APACHE MPM PREFORK

MPM Prefork

El Módulo de Multi-Proceso (MPM) Prefork implementa un servidor web sin hilos, con procesos precreados.

Fork (en inglés) se refiere a la creación, por parte de un proceso “padre” de una copia de sí mismo, que pasa a denominarse proceso “hijo”.

En este sentido, Prefork hace referencia a que los procesos hijos son creados previamente. Así pues, se convierten en procesos servidores disponibles para atender a las peticiones entrantes de conexión de los clientes del sitio web.

Un único proceso padre se encarga de gestionar el tamaño del pool, según las directivas específicas al respecto.

Pool (en inglés) hace referencia a un conjunto de recursos previamente creados y listos para su uso. El objeto del pool es disponer de dichos recursos con antelación, sin la necesidad de crearlos durante la crisis de un momento de necesidad.

La ventaja de este módulo MPM radica fundamentalmente en la no utilización de hilos, con lo cual resulta compatible con otros módulos y aplicaciones que no son compatibles con los programas multihilos. Por ejemplo, el módulo PHP no es compatible con el uso de hilos.

Otra ventaja importante es que los procesos son independientes entre sí, y un problema en la atención de una petición no afectaría al resto de procesos.

MaxRequestWorkers

El parámetro más importante que debe configurarse en este módulo es el definido por la directiva MaxRequestWorkers. Este valor delimita el número máximo de peticiones de conexión que pueden servirse de forma simultánea. Tiene un valor por defecto de 256.

Cualquier petición extra que se formule será puesta en cola, según la directiva ListenBacklog, cuyo valor delimita la longitud máxima de la cola de peticiones pendientes, que es 511 por defecto. Una vez que algún proceso hijo vuelva a estar disponible, se podrán atender entonces las peticiones de conexión por orden de cola.

El cálculo óptimo para MaxRequestWorkers es muy sencillo: en principio, debemos determinar cuánta memoria RAM queremos dedicar al servicio web.

Si nuestra máquina se dedica exclusivamente a proveer el servicio web, entonces podemos dedicar la mayor parte de la memoria para el servidor, y reservar una pequeña parte para el sistema operativo y cualquier otra necesidad que pueda surgir.

Dedicar un 50% del total es demasiado prudente. En cualquier caso, dedicar más del 80% podría ser excesivamente arriesgado. En general, depende del resto de tareas que realice nuestra máquina, así como de la carga del núcleo.

Una vez determinada la cantidad de memoria disponible para el servidor, la dividimos por el tamaño medio que van a tener nuestros procesos servidores. Este valor suele ser de unos pocos megabytes, en el caso de un servidor limpio, sin módulos añadidos. Sin embargo, con el módulo PHP instalado, los procesos podrían llegar a ocupar varias decenas de megabytes.

Lo mejor es determinar dicho tamaño con la ayuda de una herramienta de monitorización, como puede ser top, o free. Así, por ejemplo, una vez arrancado el servidor apache, iniciamos top en una terminal y anotamos el tamaño medio de cada proceso en memoria RSS.

La memoria RSS (Resident Set Size) hace referencia a la cantidad de páginas de memoria física (RAM) que está siendo efectivamente ocupada por un proceso. Sin embargo, aunque dicha ocupación es efectiva, muchas veces se trata de memoria compartida con otros procesos similares. De este modo, la memoria total efectivamente utilizada por el pool de procesos va a ser mucho menor que la suma de los valores individuales. Generalmente, menos de la mitad.

Otro método alternativo es el de controlar con el comando free la cantidad de memoria disponible antes y después de arrancar el servidor. Si dividimos la memoria utilizada por el número de procesos hijos obtendremos el tamaño medio de cada uno de ellos.

El objetivo final es que el número máximo de procesos hijos (servidores) creados por el proceso padre no supere nunca la cantidad de memoria RAM disponible.

Si, por el contrario, una configuración errónea condujera a la creación de un número mayor de procesos, el daño sería muy grave. Dada la falta de memoria disponible, el sistema operativo iniciaría el proceso de paginación (swapping). En un servidor web es absolutamente inaceptable la paginación (swapping) de la memoria principal. El efecto de paginar la memoria provocaría una latencia que resultaría totalmente inaceptable para el usuario normal que intentara conectarse a nuestro servicio web.

Por este motivo, es también aconsejable tomar medidas que impidan la paginación de la memoria. En la carpeta virtual de procesos, tenemos el archivo /proc/sys/vm/swappiness. El valor (de 0 a 100) que contiene dicho archivo representa el grado de afinidad del núcleo para la paginación de la memoria. Su valor por defecto es 60. Si le asignamos un valor unidad, el núcleo evitará la paginación en la medida de lo posible, sin llegar a deshabilitar por completo dicha función. Para ello, basta con ejecutar cualquiera de las siguientes líneas de comando:

```
# echo 1 1>/proc/sys/vm/swappiness
# /sbin/sysctl -w vm.swappiness=1
```

Es muy importante tener en cuenta que, si quisiéramos aumentar el valor por defecto de 256 de esta directiva, deberíamos también modificar el valor de la directiva ServerLimit (con el mismo valor que queramos asignarle a MaxRequestWorkers).

