Ingeniería de Servidores (2015-2016)
Grado en Ingeniería Informática
Universidad de Granada

Memoria Práctica 3

Marta Gómez Macías

2 de diciembre de 2015

Índice

Índice de figuras

1. a) ¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes? b) ¿Qué significan las terminaciones 1.gz o 2.gz de los archivos en ese directorio?

a) En el caso de CentOS, el archivo de log de yum, que se encuentra en */var/log/yum.log*. En la Figura 1(a) se ve, por ejemplo, que el día 30 de Octubre instalé el paquete `screen`.

En el caso de Ubuntu, el archivo correspondiente sería */var/log/apt/history.log*. En la Figura 2(b), se ve como, por ejemplo, el día 8 de Noviembre eliminé el paquete `python-pip`.

```
[marta@localhost ~]$ sudo cat /var/log/yum.log
[sudo] password for marta:
Oct 30 14:54:43 Installed: screen-4.1.0-0.19.20120314git3c2946.el7.x86_64
Nov 03 20:54:46 Installed: apr-1.4.8-3.el7.x86_64
Nov 03 20:54:47 Installed: apr-util-1.5.2-6.el7.x86_64
Nov 03 20:54:47 Installed: httpd-tools-2.4.6-31.el7.centos.1.x86_64
Nov 03 20:54:47 Installed: mailcap-2.1.41-2.el7.noarch
Nov 03 20:54:54 Installed: httpd-2.4.6-31.el7.centos.1.x86_64
Nov 04 15:47:25 Updated: 1:mariadb-libs-5.5.44-1.el7_1.x86_64
Nov 04 15:47:25 Installed: perl-Data-Dumper-2.145-3.el7.x86_64
Nov 04 15:47:28 Installed: 1:mariadb-5.5.44-1.el7_1.x86_64
Nov 04 15:47:28 Installed: perl-Compress-Raw-Bzip2-2.061-3.el7.x86_64
Nov 04 15:47:29 Installed: 1:perl-Compress-Raw-Zlib-2.061-4.el7.x86_64
Nov 04 15:47:29 Installed: perl-IO-Compress-2.061-2.el7.noarch
Nov 04 15:47:29 Installed: perl-Net-Daemon-0.48-5.el7.noarch
Nov 04 15:47:29 Installed: perl-PlRPC-0.2020-14.el7.noarch
Nov 04 15:47:30 Installed: perl-DBI-1.627-4.el7.x86_64
Nov 04 15:47:30 Installed: perl-DBD-MySQL-4.023-5.el7.x86_64
Nov 04 15:47:37 Installed: 1:mariadb-server-5.5.44-1.el7_1.x86_64
Nov 04 16:01:29 Installed: libzip-0.10.1-8.el7.x86_64
Nov 04 16:01:30 Installed: php-common-5.4.16-36.el7_1.x86_64
Nov 04 16:01:30 Installed: php-pdo-5.4.16-36.el7_1.x86_64
```

(a) Archivo bitácora de Yum, en el que vemos un histórico de todos los paquetes instalados, actualizados y eliminados

```
 GNU nano 2.2.6       Archivo: /var/log/apt/history.log

Start-Date: 2015-11-06  13:10:25
Commandline: apt-get install php5 libapache2-mod-php5 php5-mcrypt
Install: libapache2-mod-php5:amd64 (5.5.9+dfsg-1ubuntu4.14), php5-mcrypt:amd64 $
End-Date: 2015-11-06  13:10:33

Start-Date: 2015-11-06  13:11:08
Commandline: apt-get install phpMyAdmin
Install: libjs-jquery-mousewheel:amd64 (0-2, automatic), libjpeg-turbo8:amd64 ($
End-Date: 2015-11-06  13:13:07

Start-Date: 2015-11-08  00:56:58
Commandline: apt-get install python-pip
Install: fakeroot:amd64 (1.20-3ubuntu2, automatic), libstdc++-4.8-dev:amd64 (4.$
End-Date: 2015-11-08  00:57:32

Start-Date: 2015-11-08  00:59:08
Commandline: apt-get remove python-pip
Remove: python-pip:amd64 (1.5.4-1ubuntu3)
End-Date: 2015-11-08  00:59:09

Start-Date: 2015-11-08  01:00:19
Commandline: apt-get install -y ansible
Install: python-httplib2:amd64 (0.8-2build1, automatic), python-support:amd64 ($
End-Date: 2015-11-08  01:00:25
```

(b) Archivo bitácora de Apt, con el historial de todos los paquetes descargados, instalados, actualizados y eliminados

Figura 1.1: Consultando el historial de operaciones hechas con los gestores de paquetes

b) Tal y como se indica en [3], la existencia de dichos archivos se debe a que nuestro sistema está programado para llamar al comando `logrotate` cada x tiempo con `cron`. Estos archivos son antiguos logs que se han dejado ahí por si el administrador quiere consultarlos. Cada determinado tiempo, el archivo log se renombra y se crea uno nuevo, éste antiguo se comprime usando `gzip`. Así conseguimos que el archivo de logs no tenga un tamaño demasiado grande y podemos tener nuestros logs clasificados según el tiempo que queramos (por ejemplo, cada día se crea un nuevo archivo de log, y si queremos consultar los del día pasado consultaríamos el que tuviese la terminación `1.gz`).

2. ¿Qué archivo ha de modificar para programar una tarea? Escriba la líNa 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`).

Tal y como se indica en [11], los archivos *crontab* de cada usuario se encuentran en */var/spool/cron/crontabs/*. Sin embargo, no se recomienda editar directamente estos archivos, sino hacerlo a través del comando `crontab`. Además, como también se indica en [10], no es necesario reiniciar `cron` cuando modificamos el `crontab`.

Para hacer la tarea, hemos ejecutado con la opción `-e` para editar el archivo `crontab` del usuario actual

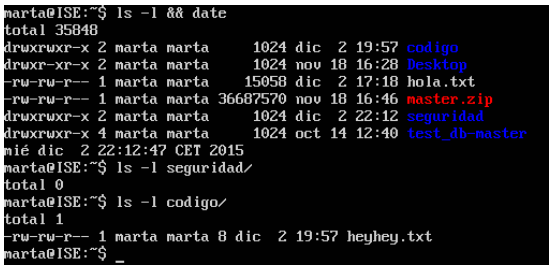
Llamando a crontab

```
$ crontab -e
```

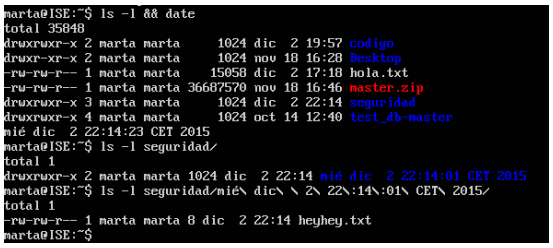
Tras esto, escribimos en el editor `nano` la siguiente línea:

```
14 22 * * * cp -R $HOME/codigo/ $HOME/seguridad/"$(date)"
```

En la Figura 2.1 se ven dos capturas de pantalla del contenido del directorio *home* del servidor, antes y después de que cron actúe. Como se ve, tras actuar cron se crea una carpeta con la fecha actual dentro del directorio *servidor* y con el mismo contenido que código.



(a) Comando `ls -l` ejecutado antes de la hora de acción de crontab



(b) Comando `ls -l` ejecutado después de la hora de acción de crontab

Figura 2.1: Consultando el historial de operaciones hechas con los gestores de paquetes

3. Pruebe a ejecutar el comando dmesg, 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.

Nada más iniciar el sistema y ejecutar `dmesg | tail`, he obtenido la información de la Figura 3.1. En ella nos muestra información sobre cómo configura la red WiFi al arrancar.

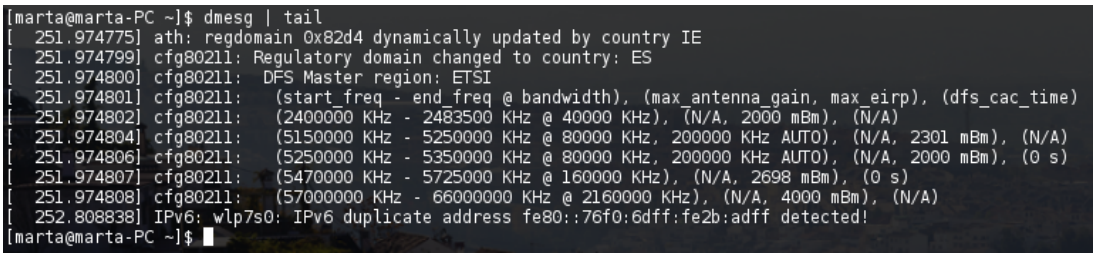


Figura 3.1: Información mostrada por `dmesg | tail` nada más iniciar el sistema

Tras conectar una memoria USB al ordenador y volver a ejecutar el comando `dmesg | tail`, vemos la información mostrada en la Figura 3.2. En la foto vemos cómo ha reconocido el pen drive y realiza las operaciones necesarias para montarlo en memoria, también nos lanza un aviso de que `umount` no funcionó correctamente sobre el sistema de archivos FAT. Sin embargo,

aún no habíamos desmontado la memoria USB. Tras desconectar la memoria USB y volver a ejecutar `dmesg | tail` obtenemos, además de la información de la Figura 3.2, la información de la Figura 3.3.

```
[marta@marta-PC ~]$ dmesg | tail
[  333.210830] usbcore: registered new interface driver usb-storage
[  333.212989] usbcore: registered new interface driver uas
[  334.209366] scsi 6:0:0:0: Direct-Access     Kingston DataTraveler 3.0 PMAP PQ: 0 ANSI: 6
[  334.211247] sd 6:0:0:0: [sdb] 30277632 512-byte logical blocks: (15.5 GB/14.4 GiB)
[  334.211845] sd 6:0:0:0: [sdb] Write Protect is off
[  334.211857] sd 6:0:0:0: [sdb] Mode Sense: 45 00 00 00
[  334.212467] sd 6:0:0:0: [sdb] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[  334.233046]  sdb: sdb1
[  334.236378] sd 6:0:0:0: [sdb] Attached SCSI removable disk
[  342.142282] FAT-fs (sdb1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
[marta@marta-PC ~]$
```

