

Ingeniería de Servidores (2015-2016)
GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS
UNIVERSIDAD DE GRANADA

Memoria Práctica 3

Iván Sevillano García

28 de agosto de 2016

Índice

1. Conociendo el subsistema de archivos.	3
1.1. Monitorización de Sistemas Linux	3
1.2. Programación de tareas con cron	3
1.3. Analizando que ocurre en el Kernel con Dmesg	4
2. Monitorización con Windows.	5
3. Monitorización del Hardware.	11
4. Otros monitores de sistema.	12
4.1. Munin	12
4.2. Zabbix	14
5. ZABBIX ME FALTA	14
5.1. Monitorizando un servicio	21
6. Profiling	21
6.1. Python	21
6.2. Bases de datos	23

1. Conociendo el subsistema de archivos.

1.1. Monitorización de Sistemas Linux

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

En el directorio `/var/log/apt/`, donde se encuentran los archivos `history.log` y `term.log`. Además, encontramos también archivos con los mismos nombres con las extensiones que abajo se especifican (donde podemos encontrar cualquier número como extensión).

- ¿Qué significan las terminaciones `.1.gz` o `.2.gz` de los archivos en ese directorio?

Cada vez que llenamos lo suficiente nuestro archivo `history.log`, el sistema lo comprime y crea un nuevo `history.log` para almacenar la información sobre nuevos paquetes que instalemos. Las extensiones con distinto número no son más que formas de numerar los distintos paquetes por orden. El orden que siguen va desde el más reciente (extensión `.1.gz`) hasta el más antiguo.

1.2. Programación de tareas con cron

- ¿Qué archivo ha de modificar para programar una tarea? Escriba la línea necesaria para ejecutar una vez al día una copia del directorio `/codigo a /seguridad/fecha` donde *fecha* es la fecha actual (puede usar el comando `date`)

Según [1], debemos modificar los archivos en `/var/spool/cron/crontabs/`, sin embargo no están hechos para ser modificados directamente. Se nos ofrece, para modificar el archivo de programación de tareas, el comando `crontab`, que con la opción `-e`, nos deja modificar desde la terminal con el editor de texto que prefiramos.

La línea que debemos añadir será la siguiente:

```
00 0 * * * /home/ivan/script.sh
```

en donde `script.sh` tiene permisos de ejecución y es el siguiente:

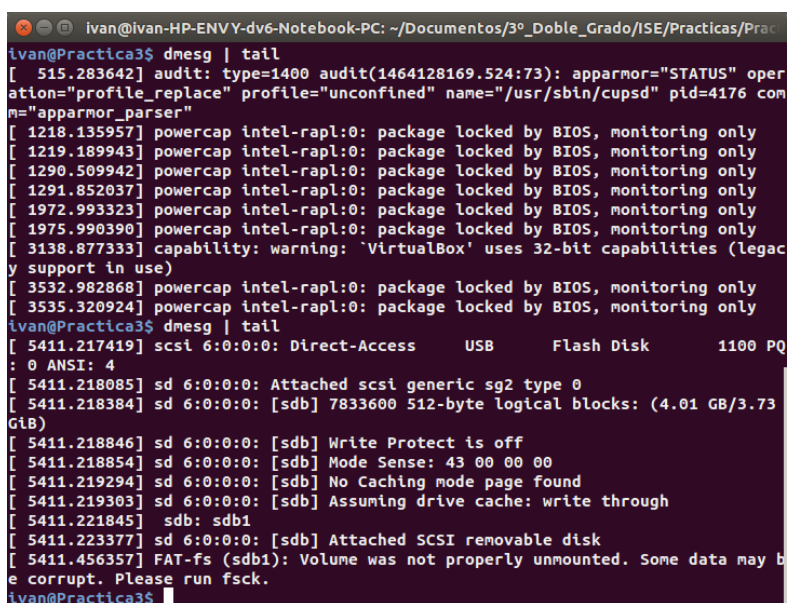
```
#!/bin/bash
cp -r /codigo /seguridad/`date +%y %m %d %H %M %S`
```

Los primeros cinco *items* de la sentencia indican cuando queremos que se realice la tarea que se muestra a continuación, por este orden: minuto hora día Del Mes Mes Día De La Semana. No hay que especificar todos los campos, ya que el poner un asterisco(*) nos dice que no tenga en cuenta este campo. Por ello, en nuestra

sentencia, queremos que se ejecute el comando siempre que sean las doce de la noche, sea el día que sea, el mes que sea y el día de la semana que sea. El parámetro 'date' nos sirve para cambiar el nombre del archivo a la fecha actual y así tener archivos con distinto nombre.

1.3. Analizando que ocurre en el Kernel con Dmesg

- Pruebe a ejecutar el comando, conectar un dispositivo USB y vuelva a ejecutar el comando. Copie y pegue la salida del comando (considere usar dmesg | tail). Comente qué observa en la información mostrada.



```
ivan@ivan-HP-ENVY-dv6-Notebook-PC: ~/Documentos/3º_Doble_Grado/ISE/Practicas/Prac
ivan@Practica3$ dmesg | tail
[ 515.283642] audit: type=1400 audit(1464128169.524:73): apparmor="STATUS" oper
ation="profile_replace" profile="unconfined" name="/usr/sbin/cupsd" pid=4176 com
m="apparmor_parser"
[ 1218.135957] powercap intel-rapl:0: package locked by BIOS, monitoring only
[ 1219.189943] powercap intel-rapl:0: package locked by BIOS, monitoring only
[ 1290.509942] powercap intel-rapl:0: package locked by BIOS, monitoring only
[ 1291.852037] powercap intel-rapl:0: package locked by BIOS, monitoring only
[ 1972.993323] powercap intel-rapl:0: package locked by BIOS, monitoring only
[ 1975.990390] powercap intel-rapl:0: package locked by BIOS, monitoring only
[ 3138.877333] capability: warning: 'VirtualBox' uses 32-bit capabilities (legac
y support in use)
[ 3532.982868] powercap intel-rapl:0: package locked by BIOS, monitoring only
[ 3535.320924] powercap intel-rapl:0: package locked by BIOS, monitoring only
ivan@Practica3$ dmesg | tail
[ 5411.217419] scsi 6:0:0:0: Direct-Access      USB          Flash Disk      1100 PQ
: 0 ANSI: 4
[ 5411.218085] sd 6:0:0:0: Attached scsi generic sg2 type 0
[ 5411.218384] sd 6:0:0:0: [sdb] 7833600 512-byte logical blocks: (4.01 GB/3.73
GiB)
[ 5411.218846] sd 6:0:0:0: [sdb] Write Protect is off
[ 5411.218854] sd 6:0:0:0: [sdb] Mode Sense: 43 00 00 00
[ 5411.219294] sd 6:0:0:0: [sdb] No Caching mode page found
[ 5411.219303] sd 6:0:0:0: [sdb] Assuming drive cache: write through
[ 5411.221845] sdb: sdb1
[ 5411.223377] sd 6:0:0:0: [sdb] Attached SCSI removable disk
[ 5411.456357] FAT-fs (sdb1): Volume was not properly unmounted. Some data may b
e corrupt. Please run fsck.
ivan@Practica3$
```

Figura 1.1: Mensaje mostrado por dmesg antes y después de conectar un pen-drive

Antes de conectar el pen-drive, no teníamos ninguna referencia del mismo. Tras conectarlo, el primer mensaje que se nos muestra es la presencia del mismo. También nos ofrece más información como que el disco no fue debidamente desmontado.

