Problema 1

Se ha ejecutado uptime 3 veces en momentos diferentes:

```
... load average:  6.85,  7.37, 7.83
... load average:  8.50, 10.93, 8.61
... load average: 37.34,  9.47, 3.30
```

Indica si la carga crece, decrece, se mantiene estacionaria o no puede decidir sobre ello.

Si observamos las cargas, vemos que muestran información sobre la carga en los últimos 1, 5 y 15 minutos, respectivamente. Como no se observa un resultado claro, no podemos decidir sobre ello.

Problema 2

Se ha ejecutado la siguiente orden:

```
$ time quicksort
real 0m40.2s
user 0m17.1s
sys  0m3.2s
```

Indica si el sistema está soportando mucha o poca carga. Razona tu respuesta.

Para saber si el sistema está soportando mucha o poca carga, debemos estudiar la diferencia entre el tiempo del sistema (real) y el tiempo del usuario (user + sys). Como el tiempo del sistema es de 40.2 segundos y el tiempo del usuario es de 20.3 segundos, vemos una diferencia grande entre ambos tiempos. Por tanto, el sistema está soportando mucha carga.

Problema 3

Considera las siguientes órdenes:

```
$ time simulador_original
real 0m24.2s
user 0m15.1s
sys  0m1.6s
```

```
$ time simulador_mejorado
real 0m32.8s
user 0m10.7s
sys  0m2.1s
```

1. Tiempo de ejecución de ambos simuladores

2. Calcula la mejora del tiempo de ejecución del simulador mejorado respecto del original.

1. Para calcular el tiempo de ejecución de los simuladores, debemos calcular el tiempo del usuario (user + sys). En el simulador original, el simulador se ejecuta durante 16.7 segundos. En el simulador mejorado, se ejecuta durante 12.8 segundos.

2. $\frac{(Simulador\ original)}{Simulador\ mejorado} = \frac{16.7}{12.8} = 1.3$

El simulador mejorado tiene una mejora de 1.3 segundos respecto al original.

Problema 4

La sobrecarga de un monitor es del 4 %. Si el monitor se activa cada 2 segundos, ¿tiempo de ejecución por cada activación?

$$Sobrecarga = \frac{(Uso\ del\ recurso)}{Capacidad} \rightarrow Uso\ del\ recurso = Sobrecarga * Capacidad = 0.04 * 2 = 0.08\ segundos$$

Sobrecarga = 4% = 0.04 Capacidad = 2 segundos

El monitor tarda 0.08 segundos en ejecutarse por cada activación.

Problema 5

El monitor sar se activa cada 15 minutos y tarda 750 ms en ejecutarse por activación.

1. Calcular la sobrecarga del monitor

2. Si la información generada por activación ocupa 8192 bytes, ¿número de ficheros históricos si se dispone de 200 MB de memoria libre?

1. Activación = 15 * 60 = 900 segundos Uso del recurso = 750 ms = 0.75 s

$$sobrecarga = \frac{Uso\ del\ recurso}{Activación} = \frac{0.75\ s}{900\ segundos} = 0.083\ \%$$

La sobrecarga del monitor es del 0.083%.

2. Numero de bytes por activación = 8192 bytes

Memoria libre = 200 MB = 200 * 1024^2 bytes

$$Numero\ de\ ficheros\ históricos = \frac{Memoria\ libre}{\frac{Capacidad * Numero\ de\ activaciones}{dia}} = \frac{200 * 1024^2}{8192 * 24 * 4} = 266\ ficheros$$

Se pueden almacenar 266 ficheros históricos.

Problema 6

El 8 de octubre se ha ejecutado la siguiente orden:

```
% ls /var/log/sar
-rw-r--r--     1 root     root     3049952 Oct     6 23:50 sa06
-rw-r--r--     1 root     root     3049952 Oct     7 23:50 sa07
-rw-r--r--     1 root     root     2372320 Oct     8 18:40 sa08
```

¿Cada cuánto tiempo se ejecuta el monitor sar instalado en el sistema? ¿Cuánto ocupa el registro de información almacenada cada vez que se activa el monitor?

Si observamos la columna del tiempo, que indica la hora de ejecución del comando, observamos que el comando sar se ejecuta cada 10 minutos.

$$Capacidad = \frac{Tamaño}{Tiempo} = \frac{3049952}{\frac{24 * 60}{10}} = 20.68\ KB$$

El registro de información almacenada ocupa 20.68 KB por activación del monitor.

