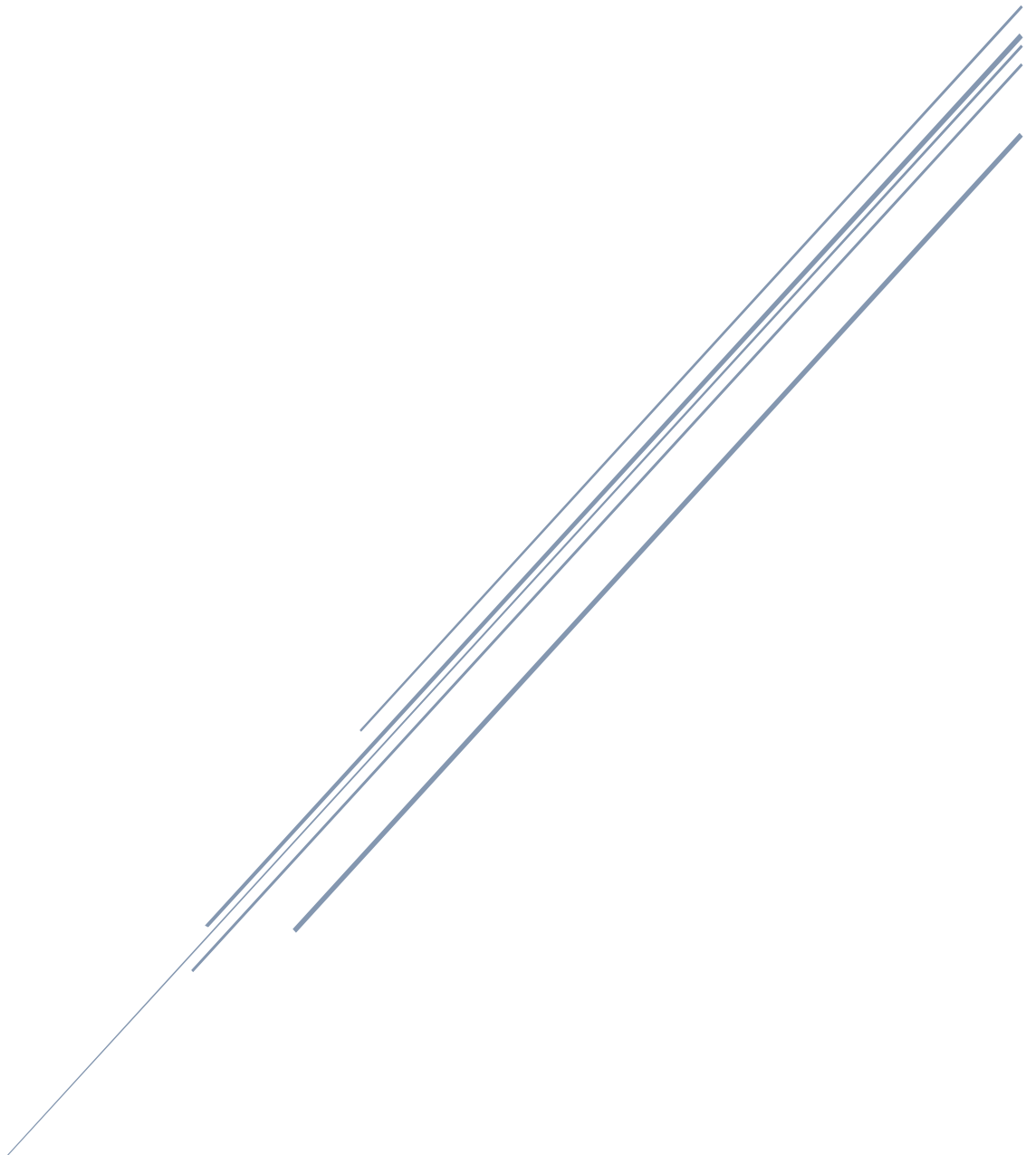


EVALUACIÓN DE CONFIGURACIONES

Práctica 4

—

Ajuste



FDI - UCM

Iván Aguilera Calle — Daniel García Moreno

1. Gestión de procesos

1.1. Parámetros del planificador

Comenzamos averiguando la utilidad de algunos parámetros del planificador CFS, así como su valor actual en el sistema, utilizando para ello la orden `sysctl`:

1. **`sched_latency_ns`**: este parámetro indica el periodo del planificador en el cual intenta planificar las tareas de la *runqueue*. Se intenta planificar todas las tareas de la *runqueue* al menos una vez en este tiempo. El valor es de 6000000 ns.

```
cat /proc/sys/kernel/sched_latency_ns
6000000
```

