

Analizando el tráfico web con Apache y MySQL

Este artículo describe el uso del módulo para Apache llamado `mod_log_sql`, que permite registrar los accesos a un servidor web en una base de datos MySQL

Por MySQL hispano

Cuando se tiene un sitio Web de cualquier tamaño, es importante saber que tipo de visitantes son los que se están atrayendo. La solución tradicional para monitorear el tráfico Web es usar una herramienta de análisis del archivo log tal como analog (www.analog.cx). analog es muy rápido, ¿pero qué sucede si se quieren obtener estadísticas en tiempo real o casi en tiempo real?. Una solución podría ser ejecutar analog desde un cron cada cinco minutos, pero qué pasa si se desean consultas más ad-hoc de nuestros registros de accesos para responder preguntas muy específicas como, "¿Cuál es el promedio de páginas que cada usuario de Internet Explorer visita?"

Las cosas comienzan a ponerse difícil cuando se tratan de configurar muchas de las herramientas de reporte de los registros Web. Hay muchas preguntas interesantes que se querrán realizar acerca del tráfico Web: "¿Desde dónde llegan los usuarios? ¿Cómo encontraron el sitio? ¿Los usuarios entraron a través de la página principal, o algún buscador los envió a una página específica? ¿Qué tipo de navegador están utilizando? ¿Cuáles son los URL's 404 más comunes?"

Desgraciadamente, muchas herramientas no pueden responder este tipo de respuestas de una manera sencilla. Para responder estas preguntas típicamente se necesita extender una herramienta existente, escribir nuestra propia herramienta o gastar algún tiempo valioso con *awk*, *grep*, y *wc*. La otra alternativa, cambiar a LAMP. Con la combinación Linux, Apache (y *mod_log_sql*), PHP y MySQL se puede construir un sistema de registros de acceso configurable y sin mucho esfuerzo. En este artículo vamos a ver como configurar las herramientas para que pueda funcionar el sistema de registros de accesos. En un artículo próximo se mostrará como construir una sencilla interfaz en PHP para revisar las estadísticas de acceso generadas por el sistema.

mod_log_sql

Si se lee la columna de la sección Perl Matters de Randal Schwartz de Septiembre del 2001 "Cleaning Out a Logging Database" (disponible en línea en http://www.linux-mag.com/2001-09/perl_01.html), se estará familiarizado con la idea de registrar todos los accesos de un sitio web utilizando *mod_perl* y MySQL. Los beneficios de registrar el tráfico en una base de datos MySQL son claros: no hay herramientas de reportes pequeñas, y puesto que MySQL es bastante rápido, se podrán obtener respuestas mucho más rápidas que "grepeando" a través de los archivos de registros de acceso.

Sin embargo, esta solución trabaja solamente si se está ejecutando *mod_perl*. Si se está mostrando contenido estático o contenido basado en PHP tenemos problemas, ¿correcto?. No, no del todo.

El *mod_log_sql* de Chris Powell (http://www.grubbybaby.com/mod_log_sql) es un módulo de Apache que almacena cada hit dentro de una base de datos MySQL. Mejor aún, el módulo no toma el lugar del tradicional archivo de registros de acceso - se pueden continuar generando los archivos *access_log* mientras se utiliza *mod_log_sql*.

Poniéndolo en uso

Como otro módulo de Apache, se puede compilar *mod_log_sql* dentro de Apache o construirlo como un DSO (objeto compartido). Las instrucciones para hacerlo vienen incluidas en el software *mod_log_sql* o pueden ser encontradas en el Web en http://www.grubbybaby.com/mod_log_sql/INSTALL, así que no se repetirán aquí. Una vez que se ha construido el módulo (o recompilado Apache), es tiempo de generar la base de datos y crear un usuario de MySQL sólo para Apache.