Problema 7

Indica el resultado que producen las siguientes ejecuciones:

```
1. sar
2. sar -A
3. sar -u 1 30
4. sar -uB -f /var/log/sa/08
5. sar -d -s 12:30:00 -e 18:15:00 -f /var/log/sa/08
6. sadc
7. sadc 2 4
8. sadc 2 4 fichero
```

1. El comando sar muestra información sobre la utilización del procesador durante el día actual.

2. El comando indicado muestra toda la información recogida sobre la utilización del procesador durante el día actual.

3. El comando indicado muestra información sobre la utilización actual del procesador, tomando 30 medidas por segundo.

4. El comando indicado muestra la utilización del procesador y la paginación de la memoria virtual el día 8 de este mes.

5. El comando indicado muestra las transferencias de disco desde las 12:30 hasta las 18:15 del día 8 de este mes.

6. El comando indicado muestra la información actual de una activación del monitor, en formato binario.

7. El comando indicado muestra, en formato binario, la información recogida al realizar 2 activaciones con un tiempo de separación de 4 segundos.

8. El comando indicado muestra, en formato binario, la información recogida al realizar 2 activaciones con un tiempo de separación de 4 segundos, y guardada en un fichero de disco de nombre <fichero>.

Problema 8

Indica el orden u órdenes para monitorizar las siguientes actividades de un computador:

1. Capacidad de memoria física ocupada por un proceso.
2. Número de cambios de contexto por segundo.
3. Carga media del sistema.
4. Número de interrupciones por segundo.
5. Capacidad libre de la unidad de disco magnético.
6. Usuarios conectados a la máquina.
7. Utilización del procesador en modo usuario.
8. Tiempo que lleva ejecutándose un proceso.
9. Tiempo que tarda un proceso en ejecutarse.

1. Comandos top y ps.
2. Comandos vmstat y sar.
3. Comandos uptime y sar.
4. Comandos vmstat y sar.
5. Comando df.
6. Comando who.
7. Comandos top, vmstat y sar.
8. Comandos top y ps.
9. Comando time.

Problema 9

Se muestra el resultado del comando gprof:

```
Flat profile:
Each sample counts as 0.01 seconds.

  %   cumulative   self              self     total
 time   seconds   seconds    calls  s/call   s/call  name
 59.36    27.72    27.72        3     9.24    14.39  reduce
 33.08    43.17    15.45        6     2.57     2.57  invierte
  7.56    46.70     3.53        2     1.76     1.76  calcula
```

El grafo muestra que el procedimiento invierte() es llamado desde el procedimiento reduce().

1. Tiempo de ejecución del código propio del procedimiento reduce()

2. Procedimiento más lento y más rápido del programa.

3. Si se sustituye el procedimiento más lento por otro 3 veces más rápido, ¿tiempo de ejecución del programa?

4. Si sustituimos el procedimiento invierte() por una versión 4 veces más rápida, ¿mejora en el tiempo de ejecución?

5. Calcula la aceleración máxima del tiempo de ejecución tras la optimización del procedimiento invierte().

1. Esta información la muestra el tiempo medio por llamada, siendo 9.24 segundos.

2. El tiempo de ejecución de los procedimientos los muestra el tiempo medio en segundos, de modo que el procedimiento más lento es el procedimiento reduce() y el más rápido es el procedimiento calcula().

3. El procedimiento más lento es el procedimiento reduce(), con un tiempo medio por llamada de 9.24 segundos.

Calculamos el tiempo de ejecución del procedimiento reduce() tras la sustitución:

$$Tmejorado = \frac{9.24}{3} * 3 = 9.24\, segundos$$

Calculamos el tiempo de ejecución del programa:

$$Tiempo\, de\, ejecución = 9.24 + 15.45 + 3.53 = 28.22\, segundos$$

Tras la sustitución del procedimiento, el programa tarda 28.22 segundos en ejecutarse.

4. Calculamos el nuevo tiempo de ejecución del procedimiento invierte():

$$Tmejorado = \frac{2.57}{4} * 6 = 3.88\, segundos$$

Calculamos el tiempo de ejecución del programa tras la mejora:

$$Ejecución\, programa\, mejorado = 27.72 + 3.86 + 3.53 = 35.11\, segundos$$

Calculamos la mejora en el tiempo de ejecución:

$$Mejora\, programa = \frac{46.70}{35.11} = 1.33\, segundos$$

Se obtendrá una mejora de 1.33 segundos en el tiempo de ejecución del programa.