2. **`sched_min_granularity_ns`**: este parámetro decide el tiempo mínimo en el que una tarea puede estar ejecutándose en la CPU antes de ser expropiada por otra tarea. El valor actual es de 750000 ns.

```
cat /proc/sys/kernel/sched_min_granularity_ns
750000
```

3. **`sched_wakeup_granularity_ns`**: este parámetro indica la capacidad de la tarea que se ha despertado para expropiar a la tarea que se está ejecutando actualmente en la CPU. Cuanto menor sea este valor, se despertarán las tareas cada menos tiempo causando una mayor probabilidad de expropiación, provocando así una mayor perturbación. Esto es, el tiempo que pasa desde que se despierta hasta que se considera candidata para expropiar la CPU. El valor actual es de 1000000 ns.

```
cat /proc/sys/kernel/sched_wakeup_granularity_ns
1000000
```

A continuación, instalaremos el `perf` ejecutando los siguientes comandos:

```
sudo apt-get update
sudo apt-get install linux-tools
```

Posteriormente, ejecutamos en una terminal el programa `matrix1.c` ejecutando el siguiente comando, con el cual estaremos lanzando procesos `matrix1`:

```
while true; do ./matrix1; done
```

En otra pestaña, ejecutamos el programa `perf`, obteniendo los siguientes resultados con los valores de los parámetros de `sysctl` sin modificar:

```
perf stat -r 5 ./matrix1
```

Se ha observado 333 cambios de contexto y un tiempo de ejecución de 2,52 segundos.

A continuación, modificamos el valor del parámetro `sched_min_granularity_ns` a un valor de 10 veces su valor original. Volvemos a ejecutar `perf`.

```
echo 7500000 > /proc/sys/kernel/sched_min_granularity_ns  
perf stat -r 5 ./matrix1
```

En esta ocasión, se observa 291 cambios de contexto y 3,25 segundos de tiempo de ejecución.

Por último, volvemos a modificar el valor del parámetro `sched_min_granularity_ns` a un valor 100 superior al valor original, y seguidamente, ejecutamos `perf`.

```
echo 75000000 > /proc/sys/kernel/sched_min_granularity_ns  
perf stat -r 5 ./matrix1
```

Obtenemos 177 cambios de contexto y un tiempo de ejecución de 2,51 segundos.

A modo de resumen, en la siguiente tabla podemos observar los siguientes valores:

	CAMBIOS DE CONTEXTO	TIEMPO DE EJECUCIÓN
VALOR ORIGINAL	333	2,52
x 10	291	3,25
x 100	177	2,51
x 1000	15	1,84

```
usuario@debian:~/Escritorio/Comandos$ perf stat -r 5 ./matrix1
```

Performance counter stats for './matrix1' (5 runs):

1241,715543 task-clock	#	0,493 CPUs utilized	(+- 3,17%)
333 context-switches	#	0,000 M/sec	(+- 3,93%)
0 CPU-migrations	#	0,000 M/sec	
2.215 page-faults	#	0,002 M/sec	(+- 0,01%)
<not supported> cycles			
<not supported> stalled-cycles-frontend			
<not supported> stalled-cycles-backend			
<not supported> instructions			
<not supported> branches			
<not supported> branch-misses			
2,521031524 seconds time elapsed			(+- 3,30%)

```
usuario@debian:~/Escritorio/Comandos$ sudo su
root@debian:/home/usuario/Escritorio/Comandos# echo 7500000 >
/proc/sys/kernel/sched_min_granularity_ns
```

```
usuario@debian:~/Escritorio/Comandos$ perf stat -r 5 ./matrix1
```

Performance counter stats for './matrix1' (5 runs):

1596,502852 task-clock	#	0,491 CPUs utilized	(+- 4,49%)
291 context-switches	#	0,000 M/sec	(+- 4,63%)
0 CPU-migrations	#	0,000 M/sec	
2.215 page-faults	#	0,001 M/sec	
<not supported> cycles			
<not supported> stalled-cycles-frontend			
<not supported> stalled-cycles-backend			
<not supported> instructions			
<not supported> branches			
<not supported> branch-misses			
3,253820982 seconds time elapsed			(+- 4,30%)

```
root@debian:/home/usuario/Escritorio/Comandos# echo 75000000 >
/proc/sys/kernel/sched_min_granularity_ns
```

```
usuario@debian:~/Escritorio/Comandos$ perf stat -r 5 ./matrix1
```

Performance counter stats for './matrix1' (5 runs):

