CÓMO AUMENTAR LAS CONEXIONES SIMULTÁNEAS EN APACHE

El límite de conexiones simultáneas.

El límite de conexiones simultáneas siempre será el mismo (150) a menos que tomemos las medidas adecuadas, lo que significa que, si queremos tener muchas personas conectadas al mismo tiempo, no solo requeriremos un buen hardware, sino que también una buena configuración.

Lo primero que habría que pensar es: ¿Qué capacidad tiene mi equipo? ¿Cuántas conexiones simultáneas puede llegar a soportar mi equipo si lo llego a forzar el máximo posible? Todo esto depende de un único factor; la memoria RAM (Random Access Memory).

A mayor la memoria RAM, mayor el número de conexiones, si bien no existe un valor fijo (es decir X clientes por cada X RAM), es por ello que antes de nada es importante hacer unos pequeños cálculos en nuestro servidor web, con el fin de conocer nuestros límites.

Lo primero que habría que saber es cuanta memoria RAM de media consume cada conexión a Apache, ya que cada conexión establecida supone un cierto consumo de RAM en el sistema… Obviamente no todas las conexiones consumen la misma ram, con lo que habría que hacer una media… Todo ello se puede obtener con el siguiente comando:

```
ps -ylC apache2 --sort:rss | awk '{SUM += $8; I += 1} END {print SUM/I/1024}'
```

El resultado obtenido estaría representado en megabytes y puede variar dependiendo del número de conexiones activas, el tipo de páginas a las que se accede, etc… Por ello es recomendable realizar la prueba con diferentes pestañas abiertas; cada una de ellas mostrando distintos contenidos a ser posible. En mi caso por ejemplo el resultado ha sido de 9.5458 lo que si lo redondeamos al superior serían 10 MB de RAM consumidos de media por conexión.

Además, es importante saber cuánta memoria RAM es consumida por el resto de procesos que hay activos en el sistema, ya que el servicio web no es el único que corre en el sistema operativo y es necesario dejar memoria RAM libre en el servidor para que pueda ejecutar el resto de tareas. Esto se puede obtener con el comando mostrado a continuación:

```
ps -N -ylC apache2 --sort:rss | awk '{SUM += $8} END {print SUM/1024}'
```

El resultado obtenido también sería representado en megabytes, y nos mostraría con bastante precisión la cantidad de RAM consumida por el resto de procesos; en mi caso 800 MB. Con esta información podríamos hacer un cálculo general de la cantidad de conexiones simultáneas que podríamos tener; calculo que obtendríamos mediante una operación bien sencilla.

(RAMTOTAL – RAM_RESTOPROCESOS) / RAM_POR_CONEXIÓN

Con esta fórmula en mano, imaginemos que tenemos un equipo con 4 GB RAM, es decir 4096 MB y que nuestro equipo ha mostrado los resultados antes mencionados; el cálculo sería:

(4096 – 800) /10= 329 conexiones simultáneas

El problema con este cálculo es que uno demasiado extremista, ya que nos consumiría toda la RAM (haciendo que el servidor consuma swap) y además, en caso de tener una base datos, como MySQL o cualquier otra, las conexiones a ésta también consumirían RAM, con lo que el número obtenido se podría calificar como un número utópico. Por ello, para dejar libre la memoria para posibles procesos adicionales y además contemplar la posibilidad de que se ejecuten conexiones a alguna base de datos, reduciríamos el número de conexiones a 250.

Ahora que tenemos nuestro número de conexiones simultáneas máximas, habría que preparar Apache para poder recibir dicho número, lo cual se realiza en el fichero de configuración de este llamado apache2.conf, el cual está alojado en /etc/apache2.

Con PHP normal

En Ubuntu/Debian el módulo se encuentra en:

sudo nano /etc/apache2/mods-available/mpm_prefork.conf

El fichero en cuestión sigue una estructura basada en módulos, cada uno con su nombre correspondiente, pero solamente nos interesaría uno de ellos, cuyo nombre es mpm_prefork_module. El módulo en cuestión posee los siguientes datos por defecto:

Este módulo posee una serie de parámetros muy importantes, aunque hay uno de ellos que nos interesaría especialmente, llamado MaxClients (MaxRequestWorkers). Dicho parámetro especifica el número máximo de conexiones simultáneas y habría que modificarlo a 250.

Un detalle a tener en cuenta es que cuando se especifica un valor distinto al de por defecto en dicho parámetro, es necesario añadir otro más justo ANTES de éste. Dicho parámetro se denomina ServerLimit y establece el límite de conexiones que podría “aguantar” el servidor aun cuando se encuentra fuera del límite.

El parámetro ServerLimit siempre tiene que ser ligeramente superior al MaxClients y aquí al tener poco margen de maniobra habría que establecer un límite de 270. Esto haría que el módulo quedase de la siguiente forma:

```
# Antes: Apache 2.2

<IfModule mpm_prefork_module>
  StartServers       5
  MinSpareServers    5
  MaxSpareServers    10
  ServerLimit        270
  MaxClients         250
  MaxRequestsPerChild 0
</IfModule>

# Ahora: Apache 2.4
<IfModule mpm_prefork_module>
 StartServers          5
 MinSpareServers       5
 MaxSpareServers       10
```

```
 ServerLimit          600
 MaxRequestWorkers    500
 MaxConnectionsPerChild  0
</IfModule>
```

Nota: Con el mpm_prefork, use esta directiva ServerLimit solo si necesita establecer MaxRequestWorkers un valor superior a 256 (predeterminado). No establezca el valor de esta directiva más alto de lo que desea establecer MaxRequestWorkers.

