Universidad de Granada

Ingeniería de Servidores

Memoria de la práctica 3

Javier León Palomares

8 de diciembre de 2016

Índice

Índice de figuras

1. Cuestión 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?

Según [1] (ya que no he encontrado referencias oficiales que mencionen lo siguiente), el directorio `/etc/logrotate.d` contiene configuraciones para guardar logs de la actividad de distintos servicios, entre ellos `apt` o `yum`.

- a) El archivo (log) `/var/log/apt/history.log` [2] almacena información acerca de las operaciones que ha realizado `apt`. Si inspeccionamos el directorio mencionado al inicio de la cuestión, el archivo (log) correspondiente a CentOS es `/var/log/yum.log`.
- b) Son versiones anteriores comprimidas de los logs creados por `logrotate`. Ya que las máquinas virtuales aún no contienen versiones previas de estos logs, vamos a comprobar si la configuración para `apt` se corresponde con lo esperado en mi máquina. Para CentOS sólo puedo mostrar los parámetros de `logrotate` y el log inicial.

```
javi@javi-X555LJ:~$ date
sáb dic  3 14:13:20 CET 2016
javi@javi-X555LJ:~$ cat /etc/logrotate.d/apt
/var/log/apt/term.log {
  rotate 12
  monthly
  compress
  missingok
  notifempty
}

/var/log/apt/history.log {
  rotate 12
  monthly
  compress
  missingok
  notifempty
}
```

Figura 1: Según `man logrotate`, lo que vemos significa que las versiones de los logs `term.log` y `history.log` son almacenadas comprimidas durante 12 rotaciones (cada una de ellas mensual) sólo si tienen algún contenido.

Deberíamos, pues, encontrar en `/var/log/apt` 12 versiones mensuales comprimidas de los dos archivos mencionados. Buscamos específicamente sus nombres en dicho directorio y obtenemos lo esperado:

```
sáb dic  3 14:16:26 CET 2016
javi@javi-X555LJ:~$ ls /var/log/apt/ | grep -E 'history|term'
history.log
history.log.10.gz
history.log.11.gz
history.log.12.gz
history.log.1.gz
history.log.2.gz
history.log.3.gz
history.log.4.gz
history.log.5.gz
history.log.6.gz
history.log.7.gz
history.log.8.gz
history.log.9.gz
term.log
term.log.10.gz
term.log.11.gz
term.log.12.gz
term.log.1.gz
term.log.2.gz
term.log.3.gz
term.log.4.gz
term.log.5.gz
term.log.6.gz
term.log.7.gz
term.log.8.gz
term.log.9.gz
```

Figura 2: Efectivamente, vemos 12 archivos numerados y comprimidos con extensión .gz además de las versiones actualizadas.

De igual manera, para CentOS existe un archivo que almacena los logs de su gestor de paquetes:

```
jlp@localhost sáb dic 03:~$ cat /etc/logrotate.d/yum
/var/log/yum.log {
    missingok
    notifempty
    size 30k
    yearly
    create 0600 root root
}
```

Figura 3: Configuración de `logrotate` para `yum`. Se observa que crea un nuevo log anualmente si supera los 30 kilobytes de tamaño, definiendo además los permisos y el propietario.

Finalmente, una muestra del primer log se encuentra en la siguiente captura:

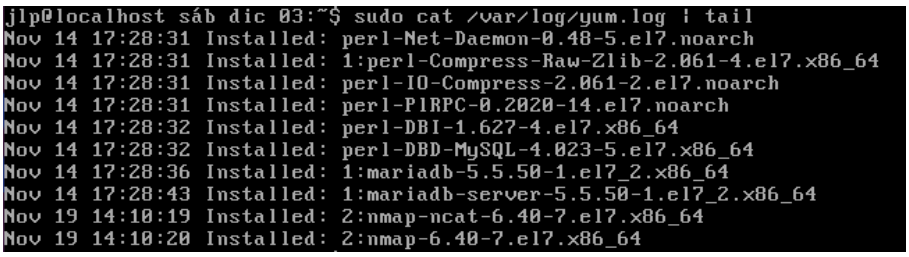

Figura 4: Contenido parcial del log de `yum`. Aparece la instalación de Perl, MariaDB y nmap.

2. Cuestión 2: ¿Qué archivo ha de modificar para programar una tarea? Escriba la línea necesaria para ejecutar una vez al día una copia del directorio /codigo a /seguridad/$fecha donde $fecha es la fecha actual (puede usar el comando date).

He de modificar el archivo correspondiente a mi usuario contenido en `/var/spool/cron/crontabs` [3]. Se recomienda editarlo mediante `crontab -e`.

El primer paso es escribir el comando que nos permite realizar la copia:

```
cp -r /home/jlp/codigo /home/jlp/seguridad/$(date +%d-%b-%Y)
```

Tras guardarlo en un fichero, por ejemplo `ej2script.sh`, editamos nuestro crontab (`sudo crontab -e`) utilizando la información disponible en `man 5 crontab`:

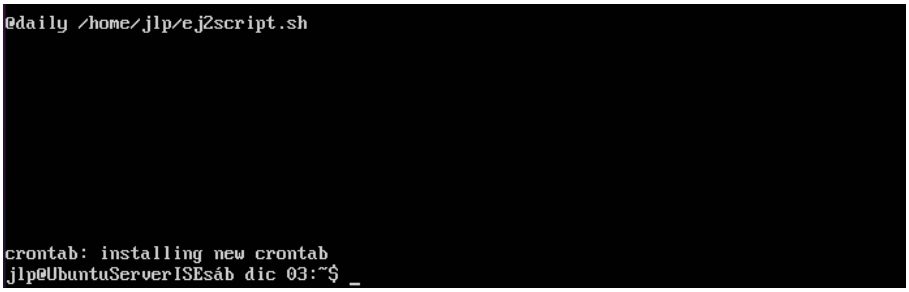

Figura 5: Añadimos la línea `@daily /home/jlp/ej2script.sh`, que programa la ejecución para las 12 de la noche de cada día.

Finalmente, la orden es ejecutada en el momento esperado:

```
jlp@UbuntuServerISEdom dic 04:~$ ls -l seguridad/ --time-style="+%b %d %H:%M"
total 1
drwxr-xr-x 2 root root 1024 dic 04 00:00 04-dic-2016
jlp@UbuntuServerISEdom dic 04:~$
```

Figura 6: El directorio tiene como fecha de creación las 00:00 del día siguiente, como estaba programado.

3. Cuestión 3: Pruebe a ejecutar el comando, conectar un dispositivo USB y vuelva a ejecutar el comando. Copie y pegue la salida del comando. (considere usar dmesg | tail). Comente qué observa en la información mostrada.

Para simplificar el proceso, he utilizado mi máquina para conectar un USB directamente en lugar de hacerlo con una máquina virtual. El dispositivo elegido es una memoria flash con una capacidad teórica de 32 GB.

```
javi@javi-X555LJ:~$ dmesg | tail
[   67.224665] 0x0000 : 14 00 10 51 03 00 00 00 01 00 00 00 01 00 00 00
[   67.224669] 0x0010 : 01 00 00 00 01 00 00 00
[   67.224671] pPacket->data: ffff880217a8b500, len = 20
[   67.224672] 0x0000 : 03 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00
[   67.224676] 0x0010 : 01 00 00 00
[   67.224678] PCIKickOutCmd (TxCpuIdx = 3)
[   67.224678] SendAndesAFH: <--
[   67.224687] ==>INT_SOURCE_CSR_7630_HCCA_DMA_DONE
[   67.229683] wlan0: associated
[   67.229690] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
javi@javi-X555LJ:~$ dmesg | tail
[  556.521254] usbcore: registered new interface driver uas
[  558.772381] scsi 4:0:0:0: Direct-Access     Kingston DataTraveler 2.0 1100 PQ: 0 ANSI: 4
[  558.773054] sd 4:0:0:0: Attached scsi generic sg2 type 0
[  558.773951] sd 4:0:0:0: [sdb] 61440000 512-byte logical blocks: (31.4 GB/29.2 GiB)
[  558.774483] sd 4:0:0:0: [sdb] Write Protect is off
[  558.774495] sd 4:0:0:0: [sdb] Mode Sense: 43 00 00 00
[  558.777967] sd 4:0:0:0: [sdb] No Caching mode page found
[  558.777976] sd 4:0:0:0: [sdb] Assuming drive cache: write through
[  558.783237]  sdb: sdb1
[  558.785849] sd 4:0:0:0: [sdb] Attached SCSI removable disk
javi@javi-X555LJ:~$ date
sáb dic  3 18:17:28 CET 2016
```

Figura 7: En la primera ejecución de `dmesg` vemos información relacionada con la inicialización de las redes en el equipo (acababa de ser encendido). Al conectar un pen drive, la misma orden nos muestra datos como el nombre, tamaño de bloque, tamaño total, que no está protegido para escritura, que no tiene caché o su punto de montaje. En principio no se ven indicios de problemas con el dispositivo.

