

Ingeniería de Servidores (2014-2015) Grupo: B3
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Memoria Práctica 3

Carlos de la Torre Fanin

23 de noviembre de 2014

Índice

1. ¿Qué archivo le permite ver qué programas se han instalado con el gestor de paquetes?	1
2. ¿Qué significan las terminaciones .1.gz o .2.gz de los archivos en ese directorio?	1
2.1. Indique qué comandos ha utilizado para realizarlo así como capturas de pantalla del proceso de reconstrucción del RAID.	1
3. ¿Qué archivo ha de modificar para programar una tarea?	1
4. Pruebe a ejecutar el comando, conectar un dispositivo USB y vuelva a ejecutar el comando.	1
5. Ejecute el monitor de “System Performance” y muestre el resultado. Incluya capturas de pantalla comentando la información que aparece.	2
6. Cree un recopilador de datos definido por el usuario (modo avanzado) que incluya tanto el contador de rendimiento como los datos de seguimiento:	3
7. 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.	5
8. 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.	6
8.1. instale Nagios en su sistema (el que prefiera) documentando el proceso y muestre el resultado de la monitorización de su sistema comentando qué aparece.	7
8.2. Haga lo mismo que con Munin.	8
8.3. Pruebe a instalar este monitor en alguno de sus tres sistemas. Realice capturas de pantalla del proceso de instalación y comente capturas de pantalla del programa en ejecución.	9
8.4. Pruebe a instalar este monitor en alguno de sus tres sistemas. Realice capturas de pantalla del proceso de instalación y comente capturas de pantalla del programa en ejecución.	11
8.5. Instale el monitor, muestre y comente algunas capturas de pantalla.	12
9. Escriba un breve resumen sobre alguno de los artículos donde se muestra el uso de strace o busque otro y coméntelo.	12
9.1. Desarrolle una página en C o C++ y analice su comportamiento usando valgrind. Visite ¹ para ver un ejemplo sencillo de una página web generada por un programa escrito en C.	12
9.2. Desarrolle un script en PHP y analice su ejecución con alguno (o los dos) profilers.	16
9.3. Escriba un script en python y analice su comportamiento usando el profiler presentado.	16
9.4. Escriba un script en PowerShell y analice su comportamiento usando el profiler presentado.	16

¹<http://www.cs.tut.fi/~jkorpela/forms/cgic.html>

10. 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).	17
10.1. Al igual que ha realizado el "profiling" con MySQL, realice lo mismo con MongoDB y compare los resultados (use la misma información y la misma consulta, hay traductores de consultas SQL a Mongo).	17

Índice de figuras

5.1. Gráficas de datos en el momento que las captura Windows®	3
6.1. Pasos a seguir para crear un recolector de datos	5
7.1. Diferentes monitores de Hardware	5
8.1. Capturas de pantalla de Munin project	6
8.2. Nagios Core	8
8.3. Capturas de pantalla de Ganglia project	9
8.4. Interfaz Web de Instalación y ejecución de Zabbix	10
8.5. Interfaz Web de Instalación y ejecución de Zabbix	12

Índice de tablas

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

Siguiendo un razonamiento lógico y conociendo la estructura del sistema de ficheros de linux podemos llegar a la conclusión de que si lo que buscamos es un fichero que tiene que tener registrados los pasos que damos a la hora de instalar y desinstalar paquetes en las diferentes distribuciones podemos empezar buscando en el directorio `/var/log` de cualquiera de los dos sistemas operativos que tenemos instalados para las practicas.

En el caso de Ubuntu® el fichero que buscamos es `/var/log/dpkg.log`, en el nos podemos encontrar con el epígrafe *status install* todos los paquetes instalados en el sistema.

En el caso de CentOS el fichero que buscamos también esta en el mismo directorio `/var/log/` pero el fichero es `yum.log` y el epígrafe es *Installed*:

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

Al igual que en **CASI** todas las distribuciones linux tanto CentOS como Ubuntu® utilizan el comando `/` servicio `logrotate` para poder mantener los ficheros de bitácora o logs del sistema operativo, este servicio se encarga de que cuando el fichero de bitácora sea por el parámetro que sea llega a su limite este lo comprime en un fichero `gz` y crea un fichero vacío para que el log del sistema este mas limpio.

2.1. Indique qué comandos ha utilizado para realizarlo así como capturas de pantalla del proceso de reconstrucción del RAID.

3. ¿Qué archivo ha de modificar para programar una tarea?

Tarea: Escriba la línea necesaria para ejecutar una vez al día una copia del directorio `~/codigo` a `~/seguridad/$fecha` donde *\$fecha* es la fecha actual (puede usar el comando `date`).

El fichero que hay que modificar para poder modificar las tareas que se van a ejecutar como tareas programadas se encuentra en `/etc/crontab`.^[3]

Y para poder ejecutar la tarea descrita en el ejercicio bastaría con añadir una linea que contenga el siguiente código:

```
1 [usuario@centos ~]$ sudo nano /etc/crontab
2 SHELL=/bin/bash
3 PATH=/sbin:/bin:/usr/sbin:/usr/bin
4 MAILTO=root
5
6 # For details see man 4 crontabs
7
8 # Example of job definition:
9 # .----- minute (0 - 59)
10 # | .----- hour (0 - 23)
11 # | | .----- day of month (1 - 31)
12 # | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
13 # | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
14 # | | | | |
15 # * * * * * user-name command to be executed
16 0 12 * * * usuario cp ~/codigo ~/seguridad/$(date + %d- %m- %Y)
```

4. Pruebe a ejecutar el comando, conectar un dispositivo USB y vuelva a ejecutar el comando.

Tarea: Copie y pegue la salida del comando. (considere usar `dmesg | tail`). Comente qué observa en la información mostrada.

```

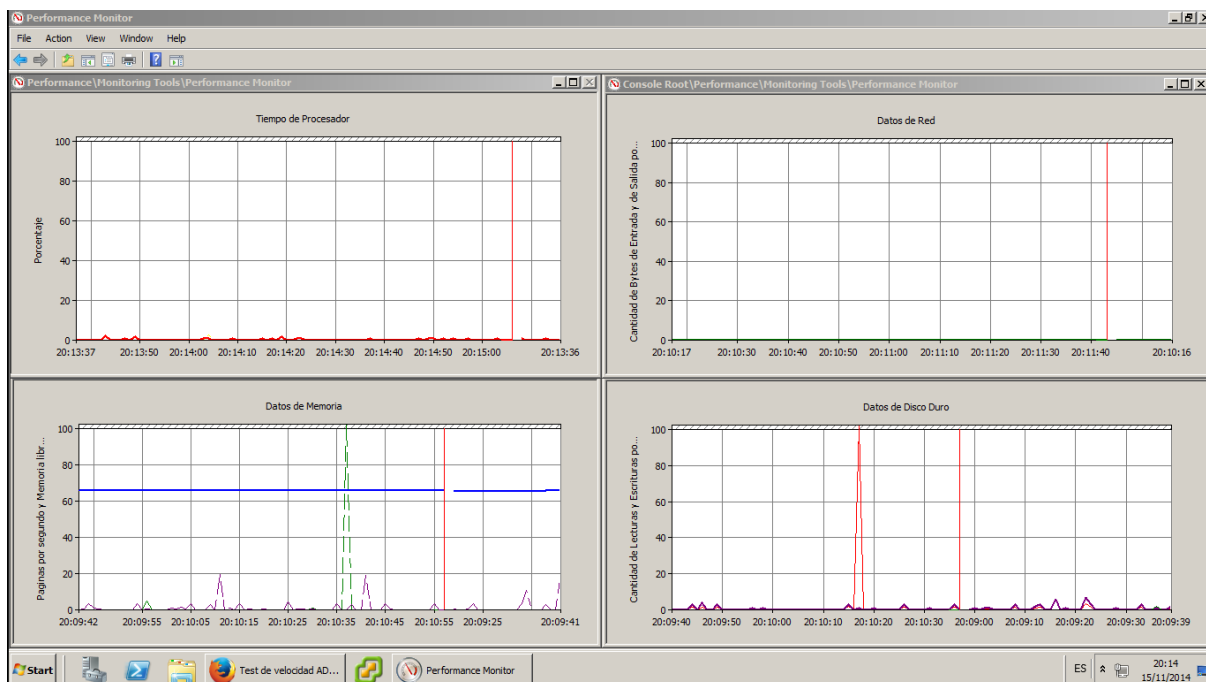
1 [usuario@centos ~]$ dmesg | grep Seagate
2 # pinchamos el USB
3 [usuario@centos ~]$ dmesg | grep Seagate
4 [ 1136.780193] usb 2-1.1: Manufacturer: Seagate
5 [ 1137.802430] scsi 6:0:0:0: Direct-Access      Seagate  USB          0419 PQ: 0 ANSI: 6

