/src/Configure` para la creación de los diferentes "Makefile" que construyen el servidor de web. Un usuario que quiera personalizar lo que se incluye dentro del servidor, deberá copiar el fichero `./apache_1.3b3/src/Configuration.tpl` a `./apache_1.3b3/src/Configuration`, editarlo, añadir o eliminar los módulos que le interesen, ejecutar `./apache_1.3b3/src/Configure` y dentro del directorio

`./apache_1.3b3/src/` ejecutar `make`. Tras ello obtiene el ejecutable `httpd` que es el demonio de web que queríamos.

Una vez finalizada la compilación, nos situamos en el directorio `./apache_1.3b3/` y adquirimos la personalidad de superusuario con la orden `su`, tras lo que ejecutamos `make install`. El servidor de web, junto con utilidades para lanzarlo, pararlo, y relanzarlo, así como los iconos por defecto, los `cgi-bin`'s genéricos, los logs, la documentación y unas páginas html de ejemplo, se crean dentro del directorio `/usr/local/apache/`. Como observará el lector, no interfiere con la versión de Apache que pudiéramos tener instalada con Debian, aunque eso sí, no podremos lanzarlo en el mismo puerto TCP (normalmente el 80) que el servidor de web actual. Para iniciar nuestro nuevo servidor de web Apache 1.3, bastará ejecutar la orden `/usr/local/apache/start`. En el caso de que haya algún problema, el lector podrá consultar el fichero `/usr/local/apache/logs/error.log` donde por ejemplo, si el puerto TCP ya está ocupado por otros servidor, nos indicará que no se puede enlazar el servidor web con dicho puerto.

Configuración del servidor

Los ficheros de configuración de Apache se buscan por defecto dentro del directorio `/usr/local/apache/conf` aunque esto es algo configurable. Allí deben de estar presentes los ficheros:

- `httpd.conf`: fichero principal de configuración de Apache.
- `srm.conf`: fichero de definición del espacio de nombres que los usuarios ven del servidor de web. En este fichero también se especifica donde se encuentran los `cgi-bin`, los iconos, el tipo de documento por defecto, como se responde ante los errores, que fichero es el índice dentro de un directorio, donde está la página personal de los usuarios del sistema.
- `access.conf`: fichero de control de acceso global a los datos del servidor de web. En el se especifica los permisos de accesos a directorios, ficheros y URLs dentro del servidor, así como diferentes configuraciones.
- `mime.types`: fichero de control de los tipos MIME que son enviados al cliente en función de la extensión del fichero.

El fichero principal es `httpd.conf`, por lo que a continuación se detalla su configuración. Los demás ficheros son de apoyo, y su comprensión es más sencilla.

El fichero de configuración `httpd.conf`

Es el fichero de configuración principal de Apache. Junto con la distribución, viene un ejemplo completo de las características que se controlan desde este fichero. Se muestran a continuación las principales

opciones de dicho fichero, que el usuario debe editar para adaptarlas a su situación concreta:

- **ServerType:** standalone o inetd El servidor de web puede ser arrancado, bien como un demonio individual, o bien como un demonio más dentro del conjunto de demonios que arranca inetd. Dadas las características de un servidor de web, que suele recibir peticiones de forma constante, es mejor arrancarlo de forma independiente, y no desde el inetd.
- **Port 80:** Puerto desde el que escucha el servidor de web. Un usuario del sistema, sin derechos de superusuario, solo puede utilizar puertos por encima del 1024.
- **HostnameLookups on u off:** Dentro de los logs del servidor, se intenta traducir la dirección IP del cliente que ha hecho la petición a un nombre utilizando DNS o no. El intento de traducción lleva asociada unas comunicaciones que pueden reducir el rendimiento del servidor.
- **User www-data Group #-1:** Identidad del usuario bajo el que se ejecuta el servidor. Si es diferente de la del usuario que lanza el demonio, el usuario que lanza el demonio debe de ser root.
- **ServerAdmin acs@ieeesb.etsit.upm.es:** Dirección de correo del usuario responsable de la administración del servidor de web, y al que se enviarán mensajes en caso de errores.
- **ServerRoot /usr/local/apache** Directorio donde se encuentra la configuración del servidor, los ficheros de logs y los de error.
- **#BindAddress *:** Con Apache se puede hacer que el servidor escuche en varias direcciones IP. Con esta opción se le dice al servidor que direcciones IP debe escuchar. Se describirá las características de servidores de web virtuales más adelante en este artículo.
- **ErrorLog logs/error_log:** Fichero donde se almacenan los errores, como peticiones de usuarios de ficheros inexistentes, o peticiones que no se pueden reconocer. El directorio es relativo a ServerRoot.
- **TransferLog logs/access_log:** Fichero donde se almacenan las transferencias de datos con los clientes. El directorio es relativo a ServerRoot. Este fichero es fundamental para sacar estadísticas de acceso a nuestro web.
- **PidFile logs/httpd.pid:** Fichero donde se almacena el pid (identificador del proceso del servidor). El directorio es relativo a ServerRoot.
- **ScoreBoardFile logs/apache_status:** Fichero que almacena información acerca del estado del servidor de web, estado que se actualiza en tiempo real, y que permite analizar a los administradores autorizados lo que está ocurriendo en el interior del servidor, como las peticiones actuales en curso, los recursos que está consumiendo el servidor.
- **ServerName www.ieeesb.etsit.upm.es** Este es el nombre que el servidor envía a los clientes, en caso de los clientes hayan accedido a la máquina con un nombre diferente. Una máquina puede tener varios nombres, y por cualquiera de ellos se puede acceder al puerto 80, donde suele estar el servidor de web. Pero normalmente, hay un nombre preferido para el web, algo como *www.dominio*. El servidor puede

indicar al cliente que en las peticiones futuras, debe utilizar ese nombre. El nombre ha de ser un nombre real registrado dentro del sistema DNS.

- Timeout 300: El número de segundos tras el cual se envía o se recibe el fin de plazo de una petición.
- KeepAlive On: Para permitir la existencia de conexiones persistentes (HTTP 1.1), es decir, de que por una misma conexión se puedan enviar varias peticiones HTTP.
- MaxKeepAliveRequests 100 Número máximo de peticiones que se pueden cursar por una misma conexión. En caso de poner 0, el número será ilimitado. Cuantas más peticiones se puedan cursar por una misma conexión, mejor será el rendimiento de la comunicación, al evitar establecer nuevas conexiones TCP, que son muy costosas.
- KeepAliveTimeout 15: Número de segundos máximos en los que se esperará la siguiente petición.
- MinSpareServers 5 - MaxSpareServers 10: número mínimo y máximo de servidores que serán lanzados en la máquina. Apache de forma dinámica lanzará un número entre los dos límites, dependiendo de la carga de las peticiones recibidas. Esta parte va a sufrir un importante cambio con la aparición de hebras en la versión 2.0 de Apache.
- StartServers 5: Número de servidores que se lanzan inicialmente.
- MaxClients 150: Número máximo de clientes que pueden conectarse al servidor de web. Debe ser un número relativamente alto, ya que una vez superado, ningún cliente más podrá acceder a la información. Es una método de protección, que evita que el sistema pueda quedar totalmente bloqueado.
- MaxRequestsPerChild 30: número máximo de peticiones que puede servir un proceso hijo del proceso servidor principal, antes de que sea eliminado. Esto es necesario en sistemas con agujeros en la gestión de memoria, como por ejemplo en algunas librerías de Solaris.
- #ProxyRequests On: Si queremos que nuestro servidor de web, haga también de servidor de proxy, es decir, que guardé una caché con las páginas más accedidas por los clientes, podemos activar esta funcionalidad con esta opción.
- - #CacheRoot /usr/local/apache/proxy
 - #CacheSize 5
 - #CacheGcInterval 4
 - #CacheMaxExpire 24
 - #CacheLastModifiedFactor 0.1
 - #CacheDefaultExpire 1
 - #NoCache www.sun.com

Diversas opciones sobre la caché, en el caso de que sea habilitada. Se debe indicar donde se guardan los datos, el tamaño de la caché, cuando expiran los documentos, y de que URL's no se debe hacer caché.

-
- #Listen 3000
- #Listen 12.34.56.78:80

Permite obligar al servidor de web a que escuche en una dirección IP determinada, junto con el puerto TCP. También escuchará el servidor en la dirección y puerto por defecto.