2. Monitorización con Windows.

- Ejecute el monitor de “System Performance” y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece. Al ejecutar el monitor de rendimiento de windows, nos aparece la siguiente interfaz general:

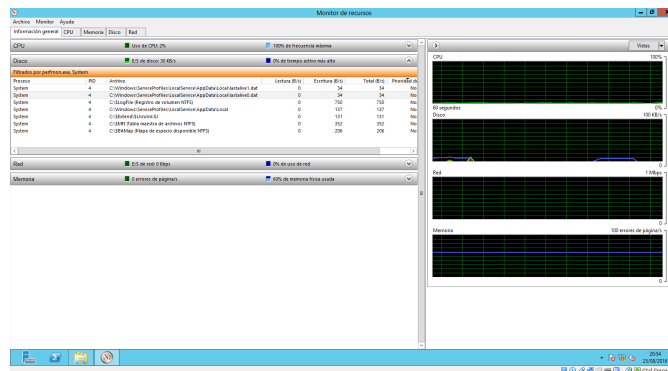


Figura 2.1: Monitor de rendimiento en Windows.

en donde vemos de forma general el rendimiento que está teniendo el sistema. Si queremos enfocarnos en algún campo en concreto haremos click en la gráfica la cuál esté monitorizando el componente que nos interese. Por ejemplo, para obtener información del porcentaje de CPU usado:

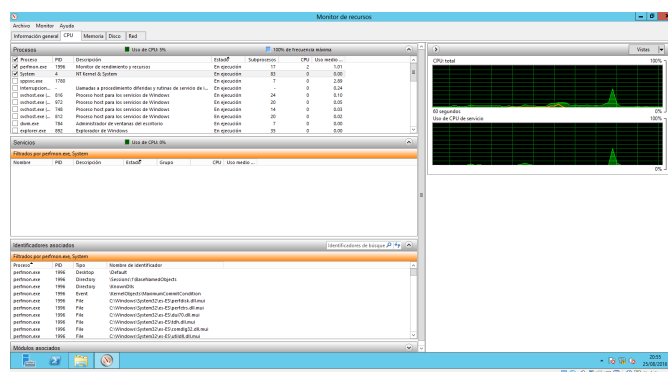


Figura 2.2: Uso de la CPU en windows.

en donde se ve en las gráficas que el uso de la CPU está por debajo del 50 % desde hace 60 segundos.

Para obtener datos de la memoria usada por windows:

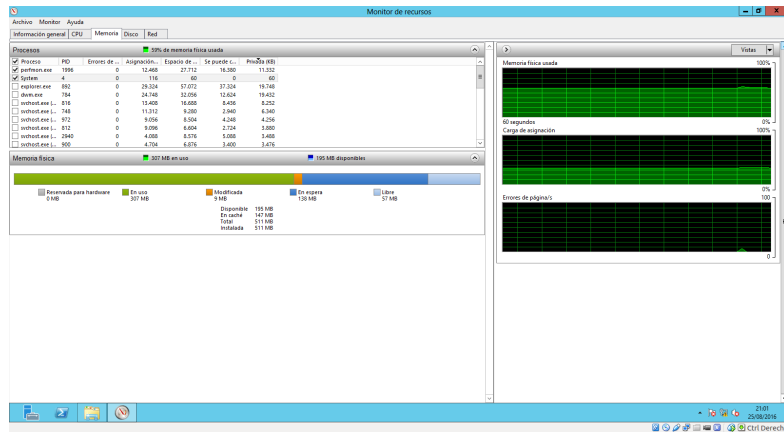


Figura 2.3: Uso de la memoria total de Windows.

en concreto, se ve que en nuestra máquina virtual se está usando mas de la mitad de la memoria que le hemos dado, que hay 195Mb de memoria disponible y 147 en caché.

- 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, al proceso y al servicio web, con un intervalo de muestra de 15 segundos y que almacene el resultado en el directorio */Escritorio/logs*. Incluya las capturas de pantalla de cada paso.

Para empezar a crear un recopilador de datos clicamos dentro del rendimiento del sistema en recopilador de datos y creamos un nuevo recopilador de datos creado por el usuario:

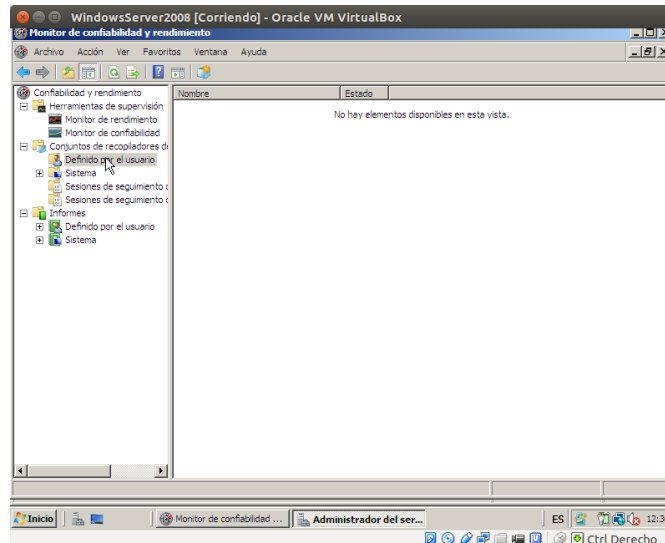


Figura 2.4: Donde encontrar la creación del recopilador de datos.

Tras esto, se nos pregunta de qué campos tiene que recopilar datos el sistema, en nuestro caso, los datos referentes al contador de rendimiento como los datos de seguimiento.

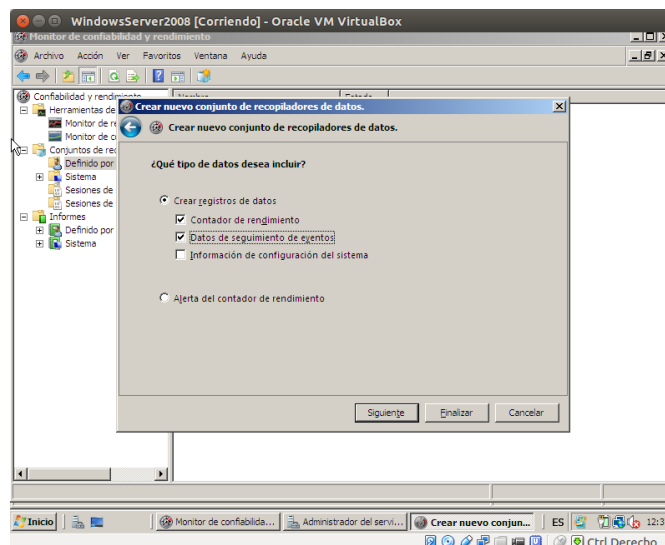


Figura 2.5: Inicio de creación y ubicación del recopilador de datos.