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

En primer lugar, lo encontramos en *Conjuntos de recopiladores de datos ->Sistema*. Se ejecuta con click derecho ->*Iniciar*.

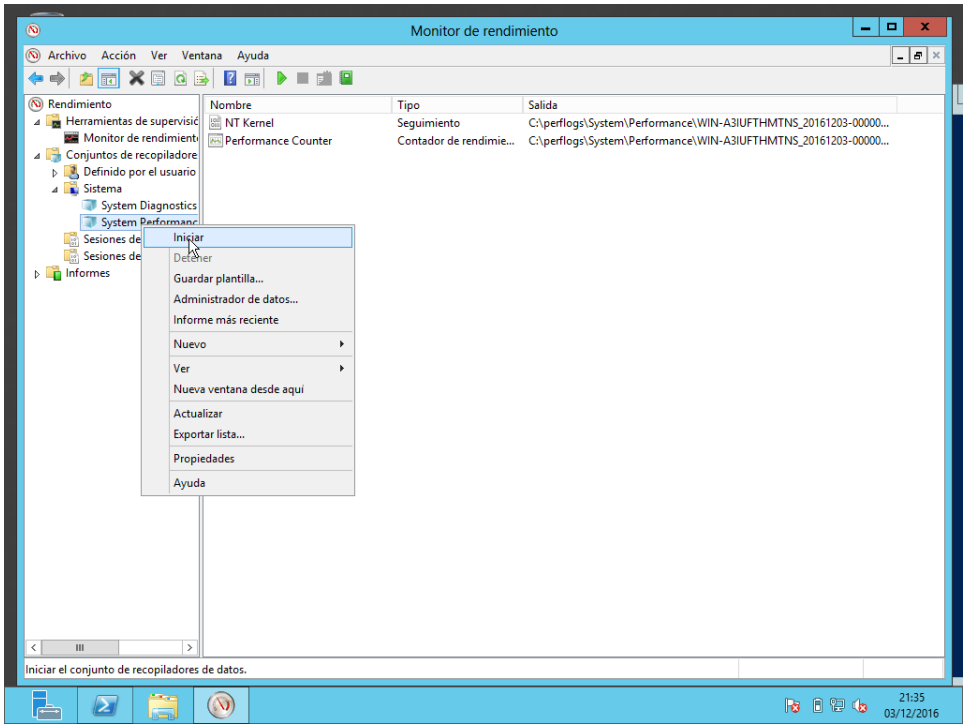

Figura 8: Iniciamos el monitor de *System Performance*.

Al terminar de ejecutarse, nos presenta los resultados. En las páginas siguientes se comentan algunas secciones.

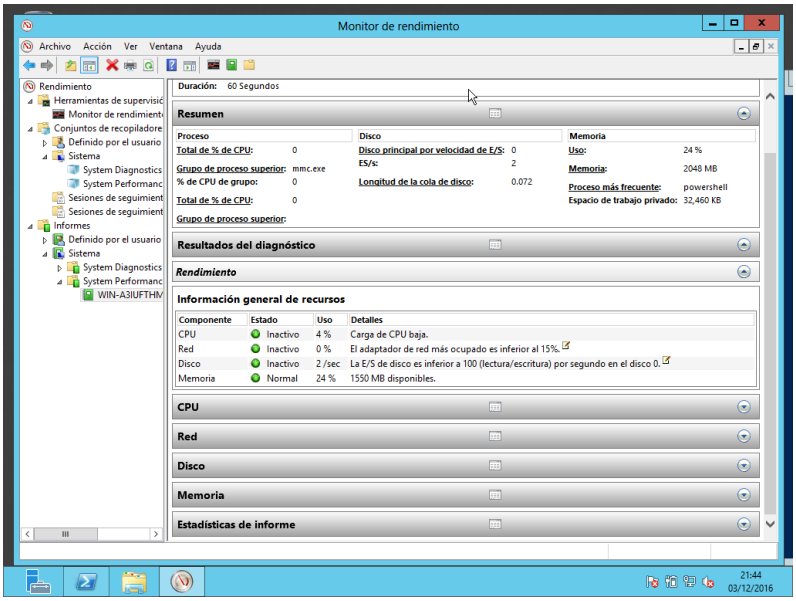

Figura 9: Aquí tenemos un resumen de los resultados del monitor. Windows Server no estaba ejecutando nada en especial, y por ello los componentes presentan una baja tasa de uso.

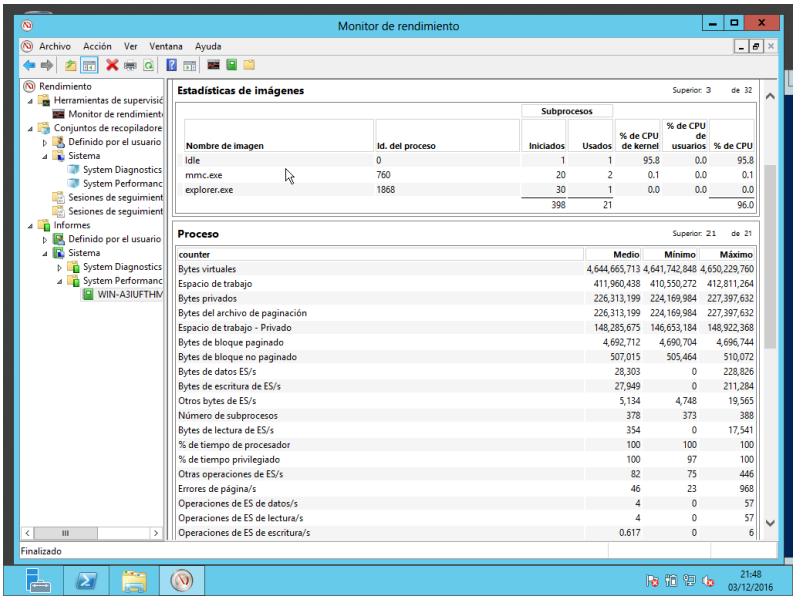

Figura 10: En esta imagen se observan los tres procesos más activos en la parte superior, aunque el identificado por *Idle* prevalece claramente (en porcentaje de CPU) debido a que el sistema no tenía nada en especial que hacer. En la parte inferior aparecen diversos datos, como las operaciones de entrada y salida, que han sido muy pocas, o el tiempo de procesador, que ha sido casi todo privilegiado; todo esto concuerda con la poca actividad de Windows Server fuera de estar en espera.

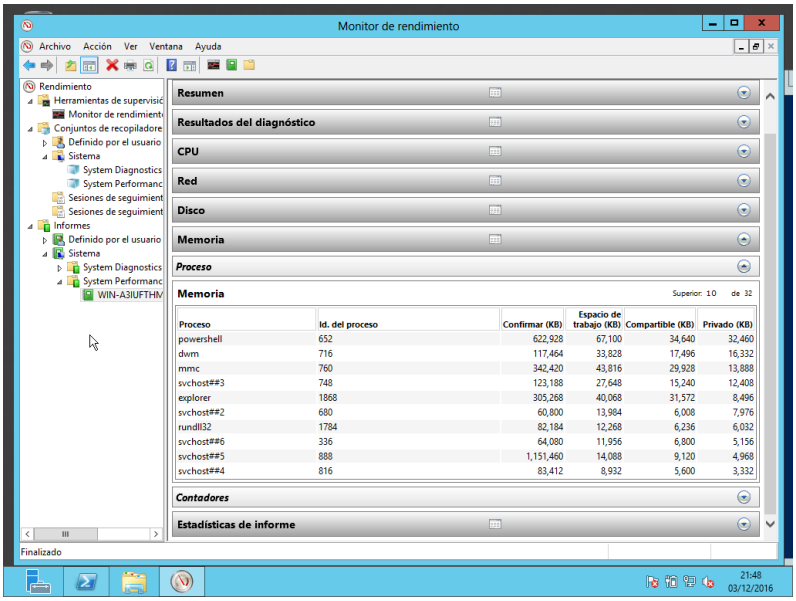

Figura 11: Esta sección nos muestra algunos procesos activos en esos momentos, junto con sus identificadores y la memoria que tenían asignada (separada además en potencialmente compartida con otros procesos o privada). De nuevo, no vemos indicios de un uso intensivo del recurso.

