

Relacion-3-Resuelta.pdf



LosCocos



Ingeniería de Servidores



3º Grado en Ingeniería Informática



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Ya puedes imprimir desde Wuolah

Tus apuntes sin publi y al mejor precio

1 Añadir a la cesta

2 Cola de impresión

3 Impresión

4 Copistería Lowcost

Te enviamos los apuntes a casa

Recogelos en tu copistería más cercana



RELACIÓN EJERCICIOS T3

① Uptime → 3 veces

... load average: 6.05, 7.37, 7.03
... load average: 8.50, 10.93, 8.61
... load average: 87.34, 9.77, 3.80

No se puede decidir sobre la evolución porque no hay una tendencia clara en los valores de las medidas.

② Time quicksort
real 0m40.2s
user 0m17.1s
sys 0m3.2s

Parece que se trata de un sistema con una carga considerable, ya que hay una notable diferencia entre el tiempo de respuesta que experimenta el usuario (parámetro real) y el tiempo que realmente tarda en ejecutarse su programa (user+sys). También podría ser que el proceso quicksort usara mucha E/S y estuviera buena parte del tiempo bloqueado por E/S (I/O blocked)

③ * Sobrecarga CPU = 4%
t = 2s
tº en ejecutarse

$$\text{Sobrecarga CPU} = \frac{\text{Uso recurso por monitor}}{\text{Capacidad recurso}} \times 100$$

$$4 = \frac{\text{Uso}}{2} \times 100 ; \text{Uso} = 0.08 \text{ s} \approx 80 \text{ ms}$$

→ El monitor tarda en ejecutarse 80 ms por activación

- ④ *
- a) Memoria física de la máquina → Mem: 257124 KiB av
 - b) % mem usada ; $\frac{253052 \text{ KiB}}{257124 \text{ KiB}} = 0.984$; 98.4% usado
 - c) Sumamos todos los CPU menos idle
 $23.8 + 14 + 17 = 54.8\%$
 - d) load average: 0.15, 0.27, 0.32
5min 10min 15min

La carga es descendiente.

WUOLAH

Scanned by CamScanner

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

e) Comparaciones stat basadas en la letra N = vice (baja prioridad)
Hay un total de 2 procesos en baja prioridad

g) Fila top, columna RSS (memoria física ocupada) = 824 KB

5

1. Simulador-original

$$T_{ej} = t_{sys} + t_{usr} = 0m 15.1s + 0m 1.6s = \underline{16.7s}$$

Simulador-mejorado

$$T_{ej} = t_{sys} + t_{usr} = 0m 10.7s + 0m 2.1s = \underline{12.8s}$$

$$2. \frac{16.7}{12.8} = \underline{1.304}; \underline{30.4\% \text{ más rápido}}$$

6

Se activa cada 1s una y tarda 750 ms en ejecutarse por activación

$$\bullet \text{ Sobrecarga CPU} = \frac{750 \cdot 10^{-3}}{15 \times 60} \times 100 = \underline{0.083\%}$$

• 8192 B

- Cada activación = 8 KB

Var / log / sa = 200 MB libres

- Se activa cada 1s una

¿cuántos se activan?

$$\frac{24h \times \frac{60min}{1h}}{1s/min} = \underline{96 \text{ activaciones/día}}$$

- Se utiliza un fichero de datos por día, por tanto

$$96 \text{ act/día} \times 8 \text{ KB/act} = \underline{768 \text{ KB/fichero}}$$

- ¿cuántos ficheros se activan?

$$\frac{200 \text{ MB} \times \frac{1024 \text{ KB}}{1 \text{ MB}}}{768 \text{ KB/fichero}} = \underline{266 \text{ ficheros}}$$

(2)

(7)

La última activación de un día fueron las 23:50. Si empezara a las 00:00, el intervalo de activación son 10 minutos.

