

INGENIERÍA DE SERVIDORES (2016-2017)
DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS
UNIVERSIDAD DE GRANADA

Memoria Práctica 3

Alicia Rodríguez Gómez

May 9, 2017

Contents

1	Cuestión 1	3
1.1	¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes?	3
1.2	¿Qué significan las terminaciones .1.gz o .2.gz de los archivos del directorio?	4
2	Cuestión 2	4
2.1	Pruebe a ejecutar el comando, conectar un dispositivo USB y vuelva a ejecutar el comando. Copie y pegue la salida del comando. Comente qué observa en la información mostrada.	4
3	Cuestión 3	5
3.1	Ejecute el monitor de "System Performance" y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.	5
4	Cuestión 4	7
4.1	Cree un recopilador de datos definido por el usuario (modo avanzado) que incluya tanto el contador de rendimiento como los datos de seguimiento: Todos los referentes al procesador y al servicio web. Intervalo de muestra 5 segundos. Almacene el resultado en el directorio Escritoriologs. Incluya las capturas de pantalla de cada paso	7
5	Cuestión 5	13
5.1	Visite la web del proyecto y acceda a la demo que proporcionan http://demo.munin-monitoring.org/ donde se muestra cómo monitorizan un servidor. Monitoree varios parámetros y haga capturas de pantalla de lo que está mostrando comentando qué observa.	13
6	Cuestión 6	16
6.1	Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de strace o busque otro y coméntelo.	16
7	Cuestión 7	17
7.1	Escriba un script en Python o PHP y analice su comportamiento usando el profiler presentado.	17
8	Cuestión 8	20
8.1	Acceda a la consola mysql (o a través de phpMyAdmin) y muestre el resultado de mostrar el "profile" de una consulta (la creación de la BD y la consulta la puede hacer libremente).	20

1 Cuestión 1

1.1 ¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes?

Como se explicó en clase de teoría, en el directorio `/var/log` se encuentran los ficheros históricos que almacenan lo que ha ocurrido en el sistema. Dentro de dicho directorio encontramos el directorio `apt` que se corresponde con nuestro gestor de paquetes.

```
aliciarodgome@UbuntuServer:/$ cd /var/log/
aliciarodgome@UbuntuServer:/var/log$ ls
alternatives.log  dist-upgrade  faillog        mysql.err
apache2           dnsmg         fontconfig.log mysql.log
apt              dnsmg.0       fsck           syslog
auth.log          dnsmg.1.gz   installer      udev
boot.log          dnsmg.2.gz   kern.log       unattended-upgrades
bootstrap.log     dnsmg.3.gz   landscape      upstart
btmpt             dnsmg.4.gz   lastlog        wtmp
dbusconf-common  dpkg.log     mysql
aliciarodgome@UbuntuServer:/var/log$ cd apt
aliciarodgome@UbuntuServer:/var/log/apt$ ls
history.log  term.log
aliciarodgome@UbuntuServer:/var/log/apt$
```

Figure 1.1: Localización del archivo que permite ver que programas se han instalado con el gestor de paquetes

Accediendo a este último, encontramos el archivo `history.log` que es el archivo buscado, es decir, es el archivo que contiene los programas que se han instalado con el gestor de paquetes

```
GNU nano 2.2.6 Archivo: history.log

Start-Date: 2015-08-05 05:12:27
Commandline: apt-get --yes upgrade
Upgrade: initramfs-tools:amd64 (2.88dsf-4ubuntu6, 2.88dsf-4ubuntu6.2), initramfs-tools$
End-Date: 2015-08-05 05:12:42

Start-Date: 2017-04-10 20:49:16
Commandline: apt-get -o APT::Status-Fd=4 -o APT::Keep-Fds::=5 -o APT::Keep-Fds:$
Install: mdadm:amd64 (3.2.5-5ubuntu4.1)
End-Date: 2017-04-10 20:49:18

Start-Date: 2017-04-10 20:49:21
Commandline: apt-get -o APT::Status-Fd=4 -o APT::Keep-Fds::=5 -o APT::Keep-Fds:$
Install: libnl-genl-3-200:amd64 (3.2.21-1, automatic), wireless-regdb:amd64 (20$
End-Date: 2017-04-10 20:50:03

Start-Date: 2017-04-10 20:50:04
Commandline: apt-get -o APT::Status-Fd=4 -o APT::Keep-Fds::=5 -o APT::Keep-Fds:$
Install: pciutils:amd64 (3.2.1-1ubuntu5), libpci3:amd64 (3.2.1-1ubuntu5, automa$
End-Date: 2017-04-10 20:50:04

Start-Date: 2017-04-10 20:50:05
Commandline: apt-get -o APT::Status-Fd=4 -o APT::Keep-Fds::=5 -o APT::Keep-Fds:$
Install: usbutils:amd64 (007-2ubuntu1), libusb-1.0-0:amd64 (1.0.17-1ubuntu2, au$
End-Date: 2017-04-10 20:50:05

^G Ver ayuda ^O Guardar ^L Leer fich ^Y Pág. ant. ^X Cortar Tex ^C Posición
^X Salir ^J Justificar ^U Buscar ^V Pág. sig. ^U PegarTxt ^I Ortografía
```

Figure 1.2: Archivo `/var/log/apt/history.log`

En el archivo anterior aparecen todos los programas instalados con el gestor de paquetes `apt`, junto con la fecha de instalación y el comando utilizado para la instalación de este.

1.2 ¿Qué significan las terminaciones .1.gz o .2.gz de los archivos del directorio?

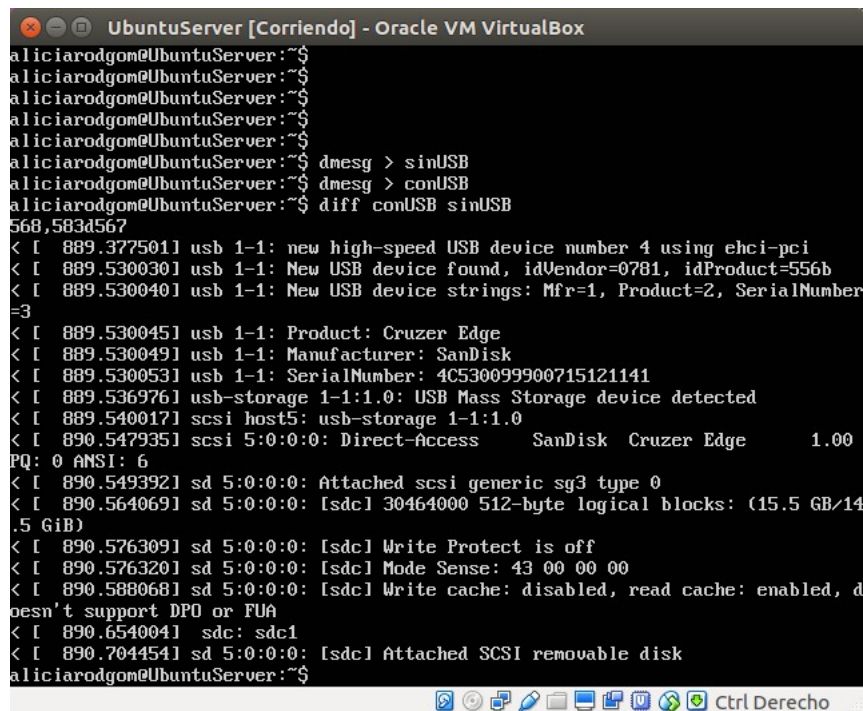
Como se muestra en la documentación de Ubuntu[1], los archivos cuyas terminaciones son del tipo .1.gz, .2.gz o *.gz donde * señalar un número natural, almacenan registros anteriores que ya no caben, debido a su gran tamaño, en el archivo .log. Además señalar que estos archivos se comprimen, de ahí su terminación .gz, porque como ya hemos dicho son de gran tamaño.

