

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

Memoria Práctica 3

Simón López Vico

9 de mayo de 2017

Índice

1. Cuestión 1.	4
1.1. a) ¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes?	4
1.2. b) ¿Qué significan las terminaciones .1.gz o .2.gz de los archivos en ese directorio?	4
2. Pruebe a ejecutar el comando <code>dmesg</code>, 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. Ejecute el monitor de “System Performance” (<code>perfmon</code>) y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.	6
4. 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 Escritorio\logs. Incluya las capturas de pantalla de cada paso.	8
5. Visite la web del proyecto y acceda a la demo que proporcionan (http://demo.munin-monitoring.org/) donde se muestra cómo monitorizan un servidor. Monitorice varios parámetros y haga capturas de pantalla de lo que está mostrando comentando qué observa.	14
6. Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de <code>strace</code> o busque otro y coméntelo.	16
7. Escriba un script en Python o PHP y analice su comportamiento usando el profiler presentado.	16
8. Acceda a la consola <code>mysql</code> (o a través de <code>phpMyAdmin</code>) y muestre el resultado de mostrar el “profile” de una consulta (la creación de la BD y la consulta la puede hacer libremente).	18

Índice de figuras

2.1. Comparación del comando <code>dmesg</code> antes y después de conectar un USB. . .	5
3.1. Información actual del sistema dada por <i>System Perfomance</i>	6
3.2. Gráfica del uso del procesador.	7
3.3. Apartado del diagnóstico del sistema.	7
3.4. Seguimiento de los eventos del sistema.	8
4.1. Creación del recopilador de datos.	9

4.2.	Elección de datos a añadir al recopilador.	9
4.3.	Selección de los contadores de rendimiento.	10
4.4.	Asignación del intervalo de muestra.	10
4.5.	Asignación del intervalo de muestra.	11
4.6.	Gráfica completa del informe del recopilador.	12
4.7.	Gráfica de las variables de “Servicio web”.	12
4.8.	Gráfica de las variables de “Procesos”.	13
4.9.	Gráfica de las variables de “Procesador”.	13
5.1.	Uso del disco del servidor durante una semana.	14
5.2.	Uso de la memoria del servidor durante una semana.	15
5.3.	Cambio de tamaño de los archivos de registro de munin a lo largo de una semana.	15
6.1.	Búsqueda de los archivos de registro relacionados con Apache.	16
7.1.	Ejecución del profiler sobre nuestro script.	17
8.1.	Bases de datos disponibles en phpMyAdmin.	19
8.2.	Consulta de los elementos de la tabla “help_relation”.	19
8.3.	Datos y gráfica del <i>profiling</i> de la consulta.	20

1. Cuestión 1.

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

El directorio `/var` contiene subdirectorios y archivos de información sobre aplicaciones de nuestros SO. Dentro del subdirectorio `/var/log` encontraremos archivos de registro y el directorio `/apt`, en el cual se encontrará el fichero `history.log` que contendrá los programas instalados con `apt` hasta la fecha, además de las aplicaciones borradas, las actualizaciones realizadas manualmente... Es decir, todas las llamadas a el comando `apt`. [1]

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

Los archivos con terminación `.gz` se encontrarán en `/var/log`, y serán archivos donde se almacenen comprimidos los archivos log generados a lo largo del tiempo del SO. El número que precede a `.gz` indicará como de antiguo es el archivo (cuanto mayor sea el número, más antiguos serán los logs que se encuentren dentro de él), ya que los log van *rotando* hasta un límite; al llegar a este límite se crea un archivo nuevo y el antiguo se renombra con el número siguiente al que antes poseía. [2] [3]

2. Pruebe a ejecutar el comando `dmesg`, 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 empezar, crearemos dos documentos, uno con la información de `dmesg` antes de conectar el USB y otro tras haberlo conectado (`dmesg > before_usb.txt // dmesg > after_usb.txt`). Tras esto, accederemos a un comparador online[6] para encontrar más rápidamente las diferencias entre la ejecución del comando cada vez. Introduciremos los dos documentos a comparar y obtendremos el siguiente resultado (a la izquierda `before_usb.txt`; a la derecha `after_usb.txt`):

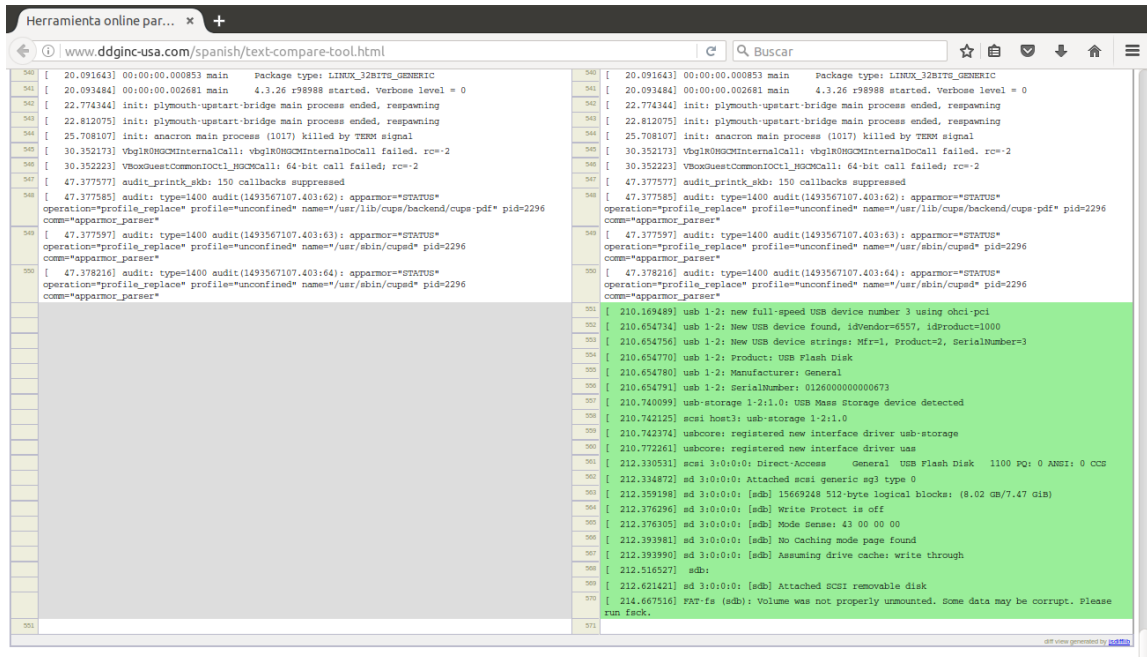


Figura 2.1: Comparación del comando `dmesg` antes y después de conectar un USB.