Los ficheros ~~pas~~ del día anterior ocupan un total de 3049952 B

$$\frac{24h \times \frac{60min}{1h}}{10min} = 144 \text{ activaciones al día (ficheros)}$$

$$\frac{3049952 \text{ B}}{144 \text{ act/día}} = 21180,2 \text{ B} \times \frac{1 \text{ KB}}{1024 \text{ B}} = 20,68 \text{ KB/fichero} \approx \underline{21 \text{ KB/fichero}}$$

(8)

1. Sar : Utilización del procesador durante el día actual
2. Sar - A: Toda la información disponible sobre el día actual
3. Sar - U 1 30:
Utilización del procesador: 30 muestras en un periodo de 1 seg.
4. Sar - UB - g /var/bg/sa/os
Utilización del procesador y paginación de la memoria virtual del día 8 del mes
5. Sar - d - s 12:30:00 - e 18:15:00 - j /var/bg/sa/os.
Transferencias de disco desde las 12:30 hasta las 18:15 del día 8 del mes.

(9)

- a) $\text{self/calls} = 9,24 \text{ s}$
- b) El más lento es reduce() (9'24s) y el más rápido atcald() (1,76s)
- c) $\text{'self seconds de reduce} = 27'72 \text{ s} / 3 = 9'24 \text{ s}$

$$E_{\text{TOTAL}} = 9'24 \text{ s} + 15'45 \text{ s} + 3'53 \text{ s} = 28'22 \text{ s}$$

$$d) \frac{15'45}{4} = 3'86 \rightarrow 3'53 + 3'86 + 27'72 = 35'41$$

$$\frac{46'7}{35'41} = 1'33 \text{ veces más rápido (33 \%)}$$

e) S_{max} invierte()

$$S_{max} = \frac{1}{1-\beta} \quad \beta = 33'08/100 = 0'3308$$

$$S_{max} = \lim_{K \rightarrow \infty} S = \frac{1}{1-\beta} = \frac{1}{1-0'3308} = \underline{\underline{1'49}}$$

10

Los resultados serían incorrectos porque, entre otras cosas, el SO preserva actividad de intercambio con el disco y algunos de los programas del benchmark deberían provocarlos.

11

1. $T_{ej} = 15'47 + 1'89 + 1'76 + 0'46 = 19'58s$

2. Tarda 0'46s en ejecutarse

3. $Main()$ llama 3 veces a $colorea(3/3)$ y una vez a $traza(1/1)$.
A su vez, una vez llamado a $traza()$, este proceso llama 5 veces a $interpola()$ (5/5).

4. $t_{ej: traza} = \frac{1'76}{2} = 0'88s$ y se elimina $main()$

• $T_{ej_{reduc}} = 0'88 + 1'89 + 15'47 = \underline{\underline{18'24s}}$

5. No se puede reducir el tiempo a 10s ~~por~~ sin afectar al proceso $colorea()$, ya que este proceso ya supera los 10 segundos de ejecución.

Ya puedes imprimir desde Wuolah

Tus apuntes sin publi y al mejor precio

12

3

1. $T_{ej} = 30'16 + 5'13 + 3'51 + 1'76 = 40'56s$

2. $Self\ seconds_{nueva} = 3'51$

$\frac{46'56}{40'56} \frac{3'51}{40'56} = 0'0865 \equiv 8,7\% \text{ del } t^o \text{ total de ejecución}$

3. Mirando la columna total s/call vemos que nube es el proceso más lento

4. $Self\ s/call\ de\ bomba() = 2'56s$

5. $Self\ s/call\ de\ nuevo() = 0'58/3 = 0'193$

$0'193\ s/call \times 52\ calls = 10'05s$

$T_{ej\ nuevo} = 10'05 + 5'13 + 3'51 + 1'76 = 20'45s$

6. Por ejemplo: mejorar en 4 el código de nuevo() = $0'58/4 = 0'145s/call$

$0'145\ s/call \times 52\ calls = 7'54s$

$T_{ej''} = 7'54 + 5'13 + 3'51 + 1'76 = 17'94s$

13

1. Load average: 0.00, 0.00, 0.00

La condición de carga es baja \Rightarrow

2. $T_{ej} = user + sys = 3m 5.20s + 0m 4.01s = 189,21s$

3. Los resultados son incoherentes porque la carga es baja (es igual a cero), y esto se contradice con el hecho de que el SO experimente un tiempo de espera de real - (user + sys) = 291,49s