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

El paso inicial es seleccionar la creación de un nuevo recopilador de datos definido por el usuario (figura 12). Se nos solicitará un nombre para este recopilador, y también deberemos elegir cómo crearlo (figura 13). En la siguiente pantalla elegiremos los tipos de datos que queremos obtener (figura 14). Posteriormente, seleccionaremos los contadores de datos relativos al procesador, al proceso y al servicio web (figura 15), que serán listados junto con el intervalo de muestra en la ventana que vemos en la figura 16. Para finalizar, diremos dónde queremos que se guarden los datos (figura 17) y crearemos el recopilador de datos (figura 18).

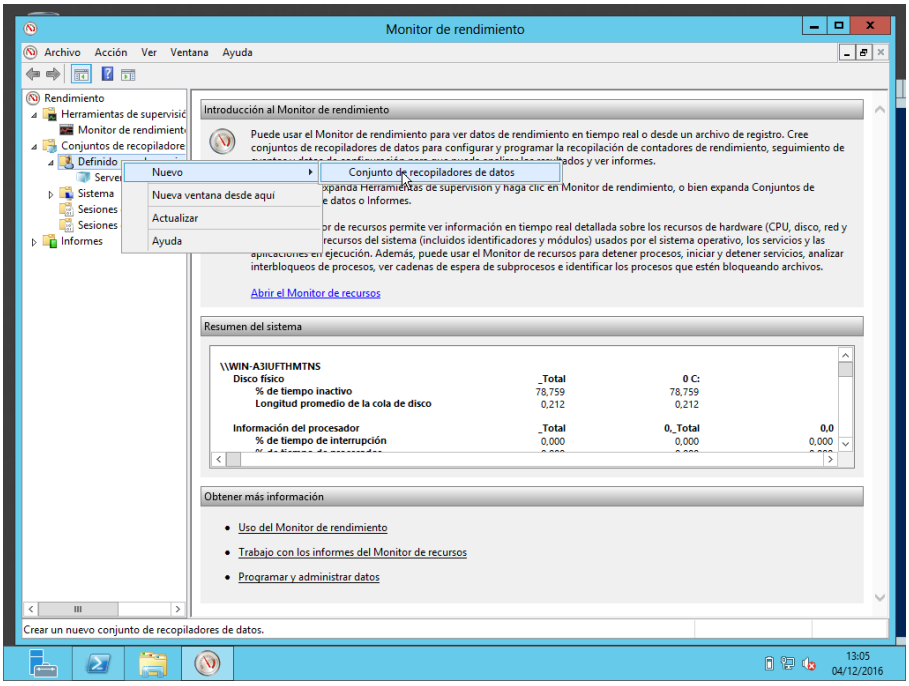

Figura 12: Iniciamos la creación del recopilador de datos.

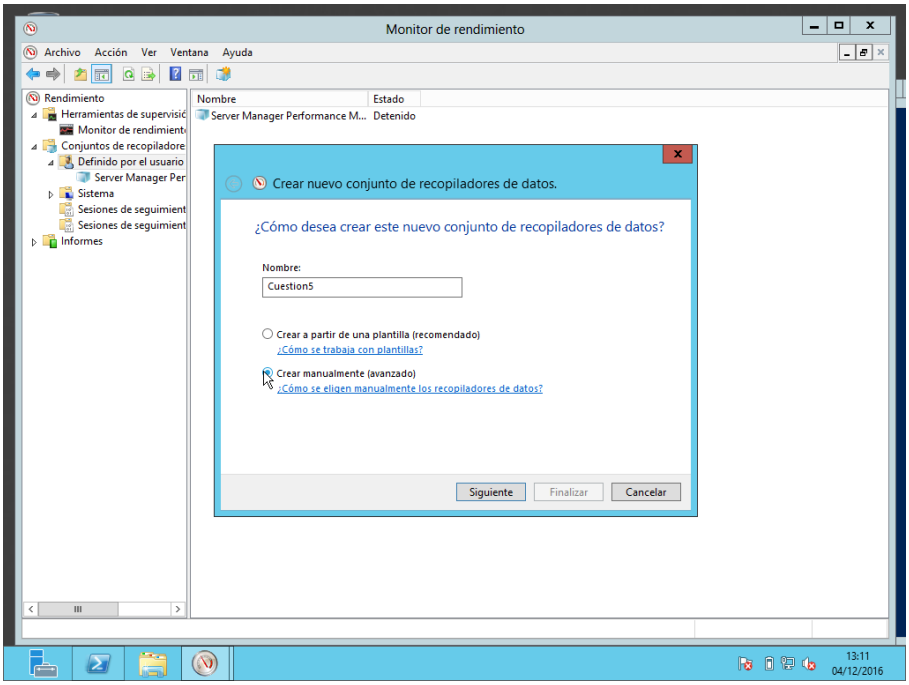

Figura 13: Selección del nombre del recopilador y el modo de creación.

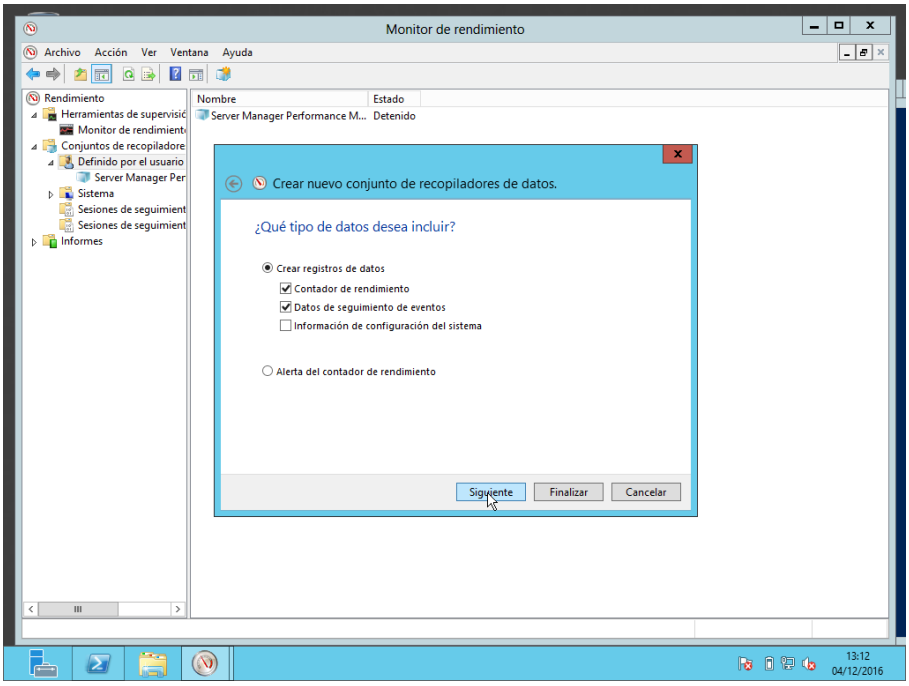

Figura 14: Selección de los tipos de datos a incluir.

