

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

---

## Memoria Práctica 3

---

Marcos Avilés Luque

9 de diciembre de 2016

## Índice

<b>1 Cuestión 1.</b>	<b>4</b>
1.1 ¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes? . . . . .	4
1.2 ¿Qué significan las terminaciones .1.gz o .2.gz de los archivos en ese directorio? . . . . .	5
<b>2 Cuestión 2. ¿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.</b>	<b>6</b>
<b>3 Cuestión 3. 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.</b>	<b>7</b>
<b>4 Cuestión 4. Ejecute el monitor de “System Performance” y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.</b>	<b>9</b>
<b>5 Cuestión 5. Cree un recopilador de datos definido por el usuario (modo avanzado) que incluya tanto el contador de rendimiento como los datos de seguimiento.</b>	<b>11</b>
<b>6 Cuestión 6. Visite la web del proyecto y acceda a la demo que proporcionan (<a href="http://demo.munin-monitoring.org/">http://demo.munin-monitoring.org/</a>) donde se muestra cómo monitorizan un servidor.</b>	<b>19</b>
<b>7 Cuestión 7. Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de strace o busque otro y coméntelo.</b>	<b>21</b>
<b>8 Cuestión 8. Escriba un script en Python o PHP y analice su comportamiento usando el profiler presentado.</b>	<b>22</b>
<b>9 Cuestión 9. 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).</b>	<b>26</b>

## Índice de figuras

1.1. Listado de paquetes instalados en el sistema Ubuntu Server. . . . .	4
1.2. Historial de paquetes que hemos instalado en el sistema Ubuntu Server. . . . .	5
1.3. Listados de archivos contenidos en el directorio '/var/log/apt/'. . . . .	5
2.1. Archivo 'crontab' de nuestro usuario root en nuestro sistema Ubuntu Server. . . . .	6
2.2. Modificando el archivo 'crontab' para añadir una tarea programada. . . . .	7
3.1. Ejecución comando 'dmesg' antes de conectar el USB. . . . .	8

3.2.	Paso que debemos realizar para que la máquina virtual, en este caso Ubuntu Server, reconozca el USB conectado a través de Virtualbox. . . . .	8
3.3.	Ejecución del comando 'dmesg' después de conectar el USB. . . . .	8
4.1.	Primer informe de 'System Performance' sin ejecuciones extras. . . . .	9
4.2.	Segundo informe de 'System Performance' ejecutando en segundo plano el navegador explorer. . . . .	10
4.3.	Gráfico correspondiente al informe de la Figura 4.1. . . . .	10
4.4.	Gráfico correspondiente al informe de la Figura 4.2. . . . .	11
5.1.	Creando un nuevo conjunto de recopiladores de datos en nuestro Sistema Windows Server. . . . .	12
5.2.	Creando un nuevo conjunto de recopiladores de datos en nuestro Sistema Windows Server. . . . .	13
5.3.	Añadiendo el registro de contador de rendimiento y el registro de datos de seguimiento de eventos a nuestro conjunto. . . . .	13
5.4.	Proceso para elegir los contadores de nuestro conjunto de recopilador. . .	14
5.5.	Eligiendo las instancias de cada contador. . . . .	14
5.6.	Todos los contadores de rendimiento elegidos para nuestro ejemplo. . . .	15
5.7.	Especificando la ruta donde queremos que se almacenen los datos cuando ejecutemos nuestro conjunto. . . . .	15
5.8.	Guardando y terminando el asistente de la creación de un conjunto de recopiladores de datos. . . . .	16
5.9.	Ejecutando el conjunto de recopiladores de datos creado anteriormente. . .	17
5.10.	Informe obtenido de la ejecución de nuestro conjunto de recopiladores de datos. . . . .	17
5.11.	Filtrado de algunas características del conjunto de recopilador de datos. .	18
6.1.	Información gráfica sobre el uso de CPU de una demo de MUNIN. . . . .	20
6.2.	Información gráfica sobre las hebras de una demo de MUNIN, también se le decir procesos. . . . .	21
8.1.	Instalación del paquete 'php5-dev' en nuestro sistema Ubuntu Server. . . .	22
8.2.	Instalación del paquete 'Xdebug' en nuestro sistema Ubuntu Server. . . .	23
8.3.	Añadiendo la herramienta 'xdebug' a nuestro servicio 'apache'. . . . .	23
8.4.	Herramienta 'Xdebug' añadida a nuestro servidor PHP. . . . .	24
8.5.	Archivos creados cada vez que ejecutamos un script PHP, cuando tenemos habilitado el profiler que estamos demostrando. . . . .	25
8.6.	Profiler 'KCacheGrind' mostrando los resultados de la ejecución del script PHP 'phpinfo.php'. . . . .	25
9.1.	Inicio de sesión en MySQL e inicializamos la variable de sesión del profiler. .	26
9.2.	Creando una base de datos, una nueva tabla, y añadiendo valores a la tabla de base de datos MySQL. . . . .	26
9.3.	Tabla de tiempos de las consultas realizadas. . . . .	27
9.4.	Tabla de tiempos de la CPU para la consulta elegida. . . . .	27

## 1. Cuestión 1.

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

Hay varias formas de analizar los paquetes instalados en nuestro sistema Ubuntu Server, yo he elegido el gestor de paquetes 'dpkg' [3] que cuenta con un archivo ubicado en '/var/log/dpkg.log' que contiene un registro de todos paquetes que se han instalado en el sistema.

Consultamos el man de 'dpkg' [9], y con la orden 'dpkg --get-selections' obtenemos una lista de los paquetes instalados en nuestro sistema como vemos en la Figura 1.1.

```
ufu                                install
unattended-upgrades              install
update-manager-core              install
update-notifier-common           install
upstart                          install
ureadahead                       install
usbutils                         install
util-linux                      install
uuid-runtime                     install
vim                              install
vim-common                       install
vim-runtime                      install
vim-tiny                        install
v2n                              install
watershed                       install
wechmin                          install
wget                             install
whiptail                        install
wireless-regdb                  install
wireless-tools                  install
wpasupplicant                   install
x11-common                      install
xauth                           install
xkb-data                        install
xul-core                         install
xul-ext-ubufox                  install
xz-utils                        install
yelp                             install
yelp-xsl                        install
yjs                              install
zeitgeist                       install
zeitgeist-core                  install
zeitgeist-database              install
zenity                           install
zenity-common                   install
zlib1g:amd64                    install
root@ubuntu:~# dpkg --get-selections'
```

Figura 1.1: Listado de paquetes instalados en el sistema Ubuntu Server.

Si elegimos el gestor de paquetes 'apt' ocurre lo mismo que 'dpkg' pero en este caso el directorio es '/var/log/apt/' y nos encontramos con el archivo 'history.log', que contiene todo el historial de la gestión de paquetes de nuestro sistema, es decir, la instalación, modificación o eliminación de los mismos, como podemos ver en la Figura 1.2.

```

Start-Date: 2016-11-24 23:16:37
Commandline: apt install phpmyadmin
Install: libjs-jquery-nousewheel:amd64 (8-2, automatic), libjs-jquery-metadata:amd64 (8-2, automatic), libjs-jquery-cookie:amd64 (8-2, automatic), libjs-jquery-event-drag:amd64 (8-2, automatic), dbcon
fig-common:amd64 (1.0.42+nmu1, automatic), php5-script:amd64 (5.4.6-2ubuntu5, automatic), libcrypt4
:amd64 (2.5.8-3.1ubuntu1, automatic), phpmyadmin:amd64 (4.0.10-1), libjs-jquery:amd64 (1.7.2+dfsg-2u
buntu1, automatic), libjs-jquery-ui:amd64 (1.10.1+dfsg-1, automatic), libjs-underscore:amd64 (1.4.4-
2ubuntu1, automatic), libjs-jquery-tablesorter:amd64 (8-2, automatic), php-gettext:amd64 (1.0.11-1,
automatic), libjs-codemirror:amd64 (2.23-1, automatic), javascript-common:amd64 (11, automatic), php
5-gd:amd64 (5.5.9+dfsg-1ubuntu4.20, automatic)
End-Date: 2016-11-24 23:17:23

Start-Date: 2016-11-25 11:09:36
Commandline: apt install php-pear
Install: php-pear:amd64 (5.5.9+dfsg-1ubuntu4.20)
End-Date: 2016-11-25 11:09:37

Start-Date: 2016-11-25 11:13:17
Commandline: apt remove phpmyadmin
Remove: phpmyadmin:amd64 (4.0.10-1)
End-Date: 2016-11-25 11:13:35

Start-Date: 2016-11-25 11:13:55
Commandline: apt install phpmyadmin
Install: phpmyadmin:amd64 (4.0.10-1)
End-Date: 2016-11-25 11:13:59

Start-Date: 2016-11-25 11:15:34
Commandline: apt install php5
Install: php5:amd64 (5.5.9+dfsg-1ubuntu4.20)
End-Date: 2016-11-25 11:15:34

Start-Date: 2016-11-25 11:16:59
Commandline: apt install php5-cgi
Install: php5-cgi:amd64 (5.5.9+dfsg-1ubuntu4.20)
End-Date: 2016-11-25 11:17:02
root@ubuntu:~# cat /var/log/apt/history.log

```

Figura 1.2: Historial de paquetes que hemos instalado en el sistema Ubuntu Server.