Como podemos ver, el único cambio obtenido al ejecutar el comando después de conectar la unidad USB es que se han añadido unas 20 líneas con la siguiente información.

```
[ 210.169489] usb 1-2: new full-speed USB device number 3 using
ohci-pci
[ 210.654734] usb 1-2: New USB device found, idVendor=6557,
idProduct=1000
[ 210.654756] usb 1-2: New USB device strings: Mfr=1, Product=2,
SerialNumber=3
[ 210.654770] usb 1-2: Product: USB Flash Disk
[ 210.654780] usb 1-2: Manufacturer: General
[ 210.654791] usb 1-2: SerialNumber: 0126000000000673
[ 210.740099] usb-storage 1-2:1.0: USB Mass Storage device detected
[ 210.742125] scsi host3: usb-storage 1-2:1.0
[ 210.742374] usbcore: registered new interface driver usb-storage
[ 210.772261] usbcore: registered new interface driver uas
[ 212.330531] scsi 3:0:0:0: Direct-Access      General  USB Flash Disk
1100 PQ: 0 ANSI: 0 CCS
[ 212.334872] sd 3:0:0:0: Attached scsi generic sg3 type 0
[ 212.359198] sd 3:0:0:0: [sdb] 15669248 512-byte logical blocks: (8.02
GB/7.47 GiB)
[ 212.376296] sd 3:0:0:0: [sdb] Write Protect is off
```

```
[ 212.376305] sd 3:0:0:0: [sdb] Mode Sense: 43 00 00 00
[ 212.393981] sd 3:0:0:0: [sdb] No Caching mode page found
[ 212.393990] sd 3:0:0:0: [sdb] Assuming drive cache: write through
[ 212.516527] sdb:
[ 212.621421] sd 3:0:0:0: [sdb] Attached SCSI removable disk
[ 214.667516] FAT-fs (sdb): Volume was not properly mounted. Some data
may be corrupt. Please run fsck.
```

El kernel de Linux nos dará a conocer que un dispositivo de almacenamiento USB ha sido conectado a la máquina, ofreciendo información sobre éste. También nos informará sobre si ha ocurrido algún problema durante el montaje; en nuestro caso, nos dice que el USB no fue desmontado correctamente, y nos pide ejecutar el comando `fsck`[7] para remontar el dispositivo y poder acceder a los datos los cuales podrían estar corruptos.

3. Ejecute el monitor de “System Performance” (perfmon) y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.

Comenzaremos abriendo *Símbolos del Sistema*, que será la terminal de Windows, y dentro de ella ejecutaremos el comando `perfmon`. Tras ello, se nos abrirá la página principal de *System Performance* en la cual aparecerá la información sobre el estado del equipo en este momento.

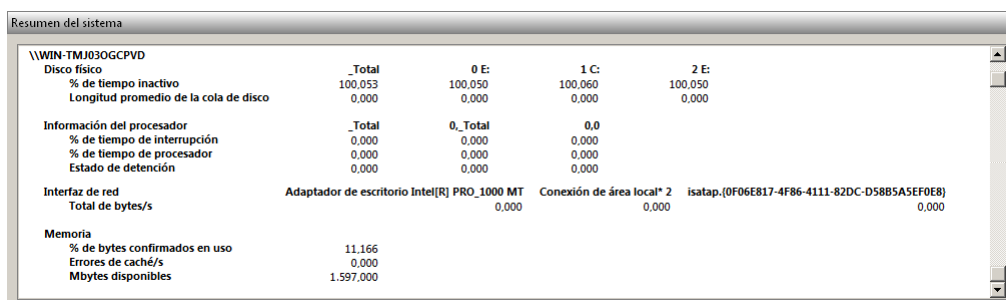


Figura 3.1: Información actual del sistema dada por *System Performance*.

Tendremos por tanto información sobre el disco, el procesador, el estado de la red y la memoria. En la zona izquierda de la ventana tendremos un menú para navegar por distintas opciones. Comenzaremos eligiendo “Herramientas de supervisión”/“Monitor de rendimiento”. Aparecerá una gráfica mostrando el uso del procesador respecto al tiempo transcurrido. Los picos de la gráfica que aparecen en 3.2 han sido provocados abriendo distintos programas y ejecutando comandos en Powershell.

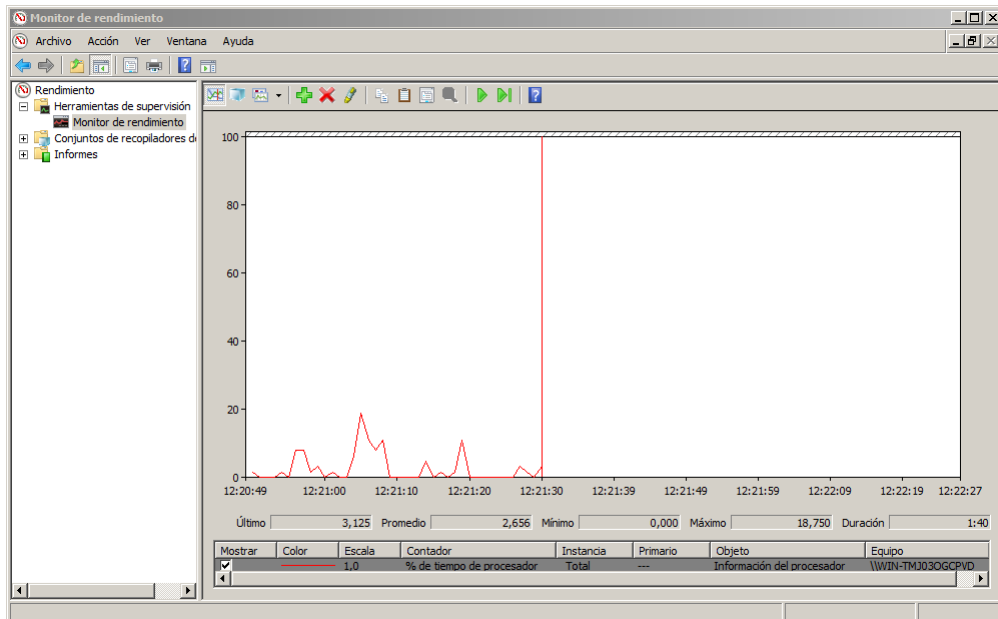


Figura 3.2: Gráfica del uso del procesador.

Dentro de “Conjuntos de recopiladores de datos” encontraremos el apartado “Sistema”, que contiene información sobre el rendimiento y el diagnóstico actual del SO. Seleccionando cualquiera de los nombres que aparecen a la derecha (Processor, System Services, Logical Disk Dirty Test...) en la figura 3.3, se nos abrirá una pequeña ventana con la información sobre dicho elemento del sistema.