```

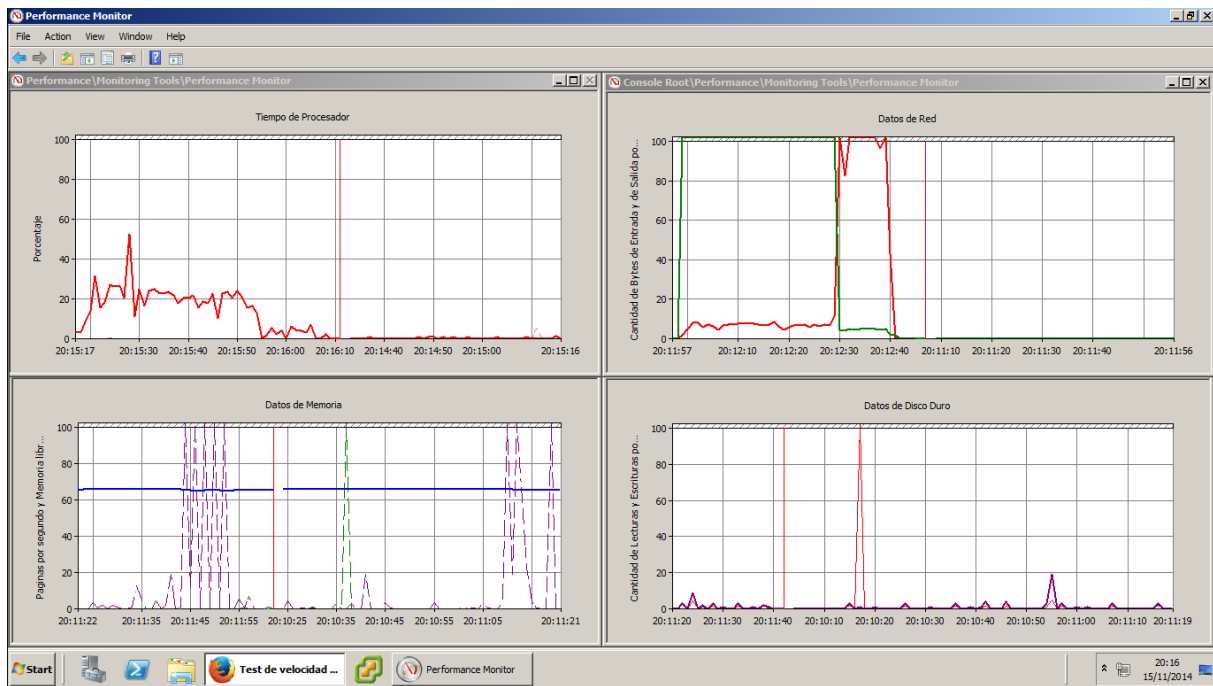
Para que la salida por consola este llena de *basura* he usando el comando grep para limitar la búsqueda de lo que quiero mostrar, así la primera vez que he ejecutado el comando no ha salido nada y posteriormente una vez conectado el disco duro he vuelto a ejecutar el mismo comando y la salida mostraba como se había conectado en el puerto numero 6 del HUB de usb un nuevo dispositivo.

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

En las imágenes que se muestran en la figura 5.1 se pueden ver diferentes gráficas que muestran los datos en vivo de la monitorización de diferentes componentes de un servidor, como pueden ser el procesador, la memoria, disco duro y tarjeta de red, en la figura 5.1a se muestran los datos antes de ejecutar un test de velocidad de ADSL y en la figura 5.1b se muestran los datos durante la prueba, las diferentes gráficas muestran el tiempo del procesador (superior izquierda), la transferencia de datos de la tarjeta de red (superior derecha), la cantidad de memoria que queda libre junto con las faltas de memoria (inferior izquierda), y por ultimo la cantidad de Bytes leídos y escritos en el disco duro por segundo (inferior derecha)



(a) Antes de ejecutar el test de velocidad

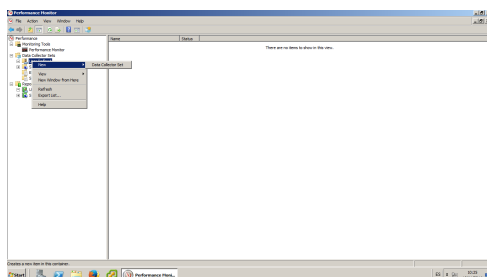


(b) Después de ejecutar el test de velocidad

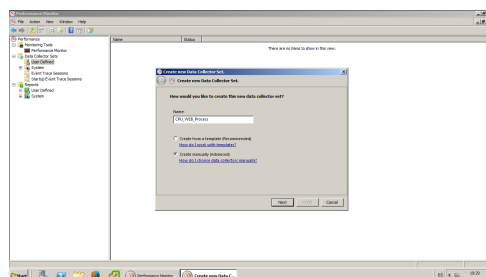
Figura 5.1: Gráficas de datos en el momento que las captura Windows®

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

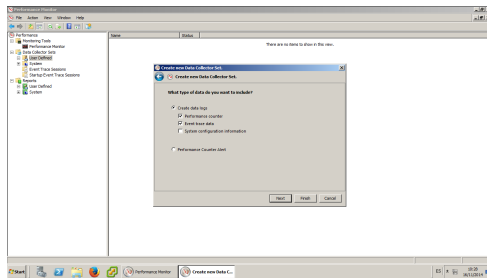
Todos los referentes al procesador, al proceso y al servicio web. Intervalo de muestra 15 segundos Almacene el resultado en el directorio *Escritorio/logs* Incluya las capturas de pantalla de cada paso.



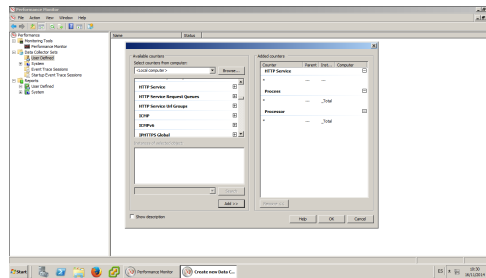
(a) Pulsamos botón derecho, elegimos nuevo colector de datos



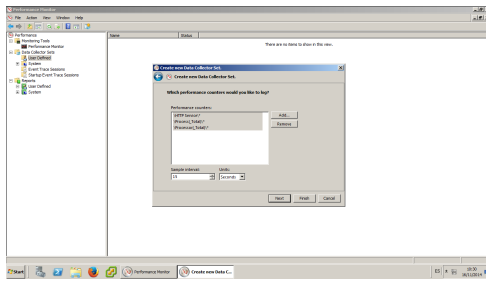
(b) Asignamos un nombre y seleccionamos modo manual



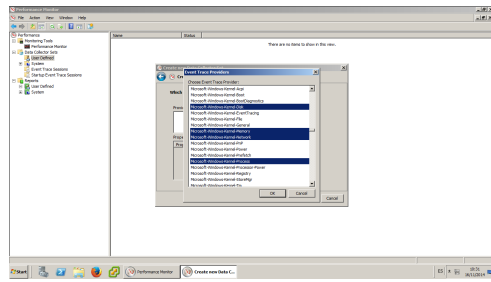
(c) Ejegimos que tipo de recolector queremos crear



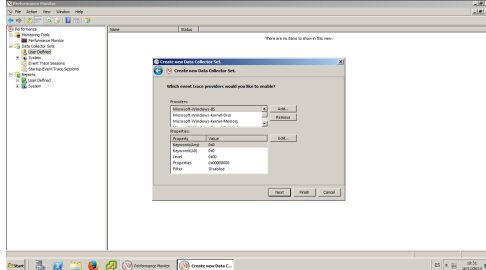
(d) Seleccionamos los datos que queremos monitorizar



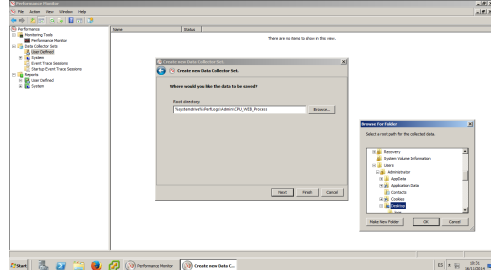
(e) Ponemos el tiempo de recolección a 15 Seg.



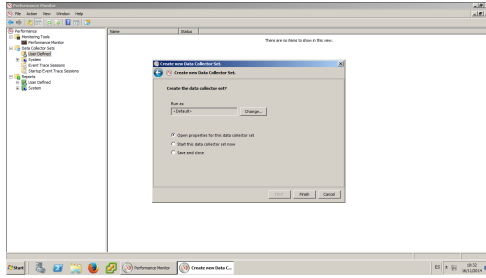
(f) Seleccionamos los datos que queremos Tracear



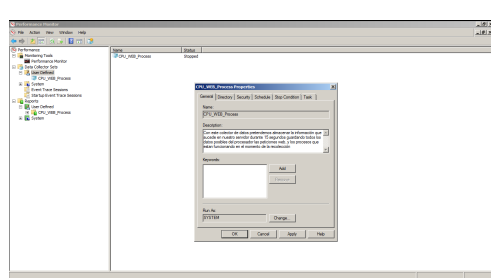
(g) Verificamos que los datos que hemos seleccionados sean los correctos



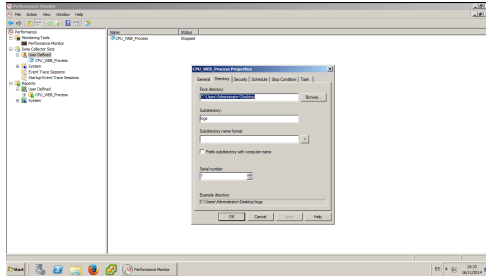
(h) Seleccionamos el sitio donde los queremos guardar *Escritorio*