Figura 3.2: Información mostrada por `dmesg | tail` al conectar una memoria USB al ordenador

```
[  387.087340] sdb: detected capacity change from 15502147584 to 0
[  390.703700] usb 4-2: USB disconnect, device number 2
```

Figura 3.3: Información adicional añadida cuando ejecutamos `dmesg | tail` tras desconectar la memoria USB

4. Ejecute el monitor perfmon de "Systen Performance" y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.

El monitor nos genera un amplio informe sobre el rendimiento del sistema, el cual está dividido en distintas secciones. La primera parte se ve en la Figura 4.1, donde nos muestra el equipo, la fecha y la duración del análisis; un resumen de los datos obtenidos y un análisis del rendimiento del sistema. En nuestro caso, como sólo estábamos ejecutando el monitor nos da como resultado poca actividad en el sistema.

5. Cree un recopilador de datos definido por el usuario en perfmon (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. Intervalo de muestra de 15 segundos. Almacene el resultado en el directorio *Escritorio\logs*. Incluya capturas de pantalla de cada paso.

1. En primer lugar, para entrar al asistente, hacemos click derecho en el submenú ***Definidos por el usuario*** del menú ***Conjuntos de recopiladores de datos***. En dicho submenú seguimos la ruta **Nuevo** > **Conjunto de recopiladores de datos** (Figura 5.1). Alternativamente podemos hacer click en el botón de **Nuevo conjunto de recopiladores de datos**.

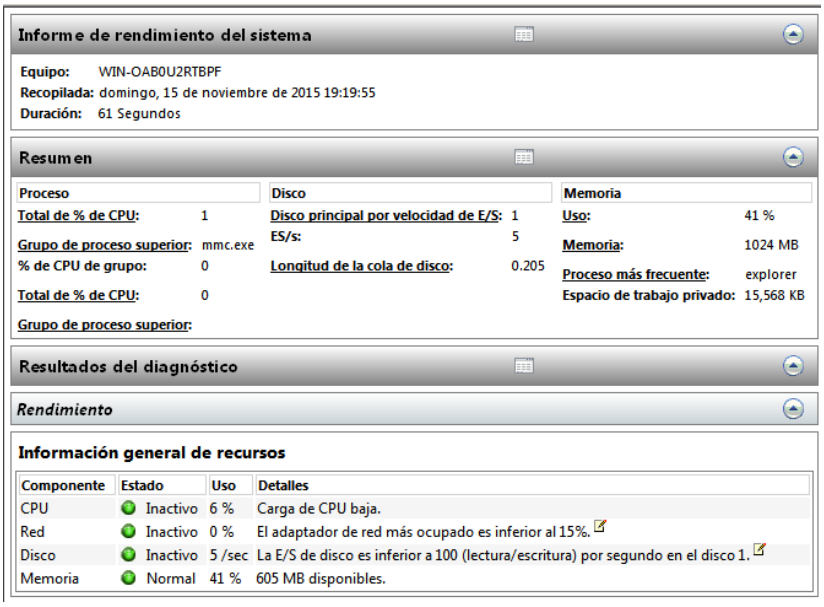


Figura 4.1: Resumen inicial de los datos obtenidos por el monitor

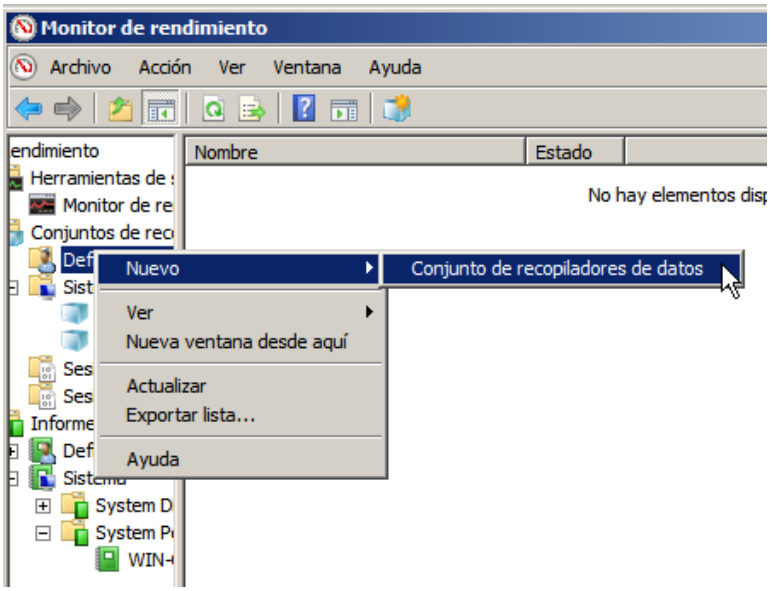


Figura 5.1: Ruta a seguir para llegar al asistente de creación de nuevo conjunto de recopiladores de datos

2. Tras esto, saldrá la ventana que vemos en la Figura 5.2. En ella, establecemos un nombre para nuestro Conjunto de recopiladores de datos y seleccionamos la opción manual.

3. Después, veremos la ventana de la Figura 5.3, en la cual seleccionaremos los datos que queremos recopilar. En nuestro caso seleccionamos los indicados en el guión de prácticas.

4. Tras darle a siguiente, nos saldrá una ventana en la que tendremos que hacer click en el botón de **Agregar**. Tras esto nos saldrá una ventana como la que se ve en la Figura 4(a). En nuestro caso, mediremos variables sobre el procesador, los procesos y el servicio web. Finalmente, cuando aceptemos volveremos a la ventana anterior, pero esta vez nos mostrará los contadores de rendimiento escogidos (Figura 4(b)). El intervalo de muestra por defecto

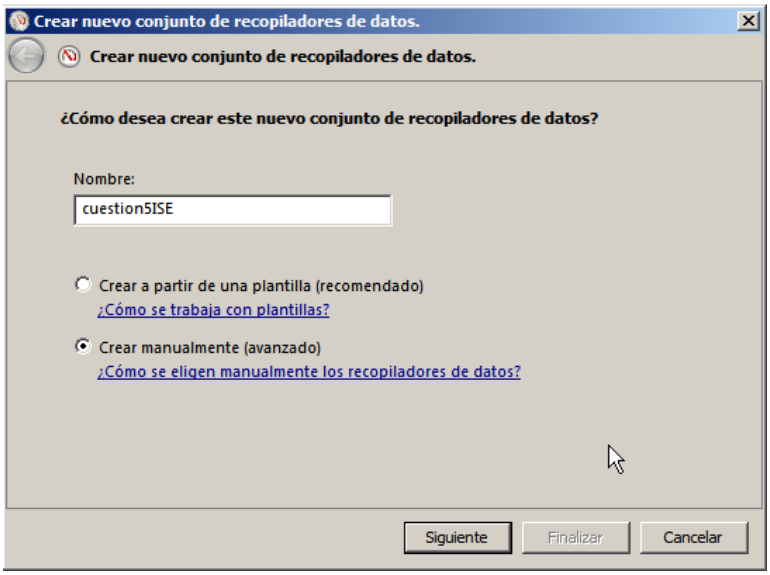


Figura 5.2: Primer paso para crear un conjunto de recopiladores de datos

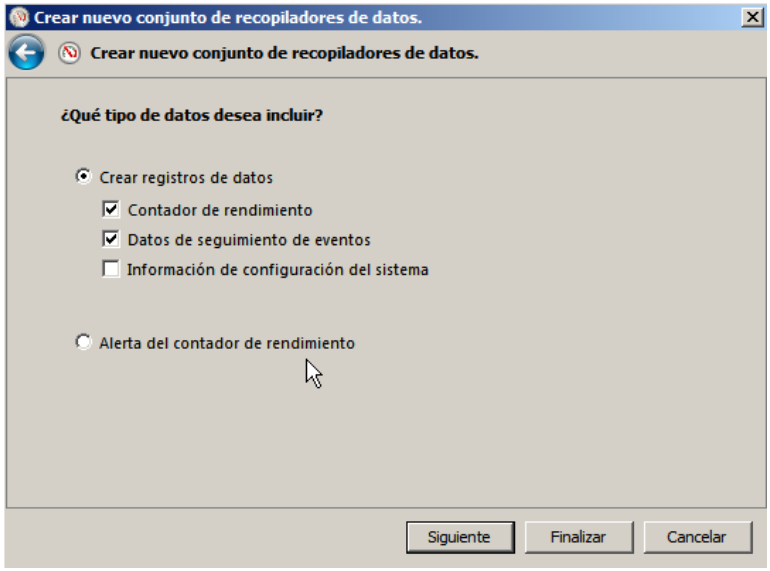


Figura 5.3: Selección de los datos que queremos recopilar

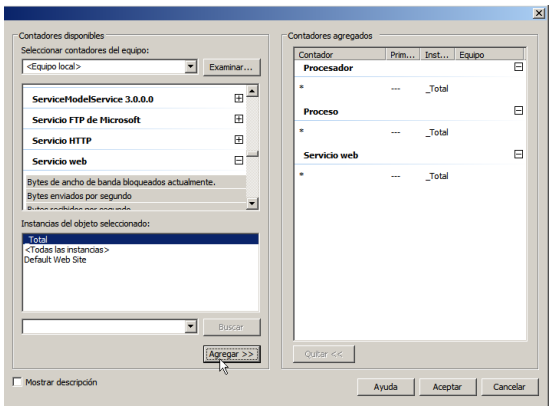
es el deseado en la práctica, por lo que no lo modificamos.