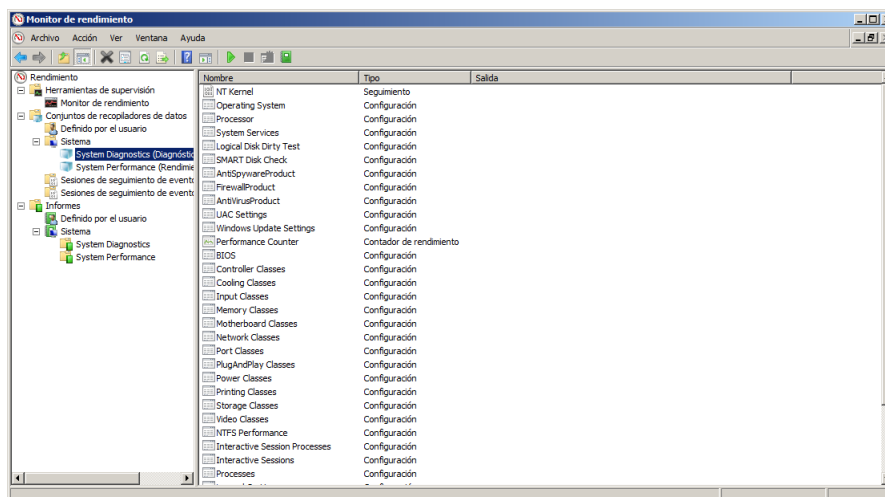


Figura 3.3: Apartado del diagnóstico del sistema.

También podremos obtener información de los distintos eventos ocurridos en el Kernel

[4]; al igual que en el apartado “Sistema”, pincharemos sobre el elemento deseado para obtener información adicional de éste.

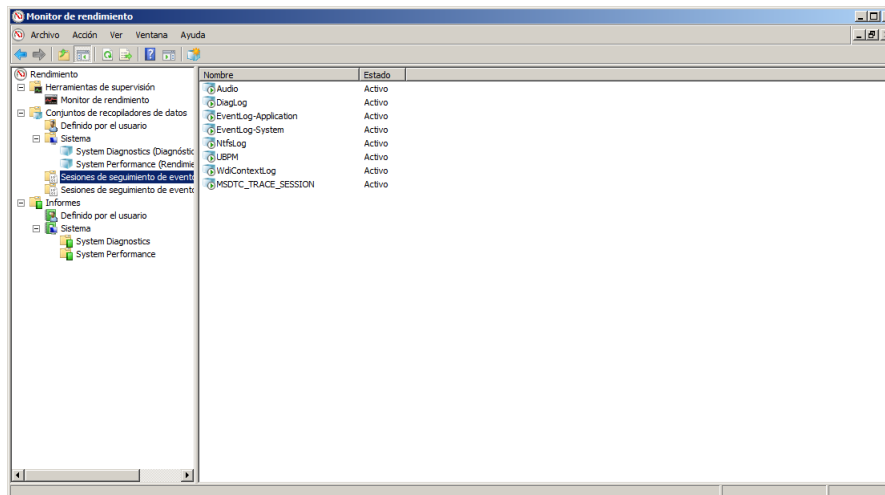


Figura 3.4: Seguimiento de los eventos del sistema.

Finalmente habrá un apartado de “Informes”, los cuales pueden haber sido definidos por el usuario o automáticamente por el sistema (sobre el diagnóstico o el rendimiento). Aún así, no habrá ningún informe disponible para comprobar su estructura y datos incluidos en él, ya que por nuestra parte no hemos programado ninguno todavía.

4. 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 Escritorio\logs. Incluya las capturas de pantalla de cada paso.

[5] Crearemos el recopilador de datos entrando al *System Performance* con el comando `perfmon` y haciendo click derecho sobre Conjunto de recopiladores de datos/Definidos por el usuario. Pincharemos sobre “Nuevo” (1) y se abrirá una ventana en la que asignaremos el nombre al recopilador de datos y seleccionaremos la creación manual (2). (Imagen 4.1)

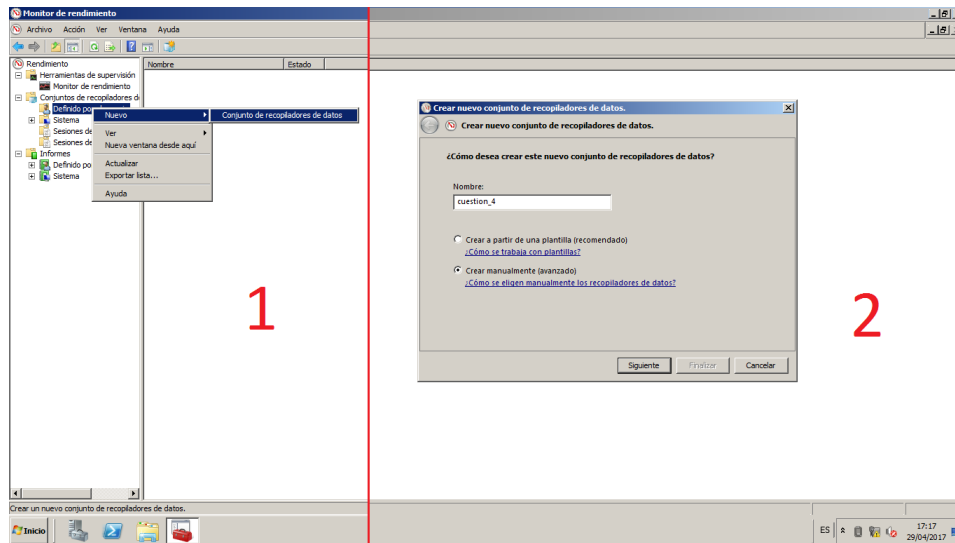


Figura 4.1: Creación del recopilador de datos.

Pinchamos en “Siguiente” y seleccionamos los tipos de datos a incluir en el recopilador, en nuestro caso el contador de rendimiento y los datos de seguimiento, como es pedido en el enunciado de esta cuestión. (Imagen 4.2)

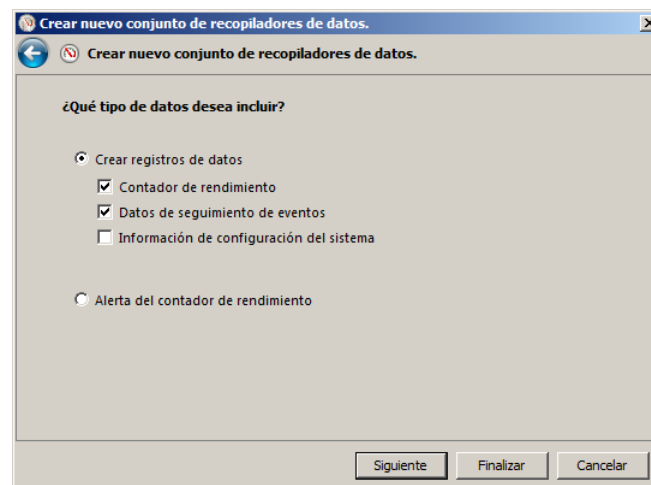


Figura 4.2: Elección de datos a añadir al recopilador.

A continuación, elegiremos los contadores de rendimiento a registrar pinchando en “Agregar”. Añadiremos “Información del procesador”, “Procesador”¹, “Proceso” y “Servicio web”. (Imagen 4.3)

¹No analizaremos los datos de “Información del procesador”, pues serán exactamente iguales a los obtenidos por “Procesador”.