$Tespera = 480'2 - 189,21 = 291,49s$

1 Añadir a la cesta

2 Cola de impresión

3 Impresión

4 Copistería Lowcost

Te enviamos los apuntes a casa

Recogelos en tu copistería más cercana



WUOLAH

Scanned by CamScanner

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

14

450 instrucciones máquina

1024 B / fichero sado

velocidad = 75 MIPS

$$\frac{450 \text{ instr}}{75 \cdot 10^6 \text{ instr/s}} = 6 \cdot 10^{-6} \text{ s}$$

a) Sobrecarga = 5%

Capacidad recurso? $S = \frac{6 \cdot 10^{-3} \text{ ms}}{\text{Capacidad}} \cdot 100$; Capacidad = 12 ms

b) 10 minutos

$$\frac{24 \text{ h} \times \frac{60 \text{ min}}{1 \text{ h}}}{10 \text{ min}} = 144 \text{ activaciones / día}$$

$$1024 \text{ B / activación} \times 144 \text{ activ / día} = \underline{147456 \text{ B / día}} \left(\begin{smallmatrix} \approx 1 \text{ fichero} \\ \text{al día} \end{smallmatrix} \right)$$

15

a) Comprobamos que cada muestra constituye 0.01 segundos.
Buscamos el tº de ejecución del programa, último valor de
cumulative seconds y calculamos:

$$\frac{116'66 \text{ s}}{0.01 \text{ s / muestra}} = 1166.6 \approx \underline{1166 \text{ muestras}}$$

b) El programa más rápido es escala (sel 8 ms/call = 369.41)

Lo mejoramos tres veces: $\frac{369.41 \text{ ms}}{3} = 123.137 \text{ ms}$

$$123.137 \text{ ms} \times 34 \text{ calls} = 4186.65 \text{ ms/call} = 4'19 \text{ s/call}$$

$$T_{ej}' = 4'19 \text{ s} + 33'78 + 70'32 = \underline{108'29 \text{ s}}$$

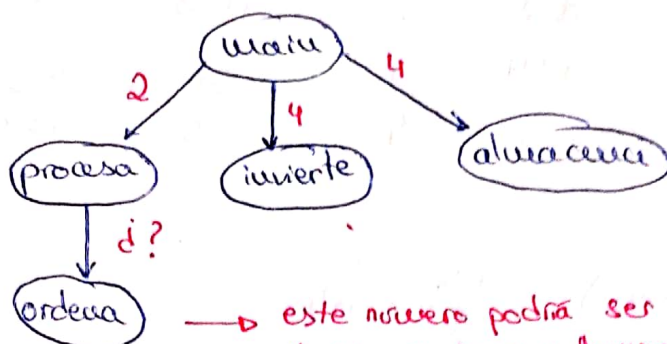
c)

ordena() = $\frac{7032 \text{ ms}}{2} = 3516 \text{ ms} \times 10 = 35.16 \text{ s/call}$	} + = 46'12s
inicio() = $\frac{4222'5 \text{ ms}}{4} = 1055.625 \text{ ms} \times 8 \text{ calls} = 8'45 \text{ s/call}$	
escala() = $\frac{369.41 \text{ ms}}{5} = 73'882 \text{ ms} \times 34 \text{ calls} = 2'51 \text{ s/call}$	

$$\frac{116'66 \text{ s}}{46'12 \text{ s}} = \underline{2'53 \text{ veces más rápido}}$$

(16)

Todos los procesos son llamados desde main, excepto ordena, que es llamado desde procesa.



→ este número podría ser un entero.
No siempre se va a llamar las mismas veces.

a)

$$\text{Self seconds} = \text{calls} \times \text{self us / call}$$

$$\text{calls} = \frac{1800 \text{ us}}{225 \text{ us / call}} = 8 \text{ llamadas}$$

$$\text{Self seconds (2º)} = 1'8 \text{ s } (2 \times 900)$$

$$\text{Self seconds (4º)} = 0'7 \text{ s } (4 \times 175)$$

Cummulative seconds ⇒ Suma progresiva de self seconds

$$1'8 + 1'8 = 3'6 + 1'4 = 5'0 + 0'7 = 5'7 \text{ s}$$

↑
Tº cu del programa.

