Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Cache de código PHP compilado en memoria: APC (20879 lectures)

Per **Ricardo Galli Granada**, ***gallir*** (http://mnm.uib.es/gallir/)

Creado el 06/05/2001 20:26 modificado el 06/05/2001 20:26

Los programadores de PHP saben que es muy rápido, pero que en servidores que estén muy cargados (centenar de conexiones por segundo), el hecho que se tenga que recompilar cada vez hace que el ordenador se sobrecargue. El APC es un sistema de cache de código compilado que evita recompilar el código PHP por cada conexión permitiendo muchos mejores tiempos de respuesta (hasta un 400%) y cargas muy superiores. En los últimos días, cuando instalé dicho sistema en Bulma, le encontré algunos bugs y bastantes ineficiencias, por lo que me puse a modificarlo y probarlo en Bulma. Como ha mejorado mucho con respecto al original ya envié *patches* a los autores originales y he abierto mi propia rama de desarrollo[1] para seguir mejorando el código. Os invito a probarlo y ayudar en el desarrollo.

[Actualizado]: *benchmarks* a Bulma

Actualización

Hice unos *benchmarks* a bulma.net probando la página índice con[2] y sin[3] el sistema de cache APC activado. También probé el acceso a un artículo individual con[4] y sin[5] el sistema de cache.

Para las pruebas usé el ApacheBenchmark, con 10 conexiones simultáneas y durante 10 segundos.

Esta implementación tiene una técnica que implementé para permitir el acceso de varios lectores pero un sólo escritor a las regiones críticas de la memoria compartida.

La conclusión rápida es que el **PostgreSQL (7.1) es muy, pero que muy malo, malísimo para las consultas simultáneas**. Aunque también debo decir que podríamos mejorar un poco las consultas SQL.

Aquí teneís una tabla con los resultados comparativos, incluí una script PHP muy sencillo para ver la mejora de las técnicas nuevas de semáforos. Los resultados de la fila "Postgres" son los accesos a artículos individuales de Bulma. Los números para la página índice son aún peores...

```
                CONCLUSIONS
              Requests per Seconds
             | New Semaphores  |  Flock   |  Diff
-------------------------------------------------
Simple PHP:  | 254.35          |  226.25  | 12.4%
Postgres:    |  14.85          |   13.94  | 6.52%
-------------------------------------------------
             | New semaphores  | No cache |  Diff
-------------------------------------------------
Simple PHP:  | 254.35          |  218.59  | 16.4%
Postgres:    |  13.94          |   11.36  | 22.7%
```

Introducción

La empresa Zend[6], que ha desarrollado bastantes módulos del PHP4, también ofrece un producto comercial [7]para hacer cache de páginas compiladas, pero es un producto cerrado y muy caro.

La gente de Community Connect[8] se ha dedicado a desarrollar un producto alternativo llamado Alternative PHP Cache[9]. Este sistema es e código abierto y con la misma licencia del PHP4 (Q Public License)[10]. Trabaja con dos métodos distintos: memoria compartida y *mmap* de ficheros. Cada una tiene sus ventajas y desventajas, pero a mi me gusta más el sistema con memoria compartida.

Como le he encontrado algunos bugs y problemas de ineficiencia, como además la necesidad de agregar algunas características , he creado mi **propia versión**[1] con esos problemas solucionados y estoy enviando parches[11] a los desarrolladores principales. Al momento de escribir este artículos, parece que iban a integrar todos en la versión "oficial".

Por supuesto, estáis invitados a probarlo y ayudarme en el desarrollo.

El Software y Configuración

El sistema es muy sencillo de instalar y configurar:

1. Poner el ejecutable (php_apc.so) en el directorio de extensiones del PHP4.
2. Indicar en el php.ini que se cargue dicho módulo:
3. Indicar las parámetros necesarios o deseados en el mismo fichero.
4. Rearrancar el Apache.

En el caso de Bulma, la configuración es la siguiente:

```
zend_extension="/usr/local/lib/php/extensions/php_apc.so"
apc.relative_includes=1
apc.check_mtime=1
apc.shm_segment_size=2097152 ; 2 MB
apc.idle=900 ; removed from cache if idle for 15 minutes
apc.hash_buckets=256
```

En mi página del APC[1], podéis bajar el ejecutable en forma de de librería compartida para cargar como módulo del PHP, también podéis bajar los fuentes, ver las modificaciones en el repositorio del CVS y las configuraciones en el *php.ini* que usamos para Bulma.

Y para ver como se comporta, también hay un pequeño *script*[12] que os muestra el estado de los objetos en el cache del servidor.

Modificaciones Realizadas

Cuando lo empecé a probar en Bulma, noté que tenia algunos problemas y que le vendrían bien algunas características adicionales:

- Descargar del cache los objetos que no han sido accedidos durante un periodo determinado de tiempo: ***idle time*** (IMPLEMENTADO). Esto ahorraría memoria que se gasta en tener en cache páginas que prácticamente no se acceden y que nunca se descargan de memoria. Se podía solucionar con el *ttl*, pero también afecta a aquellas páginas que son accedidas frecuentemente, obligando a recompilarlas cada vez.
- Optimizar el control de fechas de modificación de los ficheros PHP originales: ***mtime_ttl*** (IMPLEMENTADO). He notado, como también lo dice el FAQ que esto afecta al rendimiento, que además de ser cierto, por una ineficiencia de la generación de las claves para el *hashing*, también se sufría la ineficiencia aunque no es esté controlando la fecha. Para ello he modificado las funciones de verificación de modificación para que lo haga cada vez que haya pasado un tiempo configurable en segundos, haciendo que ahora no se note en absoluto en el rendimiento.
- Optimización de le función de generación de claves (IMPLEMENTADO). La función que genera las claves de los ficheros PHP era muy ineficiente debido a que hacía un *stat* para verificar la existencia del fichero en cada uno de los directorios de *includes* del PHP. He cambiado radicalmente dicha función y ahora ni siquiera hace uso de la llamada de sistema *stat*.
- Un bug del Linux hace que se dejase segmentos de memoria compartida sin utilizar (IMPLEMENTADO).
- Implementación de un algoritmo de CLOCK para el reemplazo de objetos, similar a la técnicas usadas en los sistemas operativos como aproximación al algoritmo LRU (en DESARROLLO).

Lista de enlaces de este artículo:

1. http://m3d.uib.es/~gallir/ext/APC/
2. http://m3d.uib.es/~gallir/ext/APC/benchs/index_cache.html
3. http://m3d.uib.es/~gallir/ext/APC/benchs/index_nocache.html
4. http://m3d.uib.es/~gallir/ext/APC/benchs/body550_cache.html
5. http://m3d.uib.es/~gallir/ext/APC/benchs/body550_nocache.html
6. http://www.zend.com/
7. http://www.zend.com/store/products/zend-cache.php
8. http://apc.communityconnect.com/about.html
9. http://apc.communityconnect.com/
10. http://m3d.uib.es/~gallir/ext/APC/apc/LICENSE
11. http://lists.communityconnect.com/pipermail/apc-cache/2001-May/author.html#start
12. http://m3d.uib.es/~gallir/ext/APC/apcinfo.php

E-mail del autor: gallir _ARROBA_ uib.es