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

Tanto con el gestor 'dpkg' como 'apt' podemos encontrar en su directorio, varias versiones de los historiales de paquetes, es decir, .1.gz, .2.gz, ...etc. Esto quiere decir que se va creando un archivo cada mes, con el historial correspondiente que se realice en ese periodo de tiempo.

Esto se realiza simplemente porque si se instalan, modifican o se eliminan muchos paquetes diarios, este historial sería interminable, por lo tanto, se crea esta clasificación para obtener la información mas rápida y cómodamente.

En la Figura 1.3 vemos el directorio donde se deben encontrar estos archivos anteriores:

```

root@ubuntu:~# ls -la /var/log/apt/
total 12
-rw-r--r-- 1 root root 1024 Nov 28 2016 history.log
-rw-r--r-- 1 root root 1024 Nov 28 2016 term.log
root@ubuntu:~#

```

Figura 1.3: Listados de archivos contenidos en el directorio '/var/log/apt/'.

En mi caso, aún no cuento con ningún archivo, por ejemplo, 'history.log.1.gz' que sería el primer archivo que se hubiera creado.

Un dato muy importante, siempre el historial más actual lo encontraremos en 'history.log', ya que cuando sea necesario la creación de un nuevo archivo, se copia la información actual y se renombrará todos los archivos existentes. Quedando la información mas actual en el primer archivo 'history.log' y la información mas antigua quedará en el nombre de archivo que contenga el número mayor de su identificador.

## 2. Cuestión 2. ¿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.

Como nos resumen un poco el guión y vemos en [4], cron es un demonio que se ejecuta normalmente al arranque del sistema, y una vez iniciado ejecuta las tareas que tenga asociadas.

Para programar una tarea y que cron la ejecute cada periodo de tiempo indicado tenemos varias formas:

- Tenemos la posibilidad de modificar el archivo 'crontab' con la opción '-e', ejecutando la orden 'crontab -e' y añadir la línea correspondiente de la tarea que queremos que se ejecute.
- Otra forma que nos indica en la referencia anterior, es añadir directamente los scripts en el directorio correspondiente de cron, por ejemplo, si añadimos nuestro script en '/etc/cron.daily/', nuestro script se ejecutará una vez por día. Podemos ver los 4 directorios disponibles para más información en la referencia mencionada.
- El usuario root contiene el archivo 'crontab' en el directorio '/etc/crontab', que es el archivo que ejecutará cron cuando sea iniciado, y evaluará las tareas que incluyan este archivo, vemos un ejemplo de nuestro crontab del sistema Ubuntu Server en la Figura 2.1.

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.
#
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
marcos@ubuntu ni6 nov 30 2016 :~$ _
```

Figura 2.1: Archivo 'crontab' de nuestro usuario root en nuestro sistema Ubuntu Server.

Por ello vamos a incluir la línea correspondiente a programar una tarea que se realice una vez al día una copia del directorio /codigo a /seguridad/\$fecha. En [5] vemos el

formato que debemos incluir en el archivo crontab, indicando los valores en tiempo para ejecutar nuestra tarea y añadimos:

```
- 00 12 * * * root cp -r /root/codigo /root/seguridad/"$(date)"
```

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
12 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* * * * * root    cp -r /root/codigo/ /root/seguridad/
#daily marcos cp ~/codigo ~/seguridad/"$(date)"
```

Figura 2.2: Modificando el archivo 'crontab' para añadir una tarea programada.

### 3. Cuestión 3. 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.

Como incluye el guión y podemos consultar [6], el comando 'dmesg' nos ofrece toda la información del trabajo del kernel, es decir, el kernel es el núcleo que controla todo el sistema a partir del arranque del mismo, y unas de las cosas que tiene definidas es la identificación y control de todos los dispositivos hardware del sistema, que nos servirá para el objetivo de esta cuestión.

Por tanto 'dmesg' se encarga de ir recopilando los mensajes producidos por el kernel y los va almacenando, lo que nos facilita comprobar el estado o funcionamiento de un dispositivo concreto, o incluso cualquier fallo ocurrido.

Vamos a demostrar como nos indica la cuestión, conectando un USB y ver que información nos ofrece 'dmesg'.

Primero vamos a comprobar antes de conectar el USB, que información tenemos disponible del kernel con la orden 'dmesg | tail' que nos llevara a la última información obtenida del kernel. Como vemos en la Figura 3.1, la última información obtenida que son las últimas líneas que encontramos en dicha figura, concretamente corresponde al inicio del sistema cuando el kernel ha habilitado la tarjeta de red 'eth0'.

```

marcos@ubuntu ~$ sudo dmesg | tail
( 2557.465946) usb 1-1: new full-speed USB device number 4 using ohci-pci
( 2557.722366) usb 1-1: New USB device found, idVendor=08ec, idProduct=0021
( 2557.722367) usb 1-1: Product USB device strings: Mfr=1, Product=3, SerialNumber=0
( 2557.722369) usb 1-1: Product: USB Tablet
( 2557.722374) usb 1-1: Manufacturer: VirtualBox
( 2557.759776) input: VirtualBox USB Tablet as /dev/input/l1000:00:0000:00:06.0usb/l1-1/l1-1.000
3:80EE:0021.0003/input:input5
( 2557.873941) hid-generic 0003:80EE:0021.0003: input,hidraw0: USB HID v1.10 Mouse [VirtualBox USB Tablet] on usb-0000:00:06.0-l1000
( 2558.194049) e1000: eth0 NIC Link is Down
( 2562.198314) e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
( 2563.162338) e1000: eth1 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
marcos@ubuntu ~$ sudo dmesg | tail

```

Figura 3.1: Ejecución comando 'dmesg' antes de conectar el USB.

Tengo que destacar que debemos realizar un proceso intermedio, ya que como estoy trabajando con máquina virtual 'VirtualBox' la conexión de USB o reconocimiento del mismo trabaja en función de la máquina anfitriona, por tanto debemos pasarle el control del USB desde la máquina anfitriona a la máquina virtual. Por ello tan sólo una vez ya conectado el USB, realizamos el paso que vemos en la Figura 3.2.

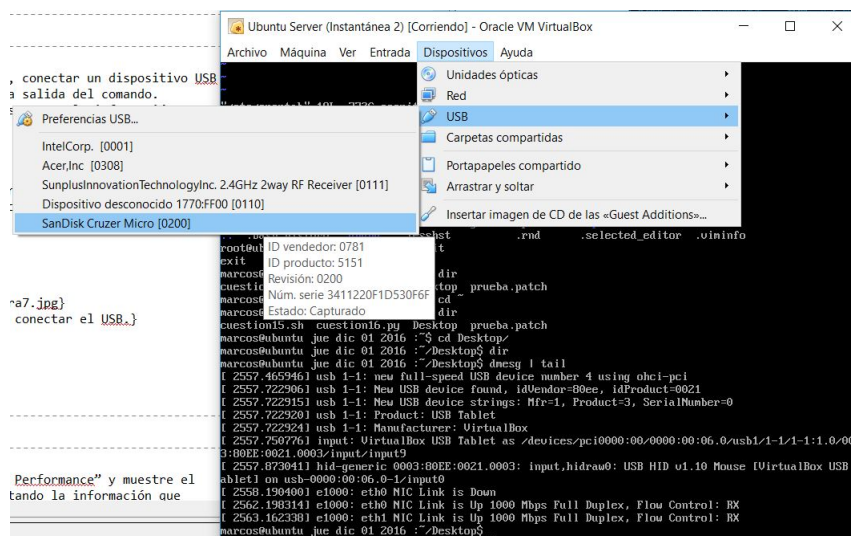


Figura 3.2: Paso que debemos realizar para que la máquina virtual, en este caso Ubuntu Server, reconozca el USB conectado a través de Virtualbox.