Analizando el tráfico web con Apache y MySQL

Primero vamos a crear un usuario en MySQL llamado **bingo** que tendrá acceso a la base de datos llamada **apache** cuando se conecte usando la contraseña **holahola**. Por supuesto, también necesitamos crear la base de datos. Nos conectamos a MySQL como super usuario y ejecutamos los siguientes comandos:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 3.23.52

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> GRANT ALL PRIVILEGES ON apache.* TO bingo IDENTIFIED BY 'holahola';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE DATABASE apache;
Query OK, 1 row affected (0.02 sec)
```

A continuación se tienen que agregar algunas directivas al archivo de configuración de Apache (*httpd.conf*) para que el **mod_log_sql** sepa que hacer. El Listado Uno muestra las directivas necesarias para un máquina con uno o más dominios.

Listado Uno: *adiciones al archivo httpd.conf para el mod_log_sql*

```
MySQLLoginInfo      localhost bingo holahola
MySQLSocketFile     /var/lib/mysql/mysql.sock
MySQLDatabase       apache
MySQLMassVirtualHosting On
```

La primera entrada **MySQLLoginInfo**, simplemente pone el host, nombre de usuario, y la contraseña que deben ser utilizados para conectarse a MySQL. **MySQLSocketFile** debe apuntar a la ubicación del socket de MySQL en tu sistema. **MySQLDatabase** especifica la base de datos a utilizar. Por su parte, el parámetro **MySQLMassVirtualHosting** registra cada dominio virtual en una tabla separada, no hay configuraciones adicionales involucradas cuando se agrega un nuevo dominio en el servidor. El registro es automático.

Existen además varias directivas de configuración que se pueden utilizar. Algunas permiten filtrar hits basados en varios criterios (agente usuario, URI, etc), también como configurar los datos que son registrados. Revisar la siguiente página para conocer todo el conjunto de directivas que se pueden usar. http://www.grubbybaby.com/mod_log_sql/directives.html

Una vez que se han agregado las directivas de configuración necesarias, debemos reiniciar Apache y visitar nuestro sitio Web. Si encontramos que hay al menos una tabla nueva en la base de datos **apache**, desde ese momento estamos teniendo registros en tiempo real de nuestro servidor web. Si algo salió mal, debemos de revisar el archivo *error_log* del servidor para tratar de averiguar que es lo que no funciona.

Debajo del antifaz

Con todo configurado y corriendo, es buen momento para ver como se están almacenando los registros de acceso.

Analizando el tráfico web con Apache y MySQL

Vamos a conectarnos a MySQL y obtener una lista de todas las tablas en la base de datos apache (deberá existir una tabla por cada dominio). El comando SHOW TABLES produce algo similar al Listado Dos.

Listado Dos: *la lista de las tablas de registros de acceso*

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.23.52

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SHOW TABLES;
+-----+
| Tables_in_apache |
+-----+
| access_casita_micasita_com |
+-----+
1 rows in set (0.00 sec)

mysql>
```

Debemos notar que cada nombre de tabla comienza con **access_** (prefijo de los archivos access_log), y que los puntos en los nombres de dominio han sido convertidos a guiones bajos ya que los puntos tiene significado especial para MySQL. Además en este ejemplo, existe una sola tabla puesto que este servidor web tiene sólo un nombre de dominio (casita.micasita.com).

Ahora vamos a ver cómo son almacenados los datos en estas tablas. El comando DESCRIBE genera la descripción de la tabla mostrada en el Listado Tres. Notar que las 22 columnas son por default NULL, y ninguna de ellas está indexada. Estos dos factores pueden resultar importantes cuando se comiencen a escribir consultas a las tablas de registro.

Listado Tres: *estructura de las tablas de registros de acceso*

```
mysql> DESCRIBE access_casita_micasita_com;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| agent | varchar(255) | YES | | NULL | |
| bytes_sent | int(10) unsigned | YES | | NULL | |
| child_pid | smallint(5) unsigned | YES | | NULL | |
| cookie | varchar(255) | YES | | NULL | |
| request_file | varchar(255) | YES | | NULL | |
| referer | varchar(255) | YES | | NULL | |
| remote_host | varchar(50) | YES | | NULL | |
| remote_logname | varchar(50) | YES | | NULL | |
| remote_user | varchar(50) | YES | | NULL | |
| request_duration | smallint(5) unsigned | YES | | NULL | |
| request_line | varchar(255) | YES | | NULL | |
| request_method | varchar(6) | YES | | NULL | |
| request_protocol | varchar(10) | YES | | NULL | |
| request_time | varchar(28) | YES | | NULL | |
| request_uri | varchar(50) | YES | | NULL | |
```

```
| server_port      | smallint(5) unsigned | YES | | NULL | | | |
| ssl_cipher      | varchar(25)          | YES | | NULL | | | |
| ssl_keysize     | smallint(5) unsigned | YES | | NULL | | | |
| ssl_maxkeysize  | smallint(5) unsigned | YES | | NULL | | | |
| status          | smallint(5) unsigned | YES | | NULL | | | |
| time_stamp      | int(10) unsigned     | YES | | NULL | | | |
| virtual_host    | varchar(50)          | YES | | NULL | | | |
+-----+-----+-----+-----+-----+-----+
22 rows in set (0.00 sec)
```

```
mysql>
```