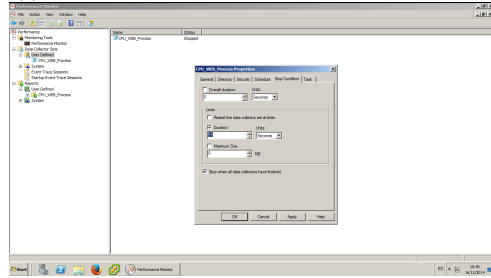
(i) Ahora terminamos pero editamos las propiedades del colector



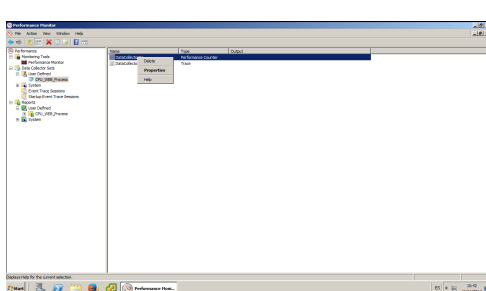
(j) Ponemos una descripción de lo que hace el colector



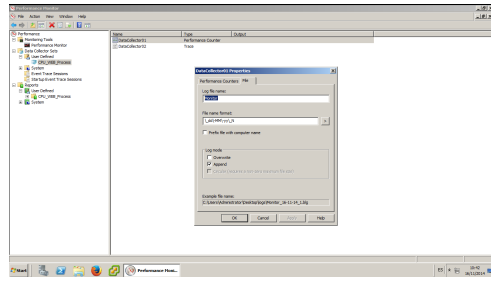
(k) Añadimos el subdirectorio logs donde se guardaran todos los ficheros



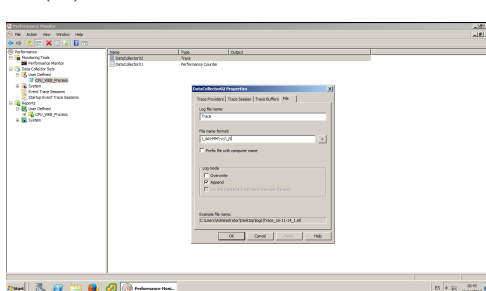
(l) Elegimos en la pestaña condición de parada que el limite son 15 seg.



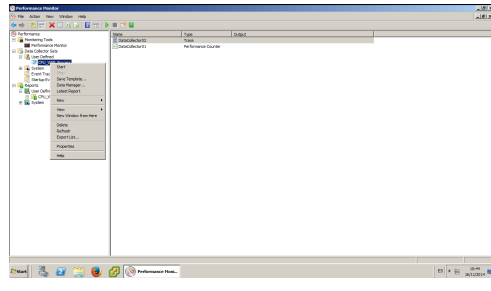
(m) Seleccionamos las propiedades del monitor



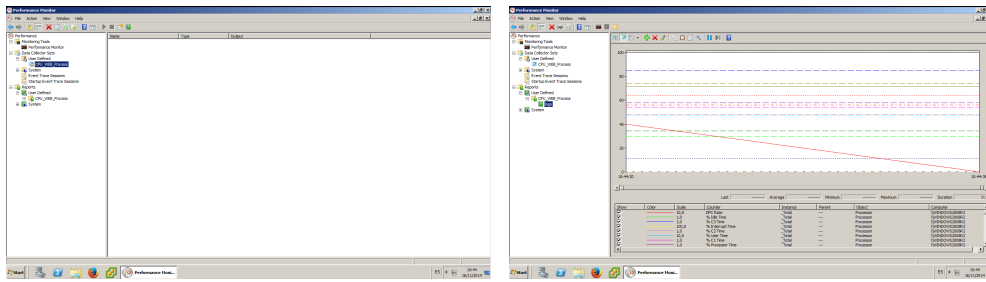
(n) Elegimos que los datos se añaden al final del fichero y ponemos la fecha al nombre



(ñ) Hacemos lo mismo con el traceador de datos



(o) Por ultimo seleccionamos el colector de datos y con el botón derecho lo iniciamos



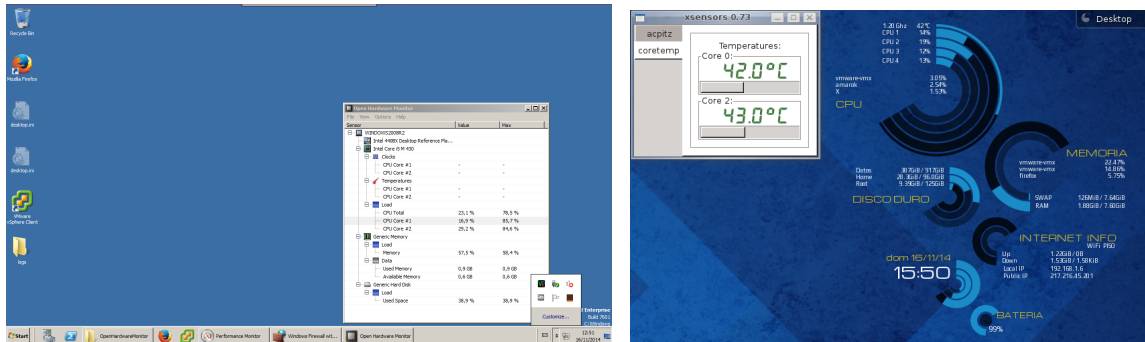
- (p) Cuando el colector esta funcionando se muestra el símbolo de play en el propio colector
- (q) Y por ultimo pulsamos el botón derecho y seleccionamos mostrar informe

Figura 6.1: Pasos a seguir para crear un recolector de datos

7. 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.

Por ser un monitor de hardware multi plataforma he decido usar Open Hardware Monitor en Windows[®] por que en Ubuntu[®] no he encontrado la manera de hacerlo funcionar, por supuesto también he usado en linux xsensors que aunque escueto es muy útil, personalmente prefiero usar conky para linux, este usa cairo junto con las librerías de lua y es mucho mas versátil que open hardware monitor puesto que se puede configurar para que aparte de los sensores de hardware muestre también sensores software o diferentes scripts en un solo monitor.

Otro software que aunque por falta de tiempo no he podido probar bien pero me parece bastante interesante es hardinfo[2]



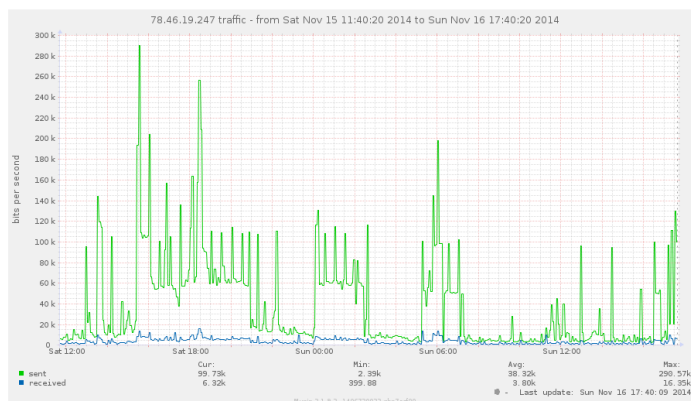
(a) Captura de pantalla Open Hardware Monitor en Windows

(b) Conky en Linux, junto con Xsensors

Figura 7.1: Diferentes monitores de Hardware

8. 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.

Para responder a esta pregunta yo me he decantado por la monitorización de la red del servidor de demostración del proyecto *munin*



Zooming is very easy, it's done in 3 clicks (regular clicks, no drag&drop):

- First click to define the start of zoom.
- Second click to define the ending of zoom.
- Third click inside the defined zone to zoom, outside to cancel the zone.

Plugin Name (domain/hostname/plugin_name) :

Start/Stop of the graph (format:2005-08-15T15:52:01+0000) : /

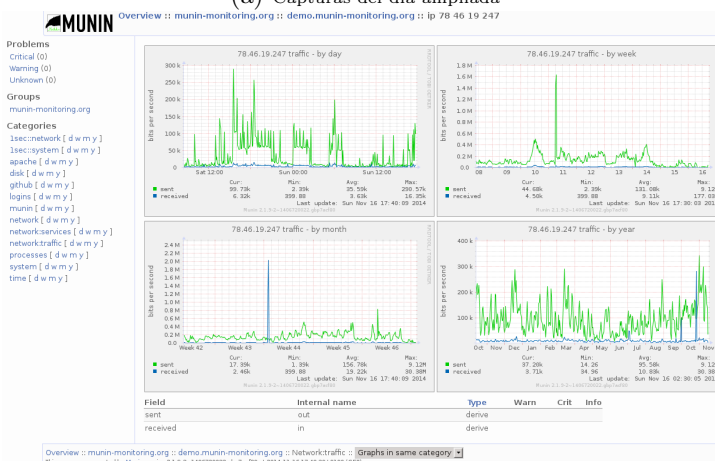
(epoch) : /

Limit low/high : /

Graph size (w/o legend) (pixels): /

Step (granularity) (seconds):

(a) Capturas del día ampliada



(b) Captura de pantalla medidas del día, semana, mes y año

Figura 8.1: Capturas de pantalla de Munin project

En la primera captura figura: 8.1a se puede apreciar claramente que el mayor trafico del día se recibe desde las 15:00 de la tarde hasta las 22:00 de la noche, los picos de descarga que se ven despues de las 24:00 supongo que se deberán a ls visitas de otro país con diferencia horaria, y en la segunda captura figura: 8.1b se puede distinguir cuales son los días de la semana (superior derecha) desde el lunes 10 hasta el viernes 14.