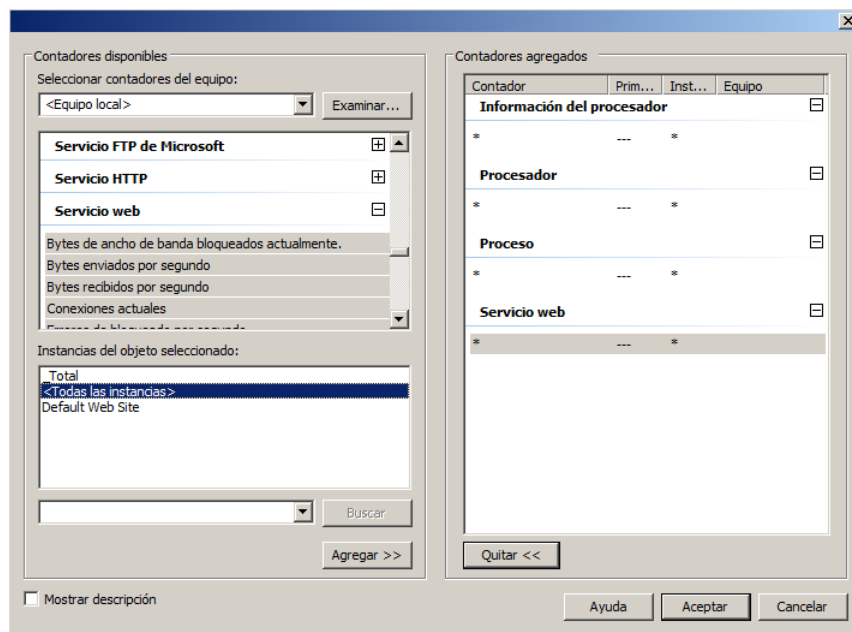


Figura 4.3: Selección de los contadores de rendimiento.

Pincharemos en “Aceptar” y asignaremos el intervalo de muestra a 5 segundos, como es pedido en el enunciado. (Imagen 4.4)

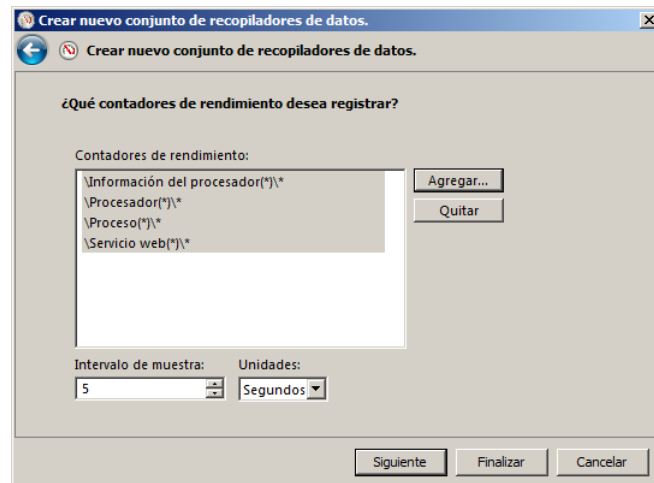


Figura 4.4: Asignación del intervalo de muestra.

Pinchando en “Siguiente” se nos ofrecerá habilitar los proveedores de eventos que queramos, aunque no elegiremos ninguno, pues no serán necesarios para la realización de esta

cuestión.

Tras esto, escogeremos el directorio Escritorio\logs como lugar en el que guardar los datos obtenidos por el recopilador. (Imagen 4.5)

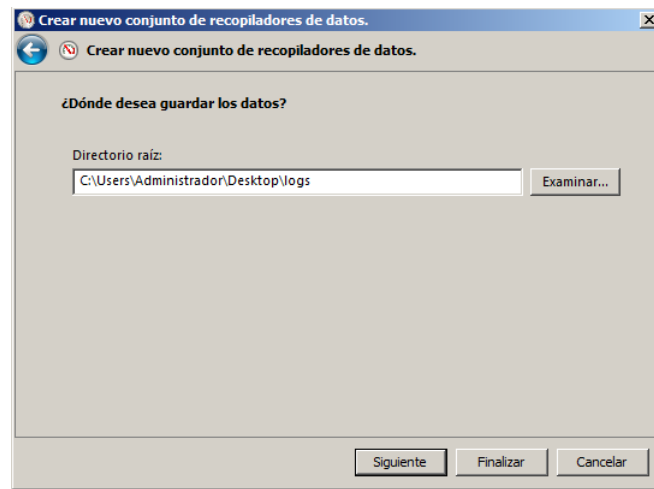


Figura 4.5: Asignación del intervalo de muestra.

Finalmente, se guardarán los datos y antes de finalizar se nos ofrecerá la opción de comenzar ya la recopilación de datos. Escogeremos finalizar empezando el análisis, aunque también podremos empezar la recopilación haciendo click derecho sobre el recopilador (que se encontrará en Conjunto de recopiladores de datos/Definidos por el usuario) y seleccionando "Iniciar".

Para terminar el análisis y poder mostrar los datos, seleccionaremos "Detener" haciendo click derecho sobre el recopilador. Para acceder al informe obtenido, podremos encontrarlo en Informes/Definidos por el usuario o en el directorio logs que creamos anteriormente en el Escritorio.

Abriremos el informe y encontraremos la siguiente gráfica:

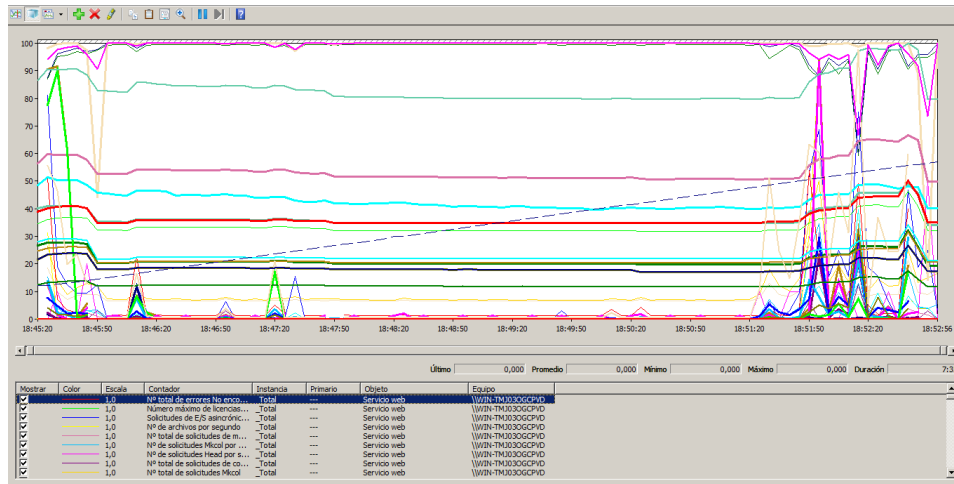


Figura 4.6: Gráfica completa del informe del recopilador.

Como podemos ver en la figura 4.6, no se pueden diferenciar bien los datos obtenidos ya que hay demasiadas variables, aunque podremos activar y desactivar los contadores que queramos para una mejor visualización seleccionando las casillas que aparecen debajo de la gráfica.