2 Cuestión 2

2.1 Pruebe a ejecutar el comando, conectar un dispositivo USB y vuelva a ejecutar el comando. Copie y pegue la salida del comando. Comente qué observa en la información mostrada.

Para conectar un dispositivo USB a la máquina virtual, previamente he tenido que descargar e instalar el paquete *Oracle VM VirtualBox Extension Pack*[4].

A continuación se muestra una captura del ejercicio planteado.



```
aliciarodgome@UbuntuServer:~$  
aliciarodgome@UbuntuServer:~$  
aliciarodgome@UbuntuServer:~$  
aliciarodgome@UbuntuServer:~$  
aliciarodgome@UbuntuServer:~$  
aliciarodgome@UbuntuServer:~$ dmesg > sinUSB  
aliciarodgome@UbuntuServer:~$ dmesg > conUSB  
aliciarodgome@UbuntuServer:~$ diff conUSB sinUSB  
568,583d567  
< [ 889.377501] usb 1-1: new high-speed USB device number 4 using ehci-pci  
< [ 889.530030] usb 1-1: New USB device found, idVendor=0781, idProduct=556b  
< [ 889.530040] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber  
=3  
< [ 889.530045] usb 1-1: Product: Cruzer Edge  
< [ 889.530049] usb 1-1: Manufacturer: SanDisk  
< [ 889.530053] usb 1-1: SerialNumber: 4C530099900715121141  
< [ 889.536976] usb-storage 1-1:1.0: USB Mass Storage device detected  
< [ 889.540017] scsi host5: usb-storage 1-1:1.0  
< [ 890.547935] scsi 5:0:0:0: Direct-Access SanDisk Cruzer Edge 1.00  
PQ: 0 ANSI: 6  
< [ 890.549392] sd 5:0:0:0: Attached scsi generic sg3 type 0  
< [ 890.564069] sd 5:0:0:0: [sdcl] 30464000 512-byte logical blocks: (15.5 GB/14  
.5 GiB)  
< [ 890.576309] sd 5:0:0:0: [sdcl] Write Protect is off  
< [ 890.576320] sd 5:0:0:0: [sdcl] Mode Sense: 43 00 00 00  
< [ 890.588068] sd 5:0:0:0: [sdcl] Write cache: disabled, read cache: enabled, d  
oesn't support DPO or FUA  
< [ 890.654004] sdc: sdc1  
< [ 890.704454] sd 5:0:0:0: [sdcl] Attached SCSI removable disk  
aliciarodgome@UbuntuServer:~$
```

Figure 2.1: Ejecución del comando dmesg con y sin un USB conectado

En primer lugar, ejecutamos el comando *dmesg* sin tener conectado el dispositivo USB. Guardamos el resultado de dicha ejecución en el archivo *sinUSB*. Seguidamente se realiza la misma acción pero con el dispositivo USB conectado. Dicha ejecución la guardamos en

el archivo *conUSB*. Finalmente vemos que información hay en el archivo *conUSB* respecto a la que hay en el archivo *sinUSB*.

En el archivo se puede ver información relacionada con el dispositivo USB conectado. Podemos ver que su capacidad es de 15.5 GB, el nombre del dispositivo, sobre el puerto que ha sido montado, *sdc1*, o detalles sobre su caché como son los permisos de lectura y escritura sobre esta.

3 Cuestión 3

3.1 Ejecute el monitor de "System Performance" y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.

Como se explica en el guión de prácticas, para ejecutar el monitor de "System Performance" se escribe en la consola el comando *perfmon*.

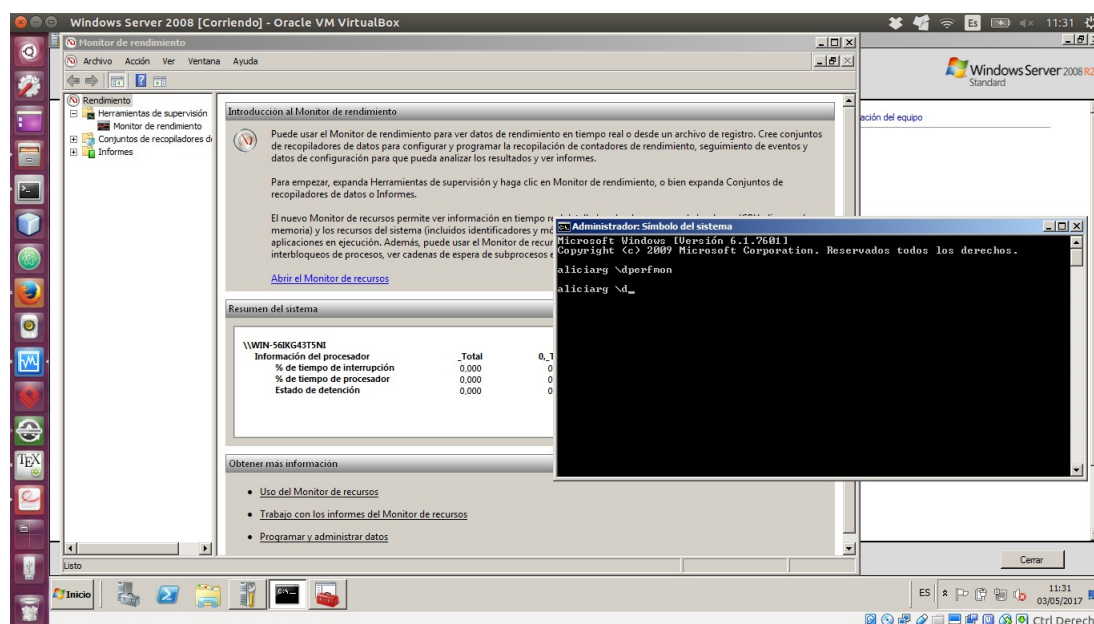


Figure 3.1: Ejecución del comando *perfmon* para abrir el monitor

Iniciamos el monitor de rendimiento. A continuación veremos una captura de este:

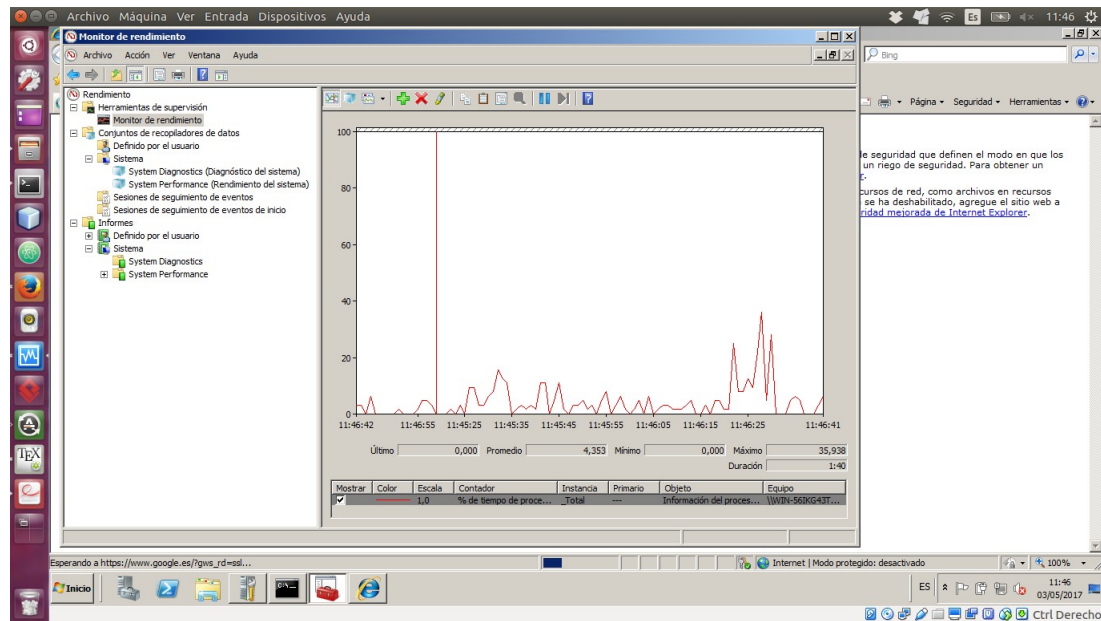


Figure 3.2: Monitor de rendimiento

En la gráfica de rendimiento anterior, se muestra el nivel de actividad de la máquina respecto al tiempo. Señalar que la gráfica muestra una actividad con altos y bajos que se corresponde al uso normal de la máquina. Los significativos picos que se encuentran a la derecha de la gráfica se corresponde con una búsqueda en el navegador. Dicha búsqueda aumenta la actividad de la máquina y este aumento se ve representado en la gráfica.