% time → % self seconds en el Tº cu del programa.

$$(1º) \frac{1'8}{5'7} = 0'316 \approx 31'6\% \quad (2º) 31'6\%$$

$$(3º) \frac{1'4}{5'7} = 0'246 \approx 24'6\% \quad (4º) \frac{0'7}{5'7} = 0'123 \approx 12'3\%$$

$$\text{Total us / call} = T^\circ \text{ ej} + T^\circ \text{ subllamadas}$$

$$\text{Total call (procesa)} = T^\circ \text{ procesa} + T^\circ \text{ ordena}$$

Como procesa() tiene 2 llamadas y se llama a ordena() 8 veces, en cada llamada a procesa() se hacen 4 llamadas a ordena()

$$\text{Total call (procesa)} = 900 + 4 \times 225 = 1800 \text{ us}$$

$$\text{Total call (almacena)} = 175 \text{ us}$$

time %	cumulative seconds	self seconds	calls	self us/call	total us/call	name
31'6	1'8	1'8	8	225	225	ordena
31'6	3'6	1'8	2	900	1800	procesa
24'6	5'0	1'4	4	350	350	inverte
42'3	5'7	0'7	4	175	175	almacena

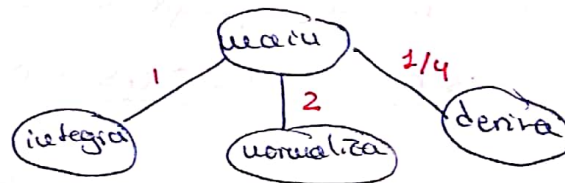
b) self seconds (ordena) = $\frac{1'8}{3} = 0'6$

$T_{ej} = 0'6 + 1'8 + 1'4 + 0'7 = 4'5$

$S = \frac{5'7}{4'5} = 1'27$ (27% + rápido)

17

Cumulative seconds	self seconds	calls	self S/call	total S/call	name
5	5		0'18		deriva
			0'5	8'5	redondea
					integra
					normaliza



Ya puedes imprimir desde Wuolah

Tus apuntes sin publi y al mejor precio

Ingeniería de Servidores

1 Añadir a la cesta

2 Cola de impresión

3 Impresión

4 Copistería Lowcost

Te enviamos los apuntes a casa

Recogelos en tu copistería más cercana



TEMA 3

Monitorización

PROBLEMA 3.1 En un sistema Linux se ha ejecutado la orden `uptime` tres veces en momentos diferentes. El resultado, de forma resumida, es el siguiente:

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

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

SOLUCIÓN: No se puede decidir sobre la evolución porque no hay una tendencia clara en los valores de las medidas.

PROBLEMA 3.2 En un sistema Linux se ha ejecutado la siguiente orden:

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

Indique si el sistema está soportando mucha o poca carga. Razone la respuesta.

SOLUCIÓN: Parece que se trata de un sistema con una carga considerable, ya que hay una notable diferencia entre el tiempo de respuesta que experimenta el usuario (parámetro real) y el tiempo que realmente tarda en ejecutarse su programa (suma de los parámetros user y sys). También podría ser que el proceso quicksort usara mucha E/S y que estuviera buena parte del tiempo bloqueado por E/S (I/O blocked).

PROBLEMA 3.3 Se sabe que la sobrecarga (*overhead*) de un monitor software sobre un computador es del 4 %. Si el monitor se activa cada 2 segundos, ¿cuánto tiempo tarda el monitor en ejecutarse por cada activación?

SOLUCIÓN: El monitor tarda en ejecutarse 80 milisegundos por activación.

