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

Optimizaciones al MySQL (y PHP) de Bulma (16559 lectures)

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

Creado el 13/07/2001 03:25 modificado el 13/07/2001 03:25

Actualización: Hemos llegado al 100% de mejora. Daneel se dio cuenta que el parser de RDF iba muy lento. Hicimos cambios radicales. Estamos en 23-24 páginas índices por segundo.

Esta semana fue muy movidita en Bulma, pero además de migrar a MySQL hemos conseguido optimizar bastante los accesos. La migración de MySQL nos dio un 30% de aumento de rendimiento (menos de lo que uno pueda pensar a priori), y las con las optimizaciones (código y base de datos) hemos podido llegar a un 60% de mejora sobre la última versión de Bulma sobre Postgres.

En los últimos *ApacheBenchmarks* con Postgres nos daba una 10 páginas índice por segundo con 100 conexiones simultáneas. Ahora estamos **en unas 16 páginas servidas por segundo** (y 100 conexiones simultáneas). Con respecto a los artículos, hemos pasado de **11 a casi 17** por segundo.

Lo primero que hemos hecho fue reducir al mínimo posible la tabla principal de artículos, para ellos hemos movido el cuerpo del texto a una tabla distinta. El objetivo era reducir los *seek* de las cabezas del disco para mostrar las páginas índices donde no interesa el cuerpo del texto. Esto nos da una ventaja adicional: ahora podremos tener artículos divididos en páginas distintas para poder mostrarlas secuencialmente, como hacen la mayoría de los sitios donde se escriben artículos más o menos largos.

Esto demuestra empíricamente algo que siempre afirmo (y me quieren matar ;-) y lo dicen los manuales de MySQL: no basta con normalizar una tabla (muchas veces su efecto es contrario en temas de rendimiento), sino que hay que ajustar la estructura para que la mayoría de los *selects* que se hagan sea sobre la mínima cantidad de datos posibles. El tiempo de retorno dela pagina principal de Bulma, sólo con este cambio, ha pasado **de 0.11 a 0.08-0.082** segundos. Casi un 30% de mejora en tiempo.

La tabla de artículos queda entonces dividida en dos (se muestran con los tipos de campos ya cambiados):

```
mysql> describe bul_tbl_noticias;
+---------------------------+----------------------+------+-----+---------+
| Field                     | Type                 | Null | Key | Default |
+---------------------------+----------------------+------+-----+---------+
| id_noticia                | mediumint(8) unsigned |      | PRI | NULL    |
| fecha_alta_noticia        | datetime             | YES  |     | NULL    |
| fecha_caducidad_noticia   | datetime             | YES  |     | NULL    |
| fecha_modificacion_noticia | timestamp(14)       | YES  | MUL | NULL    |
| formato_noticia           | tinyint(3) unsigned  | YES  |     | NULL    |
| id_tipo_noticia           | tinyint(3) unsigned  |      | MUL | 0       |
| id_autor                  | smallint(5) unsigned | YES  |     | NULL    |
| contador_comentarios      | smallint(5) unsigned |      |     | 0       |
| contador_lecturas         | int(10) unsigned     |      |     | 0       |
| titulo_noticia            | varchar(150)         | YES  |     | NULL    |
| keywords_noticia          | varchar(255)         | YES  |     | NULL    |
| resumen_noticia           | text                 | YES  |     | NULL    |
+---------------------------+----------------------+------+-----+---------+

mysql> describe bul_tbl_cuerpo_noticias;
+----------------+----------------------+------+-----+---------+
| Field          | Type                 | Null | Key | Default |
+----------------+----------------------+------+-----+---------+
| id_noticia     | mediumint(8) unsigned |      | PRI | 0       |
```

```
| numero_pagina  | tinyint(3) unsigned   |      | PRI | 1       |
| cuerpo_noticia | mediumtext            |      |     |         |
+----------------+-----------------------+------+-----+---------+
```

Una vez hecha esta división de la tabla de noticias, hemos optimizado los índices para que se ajusten lo máximo posible a los *selects* que se hacen en los *scripts*.

Todavía había cosas para optimizar, los tipos de los campos de la base de datos. Básicamente hemos cambiado los text (hasta 64 KB) a varchar para los casos donde la longitud es siempre pequeña: título, palabras claves, etc. también hemos cambiado los int por tinyint o smallint (mostrados en las tablas anteriores). Veréis que ya no usamos text, sino mediumtext que nos permite artículos con tamaño suficiente (2^24 =~ 16.000.000 de caracteres).

Por último, y para optimizar el ordenamiento por fecha de los artículos (lo usamos para las páginas índices y la columna de la derecha), cambié el campo del tipo *datetime* (8 bytes) a *timestamp* (4 bytes). Como siempre trabajamos con los *epoch*, no hay problemas. Bueno, sí, sufriremos el efecto del 2.37KY sino hacemos una ALTER TABLE antes de ese año. espero que alguien se acuerde porque yo ya no seré el BOFH de Bulma ;-).

ACTUALIZACIÓN: La optimización de las noticias de Barrapunto

Eduard (Daneel) se dio cuenta que quitando el parser del rdf para mostrar las noticias de Barrapunto hacía que se gane más de un 100% en total sobre nuestra versión anterior en Postgres (el PG no tenía nada que ver).

Además de re-estructurar el código, metí toda la lógica de bajar por HTTP el XML de las noticias y generar un fichero cache en los scripts php. Ya no hace falta tener un programa externo ni interpretar el XML para cada conexión.

Con estos cambios ahora llegamos a **23-24 páginas por segundo** para la página índice alrededor de 0.058 a 0.06 segundos de tiempos de retorno.

Resumen: no evitaremos un nuevo efecto Slashdot, pero fue divertido, aprendimos un montón y funciona mucho mejor.

E-mail del autor: gallir _ARROBA_ uib.es
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=726