Volvemos a ejecutar la orden `'dmesg | tail'` y como vemos en la Figura 3.3, ya el kernel ha reconocido el USB, así como su tamaño, si tiene protección contra escritura..., y realiza el montaje de dicho USB. Lo vemos en las últimas líneas de la Figura 3.3.

```
marcos@ubuntu:~$ sudo fdisk -l
[ 3296.041787] sd z:2:0:0:0: [sdb] No Caching mode page found
[ 3296.044954] sd z:2:0:0:0: [sdb] Assuming drive cache: write through
msg: I fail
[ 3295.962677] usbcore: registered new interface driver us
[ 3295.991513] sscsi z:2:0:0: Direct-Access SanDisk Cruzer Micro 8.01 PQ: 0 ANSI: 0 CCS
[ 3295.995246] sd z:2:0:0: Attached scsi generic sg2 type 0
[ 3296.015135] sd z:2:0:0: [sdb] 7956127 512-byte logical blocks: (4.02 GB/3.75 GiB)
[ 3296.016466] sd z:2:0:0: [sdb] Write Protect: is on
[ 3296.024652] sd z:2:0:0: [sdb] Mode Sense: 45 00 00 00
[ 3296.041787] sd z:2:0:0: [sdb] No Caching mode page found
[ 3296.044954] sd z:2:0:0: [sdb] Assuming drive cache: write through
[ 3296.245337] sdb: sd1
[ 3296.362402] sd z:2:0:0: [sdb] Attached SCSI removable disk
marcos@ubuntu:~$ sudo fdisk -l
```

Figura 3.3: Ejecución del comando 'dmesg' después de conectar el USB.



#### 4. Cuestión 4. Ejecute el monitor de “System Performance” y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.

Siguiendo el guión o como nos indica [10], he realizado dos informes, ya que el primer informe que realicé fué sin ejecutar ninguna tarea Figura 4.1, y el segundo lo realicé mientras ejecutaba el navegador explorer para que afectara al rendimiento del Windows Server Figura 4.2, veamos los resultados:

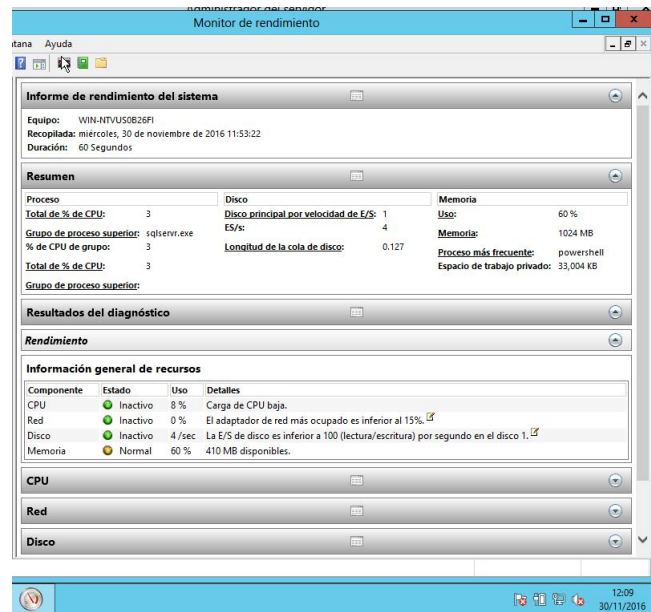


Figura 4.1: Primer informe de 'System Performance' sin ejecuciones extras.

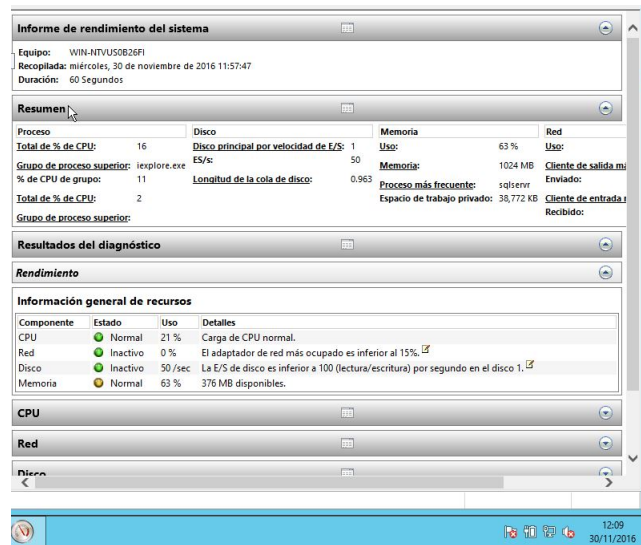


Figura 4.2: Segundo informe de 'System Performance' ejecutando en segundo plano el navegador explorer.

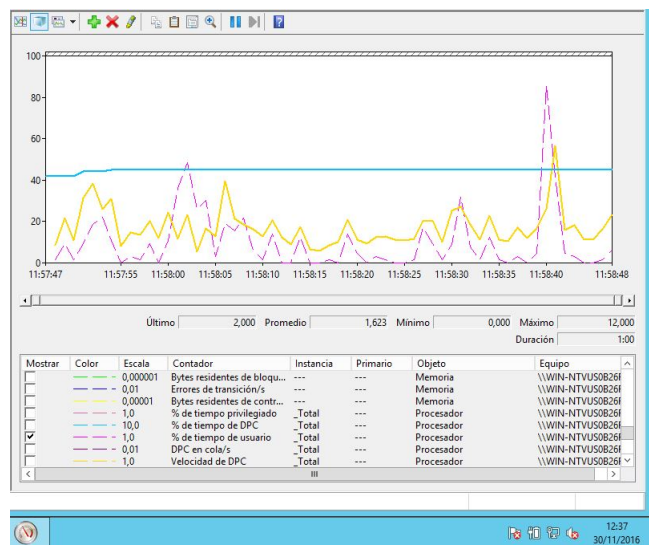


Figura 4.3: Gráfico correspondiente al informe de la Figura 4.1.

Referente a la Figura 4.3 y la Figura 4.4, hemos filtrado todos los datos obtenidos y he elegido tres características a su respectivo informe:

- La línea azul, que nos muestra los procesos del sistema
- La línea amarilla, que nos indica los cambios de contexto realizados por el sistema.
- Y la línea discontinua rosa que nos da la información del tiempo de procesador que ha utilizado el usuario.

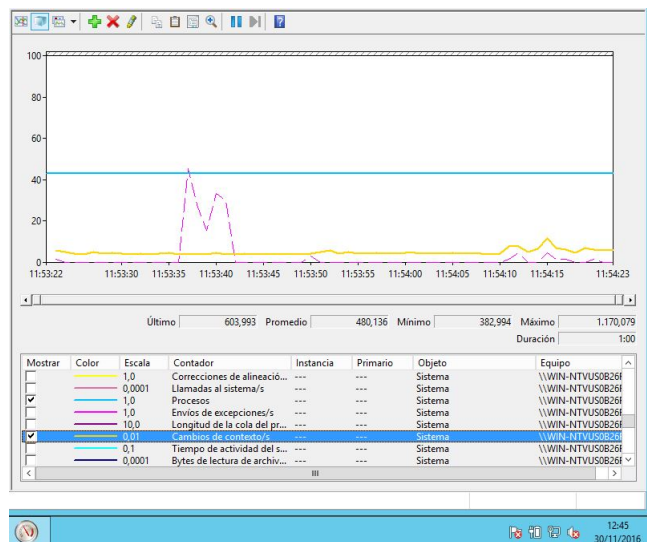


Figura 4.4: Gráfico correspondiente al informe de la Figura 4.2.

Como conclusión obtenemos la diferencia lógica y esperada de los resultados. Comparamos los gráficos de la Figura 4.3 y la Figura 4.4:

- La línea azul que nos indicaba los procesos del sistema, vemos que en primer gráfico es constante, aparte de los procesos básicos del sistema no se ejecutaba nada más, en cambio en el segundo gráfico vemos entre el segundo 47 y el segundo 55 del minuto 57 que se incrementan la ejecución de algunos procesos, que en este caso nos coincide con la ejecución del navegador.
- La línea amarilla y la línea rosa son muy similares ya que coinciden de cierta manera los cambios de contexto con el tiempo de procesador usado por el usuario, de cierta manera tiene su relación, ya que cuando el usuario ejecuta un proceso, el sistema debe de realizar cambios de contexto para poder llevar a cabo las tareas del usuario y sus propias tareas de sistema. Después de esto, volviendo a los resultados vemos como los picos más altos de ambas líneas las encontramos entorno al segundo 03 del minuto 58, y en el segundo 39 del minuto 58. El primer pico se debe a que solicité una búsqueda de una página de internet, y el segundo pico fué cuando solicité la pagina buscada. Este segundo pico es mayor debido a la carga de todo el contenido de dicha página.

## 5. Cuestión 5. Cree un recopilador de datos definido por el usuario (modo avanzado) que incluya tanto el contador de rendimiento como los datos de seguimiento.

Según nos dice [2], un recopilador de datos podemos decir que es un conjunto de contadores que podemos elegir para la monitorización de nuestro sistema y nos ofrece un sólo

informe de los datos obtenidos de los valores elegidos.