Se nos pregunta ahora sobre que componentes vamos a monitorizar. Nosotros escogemos los referentes a los procesos, procesamiento y servicio web:

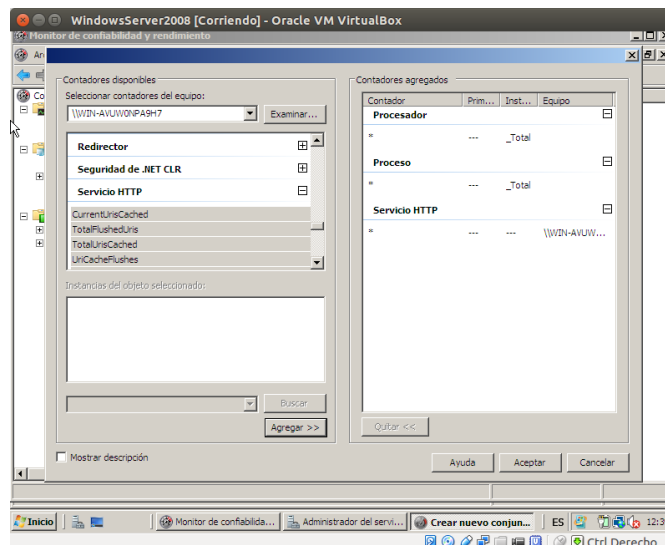


Figura 2.6: Escogiendo los componentes a monitorizar.

A la hora de darle a aceptar, la opción por defecto de tiempo para monitorizar el sistema de 15 segundos:

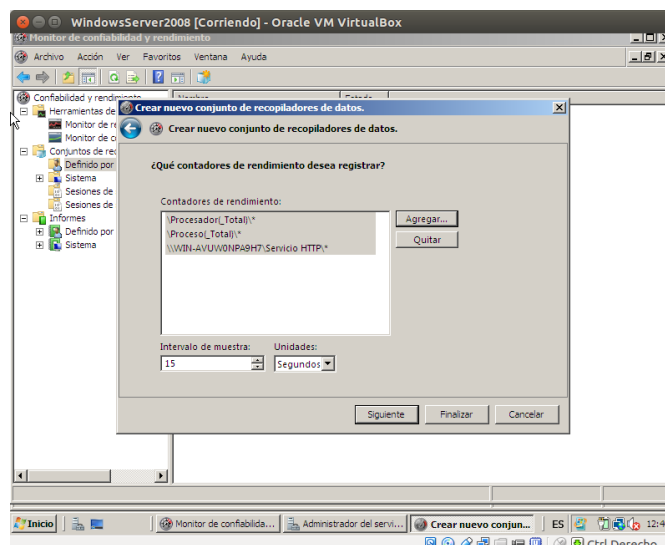


Figura 2.7: Intervalos de tiempo para obtener mediciones.

Por último, se nos pregunta donde queremos guardar los datos:

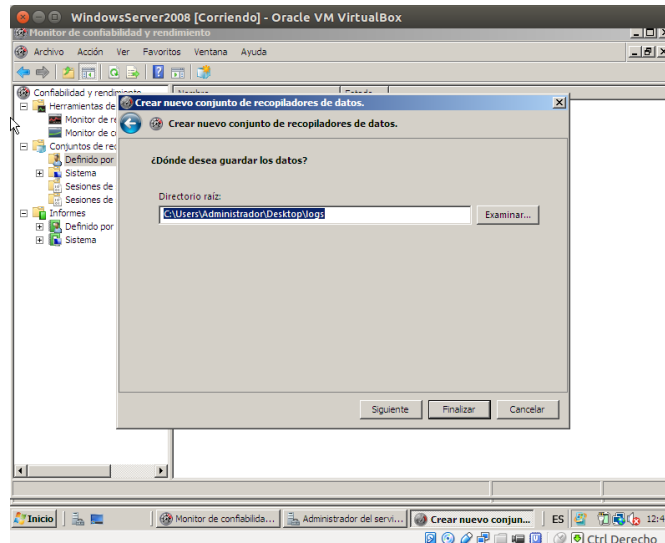


Figura 2.8: Guardar los datos en el */Escritorio/logs (/Desktop/logs)*.

Ahora ya tenemos creado nuestro recopilador de datos, pero para que este comience a monitorizar, tenemos que iniciar el seguimiento:

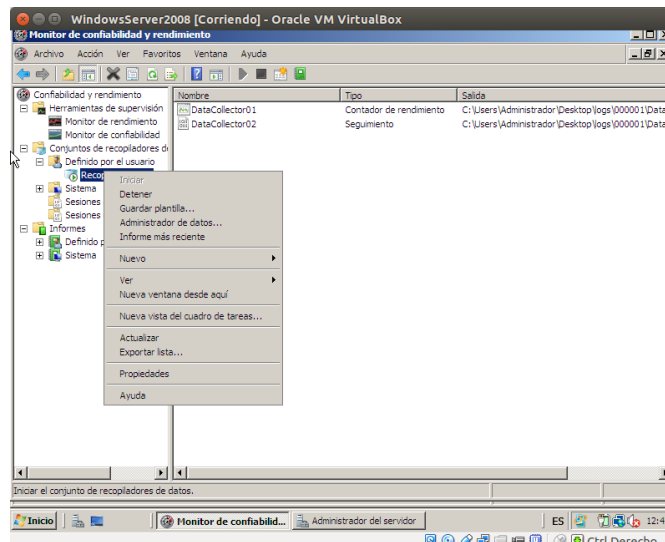


Figura 2.9: Iniciar seguimiento con click derecho->Iniciar.

Nos vamos al directorio donde hemos guardado el monitor y lo ejecutamos:

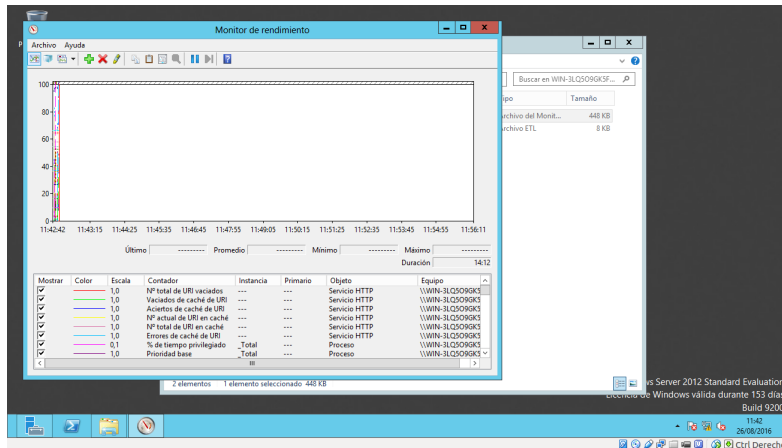


Figura 2.10: Monitor de rendimiento en ejecución.