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

Yo he decidido instalar *Nagios* en CentOS, el proceso es muy sencillo puesto que teniendo instalado EPEL basta con instalar todos los paquetes de Nagios, después asignarle una clave al administrador de Nagios y iniciar el servicio, todo esto se hace con los siguientes comandos.

```
1 [usuario@centos ~/]$ sudo yum -y install nagios*
2 ...
3 Instalado:
4 nagios-devel.x86_64 0:3.5.1-1.el7 nagios-plugins.x86_64 0:2.0.1-1.el7
5 nagios-plugins-all.x86_64 0:2.0.1-1.el7 nagios-plugins-bonding.x86_64 0:1.4-3.el7
6 nagios-plugins-apt.x86_64 0:2.0.1-1.el7 nagios-plugins-bacula.x86_64 0:5.2.13-18.el7
7 nagios-plugins-breeze.x86_64 0:2.0.1-1.el7 nagios-plugins-by_ssh.x86_64 0:2.0.1-1.el7
8 nagios-plugins-check-updates.x86_64 0:1.6.7-1.el7 nagios-plugins-dhcp.x86_64 0:2.0.1-1.el7
9 nagios-plugins-cluster.x86_64 0:2.0.1-1.el7 nagios-plugins-dbi.x86_64 0:2.0.1-1.el7
10 nagios-plugins-dig.x86_64 0:2.0.1-1.el7 nagios-plugins-disk.x86_64 0:2.0.1-1.el7
11 nagios-plugins-disk_smb.x86_64 0:2.0.1-1.el7 nagios-plugins-file_age.x86_64 0:2.0.1-1.el7
12 nagios-plugins-dns.x86_64 0:2.0.1-1.el7 nagios-plugins-dummy.x86_64 0:2.0.1-1.el7
13 nagios-plugins-flexlm.x86_64 0:2.0.1-1.el7 nagios-plugins-fping.x86_64 0:2.0.1-1.el7
14 nagios-plugins-game.x86_64 0:2.0.1-1.el7 nagios-plugins-icmp.x86_64 0:2.0.1-1.el7
15 nagios-plugins-hpjd.x86_64 0:2.0.1-1.el7 nagios-plugins-http.x86_64 0:2.0.1-1.el7
16 nagios-plugins-ide_smart.x86_64 0:2.0.1-1.el7 nagios-plugins-ifoperstatus.x86_64 0:2.0.1-1.el7
17 nagios-plugins-iftstatus.x86_64 0:2.0.1-1.el7 nagios-plugins-load.x86_64 0:2.0.1-1.el7
18 nagios-plugins-ircd.x86_64 0:2.0.1-1.el7 nagios-plugins-ldap.x86_64 0:2.0.1-1.el7
19 nagios-plugins-log.x86_64 0:2.0.1-1.el7 nagios-plugins-mailq.x86_64 0:2.0.1-1.el7
20 nagios-plugins-mrtg.x86_64 0:2.0.1-1.el7 nagios-plugins-nagios.x86_64 0:2.0.1-1.el7
21 nagios-plugins-mrtgtraf.x86_64 0:2.0.1-1.el7 nagios-plugins-mysql.x86_64 0:2.0.1-1.el7
22 nagios-plugins-nrpe.x86_64 0:2.15-2.el7 nagios-plugins-nt.x86_64 0:2.0.1-1.el7
23 nagios-plugins-ntp.x86_64 0:2.0.1-1.el7 nagios-plugins-openmange.x86_64 0:3.7.12-1.el7
24 nagios-plugins-ntp-perl.x86_64 0:2.0.1-1.el7 nagios-plugins-nwstat.x86_64 0:2.0.1-1.el7
25 nagios-plugins-oracle.x86_64 0:2.0.1-1.el7 nagios-plugins-overcr.x86_64 0:2.0.1-1.el7
26 nagios-plugins-perl.x86_64 0:2.0.1-1.el7 nagios-plugins-procs.x86_64 0:2.0.1-1.el7
27 nagios-plugins-pgsql.x86_64 0:2.0.1-1.el7 nagios-plugins-ping.x86_64 0:2.0.1-1.el7
28 nagios-plugins-radius.x86_64 0:2.0.1-1.el7 nagios-plugins-real.x86_64 0:2.0.1-1.el7
29 nagios-plugins-rpc.x86_64 0:2.0.1-1.el7 nagios-plugins-snmp.x86_64 0:2.0.1-1.el7
30 nagios-plugins-sensors.x86_64 0:2.0.1-1.el7 nagios-plugins-smtp.x86_64 0:2.0.1-1.el7
31 nagios-plugins-ssh.x86_64 0:2.0.1-1.el7 nagios-plugins-swap.x86_64 0:2.0.1-1.el7
32 nagios-plugins-tcp.x86_64 0:2.0.1-1.el7 nagios-plugins-uptime.x86_64 0:2.0.1-1.el7
33 nagios-plugins-time.x86_64 0:2.0.1-1.el7 nagios-plugins-ups.x86_64 0:2.0.1-1.el7
34 nagios-plugins-users.x86_64 0:2.0.1-1.el7 nagios-plugins-wave.x86_64 0:2.0.1-1.el7
35
36 Dependencia(s) instalada(s):
37 autogen-libopts.x86_64 0:5.18-5.el7 bacula-libs.x86_64 0:5.2.13-18.el7
38 bind-libs.x86_64 32:9.9.4-14.el7 bind-utils.x86_64 32:9.9.4-14.el7
39 fping.x86_64 0:3.5-3.el7 libdbi.x86_64 0:0.8.4-6.el7
40 libsmclient.x86_64 0:4.1.1-37.el7_0 libtirpc.x86_64 0:0.2.4-0.3.el7
41 net-snmp-libs.x86_64 1:5.7.2-18.el7 net-snmp-utils.x86_64 1:5.7.2-18.el7
42 ntp.x86_64 0:4.2.6p5-18.el7.centos ntpdate.x86_64 0:4.2.6p5-18.el7.centos
43 perl-Class-Accessor.noarch 0:0.34-12.el7 perl-Config-Tiny.noarch 0:2.14-7.el7
44 perl-Crypt-DES.x86_64 0:2.05-20.el7 perl-Digest.noarch 0:1.17-245.el7
45 perl-Digest-HMAC.noarch 0:1.03-5.el7 perl-Digest-MD5.x86_64 0:2.52-3.el7
46 perl-Digest-SHA.x86_64 1:5.85-3.el7 perl-Digest-SHA1.x86_64 0:2.13-9.el7
47 perl-Math-Calc-Units.noarch 0:1.07-9.el7 perl-Module-Implementation.noarch 0:0.06-6.el7
48 perl-Module-Runtime.noarch 0:0.013-4.el7 perl-Nagios-Plugin.noarch 0:0.36-7.el7
49 perl-Net-SNMP.noarch 0:6.0.1-7.el7 perl-Params-Validate.x86_64 0:1.08-4.el7
50 perl-Readonly.noarch 0:1.03-22.el7 perl-Readonly-XS.x86_64 0:1.05-15.el7
51 perl-Socket6.x86_64 0:0.23-15.el7 perl-Sort-Versions.noarch 0:1.5-22.el7
52 perl-Try-Tiny.noarch 0:0.12-2.el7 postgresql-libs.x86_64 0:9.2.7-1.el7
53 qstat.x86_64 0:2.11-13.20080912svn311.el7 radiusclient-ng.x86_64 0:0.5.6-9.el7
54 rpcbind.x86_64 0:0.2.0-23.el7 samba-client.x86_64 0:4.1.1-37.el7_0
55
56 Listo
57 [usuario@centos ~/]$ sudo htpasswd /etc/nagios/passwd nagiosadmin
58 New password:
59 Re-type new password:
60 Updating password for user nagiosadmin
61 [usuario@centos ~/]$ sudo systemctl start nagios
```

Por ultimo lo que resta es acceder a la consola de administración Web <http://192.168.50.130/nagios>, todavía quedaría configurar los diferentes host que queremos monitorizar pero eso ya es *harina de otro costal*