Siguiendo [15] creamos un recopilador de datos. Primero nos situamos en el monitor de rendimiento y definimos un nuevo conjunto de recopiladores de datos como vemos en la Figura 5.1.

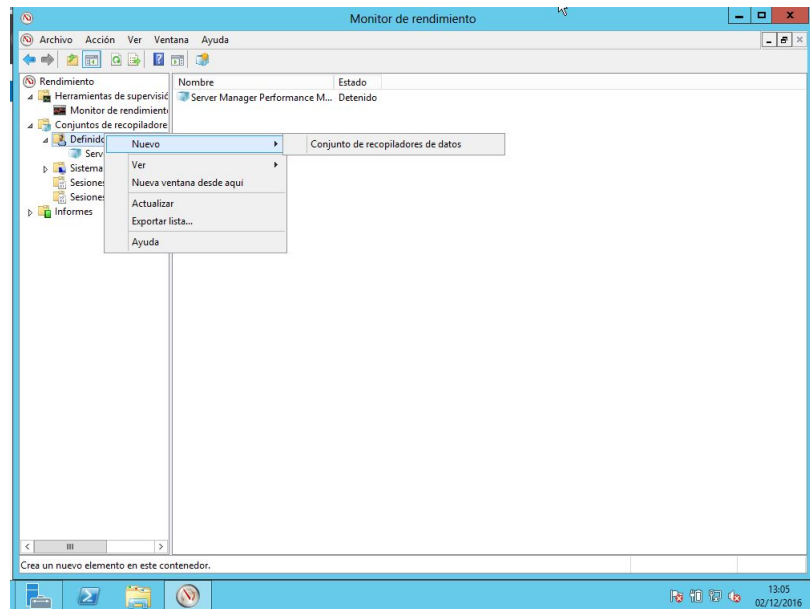


Figura 5.1: Creando un nuevo conjunto de recopiladores de datos en nuestro Sistema Windows Server.

El siguiente paso definimos un nombre para nuestro conjunto y elegimos la opción crear manualmente como vemos en la Figura 5.2, para poder incluir las características a nuestra medida.

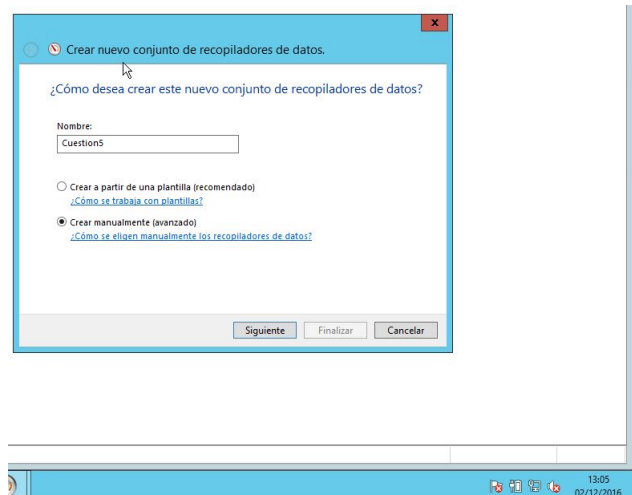


Figura 5.2: Creando un nuevo conjunto de recopiladores de datos en nuestro Sistema Windows Server.

Ahora continuamos eligiendo la opción crear registros de datos, y seleccionamos los registros de datos que queremos incluir, como en este caso, contador de rendimiento y datos de seguimiento de eventos, nos fijamos en la Figura 5.3.

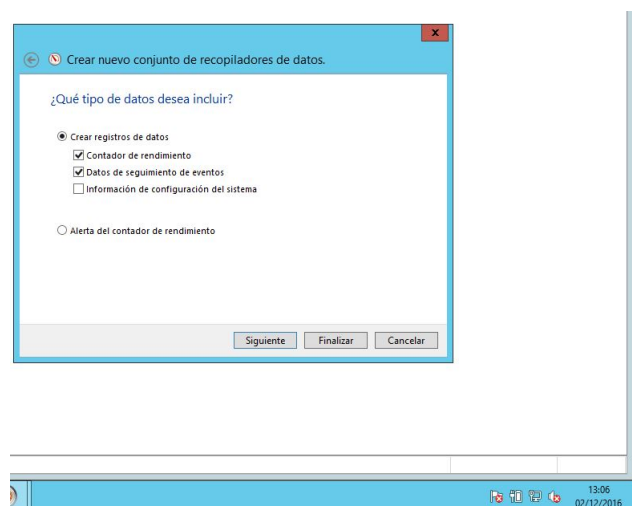


Figura 5.3: Añadiendo el registro de contador de rendimiento y el registro de datos de seguimiento de eventos a nuestro conjunto.

En el siguiente paso, vamos a elegir algo importante, como en realidad sobre qué queremos obtener los datos, dicho de otra manera, qué contadores de rendimiento vamos a monitorizar. Como vemos en la Figura 5.4, pinchamos sobre agregar.

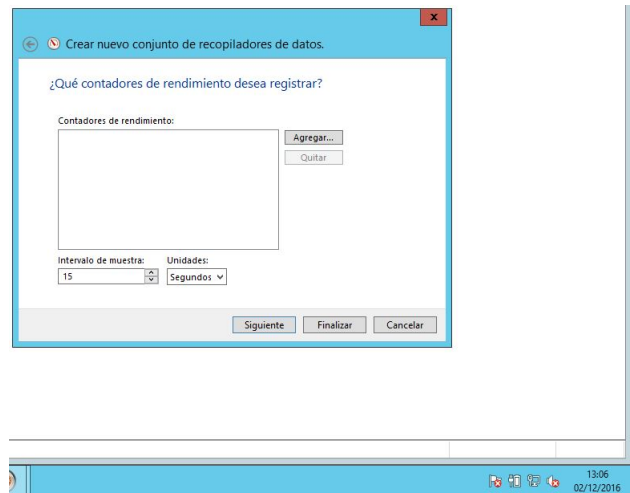


Figura 5.4: Proceso para elegir los contadores de nuestro conjunto de recopilador.

Ahora ya si debemos de buscar entre todos los contadores disponibles, los contadores que estamos interesados, en nuestro caso vamos añadir todo lo relacionado con procesador, proceso y servicio web.

Un contador puede incluir varias instancias, podemos decir que muchísimas, y podríamos elegir entre todas ellas una instancia específica, pero en nuestro caso vamos a seleccionar todas las instancias que tienen disponible los contadores que hemos elegido.

Así que primero, buscamos el contador, elegimos todas las instancias y la agregamos como vemos en la Figura 5.5.

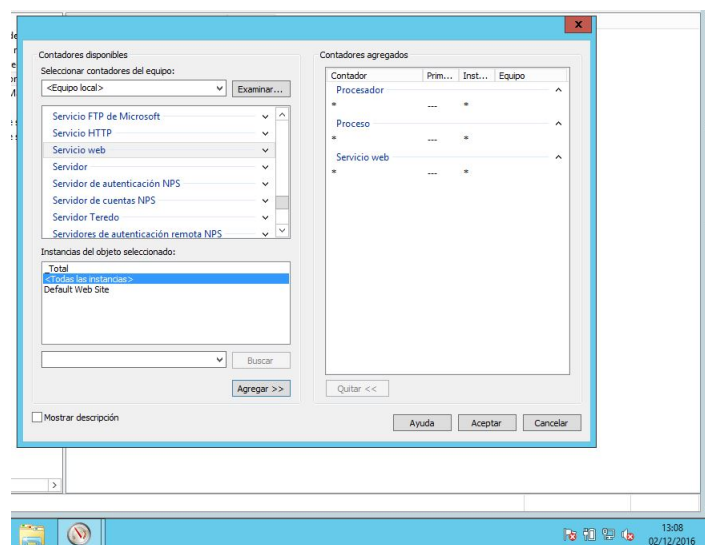


Figura 5.5: Eliendo las instancias de cada contador.

Una vez todo elegido obtendremos una cosa así como la Figura 5.6. Dejamos el intervalo de muestra en 15 segundos, como viene por defecto y le damos a siguiente.

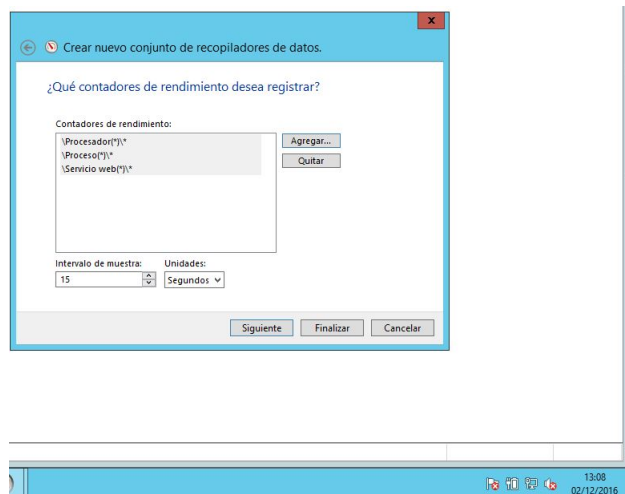


Figura 5.6: Todos los contadores de rendimiento elegidos para nuestro ejemplo.