5. En la siguiente ventana nos preguntará sobre *Proveedores de seguimiento de eventos*. Como no se pide habilitar ninguno, dejamos todo por defecto y le damos a **Siguiente**.

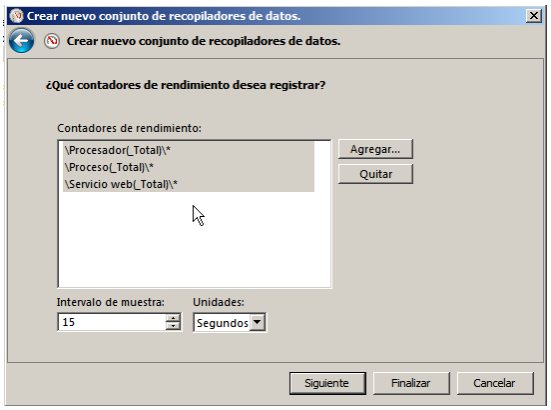
6. Después, debemos especificar dónde guardar los *logs*. En nuestro caso debe de ser en una carpeta llamada logs y ubicada en el escritorio (Figura 5.5).

7. Por último, en la siguiente ventana lo dejamos todo por defecto y hacemos click en **Finalizar**.

8. Una vez finalizado, hacemos click derecho sobre nuestro recopilador de datos y entramos al menú **Propiedades**. Ahí seleccionamos la pestaña **Detener condición** y establecemos la duración total a 1 minuto (Figura 5.6). Si no hacemos esto el recopilador de datos estará eternamente recopilando datos.



(a) Agregando variables a analizar a nuestro recopilador de datos



(b) Estableciendo los contadores de rendimiento y el intervalo de muestra

Figura 5.4: Ultimando los detalles del conjunto de recopiladores de datos creado.

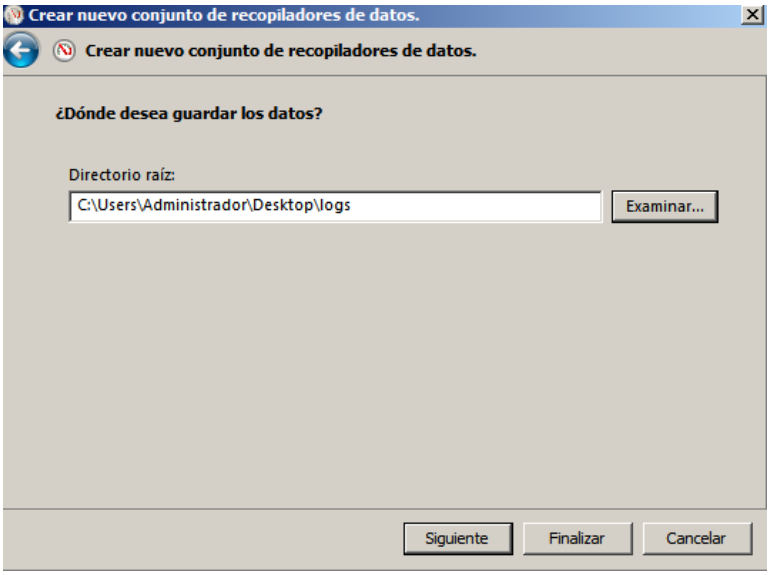


Figura 5.5: Estableciendo dónde almacenar los logs

Para ejecutar el recopilador de datos que hemos hecho, lo seleccionamos y hacemos click en el botón de **Iniciar**.

Tras finalizar la toma de datos, en la carpeta de *logs* ubicada en el escritorio podremos encontrar los datos obtenidos. El archivo final de datos obtenido se ve en la Figura 5.8. Poniendo el ratón sobre cada una de las líneas de la gráfica podemos ver con qué medida se corresponde. Así por ejemplo, el pico azul que se ve en la imagen se corresponde con las operaciones de E/S, en este caso ha habido un aumento del número de lecturas de ficheros sobre el disco duro sobre las 8:52 de la tarde. En cambio, el color fucsia se corresponde con el número de escrituras, que se ha reducido sobre esa misma hora.

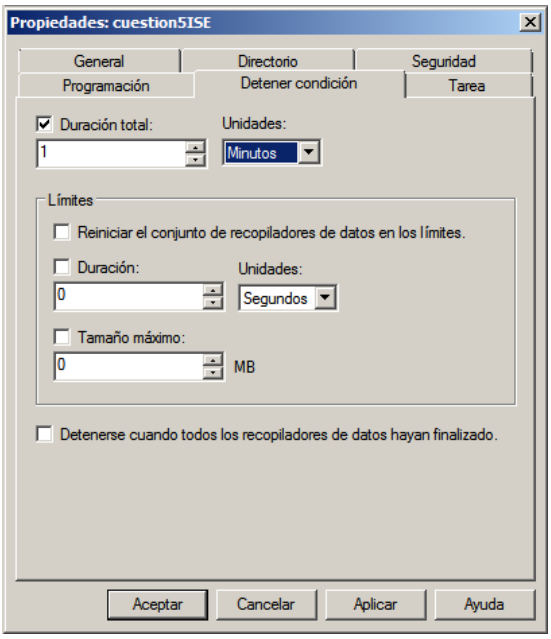


Figura 5.6: Estableciendo la duración total de la muestra

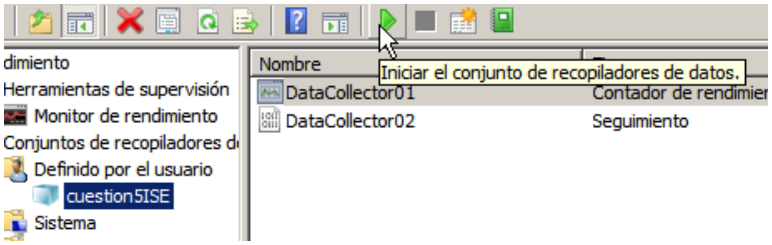


Figura 5.7: Iniciando el recopilador de datos.

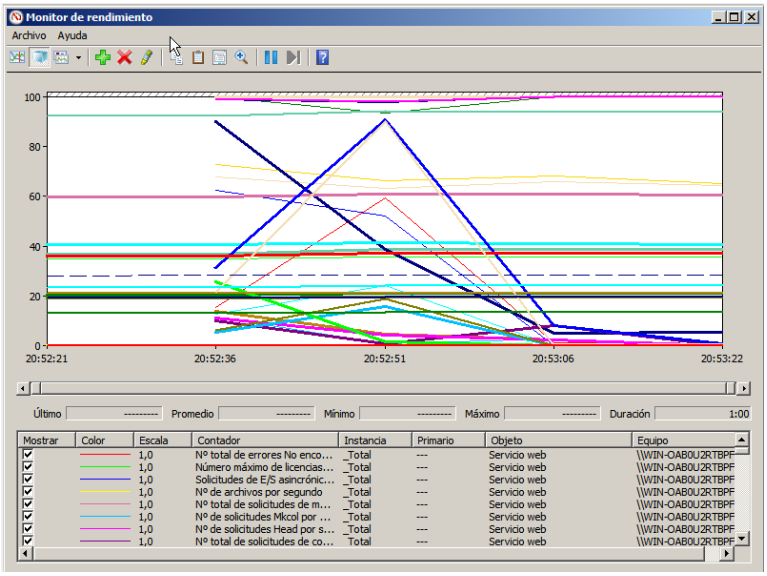


Figura 5.8: Analizando los datos recopilados

6. 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

6.1. Instalando y probando monitores hardware

6.1.1. `hddtemp`

En [1] se explica un uso básico de éste monitor. Su sintaxis requiere de, al menos, la ruta de un dispositivo de disco duro para monitorizar. Nosotros usaremos `/dev/sda1`. El resultado obtenido es el de la Figura 6.1, 35° que es una temperatura bastante buena.