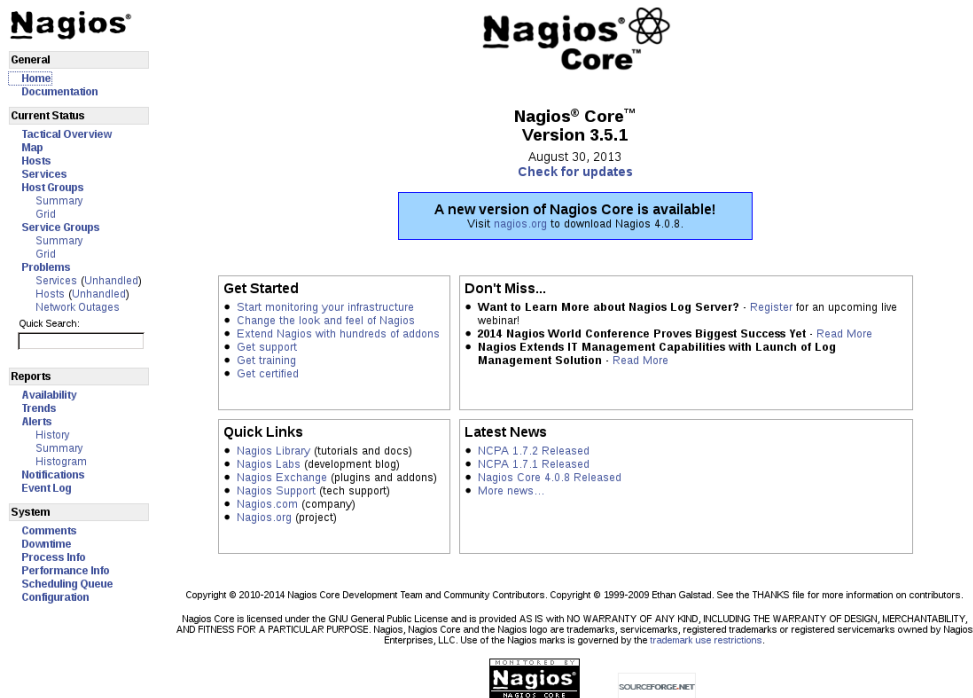
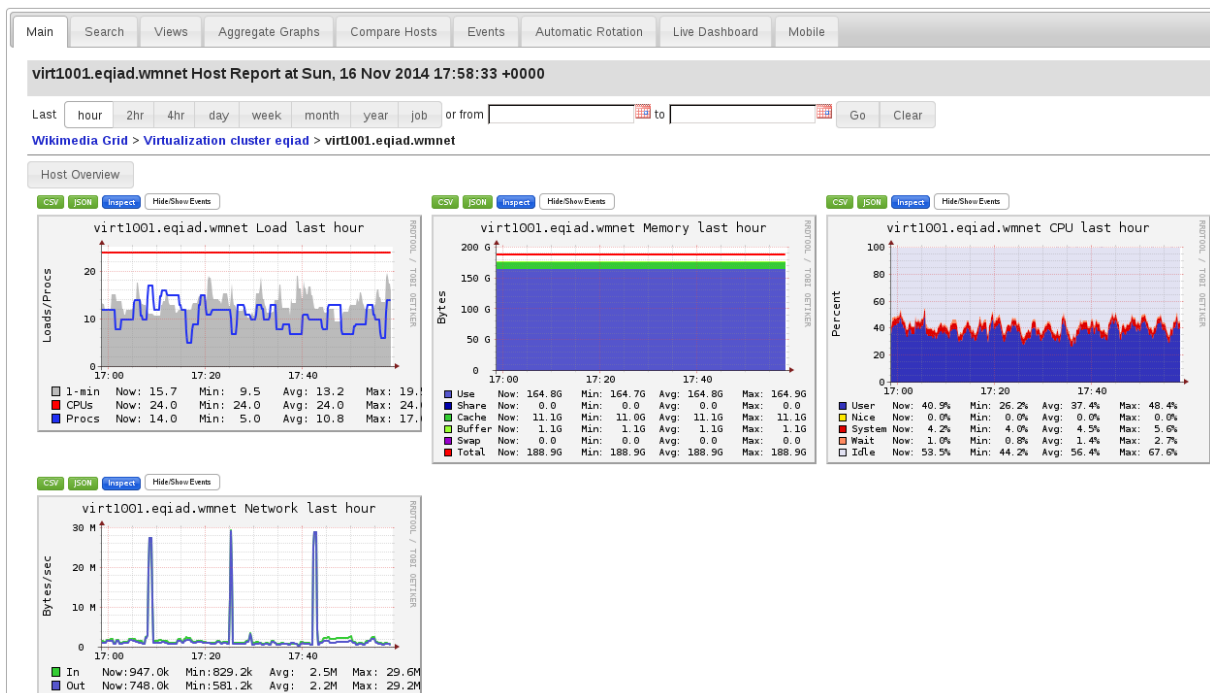


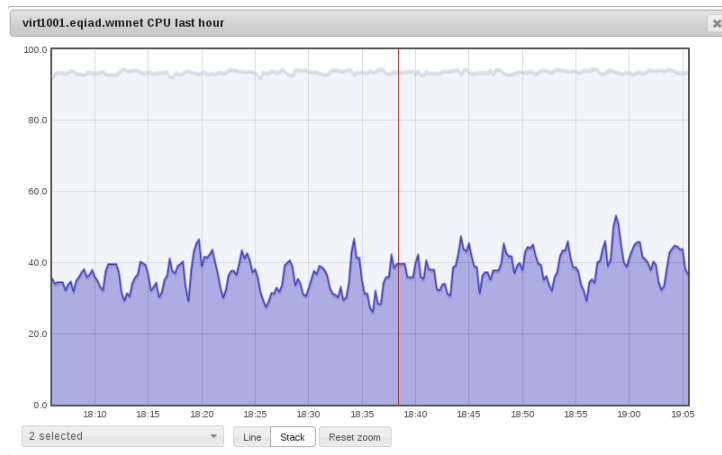
Figura 8.2: Nagios Core

8.2. Haga lo mismo que con Munin.

Realmente hay tanta información relevante para mostrar que no se que gráfica es mejor enseñar, me he decantado por las gráficas de servidores virtuales por que creo que es desde donde podemos sacar información mas real a la hora de conocer la carga de trabajo que tiene <http://www.wikimedia.org> por que son sobre estos servidores donde trabaja directamente las diferentes aplicaciones de Wikimedia



(a) Ganglia, Servidor virtual Wikimedia.org, LOAD, CPU, MEM, NET



(b) Datos de CPU Idle y CPU User del mismo servidor

Figura 8.3: Capturas de pantalla de Ganglia project

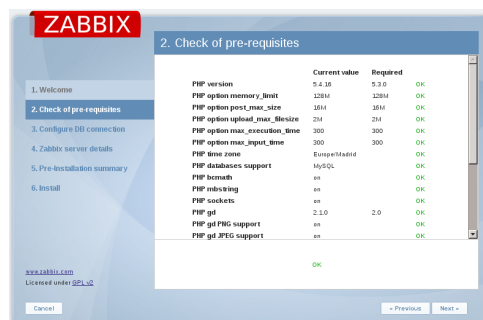
Después de ver este pedazo de proyecto open source estoy gratamente sorprendido por que la cantidad de detalles que se pueden observar en sus diferentes gráficas es sorprendente, bien estructuradas y con micro información de las gráficas para que sea fácilmente reconocible el problema, si lo hubiera, en cualquier servidor y/o servicio.

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

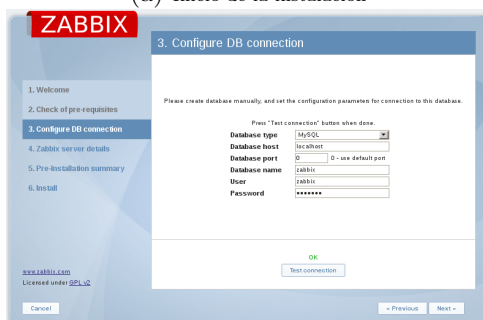
He instalado Zabbix en CentOS tal y como explican en su página de documentación[4], la verdad es que la instalación es bastante fácil quizás la parte más difícil es instalar todos los paquetes necesarios para los requisitos previos, los pasos mostrados en las figuras: 8.4a a la 8.4f son la instalación propiamente dicha, las figuras: 8.4g y 8.4h forman parte de la interfaz web de la propia administración de Zabbix.



(a) Inicio de la instalación



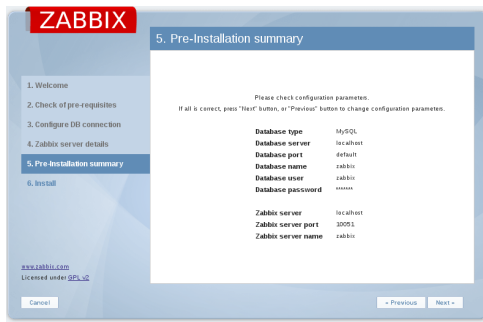
(b) Comprobando todos los pre-requisitos



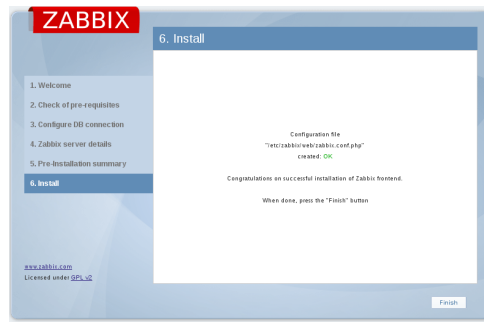
(c) Configurando la conexión de la BD



(d) Configurando los parámetros del servidor



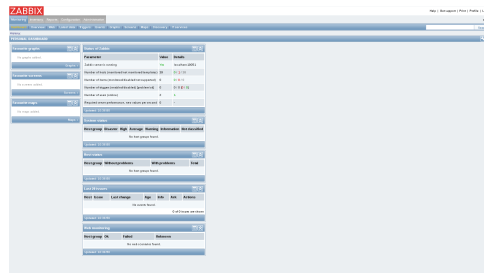
(e) Sumario de la configuración antes de la instalación