-
- #<VirtualHost host.some_domain.com>
- #ServerAdmin webmaster@host.some_domain.com
- #DocumentRoot /www/docs/host.some_domain.com
- #ServerName host.some_domain.com
- #ErrorLog logs/host.some_domain.com-error_log
- #TransferLog logs/host.some_domain.com-access_log #<VirtualHost>

Una de las características más destacadas en Apache es que puede escuchar las peticiones para diferentes direcciones IP. Con ello, se puede centralizar el servicio de diferentes dominios web con direcciones IP diferentes, en un único servidor de web (hosting de varios dominios). En Linux esta característica se puede implementar compilando el núcleo con soporte para ?alias IP?, lo que permite definir en una única tarjeta de red, varias direcciones IP. De este modo, nuestra tarjeta de red aceptará paquetes con diferentes direcciones IP de destino, peticiones que se gestionarán de forma diferente por el servidor de web, según a que dirección IP o nombre DNS, vayan dirigidas. Cada servidor virtual tiene su propia estructura de direcciones, y su configuración.

Documentación

Sin lugar a dudas es la documentación de Apache lo que más ha ganado en las últimas versiones de Apache. Se incluye junto con el fichero de la distribución, y tras la instalación, está dentro del directorio: /usr/local/apache/docs.

Proyectos asociados

Hay muchos proyectos asociados a Apache cuyo objetivo es aumentar su funcionalidad. Desde aquí vamos a destacar dos de ellos: uno muy útil para los desarrolladores, conocido como PHP, y otro orientado a la privacidad de las comunicaciones, Apache-SSL.

PHP

En la programación de aplicaciones en Internet, es importante las herramientas de programación que se utilizan tanto en el lado del cliente, como del servidor. El lector puede haber utilizado lenguajes como Javascript, JScript, o VBScript del lado del cliente, para aumentar la funcionalidad de las páginas HTML. De forma similar, hay lenguajes de programación del lado del servidor, que se introducen dentro de las páginas HTML.

En el servidor de web Apache, un lenguaje de programación que se puede embeber en las páginas HTML se conoce como PHP. Es un lenguaje similar a Perl, y muy sencillo de utilizar. Es un lenguaje en constante desarrollo, y cuya principal característica es que proporciona una librería de funciones que permite acceder a las principales bases de datos del mercado (Adabas, Ilustra de Informix, Oracle, MySQL, PostgreSQL entre muchas otras), lo que facilita mucho la integración del web con el mundo de las bases de datos.

PHP en la actualidad ha alcanzado la versión 4.0, y en cada nueva versión se aumenta su funcionalidad. Es un lenguaje muy potente en la programación del lado del servidor, y que sustituye de forma elegante a la programación de cgi-bin. Se recomienda al lector que consulte la referencia [6] y que profundice en este proyecto, ya que es muy atractivo para todo desarrollador de aplicaciones en Internet.

Apache-SSL

El comercio electrónico es el campo que va a arrastrar a Internet con más fuerza, y es uno de los sectores con mayores perspectivas de futuro. En la actualidad, el único freno al comercio electrónico en Internet es la seguridad, o la falta de ella.

Debido al gran interés que hay, se están desarrollando rápidamente estándares que aseguran la seguridad en Internet, en especial, dentro del web. Apache-SSL es el servidor Apache descrito en el presente artículo, pero con nuevas características de seguridad, como son la encriptación y la autenticación. Lo que más puede sorprender al lector, es que dichos parches son también de libre distribución, utilizándose una librería de encriptación (RSA, DES, MD5 ?) libre distribución.

El mundo de la encriptación está lleno de connotaciones militares, y depende mucho del país en el que estemos, de las utilidades de encriptación que podamos utilizar. En España no hay ningún problema en utilizar por ejemplo, la encriptación RSA de 128 bits, pero en el caso de utilizar dicha encriptación en comunicaciones internacionales, las cosas se complican mucho. El lector que quiera profundizar en esta versión de Apache con SSL (Secure Socket Layer), y en general en el mundo de la encriptación, puede

consultar la referencia [2] del artículo.