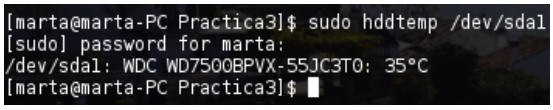
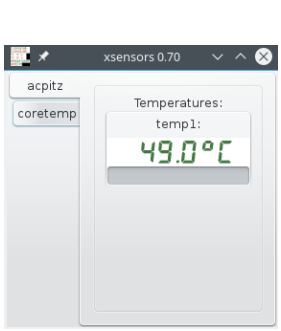


Figura 6.1: Resultado tras ejecutar `hddtemp` sobre el disco duro del ordenador

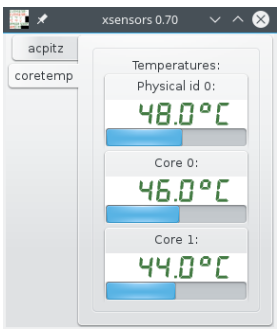
6.1.2. `xsensors`

Con `xsensors` podemos ver la temperatura de los distintos procesadores que tenemos en el ordenador cada segundo. Podemos cambiar el tiempo de refresco con la opción `-t`.

Al ejecutar `xsensors` obtenemos la ventana de la



(a) Pestaña en la que vemos la temperatura de la placa base en la zona de la CPU



(b) Pestaña en la que medimos la temperatura de los procesadores

Figura 6.2: Ejecutando `xsensors`

Al ejecutar el script `sensors-detect`, según [2], podemos detectar los distintos sensores que tienen incorporados nuestro hardware para realizar mediciones. En mi caso, como se ve en la Figura 6.3, sólo detecta los ya mostrados.

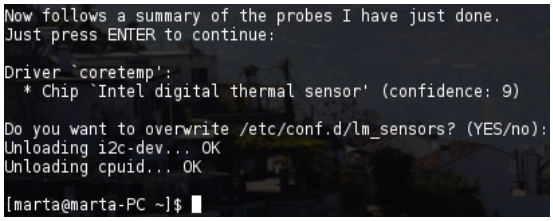


Figura 6.3: Resumen final de los sensores detectados

6.2. Otros monitores hardware

6.2.1. Smartmontools

En [14], se citan algunos monitores hardware y comandos para saber el hardware del que disponemos en Linux. Uno de los monitores de los que se habla es **Smartmontools**, que sirve para probar el correcto funcionamiento de nuestro disco duro, para ello ejecutan pruebas y leen datos que el propio disco guarda para su monitorización. Tiene una interfaz gráfica llamada *SmartControl*. Éste monitor también está disponible para Windows.

6.2.2. Hmonitor

Éste monitor sólo se encuentra disponible para Windows, en su página web encontramos una breve explicación de su funcionalidad:

- ★ Monitor de temperatura, que usa los sensores integrados en el hardware para saber su temperatura.
- ★ Alerta (o ejecución de una acción definida por el usuario) cuando algún componente alcanza una temperatura más alta de lo normal
- ★ Nos puede mostrar gráficos sobre la actividad del procesador y su temperatura con la utilidad *W2K PerfMon*.

7. Visite la web del proyecto *Munin* y acceda a la demo que proporcionan donde se muestra cómo monitorizan un servidor. Monitorice varios parámetros y haga capturas de pantalla de lo que está mostrando comentando qué observa.

En la demo, sólo podemos consultar valores ya monitorizados, pero no podemos escribir datos ni hacer nuevas monitorizaciones. Los datos que podemos consultar están en el menú de la izquierda puestos por categorías: podemos consultar problemas que haya encontrado Munin, los datos organizados según el servidor en el que se esté midiendo, los datos organizados según grupos (red, disco, logins, etc.). Éstos datos pueden consultarse según día, mes, año, etc. Ésta interfaz que describo se ve en la Figura 7.1. En este caso, Munin ha encontrado una advertencia que darnos, estamos midiendo datos en tres servidores distintos (munin-monitoring.org, vm y vpn) y tenemos varias categorias.

Por ejemplo, si hacemos click en la categoría *processes* vemos todos los datos relacionados con procesos que ha obtenido Munin que en este caso serían tasa de *fork* en procesos, número de hebras, prioridad de procesos, etc. Algunas de éstas gráficas de las que hablo se ven en

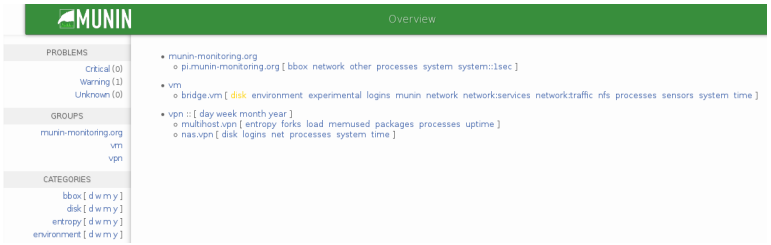


Figura 7.1: Pantalla inicial de Munin

la Subfigura 2(a). Cada gráfico corresponde a un servidor distinto, en este caso el primero corresponde al servidor munin-monitoring.org y el segundo, al vm. Para saber esto debemos hacer click en el gráfico que queramos consultar, donde accederemos a una página similar donde se indicará ya el servidor correspondiente. En la Subfigura 2(b) vemos la tasa de fork para el servidor munin-monitoring.org, tenemos varias gráficas según queramos consultarlo por año, por día, por mes o por semana.

(a) Algunos de los gráficos que vemos al acceder a la categoría de procesos

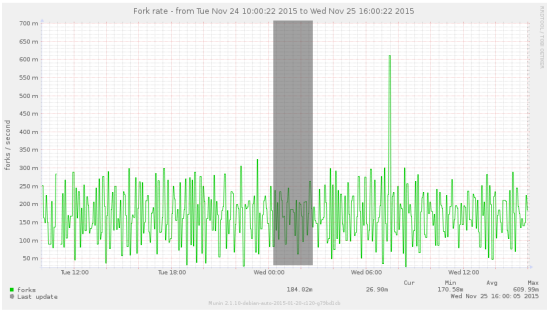
(b) Gráficos sobre la tasa de fork en el servidor munin-monitoring.org

Figura 7.2: Consultando datos concretos en Munin

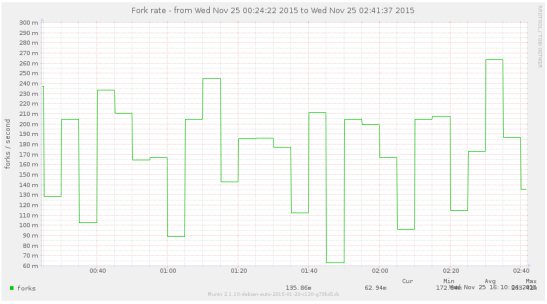
Si hacemos click en uno de los gráficos iremos a una página en la que podremos consultar el gráfico con detalle, ésto se ve en la Subfigura 3(a). Podemos interactuar con él eligiendo con tres clicks una zona en la que queramos hacer zoom, ésto se ve en las Subfiguras 3(b) y 3(c) de la Figura 7.3. Con el zoom podemos consultar los forks que hubo a una determinada hora (en este caso la media noche del miércoles). En este caso veríamos por ejemplo que sobre las 2:30 de la madrugada se ejecutó un proceso que tuvo muchos forks y sobre las 1:45 todo lo contrario. Observando la gráfica podemos decir que el número de forks que se hacen en el sistema es bastante cambiante.

En lo que a procesos se refiere, sólo tenemos medidas del servidor vm, éstas se ven en la Subfigura 4(a). Para monitorizar el número de procesos que ha habido en una semana hacemos click en la segunda gráfica y obtenemos la gráfica de la Subfigura 9(b), en dicha gráfica vemos unos huecos que nos resultan extraños, así que hacemos zoom en uno de ellos obteniendo la gráfica de la Subfigura 4(c). En ella vemos que, durante dos horas (18:00-20:00), o bien se apagó el servidor o bien se dejó de monitorizar porque no hay ningún proceso ejecutándose en el sistema. El resto del tiempo el número de procesos es bastante homogéneo, unos 200, de los cuales la mayoría son procesos suspendidos (*sleeping*) y unos 5 o 6 están ejecutándose.

(a) Página detallada del gráfico de forks por día



(b) Eligiendo la zona en la que queremos hacer zoom



(c) Cómo queda el gráfico tras hacer zoom en una zona

Figura 7.3: Interactuando con una gráfica

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

El artículo consultado ha sido [4], en él se explica la funcionalidad básica de `strace` y varios ejemplos de uso.

`strace` sirve para hacer un seguimiento de las llamadas al sistema que realiza un determinado proceso. Ésto sirve para detectar errores que no se reflejan en los archivos de logs. Sobre todo se usa para hacer seguimiento de llamadas al sistema bien conocidas tales como `open`, `close`, `read`, `write`, etc.

Para ver las llamadas al sistema realizadas al leer un archivo con `cat`, ejecutamos `strace` de la siguiente manera:

Ejecutando strace

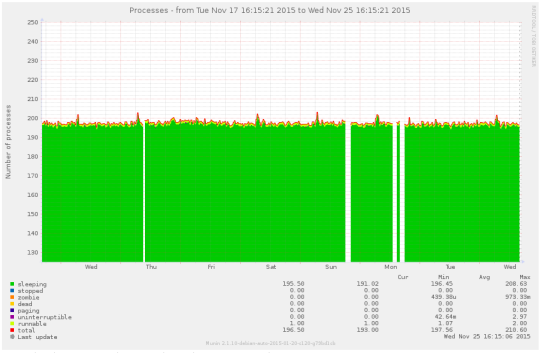
```
strace cat text.txt
```

En la salida de dicho comando veremos, en las primeras líneas las llamadas al sistema correspondientes para ejecutar el proceso `cat`, y después, veremos las llamadas al sistema que realiza `cat` para abrir el archivo e imprimirlo por la salida estándar.