Comenzaremos visualizando los valores correspondientes a “Servicio web”. Ya que no hemos estado utilizando la máquina como servidor durante el análisis, no dispondremos de información y todos los valores serán 0, excepto el “Tiempo límite del servicio FTP” (en negro con línea discontinua), el cual irá creciendo con el tiempo. (Imagen 4.7)

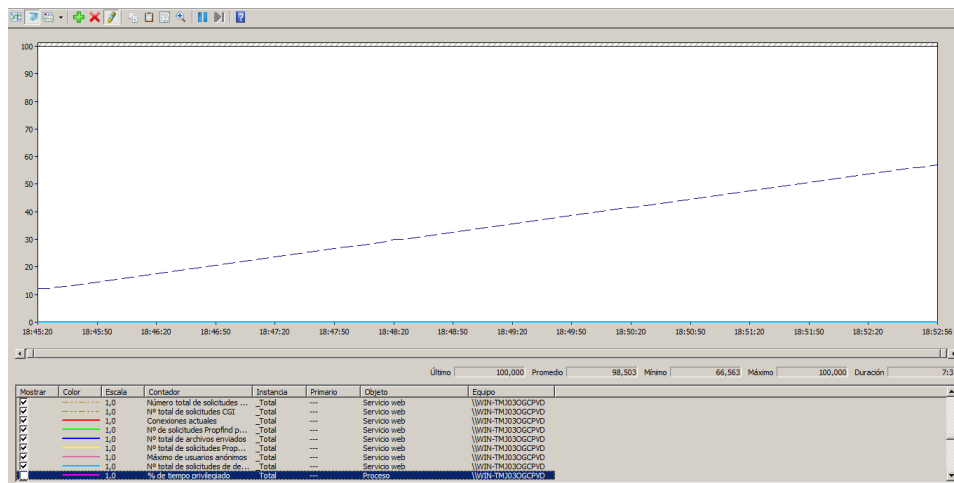


Figura 4.7: Gráfica de las variables de “Servicio web”.

Por otra parte, tendremos la información sobre “Procesos” la cual trabajará con valores como las operaciones de E/S de datos, el número de subprocesos activos, el número de

bytes de E/S escritos por segundo, etc. (Imagen 4.8)

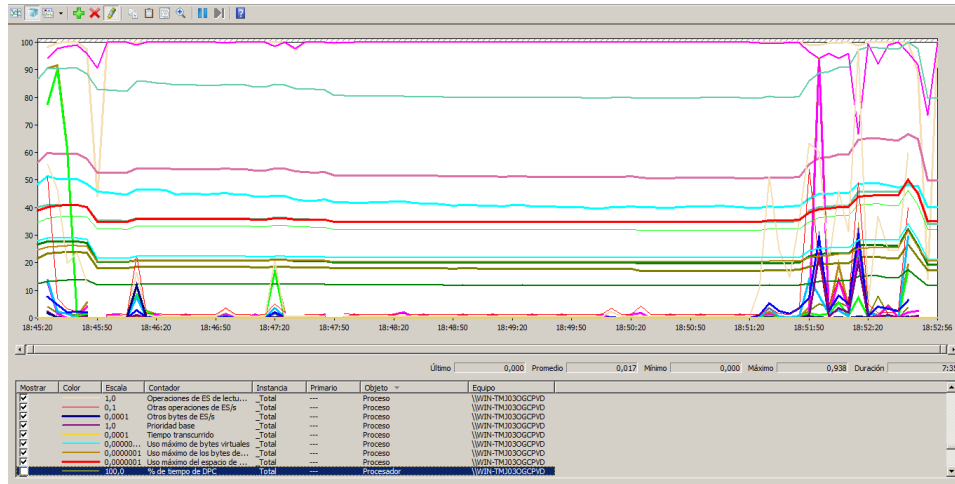


Figura 4.8: Gráfica de las variables de “Procesos”.

Finalmente trataremos los datos de los contadores de “Procesador”, relacionados con la actividad del procesador durante el tiempo activo del compilador. Los picos del final de la gráfica son los correspondientes a entrar en Internet Explorer, realizar búsquedas en Google, abrir y cerrar programas del SO y diversas operaciones para hacer trabajar el procesador. El resto de datos de la gráfica, los cuales son más “estables”, son debidos a que no he utilizado el SO durante ese tiempo, y los diferentes picos serán causa de procesos ejecutados en segundo plano. (Imagen 4.9)

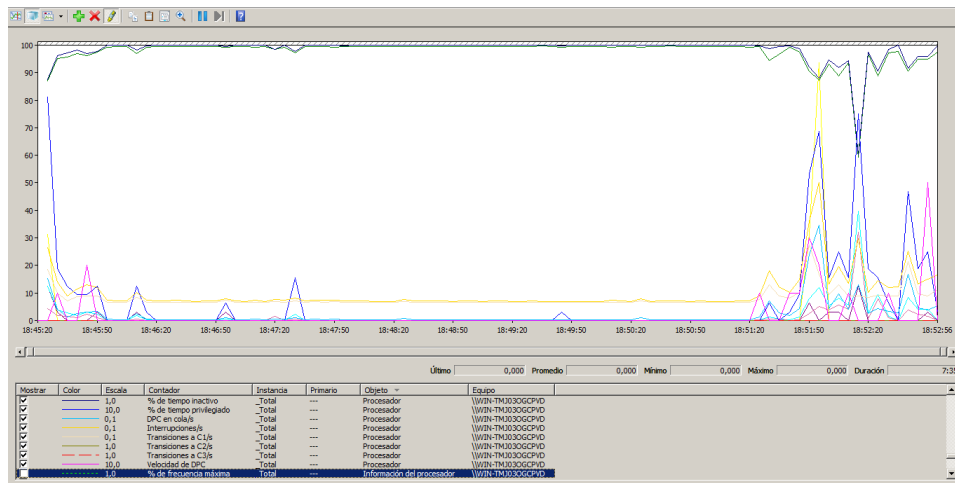


Figura 4.9: Gráfica de las variables de “Procesador”.

5. 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.

[12] En la página principal de la demo de Munin, tendremos la vista general (*Overview*) de los servidores que podremos monitorizar. Elegiremos el servidor que aloja demo.munin-monitoring.org, del cual podremos monitorizar la actividad del disco, de la cache, del kernel, etc.

Comenzaremos observando la gráfica referida al uso del disco durante una semana (imagen 5.1), en la cual podremos percibir un lento crecimiento del porcentaje de uso durante la semana frente a una caída en picado aproximadamente los domingos. Esto podría ser debido a un reinicio del servidor, o a la actividad de algún administrador del servidor durante el fin de semana para desfragmentar el disco, reduciendo así su uso al comienzo de la semana.