Como vemos, la gráfica está demasiado llena. Deseleccionando algunas de ellas:

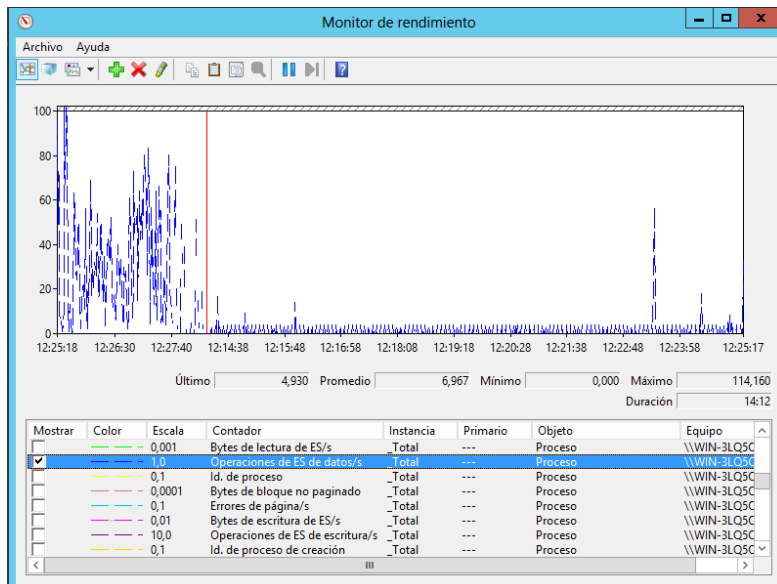


Figura 2.11: Operaciones de entrada y salida en el sistema.

obtenemos la gráfica de un aspecto concreto. En la gráfica mostrada se ilustra el número de operaciones de entrada/salida por segundo del sistema. Podemos ver el número medio de operaciones, la última medición realizada y otros datos estadísticos de estas mediciones. Ej: El número de operaciones entrada/salida por segundo máximo que ha realizado el sistema es de más de 114 operaciones por segundo.

3. Monitorización del Hardware.

- Instale alguno de los monitores comentados arriba en su máquina y pruebe a ejecutarlos (tenga en cuenta que si lo hace en la máquina virtual, los resultados pueden no ser realistas). Alternativamente, busque otros monitores para hardware comerciales o de código abierto para Windows y Linux.

El primer monitor instalado en el sistema de *hddtemp* [2], que nos muestra la temperatura de los discos duros de nuestro PC en Linux. Tras ejecutar *hddtemp* sobre un disco duro en concreto (En nuestro caso */dev/sda*), nos puede salir una salida como la siguiente:

```
/dev/sda : ST500LM012 HN – M500MBB : 48°C
```

en el cual nos muestra que la temperatura del mismo es 48 grados celsius. Mas información en el manual, antes citado.

Para detectar los sensores de la placa base necesitamos instalar la librería *lm – sensors* [3] (la página del proyecto está caída). Esta librería es necesaria para el siguiente monitor gráfico del sistema, *psensor*, que se instala con el paquete del mismo nombre. Este monitor nos ofrece de forma gráfica la información sobre temperatura y porcentaje de la CPU usado. Un ejemplo de la interfaz es la siguiente:

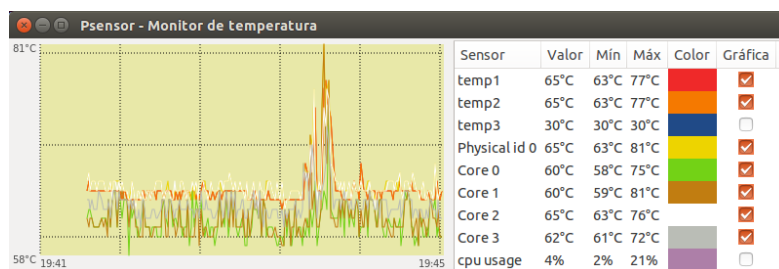


Figura 3.1: Gráfica que muestra psensor sobre la temperatura de algunos componentes del sistema.

Además de estos, se pueden encontrar en [4] otras herramientas de monitorización, como por ejemplo Hardinfo[5], que es una aplicación libre la cual muestra información acerca de los distintos componentes del sistema como el procesador, distintos dispositivos como USB's, o incluso realizar benchmarking sobre los mismos.

En Windows[6], algunas de las herramientas que podemos usar para monitorizar el sistema en lugar de las ya dadas son las siguientes, entre otros:

- Prime95[7], que se encarga de cargar de trabajo todos los núcleos para comprobar su rendimiento.
- MemTest86[8], utilizado para testear la memoria RAM.

4. Otros monitores de sistema.

4.1. Munin

- 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.

El primer parámetro que vamos a monitorizar es el uso de la CPU. La siguiente imagen nos dice cual es el porcentaje usado por distintos consumidores al día:

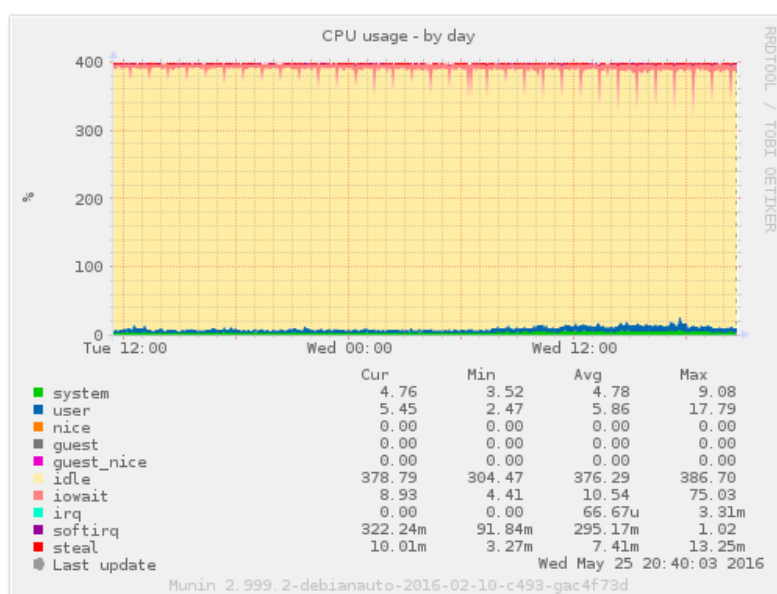


Figura 4.1: Uso de la CPU durante un día en el servidor que monitoriza Munin

En el se muestra los usos que ha tenido la CPU. Entre ellos, está el porcentaje de uso de la CPU de el usuario user, que es de alrededor del 5.48

Otro parámetro a monitorizar que nos ofrece Munin es el número de procesos y el estado de los mismos:

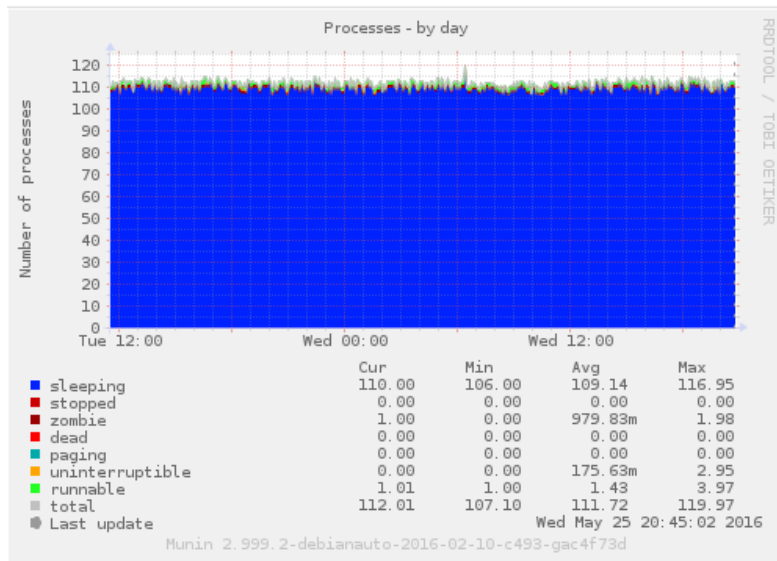


Figura 4.2: Número de procesos y estado de los mismos.

En este gráfico vemos que hay alrededor de 110 procesos lanzados de los cuales la gran mayoría está *sleeping*, o esperando a estar corriendo. También vemos que apenas han quedado procesos zombies. Es importante notar que siempre hay al menos un proceso ejecutandose.

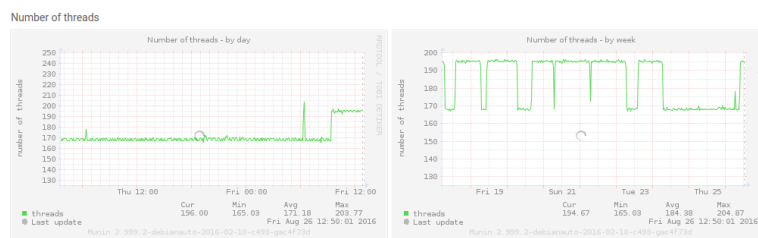


Figura 4.3: Número de hebras que hay en cada momento en el servidor.

Este parámetro nos indica el número de hebras en cada momento. Los dos gráficos nos muestran distintos intervalos de tiempo: por día y por semana. Hay alrededor de 171.32 hebras de media en el día de hoy en contraste con una media de 184 hebras en la semana. Esto es debido a que a lo largo de la semana ha tenido más hebras que hoy.

Por último, en el parámetro experimental:

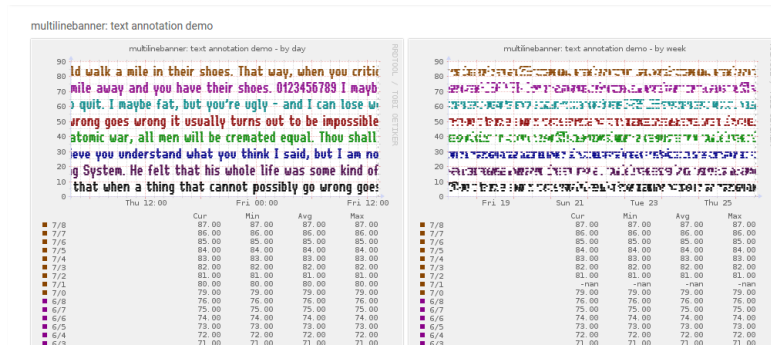


Figura 4.4: Parámetro Experimental de Munin.

que a priori parece que está hecho para anotar en una gráfica texto con sentido. No tengo más datos sobre este parámetro.

4.2. Zabbix

5. ZABBIX ME FALTA

- Prueba a instalar este monitor es alguno de sus tres sistemas. Realice capturas de pantalla del proceso de instalación y comente capturas de pantalla del programa en ejecución.

Para instalar el monitor de zabbix he consultado los manuales de la página[13] para descargar este desde paquetes. Para empezar tenemos que añadir a la lista de repositorios el repositorio de este proyecto:

```
wget http://repo.zabbix.com/zabbix/3.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_3.0-1+trusty_all.deb
dpkg -i zabbix-release_3.0-1+trusty_all.deb
apt-get update
```

Tras lo cual podremos instalar los paquetes necesarios para el servidor de zabbix:

```
apt-get install zabbix-server-mysql zabbix-frontent-php zabbix-agent
```

Para tener zabbix inicializado con una base de datos ya creada, se nos facilita un ejemplo que podemos utilizar para comprobar que está funcionando. Para ello, ejecutamos el siguiente script que nos facilita la página:

```
shell>mysql -uroot -p<password>
mysql>create database zabbix character set utf8 collate utf8_bin;
mysql>grant all privileges on zabbix.* to zabbix@localhost identified by '<password>';
mysql>quit;
```

En el manual, parecen otros archivos como schema.sql, data.sql y images.sql que no los encontramos en el directorio donde nos indican. Esto es porque Ubuntu tiene preparado un script para ello:

```
/usr/share/doc/zabbix-server-mysql:
zcat create.sql.gz | mysql -uroot zabbix
```

Ahora, como cada vez que instalamos un servicio, debemos asegurarnos que está configurado como queremos, o como nos aconsejan que lo tengamos en la página web del proyecto. Para ello modificamos con nuestro editor favorito su archivo de configuración y descomentamos las siguientes líneas de opciones y añadimos lo necesario para que estas opciones queden en el archivo:

```
DBHost = localhost
DBName = zabbix
DBUser = zabbix
DBPassword = zabbix
```

tras lo cual, como debemos hacer cuando modificamos el archivo de configuración de cualquier servicio, lo reiniciamos:

```
service zabbix-server restart
```

El último retoque que debemos hacer para terminar de configurar zabbix es descomentar la línea de zona horario en apache2 y poner nuestra zona horaria. Por supuesto, tras esto, reiniciamos apache:

```
service apache2 restart
```

Ahora ya podremos entrar en zabbix desde cualquier navegador, en localhost/zabbix/setup.php:

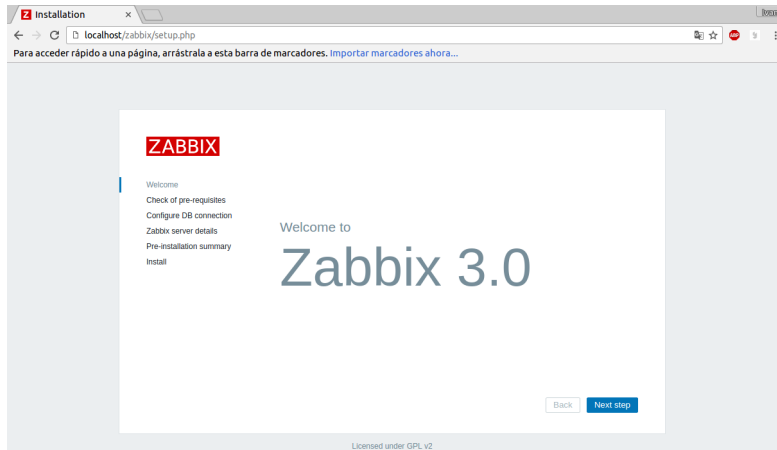


Figura 5.1: Ventana que aparece la primera vez que te conectas a zabbix.