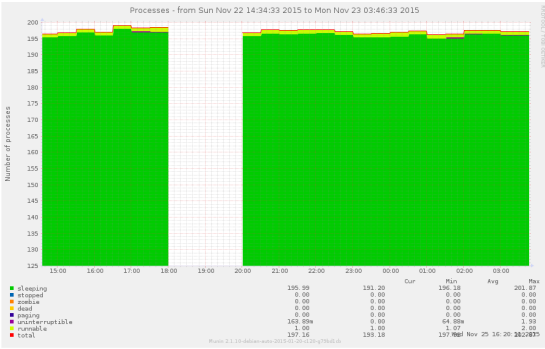
Tras explicar el ejemplo simple con `cat`, el artículo nos muestra algunos ejemplos más complejos e interesantes.

En el primero de ellos, nos muestra tres opciones de `strace`: la opción `-o`, la opción `-e` y la opción `-Ff`. La primera de ellas sirve para redirigir el output a un archivo, la segunda, para filtrar el output de tal forma que sólo veamos las llamadas al sistema `open` y la tercera, para seguir todos los procesos hijos que haga el proceso.

(a) Gráficas obtenidas al medir el número de procesos en el servidor vm



(b) Gráfica detalla sobre el número de procesos por semana



(c) Haciendo zoom sobre una zona del gráfico que nos ha resultado algo rara

Figura 7.4: Monitorizando el número de procesos

El segundo de ellos consiste en encontrar un fallo al iniciar un servidor Apache, dicho fallo consiste en que no puede abrir correctamente el archivo de logs, por lo que si no fuese usando `strace` no habría ninguna forma de encontrar dicho error.

Finalmente el artículo nos anima a probar `strace` mientras editamos un fichero de texto para ir viendo las llamadas al sistema "en directo" que vamos haciendo mientras escribimos. En la Figura 8.1 se ve mi prueba. Por cada carácter que introduzco, se hace una llamada a `read` y unas cuantas llamadas más adelante, se hace una llamada a `write`. Por último, el programa se queda esperando a que escriba otra tecla para realizar otra llamada más a `read`.

Al pulsar **Control+o** para guardar, se ha leído la tecla `"\17"` y nos imprime por pantalla "Nombre del fichero". (Figura 8.2).

Al presionar **Enter** para guardar el fichero, intenta abrir el fichero, al no poder encontrarlo lo crea con la llamada al sistema `open`. Tras ésto escribe en el fichero el texto que hemos introducido en él con `write` y por último, lo cierra con `close` (Figura 8.3).

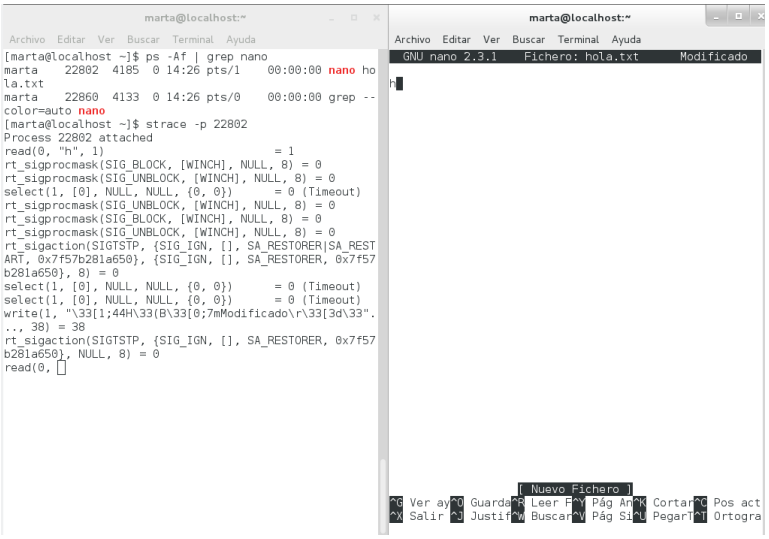


Figura 8.1: Haciendo la prueba con **strace** que nos sugiere el artículo

```
read(0, "\17", 1)                       = 1
rt_sigprocmask(SIG_BLOCK, [WINCH], NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [WINCH], NULL, 8) = 0
select(1, [0], NULL, NULL, {0, 0})      = 0 (Timeout)
rt_sigprocmask(SIG_UNBLOCK, [WINCH], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, [WINCH], NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [WINCH], NULL, 8) = 0
rt_sigaction(SIGTSTP, {SIG_IGN, [], SA_RESTORER|SA_REST
ART, 0x7f57b281a650}, {SIG_IGN, [], SA_RESTORER, 0x7f57
b281a650}, 8) = 0
select(1, [0], NULL, NULL, {0, 0})      = 0 (Timeout)
select(1, [0], NULL, NULL, {0, 0})      = 0 (Timeout)
write(1, "\r\33[33d\33(B\33[0;7mNombre del ficher"...,
274) = 274
rt_sigaction(SIGTSTP, {SIG_IGN, [], SA_RESTORER, 0x7f57
b281a650}, NULL, 8) = 0
```

Figura 8.2: Llamadas al sistema realizadas cuando vamos a guardar un fichero

```
read(0, "\r", 1)                        = 1
rt_sigprocmask(SIG_BLOCK, [WINCH], NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [WINCH], NULL, 8) = 0
select(1, [0], NULL, NULL, {0, 0})      = 0 (Timeout)
getcwd("/home/marta", 4097)             = 12
stat("hola.txt", 0x7ffdd04449c0)        = -1 ENOENT (No
 such file or directory)
getcwd("/home/marta", 4097)             = 12
stat("hola.txt", 0x7ffdd04449c0)        = -1 ENOENT (No
 such file or directory)
stat("/home/marta/hola.txt", 0x7ffdd0444ae0) = -1 ENOEN
T (No such file or directory)
lstat("hola.txt", 0x7ffdd04449a0)       = -1 ENOENT (No
 such file or directory)
stat("hola.txt", 0x7ffdd0444910)        = -1 ENOENT (No
 such file or directory)
umask(0)                                = 02
umask(02)                               = 0
open("hola.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
umask(02)                               = 02
fcntl(3, F_GETFL)                       = 0x8001 (flags
 O_WRONLY|O_LARGEFILE)
fstat(3, {st_mode=S_IFREG|0664, st_size=0, ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_
ANONYMOUS, -1, 0) = 0x7f57b3217000
write(3, "ho\n", 3)                     = 3
close(3)                                = 0
```

Figura 8.3: Llamadas al sistema realizadas cuando guardamos un fichero

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 líbremente)

Para activar el *profiling* debemos activar el *flag* `profiling` (ya que por defecto su valor es 0). Ésto se hace con el siguiente comando:

Activando el profiling en SQL

```
mysql> SET profiling = 1;
```

Así, habilitamos el profiling de cada consulta que hagamos y que luego podremos consultar con el comando `SHOW PROFILES`.

La base de datos usada es la misma que se usó para importar en la sesión anterior ([5]). Para poder instalarla usamos el siguiente comando:

Instalando la base de datos test_db

```
mysql -h localhost -u root -p < employees.sql
```

La parte de la izquierda nos sirve para conectarnos a nuestra base de datos ([6]) y la de la derecha, para instalar la base de datos de empleados.

Una vez hecho eso, entramos en la consola de MySQL, habilitamos el *profiling* e indicamos a SQL que usaremos la base de datos recién instalada. Para ello, según [7], usamos el siguiente comando:

Indicando a SQL que use la base de datos employees

```
mysql> USE employees;
```

En caso de no saber cómo se llama la base de datos recién instalada, podemos usar el siguiente comando ([8]):

Consultando las bases de datos que tenemos

```
mysql> SHOW DATABASES;
```

La consulta usada, en base a los datos de empleados y su estructura, se basará en los empleados cuyo nombre sea *Perry*. Así, obtenemos una consulta rápida (hacer una consulta sin ningún filtro tardaría demasiado tiempo):

Haciendo una consulta a la base de datos

```
mysql> SELECT * FROM employees WHERE first_name="Perry";
```

Tras todo esto, ejecutamos el profiling y obtenemos la salida que se ve en la Figura 9.1:

Ejecutando el profiler de MySQL

```
mysql> SHOW PROFILES;
```

Si comparamos cada tiempo que nos proporciona el *profiler* con el tiempo que MySQL nos ha indicado vemos que ambos coinciden, aunque el tiempo indicado por el *profiler* es más exacto. Por ejemplo, al finalizar la consulta a la base de datos de empleados, MySQL nos indicaba un tiempo de 0.58 segundos. Sin embargo, dicho número es redondeado, pues en realidad han sido 0.57862550 segundos.

```
mysql> SHOW PROFILES;
+----------+------------+-------------------------------------------------------+
| Query_ID | Duration   | Query                                                 |
+----------+------------+-------------------------------------------------------+
|        1 | 0.02184350 | SHOW DATABASES                                        |
|        2 | 0.00019475 | SELECT DATABASE()                                     |
|        3 | 0.00054350 | show databases                                        |
|        4 | 0.00035050 | show tables                                           |
|        5 | 0.57862550 | SELECT * FROM employees WHERE first_name="Perry"      |
+----------+------------+-------------------------------------------------------+
5 rows in set (0.00 sec)
```

Figura 9.1: Consultando los *profiles* de los últimos comandos ejecutados en MySQL

10. Cuestiones Opcionales

10.1. Indique qué comandos ha utilizado para realizar la sustitución del disco RAID1 dañado por uno nuevo así como capturas de pantalla del proceso de reconstrucción del RAID.