1238,615079 task-clock	#	0,493 CPUs utilized	(+- 2,50%)
177 context-switches	#	0,000 M/sec	(+- 4,30%)
0 CPU-migrations	#	0,000 M/sec	
2.215 page-faults	#	0,002 M/sec	(+- 0,01%)
<not supported> cycles			
<not supported> stalled-cycles-frontend			
<not supported> stalled-cycles-backend			
<not supported> instructions			
<not supported> branches			
<not supported> branch-misses			
2,514931147 seconds time elapsed			(+- 3,23%)
(+- 4,30%)			

```
root@debian:/home/usuario/Escritorio/Comandos# sysctl -w
kernel.sched_min_granularity_ns=750000
kernel.sched_min_granularity_ns = 750000
```

Podemos apreciar una disminución del número de cambios de contexto a medida que damos un valor más alto al parámetro *sched_min_granularity_ns*. El cambio desde el valor original al valor x10 no es lo suficientemente significativo como para apreciar este comportamiento. Es por ello, que con un valor de x10, el tiempo de ejecución no disminuye. Por otro lado, con un valor de x100 y sobre todo con un valor de x1000 se puede apreciar una disminución bastante significativa del número de cambios de contexto, y por ende, el tiempo de ejecución.

Al modificar el valor de *sched_min_granularity_ns*, lo que estamos haciendo es que el proceso permanezca más tiempo ejecutándose en la CPU, y por tanto, se producen menos cambios de contexto.

1.2.Reparto de la CPU

En esta sección, ejecutaremos el comando `yes` en dos procesos distintos. Uno con un valor de `nice` por defecto y el otro con un valor de `nice` elegido manualmente. Para ello, ejecutamos el siguiente comando:


```
yes > /dev/null/ & nice -4 yes > /dev/null &
```

Utilizamos la orden `top` para conocer el porcentaje de CPU utilizado.

```
usuario@debian:~/Escritorio/Comandos$ yes > /dev/null & nice -4 yes > /dev/null &  
[1] 8541  
[2] 8542
```

```
top - 19:29:03 up 1:25, 5 users, load average: 0,85, 0,46, 0,66  
Tasks: 99 total, 3 running, 96 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 69,1 us, 0,7 sy, 30,2 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st  
KiB Mem: 1027008 total, 958076 used, 68932 free, 35308 buffers  
KiB Swap: 477180 total, 4172 used, 473008 free, 494712 cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
8541	usuario	20	0	5604	584	492	R	68,9	0,1	0:19.75	yes
8542	usuario	24	4	5604	584	492	R	28,6	0,1	0:08.15	yes
3397	usuario	20	0	932m	160m	27m	S	1,3	16,0	0:52.71	chromium
2044	root	20	0	158m	45m	9084	S	0,7	4,6	0:27.91	Xorg



Observamos, que el proceso que tienen el valor de `nice` por defecto (0), al tener una mayor prioridad, tiene un mayor porcentaje de CPU asignado.

Repetiremos el ejercicio utilizando ahora, un valor de `nice` negativo. Para realizar esta acción, necesitamos los permisos necesarios.

```
sudo yes > /dev/null/ & nice --4 yes > /dev/null & & nice --15 yes > /dev/null  
&
```

```
sudo killall yes
sudo su
```

```
root@debian:/home/usuario/Escritorio/Comandos# sudo yes > /dev/null & nice -4 yes > /dev/null & nice
--15 yes > /dev/null &
[1] 8605
[2] 8606
[3] 8607
```

```
Tasks: 105 total, 4 running, 100 sleeping, 1 stopped, 0 zombie
%Cpu(s): 97,7 us, 1,0 sy, 1,3 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 1027008 total, 964876 used, 62132 free, 35760 buffers
KiB Swap: 477180 total, 4164 used, 473016 free, 495580 cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
8607	root	5	-15	5604	580	492	R	92,5	0,1	0:14.74	yes
8608	root	20	0	5604	584	492	R	3,0	0,1	0:00.50	yes
8606	root	24	4	5604	584	492	R	1,7	0,1	0:00.21	yes
3397	usuario	20	0	933m	162m	27m	S	1,0	16,2	1:00.05	chromium

En esta ocasión, podemos observar como el proceso con el valor de nice negativo es el que tiene mayor porcentaje de CPU asignado (mayor prioridad, -20 a 19, cuanto menor es el valor, mas prioritario), seguido del proceso con el valor de nice por defecto y con el valor de nice igual a 4.