PROBLEMA 3.4 A continuación se muestra el resultado obtenido tras ejecutar la orden top en un sistema informático que emplea Linux como sistema operativo:

```
2:52pm up 17 days, 3:41, 1 user, load average: 0.15, 0.27, 0.32
54 processes: 51 sleeping, 3 running, 0 zombie, 0 stopped
CPU states: 23.8% user, 14.0% system, 17.0% nice, 45.2% idle
Mem: 257124K av, 253052K used, 4072K free, 8960K shrd, 182972K buff
Swap: 261496K av, 21396K used, 240100K free, 26344K cached
```

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	LC	%CPU	%MEM	TIME	COMMAND
807	joan	18	2	5708	5708	532	R N	0	23.0	2.2	6:16	p_exec
809	joan	14	2	5708	5708	532	R N	0	14.0	2.2	3:42	p_exec
185	tomi	1	0	824	824	632	R	0	0.5	0.3	0:00	top
201	xp	1	0	1272	1208	644	S	0	0.1	0.4	5:49	xp_stat
1	root	0	0	60	56	36	S	0	0.0	0.0	0:03	init
2	root	0	0	0	0	0	SW	0	0.0	0.0	0:13	kflushd
7	root	0	0	0	0	0	SW	0	0.0	0.0	0:00	nfsiod
194	root	0	0	72	4	4	S	0	0.0	0.0	0:00	migetty
195	root	0	0	68	0	0	SW	0	0.0	0.0	0:00	migetty
179	root	0	0	532	312	236	S	0	0.0	0.1	0:00	sndmail

- ¿Cuánta memoria física tiene la máquina?
- ¿Qué porcentaje de la memoria física está marcada como usada según el monitor?
- ¿Cuál es la utilización media del procesador?
- ¿Cómo es la evolución de la carga media del sistema, ascendente o descendente?
- ¿Hay algún proceso ejecutándose en baja prioridad?
- ¿Cuánta memoria física ocupa el monitor?

SOLUCIÓN:

- 257124KiB.
- 98.4%.
- 54.8%.
- Descendente.
- Dos procesos.
- 824KiB.

■

PROBLEMA 3.5 Considere las órdenes siguientes ejecutadas en un sistema Linux:

\$ time simulador_original	\$ time simulador_mejorado
real 0m24.2s	real 0m32.8s
user 0m15.1s	user 0m10.7s
sys 0m1.6s	sys 0m2.1s

- ¿Cuál es el tiempo de ejecución de ambos simuladores?
- Calcule, si es el caso, la mejora en el tiempo de ejecución del simulador mejorado respecto del original.

SOLUCIÓN: El simulador original se ejecuta en 16,7 segundos y el mejorado en 12,8 segundos (en realidad, los datos muestran que al sistema le cuesta más tiempo porque ha de atender otras tareas). El simulador mejorado es 1,3 veces más rápido que el original.

■

PROBLEMA 3.6 El monitor sar (*system activity reporter*) de un computador se activa cada 15 minutos y tarda 750 ms en ejecutarse por cada activación. Se pide:

- Calcular la sobrecarga que genera este monitor sobre el sistema informático.
- Si la información generada en cada activación ocupa 8192 bytes, ¿cuántos ficheros históricos del tipo *saDD* se pueden almacenar en el directorio */var/log/sa* si se dispone únicamente de 200 MB de capacidad libre?

SOLUCIÓN: La sobrecarga es del 0,083 % y se pueden almacenar en total 266 ficheros históricos.

PROBLEMA 3.7 El día 8 de octubre se ha ejecutado la siguiente orden en un sistema Linux:

```
% 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 2372184 Oct 8 18:40 sa08
```

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

SOLUCIÓN: El monitor se activa cada 10 minutos (la última activación del día se hace a las 23:50 horas). La información generada en cada activación ocupa aproximadamente 21 KB de capacidad.

PROBLEMA 3.8 Indique el resultado que produce la ejecución de las siguientes órdenes sobre un sistema Linux con el monitor sar instalado:

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`

SOLUCIÓN:

1. Utilización del procesador durante el día actual.
2. Toda la información recogida durante el día actual.
3. Utilización actual del procesador: 30 medidas tomadas con un período de un segundo.
4. Utilización del procesador y paginación de la memoria virtual durante el día 8 del mes.
5. Transferencias de disco desde las 12:30 hasta las 18:15 horas del día 8 del mes.