Para hacer una prueba se tiene el monitor encendido y durante un tiempo no ejecutamos ninguna acción en la máquina. Los resultados son los siguientes y además son coherentes con lo explicado anteriormente. La máquina no ejecuta nada por lo tanto su actividad es nula.

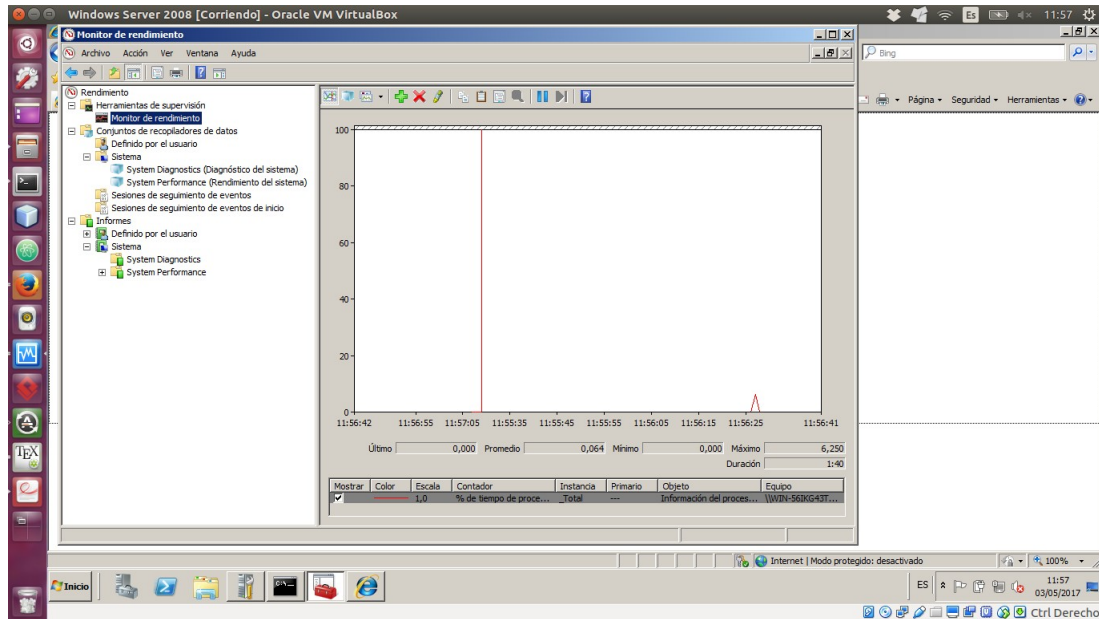


Figure 3.3: Monitor de rendimiento

4 Cuestión 4

- 4.1 Cree un recopilador de datos definido por el usuario (modo avanzado) que incluya tanto el contador de rendimiento como los datos de seguimiento: Todos los referentes al procesador y al servicio web. Intervalo de muestra 5 segundos. Almacene el resultado en el directorio Escritoriologs. Incluya las capturas de pantalla de cada paso**

Para crear un recopilador de datos continuamos en el mismo monitor del ejercicio anterior. Ahí accedemos en el panel de la parte izquierda a la sección *Conjunto de recopiladores de datos* y en dicha sección pinchamos con el botón derecho sobre el apartado *Definido por el usuario*, donde seleccionamos la opción de *nuevo*:

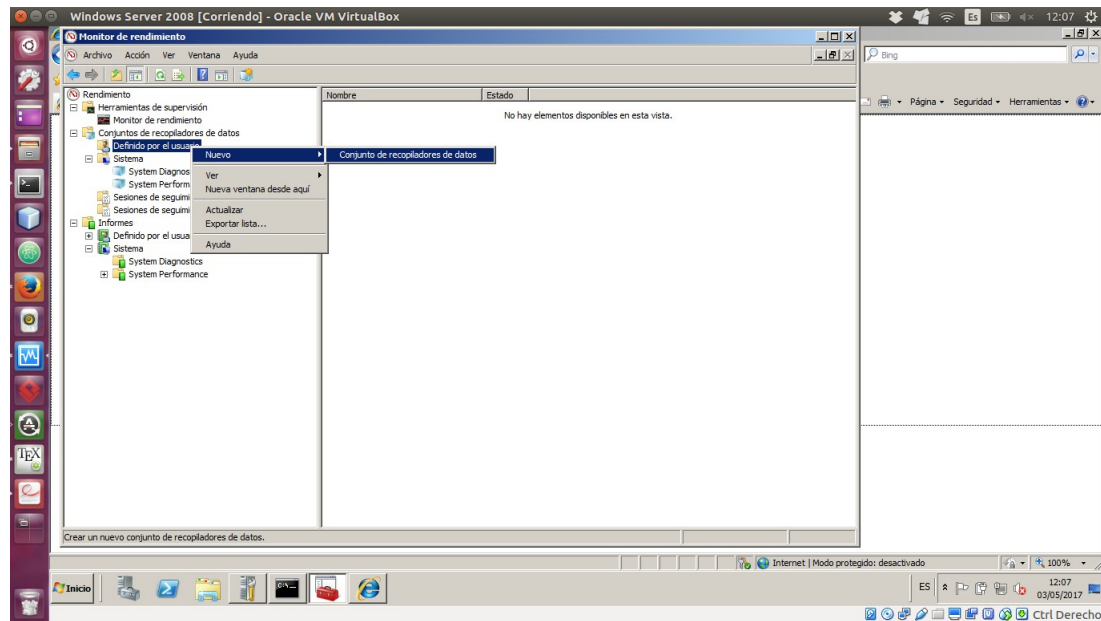


Figure 4.1: Creando un recopilador de datos definido por el usuario

A continuación le asignamos un nombre al recopilador, además seleccionamos el modo avanzado para crear el recopilador.