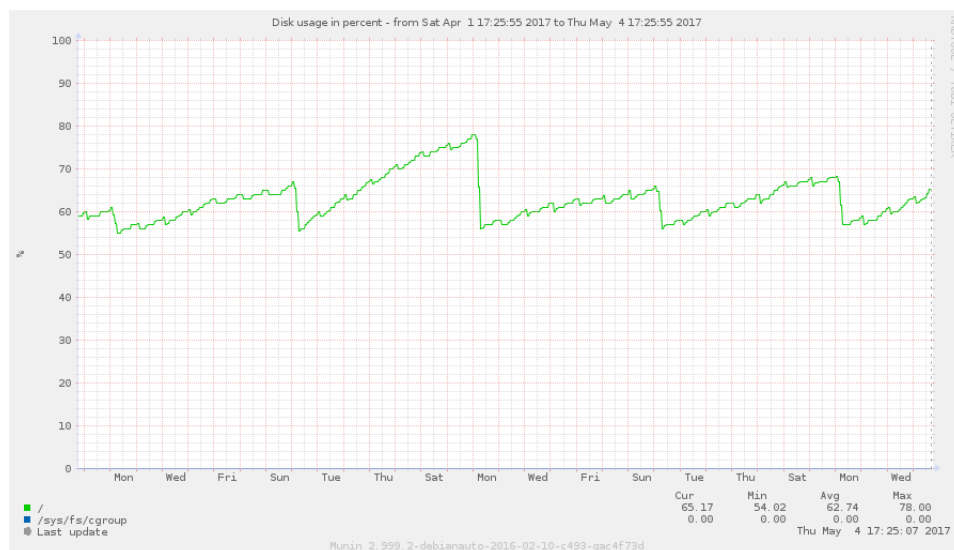


Figura 5.1: Uso del disco del servidor durante una semana.

Por otra parte, trataremos el uso de la memoria del servidor durante una semana (imagen 5.2). Podemos observar un mayor uso de aplicaciones (*apps*) frente al resto de opciones, siguiéndole el uso del *swap* (visto en la práctica anterior), el cual es usado en el momento que se llena la memoria RAM, que como podemos ver es muy a menudo.



Figura 5.2: Uso de la memoria del servidor durante una semana.

Finalmente veremos el cambio de tamaño que sufren los archivos de registro del servidor durante una semana. En la imagen 5.3 observaremos que cada cierto tiempo el tamaño de los archivos de registro disminuye considerablemente, lo cual puede ser debido a la compresión de éstos en el formato [número].gz para ahorrar espacio en el disco (visto en la cuestión 1, apartado b).

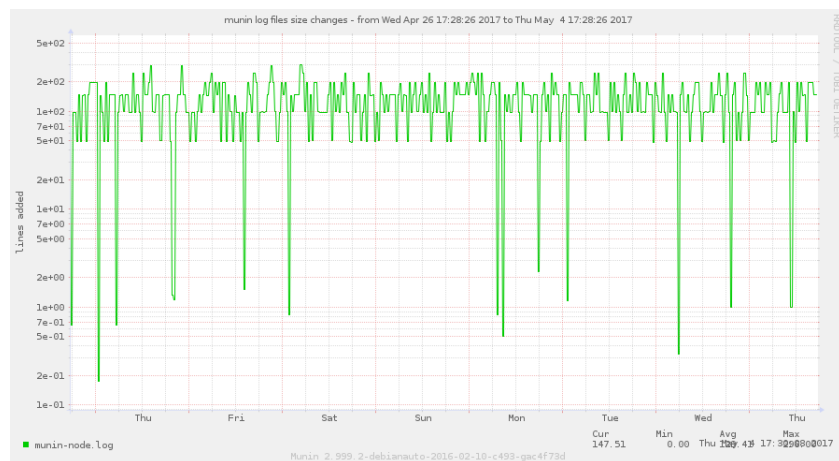


Figura 5.3: Cambio de tamaño de los archivos de registro de munin a lo largo de una semana.

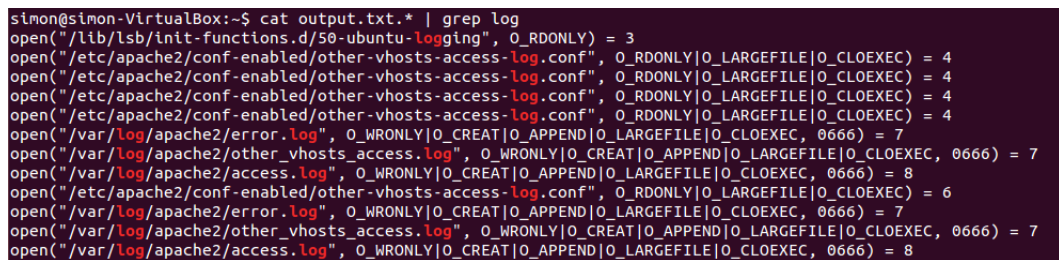
6. Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de `strace` o busque otro y coméntelo.

[8] El comando `strace` es una herramienta muy útil para la monitorización de las llamadas y señales del sistema durante la ejecución de un programa. Para el uso de `strace`, simplemente lo “adjuntaremos” al proceso el cual queramos monitorizar, mostrando por pantalla las llamadas y señales realizadas al sistema durante ese proceso. Además, `strace` devolverá los errores que haya encontrado en la ejecución.

Un ejemplo simple del uso de `strace` consistirá en encontrar los archivos de registro de un programa, en nuestro caso del servidor web Apache. Para ello, introduciremos el comando:

```
strace -ff -o output.txt -e open /etc/init.d/apache2 restart,
```

donde `-ff` hará que sigamos todos los *forks*, escribiendo cada uno de ellos en `[nombre] . [pid]` (donde `[nombre]` será el nombre asignado al archivo en el que escribir la salida de `strace` mediante la opción `-o` y `[pid]` el ID de cada subprocesso). También usaremos la opción `-e open` para buscar solo las llamadas al sistema de `open`, y elegiremos ejecutar el comando `/etc/init.d/apache2 restart` [9] para reiniciar el servidor Apache, haciendo que tenga que volver a abrir sus archivos de registro. Se nos crearán muchos ficheros, así que para encontrar los archivos de registro ejecutaremos `cat output.txt.* | grep log`, obteniendo el siguiente resultado:



```
simon@simon-VirtualBox:~$ cat output.txt.* | grep log
open("/lib/lsb/init-functions.d/50-ubuntu-logging", O_RDONLY) = 3
open("/etc/apache2/conf-enabled/other-vhosts-access-log.conf", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 4
open("/etc/apache2/conf-enabled/other-vhosts-access-log.conf", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 4
open("/etc/apache2/conf-enabled/other-vhosts-access-log.conf", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 4
open("/etc/apache2/conf-enabled/other-vhosts-access-log.conf", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 4
open("/var/log/apache2/error.log", O_WRONLY|O_CREAT|O_APPEND|O_LARGEFILE|O_CLOEXEC, 0666) = 7
open("/var/log/apache2/other_vhosts_access.log", O_WRONLY|O_CREAT|O_APPEND|O_LARGEFILE|O_CLOEXEC, 0666) = 7
open("/var/log/apache2/access.log", O_WRONLY|O_CREAT|O_APPEND|O_LARGEFILE|O_CLOEXEC, 0666) = 8
open("/etc/apache2/conf-enabled/other-vhosts-access-log.conf", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 6
open("/var/log/apache2/error.log", O_WRONLY|O_CREAT|O_APPEND|O_LARGEFILE|O_CLOEXEC, 0666) = 7
open("/var/log/apache2/other_vhosts_access.log", O_WRONLY|O_CREAT|O_APPEND|O_LARGEFILE|O_CLOEXEC, 0666) = 7
open("/var/log/apache2/access.log", O_WRONLY|O_CREAT|O_APPEND|O_LARGEFILE|O_CLOEXEC, 0666) = 8
```

Figura 6.1: Búsqueda de los archivos de registro relacionados con Apache.