Ahora nos mostrarán unas opciones de configuración que deberemos rellenar:

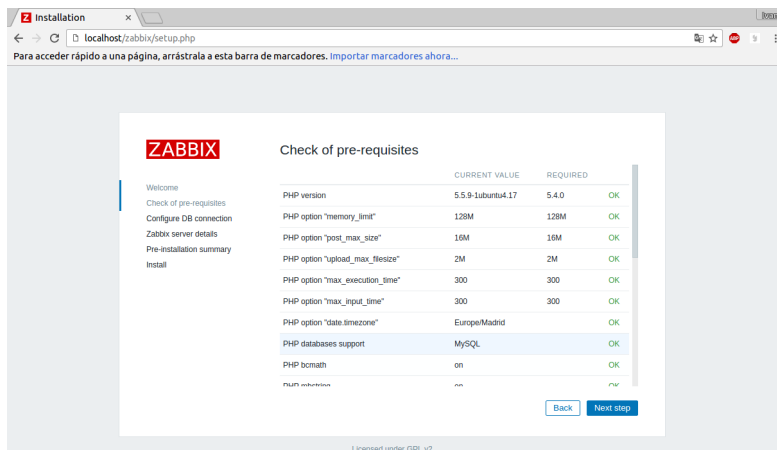


Figura 5.2: Opciones por defecto de zabbix.

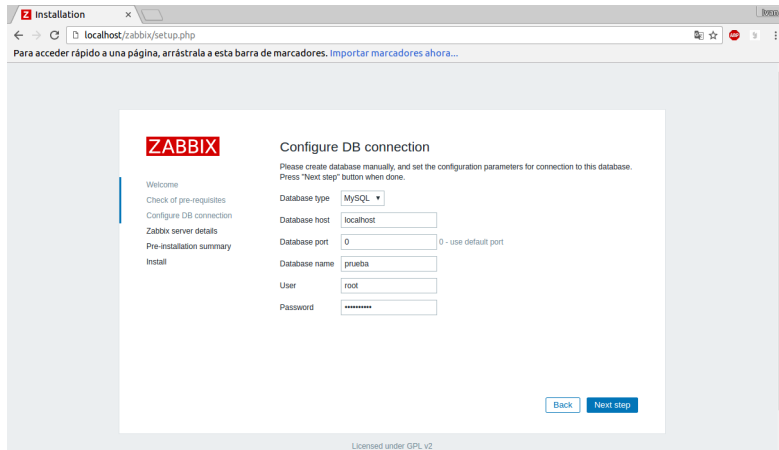


Figura 5.3: Usuario de mysql.

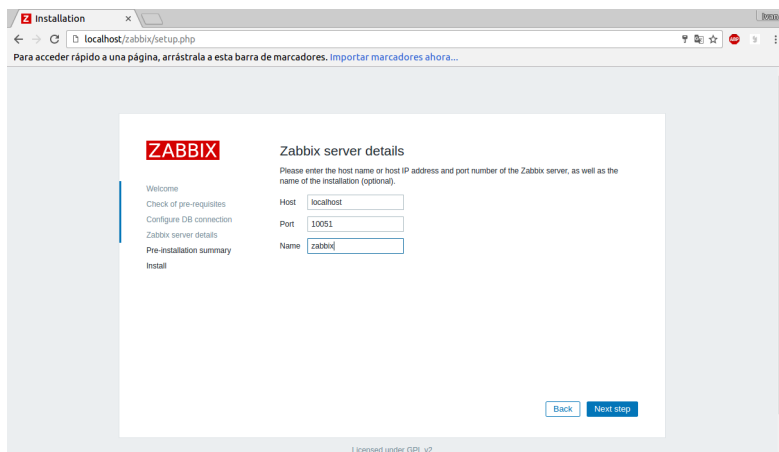


Figura 5.4: Usuario y contraseña usada para zabbix. Por defecto Admin/zabbix

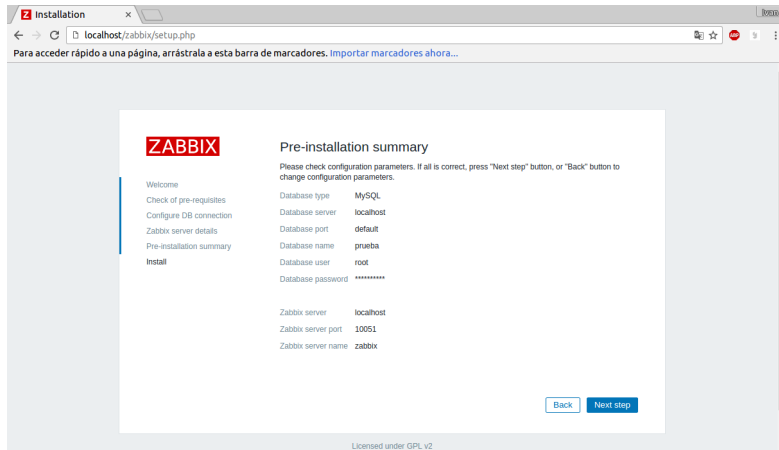


Figura 5.5: Sumario de la instalación.

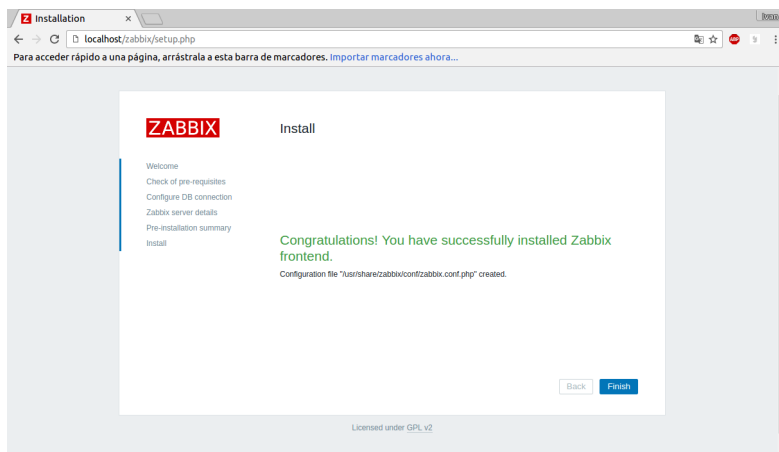


Figura 5.6: Configuración completada.