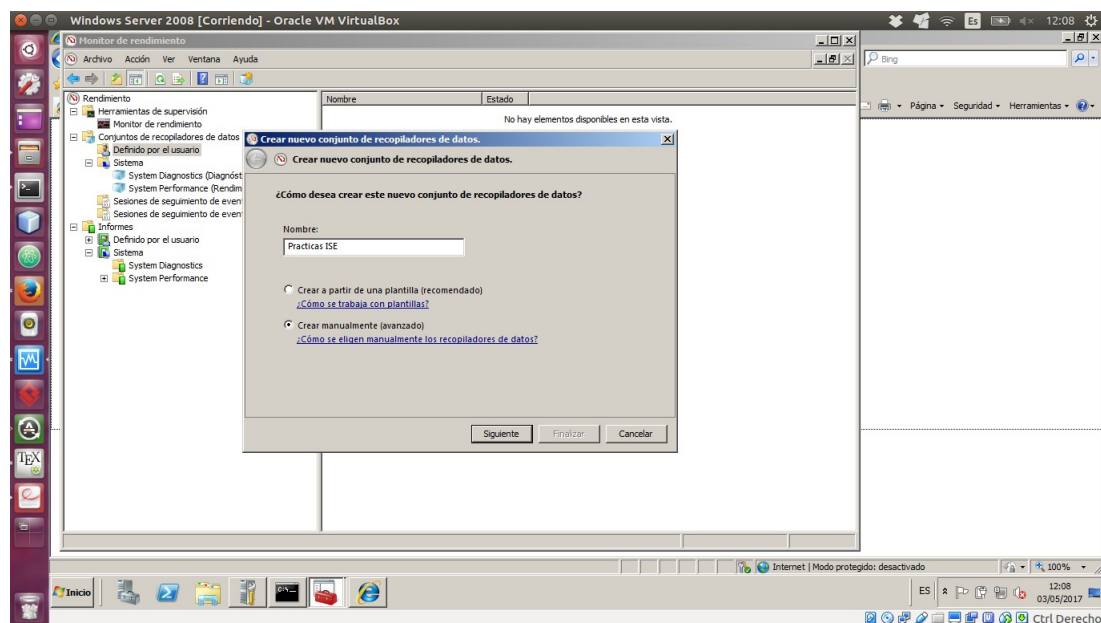


Figure 4.2: Asignación de nombre al recopilador de datos

Seguidamente seleccionamos las opciones *Contador de rendimiento* y *Datos de seguimiento de eventos*, tal y como se pide en el enunciado del ejercicio.

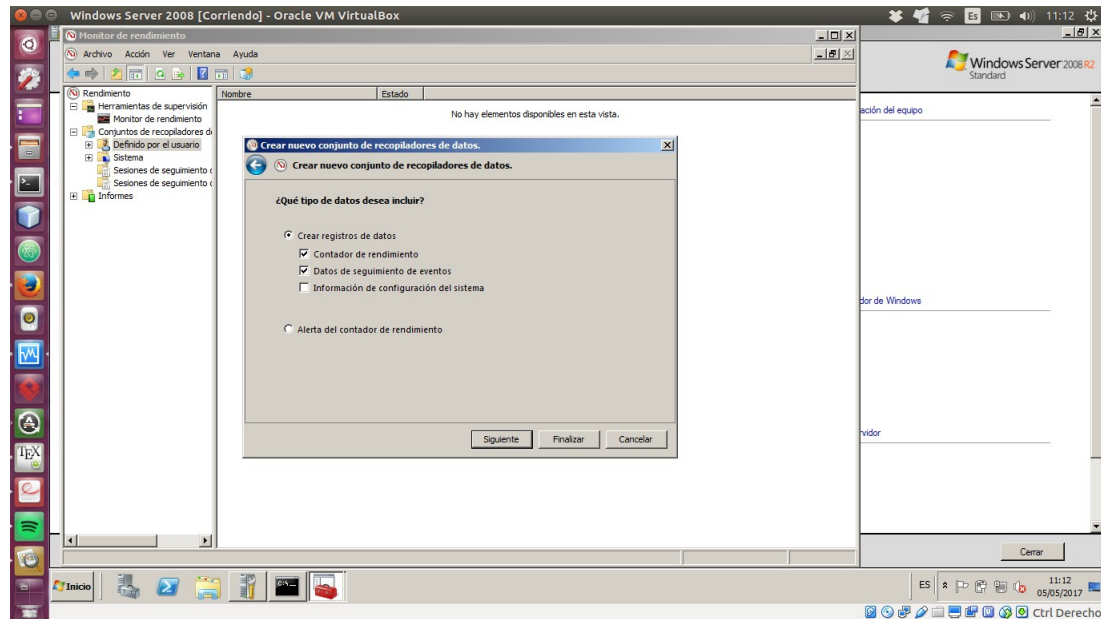


Figure 4.3: Selección del tipo de datos que desea incluir en el recopilador

Después se establece el intervalo de muestra, tal y como se pide es de 5 segundo. Además se selecciona la opción de *Agregar* para añadir contadores de rendimiento.

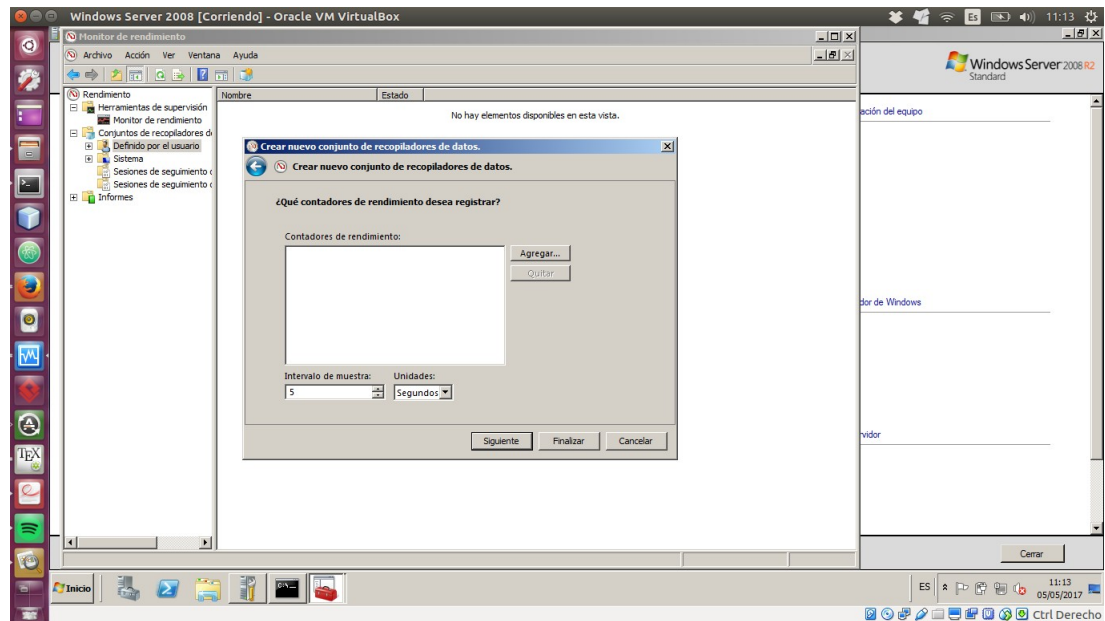


Figure 4.4: Estableciendo el intervalo de muestra

Tal y como se pide, se añaden los contadores de rendimiento referentes a el procesador y el servicio web

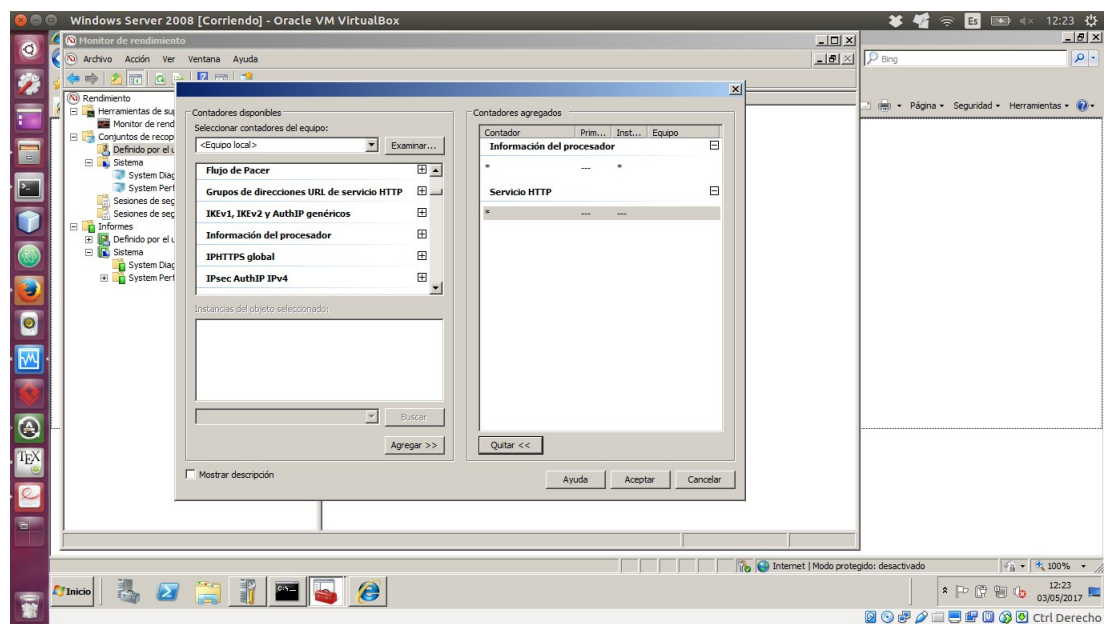


Figure 4.5: Agregando contadores del rendimiento

A continuación se muestra como han sido añadidos los contadores anteriormente nombrados.