2. Gestión de la memoria virtual

2.1. Parámetros de la memoria virtual

Comenzamos este apartado investigando la utilidad de algunos de los parámetros del sistema de memoria virtual y hallamos su valor actual, esta vez utilizando el comando sysctl:

<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/sysctl/vm.txt>

1. **min_free_kbytes:** este parámetro fuerza al sistema a mantener un mínimo de kilobytes libres.

```
sysctl vm.min_free_kbytes
```

2. **dirty_background_ratio:** este parámetro es un porcentaje del total de la memoria libre que tiene “free pages” y “reclaimable pages”, el número de páginas a partir del cual el sistema comienza a limpiar páginas sucias.

```
sysctl vm.dirty_background_ratio
```

3. **dirty_expire_centisecs:** este valor indica cuánto tiempo puede estar los datos en caché antes de que sea necesario escribirlos.

```
sysctl vm.dirty_expire_centisecs
```

4. **swappiness:** se pueden establecer valores entre 0 y 100, y cuanto más bajo sea el valor, se utilizará menos la memoria de intercambio, mientras que si este valor es muy elevado el sistema intentará mantener la memoria RAM lo más libre posible, realizando intercambio.

```
sysctl vm.swappiness
```


3. Gestión de la E/S de disco

3.1. Parámetros del sistema de ficheros

Consultando el manual de mke2fs, se observa que con esta orden podemos crear sistemas de ficheros ext2, ext3 y ext4, usualmente, en una partición del disco, pudiendo elegir valores como el tamaño de bloque, el stride, el tamaño de fragmento, el tamaño de inodo...

Por otro lado, tune2fs está orientado a modificar valores de un sistema de ficheros ya creados.

Consultamos el modo de escritura de los datos ejecutando el siguiente comando:

```
mount -l -t ext4

usuario@debian:~/Escritorio/Comandos$ mount -l -t ext4
/dev/disk/by-uuid/77140bc6-ba14-40f1-ac05-996924bb65bc on / type ext4 (rw,relatime,errors=remount-ro,user_xattr,barrier=1,data=ordered)
```

Vamos observar cuándo se produce la escritura de los datos con diferentes valores del parámetro *dirty_expire_centisecs* ejecutando la siguiente orden:

```
dd if=/dev/zero of=/var/tmp/prueba count=10K conv=notrunc
```

```
usuario@debian:~/Escritorio$ vmstat 1
procs -----memory----- ---swap-- -----io----- -system-- ----cpu----
 r b swpd free buff cache si so bi bo in cs us sy id wa
 0 0  0 295984 26248 454208 0 0 67 97 98 170 43 1 56 0
 0 0  0 295868 26248 454208 0 0 0 4 47 159 0 0 100 0
...
 0 0  0 295908 26272 454216 0 0 0 0 20 55 0 0 100 0
 0 0  0 295908 26272 454216 0 0 0 0 26 59 0 1 99 0
procs -----memory----- ---swap-- -----io----- -system-- ----cpu----
 r b swpd free buff cache si so bi bo in cs us sy id wa
 1 0  0 295908 26272 454216 0 0 0 0 22 48 0 0 100 0
...
 0 0  0 295900 26288 454216 0 0 0 0 29 59 0 0 100 0
 0 0  0 295900 26288 454216 0 0 0 0 16 32 0 0 100 0
 0 0  0 295900 26288 454216 0 0 0 5172 43 66 0 0 100 0
```

Con el valor de *dirty_expire_centisecs* sin modificar (3000), observamos con la ayuda de vmstat, que la escritura se produce, más o menos, 30 segundos más tarde de

ejecutar la orden. Anteriormente, permanece en los buffers del kernel hasta que el timer expira.

```
sudo killall yes
sudo su
```

```
usuario@debian:~/Escritorio$ vmstat 1
procs -----memory----- ---swap-- -----io----- -system-- ----cpu----
 r b  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id  wa
0 0    0 292676 26636 457092  0  0  58  94  90 161 37  1 61  0
0 0    0 292668 26636 457096  0  0  0   0  53 203  1  0 99  0
0 0    0 292668 26636 457096  0  0  0   0  27  75  0  0 100  0
0 0    0 292544 26636 457096  0  0  0   0  41 134  0  1 99  0
0 0    0 292544 26636 457096  0  0  0   0  67 273  0  0 100  0
0 0    0 292544 26636 457096  0  0  0   0  38  84  0  0 100  0
0 0    0 292544 26636 457096  0  0  0   0  14  29  0  0 100  0
0 0    0 292544 26644 457092  0  0  0  28  34  71  0  0 100  0
0 0    0 292544 26644 457096  0  0  0   0  15  31  0  0 100  0
0 0    0 292544 26644 457096  0  0  0  12  33  68  0  0 100  0
0 0    0 292544 26644 457096  0  0  0   0  17  52  0  0 100  0
0 0    0 292544 26644 457096  0  0  0   0  28  61  0  0 100  0
0 0    0 292544 26644 457096  0  0  0   0  18  40  0  0 100  0
0 0    0 292544 26644 457096  0  0  0   0  37 174  0  0 100  0
0 0    0 292544 26644 457096  0  0  0 5120  35  78  0  0 100  0
```

Modificado el valor de *dirty_expire_centisecs* a un tercio de su valor original (1000), observamos con la ayuda de *vmstat*, que la escritura se ha realizado bastante antes que en el ejemplo anterior (más o menos 10 filas posteriores desde la línea donde se ejecutó la orden).