De esta manera, obtendremos la localización de todos los archivos de registro relacionados con el servidor web Apache.

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

El script desarrollado (en Python) será el siguiente:

```
# script.py
# Desarrollado para analizar mediante un profiler
```



```

def suma(a,b):
    return a+b

def factorial(x):
    if x == 0:
        return 1
    else:
        return x * factorial(x - 1)

i=0

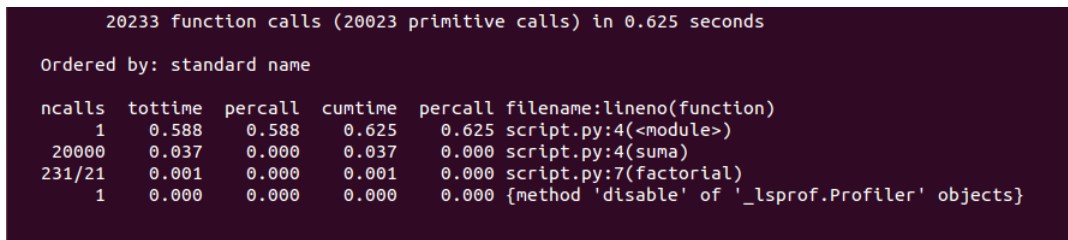
while i<20000:
    x=suma(10,i)
    print "La suma es %d." %(x)
    i+=1

j=0

while j<=20:
    y=factorial(j)
    print "El factorial de %d es %d." %(j, y)
    j+=1

```

Para ejecutar el profiler, en nuestro caso cProfile, simplemente ejecutaremos el script mediante el comando `python -m cProfile script.py` [10], obteniendo la salida que se muestra en 7.1.



```

20233 function calls (20023 primitive calls) in 0.625 seconds

Ordered by: standard name

ncalls  tottime  percall  cuntime  percall  filename:lineno(function)
      1    0.588    0.588    0.625    0.625  script.py:4(<module>)
    20000  0.037    0.000    0.037    0.000  script.py:4(suma)
    231/21  0.001    0.000    0.001    0.000  script.py:7(factorial)
      1    0.000    0.000    0.000    0.000  {method 'disable' of '_lsprof.Profiler' objects}

```

Figura 7.1: Ejecución del profiler sobre nuestro script.

La primera línea indicará que han sido monitorizadas 20233 llamadas, de las cuales 20023 son primitivas, es decir, que no han sido realizadas vía recursión. La información dada sobre la ejecución será la siguiente:

- **ncalls:** número de veces que se ha llamado cada función. La línea en la que aparece 231/21 indica que se ha llamado 231 veces a la función, de las cuales solo 21 de ellas son llamadas primitivas (debido a la función factorial).

- **tottime**: tiempo requerido para ejecutar cada función; la que más tiempo ha requerido para nosotros ha sido la función `<module>`, que se refiere a la ejecución del script completo.[11]
- **percall (tottime)**: mostrará el cociente del tiempo total (tottime) entre el número de llamadas (ncalls).
- **cumtime**: tiempo requerido en cada función incluyendo el tiempo de ejecución de cada subfunción de ésta.
- **percall (cumtime)**: mostrará el cociente del tiempo acumulado (cumtime) entre el número de llamadas (ncalls).
- **filename:lineno(function)**: nombre de las funciones sobre las que se hace el profiling, incluyendo el archivo en el que se encuentran.

8. 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).

Accederemos a phpMyAdmin introduciendo en el navegador la ruta `http://localhost/phpmyadmin/`; tras ello nos pedirá el usuario y la contraseña de la base de datos.

Nada más entrar encontraremos los ajustes generales, con los que podremos cambiar la contraseña, ajustar la apariencia de la web, etc. Seleccionaremos *Databases* en la barra superior, apareciéndonos cuatro bases de datos, que serán las que se encuentran alojadas en nuestro servidor.

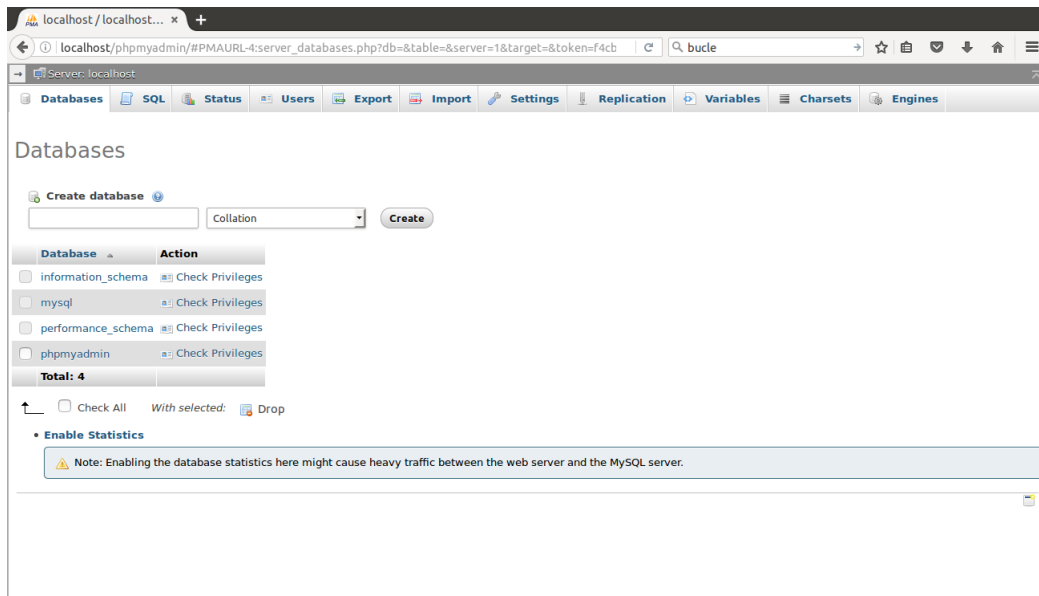


Figura 8.1: Bases de datos disponibles en phpMyAdmin.

Elegiremos la base de datos llamada “mysql” y realizaremos una consulta que muestre todos los elementos de la tabla “help_relation”.