5.

$$Aceleración\, máxima = \frac{Tiempo\, de\, ejecución\, del\, programa}{Tiempo\, ejecución\, reduce() + Tiempo\, ejecución\, calcula()} = \frac{46.70}{31.25} = 1.49$$

La aceleración máxima que se podría conseguir es de 1.49

Problema 10

Considera las siguiente secuencia de órdenes:

```
% sadc resultado
% ls -l resultado
  -rw-r--r--  1 usuario grupo 712 2006-01-16 10:55 resultado
```

1. Indica qué contiene el fichero resultado y cómo se codifica la información

2. Orden a implementar para visualizar la información en formato ASCII

3. Si se ejecuta cada 3 minutos y disponemos de 50 MB de espacio de almacenamiento en el disco duro, ¿cuántos ficheros históricos diarios podemos mantener?

1. La primera sentencia de órdenes nos indica que el fichero resultado contiene toda la información de la actividad del sistema, codificada en formato binario.

2. Debemos utilizar la orden: sar -A -f resultado

3. Tiempo de ejecución = 3 minutos

Espacio libre = 50 MB = 50 * 1024^2 B

Capacidad = 712 B

Calculamos el número de ficheros históricos:

$$Número\ de\ ficheros\ históricos = \frac{Espacio\ libre}{\frac{Capacidad * Numero\ de\ activaciones}{dia}} = \frac{50 * 1024^2}{\frac{712 * 24 * 60}{3}} = 153\ ficheros$$

Podemos mantener 153 ficheros históricos diariamente.

Problema 11

Se muestra la siguiente información mediante la herramienta gprof:

Flat profile:

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
xxxxx	xxxxx	15.47	3	5.16	5.16	colorea
xxxxx	xxxxx	1.89	5	0.38	0.38	interpola
xxxxx	xxxxx	1.76	1	1.76	3.65	traza
xxxxx	xxxxx	0.46				main

Call graph:

index	% time	self	children	called	name	
[1]	100.0	0.46	19.12		main	[1]
		15.47	0.00	3/3	colorea	[2]
		1.76	1.89	1/1	traza	[3]

		15.47	0.00	3/3	main	[1]
[2]	79.0	15.47	0.00	3	colorea	[2]

		1.76	1.89	1/1	main	[1]
[3]	18.6	1.76	1.89	1	traza	[3]
		1.89	0.00	5/5	interpola	[4]

		1.89	0.00	5/5	traza	[3]
[4]	9.7	1.89	0.00	5	interpola	[4]

1. Tiempo de ejecución del programa de dibujo.

2. Tiempo de ejecución del código propio de main().

3. Relación de llamadas entre los procedimientos del programa y el número de veces de ejecución de cada uno de ellos.

4. Tiempo de ejecución del programa si eliminamos el código propio de main() y reducimos a la mitad el tiempo de ejecución del código propio del procedimiento traza().

5. Proponga y justifique numéricamente una acción sobre el programa original que no afecte al procedimiento colorea(), con el fin de conseguir que el programa se ejecute en 10 segundos.

1. Para obtener el tiempo de ejecución del programa, debemos obtener el tiempo total de todos los procedimientos del programa.

$$Tiempo\,de\,ejecución\,del\,programa = 15.47 + 1.89 + 1.76 + 0.46 = 19.58\,segundos$$

El programa de dibujo se ejecuta en 19.58 segundos

2. El tiempo propio de ejecución del main() viene dado por el tiempo medio en segundos, siendo dicho tiempo de 0.46 segundos.

3. El proceso main() llama 3 veces al proceso colorea(),1 vez al proceso traza() y 5 veces al proceso interpola()

4. Calculamos el nuevo tiempo de ejecución:

$$Tiempo\,de\,ejecución = 15.47 + 1.89 + \frac{1.76}{2} = 18.24\,segundos$$

El nuevo tiempo de ejecución del programa con las condiciones dadas es de 18.24 segundos.

5. Tiempo de ejecución del programa deseado es de 10 segundos

Como el tiempo de contribución del procedimiento colorea() es de 15.47 segundos, superando el tiempo de ejecución del programa deseado, no se puede conseguir la reducción deseada sin afectar al procedimiento colorea()

Problema 12

Tras monitorizar el sistema con la orden vmstat 1 5, se obtienen los siguientes resultados:

procs		memory				swap		io		system		cpu				
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	
0	0	8	14916	92292	833828	0	0	0	3	0	7	3	1	96	0	
1	0	8	14916	92292	833828	0	0	0	0	1022	40	100	0	0	0	
3	0	8	14916	92292	833828	2	1	16	3	1016	34	99	1	0	0	
1	0	8	14916	92292	833828	0	4	0	8	1035	36	98	2	0	0	
2	0	8	14916	92292	833828	1	5	4	28	1035	36	99	1	0	0	

Indica si los resultados obtenidos son correctos o no. Justifica tu respuesta

Si observamos la columna swap, podemos comprobar la existencia de intercambio con el disco magnético. Por tanto, los resultados obtenidos no son correctos.