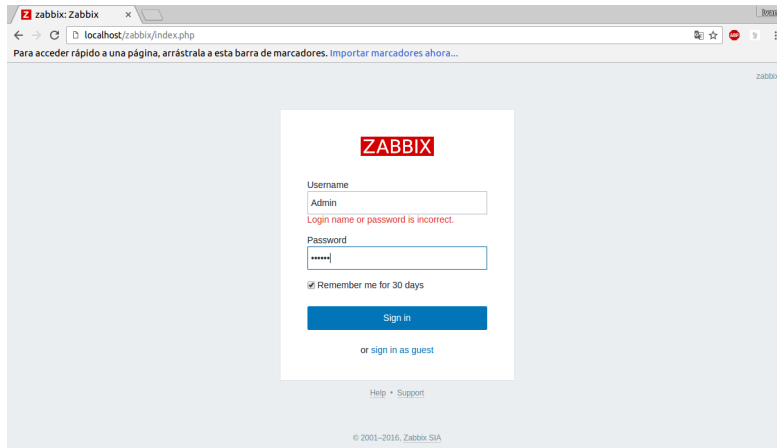


Figura 5.7: Entrar por primera vez en zabbix

Una vez nos registramos, nos aparece la siguiente interfaz:

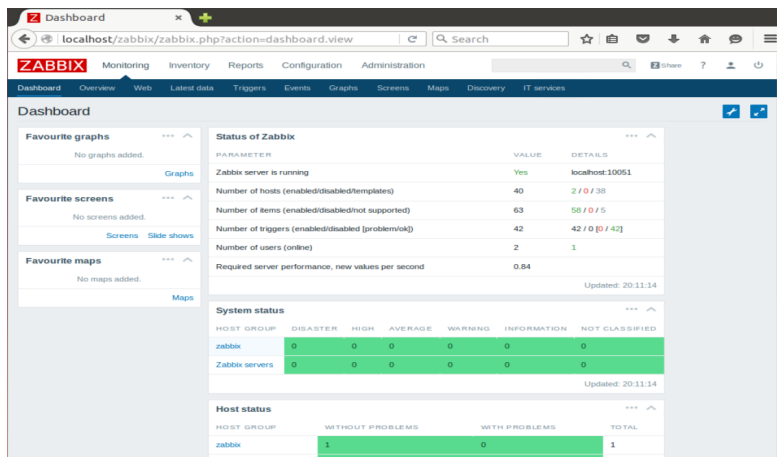


Figura 5.8: Interfaz general de zabbix.

Ahora, para monitorizar un servicio nos vamos a la sección Graphs y seleccionamos algún servicio. Por ejemplo:

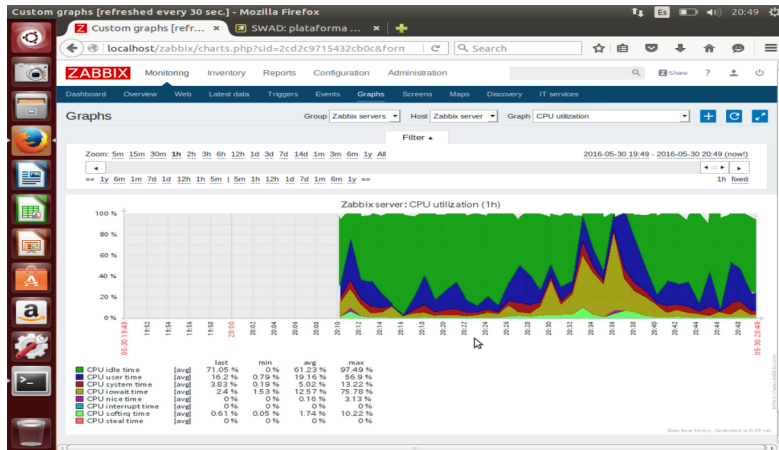


Figura 5.9: Gráfico que muestra el porcentaje de uso de la CPU de nuestro servidor.

Así por ejemplo vemos en verde el porcentaje de CPU no usado, en azul el usado por parte del usuario y en rojo, el porcentaje usado por el sistema en cada momento de la última hora(en realidad menos por ser inicializado hace menos de una hora).

5.1. Monitorizando un servicio

- **Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de strace, o busque otro programa similar y coméntelo.**

En mi caso, he visitado la página[10], en la que se habla de algunas utilidades de strace y de un problema que el escritor solucionó gracias a este monitor de servicios. Las primeras líneas nos explican que esta herramienta no solo puede monitorizar un servicio desde el principio (lanzándolo con strace), si no que también puede lanzarse aunque el servicio ya esté ejecutándose. Esto es muy útil a la hora de monitorizar daemons que no deban ser reiniciados muy a menudo. En cuanto al problema que plantea, se resume de la siguiente manera: Se está utilizando un servicio SVN que fallaba aleatoriamente y nadie podía acceder al mismo. Reiniciar el servidor arreglaba el problema, pero no permanentemente, volvía a fallar. Al escritor se le ocurrió buscar en los archivos log, pero no había ninguno. Además, a la hora de observar cómo se iniciaban los procesos, todo parecía correcto, pero por alguna razón, el sistema se colgaba. Entonces recurrió, puesto que no podía modificar el código con opciones de debugueo, a utilizar un script que lanzara svnserver pero cazando la información que nos pueda dar strace. De esta forma, los clientes podrían seguir utilizando el servicio. Este cambio hizo visible el problema fácilmente en cuanto varios usuarios utilizaron el servicio. El problema era que a la hora de dar *filehandle* para lectura, se daba el filehandle número 5, pero este fallaba.

Por lo visto, este problema ocurre cuando el sistema no tiene la suficiente entropía para generar un stream aleatorio, por lo que reiniciar el sistema podría solucionar parcialmente el problema(podría reiniciar"la entropía). El por qué de esta falta de entropía no se aclara ya que, según el autor, no era necesario. Utilizando, en vez de el generador de streams aleatorios */dev/random*, otro generador que no fallase con la falta de entropía, */dev/urandom*, solucionó fácilmente el problema. En principio esto es un fallo de seguridad, pero en el trabajo que estaban realizando y según el criterio del autor, este fallo era irrelevante.

La entrada del blog termina diciendo que este problema hubiese sido imposible de resolver sin un debugger o strace, siendo este último bastante más sencillo de utilizar.

6. Profiling

6.1. Python

- **Escriba un script en python y analice su comportamiento usando el profiler presentado.**

Utilizando cProfile[14] como profiler para analizar el comportamiento de el siguien-

te script:

```
1 import cProfile
2
3 def g(t,y):
4     w=list()
5     for i in range(t):
6         w.append(i)
7
8 def f(a):
9     z=0
10    for i in range(a):
11        z=g(i,z)
12
13    return z
14
15 cProfile.run('f(5000)')
```

Figura 6.1: Script de python sobre el que vamos a realizar el profiling.

nos da la siguiente salida:

```
ivan@Practica3$ python programa.py
12507504 function calls in 2.691 seconds

Ordered by: standard name

ncalls  tottime  percall  cuntime  percall  filename:lineno(function)
      1      0.000      0.000      2.691      2.691  <string>:1(<module>)
    5000      1.883      0.000      2.619      0.001  programa.py:3(g)
      1      0.072      0.072      2.691      2.691  programa.py:8(f)
12497500      0.673      0.000      0.673      0.000  {method 'append' of 'list' objects}
      1      0.000      0.000      0.000      0.000  {method 'disable' of '_lsprof.Profiler' objects}
    5001      0.062      0.000      0.062      0.000  {range}
```

Figura 6.2: Salida del script anterior.

en la cual vemos varias llamadas a cada una de las funciones que hemos creado junto con el tiempo que pasa el programa en total en cada función o en particular en cada llamada a cada función.