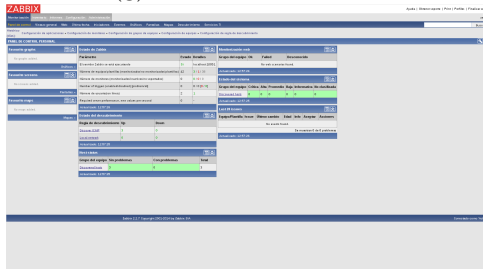
(f) Confirmación de que la instalación ha sido un éxito



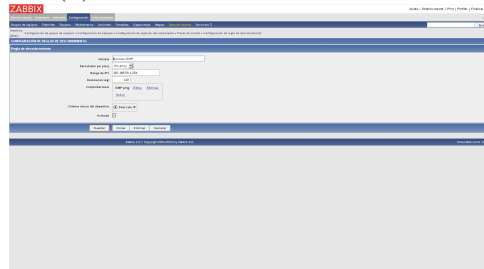
(g) Pantalla de autenticación



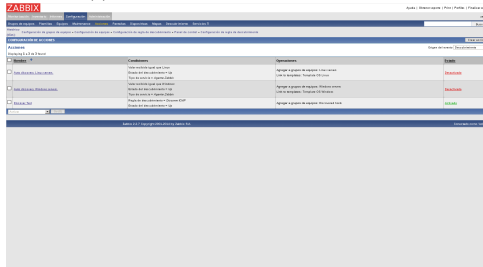
(h) Primera vista del programa en ejecución



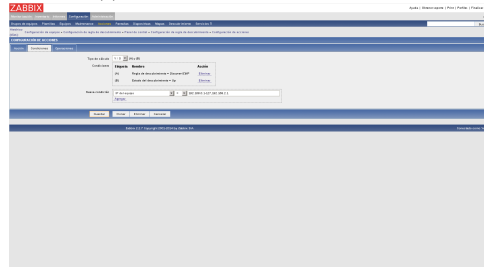
(i) Modificación del Panel de Control



(j) Creación de un Descubrimiento



(k) Lista que muestra las Acciones creadas



(l) Ejemplo de creación de un Acción

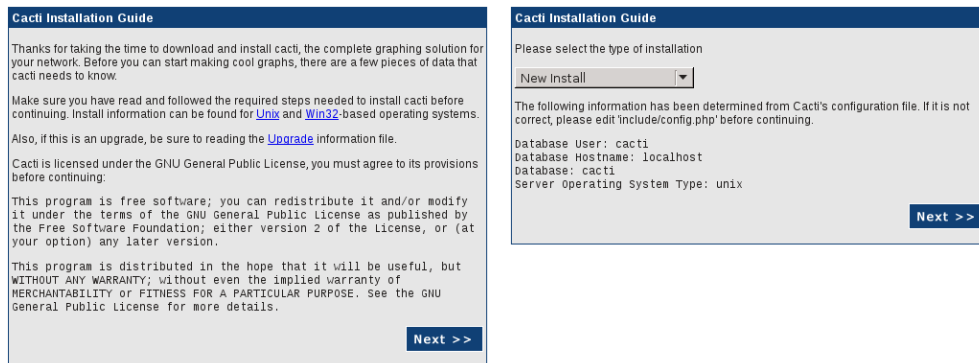
Figura 8.4: Interfaz Web de Instalación y ejecución de Zabbix

Una vez que estamos dentro de la administración de Zabbix lo que vamos a realizar es poner bajo monitorización algún host que se encuentre en nuestra red, así pues esta tarea es como se muestra en las imágenes: 8.4i a 8.4l.

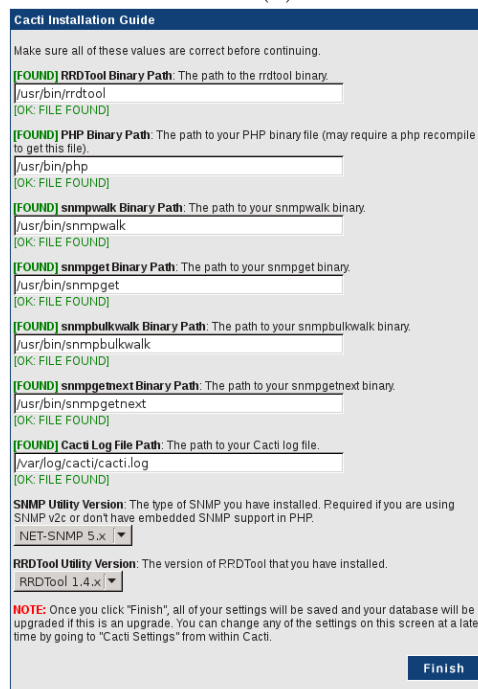
En la imagen: 8.4i se muestra la modificación del Panel de Control para que se pueda ver toda la información de un solo vistazo, para poder descubrir los diferentes dispositivos que se encuentran en nuestra red es necesario realizar algunas configuraciones, eso es lo que se muestra en la imagen: 8.4j la configuración de lo que en Zabbix se denomina descubrimiento, que no es mas que unas reglas predeterminadas para según que servicio o que situación queramos descubrir, para la practica simplemente he usado un descubrimiento por ICMP Ping, con esta configuración simplemente no podremos saber cuantos dispositivos tenemos en nuestra red, aun falta configurar las acciones que tenemos que realizar cuando usemos este tipo de *Descubrimiento*, el como configurar estas acciones es lo que se muestra en las imágenes 8.4k y 8.4l

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

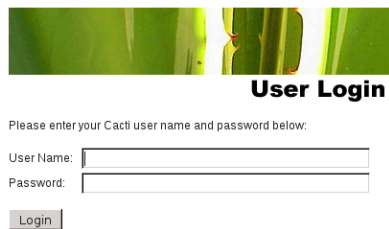
Para Cacti he usado Ubuntu[®] es un monitor del sistema bastante sencillo de usar, la interfaz puede llegar a ser igual de tosca que la de Nagios pero parece que esta en un nivel medio de monitorización con respecto a los demás monitores, por ahora el que se lleva la palma es Ganglia. Cacti no es difícil de instalar y en Ubuntu[®] menos puesto que al instalarlo te pregunta si quieres configurar la base de datos y te auto configura los parámetros de PHP



(a) Sumario de la configuración antes de la instalación (b) Confirmación de que la instalación ha sido un éxito



(c) Pantalla de autenticación

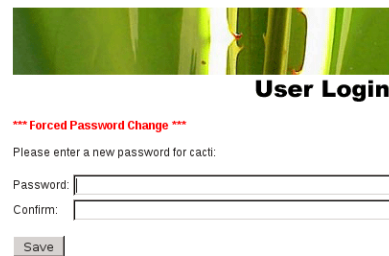


User Login

Please enter your Cacti user name and password below:

User Name:

Password:



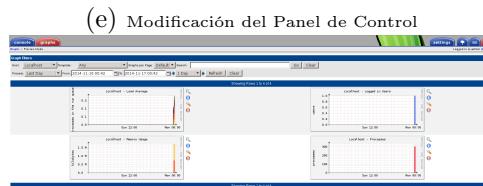
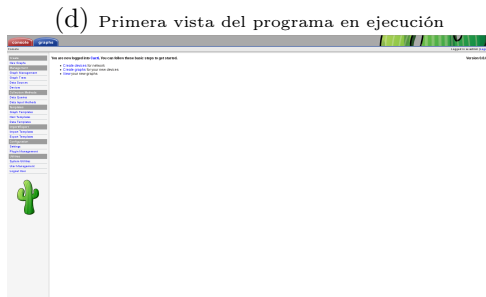
User Login

*** Forced Password Change ***

Please enter a new password for cacti:

Password:

Confirm:



(f) Creación de un Descubrimiento

(g) Lista que muestra las Acciones creadas

Figura 8.5: Interfaz Web de Instalación y ejecución de Zabbix

8.5. Instale el monitor, muestre y comente algunas capturas de pantalla.

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

Básicamente lo que se explica en el artículo es una introducción de como se tiene que empezar a trabajar con strace, te explica un caso teórico y luego un caso práctico, el cual me hubiera venido muy bien en las prácticas anteriores puesto que al igual que explica en el ejemplo práctico a mí me paso algo parecido al configurar Apache en CentOS, y es que en mi caso había un directorio, donde Apache tenía que guardar parte de su configuración pero el usuario de Apache no tenía permisos de escritura por lo tanto daba un error, pero como este error se producía antes de que Apache levantara el servicio no se registraba nada en el fichero de bitácoras (LOG) de Apache por lo tanto no se mostraba cual era el error al igual que pasa en el ejemplo del artículo, sin embargo en el artículo se resuelve usando strace en unos 3 minutos mientras que yo le tuve que dedicar como unas 3 horas ha repasar los permisos de todas las carpetas en las cuales Apache tenía posibilidad de leer y escribir hasta dar con el problema, así que, una vez aprendido lo usare bastante mas a menudo.