Para hacerlo vamos a seguir los pasos indicados en el guión de prácticas:

1. En primer lugar, vamos a la configuración de la máquina virtual y añadimos un segundo disco. Nos tiene que quedar como se ve en la Figura 10.1.

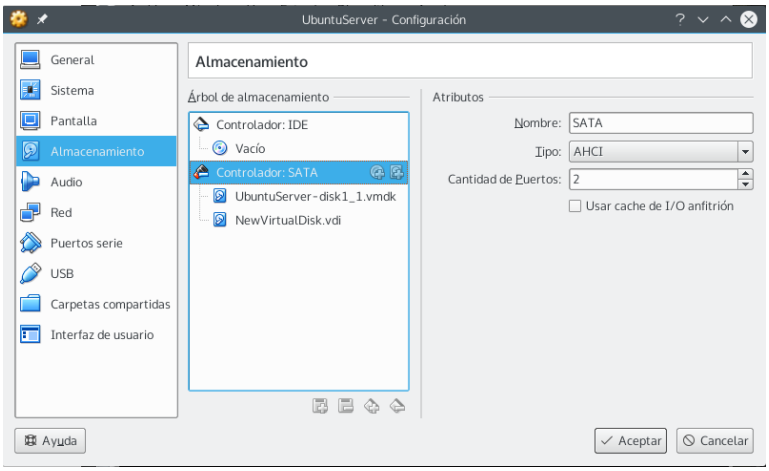


Figura 10.1: Configuración de la máquina virtual tras añadir un nuevo disco

Tras añadir el disco, veremos el mensaje que se ve en la Figura 10.2.

```
marta@ISE:~$ [  577.737061] ata4: exception Emask 0x10 SAct 0x0 SErr 0x4010000 a
ction 0xe frozen
[  577.737274] ata4: irq_stat 0x80400040, connection status changed
[  577.737352] ata4: SError: { PHYRdyChg DevExch }
```

Figura 10.2: Mensaje tras añadir un nuevo disco duro en caliente

2. Usamos el comando `mdadm` para eliminar el disco defectuoso del RAID y añadir el nuevo. Consultando [12], vemos que para eliminar dispositivos del RAID podemos usar la opción `-remove` y para añadir, la opción `-add`.

   Primero, tenemos que consultar los discos que tenemos en nuestro ordenador, para ello usamos el comando `lsblk`. Obtenemos el output de la Figura 10.3, en el cual vemos que

el disco actual es el `sda1` y el que hemos insertado nuevo, `sdb`. También hemos consultado todos los dispositivos que tenemos en el sistema para saber cuál eliminar, en este caso sería `sda`.

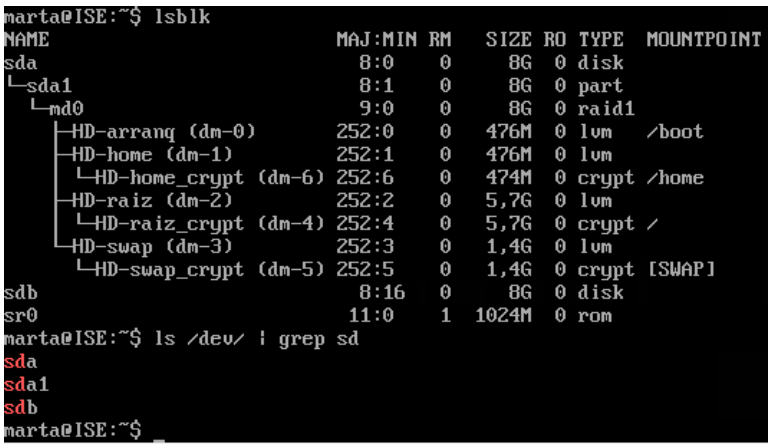


Figura 10.3: Consultando cómo está el disco particionado y los discos que tenemos.

Con esa información, ejecutamos el comando `mdadm`:

Eliminando el disco defectuoso y añadiendo el nuevo

```
sudo mdadm /dev/md0 --add /dev/sdb --remove /dev/sda
```

Tras ejecutarlo, nos dice que se ha añadido con éxito el disco que hemos añadido nuevo, pero que no se ha encontrado el disco que íbamos a eliminar. Aún así, vemos que la ejecución ha sido exitosa en la Figura 10.4.

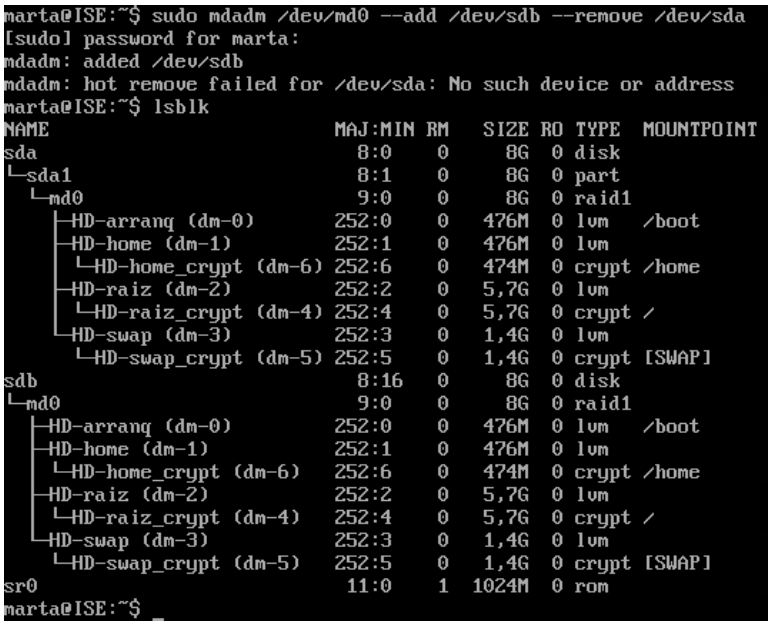


Figura 10.4: Consultando cómo ha quedado particionado el disco tras añadir el disco nuevo al RAID

10.2. Instale *Nagios* en su sistema (el que prefiera) documentando el proceso y muestre el resultado de la monitorización de su sistema comentando qué aparece.

10.2.1. Instalación

En [9] se documenta paso a paso cómo instalar *Nagios* en CentOS. Los pasos a seguir son los siguientes:

1. En primer lugar, nos vamos al directorio temporal para trabajar desde ahí:

   Accediendo al directorio /tmp
   ```
   cd /tmp
   ```

2. Descargamos la última versión estable de *Nagios*:

   Descargando la última versión estable de Nagios
   ```
   wget http://assets.nagios.com/downloads/nagiosxi/xi-latest.tar.gz
   ```

3. Extraemos el archivo que hemos descargado:

   Extrayendo el contenido del archivo descargado
   ```
   tar xzf xi-latest.tar.gz
   ```

4. Una vez extraído el archivo vamos al directorio de nagios:

   Accediendo al directorio de nagios
   ```
   cd nagiosxi/
   ```

5. Tras esto ejecutamos el script de instalación:

   Ejecutando el script de instalación
   ```
   sudo ./fullinstall
   ```

   A mitad de instalación nos pedirá la contraseña para MySQL.

6. Una vez instalado, para acceder a nagios basta con acceder a la dirección `localhost` o a la IP de nuestro servidor:

   Accediendo a nagios desde Firefox
   ```
   firefox http://10.0.2.10/
   ```

   Una vez hecho eso, veremos la página de la Figura 10.5.

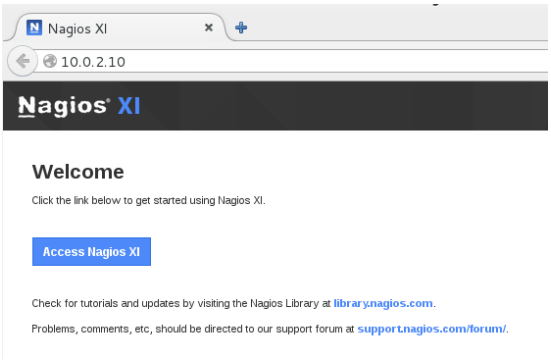


Figura 10.5: Página inicial de Nagios

7. Hacemos click en *Access Nagios* y nos saldrá una ventana como la de la Figura 10.6 en la que debemos establecer algunos parámetros del sistema antes de empezar a usarlo.