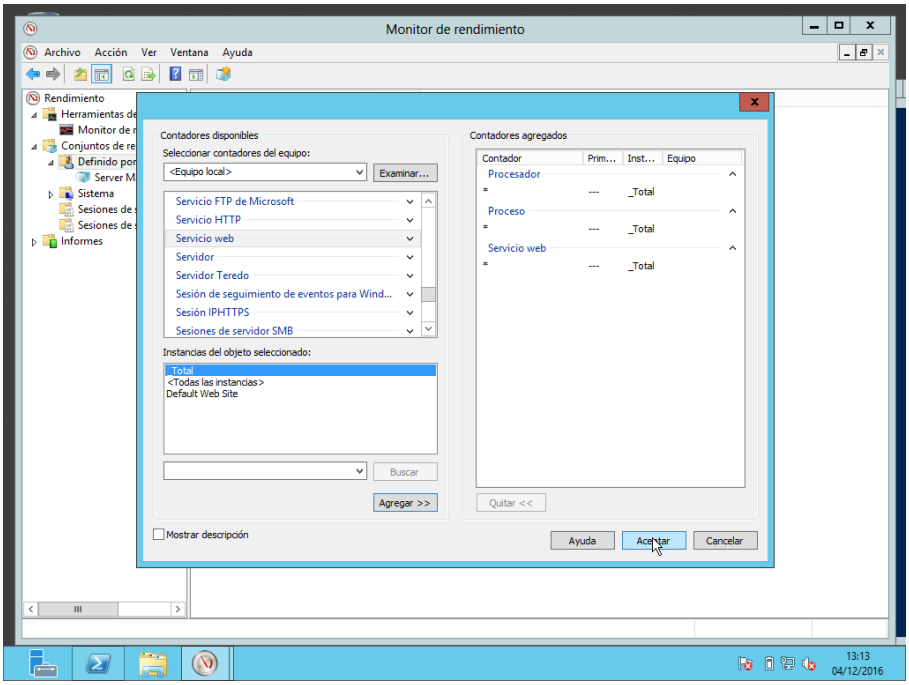

Figura 15: Añadimos los contadores de datos de procesador, proceso y servicio web.

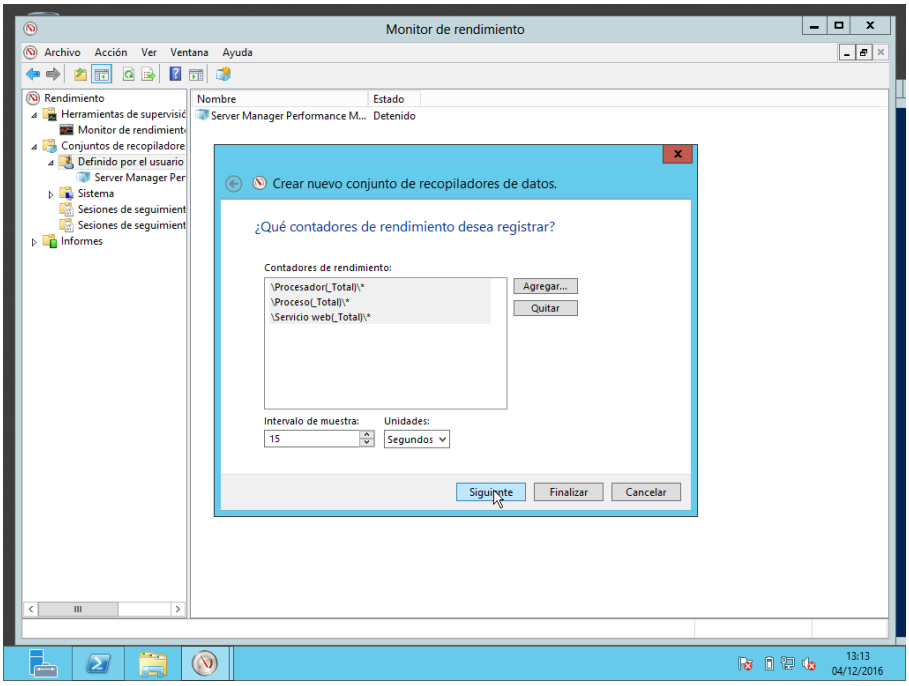

Figura 16: Aquí aparecen los contadores de datos seleccionados en el paso anterior, junto con el intervalo de muestra de 15 segundos.

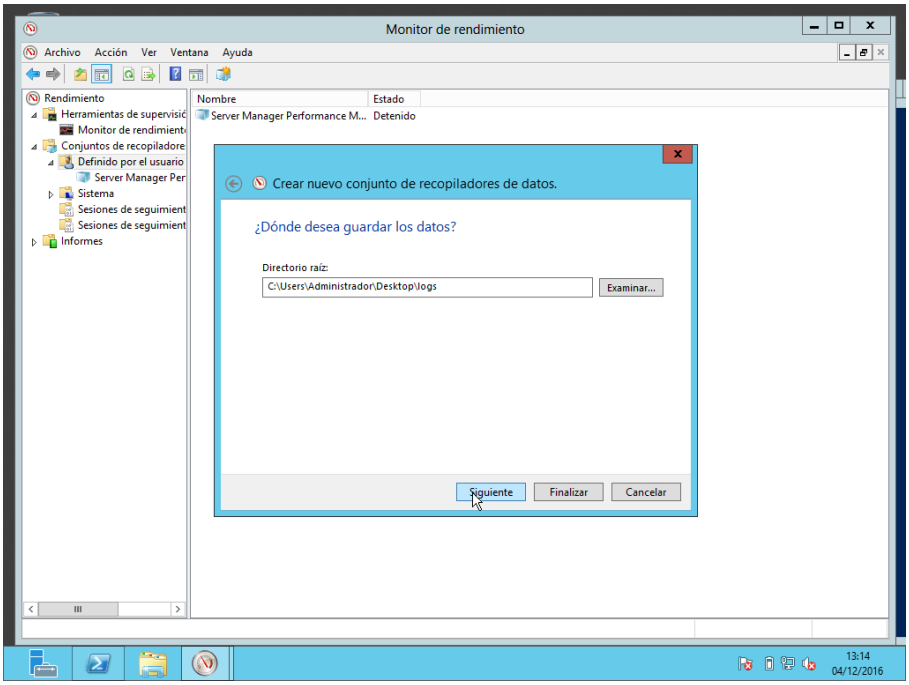

Figura 17: Como penúltima etapa hemos de elegir dónde se guardarán los datos obtenidos por el recopilador.

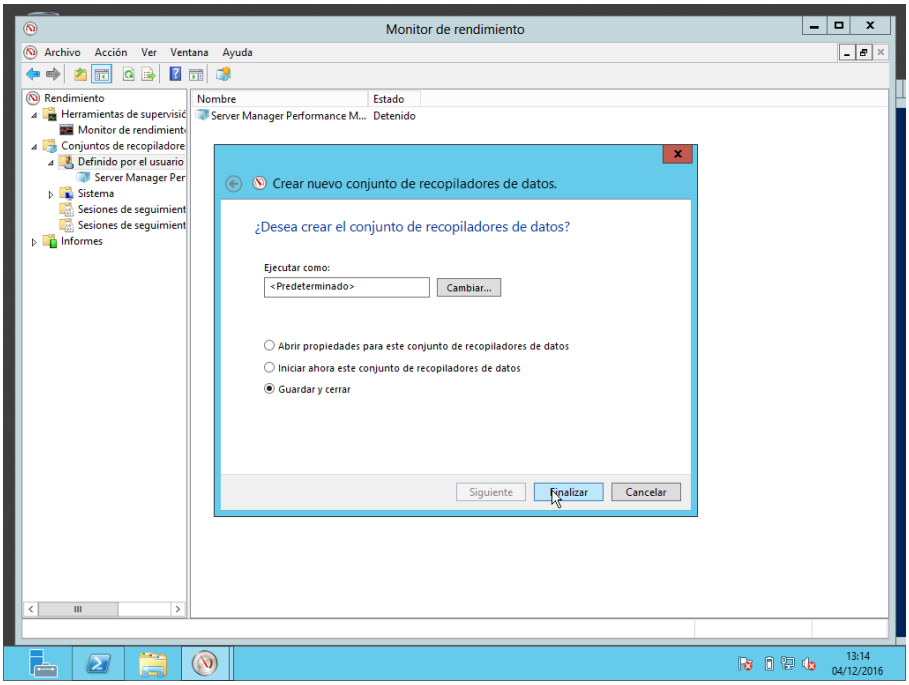

Figura 18: Sólo queda guardar el recopilador de datos.