Ya sólo nos falta especificar la ruta donde queremos guardar los datos, esto no quiere decir que nuestro conjunto de recopilador de datos se va a guardar en esta ruta, sino que cuando ejecutemos el conjunto los datos recogidos se almacenarán en la ruta especificada. Cambiamos la ruta como nos indica el guión y obtenemos la Figura 5.7.

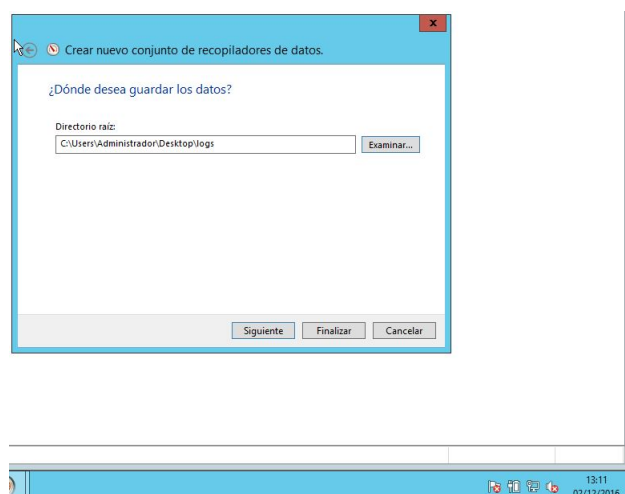


Figura 5.7: Especificando la ruta donde queremos que se almacenen los datos cuando ejecutemos nuestro conjunto.

Ya por último el asistente nos pregunta si deseamos ejecutar en este momento el conjunto, o guardarlo..., vamos a elegir la opción guardar y cerrar. Y finalizamos el asistente. Nos

fijamos en la Figura

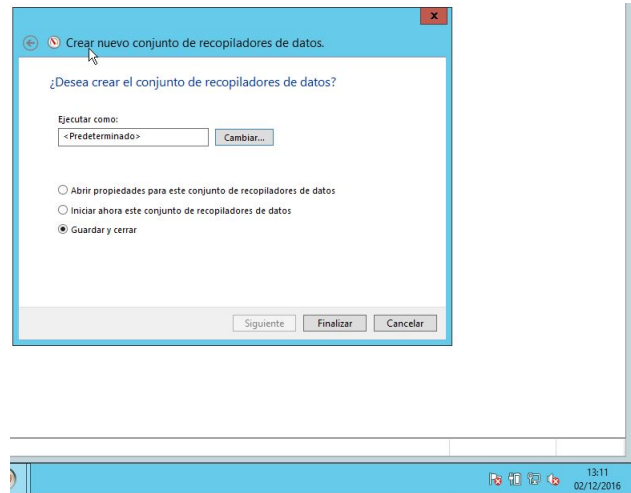


Figura 5.8: Guardando y terminando el asistente de la creación de un conjunto de recopiladores de datos.

Ya tenemos nuestro conjunto de recopiladores de datos creado. El guión solo habla de como crear este conjunto anterior, pero ¿cómo se ejecuta?, ¿funciona realmente?, ¿qué datos o gráfico obtenemos realmente?. Vamos a ver un ejemplo.

Ya nos debe aparecer nuestro conjunto como vemos en la Figura 5.9, lo voy a ejecutar durante 1 minuto y 30 segundo aprox.



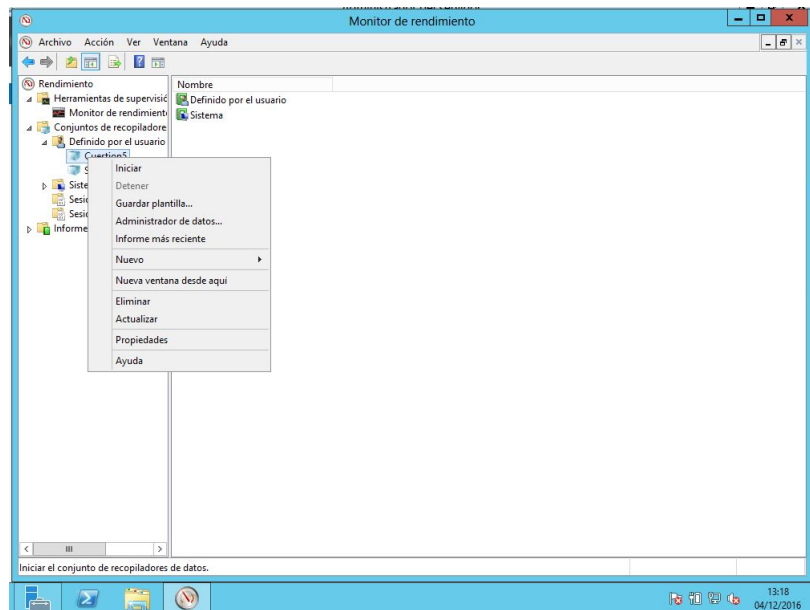


Figura 5.9: Ejecutando el conjunto de recopiladores de datos creado anteriormente.

Cuando lo paramos, debe de haberse creado nuestro informe correspondiente a la ejecución anterior del conjunto como vemos en la Figura 5.10.

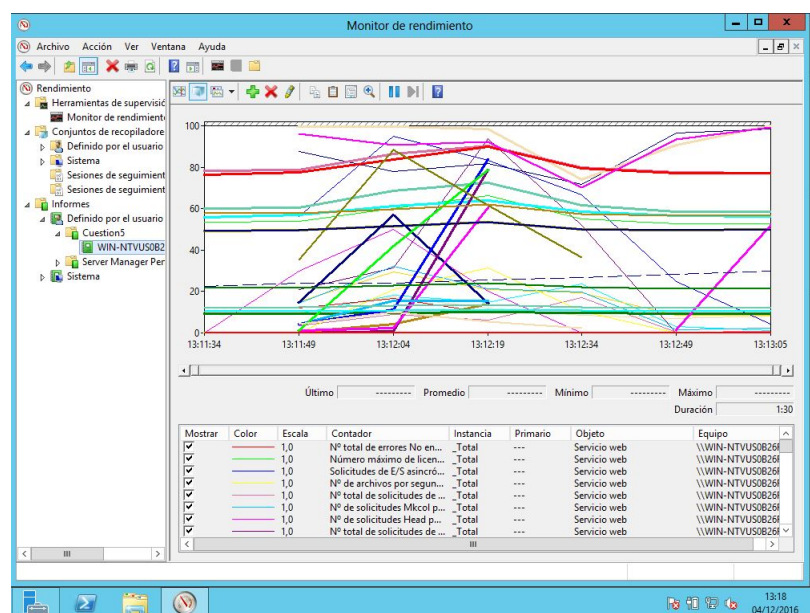


Figura 5.10: Informe obtenido de la ejecución de nuestro conjunto de recopiladores de datos.

Como vemos hay muchísima información vamos a quedarnos con al menos tres datos y vamos a comentarlos. Voy a elegir las siguientes características como vemos en la Figura 5.11:

- La línea azul oscuro, que pertenece a nuestro servidor web y nos va a mostrar el número total de archivos enviados.
- La línea azul claro, que pertenece a nuestro control de procesos y nos va a mostrar el número de subprocesos que ejecuta nuestro sistema.
- La línea rosa, que pertenece al conjunto de las instancias del procesador, concretamente al tiempo de procesador ejecutado por el usuario.

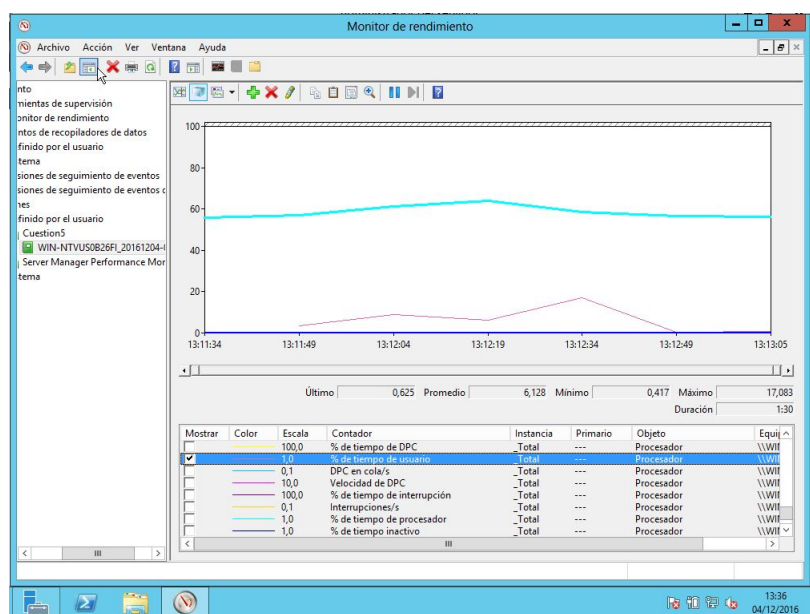


Figura 5.11: Filtrado de algunas características del conjunto de recopilador de datos.