Modo de funcionamiento

Un único proceso padre (root) controla la creación del resto de procesos hijos (apache). Dichos procesos se encargan de servir y atender las peticiones de conexión de los clientes a medida que van llegando.

Dada la dificultad y consumo de recursos del sistema que supone la creación (fork) de procesos hijos, es conveniente mantener una reserva (pool) de procesos pre-creados (prefork) que estarán disponibles (idle/spare) esperando la llegada de peticiones de conexión. De este modo, cuando llega una petición de conexión, el cliente no tendrá que esperar a la creación (fork) de un proceso servidor (worker/server) para ser atendido.

La directiva StartServers (5 por defecto) controla el número de servidores (procesos hijos) que se crean al arrancar el servidor Apache.

La directiva MaxRequestWorkers (256 por defecto) controla el número de servidores que pueden atender peticiones de conexión de forma simultánea. Se ha explicado ampliamente en el apartado anterior.

La directiva ServerLimit (256 por defecto) controla el número máximo que puede alcanzar la directiva MaxRequestWorkers. En nuestro MPM Prefork, deberían tener ambas directivas el mismo valor. Una vez fijado su valor, al iniciar el servidor Apache se reserva una cantidad proporcional de memoria compartida. Por este motivo, debe estar justificado aumentar el valor por defecto. Asimismo, sería un desperdicio de memoria que ambas directivas no tuvieran el mismo valor.

La directiva MinSpareServers (5 por defecto) controla el número mínimo de procesos hijos inactivos que estarán disponibles para la atención de peticiones de conexión.

El proceso de creación de procesos hijos es el siguiente: en un principio se crean el número de procesos especificado en la directiva StartServers. Un segundo después se crea un nuevo hijo, otro segundo después se crean dos nuevos hijos, otro segundo después se crean cuatro nuevos hijos, y así sucesivamente, hasta que el ritmo de creación de hijos es de treinta y dos por segundo. A partir de ahí, se siguen creando nuevos hijos al mismo ritmo de treinta y dos cada segundo, hasta alcanzar el valor fijado por MinSpareServers. Evidentemente, el proceso se detiene cuando se alcanza dicho valor.

Aparte de los procesos hijos inactivos creados a propósito para mantener un pool de reserva, otros procesos quedarán igualmente inactivos cuando finalicen las conexiones que los mantenían ocupados. Por este motivo, es necesario disponer de otra directiva que controle el número total de procesos inactivos en un momento dado.

La directiva MaxSpareServers (10 por defecto) controla el número máximo de procesos hijos inactivos que estarán disponibles para la atención de peticiones de conexión. Si en un momento dado hay un número mayor de procesos inactivos, el proceso padre se encargará de matar los procesos excedentes.

El motivo de que el proceso padre tenga privilegios de root es para poder asociar a sus propios procesos hijos al puerto 80 (privilegiado). Por el contrario, los procesos hijos se ejecutan por un usuario sin privilegios (apache). Estos valores pueden modificarse mediante las directivas User y Group en el fichero de configuración principal /etc/httpd/conf/httpd.conf. Evidentemente, los procesos hijos que proveen el servicio web deben ser capaces de leer el contenido que se sirve, aunque es conveniente al mismo tiempo reducir sus privilegios al mínimo.

Una última directiva MaxConnectionsPerChild controla el número máximo de conexiones que un proceso hijo (servidor) puede llegar a atender a lo largo de su vida útil. Una vez superado este límite, el proceso padre se encarga de eliminarlo. La función de esta directiva es la de evitar que un proceso que haya sufrido algún tipo de problema de conexión o de memoria pueda seguir viviendo para siempre y perjudicar así el rendimiento del sistema.

A un nivel más avanzado, existe un problema conocido como la “manada vociferante” (thundering herd) que merece una atención especial.

Supongamos que tenemos un pool de procesos inactivos durmientes (sleeping), esperando la llegada de una petición de conexión. Cuando llega dicha petición, se les despierta a todos a la vez, aunque solamente uno de ellos podrá atender la petición, yendo el resto de nuevo a dormir en un estado inactivo. La misma historia va a repetirse cada vez que llegue una nueva petición de conexión, provocando un evidente desperdicio de recursos, dado el consumo que supone despertar a tantos procesos disponibles al mismo tiempo.

El MPM Prefork utiliza la directiva Mutex con el mecanismo por defecto de mpm-accept para serializar el acceso a las peticiones de conexión entrantes y evitar así el problema de la “manada vociferante”.