Una vez realizado este proceso, es el momento de probar el recopilador de datos. En primer lugar, examinaremos el sistema cuando está en reposo:

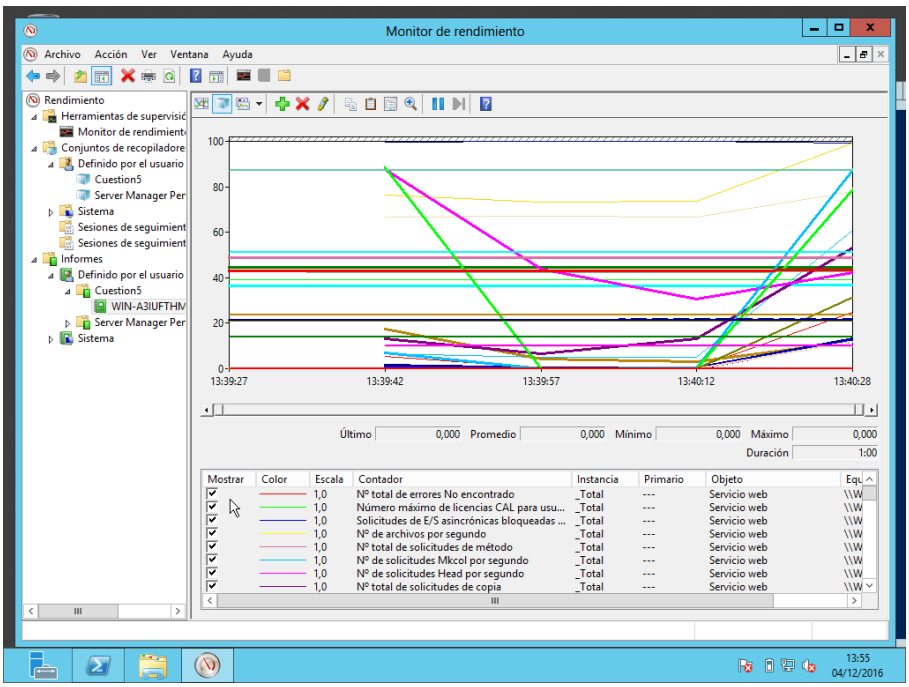

Figura 19: Gráfica con todos los parámetros.

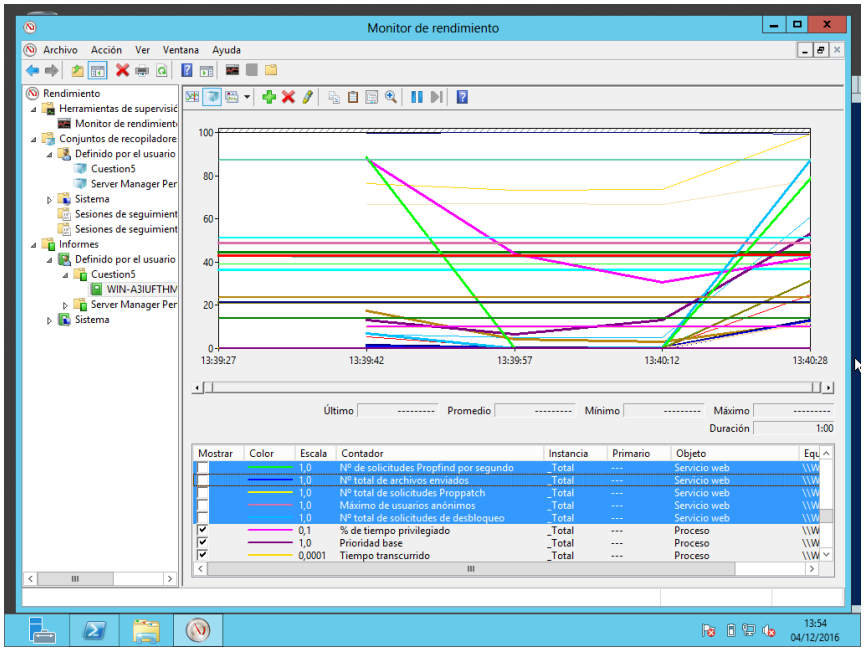

Figura 20: Gráfica sin los parámetros relativos al servicio web.

Respecto al servidor web, si comparamos las figuras 19 y 20 observamos que la gráfica apenas cambia si ocultamos sus parámetros; esto es debido a que la mayoría se refieren a la actividad del servidor, la cual es prácticamente inexistente en nuestro caso.

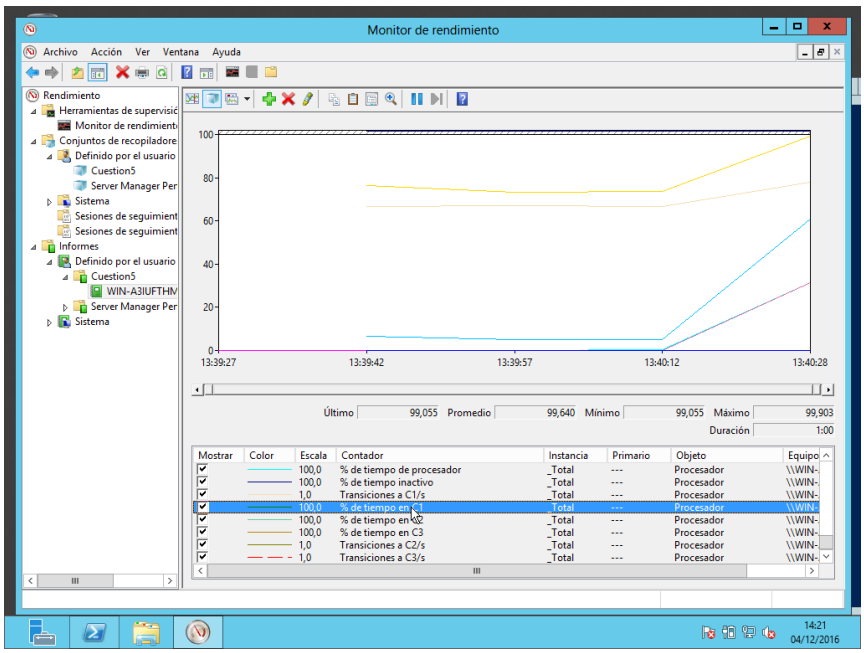

Figura 21: Gráfica con los parámetros del procesador, que nos indican que apenas ha tenido actividad.

Centrándonos en el procesador (figura 21), los datos indican que no ha tenido mucho que hacer. Según [4], los estados C son diferentes niveles de actividad del procesador, siendo C0 el nivel activo y los demás distintos modos de inactividad; teniendo esto en cuenta, sabemos que el procesador no ha hecho casi nada al ver que el porcentaje de tiempo en el nivel C1 es cercano al 100 %; además, el porcentaje de tiempo inactivo también roza el 100 %, como evidencia la línea azul de la parte superior. Por último, el aumento en los parámetros hasta el momento 13:40:28 probablemente se deba a que he empezado a utilizar otra vez la máquina para detener manualmente la ejecución del recopilador, lo que se traduce en interrupciones de dispositivos de entrada (los parámetros involucrados son DPCs en cola/s, interrupciones/s, porcentaje de tiempo de usuario, porcentaje de tiempo de procesador y transiciones a nivel C1/s).

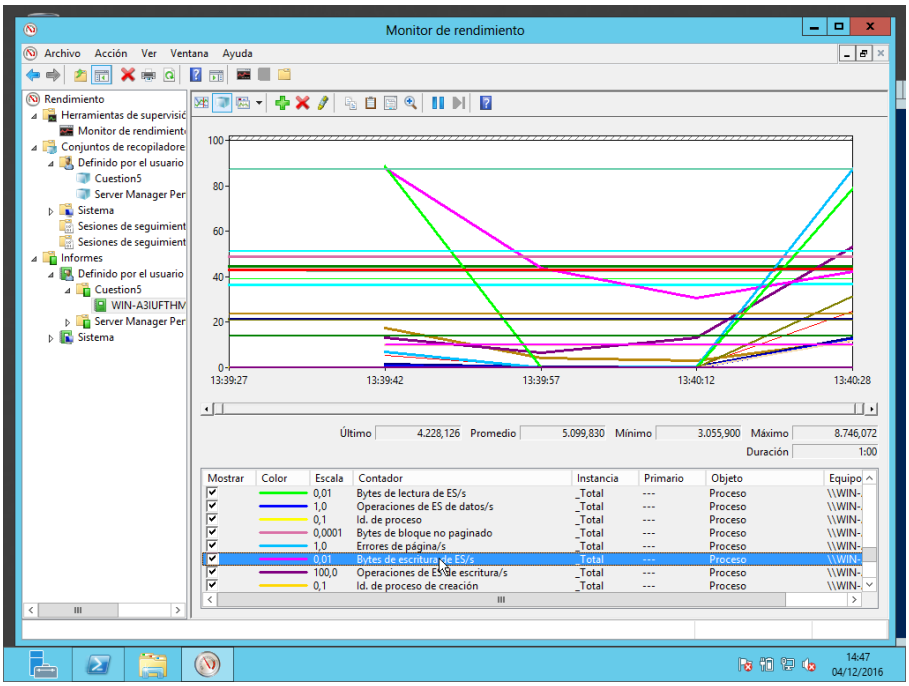

Figura 22: Gráfica con los datos de proceso.