Como vemos en la Figura 5.11, observamos:

- La línea azul oscuro, es constante en 0, por lo que no envía ningún archivo, esto es debido a que nuestro servidor no ha recibido ningún trabajo solicitando algún fichero, o también puede ser debido a que nuestro servidor web no este funcionando correctamente.
- La línea azul claro, nos indica que a partir del minuto 11 y 49 segundos comienza el inicio de la ejecución de mas procesos, corresponde cuando estaba realizando el conjunto, inicie varios navegadores para que el monitor se viera afectado.

- La línea rosa, nos verifica la línea anterior, que comienza justo cuando inicié varios procesos de navegadores y a partir de ahí comienza el uso del procesador por parte del usuario.

## 6. Cuestión 6. Visite la web del proyecto y acceda a la demo que proporcionan (<http://demo.munin-monitoring.org/>) donde se muestra cómo monitorizan un servidor.

En [12] sabemos que 'MUNIN' es una herramienta de monitorización de rendimiento como por ejemplo, el rendimiento de red, aplicaciones, del propio sistema, mediciones meteorológicas..., que incluso nos facilita el monitorizar varias máquinas paralelamente. Una de las cosas que más me ha resultado bastante útil, es que podemos instalar un monitor en una máquina o varias, y podemos obtener los resultados desde cualquier otra máquina remotamente.

Como nos indica el guión vamos a centrarnos en varios ejemplos que podemos encontrar en [13] de 'MUNIN', que nos ofrece bastantes ejemplos de monitorizaciones de servidores realizados.

Por ejemplo vemos un ejemplo del uso de CPU en la Figura 6.1, en este caso los datos evaluados son sobre las muestras recogidas durante un año, he elegido esta gráfica para que los datos sean mas significantes, ya que comparándolos sobre diariamente o semanalmente, apenas se podían apreciar los datos recogidos.

Una característica importante que puedo ver son las 'franjas blancas' que se encuentran a finales de Noviembre, a principios de Enero, finales de Febrero y a principios de Junio, que son caídas del servidor debido a reparaciones o modificaciones de mejora.

Efectivamente podemos ver que posteriormente a cada modificación del servidor, se aprecia una mejora en el uso de la CPU, destacando sobretodo en la actualización de principios de Junio, en la que se visualiza una buena mejora del rendimiento de la CPU durante el resto de año.

- La zona azul, nos indica el uso de CPU por el usuario, y como vemos en la gráfica, el usuario ha utilizado más la CPU sobre el mes de Enero y Febrero, es decir, su pico más elevado sobrepasando el 20 %. Durante los siguientes meses hasta Julio aprox. se realiza un uso medio entorno al 11 % o 12 %. A partir de Julio en adelante durante el resto del año se visualiza un decremento entorno al 7 % u 8 % de media, ya que podemos ver unos pequeños altibajos.
- La zona verde, nos indica el uso de CPU por parte del sistema, en la gráfica podemos comentar varias acentuaciones. El uso mayor de la CPU por parte del sistema se

encuentra entorno a los meses Diciembre, Enero y Febrero, llegando a un pico máximo del 20 % en Noviembre y el 10 % en Enero.

A partir de Febrero el uso de la CPU por parte del sistema va decreciendo conforme el resto del año, llegando hasta Junio con una media por debajo del 10 %. Y a partir del mes de Julio, prácticamente el sistema no hace uso de la CPU, llegando al resto del año sobre una media del 1 %.

- La línea 'rosita' que hablamos sobre el tiempo de CPU que ha estado ociosa a la espera de que las operaciones de entrada y salida terminaran. Por mencionar un poco vemos como prácticamente la mitad del año desde Noviembre hasta Junio aproximadamente tiene un bastante gasto de CPU, y durante los meses restantes disminuye prácticamente a la mitad. Prácticamente el 400 % es debido a que el sistema cuenta con 4 núcleos, y podemos decir que cada núcleo se encuentra prácticamente al 100 %, y la suma de ambos resulta el 400 % mencionado.

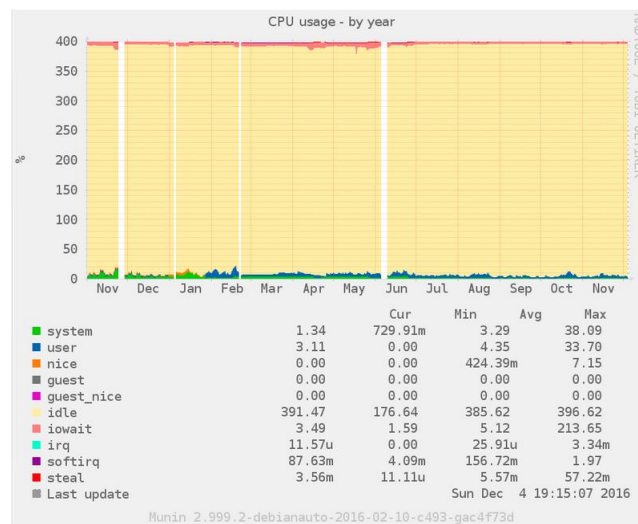


Figura 6.1: Información gráfica sobre el uso de CPU de una demo de MUNIN.

Y otro ejemplo, un poco mas sencillo, ya que nos refleja solo una línea que nos dice el número de hebras o procesos que se ejecutan en el sistema, nos referimos a la Figura 6.2.

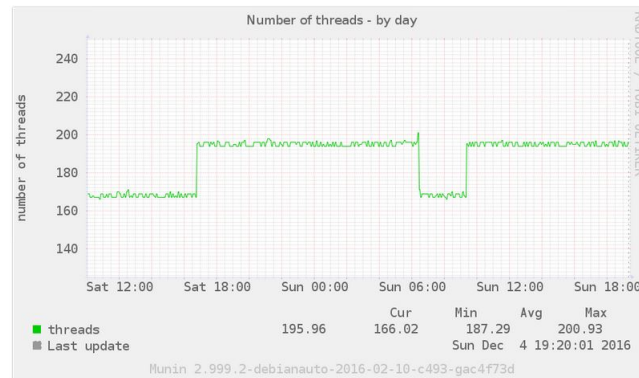


Figura 6.2: Información gráfica sobre las hebras de una demo de MUNIN, también se le decir procesos.

En este gráfico anterior visualizamos la monitorización de unas 30 horas aproximadamente.

Vemos que a partir del Sábado a las 18:00 hasta el Domingo a las 06:00 se utiliza un poco menos de 200 hebras o procesos de media.

Desde el domingo a las 12:00 hasta las 18:00 del mismo día también unas 200 hebras o procesos.

Y luego observamos unos intervalos de reposo, como por ejemplo el Sábado de las 12:00 horas hasta las 18:00 horas, con un uso de 170 procesos o hebras aproximadamente. A igual que ocurre lo mismo aproximadamente el Domingo a las 6:00 hasta las 10:00 horas del mismo día.

Realmente si esta gráfica se representara centrada en el origen '0,0' de los ejes, podríamos apreciar una gráfica prácticamente lineal, con un pequeño ruido, ya que tan sólo se podría observar un 15 % de incremento en el número de hebras.

Quizás podríamos estar hablando de un servidor streaming de radio, dónde la hora más solicitada podría situarse en torno a las 18:00 del sábado hasta las 06:00, donde se emite un programa de radio más favorito para los oyentes, y otro programa que comienza a las 12:00 del domingo.

## 7. Cuestión 7. Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de strace o busque otro y coméntelo.

En la web de Chad Fowler's [14], encontramos un artículo sobre el uso de 'strace' que me resultó muy interesante, y que dicho autor consiguió solucionar un problema con su servidor, dónde las consultas a la base de datos se volvieron muy lentas, incluso disminuyeron las solicitudes a sus sistema web.

Gracias a 'strace' podemos realizar un seguimiento total de todas las llamadas al sistema que produce una aplicación o cualquier programa. Por lo que el autor decidió utilizar

esta opción para ver que pasaba con la ejecución de su servidor Web.