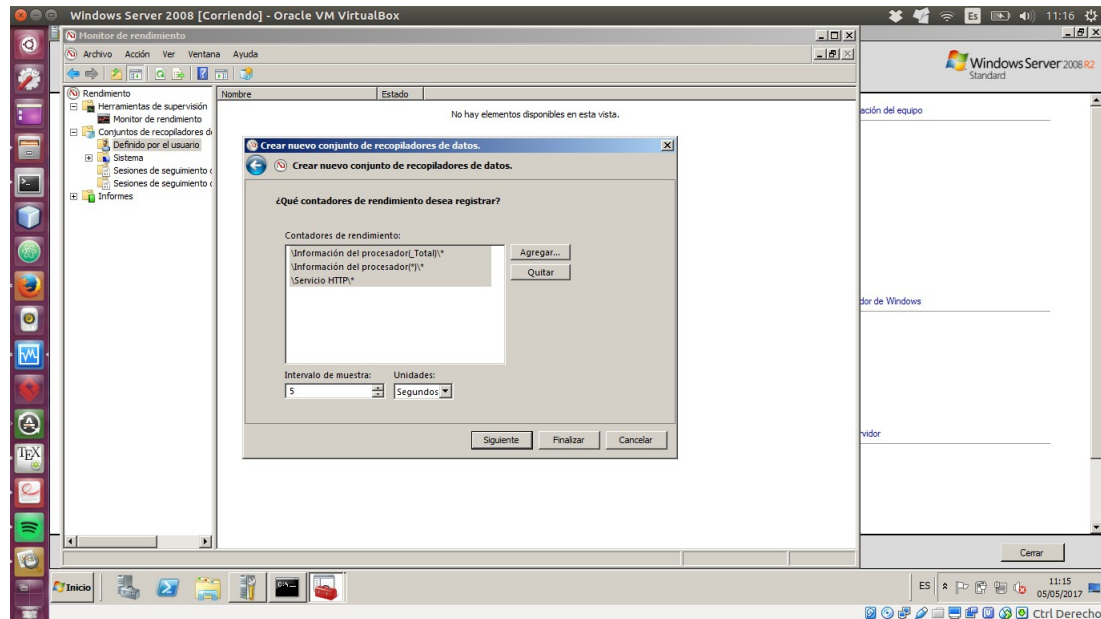


Figure 4.6: Contadores añadidos

Finalmente se selecciona el directorio donde se almacenaran los datos recopilados. Tal y como se pide el directorio es *Escritorio/log*.

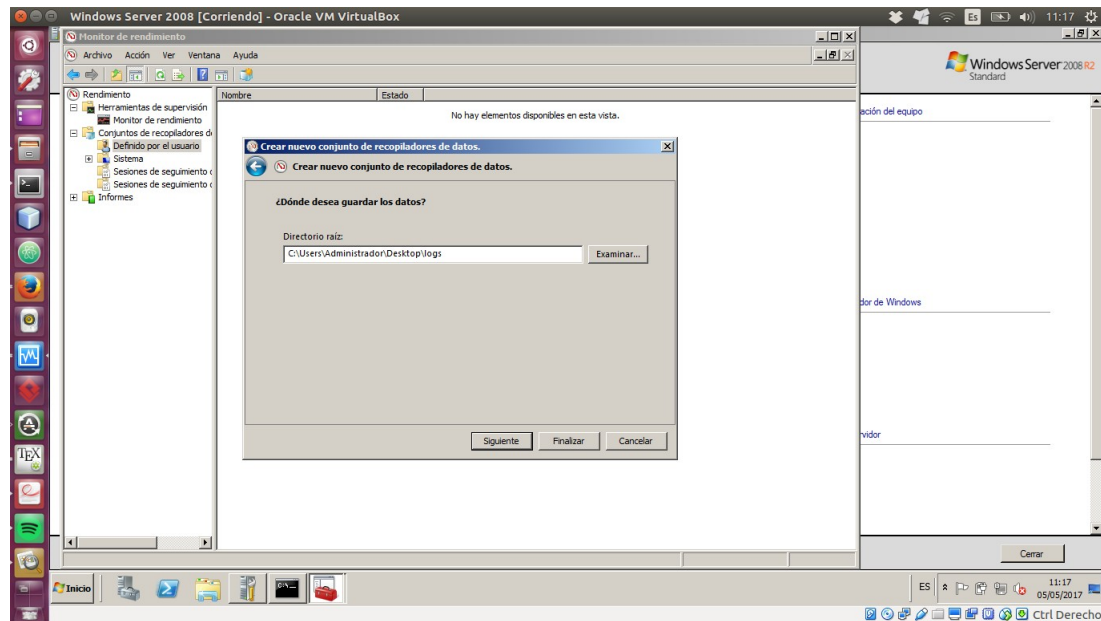


Figure 4.7: Selección del directorio de almacenamiento de los datos

Por último, una vez creado el recopilador vamos a ponerlo en marcha y veamos que resultados obtenemos. Para ello iniciamos el recopilador y esperamos el tiempo suficiente para que los datos sean recopilados. Cuando haya pasado dicho tiempo, podemos acceder al directorio /log donde se almacenan los datos y verlos.