- Vemos que hay una función que se llama 5000 veces. Esta es la función g definida anteriormente. Esto es porque se ha llamado desde la función f 5000

veces a g. Por esta razón, vemos que el tiempo que el programa ha estado en esta función es de 1.883 segundos, pero tarda alrededor de 0.001 segundos por llamada.

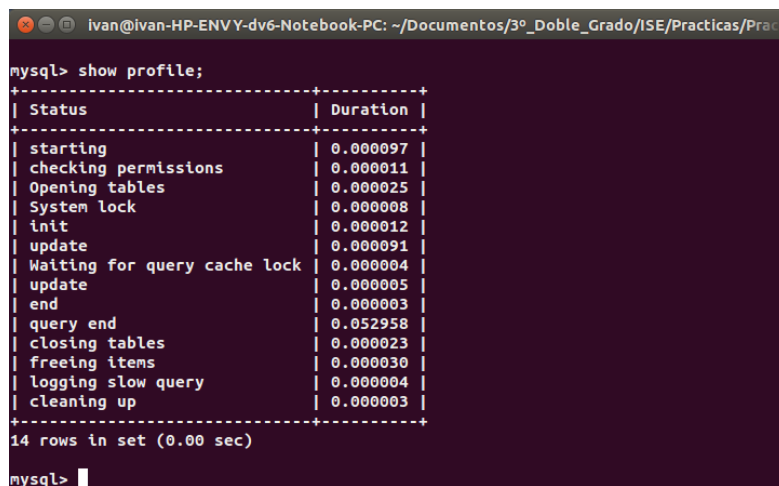
- La función f pasa dentro de la misma(no en subllamadas) alrededor de 0.072 segundos. Sin embargo, el tiempo total que el programa está en esta función y en subllamadas del mismo es 2.691 segundos.
- La función append de listas es llamada 124975 veces. Esto es porque en cada subllamada a la función g se llama muchas veces a la función append.

Estas son algunas de las anotaciones que se pueden hacer del profiling que hemos realizado del programa en python.

6.2. Bases de datos

- 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)

Tras observar cómo usar el comando SHOW PROFILE y SHOW PROFILES en mysql[11, 12], para los cuales es necesario cambiar a 1(true) la variable profiling para que estas puedan trabajar, y haber hecho la creación de dos bases de datos, la inserción de unos cuantos datos y unas cuantas consultas pertinentes, el resultado de ejecutar ambos comandos son los siguientes:



```
ivan@ivan-HP-ENVY-dv6-Notebook-PC: ~/Documentos/3º_Doble_Grado/ISE/Practicas/Prac
mysql> show profile;
+-----+-----+
| Status                               | Duration |
+-----+-----+
| starting                             | 0.000097 |
| checking permissions                 | 0.000011 |
| Opening tables                       | 0.000025 |
| System lock                         | 0.000008 |
| init                                | 0.000012 |
| update                              | 0.000091 |
| Waiting for query cache lock         | 0.000004 |
| update                              | 0.000005 |
| end                                  | 0.000003 |
| query end                           | 0.052958 |
| closing tables                      | 0.000023 |
| freeing items                       | 0.000030 |
| logging slow query                   | 0.000004 |
| cleaning up                         | 0.000003 |
+-----+-----+
14 rows in set (0.00 sec)

mysql>
```

Figura 6.3: Salida en mysql tras ejecutar SHOW PROFILE.

Con este comando podemos ver los tiempos de carga, inicio, comprobación de permisos, etc... que realiza la base de datos. Ahora veamos cuanto ha tardado la base de datos en concreto en realizar las consultas y la creación de tablas:

```
ivan@ivan-HP-ENVY-dv6-Notebook-PC: ~/Documentos/3º_Doble_Grado/ISE/Practicas/Prac
mysql> show profiles;
+-----+
| Query_ID | Duration | Query
+-----+
| 1 | 0.00006825 | select * from personas
| 2 | 0.04464625 | insert into personas values('Juan',85852598)
| 3 | 0.00032575 | select nombre from personas
| 4 | 0.12255200 | create table tabla2(entero int, nombre VARCHAR(50), fecha DATE)
| 5 | 0.00030900 | insert into tabla2 values(2,'siete')
| 6 | 0.06158150 | insert into tabla2 values(2,'siete',NULL)
| 7 | 0.03759200 | insert into tabla2 values(3,'siete',NULL)
| 8 | 0.04728825 | insert into tabla2 values(4,'siete',NULL)
| 9 | 0.05327300 | insert into tabla2 values(4,'dieciocho',NULL)
+-----+
9 rows in set (0.00 sec)

mysql>
```

Figura 6.4: Salida en mysql tras ejecutar SHOW PROFILES.

Aquí podemos ver los tiempos que ha tardado en cada una de las operaciones que le hemos ordenado que haga. En la misma, podemos ver que la creación de la tabla ha tardado alrededor de una décima de segundo y que en las inserciones, ha tardado alrededor de media décima. También vemos que al realizar mal una consulta, descartarla ha sido rápido.

Un dato curioso es que a la hora de mostrar la primera tabla completa ha tardado menos que en mostrar solo parte de la misma. Esto se debe a que a la hora de filtrar parte de la tabla se gasta más tiempo que en mostrar todo directamente.

Referencias

- [1] <http://linux.die.net/man/1/crontab>
- [2] <http://linux.die.net/man/8/hddtemp>
- [3] <http://www.tecmint.com/psensor-monitors-hardware-temperature-in-linux/>
- [4] <http://www.educadictos.com/herramientas-de-monitorizacion-de-hardware-para-ubuntu-12-04/>
- [5] <http://www.guia-ubuntu.com/index.php/HardInfo>
- [6] <http://hardzone.es/programas-para-testear-monitorizar-y-comprobar-el-rendimiento-de-tu-pc/>
- [7] <http://www.mersenne.org/>
- [8] <http://www.memtest86.com/>
- [9] <https://www.nagios.org/>
- [10] https://debian-administration.org/article/352/Using_strace_to_debug_application_errors
- [11] <http://dev.mysql.com/doc/refman/5.7/en/show-profile.html>
- [12] <http://dev.mysql.com/doc/refman/5.7/en/show-profiles.html>
- [13] https://www.zabbix.com/documentation/3.0/manual/installation/install_from_packages
- [14] <https://docs.python.org/2/library/profile.html>