9.1. Desarrolle una página en C o C++ y analice su comportamiento usando valgrind. Visite² para ver un ejemplo sencillo de una página web generada por un programa escrito en C.

He estado buscando diferentes formas de hacer esta pregunta, pero programando una página web en C, los tiempos que me salían eran ínfimos, así que recycle un programa del año anterior para poder realizar esta parte de la práctica, los resultados y el programa en cuestión están puestos a continuación:

```

1 <?xml version="1.0"?>
2 <valgrindoutput>
3   <protocolversion>4</protocolversion>
4   <protocoltool>memcheck</protocoltool>

```

²<http://www.cs.tut.fi/~jkorpele/forms/cgic.html>


```

5 <preamble>
6 <line>Memcheck, a memory error detector</line>
7 <line>Copyright (C) 2002-2013, and GNU GPL, by Julian Seward et al.</line>
8 <line>Using Valgrind-3.9.0 and LibVEX; rerun with -h for copyright info</line>
9 <line>Command: ./bin/monedas 6 1,2,5,10,20,25 3491</line>
10 </preamble>
11 <pid>4341</pid>
12 <ppid>2868</ppid>
13 <tool>memcheck</tool>
14 <args>
15 <vargv>
16 <exe>/usr/bin/valgrind</exe>
17 <arg>--leak-check=full</arg>
18 <arg>-v</arg>
19 <arg>--read-var-info=yes</arg>
20 <arg>--xml=yes</arg>
21 <arg>--xml-file=result.xml</arg>
22 </vargv>
23 <argv>
24 <exe>./bin/monedas</exe>
25 <arg>6</arg>
26 <arg>1,2,5,10,20,25</arg>
27 <arg>3491</arg>
28 </argv>
29 </args>
30 <status>
31 <state>RUNNING</state>
32 <time>00:00:00.034 </time>
33 </status>
34 <status>
35 <state>FINISHED</state>
36 <time>00:00:00.053 </time>
37 </status>
38 <errorcounts>
39 </errorcounts>
40 <suppcounts>
41 <pair>
42 <count>3</count>
43 <name>glibc-2.5.x-on-SUSE-10.2-(PPC)-2a</name>
44 </pair>
45 </suppcounts>
46 </valgrindoutput>

```

```

1 /*
2  * monedas.c
3  *
4  *           Fecha: 03/06/14
5  *         Desarrollado: Carlos de la Torre
6  *
7  */
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <string.h>
11
12 /*
13  * Con esto podemos saber el tamaño de una array en C
14  */
15 #define NELEMENTOS(x) (sizeof(x)/sizeof(x[0]))
16
17 /*
18  * Variable global para saber el nombre del programa
19  */
20 char* nomPrograma;
21
22 /*
23  * Este es un truco para poder pasar matrices entre
24  * funciones en C sin tener que usar punteros
25  */
26 typedef unsigned int matriz[15][9999]; // <-- Definir los tamaños máximos para el monedero y la vuelta
27
28 /*
29  * Con esto sacamos por pantalla el valor
30  * de todas las monedas que tiene el monedero
31  */
32 void imprimeMonedero(int MONEDERO[], int CANTI_MONEDAS){
33     int i = 0;
34     printf ("Monedero: ");
35     for (;i<CANTI_MONEDAS;++i){
36         printf ("%d",MONEDERO[i]);
37         if (i < CANTI_MONEDAS-1)
38             printf (" ");
39     }
40     printf("]");
41 }
42 printf("\n");
43 }
44
45 /*
46  * Con esto sacamos por pantalla el valor
47  * de la tabla que contiene la cantidad de
48  * monedas a devolver según cuanto tengamos
49  * que devolver
50  * Pasamos el monedero por parámetros para
51  * poder hacer los incrementos de las columnas
52  * por el valor de la moneda mas pequeña
53  */
54 void imprimeTabla(int MONEDERO[], matriz TABLA, int FIL, int COL){
55     int fil = 0, col = 0;

```



```

56 printf("\nA Devolver --> ");
57 for (col = 0; col <= COL; col += MONEDERO[0]){ // <-- el incremento es igual a la moneda mas pequeña
58 if (MONEDERO[0] > (COL - col)) // <-- El valor de la moneda es mas grande que lo que me queda por recorrer
59 printf("%2d ", COL);
60 else
61 printf("%2d ", col);
62 }
63 printf("\n-----");
64 for (col = 0; col < COL; col += MONEDERO[0]){
65 printf("----");
66 }
67 printf("\n");
68 for (fil = 0; fil < FIL; fil++){
69 printf("Mon. tipo %d -> ", fil+1);
70 for (col = 0; col <= COL; col += MONEDERO[0]){
71 if (MONEDERO[0] > (COL - col)) // <-- El valor de la moneda es mas grande que lo que me queda por recorrer
72 printf("%2d ", TABLA[fil][COL]);
73 else
74 printf("%2d ", TABLA[fil][col]);
75 }
76 printf("\n");
77 }
78 }
79
80 /*
81 * Esta función controla los posibles errores del programa
82 */
83 void funError(int ERROR){
84 switch (ERROR){
85 case 1:
86 printf(" La cantidad de parametros es incorrecta \n");
87 break;
88 case 2:
89 printf(" La cantidad valores de las monedas es diferente al numero de monedas \n");
90 break;
91 case 3:
92 printf(" No hay una solución optima posible \n");
93 break;
94 }
95
96 printf(" Uso: %s [Cantidad Monedas] [Valor Monedas separadas por ,] [Euros a devolver]\n\n", nomPrograma);
97 printf(" Ejemplo: %s 3 1,2,5 13\n", nomPrograma);
98 exit(-ERROR);
99 }
100
101 /*
102 * Función de mínimo en C
103 * la he creado yo por que no he encontrado
104 * ninguna librería de C que la tuviera
105 */
106 int min(int VALOR_A, int VALOR_B){
107 return (VALOR_A < VALOR_B) ? VALOR_A : VALOR_B;
108 }
109
110 /*
111 * Con esta función comprobamos cuantas monedas tenemos
112 * que usar para devolver el dinero que nos queda
113 */
114 int Cambio(int MONEDERO[], int CANTI_MONEDAS, int DEVOLVER){
115 // Este es el indice que uso para los for de recorrido
116 int idx = 0;
117 /* Este vector guarda la cantidad de monedas que hemos usado
118 según sea del primer tipo de monedas o del segundo tipo
119 así sucesivamente la posición es el tipo de monedas
120 y el valor es la cantidad de veces que la hemos usado
121 según la vuelta que necesitamos dar en cada iteración */
122 int monedas_usadas[CANTI_MONEDAS];
123 /* Esta variable la uso para no tener que modificar
124 la cantidad de monedas que me llegan a la función */
125 int auxCantiMonedas = CANTI_MONEDAS;
126 /* Con este for inicializo todos el vector de monedas
127 usadas a 0 de esa forma cada vez que llamo a cambio
128 me aseguro de la veracidad de la cantidad de monedas
129 se supone que el for debería llegar hasta el tamaño
130 total de monedas_usadas menos 1 que es el último
131 elemento de monedas_usadas */
132 for (idx = 0; idx < CANTI_MONEDAS; idx++){
133 monedas_usadas[idx] = 0; // <-- inicializo el vector a 0
134 }
135 /* Me aseguro que la variable que voy a devolver esté a 0 */
136 int cantidad_monedas = 0;
137
138 /* El condicional que viene a continuación contempla los
139 dos casos base de la función de recurrencia y también
140 el caso general de la misma */
141 if (DEVOLVER == 0)
142 cantidad_monedas = 0;
143 else if (DEVOLVER < 0 || auxCantiMonedas <= 0)
144 cantidad_monedas = 999999; // <-- mas infinito
145 else{
146 while (DEVOLVER > 0 && auxCantiMonedas > 0){ // <-- Mientras tenga que devolver y no este en la ultima moneda
147 if (DEVOLVER >= MONEDERO[auxCantiMonedas-1]){ // <-- Si el valor de la moneda es mas pequeño de lo que tengo que
148 devolver
149 DEVOLVER -= MONEDERO[auxCantiMonedas-1]; // <-- resto el valor de la moneda a lo que me queda por devolver
150 monedas_usadas[auxCantiMonedas-1]++; // <-- Y añado una moneda de ese tipo a la solución
151 } else // <-- Si el valor de la moneda es mas grande que lo que me queda por devolver
152 auxCantiMonedas--; // <-- cojo la siguiente moneda mas pequeña
153 }
154 /* Este for se encarga de recorrer el vector de monedas
155 usadas para contarlas y acumularlas en cantidad_monedas
156 que es la variable que se devuelve */