PROBLEMA 3.9 Después de instrumentar un programa con la herramienta `gprof` el resultado obtenido ha sido el siguiente:

Flat profile:
Each sample counts as 0.01 seconds.

%time	cumulative seconds	self seconds	calls	self s/call	total 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 de dependencias muestra que `invierte()` es llamado desde el procedimiento `reduce()`.

1. ¿Cuánto tarda en ejecutarse el código propio del procedimiento `reduce()`?
2. ¿Cuál es el procedimiento más lento del programa? ¿Y el más rápido?
3. Si el código propio del procedimiento más lento de todos se sustituye por otro tres veces más rápido, ¿cuánto tiempo tardará en ejecutarse el programa?
4. Si el procedimiento `invierte()` se sustituye por una nueva versión cuatro veces más rápida, ¿qué mejora se obtendrá en el tiempo de ejecución?
5. Calcule cuál es la ganancia en velocidad máxima que se podría conseguir en el tiempo de ejecución mediante la optimización del código del procedimiento `invierte()`.

SOLUCIÓN:

1. El código propio de `reduce()` tarda 9,24 s.
2. El procedimiento más lento es `reduce()` y el más rápido es `calcula()`.
3. El programa se ejecutaría en 28,22 s.
4. El programa se ejecutaría 1,33 veces más rápidamente.
5. La máxima ganancia en velocidad que se podría conseguir es 1,49.



PROBLEMA 3.10 Un informático desea evaluar el rendimiento de un computador por medio del benchmark SPEC CPU 2017. Una vez compilados todos los programas del paquete y lanzado su ejecución monitoriza el sistema con la orden `vmstat 1 5`. El resultado de las medidas de este monitor es el siguiente:

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

Indique si, a la vista de los datos anteriores, los resultados obtenidos en la prueba evaluación serán correctos o no. Justifique la respuesta.

SOLUCIÓN: Los resultados serían incorrectos porque, entre otras cosas, el sistema operativo presenta actividad de intercambio con el disco y ninguno de los programas del benchmark debería provocarlos.

Ya puedes imprimir desde Wuolah

Tus apuntes sin publi y al mejor precio

Ingeniería de Servidores

PROBLEMA 3.11 La monitorización de un programa de dibujo en tres dimensiones mediante la herramienta gprof ha proporcionado la siguiente información (por errores en la transmisión hay valores que no están disponibles):

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]

- ¿En cuánto tiempo se ejecuta el programa de dibujo?
- Indique cuánto tiempo tarda en ejecutarse el código propio de main().
- Establezca la relación de llamadas entre los procedimientos del programa así como el número de veces que se ejecuta cada uno de ellos.
- Calcule el nuevo tiempo de ejecución del programa si se elimina el código propio de main() y se reduce a la mitad el tiempo de ejecución del código propio del procedimiento traza().
- Proponga y justifique numéricamente una acción sobre el programa original que no afecte el procedimiento colorea() (ni su código ni el número de veces que es ejecutado) con el fin de conseguir que el programa se ejecute en 10 segundos.

SOLUCIÓN:

- El programa se ejecuta en 19,58 s.
- El código propio de main() se ejecuta en 0,46 s.
- El procedimiento main() llama 3 veces a colorea() y una vez a traza(); a su vez, traza() llama 5 veces a interpola().
- El tiempo de ejecución sería de 18,24 s.
- El tiempo de ejecución no se puede reducir a 10 segundos sin afectar el procedimiento colorea() porque su contribución ya sobrepasa este valor.

PROBLEMA 3.12 El resultado de la monitorización de una aplicación informática dedicada al análisis de modelos atmosféricos se muestra a continuación (nótese que hay información no disponible):

Flat profile:

%	cumulative	self		self	total	
time	seconds	seconds	calls	s/call	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. Indique cuánto tiempo tarda en ejecutarse el programa.
2. Determine el porcentaje del tiempo de ejecución que consume el procedimiento lluvia().
3. ¿Cuál es el procedimiento más lento de todo el programa (incluyendo código propio + código de las subrutinas a las que llama)?
4. ¿Cuánto tiempo tarda en ejecutarse el código propio de borrasca()?
5. Calcule el nuevo tiempo de ejecución del programa si el procedimiento nimbo() se rediseña y mejora 3 veces.
6. Proponga y justifique numéricamente alguna manera de reducir el tiempo de ejecución del programa original hasta los 20 segundos.