Para terminar con este análisis, tenemos los contadores relacionados con procesos en la figura 22. Muchas de las líneas que se mantienen estables tienen que ver con la memoria principal, y es coherente debido a que no ha habido una tasa de creación o finalización de procesos significativa durante el tiempo de monitorización. Las otras líneas tienen que ver con la entrada/salida, y sus variaciones podrían deberse tanto a mi interacción con el sistema como a la escritura de los datos obtenidos por parte del recopilador.

Una vez monitorizado el sistema en reposo, podemos probar a darle algo de trabajo para ver cómo cambian los datos; he utilizado Internet Explorer para este fin. Las gráficas se muestran a continuación:

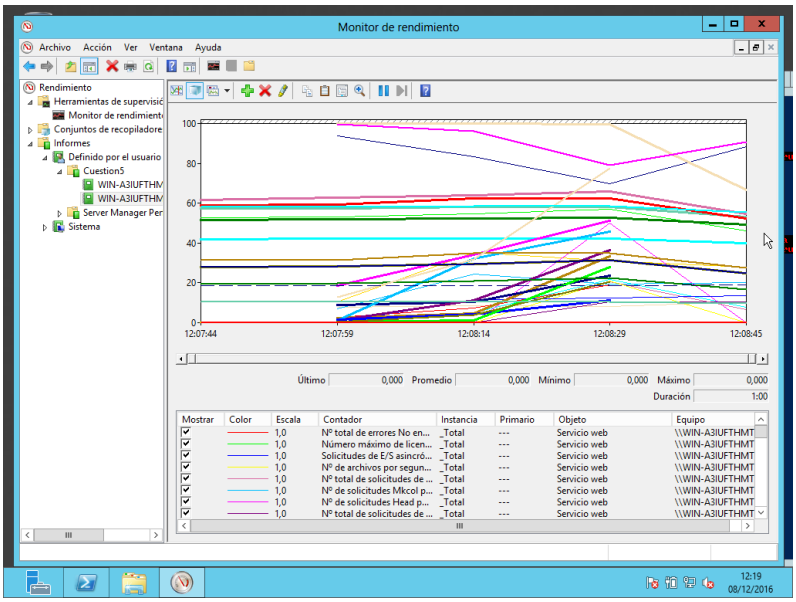

Figura 23: Gráfica que muestra todos los parámetros.

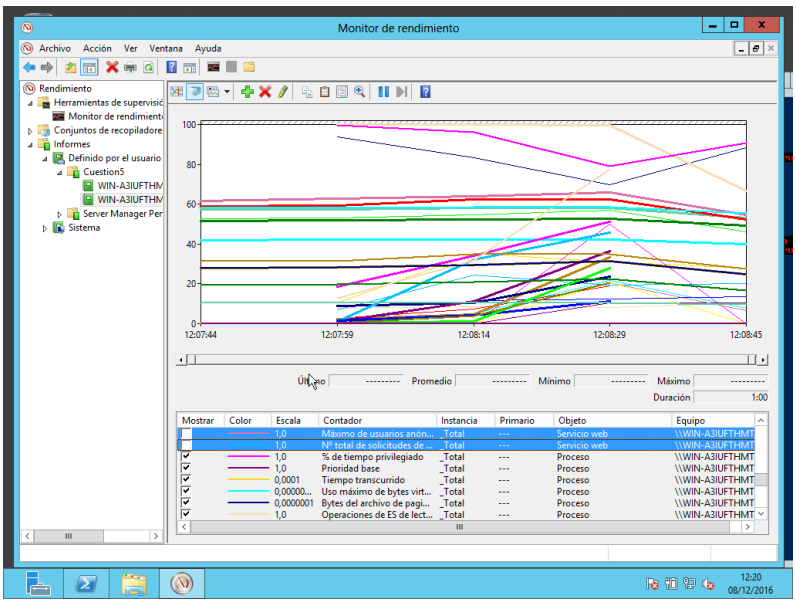

Figura 24: Gráfica sin los parámetros del servidor web.

Como en el caso de la monitorización anterior, en las figuras 23 y 24 también se observa que ocultar las líneas que representan al servicio web apenas cambia el conjunto ya que sigue sin tener carga de trabajo.

Por ello, vamos a ver la información relativa al procesador y a los procesos:

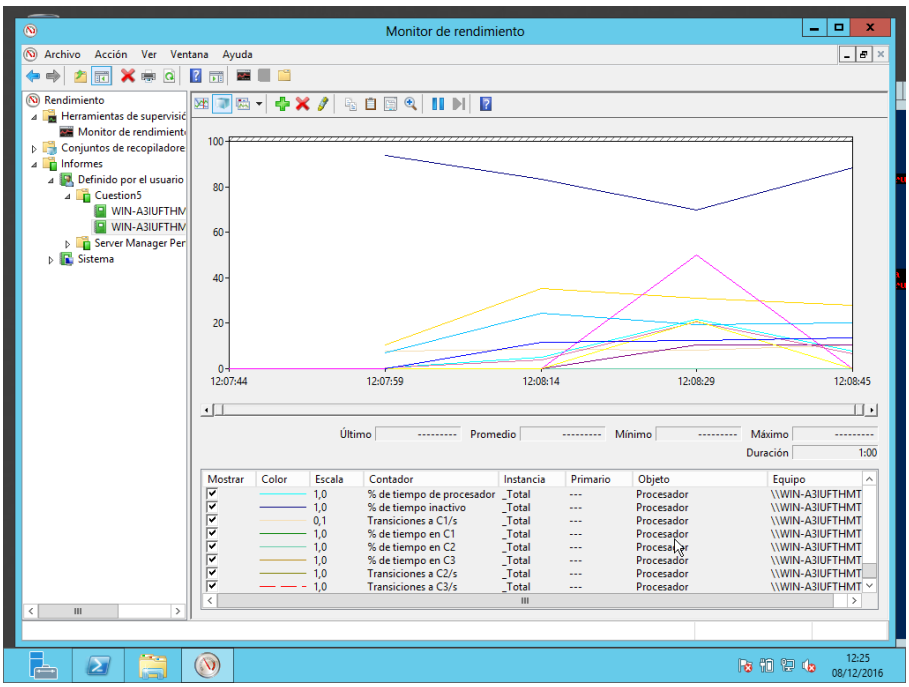

Figura 25: Gráfica correspondiente al procesador.

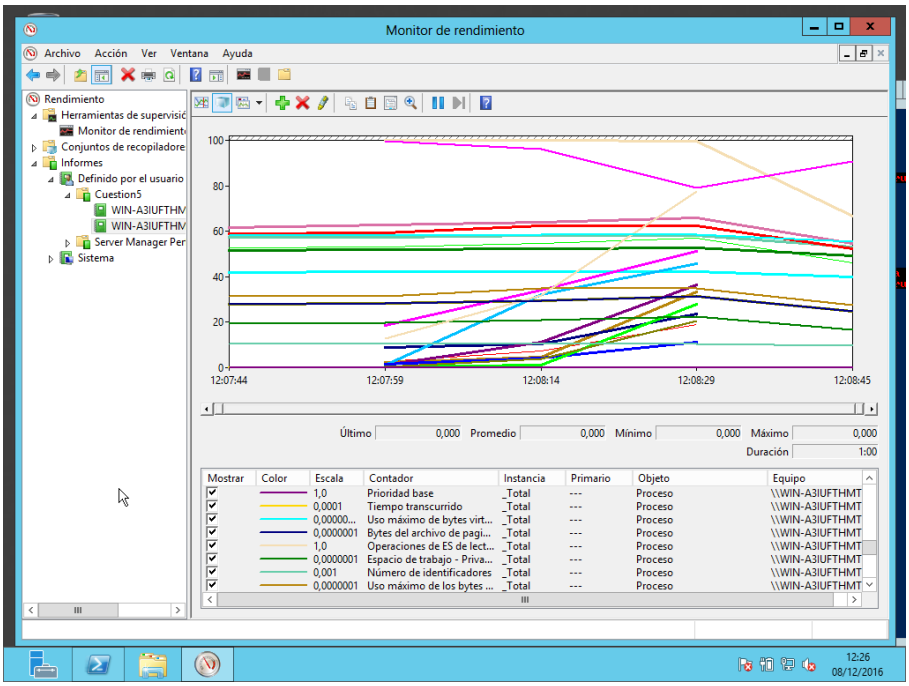

Figura 26: Gráfica correspondiente a los procesos.