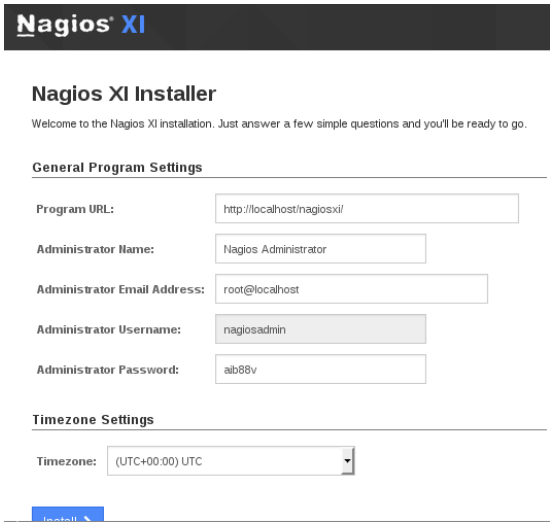


Figura 10.6: Modificando algunos parámetros de Nagios antes de iniciar el sistema por primera vez

8. Por último nos mostrará una ventana con nuestro nombre de usuario y contraseña para acceder al sistema. Haciendo click en *Login Nagios XI* e introduciendo el nombre de usuario y contraseña proporcionados, ya habremos terminado la instalación.

9. Tras acceder al sistema deberemos de aceptar la licencia y tras aceptar la licencia nos dará un pequeño tutorial inicial. La interfaz inicial de *Nagios* se ve en la Figura 10.7.

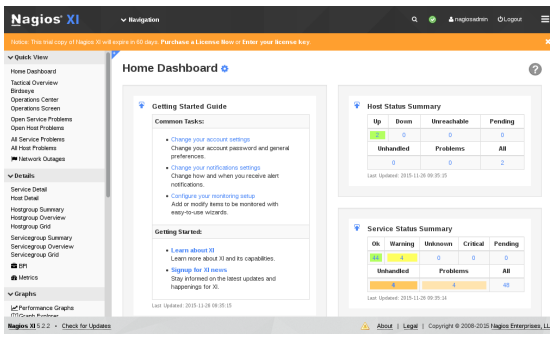


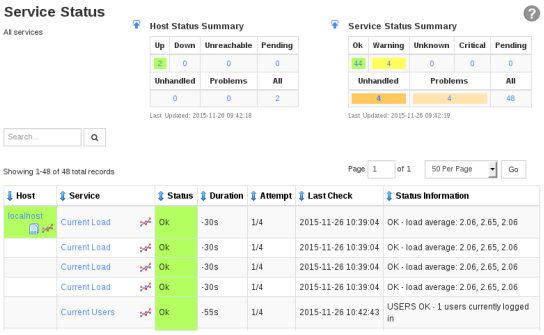
Figura 10.7: Interfaz de Nagios al acceder al sistema

10.2.2. Monitorización del sistema

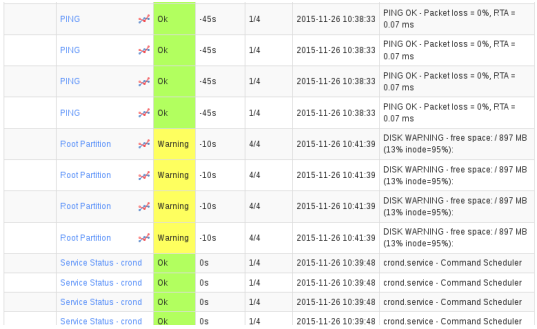
En el menú de la izquierda, tenemos algunas de las operaciones que podemos realizar con *Nagios*. Por ejemplo, en el submenú *Service Detail*, podemos comprobar el estado de cada uno de los servicios que tenemos a activos en el sistema, incluyendo el propio *Nagios* y el estado del propio sistema. En la Figura 10.8 se ve el ejemplo realizado por nosotros, en el que obtenemos varias advertencias sobre el poco espacio en disco que nos queda.

En el submenú *Metrics*, podemos obtener gráficas y datos sobre el uso de CPU, de memoria, de disco, de swap y la carga del sistema. En la Figura 10.9 estamos monitorizando el uso de la memoria de intercambio en el sistema, que en este caso vemos que tiene un uso aceptable y que no está saturada ya que tenemos un 54 % de memoria libre.

En el submenú *Graph Explorer*, podemos consultar algunos gráficos realizados por *Nagios* sobre nuestro servidor y nuestro sistema. En la Figura 10(a) se muestra un gráfico que muestra

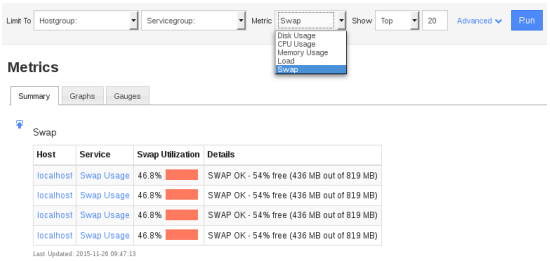


(a) Resumen del estado de los servicios del sistema

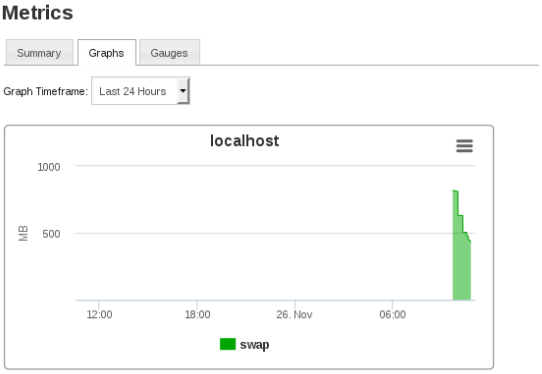


(b) Parte de la tabla detallada sobre cada servicio con varias advertencias

Figura 10.8: Monitorizando el estado de los servicios del sistema y del propio sistema

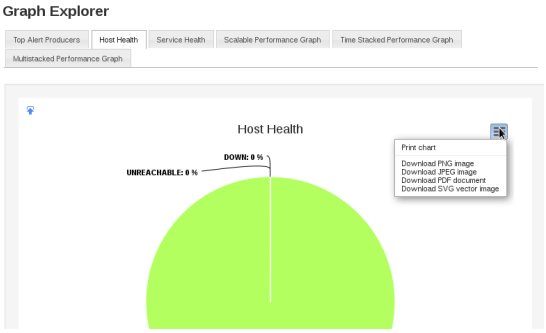


(a) Resumen del uso de swap en el sistema

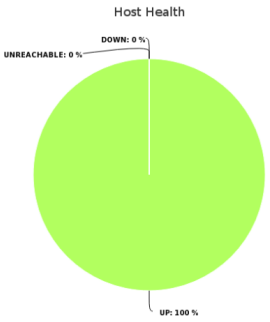


(b) Gráfica del uso de swap en el sistema

Figura 10.9: Monitorizando el uso de swap en el sistema



(a) Gráfico resumen de las veces que nuestro sistema no ha sido accesible



(b) Gráfico que hemos descargado en formato PNG sobre la "salud" de nuestro sistema

Figura 10.10: Monitorizando la "salud" del sistema

el número de veces que nuestro servidor ha estado innacesible (o bien porque se ha saturado, o porque se ha perdido la conexión, etc). En este caso, nuestro servidor ha estado siempre accesible y por eso muestra toda la gráfica en color verde. Además, *Nagios* nos permite descargar el gráfico en varios formatos, por ejemplo en la Figura 10(b) se ve el archivo PNG de la gráfica

descargado directamente desde *Nagios*.

10.3. Acceda a la demo online de *Ganglia* de WikiMedia y haga lo mismo que hizo con *Munin*

En WikiMedia nos ofrecen una demo online del profiler *Ganglia*, mostrándonos datos reales de sus servidores.

Nada más entrar a la página, nos encontramos con la página que se ve en la Figura 10.11 vemos un resumen estadístico del grid que usa WikiMedia, con el número total de CPUs y servidores tanto activos como no activos. También nos ofrecen una estadística sobre la carga desde los últimos 15 minutos y la utilización media. A la derecha de éstos datos vemos cuatro gráficas con datos sobre la carga del sistema, el uso de memoria, de CPU y de red. El uso de memoria sí es algo alto (más de la mitad), pero tanto la carga del sistema como el uso de CPU es bastante bajo. El uso de red tiene picos que varían bastante, por ejemplo, desde las 20:40 hasta las 21:00 no ha tenido actividad ninguna, sin embargo, sí ha tenido un pico de actividad a las 9.