Ejecutando algunas consultas

Con mucho del trabajo ya realizado, vamos a escribir algunas consultas para responder algunas preguntas comunes. Comenzaremos con unas consultas sencillas y trabajaremos posteriormente con ejemplos más complejos. Primero, ¿cuántos accesos tenemos registrados?

```
mysql> SELECT COUNT(*) FROM access_casita_micasita_com;
+-----+
| COUNT(*) |
+-----+
|      149 |
+-----+
1 row in set (0.28 sec)
```

¿Cuántos accesos tenemos registrados en las últimas 24 horas?

```
mysql> SELECT COUNT(*) FROM access_casita_micasita_com WHERE time_stamp BETWEEN
-> UNIX_TIMESTAMP(NOW()) - 86400 AND UNIX_TIMESTAMP(NOW());
+-----+
| COUNT(*) |
+-----+
|       34 |
+-----+
1 row in set (0.11 sec)
```

¿Cuántos accesos hemos registrado en este mes?

```
mysql> SELECT COUNT(*) FROM access_casita_micasita_com WHERE
-> MONTH(FROM_UNIXTIME(time_stamp)) = MONTH(NOW());
+-----+
| COUNT(*) |
+-----+
|       82 |
+-----+
1 row in set (0.19 sec)
```

Por supuesto, se pueden restringir cualquiera de estas consultas agregando más restricciones a la cláusula WHERE. Finalmente, vamos a contestar una de las preguntas formuladas anteriormente.

¿Cuáles son mis 10 URL's 404 más "famosos"? Ver el Listado Cuatro.

Listado Cuatro: *los 10 URL's 404 más "famosos"*

```
mysql> SELECT request_uri, count(*) AS cnt FROM access_casita_micasita_com
-> WHERE status = 404 GROUP BY request_uri ORDER BY cnt DESC LIMIT 10;
+-----+-----+
| request_uri          | cnt |
+-----+-----+
| /manual/index.html   | 6   |
| /manual/mod/core.html | 4   |
| /manual.html         | 2   |
+-----+-----+
3 rows in set (0.09 sec)
```

La profundidad del análisis que se puede ejecutar está limitado solamente por la habilidad que se tenga para escribir las sentencias SQL necesarias para obtener datos.

Cuando combinemos SQL con nuestro lenguaje de programación favorito podremos generar algunos reportes impresionantes. "Los datos están esperando para ser explorados".

Acelerando las cosas

Después de que se haya tenido la oportunidad de acumular algunos datos y escribir algunas consultas, se puede pensar acerca de cómo aumentar la velocidad de las consultas más comunes. Ya que ninguna de las columnas está indexada por default, se pueden agregar índices en aquellos lugares que tienen más sentido.

Si se necesita un repaso de utilización de índices en MySQL, hay que consultar "MySQL Performance Tuning" en el ejemplar de Julio de 2001, disponible en línea en http://www.linuxmagazine.com/2001-06/mysql_01.html (en inglés). Sin embargo, se recomienda no indexar campos hasta que esté uno cien por ciento seguro de hacerlo. Al tener demasiados índices puede que las cosas se vuelvan más lentas y se decremente la eficiencia del servidor MySQL.

La próxima vez...

Hemos visto cuál es la idea de usar **mod_log_sql** para tener los registros de acceso a nuestro sitio Web en una base de datos MySQL. Con estos registros efectivamente guardados en MySQL podemos generar muchas estadísticas interesantes. En un artículo siguiente escribiremos algún código en PHP para construir una aplicación Web que nos permita navegar interactivamente sobre los datos registrados.