En la figura 25 vemos una bajada del tiempo de inactividad y del tiempo en el nivel C1 del procesador; esto se corresponde con los momentos en los que estaba usando Internet Explorer para añadir carga al sistema. También vemos una subida de los contadores relacionados con mi interacción, como las interrupciones/s (línea amarilla superior).

De la figura 26, que representa las variables de los procesos, podemos destacar la actividad de E/S de datos (líneas ascendentes centrales) que aumenta durante el tiempo en el que utilizo Internet Explorer. También se ve que disminuye el porcentaje de tiempo privilegiado (línea rosa superior) debido a que se está ejecutando un proceso de usuario y que, cuando cierro Internet Explorer, baja el tiempo de procesador (línea marrón claro superior).

6. Cuestión 6: Visite la web del proyecto y acceda a la demo que proporcionan (http://demo.munin-monitoring.org/) donde se muestra cómo monitorizan un servidor. Monitorice varios parámetros y haga capturas de pantalla de lo que está mostrando comentando qué observa.

A partir de esta página podemos acceder a la monitorización de diversos parámetros.

Para empezar, vamos a monitorizar el uso de CPU. Elegimos la gráfica que representa un mes de actividad:

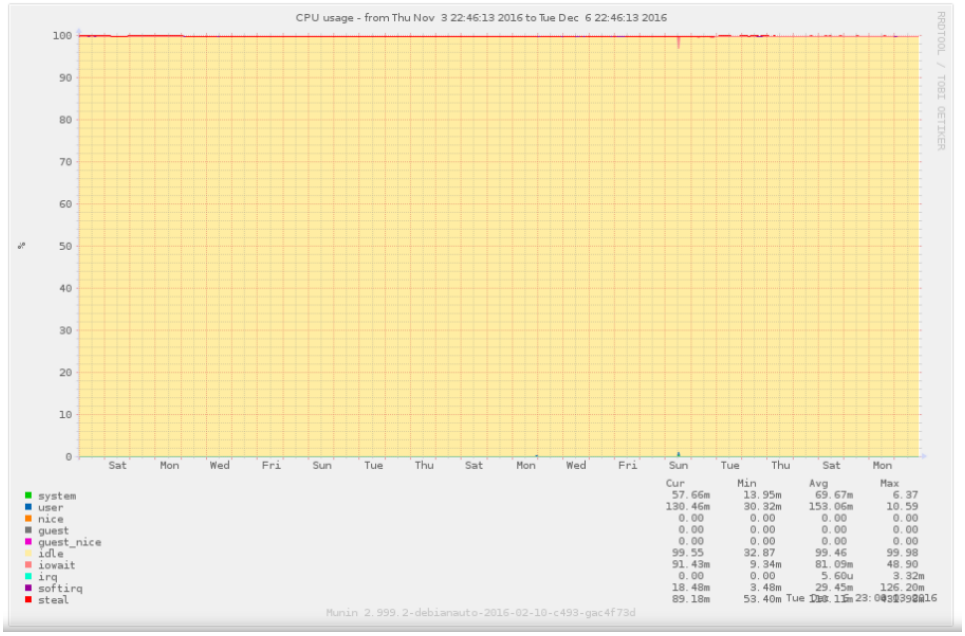

Figura 27: En esta gráfica queda patente que, durante la práctica totalidad del tiempo, la CPU no ha tenido actividad. Lo único destacable es la existencia de algunos picos pequeños y muy puntuales de actividad debida al sistema.

Ahora vamos a ver el consumo de memoria. Esta vez escogemos la gráfica semanal:

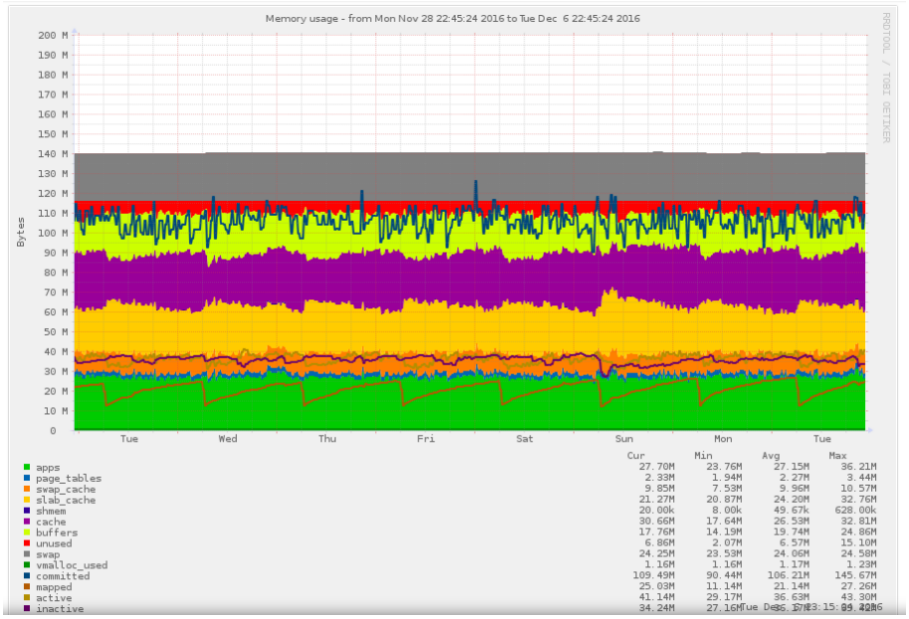

Figura 28: En consonancia con la gráfica de CPU, vemos que el uso de memoria consiste en una gran proporción de cachés, junto con swap y memoria sin usar. Aparte de las aplicaciones de usuario, no parece haber más indicios de algún tipo de actividad seria.

Para concluir, veamos los procesos del servidor en el último día:

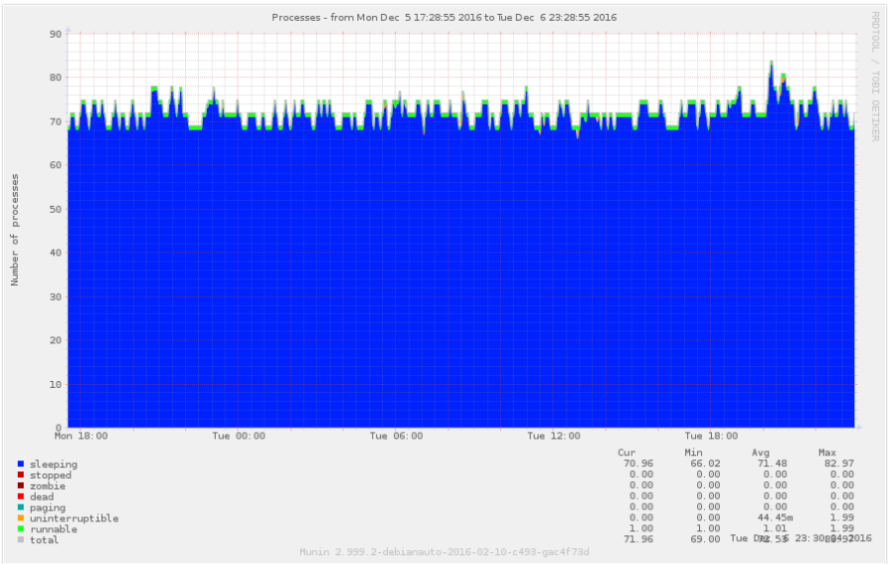

Figura 29: El servidor ha tenido una media de 71-72 procesos, pero casi todos se han encontrado en estado `sleeping`, y los pocos restantes estaban en la cola de ejecutables, por lo que no han representado una carga de trabajo relevante.

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

Voy a hacer un resumen del artículo del segundo enlace.

El artículo empieza comentando que `strace` es una herramienta para administradores (y no para programadores), y que es fácil usarlo.

A continuación, vemos un ejemplo práctico de cómo `strace` muestra todas las llamadas al sistema del programa con el que lo ejecutamos; en este caso, `cat`. Podemos observar la llamada que ejecuta el programa, algunos intentos de abrir librerías y, finalmente, la lectura del contenido del archivo y su posterior escritura por pantalla (la salida estándar).

Después, nos destaca y explica algunas de las llamadas más importantes, como `open` para abrir el archivo pasado como argumento a `cat`.