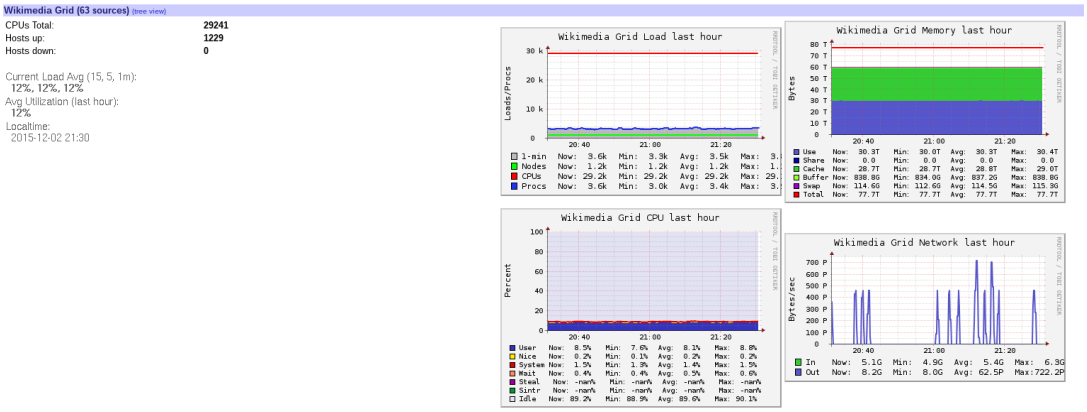


Figura 10.11: Información general del grid de WikiMedia

Después de esto, nos encontramos información sobre cada cluster de computadores de los que dispone WikiMedia (*codfw* y *equiad*). Si hacemos click en una de las gráficas, accedemos a una página en la que nos muestra información general sobre ese clúster, pero, dentro de dicha página podemos elegir ver información individual sobre un nodo concreto del clúster. Ésto se ve en la Figura 10.12. Si elegimos un *host*, vemos que tiene unas gráficas de rendimiento bastate parecidas a las del cluster (Figura 10.13).

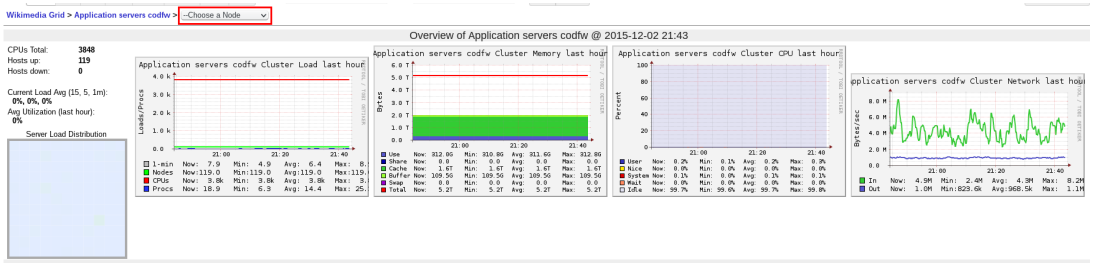


Figura 10.12: Información general sobre los servidores de aplicación del clúster codfw

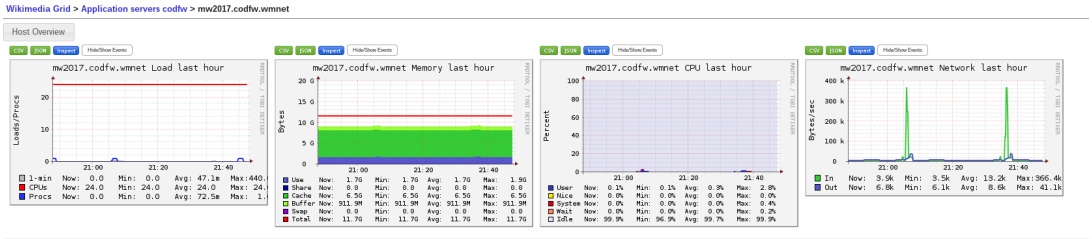


Figura 10.13: Información general sobre uno de los hosts del clúster codfw

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

El script realizado consiste en cifrar usando el código de Polibio [15]. El script en cuestión es el siguiente:

polibio.py

```
tabla_polibio = [['a','b','c','d','e'],['f','g','h','i','k'],['l','m','n','o','p'],
['q','r','s','t','u'],['v','w','x','y','z']]

def cifra_polibio (frase):
    cifrado = []
    for caracter in frase:
        for lista in tabla_polibio:
            for letra in lista:
                if letra == caracter:
                    cifrado.append((tabla_polibio.index(lista)+1)*10+(lista.index(letra)+1))

    return cifrado

def descrifrado_polibio (cifrado):
    frase = ""

    for numero in cifrado:
        frase += tabla_polibio[int(numero/10)-1][int(numero%10)-1]

    return frase

if __name__ == '__main__':
    frase_original = input("Introduce una frase: ")
    frase = frase_original.replace("j", "i");

    cifrado = cifra_polibio(frase)

    print("Texto cifrado: "+' '.join(str(n) for n in cifrado))

    descrifrado = descrifrado_polibio(cifrado)

    print("Texto descrifrado: "+descrifrado)
```

Para hacer el profiling he usado la librería *cProfile*, la cual, según [13], se usa con el siguiente comando:

Haciendo profile a un script en Python

```
python -m cProfile polibio.py
```

de forma análoga también podemos usar la librería *profile*, con la cual obtenemos también la misma salida.

Tras ejecutar el comando anterior, he obtenido la salida que se ve en la Figura 10.14. En primer lugar, el *profiler* nos muestra el número de llamadas a funciones que ha realizado nuestro programa y el tiempo total de ejecución. El tiempo es tan alto debido a que también cuenta el tiempo que el programa ha estado dormido esperando la entrada desde teclado.

Por último, nos muestra cada una de las funciones a las que ha llamado nuestro programa junto al número de veces que se ha llamado a cada función. Por ejemplo, las dos funciones que incluye el script (`descrifrado_colibio` y `cifra_polibio`) se llaman una única vez cada una, como debe de ser pues en el script sólo se hace una llamada a cada función. También se reflejan llamadas a métodos primitivos de Python tales como `index`, `append`, etc los cuales se llaman más veces debido a que se llaman dentro de bucles `for`, en concreto, números relacionados con el tamaño de la frase a cifrar introducida.

```
[marta@marta-PC Practica3]$ python -m cProfile polibio.py
Introduce una frase: el poder desgasta al que no lo tiene
Texto cifrado: 15 31 35 34 14 15 42 14 15 43 22 11 43 44 11 11 31 41 45 15 33 34 31 34 44 24 15 33 15
Texto descrifrado: elpoderdesgastaalquenolotiene
         127 function calls in 5.551 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    5.551    5.551 polibio.py:1(<module>)
        1    0.000    0.000    0.000    0.000 polibio.py:14(descrifrado_polibio)
       30    0.000    0.000    0.000    0.000 polibio.py:28(<genexpr>)
        1    0.000    0.000    0.000    0.000 polibio.py:4(cifra_polibio)
        1    0.000    0.000    5.551    5.551 {built-in method builtins.exec}
        1    5.550    5.550    5.550    5.550 {built-in method builtins.input}
        2    0.000    0.000    0.000    0.000 {built-in method builtins.print}
       29    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
       58    0.000    0.000    0.000    0.000 {method 'index' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'join' of 'str' objects}
        1    0.000    0.000    0.000    0.000 {method 'replace' of 'str' objects}
```

Figura 10.14: Salida obtenida tras ejecutar *cProfile* sobre un script Python

Referencias

[1] ArchWiki, *Hddtemp*. *Disponible en https://wiki.archlinux.org/index.php/Hddtemp*. Consultado el 19/11/2015.

[2] ——, *lm_sensors*. *Disponible en https://wiki.archlinux.org/index.php/Lm_sensors*. Consultado el 23/11/2015.

[3] U. Documentation, *Linuxlogfiles*. *Disponible en https://help.ubuntu.com/community/LinuxLogFiles#Log_Rotation*. Consultado el 12/11/2015.

[4] L. Latham, *Sysadmin tips and tricks: Using strace to monitor system calls*. *Disponible en http://blog.softlayer.com/2013/sysadmin-tips-and-tricks-using-strace-to-monitor-system-calls*. Consultado el 27/11/2015.

[5] G. Maxia, *test_db*. *Disponible en https://github.com/datacharmer/test_db/*. Consultado el 30/11/2015.

[6] MySQL, *Connecting to the mysql server. Disponible en https://dev.mysql.com/doc/refman/5.7/en/connecting.html*. Consultado el 30/11/2015.

[7] ——, *Creating and selecting a database. Disponible en https://dev.mysql.com/doc/refman/5.7/en/creating-database.html*. Consultado el 30/11/2015.

[8] ——, *Show databases syntax. Disponible en https://dev.mysql.com/doc/refman/5.7/en/show-databases.html*. Consultado el 30/11/2015.

[9] Nagios, *Nagios xi – manual installation instructions. Disponible en https://assets.nagios.com/downloads/nagiosxi/docs/XI_Manual_Installation_Instructions.pdf*. Consultado el 25/11/2015.

[10] L. M. Pages, *cron: daemon to execute scheduled commands. Disponible en http://linux.die.net/man/8/cron*. Consultado el 13/11/2015.

[11] ——, *crontab. Disponible en http://linux.die.net/man/1/crontab*. Consultado el 12/11/2015.

[12] ——, *mdadm: Manage md devices aka software raid. Disponible en http://linux.die.net/man/8/mdadm*. Consultado el 12/11/2015.

[13] Python, *The python profilers. Disponible en https://docs.python.org/2/library/profile.html*. Consultado el 29/11/2015.

[14] C. Schroder, *Discovering and monitoring hardware on linux. Disponible en http://www.linux.com/learn/tutorials/620416:discovering-and-monitoring-hardware-in-linux-*. Consultado el 23/11/2015.

[15] Wikipedia, *Cuadrado de polibio. Disponible en https://es.wikipedia.org/wiki/Cuadrado_de_Polibio*. Consultado el 29/11/2015.