SOLUCIÓN:

1. El programa se ejecuta en 40,56 s.
2. El procedimiento lluvia() es responsable del 8,7 % del tiempo de ejecución.
3. El procedimiento más lento del programa es nube().
4. El código propio de borrasca() se ejecuta en 2,56 s.
5. El nuevo tiempo de ejecución sería de 20,45 s.
6. Solución libre. Por ejemplo, mejorando 4 veces el tiempo de ejecución del procedimiento nimbo() el tiempo total del programa se reduce hasta 17,94 s.

■

PROBLEMA 3.13 Después de conectarse a un sistema informático, un usuario ejecuta las dos órdenes siguientes con el resultado que se muestra:

```
% 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. ¿En qué condición de carga se encuentra el computador (baja, media o alta) en el momento de conexión del usuario?
2. ¿Cuál es el tiempo (en segundos) de ejecución del programa simulador?
3. ¿Encuentra alguna incoherencia en los resultados anteriores? Justifique la respuesta con argumentos sólidos.

SOLUCIÓN:

1. El computador está en una situación de baja carga.
 2. El computador ejecuta el simulador en 480,7 segundos, aunque el tiempo de ejecución efectivo del mismo es de 189,21 segundos.
 3. En efecto, el hecho de que la carga sea baja (parámetros de load average a cero) está en contradicción con la espera de 291,49 segundos que experimenta la ejecución del simulador.
-
-

PROBLEMA 3.14 En un servidor con S.O. Linux se tiene instalado un monitor de actividad sar (system activity reporter). Se sabe que cada activación del monitor implica la ejecución de un total de 450 instrucciones máquina y almacena un total de 1024 bytes de información en el fichero /var/log/sa/saDD del día DD correspondiente. Si el procesador del equipo tiene una velocidad de ejecución de 75 MIPS (millions of instructions per second):

- a) ¿Qué valor debe tener el periodo de muestreo (en milisegundos) si se quiere una sobrecarga (overhead) del 5%?
- b) Suponiendo ahora que el monitor se activa una vez cada 10 minutos, ¿cuál será el tamaño máximo de cada fichero del directorio /var/log/sa?

SOLUCIÓN:

- a) 0,12 ms. b) 147456 bytes.
-
-

PROBLEMA 3.15 Después de instrumentar un programa con la herramienta gprof el resultado obtenido ha sido el siguiente:

Flat profile:
Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
60.28	70.32	70.32	10	7032.00	7032.00	ordena
28.96	104.10	33.78	8	4222.50	4222.50	inicio
10.76	116.66	12.56	34	369.41	369.41	escala

- a) ¿Cuántas medidas se han tomado para obtener estos resultados?
- b) Si el procedimiento más rápido de los tres se sustituye por otro tres veces más rápido, ¿cuánto tiempo tardará en ejecutarse el programa?
- c) Si los tres procedimientos *ordena*, *inicio* y *escala* se sustituyen al mismo tiempo por nuevas versiones 2, 4 y 5 veces más rápidas, respectivamente, ¿cuál sería la ganancia en velocidad conseguida con respecto al programa original?

SOLUCIÓN:

- a) 11666 muestras. b) 108,29 segundos. c) 2,53.
-
-

PROBLEMA 3.16 El resultado de la monitorización de la actividad de una aplicación informática que está siendo ejecutada dentro de un servidor dedicado a streaming de vídeo se muestra a continuación (nótese que hay información no disponible). Como información adicional, el grafo de llamadas indica que todos los procedimientos son llamados únicamente desde el programa principal `main` (cuyo tiempo propio de ejecución se puede despreciar), excepto `ordena`, que solo es llamado desde el procedimiento `procesa`.

Flat profile:

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
xx	xx	1,8	xx	225	225	ordena