3.2. Parámetros de los discos

Consultando el manual de blockdev, se observa que con esta orden podemos llamar al dispositivo ioctl desde la línea de comandos.

Por otro lado, hdparm nos proporciona una interfaz de líneas de comandos que nos permite ver y ajustar los parámetros del hardware de los discos IDE y SATA.

Ejecutamos hdparm para obtener las características del disco virtual. Para ello ejecutamos la siguiente orden:

```
hdparm -I /dev/sda5
```

```
root@debian:/home/usuario/Escritorio/Comandos# hdparm -I /dev/sda5

/dev/sda5:

ATA device, with non-removable media
    Model Number:    VBOX HARDDISK
    Serial Number:   VBfbb4afb9-f2333f6a
    Firmware Revision: 1.0
Standards:
    Used: ATA/ATAPI-6 published, ANSI INCITS 361-2002
    Supported: 6 5 4
Configuration:
    Logical          max      current
cylinders 16383    16383
heads      16      16
sectors/track 63      63
--
CHS current addressable sectors: 16514064
LBA  user addressable sectors: 20971520
LBA48 user addressable sectors: 20971520
Logical/Physical Sector size: 512 bytes
device size with M = 1024*1024: 10240 MBytes
device size with M = 1000*1000: 10737 MBytes (10 GB)
cache/buffer size = 256 KBytes (type=DualPortCache)
Capabilities:
    LBA, IORDY(cannot be disabled)
    Queue depth: 32
    Standby timer values: spec'd by Vendor, no device specific minimum
    R/W multiple sector transfer: Max = 128      Current = 128
    DMA: mdma0 mdma1 mdma2 udma0 udma1 udma2 udma3 udma4 udma5 *udma6
        Cycle time: min=120ns recommended=120ns
    PIO: pio0 pio1 pio2 pio3 pio4
        Cycle time: no flow control=120ns IORDY flow control=120ns
Commands/features:
    Enabled Supported:
    * Power Management feature set
    * Write cache
    * Look-ahead
    * 48-bit Address feature set
    * Mandatory FLUSH_CACHE
    * FLUSH_CACHE_EXT
    * Gen2 signaling speed (3.0Gb/s)
    * Native Command Queueing (NCQ)
Checksum: correct
```

4. Gestión de la E/S de red

4.1. Parámetros de los interfaces de red

Consultando el manual de ethtool, se observa que con esta orden podemos gestionar los parámetros de nuestra tarjeta de red.

Por lo tanto, ejecutaremos el siguiente comando para obtener la configuración de la interfaz de red virtual:

```
ethtool -k eth0
```

```
@debian:/home/usuario/Escritorio/Comandos# ethtool -k eth0
```

Features for eth0:

rx-checksumming: off

tx-checksumming: on

tx-checksum-ipv4: off [fixed]

tx-checksum-unnneeded: off [fixed]

tx-checksum-ip-generic: on

tx-checksum-ipv6: off [fixed]

tx-checksum-fcoe-crc: off [fixed]

tx-checksum-sctp: off [fixed]

scatter-gather: on

tx-scatter-gather: on

tx-scatter-gather-fraglist: off [fixed]

tcp-segmentation-offload: on

tx-tcp-segmentation: on

tx-tcp-ecn-segmentation: off [fixed]

tx-tcp6-segmentation: off [fixed]

udp-fragmentation-offload: off [fixed]

generic-segmentation-offload: on

generic-receive-offload root: on

large-receive-offload: off [fixed]

rx-vlan-offload: on

tx-vlan-offload: on [fixed]

ntuple-filters: off [fixed]

receive-hashing: off [fixed]

highdma: off [fixed]

rx-vlan-filter: on [fixed]

vlan-challenged: off [fixed]

tx-lockless: off [fixed]

netns-local: off [fixed]

tx-gso-robust: off [fixed]

tx-fcoe-segmentation: off [fixed]

fcoe-mtu: off [fixed]

tx-nocache-copy: on

loopback: off [fixed]