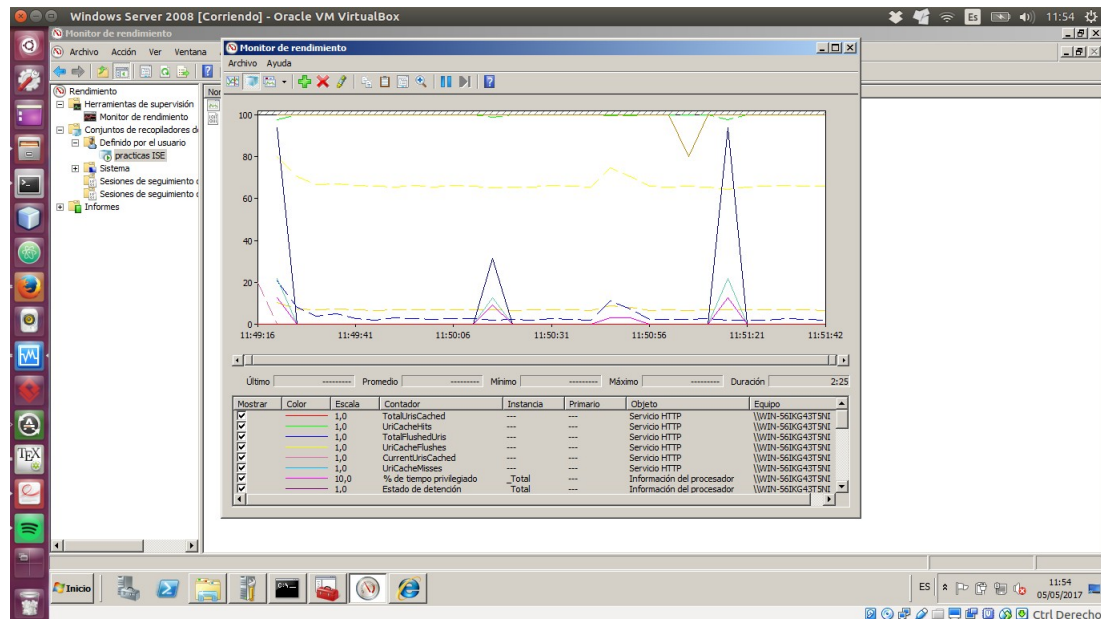


Figure 4.8: Resultados del recopilador de los datos definido anteriormente

Para interpretar la gráfica anterior debemos utilizar la leyenda que hay en la parte inferior de la gráfica. Cabe destacar de la gráfica que tanto en el procesador y el servicio HTTP, la actividad durante la recopilación de datos ha sido mínima ya que en la gráfica no se encuentran altas cotas de cada uno de los servicios analizados. Es más, todo lo referente al servicio HTTP está a cero y eso es porque durante la ejecución del recopilador de datos no ha habido ninguna petición al servicio HTTP.

5 Cuestión 5

5.1 Visite la web del proyecto y acceda a la demo que proporcionan <http://demo.munin-monitoring.org/> donde se muestra cómo monitorizan un servidor. Monitoree varios parámetros y haga capturas de pantalla de lo que está mostrando comentando qué observa.

Al acceder a la URL dada <http://demo.munin-monitoring.org/>, encontramos una demo que monitoriza un servidor. En el panel del lado izquierdo encontramos muchos

parámetros que se pueden monitorizar. Vamos a monitorizar varios de ellos y según los resultados de las gráficas dadas obtendremos datos de dicho servidor. Destacar que las gráficas mostradas a continuación muestran los datos recopilados en la última semana.



Figure 5.1: Monitorización del *uptime* del servidor

En la imagen anterior se muestra la monitorización durante la última semana del *System uptime*. Podemos observar que el tiempo en el que el servidor se mantiene activo es relativamente estable durante toda la última semana. Señalar que entre el Viernes y Sábado existe un intervalo en el que la gráfica tiene un valor nulo, esto se puede deber a algún fallo a la hora de tomar datos o cualquier otro problema similar. En este caso diríamos que en ese intervalo de tiempo tendríamos incertidumbre.

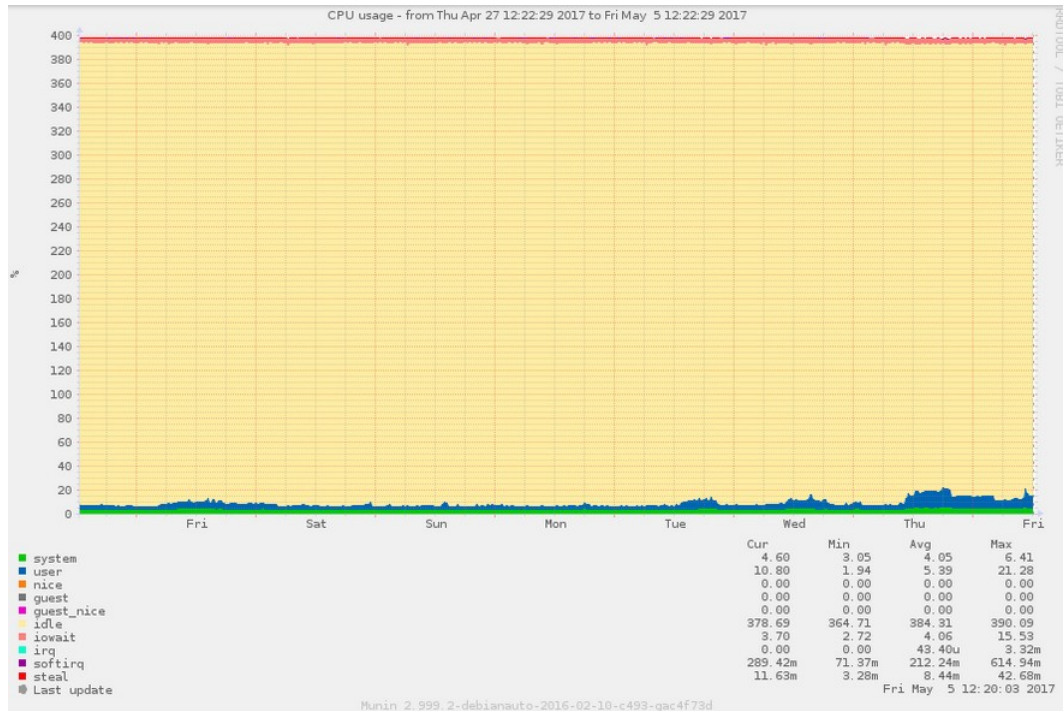


Figure 5.2: Monitorización del uso de la CPU del servidor

La gráfica anterior muestra detalladamente el uso de la CPU durante la última semana que ha hecho cada uno de los diferentes componentes. Es destacable de la gráfica el reducido uso de la CPU por parte del usuario y del sistema. Aunque en los últimos días el usuario ha hecho un mayor uso de esta, en término general en bastante reducida la cantidad. Por el contrario, destacamos que hay varios parámetros que si son elevados. Destacar que el servidor debe tener una actividad reducida porque su *idle* es bastante elevado. Junto a este vemos también que *softirq* y *steal* han estado muy activos durante la última semana. Por último, señalar en esta gráfica que el uso de el resto de recurso o parámetros que aparecen en la leyenda es casi nulo. Como conclusión destaco de esta gráfica que este servidor que ha sido monitorizado, ha tenido un reducido número de peticiones durante la última semana.

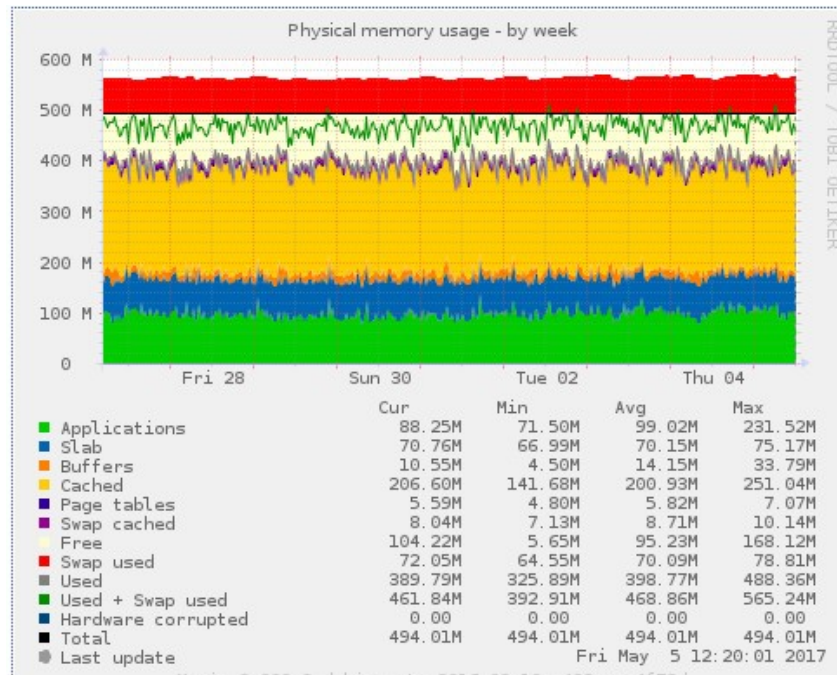


Figure 5.3: Monitorización del uso de la memoria física del servidor

Finalmente comentaremos esta gráfica sobre el uso de la memoria física durante la última semana. A primera vista puede parecer demasiado compleja la gráfica pero realmente no es así. Se puede observar que, dentro de su rango normal, cada recurso ha utilizado la memoria de manera regular. Por destacar ciertas irregularidades, *swap cached* y *used + swap used* son los recursos que no han utilizado la memoria de manera regular. Por lo demás todos han sido relativamente regulares en su uso.