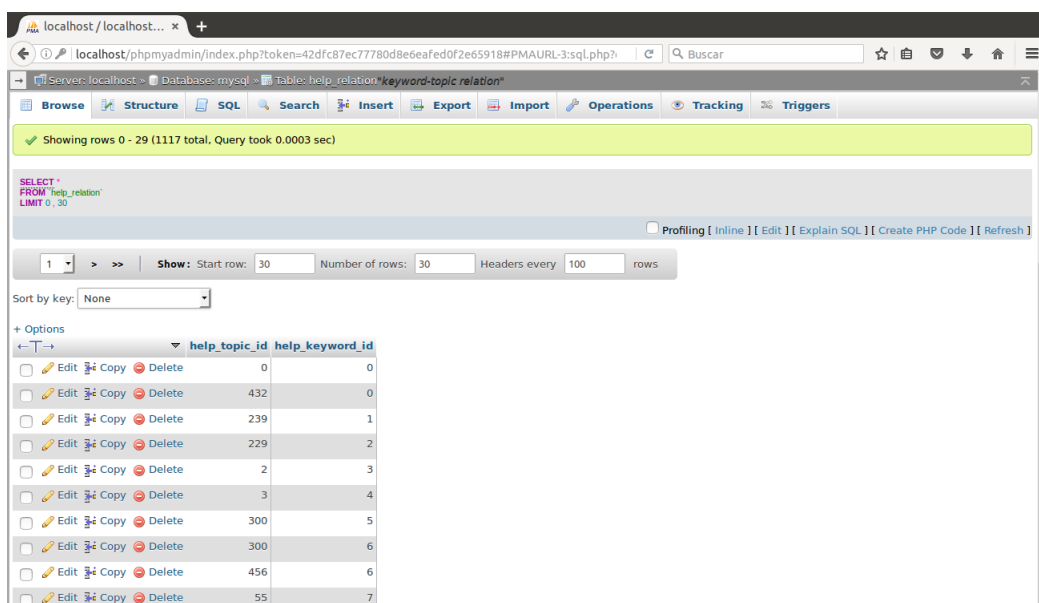


Figura 8.2: Consulta de los elementos de la tabla “help_relation”.

Si seleccionamos la casilla “Profiling”, obtendremos automáticamente una gráfica circular y un conjunto de datos del tiempo que tarda en realizarse cada una de las acciones para

obtener los resultados de la consulta.

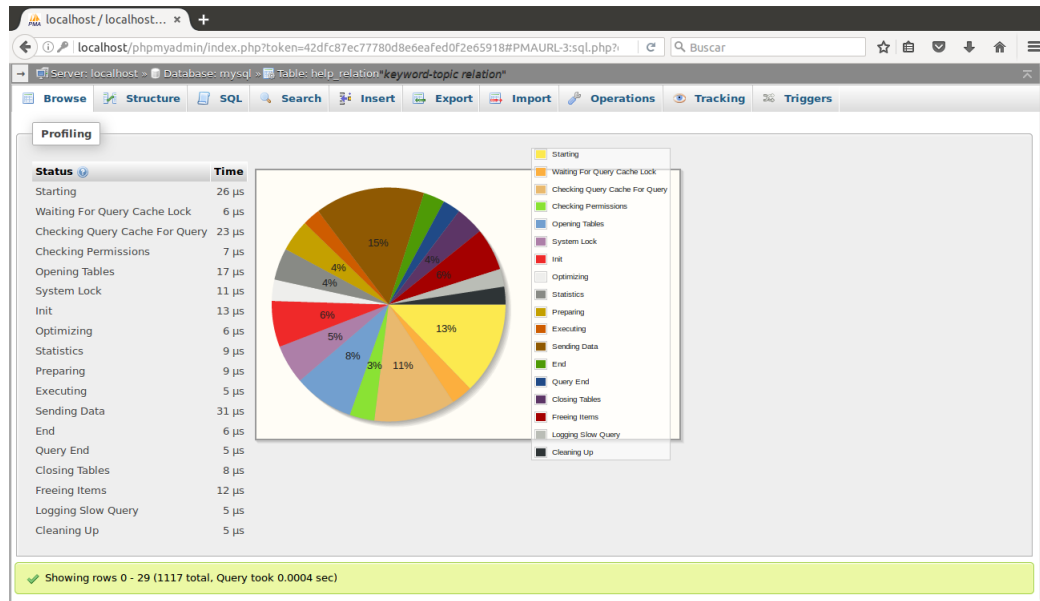


Figura 8.3: Datos y gráfica del *profiling* de la consulta.

De la imagen 8.3 podremos comentar que lo que más tiempo ha consumido para realizar la consulta ha sido el envío de datos (*Sending Data*), con un 15 % del tiempo total, debido a que se trata de una lectura en disco para obtener los elementos de la base de datos. Por detrás suya se encontrará el arranque (*Starting*) con un 13%; realizando *profiling* con otras consultas distintas comprobamos que el arranque suele estar siempre entre las operaciones que más porcentaje del tiempo total consumen.

Referencias

- [1] EL SISTEMA DE ARCHIVOS /VAR, <http://www.tldp.org/pub/Linux/docs/ldp-archived/system-admin-guide/translations/es/html/ch04s06.html>, consultado el 26 de abril de 2017.
- [2] MONITORIZACIÓN, <https://rm.aloxa.eu/projects/linux/wiki/Monitorizaci%C3%B3n>, consultado el 26 de abril de 2017.
- [3] LOGROTATE(8) - LINUX MAN PAGE, <https://linux.die.net/man/8/logrotate>, consultado el 27 de abril de 2017.
- [4] CÓMO: RECOPIRAR DATOS DE SEGUIMIENTO DE EVENTOS PARA WINDOWS (ETW), <https://msdn.microsoft.com/es-es/library/ms182379.aspx>, consultado el 27 de abril de 2017.
- [5] CREAR UN CONJUNTO DE RECOPIRADORES DE DATOS PARA SUPERVISAR LOS CONTADORES DE RENDIMIENTO, [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 29 de abril de 2017.
- [6] HERRAMIENTA ONLINE PARA COMPARAR TEXTOS (DIFF), <http://www.ddginc-usa.com/spanish/text-compare-tool.html>, consultado el 30 de abril de 2017.
- [7] FSCK(8) - LINUX MAN PAGE, <https://linux.die.net/man/8/fsck>, consultado el 30 de abril de 2017.
- [8] SYSADMIN TIPS AND TRICKS - USING STRACE TO MONITOR SYSTEM CALLS http://blog.softlayer.com/2013/sysadmin-tips-and-tricks-using-strace-to-monitor-system-calls#utm_source=twitter&utm_medium=social&utm_content=beyond-the-command-line-with-strace&utm_campaign=blog_development-tips-and-tricks, consultado el 2 de mayo de 2017.
- [9] STRACE(1) - LINUX MAN PAGE, <https://linux.die.net/man/1/strace>, consultado el 2 de mayo de 2017.
- [10] 26.4. THE PYTHON PROFILERS, <https://docs.python.org/2/library/profile.html>, consultado el 2 de mayo de 2017.
- [11] 6. MODULES, <https://docs.python.org/2/tutorial/modules.html>, consultado el 4 de mayo de 2017.
- [12] OVERVIEW, <http://demo.munin-monitoring.org/>, consultado el 4 de mayo de 2017.