Problema 13.

Se muestra la siguiente información como resultado de la monitorización de una aplicación:

Flat profile:

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
xxxxx	xxxxx	30.16	52	0.58	0.58	nimbo
xxxxx	xxxxx	5.13	2	2.56	2.56	borrasca
xxxxx	xxxxx	3.51	2	1.75	1.75	lluvia
xxxxx	xxxxx	1.76	1	1.76	34.17	nube

1. Tiempo de ejecución del programa.

2. Porcentaje del tiempo de ejecución del procedimiento lluvia()

3. Procedimiento más lento del programa.

4. Tiempo de ejecución del código propio del procedimiento borrasca()

5. Tiempo de ejecución del programa si mejoramos 3 veces el procedimiento nimbo().

6. Proponga y justifique numéricamente alguna forma de reducir el tiempo de ejecución original del programa hasta los 20 segundos.

1. Calculamos el tiempo de ejecución total:

$$Tiempo\,de\,ejecución\,del\,programa = 30.16 + 5.13 + 3.51 + 1.76 = 40.56\,segundos$$

El programa tarda 40.56 segundos en ejecutarse.

2. Calculamos el porcentaje del tiempo de ejecución del procedimiento lluvia():

$$Porcentaje\,ejecución = \frac{Uso\,del\,recurso}{Capacidad} * 100 = \frac{3.51}{40.56} * 100 = 8.65\,\%$$

El procedimiento lluvia() tarda 8.65% del tiempo de ejecución del programa.

3. El procedimiento más lento es nube(), con un tiempo total de llamada de 34.17 segundos

4. El código propio del procedimiento borrasca() tarda 2.56 segundos en ejecutarse.

5. Se mejora el procedimiento nimbo() 3 veces.

Calculamos el nuevo tiempo de ejecución:

$$Tiempo\,de\,ejecución = \frac{0.58}{3} * 52 + 5.13 + 3.51 + 1.76 = 20.45\,segundos$$

El tiempo de ejecución tras la mejora es de 20.45 segundos.

6.Se desea un tiempo de ejecución de 20 segundos.

Si no realizamos ninguna mejora, el tiempo de ejecución de los procedimientos borrasca(), nuebe() y lluvia() es de 10.4 segundos, quedando un tiempo de ejecución máximo del procedimiento nimbo() de 9.6 segundos. Por tanto, el tiempo propio máximo del procedimiento nimbo() es de 9.58%52 = 0.18 segundos. Para obtener

este tiempo, debemos mejorar el procedimiento nimbo() 0.58%0.18=4 veces. De este modo, el tiempo total del programa se reduce hasta 17.94 segundos.

Problema 14

Tras la ejecución de 2 órdenes, se muestra la siguiente información:

```
% uptime
  9:50am up 173 days, 23:02, 1 user, load average: 0.00, 0.00, 0.00

% time simulador
real 8m0.70s
user 3m5.20s
sys  0m4.01s
```

1. Condición de carga del computador al conectarse el usuario

2. Tiempo en segundos de ejecución del programa simulador

3. ¿Se encuentra alguna incoherencia en los resultados anteriores? Justifica tu respuesta.

1. Si observamos la información dada por la orden uptime, observamos que la carga del computador en los últimos 1,5 y 15 minutos son 0. Por tanto, el computador está sometido a una baja carga en el momento de conexión del usuario.

2. Calculamos los tiempos de ejecución del simulador:

$$Tiempo\,de\,ejecución\,del\,computador = Tiempo\,real = 8*60+0.7 = 480.7\ segundos$$

$$Tiempo\,de\,ejecución\,del\,usuario = Tiempo\,user + Tiempo\,sys = 3*60+5.20+4.01 = 189.21\ segundos$$

3. Si observamos los resultados, observamos que existe una incoherencia entre el tiempo de carga del sistema, obtenido por la orden uptime, y el tiempo de espera de ejecución del simulador, de 480.7-189.21 = 291.49 segundos.