6 Cuestión 6

6.1 Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de strace o busque otro y coméntelo.

A continuación comentaré el uso de strace[2] como herramienta para monitorizar llamadas al sistema. Esta herramienta, entre otras cosas, es útil cuando queremos averiguar la causa que ha provocado la muerte de procesos sin explicación o fallos en alguna componente del sistema que no dejan ningún rastro que nos permita solucionarlo.

Para problemas como los anteriores se podrá monitorizar llamadas y señales del sistema con la herramienta **strace**. Dicha herramienta se encuentra en cualquier plataforma Linux y no requiere una difícil configuración. Además utiliza comandos que nos son familiares como son algunos de los de bash.

Su funcionamiento es realmente sencillo, basta añadir el comando **strace** cuando lances un proceso. Este devuelve las llamadas del sistema y las señales del proceso lanzado junto con los errores encontrados, en el caso de que existan.

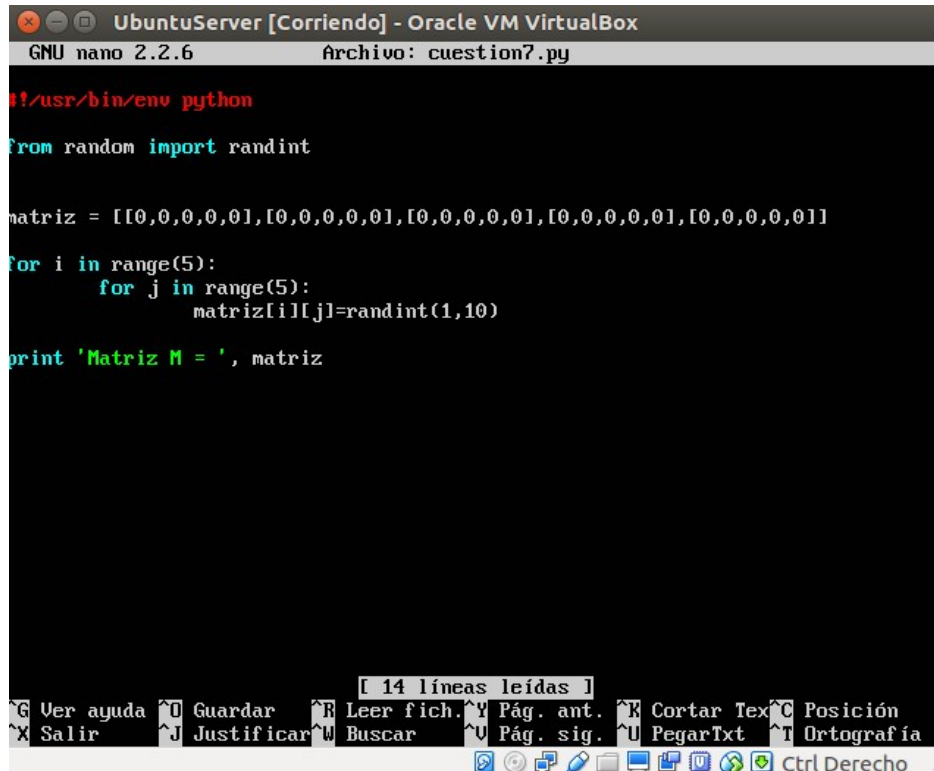
La utilidad real de los resultados obtenidos son las "*open calls*" porque estas te dan información sobre que archivo quiere abrir un archivo determinado y esto te facilitará para encontrar donde pueden estar tus problemas. Además si un programa falla cuando realiza una llamada al sistema, es útil identificar con precisión la causa de los errores. Esta herramienta añade donde se haya producido un fallo una línea con "`==1`" seguida de una explicación.

Por lo tanto se trata de una herramienta de monitorización de llamadas al sistema que nos permite solucionar en pocos minutos problemas que sin ella nos llevaría bastante más tiempo.

7 Cuestión 7

7.1 Escriba un script en Python o PHP y analice su comportamiento usando el profiler presentado.

En primer lugar, escribimos un script en Python. En este caso mi script es bastante sencillo y lo que hace es rellenar una matriz 5x5 de números aleatorios entre el 0 y el 10.



The screenshot shows a terminal window titled "UbuntuServer [Corriendo] - Oracle VM VirtualBox". The terminal is running the GNU nano 2.2.6 editor, editing a file named "cuestion7.py". The script content is as follows:

```
#!/usr/bin/env python

from random import randint

matriz = [[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]

for i in range(5):
    for j in range(5):
        matriz[i][j]=randint(1,10)

print 'Matriz M = ', matriz
```

The bottom of the terminal shows the nano editor's status bar with various keyboard shortcuts and a message "[14 líneas leídas]".

Figure 7.1: Script en Python

Seguidamente, tal y como se muestra en la documentación ofrecida en el gui3n de pr3cticas[3], se ejecuta *profiler* sobre el script Python creado anteriormente.



The screenshot shows a terminal window with the following commands and output:

```
aliciarodgom@UbuntuServer:~$ python -m profile cuestion7.py > resultado
aliciarodgom@UbuntuServer:~$ nano resultado
```

Figure 7.2: Ejecuci3n de profile

A continuaci3n abrimos el archivo donde se ha almacenado los resultados de la ejecuci3n anterior.

```

UbuntuServer [Corriendo] - Oracle VM VirtualBox
GNU nano 2.2.6 Archivo: resultado

Matriz M = [[4, 2, 1, 9, 1], [3, 8, 8, 9, 8], [1, 3, 10, 5, 2], [2, 1, 10, 10, 5]]
131 function calls in 0.008 seconds

Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   1   0.000    0.000    0.000    0.000  :0(exp)
   6   0.000    0.000    0.000    0.000  :0(getattr)
   6   0.000    0.000    0.000    0.000  :0(globals)
   1   0.000    0.000    0.000    0.000  :0(hexlify)
   2   0.000    0.000    0.000    0.000  :0(log)
   1   0.000    0.000    0.000    0.000  :0(openssl_md5)
   1   0.000    0.000    0.000    0.000  :0(openssl_sha1)
   1   0.000    0.000    0.000    0.000  :0(openssl_sha224)
   1   0.000    0.000    0.000    0.000  :0(openssl_sha256)
   1   0.000    0.000    0.000    0.000  :0(openssl_sha384)
   1   0.000    0.000    0.000    0.000  :0(openssl_sha512)
  25   0.000    0.000    0.000    0.000  :0(random)
   6   0.000    0.000    0.000    0.000  :0(range)
   1   0.000    0.000    0.000    0.000  :0(seed)
   1   0.008    0.008    0.008    0.008  :0(setprofile)
   1   0.000    0.000    0.000    0.000  :0(sqrt)
   1   0.000    0.000    0.000    0.000  :0(urandom)
   1   0.000    0.000    0.000    0.000  __future__.py:48(<module>)
   1   0.000    0.000    0.000    0.000  __future__.py:74(_Feature)

^G Ver ayuda ^O Guardar ^R Leer fich. ^Y Pág. ant. ^K Cortar Tex ^C Posición
^X Salir ^J Justificar ^W Buscar ^U Pág. sig. ^U PegarTxt ^T Ortografía