xx	xx	xx	2	900	xx	procesa
xx	xx	1,4	4	350	350	invierte
xx	xx	xx	4	175	xx	almacena

- Complete la información no disponible en la tabla (marcada como “xx”). ¿Cuánto tiempo de CPU consume la aplicación?
- Determine la ganancia en velocidad (speedup) que se obtendría si reemplazamos el procedimiento `ordena` por otro 3 veces más rápido. Expresese esa ganancia en velocidad también como tanto por ciento de mejora.

SOLUCIÓN:

a)

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
31,6	1,8	1,8	8	225	225	ordena
31,6	3,6	1,8	2	900	1800	procesa
24,6	5,0	1,4	4	350	350	invierte
12,3	5,7	0,7	4	175	175	almacena

La aplicación consume 5,7s de tiempo CPU.

- $S=1,27$ (27% más rápido).

■

PROBLEMA 3.17 La monitorización de un programa de cálculo numérico, escrito en C, mediante la herramienta `gprof` en Linux ha proporcionado la siguiente información (nótese que hay información no disponible y que el tiempo propio de la función `main` puede despreciarse):

Flat profile:

Each sample counts as 0.01 seconds.

cumulative seconds	self seconds	calls	self s/call	total s/call	name
	5				deriva
			0.18		redondea
				8.5	integra
			0.5		normaliza

Ya puedes imprimir desde Wuolah

Tus apuntes sin publi y al mejor precio

Ingeniería de Servidores

Call graph:

index	called	name	
[1]		main	[1]
	1/1	integra	[3]
	2/2	normaliza	[5]
	1/4	deriva	[2]
	1/4	main	[1]
	3/4	integra	[3]
[2]	4	deriva	[2]
	12/25	redondea	[4]
	1/1	main	[1]
[3]	1	integra	[3]
	3/4	deriva	[2]
	3/25	redondea	[4]

index	called	name	
	3/25	integra	[3]
	10/25	normaliza	[5]
	12/25	deriva	[2]
[4]	25	redondea	[4]
	2/2	main	[1]
[5]	2	normaliza	[5]
	10/25	redondea	[4]

- Complete las celdas en blanco de la tabla e indique el razonamiento que ha seguido para ello. ¿Cuánto tarda en ejecutarse el programa (tiempo de CPU)?
- ¿Qué quiere decir que "Each sample counts as 0.01 seconds"? ¿Qué tiene esto que ver con el funcionamiento gprof?
- De tener que optimizar el código propio de una de las funciones, ¿de cuál la haría? Razone la respuesta.

SOLUCIÓN:

a)

cumulative seconds	self seconds	calls	self s/call	total s/call	name
5	5	4	$5/4=1,25$	$1,25+3*0,18=1,79$	deriva
9,5	4,5	25	0,18	0,18	redondea
12,09	2,59	1	$8,5-3*1,79-3*0,18=2,59$	8,5	integra
13,09	1	2	0,5	$0,5+5*0,18=1,4$	normaliza

El programa se ejecuta en 13,09s.

- gprof usa temporizadores de profiling en Linux que interrumpen cada cierto tiempo la ejecución del programa para ver qué parte del código se está ejecutando. Estos temporizadores solo avanzan la cuenta cuando el procesador está ejecutando código del programa. Esa línea nos dice que cada 0,01s de tiempo de CPU real del programa se está interrumpiendo dicha ejecución para muestrear qué parte del código se está ejecutando.
- gprof siempre ordena las funciones por el tiempo de ejecución TOTAL de su código propio. La ejecución del código propio de "deriva" supone un 38% del tiempo total de ejecución del programa, la de "redondea" un 34%, la de "integra" un 20% y la de "normaliza" un 8%. Por lo tanto, es el código propio de "deriva" el que debe optimizarse como primera opción. Aunque, individualmente, "integra" tiene un código propio que tarda más tiempo en ejecutarse que las otras funciones pero al ser llamado solamente una vez en el programa hace que no sea preferente su optimización (antes convendría optimizar "deriva" y, después de ésta, "redondea" que aunque tarda muy poco en ejecutarse, se llama 25 veces a lo largo de todo el programa).



Te enviamos los apuntes a casa

Recógelos en tu copistería más cercana