Conclusiones Apache

Apache es en la actualidad el principal servidor de web. Es el más rápido, eficiente y el que evoluciona a mayor velocidad. Y Apache, por su naturaleza de software abierto, es ideal para instalar en máquinas GNU/Linux, que aseguran un S.O. con unas comunicaciones excelentes. Apache y GNU/Linux es una combinación que se está utilizando en el mundo empresarial, Apache ha ayudado a que el campo de GNU/Linux se amplíe de forma muy sólida en el mundo Internet, creando una Internet-box que difícilmente puede ser superada por otra plataforma en los sistemas actuales, tanto en coste como en potencia.

Apache 2.0

Apache en la actualidad ya ocupa el 62% del mercado de servidores web, después de subir un 2% en Junio de 2000. A pesar de este predominio el grupo de desarrolladores de Apache no se está durmiendo en los laureles y en la nueva versión, la 2.0, destaca que el servidor Apache tendrá soporte para hebras de ejecución, lo que permitirá un modelo mixto de ejecución utilizando procesos y/o hebras según se configure el servidor. El rendimiento y escalabilidad de esta nueva arquitectura serán muy superiores a los que existen en la actualidad, además de mucho más flexibles de configurar.

Otro de los detalles importantes en esta nueva versión es el soporte para el sistema operativo BeOS, que se une a la saga de Unix, con GNU/Linux a la cabeza, y a la plataforma Microsoft Windows.

Apache 2.0 están aún en fase alpha, es decir, que aún es impensable utilizarlo para nada más que comenzar a conocerlo. Se han liberado ya 4 alphas de Apache 2.0 y vamos a probar la última versión a la que tenemos acceso a través del CVS aunque ya se nos advierte: no tiene ni que compilar. Esperemos que haya suerte y podamos jugar un poco con Apache 2.0.

Las mejoras en Apache 2.0 las podemos ver en el plan del proyecto en la referencia [7]. Vemos que en la versión 3.0, aún muy muy lejana, se tiene previsto dar soporte para HTTP-NG, algo que podría ser un nuevo hito para la era de los servidores web.

Las pruebas que vamos a realizar son con el fichero `apache-2.0_20000702041230.tar.gz` que como veremos a posteriori, es la versión alpha 5 de Apache 2.0.

Lo primero a destacar es el tamaño del "tar.gz". Más de 2 Mb de código fuente y documentación. Con cuidado vamos leyendo los ficheros "README" de los desarrolladores para no perder tiempo y siguiendo sus indicaciones nos vamos al directorio "src" y leemos "README.MPM". Bien, como

siempre, parece que con "configure" vamos a poder preparar los Makefiles y compilar sin muchos problemas. Vamos con ello.

- tar xvfz apache-2.0_20000702041230.tar.gz
- cd apache-2.0
- cd src
- ./buildconf
- ./configure
- make

El lector podrá observar que durante el make se utiliza para compilar con el gcc el flag -lpthread. Sí, esta usando Posix Threads para compilar. Era verdad lo el uso de hebras en esta nueva versión ;-D. También se ve muchos que se compilan muchos fuentes con "xml" como parte de sus nombres. XML se está convirtiendo en el estándar para el intercambio de datos entre aplicaciones y Apache parece que algo ha adoptado.

Bueno pues no hemos tenido ningún problema para compilar Apache 2.0. Sabíamos que estos chicos de Apache eran precavidos cuando decían que podría ni compilar. Seguimos con la investigación. Y lo primero será intentar arrancarlo. Para ello vamos a arriesgarnos a instalarlo bajo "/usr/local".

- su
- make install

Parece que hace uso de los directorios:

- /usr/local/conf: ficheros de configuración
- /usr/local/htdocs: directorio con los ficheros HTML del servidor
- /usr/local/icons: iconos del servidor Apache
- /usr/local/logs: logs de acceso y errores
- /usr/local/cgi-bin: directorio de cgi-bin del servidor Apache
- /usr/local/include/xml: el parser de XML de Apache
- /usr/local/include/apr: Apache Portable Runtime
- /usr/local/bin/httpd: el binario del servidor web

Vamos a probar a arrancar el servidor de web. En la máquina está corriendo el servidor 1.3.9 por lo que puede haber algún conflicto con los puertos.

```
/usr/local/bin/httpd
```