```

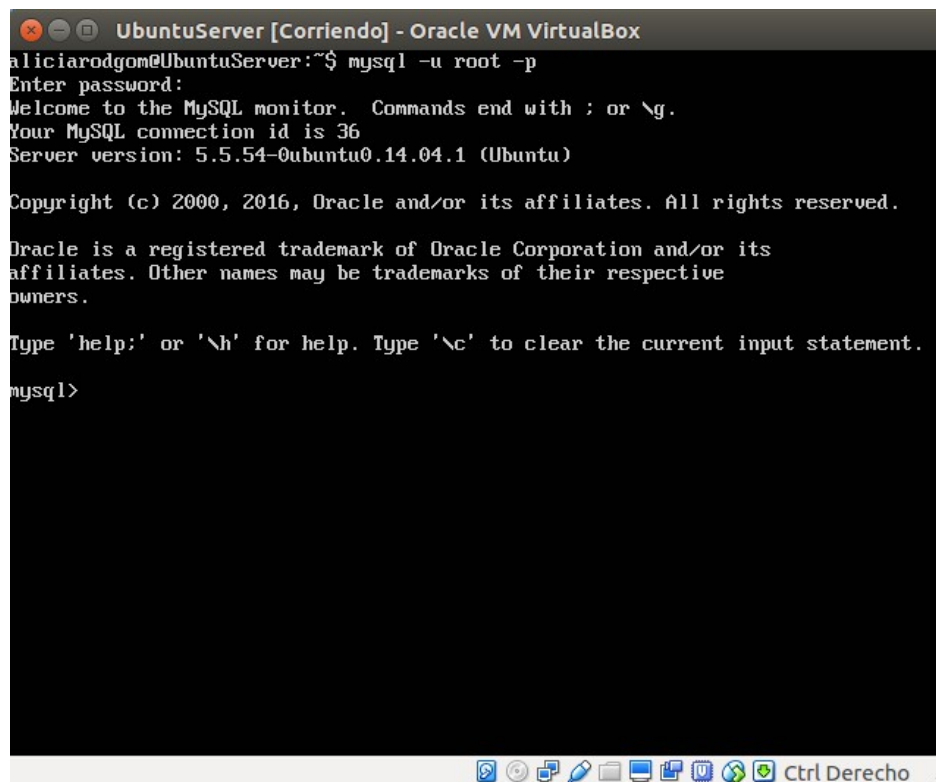
Figure 7.3: Archivo con los resultados de la ejecución del profile

En este archivo se encuentran datos de la ejecución del programa. Por ejemplo, nos indica que se han realizado 25 llamadas a la función *random*. Pero un dato más general del script que nos indica este archivo es que hemos realizado en total 131 llamadas a funciones y se ha tardado en ejecutar el programa 0.008 segundos. Por lo tanto destacar que el "*profiling*" te permite obtener toda la información de lo sucedido durante la ejecución de un script determinado.

8 Cuestión 8

8.1 Acceda a la consola mysql (o a través de phpMyAdmin) y muestre el resultado de mostrar el "profile" de una consulta (la creación de la BD y la consulta la puede hacer libremente).

En primer lugar, he accedido desde Ubuntu Server a la consola mysql. Para ello utilizamos el comando `mysql -u root -p` y seguidamente introducimos la contraseña. Señalar que *root* es mi usuario que tiene cuenta en mysql.

A screenshot of a terminal window titled "UbuntuServer [Corriendo] - Oracle VM VirtualBox". The terminal shows the command `mysql -u root -p` being executed. The prompt is `aliciarodgome@UbuntuServer:~$`. The user enters a password, and the MySQL monitor starts. The output includes: "Welcome to the MySQL monitor. Commands end with ; or \g.", "Your MySQL connection id is 36", "Server version: 5.5.54-0ubuntu0.14.04.1 (Ubuntu)", "Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.", "Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.", and "Type 'help;' or '\h' for help. Type '\c' to clear the current input statement." The prompt `mysql>` is shown at the bottom of the terminal window. The window has a standard Ubuntu interface with a taskbar at the bottom showing various icons and the text "Ctrl Derecho".

```
aliciarodgome@UbuntuServer:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.5.54-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Figure 8.1: Iniciando sesión en mysql

Una vez conectada a mysql ponemos el profiling a 1 y se hace un par de consultas a la base de datos. Señalar que se ha creado una simple base de datos llamada *alumno*, que contiene una única tabla llamada *nombre*.

```

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SET profiling = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| alumno |
| mysql |
| performance_schema |
| practicas3 |
+-----+
5 rows in set (0.01 sec)

mysql> SHOW FULL TABLES FROM alumno;
+-----+-----+
| Tables_in_alumno | Table_type |
+-----+-----+
| nombre | BASE TABLE |
+-----+-----+
1 row in set (0.00 sec)

mysql>

```

Figure 8.2: Consultas sobre mysql

A continuación ejecutamos el comando *show profiles* que nos muestra todas las consultas realizadas sobre la base de datos. Como se puede comprobar aparecen las dos consultas realizadas anteriormente.

```

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| alumno |
| mysql |
| performance_schema |
| practicas3 |
+-----+
5 rows in set (0.01 sec)

mysql> SHOW FULL TABLES FROM alumno;
+-----+-----+
| Tables_in_alumno | Table_type |
+-----+-----+
| nombre | BASE TABLE |
+-----+-----+
1 row in set (0.00 sec)

mysql> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 1 | 0.00051075 | SHOW DATABASES |
| 2 | 0.00046650 | SHOW FULL TABLES FROM alumno |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>

```

Figure 8.3: Ejecución de show profiles

Seguidamente elegimos cualquiera de las consultas que vienen representadas por un identificador. En este caso yo he elegido la segunda consulta para ejecutar sobre ella el "profile".

```

2 rows in set (0.00 sec)

mysql> show profile for query 2;
+-----+-----+
| Status                               | Duration |
+-----+-----+
| starting                             | 0.000061 |
| checking permissions                  | 0.000029 |
| Opening tables                        | 0.000070 |
| System lock                           | 0.000014 |
| init                                 | 0.000012 |
| optimizing                            | 0.000006 |
| statistics                            | 0.000010 |
| preparing                             | 0.000010 |
| executing                             | 0.000007 |
| checking permissions                  | 0.000174 |
| Sending data                          | 0.000024 |
| end                                   | 0.000005 |
| query end                             | 0.000005 |
| closing tables                        | 0.000002 |
| removing tmp table                    | 0.000008 |
| closing tables                        | 0.000004 |
| freeing items                         | 0.000019 |
| logging slow query                    | 0.000003 |
| cleaning up                           | 0.000004 |
+-----+-----+
19 rows in set (0.00 sec)

mysql>

```

Figure 8.4: Ejecución de profile sobre la segunda consulta

Por último comentar la información que obtenemos al hacer profile sobre la segunda consulta. La información obtenida está relacionada con el tiempo que se tarda en ejecutar cada una de las subconsultas que provoca la consulta realizada. Estas subconsultas son del tipo comprobar permisos, abrir y cerrar tablas, optimización, etc. Señalar que la consulta realizada es bastante rápida ya que la base de datos que se ha hecho la consulta ha sido creada por mí y solo he creado una tabla. Por lo tanto las búsquedas son bastante rápidas.

References

- [1] <https://help.ubuntu.com/community/ListInstalledPackagesByDate>, Consultado el 3 de Mayo del 2017.
- [2] <http://blog.softlayer.com/2013/sysadmin-tips-and-tricks-using-strace-to-monitor-system>
utm_source=twitter&utm_medium=social&utm_content=

beyond-the-command-line-with-strace&utm_campaign=blog_
development-tips-and-tricks, Consultado el 7 de Mayo del 2017.

[3] <https://docs.python.org/2/library/profile.html>, Consultado el 7 de Mayo del 2017.

[4] <http://download.virtualbox.org/virtualbox/5.0.36/>, Descargado el 8 de Mayo del 2017.