Por último, nos habla de dos ejemplos reales: encontrar la ubicación de los logs de un servicio a partir de los archivos que abre con `open` al iniciarse; y encontrar errores a partir de llamadas al sistema que devuelvan -1 según la salida de `strace`.

Para probarlo, he ejecutado `strace` al mostrar el contenido del script de Python del ejercicio siguiente (`cat script8.py`). Al ser la salida demasiado grande para mostrarla entera en este documento, he adjuntado un archivo con la totalidad de las líneas.

De esta prueba podemos destacar la línea en la que se ejecuta el programa propiamente (1), como se preveía. También podemos ver cómo abre el script de Python en modo sólo lectura (33) y procede a leer su contenido (36), exactamente 93 caracteres; posteriormente, muestra todos ellos por la salida estándar (37) y cierra los descriptores de archivo que ha estado usando (42-44). Ninguna de estas operaciones devuelve -1, así que sabemos que se han realizado con éxito.

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

He escrito un script en Python que crea una lista, la invierte, y finalmente la devuelve a su estado original ordenándola:

```
import cProfile

mylist = range(0,2000000)
not_reverse = [x for x in reversed(mylist)].sort()
```

Analizando los resultados de ejecutar el profiler *cProfiler* con el script mediante `python -m cProfile -s 'cumulative' script8.py` obtenemos la figura 30. En ella podemos ver una serie de columnas:

- `ncalls` es el número de llamadas a la función correspondiente.
- `tottime` es el tiempo de la función sin contar llamadas a otras funciones.
- `percall` es el tiempo por llamada (`tottime`/`ncalls`).
- `cumtime` es el tiempo consumido por la función y sus llamadas a otras funciones.
- `percall` es el tiempo por llamada pero respecto al tiempo acumulado: `cumtime`/`ncalls`.
- `filename:lineno(function)` nos da información sobre la función.

```
javi@javi-X555LJ:~$ python -m cProfile -s 'cumulative' script8.py
         6 function calls in 0.135 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.088    0.088    0.135    0.135 script8.py:1(<module>)
        1    0.023    0.023    0.023    0.023 {range}
        1    0.023    0.023    0.023    0.023 {method 'sort' of 'list' objects}
        1    0.000    0.000    0.000    0.000 cProfile.py:5(<module>)
        1    0.000    0.000    0.000    0.000 cProfile.py:66(Profile)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


javi@javi-X555LJ:~$ date
dom dic  4 19:56:38 CET 2016
```

Figura 30: Salida del profiler al ejecutarlo junto con el script de Python.

Los números nos indican lo que vemos en el código del script: hay un archivo que contiene una llamada a `range` para generar una lista de enteros en orden ascendente, emplea cierto tiempo creando otra lista dándole los elementos de la lista inicial en orden inverso, y finalmente ordena la nueva lista; el resto de las funciones corresponden al profiler. Por ello, el tiempo acumulado del script (0,135s) es aproximadamente la suma de su tiempo (0,088s) más el tiempo de la llamada a `range` (0,023s) más el tiempo de la llamada a `sort` (0,023s). El tiempo por llamada equivale al tiempo total debido a que sólo se realiza una llamada para cada función.

9. Cuestión 9: Acceda a la consola mysql (o a través de phpMyAdmin) y muestre el resultado de mostrar el "profile" de una consulta (la creación de la BD y la consulta la puede hacer libremente).

En primer lugar, para poder realizar una consulta de prueba he creado una base de datos llamada "Cuestion9" con una tabla ("Prueba") que consta de 4 columnas. El resultado de introducir algunas filas en ella es el siguiente:

	ID	ENTERO_1	ENTERO_2	ENTERO_3
Editar Copiar Borrar	A	1	1	1
Editar Copiar Borrar	B	2	2	2
Editar Copiar Borrar	C	3	3	3
Editar Copiar Borrar	D	4	4	4
Editar Copiar Borrar	E	5	5	5

Figura 31: Tabla "Prueba" con 5 filas.

El siguiente paso es hacer una consulta. La consulta más inmediata es obtener toda la información almacenada en cada fila de la tabla, lo cual haremos mediante `select * from Prueba`:

```
SET profiling = 1;
SELECT * FROM Prueba;
SHOW PROFILES;
```

Figura 32: Como se indica en los enlaces proporcionados en el guión de la práctica, la primera línea habilita el profiling y la tercera nos muestra los resultados.

Tras ejecutar lo anterior, obtenemos un desglose de las acciones que se han realizado:

Query_ID	Duration	Query
4	0.00077200	SHOW SESSION VARIABLES LIKE 'FOREIGN_KEY_CHECKS'
5	0.00004775	SELECT DATABASE()
6	0.00015725	SELECT * FROM Prueba LIMIT 0, 25
7	0.00020325	SELECT TABLE_NAME FROM information_sch...
8	0.00205650	SELECT *, `TABLE_SCHEMA` ...
9	0.00006675	SELECT COUNT(*) FROM `Cuestion9`.`Prueba`
10	0.00017600	SELECT TABLE_NAME FROM information_sch...
11	0.00009725	SHOW INDEXES FROM `Cuestion9`.`Prueba`
12	0.00005175	SELECT @@have_profiling
13	0.00004625	SHOW CREATE TABLE `Cuestion9`.`Prueba`
14	0.00022425	SHOW FULL COLUMNS FROM `Cuestion9`.`Prueba`
15	0.00094550	SHOW SESSION VARIABLES LIKE 'FOREIGN_KEY_CHECKS'
16	0.00082375	SHOW SESSION VARIABLES LIKE 'FOREIGN_KEY_CHECKS'
17	0.00102950	SHOW SESSION VARIABLES LIKE 'FOREIGN_KEY_CHECKS'
18	0.00006200	SELECT DATABASE()

Figura 33: Podemos ver cómo ha realizado el proceso *PhpMyAdmin*: selecciona la base de datos ("Cuestion9"), extrae la información de las filas de la tabla "Prueba", cuenta el número de filas y muestra información sobre la propia tabla para su visualización.

Además, adicionalmente podemos ver los detalles acerca de cómo se ha desarrollado una operación en particular. Por ejemplo, la de recuperar la información contenida en la tabla consta de las siguientes etapas con sus tiempos correspondientes:

Orden	Estado	Tiempo
1	Starting	23 µs
2	Checking Permissions	3 µs
3	Opening Tables	15 µs
4	Init	9 µs
5	System Lock	4 µs
6	Optimizing	2 µs
7	Statistics	8 µs
8	Preparing	6 µs
9	Executing	2 µs
10	Sending Data	28 µs
11	End	2 µs
12	Query End	3 µs
13	Closing Tables	3 µs
14	Freeing Items	4 µs
15	Cleaning Up	5 µs

Estado	Tiempo Total	% de Tiempo	Llamadas	ø de Tiempo
Sending Data	28 µs	23.93%	1	28 µs
Starting	23 µs	19.66%	1	23 µs
Opening Tables	15 µs	12.82%	1	15 µs
Init	9 µs	7.69%	1	9 µs
Statistics	8 µs	6.84%	1	8 µs
Preparing	6 µs	5.13%	1	6 µs
Cleaning Up	5 µs	4.27%	1	5 µs
System Lock	4 µs	3.42%	1	4 µs
Freeing Items	4 µs	3.42%	1	4 µs
Checking Permissions	3 µs	2.56%	1	3 µs
Query End	3 µs	2.56%	1	3 µs
Closing Tables	3 µs	2.56%	1	3 µs
Optimizing	2 µs	1.71%	1	2 µs
Executing	2 µs	1.71%	1	2 µs
End	2 µs	1.71%	1	2 µs

Figura 34: Profiling de la línea `select * from Prueba`.

Referencias

[1] "Comando logrotate. https://linuxconfig.org/logrotate," consultado el 3 de Diciembre de 2016.

[2] "Ayuda de Ubuntu. https://help.ubuntu.com/community/ListInstalledPackagesByDate," consultado el 3 de Diciembre de 2016.

[3] "Ayuda de Ubuntu. https://help.ubuntu.com/community/CronHowto," consultado el 3 de Diciembre de 2016.

[4] "Entradas acerca de los estados C, por un autor reconocido por Intel. https://software.intel.com/en-us/blogs/2008/03/27/update-c-states-c-..., https://software.intel.com/en-us/articles/power-management-states-p-...," consultado el 4 de Diciembre de 2016.