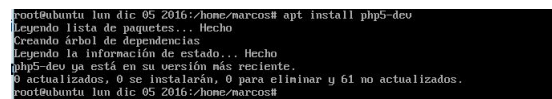
No nos muestra el problema real, pero no das una orientación como pudo obtener el problema, ejecutando el programa mediante 'strace' con la opción -T, el sistema ejecuta el programa, y muestra por la salida cada llamada que se realiza al sistema y el tiempo en ejecutarse cada una de ellas. Por lo que en resumen viendo los tiempos de cada llamada al sistema descubrió que su programa realizaba una llamada al sistema que demoraba bastante tiempo. Y gracias a esta simple comprobación pudo localizar rápidamente y sin apenas trabajo el problema de su sistema.

## 8. Cuestión 8. Escriba un script en Python o PHP y analice su comportamiento usando el profiler presentado.

Voy a elegir el script PHP realizado en la práctica 2 ya que lo tenemos creado y funcionando. Para ello voy a elegir también el profiler 'Xdebug' para poder depurar los script PHP.

Xdebug [11] es una herramienta como extensión de PHP, que nos permite depurar nuestros script PHP, pudiendo realizar trazas de ejecución o incluso utilizando puntos de ruptura, poder observar que esta ocurriendo a través de las instrucciones ejecutadas.

En primer lugar vamos a proceder a la instalación del paquete 'Xdebug' [8] en nuestro sistema Ubuntu Server, para ello en primer lugar debemos de tener instalado el paquete 'php5-dev', si no lo tenemos disponible como es mi caso procedemos a instalar mediante la orden 'apt install php5-dev' como vemos en la Figura 8.1. En esta captura ya tenía instalado el paquete por lo que nos muestra que el paquete ya ha sido instalado.



```
root@ubuntu:~# apt install php5-dev
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
php5-dev ya está en su versión más reciente.
0 actualizados, 0 se instalarán, 0 para eliminar y 61 no actualizados.
root@ubuntu:~#
```

Figura 8.1: Instalación del paquete 'php5-dev' en nuestro sistema Ubuntu Server.

Una vez ya tenemos disponible el paquete 'php5-dev' procedemos a instalar el paquete 'Xdebug' mediante la orden 'pecl install xdebug' como podemos observar en la Figura 8.2.

```

Documentation is available online as well:
- A list of all settings: http://xdebug.org/docs-settings.php
- A list of all functions: http://xdebug.org/docs-functions.php
- Profiling instructions: http://xdebug.org/docs-profiling2.php
- Remote debugging: http://xdebug.org/docs-debugger.php

NOTE: Please disregard the message
      You should add "extension=xdebug.so" to php.ini
      that is emitted by the PECL installer. This does not work for
      Xdebug.

running: find "/tmp/pear/temp/pear-build-root@dfgm/install-xdebug-2.5.0" | xargs ls -dls
297515  4 drwxr-xr-x 3 root root  4096 dic  5 10:23 /tmp/pear/temp/pear-build-root@dfgm/install
-xdebug-2.5.0
298633  4 drwxr-xr-x 3 root root  4096 dic  5 10:23 /tmp/pear/temp/pear-build-root@dfgm/install
-xdebug-2.5.0/usr
298634  4 drwxr-xr-x 3 root root  4096 dic  5 10:23 /tmp/pear/temp/pear-build-root@dfgm/install
-xdebug-2.5.0/usr/lib
299924  4 drwxr-xr-x 3 root root  4096 dic  5 10:23 /tmp/pear/temp/pear-build-root@dfgm/install
-xdebug-2.5.0/usr/lib/php5
299925  4 drwxr-xr-x 2 root root  4096 dic  5 10:23 /tmp/pear/temp/pear-build-root@dfgm/install
-xdebug-2.5.0/usr/lib/php5/20121212
298629 1140 -rwxr-xr-x 1 root root 1164076 dic  5 10:23 /tmp/pear/temp/pear-build-root@dfgm/install
-xdebug-2.5.0/usr/lib/php5/20121212/xdebug.so

Build process completed successfully
Installing '/usr/lib/php5/20121212/xdebug.so'
install ok: channel://pecl.php.net/xdebug-2.5.0
configuration option 'php.ini' is not set to php.ini location
You should add "zend_extension=xdebug.so" to php.ini
root@ubuntu lun dic 05 2016:/home/marcos#
root@ubuntu lun dic 05 2016:/home/marcos# pecl install xdebug

```

Figura 8.2: Instalación del paquete 'Xdebug' en nuestro sistema Ubuntu Server.

Ya nos falta solo añadir la línea que nos dice [8], en la que le decimos al servicio 'apache' que nos cargue el servicio 'xdebug'. Para ello tenemos que editar el archivo 'php.ini' y añadimos las líneas siguientes:

- 'zend\_extension="/usr/lib/php5/20121212/xdebug.so'
- 'xdebug.profiler\_enable=1'
- 'xdebug.profiler\_output\_dir="/var/www/'

Tengo que destacar que '20121212' en la primera línea que añadimos se corresponde a mi directorio donde se encuentra el archivo 'xdebug.so' para cualquier otro usuario debemos de buscar donde se encuentra nuestra archivo 'xdebug' y agregar la ruta correcta. Vemos tal proceso en la Figura 8.3.

```

; http://php.net/ignore-user-abort
;ignore_user_abort = On

; Determines the size of the realpath cache to be used by PHP. This value should
; be increased on systems where PHP opens many files to reflect the quantity of
; the file operations performed.
; http://php.net/realpath-cache-size
;realpath_cache_size = 16k

; Duration of time, in seconds for which to cache realpath information for a given
; file or directory. For systems with rarely changing files, consider increasing this
; value.
; http://php.net/realpath-cache-ttl
;realpath_cache_ttl = 120

; Enables or disables the circular reference collector.
; http://php.net/zend.enable_gc
;zend.enable_gc = 1
xdebug.profiler_enable=1
xdebug.profiler_output_dir="/var/www/"

zend.enable_gc = On

zend_extensions="/usr/lib/php5/20121212/xdebug.so"
; If enabled, scripts may be written in encodings that are incompatible with
; the scanner. CP936, Big5, CP949 and Shift_JIS are the examples of such
; encodings. To use this feature, mbstring extension must be enabled.
; Default: Off
;zend.multibyte = Off

; allows to set the default encoding for the scripts. This value will be used
; unless "declare(encoding=...)" directive appears at the top of the script.
; Only affects if zend.multibyte is set.
; Default: ""
;zend.script_encoding =

root@ubuntu vie dic 09 2016:/var/www#

```

Figura 8.3: Añadiendo la herramienta 'xdebug' a nuestro servicio 'apache'.

No se nos puede olvidar reiniciar el servicio 'apache' por lo que recuerdo utilizar la orden 'service apache2 restart' para reiniciar nuestro servicio 'apache' y tenga efecto estos cambios realizados anteriormente.

Comprobamos que todo a sido correcto y mediante una conexión ssh con entorno gráfico vamos a verificar que la herramienta 'xdebug' se ha añadido a nuestro servidor PHP, nos fijamos en la Figura 8.4.

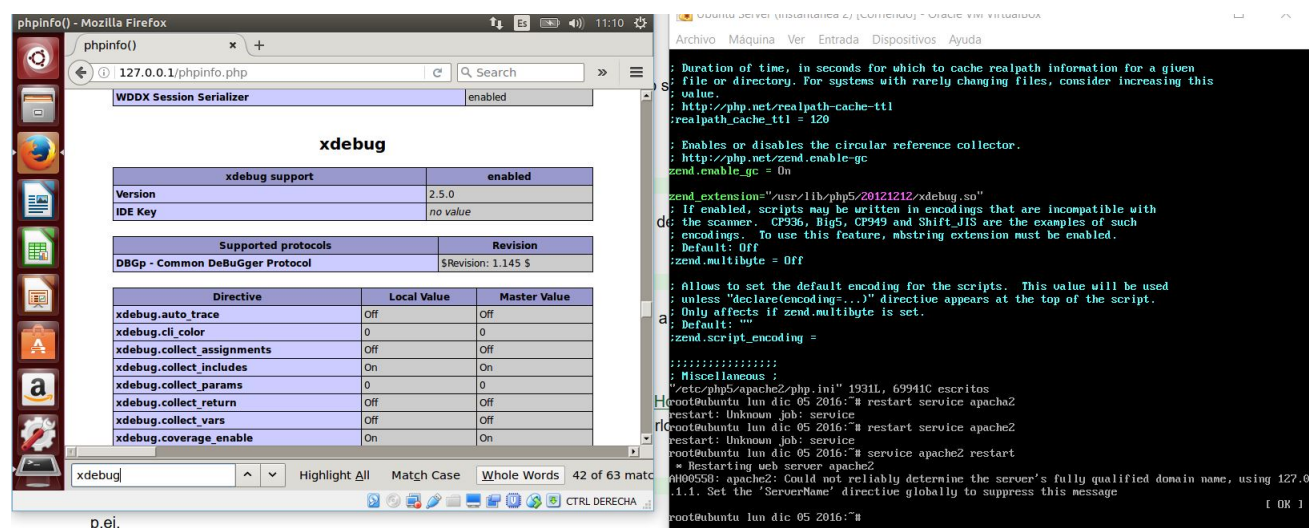


Figura 8.4: Herramienta 'Xdebug' añadida a nuestro servidor PHP.