```

```

156 for (idx=0;idx<=CANTI_MONEDAS-1;idx++){
157     cantidad_monedas += monedas_usadas[idx];
158 }
159 }
160 return cantidad_monedas;
161 }
162
163 /*
164 * Este es el menú principal
165 */
166 int main(int argc, const char *argv[]){
167     if (argc == 1)
168         funError(1);
169     int misMonedas[atoi(argv[1])]; // <-- Cantidad de Monedas disponibles
170     nomPrograma = argv[0];
171     char* parametro2 = argv[2];
172     //int misMonedas[] = {1,2,5,9};
173     char *charMoneda = malloc(2);
174     int CD = 0, M = 0, V = 0;
175     int idx = 0; // <-- Indice de cualquier for
176
177     int DEBUGMODE = 0;
178     if ((argc == 5) && (argv[4] != NULL))
179         DEBUGMODE = 1;
180     else
181         DEBUGMODE = 0;
182
183     if (argc == 4 || argc == 5){
184         charMoneda = strtok(parametro2, ",");
185         while (charMoneda){
186             misMonedas[idx] = atoi(charMoneda);
187             idx++;
188             charMoneda = strtok(NULL, ",");
189         }
190         if (idx!=NELEMENTOS(misMonedas))
191             funError(2);
192         CD = atoi(argv[3]);
193     }else if (argc < 4)
194         funError(1);
195
196     matriz tablaDinamica;
197     // OJO: aquí hay un "fallo" y es que solo se pueden tener
198     // monederos de 15 monedas y solo se puede devolver hasta 9999
199     // Se define en la parte superior del programa.
200
201     /*
202     * Este es el núcleo del algoritmo:
203     * -El 1er for recorre la cantidad que tenemos que devolver desde 0 ya que también se contempla la posibilidad de no
204     *   devolver nada
205     * -El 2do for recorre la cantidad de monedas, nótese que este no recorre hasta <= si no que se detiene en el ultimo
206     *   elemento del array
207     * -El condicional se encarga de llenar las posiciones 0 en los elementos que no tenemos que devolver dinero en otro
208     *   caso ver memoria
209     * para poder entender por que es necesario un par de diagramas para entenderlo
210     * Comentarios para entender el código:
211     * -Variable misMonedas[]: Este sería el monedero del ejercicio
212     * -Variable tablaDinamica: Tabla dinámica que almacena la cantidad de monedas a devolver en cada caso
213     * -Variable CD: esta es la cantidad que le tenemos que devolver al cliente.
214     * -Variable CM: esta es la cantidad de monedas optima que se tienen que devolver al final.
215     * -Variable M: estas son la cantidad de monedas disponibles que tendrá la función Cambio() en cada momento.
216     * -Variable V: este sera el valor que tendrá que devolver Cambio() en cada iteración
217     * -Variable C1 y C2: Son variables auxiliares para hacer el código mas legible
218     */
219     unsigned int C1, C2;
220     for (V = 0; V<=CD; V+=misMonedas[0]){ // aquí es donde tengo que poner un if para ver si me paso de lo que voy a devolver
221         for (M = 0; M<NELEMENTOS(misMonedas); M++){
222             if (V==0){
223                 tablaDinamica[M][V] = 0; // <-- Cuando el valor a devolver es 0 pongo en la tabla 0 monedas a devolver
224             }else{
225                 C1 = Cambio(misMonedas, M+1, V); // <-- Comprobamos el cambio con una moneda mas de otro tipo
226                 C2 = 1+Cambio(misMonedas, M, V-misMonedas[M]); // <-- Comprobamos el cambio con las mismas monedas pero con menos a
227                 devolver
228                 tablaDinamica[M][V] = min(C1, C2); // <-- Comprobamos con cual de las dos situaciones damos menos monedas
229             }
230         }
231     }
232
233     /*
234     * Aquí presentamos los datos que tenemos hasta el momento
235     */
236     printf ("Cantidad a devolver: %d \n", CD);
237
238     /*
239     * Como en C no hay un length o size, hay que estar
240     * continuamente pasando el tamaño del vector
241     * como los de la matriz
242     */
243     imprimeMonedero(misMonedas, NELEMENTOS(misMonedas));
244     if (DEBUGMODE)
245         imprimeTabla(misMonedas, tablaDinamica, NELEMENTOS(misMonedas), CD);
246
247     /*
248     * Las variables aux1, aux2 y tamSol son para que el código sea mas fácil de leer
249     * esta parte es donde se recompone la solución a partir de los datos que hemos
250     * rellenado en la tabla
251     * Tanto esta parte como la anterior se podrían meter en una función pero al tener
252     * que pasar tantos parámetros he decidido dejarlo en el main
253     */
254     int aux1 = NELEMENTOS(misMonedas)-1; // <-- Le pongo -1 por que es el ultimo elemento del array
255     int aux2 = CD; // <-- Esta es la cantidad de dinero a devolver
256     int solucion[NELEMENTOS(misMonedas)]; // <-- Array Solución

```

```

253 for (idx = 0; idx<NELEMENTOS(solucion);idx++) // <-- Inicializo solución a 0
254 solucion[idx] = 0;
255
256 while (aux1 >= 0 && aux2 != 0){
257 if (aux1 > 0 && tablaDinamica[aux1][aux2]>=tablaDinamica[aux1-1][aux2]) // <-- El orden de las comprobaciones
    IMPORTANTE
258 aux1 -= 1;
259 else{
260 solucion[aux1] += 1;
261 aux2 = aux2 - misMonedas[aux1];
262 }
263 }
264
265 printf ("\nSolucion: \n");
266 for (aux1 = 0 ;aux1<NELEMENTOS(solucion);aux1++){ // <-- Como tengo que llegar al final de los elementos uso <=
267 printf("%2d Monedas de %d \n",solucion[aux1],misMonedas[aux1]);
268 }
269
270 return 0;
271 }

```

9.2. Desarrolle un script en PHP y analice su ejecución con alguno (o los dos) profilers.

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

```

1  #!/usr/bin/python
2  import sys,os
3  import time
4
5  # esto lo usamos para saber si somos superusuario
6  usuario = os.getenv("USER")
7  sudo_usuario = os.getenv("SUDO_USER")
8
9  #esto es para saber el nombre del script
10 script = sys.argv[0]
11
12 # Esta sera el fichero que leamos
13 open_file = "/etc/ssh/sshd_config"
14
15 # Usamos estas dos variables para buscar en un fichero y intercambiarlas
16 var_original = "PasswordAuthentication no\n"
17 var_reemplazada = "PasswordAuthentication yes\n"
18
19 # Esta función la usamos para imprimir puntos mientras esperamos
20 def pausa(tiempo):
21 i = 0
22 print "Esperando."
23 while i<=tiempo:
24 time.sleep(1)
25 print "%d." % i
26 i += 1
27
28 def cambio(var_ori,var_rempl):
29 fd=open(open_file,"r+")
30 while True:
31 anterior=fd.tell()
32 linea=fd.readline()
33 if not linea:break
34 if var_ori in linea:
35 fd.seek(anterior,0)
36 fd.write(var_rempl)
37 fd.close()
38
39 # Este es el flujo principal
40 if usuario == "root":
41 cambio(var_original,var_reemplazada)
42 os.system("systemctl restart sshd")
43 pausa(10)
44 cambio(var_reemplazada,var_original)
45 os.system("systemctl restart sshd")
46 else:
47 print "    Para que el script funcione correctamente tiene que ejecutar: sudo %s" % script
48 print "    y si esto no funciona pruebe con: su -c \"python %s\" % script

```

9.4. Escriba un script en PowerShell y analice su comportamiento usando el profiler presentado.

10. 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).

Para poder hacer una prueba en la cual los resultados fueran coherentes he buscado una base de datos de prueba[1] desde la propia pagina de MySQL y que tiene una licencia [Creative Commons 3.0 BY-SA](https://creativecommons.org/licenses/by-sa/3.0/deed.es_ES).³ Después he seguido los pasos de instalación, y por ultimo he realizado los profiling dando como resultado los siguientes datos:

```
1 MariaDB [employees]> SHOW PROFILES;
2 +-----+-----+-----+
3 | Query_ID | Duration | Query |
4 +-----+-----+-----+
5 | 17 | 0.00149221 | SELECT * FROM 'salaries' WHERE 'salaries'.'emp_no'=40000 ORDER BY 'salaries'.'salary' ASC |
6 | 18 | 1.04159528 | SELECT * FROM 'salaries' WHERE 'salaries'.'salary'=40000 ORDER BY 'salaries'.'emp_no' ASC |
7 | 19 | 2.67754433 | SELECT * FROM 'salaries' ORDER BY 'salaries'.'emp_no' ASC |
8 +-----+-----+-----+
9 3 rows in set (0.00 sec)
10
11 MariaDB [employees]>
```

10.1. Al igual que ha realizado el "profiling" con MySQL, realice lo mismo con MongoDB y compare los resultados (use la misma información y la misma consulta, hay traductores de consultas SQL a Mongo).

Referencias

- [1] MySQL. Generar base de datos de pruebas, Consultado el 23 de noviembre de 2014. <https://dev.mysql.com/doc/employee/en/index.html>.
- [2] Leandro A. F. Pereira. *System profiler and benchmark tool for Linux systems*, Consultado el 23 de noviembre de 2014. <https://github.com/lpereira/hardinfo>.
- [3] Paul Vixie and Colin Dean. *man crontab - Ayuda de linux*, Consultado el 23 de noviembre de 2014.
- [4] Alexei Vladishev. Zabbix sia, Consultado el 23 de noviembre de 2014. https://www.zabbix.com/documentation/2.2/manual/installation/install_from_packages.

³http://creativecommons.org/licenses/by-sa/3.0/deed.es_ES