Importante: Hay un límite ServerLimit 20000 estricto de compilación en el servidor. Esto está destinado a evitar los efectos desagradables causados por errores tipográficos. Para aumentarlo aún más allá de este límite, deberá modificar el valor de MAX_SERVER_LIMIT en el archivo fuente mpm y reconstruir el servidor.

En Ubuntu/Debian, activar el módulo mpm_prefork (normalmente ya está activado por defecto):

sudo a2enmod mpm_prefork

Con PHP-FPM

Si en lugar de mpm_prefork estás usando mpm_event con PHP-FPM, la configuración del módulo está en:

sudo nano /etc/apache2/mods-available/mpm_event.conf

Y la configuración agrega unos parámetros mas:

```
<IfModule mpm_prefork_event>
 StartServers          5
 MinSpareThreads       25
 MaxSpareThreads       75
 ThreadLimit           64
 ThreadsPerChild       25
 MaxRequestWorkers     250
 MaxConnectionsPerChild  0
</IfModule>
```

Ahora únicamente faltaría reiniciar el servicio Apache mediante el comando:

sudo /etc/init.d/apache2 restart

Con esto ya podríamos disfrutar de nuestro servidor web optimizado.

Información extra:

MaxClients

La directiva MaxClients establece el número máximo de conexiones simultáneas que pueden ser soportadas por el servidor. No debe configurarse con un valor demasiado bajo para que las nuevas conexiones no deban ponerse en cola, lo que provocaría un timeout en las conexiones y los recursos del servidor se quedarían sin usar. Si por el contrario establecemos este valor demasiado alto, el servidor podría saturarse y el tiempo de respuesta se degradará drásticamente. El valor adecuado para la directiva MaxClients se puede calcular así:

MaxClients = RAM dedicada a Apache / Profundidad máxima de los procesos hijo (el tamaño de los procesos hijo para servir archivos estáticos está sobre los 2-3 MB. Para servir archivos dinámicos está sobre los 15MB).

Si hay más usuarios simultáneos que los configurados en MaxClients, las peticiones se pondrán en cola hasta el número especificado en la directiva ListenBackLog. Debes aumentar la directiva ServerLimit si quieres aumentar la directiva MaxClients por encima de 256.

MinSpareServers, MaxSpareServers y StartServers

Las directivas MaxSpareServers y MinSpareServers determinan cuántos procesos hijo se mantendrán a la espera de las solicitudes. Si MinSpareServers tiene un valor demasiado bajo y llegan muchas peticiones, Apache tendrá que generar procesos adicionales secundarios para atender esas peticiones. La creación de procesos hijo debe evitarse en lo posible. Si el servidor está ocupado creando procesos hijo, no podrá atender las solicitudes de los clientes de forma inmediata. La directiva MaxSpareServers no debe ser demasiado alta, ya que podría causar problemas de recursos porque los procesos hijo los consumen.

Configura MaxSpareServers y MinSpareServers para que Apache no necesite crear frecuentemente más de 4 procesos hijo por segundo (Apache puede crear un máximo de 32 procesos hijo por segundo). Cuando se creen más de 4 hijos por segundo se generará un registro en el error log.

La directiva StartServers establece el número de procesos hijo creados al inicio. Apache continuará creando procesos hijo hasta que alcance el valor estipulado en la directiva MinSpareServers. Esto no tiene mucha incidencia en el rendimiento del servidor si éste no se reinicia con frecuencia. Si hay muchas peticiones y Apache se reinicia con frecuencia, ajusta esta directiva a un valor relativamente alto.

MaxRequestsPerChild

La directiva MaxRequestsPerChild establece el número máximo de solicitudes que un proceso hijo puede manejar. Después de alcanzar ese valor, el proceso hijo morirá. Este valor es «0» por defecto, lo que significa que el proceso hijo nunca caducará. Es conveniente otorgarle a esta directiva un valor de unos miles para ayudar a prevenir un uso excesivo de la memoria del servidor, ya que después de cumplir con cierto número de peticiones el proceso hijo morirá. No establezcas un valor demasiado bajo para que las solicitudes que debe atender el proceso hijo nunca estén por encima.

KeepAlive y KeepAliveTimeout

La directiva KeepAlive permite que se reciban múltiples solicitudes a través de la misma conexión TCP. Esto será útil sobretodo cuando sirva páginas con gran cantidad de imágenes. Si KeepAlive estuviese desactivado, el servidor realizaría una conexión diferente para cada imagen.

KeepAliveTimeout establece cuánto tiempo tendrá que esperar hasta recibir la próxima petición. Ajústalo a un valor bajo, entre 2 y 5 segundos. Si el valor es demasiado alto, los procesos hijo estarán a la espera sin ser productivos hasta que se agote el tiempo para poder seguir sirviendo a nuevos clientes.