En la documentación [7] nos habla sobre 'KCacheGrind' que es una herramienta externa para visualizar y depurar nuestros script, vamos a realizar un ejemplo ejecutando 'KCacheGrind' mediante una conexión 'ssh' desde mi máquina anfitriona con efectos gráficos, para poder ver lo que ocurre.

En primer lugar si todo funciona correctamente debemos de hacer una ejecución del script PHP que deseemos, en mi caso voy a ejecutar desde el navegador anfitrión '192.168.56.101/phpinfo.php', realizando esta ejecución se debe crear un fichero de salida 'cachegrind.out' como vemos en la Figura 8.5.





'ini\_get', y un 97,14 % es utilizado por nuestro programa principal 'main'.

## 9. Cuestión 9. 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).

Podemos utilizar un simple profiler de MySQL, que ya viene incluido cuando realicemos la instalación de dicho servicio. Para ello consultamos [1] y vemos que necesitamos habilitar la variable de sesión profiling.

Una vez que iniciamos sesión en nuestro servicio MySQL, habilitamos la variable de sesión con la orden 'SET profiling=1', como vemos en la Figura 9.1.

```
root@ubuntu:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 51
Server version: 5.5.53-0ubuntu0.14.04.1 (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> SET profiling=1;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Figura 9.1: Inicio de sesión en MySQL e inicializamos la variable de sesión del profiler.

Ahora ya todo en funcionamiento vamos a crear una base de datos, le vamos añadir alguna información y realizamos la consulta para comentar los tiempos. Vemos en la Figura 9.2 unos simples pasos.

```
root@ubuntu:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 51
Server version: 5.5.53-0ubuntu0.14.04.1 (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> SET profiling=1;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE DATABASE prueba;
Query OK, 1 row affected (0.00 sec)

mysql> USE prueba
Database changed
mysql> CREATE TABLE tablaPrueba (nombre VARCHAR(20), apellidos VARCHAR(50));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MySQL server version for the right syntax to use near '' at line 1
mysql> CREATE TABLE tablaPrueba (nombre VARCHAR(20));
Query OK, 0 rows affected (0.07 sec)

mysql> insert into tablaPrueba (nombre) values ('Marcos');
Query OK, 1 row affected (0.01 sec)

mysql>
```

Figura 9.2: Creando una base de datos, una nueva tabla, y añadiendo valores a la tabla de base de datos MySQL.

Ahora ya todo listo vamos a obtener los tiempos de las consultas realizadas mediante la orden 'SHOW PROFILES' y obtenemos la Figura 9.3. Podemos observar los tiempos que ha tardado en ejecutarse cada consulta, como comentario, podemos ver que la consulta que más tiempo a consumido a sido la consulta número 6, que pertenece a la creación de la nueva tabla con un tiempo de 0.0657 segundos aproximadamente.

```
mysql> SHOW PROFILES;
```

Query_ID	Duration	Query
1	0.00297950	CREATE DATABASE prueba
2	0.00213100	SELECT DATABASE()
3	0.01735775	show databases
4	0.00105375	show tables
5	0.00100225	CREATE TABLE tablaPrueba (nombre VARCHAR(20), apellidos VARCHAR(50))
6	0.06571300	CREATE TABLE tablaPrueba (nombre VARCHAR(20))
7	0.00793900	insert into tablaPrueba (nombre) values ('Marcos')

7 rows in set (0.00 sec)

Figura 9.3: Tabla de tiempos de las consultas realizadas.

Podemos obtener más detalles sobre los tiempos de una consulta concreta, por ejemplo, vamos a mostrar los tiempos obtenidos para la ejecución de la consulta para crear una tabla.

Utilizamos la orden 'SHOW PROFILE CPU FOR QUERY 5;' para obtener los tiempos que se han consumido cuando se ha ejecutado dicha consulta.

Nos fijamos en la Figura 9.4 y como podemos ver obtenemos los tiempos de CPU totales, de usuario, y de sistema. Por detallar algo, podemos ver que el tiempo mayor se encuentra en el estado de creando la tabla, con un total de 0.026 segundos aproximadamente, 0 segundos de CPU por el usuario, pero en cambio 0.008 segundos ha utilizado la CPU por parte del sistema.

```
mysql> SHOW PROFILE CPU FOR QUERY 5;
```

Status	Duration	CPU_user	CPU_system
starting	0.000066	0.000000	0.000000
checking permissions	0.000015	0.000000	0.000000
Opening tables	0.001376	0.000000	0.000000
System lock	0.000063	0.000000	0.000000
creating table	0.026414	0.000000	0.000000
After create	0.000017	0.000000	0.000000
query end	0.000006	0.000000	0.000000
closing tables	0.000033	0.000000	0.000000
freeing items	0.007064	0.000000	0.000000
logging slow query	0.000016	0.000000	0.000000
cleaning up	0.000011	0.000000	0.000000

11 rows in set (0.01 sec)

```
mysql> ^Ctrl-C -- exit!
^C
root@ubuntu:~#
```

Figura 9.4: Tabla de tiempos de la CPU para la consulta elegida.

## Referencias

- [1] Oracle. 14.7.5.30 SHOW PROFILE Syntax. © 2016. <http://dev.mysql.com/doc/refman/5.7/en/show-profile.html>. consultado el 5 de Diciembre de 2016.
- [2] Creación de conjuntos de recopiladores de datos. © 2016 Microsoft. [https://technet.microsoft.com/es-es/library/cc749337\(v=ws.11\).aspx](https://technet.microsoft.com/es-es/library/cc749337(v=ws.11).aspx). consultado el 30 de Noviembre de 2016.

- [3] 5.4.4. Archivo de registro de dpkg. El manual del Administrador de Debian. <https://debian-handbook.info/browse/es-ES/stable/sect.manipulating-packages-with-dpkg.html>. consultado el 28 de Noviembre de 2016.
- [4] 9.7. Programación de tareas con cron y atd. El manual del Administrador de Debian. <https://debian-handbook.info/browse/es-ES/stable/sect.task-scheduling-cron-atd.html>. consultado el 30 de Noviembre de 2016.
- [5] 9.7.1. Formato de un archivo crontab. El manual del Administrador de Debian. <https://debian-handbook.info/browse/es-ES/stable/sect.task-scheduling-cron-atd.html>. consultado el 30 de Noviembre de 2016.
- [6] dmesg(8) Linux man page. <https://linux.die.net/man/8/dmesg>. consultado el 2 de Diciembre de 2016.
- [7] Introduction. XDEBUG EXTENSION FOR PHP | DOCUMENTATION. <https://xdebug.org/docs/profiler>. consultado el 5 de Diciembre de 2016.
- [8] PECL Installation. XDEBUG EXTENSION FOR PHP | DOCUMENTATION. <https://xdebug.org/docs/install>. consultado el 5 de Diciembre de 2016.
- [9] dpkg(1) Linux man page. <https://linux.die.net/man/1/dpkg>. consultado el 28 de Noviembre de 2016.
- [10] Ver informes en el Monitor de rendimiento de Windows. © 2016 Microsoft. [https://technet.microsoft.com/es-es/library/cc766130\(v=ws.11\).aspx](https://technet.microsoft.com/es-es/library/cc766130(v=ws.11).aspx). consultado el 30 de Noviembre de 2016.
- [11] 3.5. XDebug. © 2006-2016 LibrosWeb.es. [http://librosweb.es/libro/php\\_correcto/capitulo\\_3/xdebug.html](http://librosweb.es/libro/php_correcto/capitulo_3/xdebug.html). consultado el 5 de Diciembre de 2016.
- [12] MUNIN. <http://munin-monitoring.org/>. consultado el 4 de Diciembre de 2016.
- [13] Demos MUNIN. <http://demo.munin-monitoring.org/>. consultado el 4 de Diciembre de 2016.
- [14] Finding out what's slow. Chad Fowler's web site. <http://chadfowler.com/2014/01/26/the-magic-of-strace.html>. consultado el 5 de Diciembre de 2016.
- [15] Crear un conjunto de recopiladores de datos para supervisar los contadores de rendimiento. © 2016 Microsoft. [https://technet.microsoft.com/es-es/library/cc722414\(v=ws.11\).aspx](https://technet.microsoft.com/es-es/library/cc722414(v=ws.11).aspx). consultado el 30 de Noviembre de 2016.