Parece que ha arrancado correctamente, con un par de avisos sobre la configuración, la cual no ha sido tocada para nada. Ha sido sencillo llegar a tener corriendo Apache 2.0. Vamos a seguir profundizando. Y para ello lo mejor que podemos hacer es irnos al directorio de configuración para ver al menos en que puerto tenemos lanzado el servidor de web Apache 2.0 y poder acceder a él. Mirando con más detenimiento los ficheros de logs, /usr/local/logs, en el fichero error_log se nos informa de que no se ha podido arrancar de forma correcta porque el puerto ya estaba en uso. Vamos a cambiarlo en el fichero de configuración.

```
cd /usr/local/conf/
```

En este directorio nos encontramos 4 ficheros:

- httpd.conf: viejo conocido de las versiones 1.x, y explicado en este mismo artículo, nos encontramos con las primeras diferencias en la sección del número de servidores que se arrancan por defecto, el número máximo ... y sí, las diferencias son debidas a que tenemos una sección para configurar por hebras en vez de por procesos.

```
## Server-Pool Size Regulation (MPM specific)
```

Aquí comenzamos a ver hablar de MPM, que es una máquina alrededor de Apache que se encarga de crear y gestionar los procesos, hebras y sockets. El MPM y el APR (Apache Portable Runtime) son las únicas parte de Apache que son dependientes de la arquitectura. Para más detalles de MPM y como se integra dentro de la arquitectura de Apache 2.0 está el fichero "mpm-design.txt" dentro del directorio "doc" donde se encuentran las discusiones entre desarrolladores para dilucidar el papel de MPM. Podemos seleccionar entre distintos MPM, que son módulos de Apache, según las necesidades de nuestro servicio. Como vemos, la flexibilidad que tenemos es mucho más grande que en Apache 1.x. Todo lo demás en él es exactamente igual que en las versiones 1.x aunque incluye el contenido de srm.conf y el de access.conf de la versión 1.x.

- highperformance.conf: los chicos de Apache han preparado este ejemplo de configuración para lograr un servidor de web con un gran rendimiento. La configuración no tiene nada especial y el supuesto alto rendimiento viene de las opciones:

```
MaxClients 8
StartServers 5
MinSpareServers 5
MaxSpareServers 10
# Assume no memory leaks at all
MaxRequestsPerChild 0
```

No se utiliza el módulo "mpmt_thread.c" de Posix Threads.

- mime.types: tipos MIME que reconoce el servidor de web.
- magic: datos para el módulo mod_mime_magic para detectar de forma automática a que tipo MIME pertenece un fichero, sin tener en cuenta su extensión (comando file de GNU/Linux).

Tras modificar el puerto al 8080 y cambiar el grupo bajo el que se ejecuta el servidor de web, en nuestro caso se ha puesto webs, se ha arrancado de nuevo el servidor y esta vez no ha habido ningún problema. Con "tail /usr/local/logs/error_log" podemos ver el mensaje:

```
[Sun Jul 02 11:45:22 2000] [notice] Apache/2.0a5-dev (Unix) configured -  
resuming normal operations
```

Bueno pues ya tenemos el Apache 2.0 alpha 5 funcionando. Vamos a acceder a la documentación del servidor a través de la URL <http://localhost:8080/manual/index.html>.

Bueno, quedarían por analizar muchos aspectos de esta nueva versión de Apache pero de momento vamos a pararnos aquí. Hemos logrado los objetivos principales: obtener Apache 2.0, compilarlo, instalarlo y configurarlo, así como ver como afecta la nueva arquitectura de procesos y hebras al funcionamiento de Apache.

Referencias

- [1]: Fundación Apache (<http://www.apache.org>)
- [2] <http://www.apache-ssl.org>
- [3] <http://www.netcraft.co.uk/Survey/>
- [4] <http://www.apacheweek.com/>
- [5] <http://www.develnet.es/apache/>, <http://www.cs.us.es/archive/apache/>,
<http://ftp.rediris.es/ftp/mirror/apache/>, http://slug.ctv.es/mirror/apache_httpd/,
<http://www.arrakis.es/pub/apache/>
- [6] <http://php.iquest.net>
- [7] <http://dev.apache.org/project-plan.html>

Appendix A.

