

Práctica 7. Ajuste del rendimiento

En esta práctica trabajaremos con los grupos de control y revisaremos algunos parámetros de ajuste de Linux. Utilizaremos la **máquina virtual ECO**.

Contenido:

[Grupos de control \(~25 min.\)](#)

[Ajuste del planificador \(~20 min.\)](#)

[Ajuste de la memoria virtual \(~15 min.\)](#)

[Ajuste del sistema de ficheros \(~20 min.\)](#)

[Ajuste del *hardware* \(~10 min.\)](#)

Grupos de control (~25 min.)

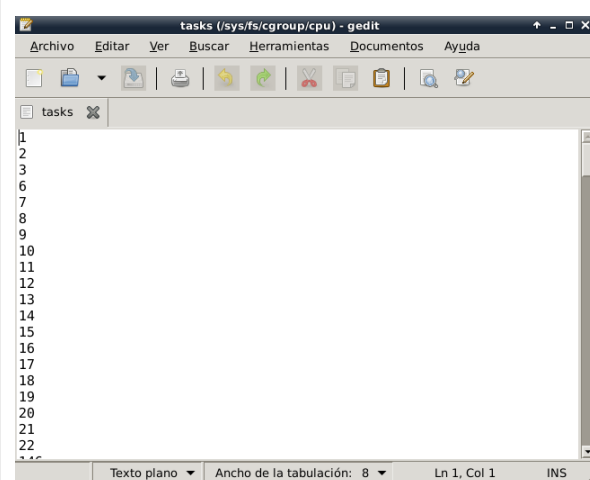
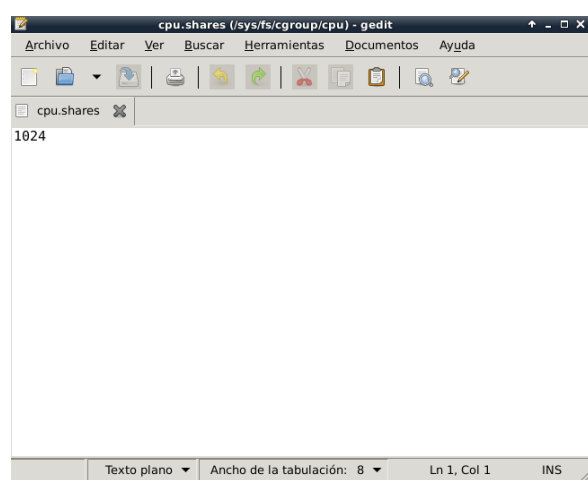
Monta el sistema de ficheros `cgroup` de tipo `tmpfs`, crea el directorio `cpu` y monta un sistema de ficheros de tipo `cgroup` para el subsistema `cpu` con:

```
$ sudo mount -t tmpfs cgroup /sys/fs/cgroup
```

```
$ sudo mkdir /sys/fs/cgroup/cpu
```

```
$ sudo mount -t cgroup -o cpu cpu /sys/fs/cgroup/cpu
```

Accede al directorio `/sys/fs/cgroup/cpu` y observa su contenido. En particular, el de los ficheros `tasks`, que contiene las tareas del grupo, y `cpu.shares`, que contiene la parte relativa (*share*) de procesador del grupo.

<p>tasks: cada número es el PID de una tarea asociada con el cgroup.</p>	<p>cpu.shares: cantidad de recursos compartidos asignados al cgroup</p>
	

Crea un nuevo grupo con:

```
$ sudo mkdir /sys/fs/cgroup/cpu/cg1
```

Accede al directorio `/sys/fs/cgroup/cpu/cg1` y observa su contenido. En particular, el de los ficheros `tasks` y `cpu.shares`.

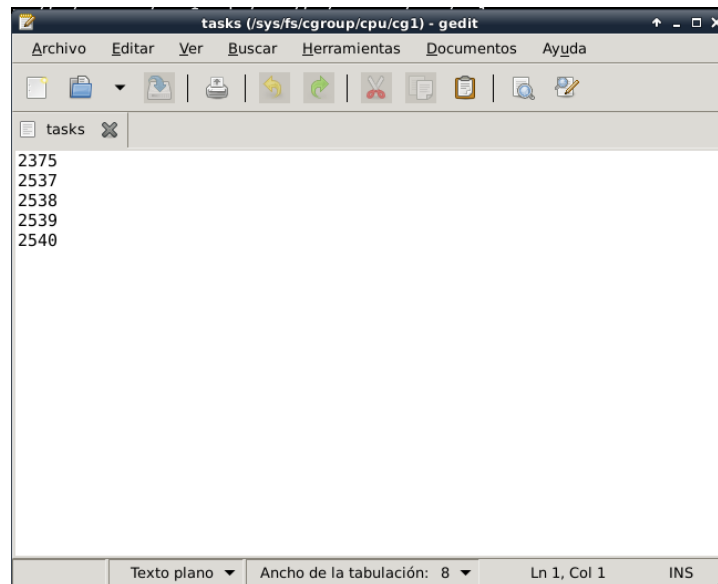
-cpu.shares está igual que antes pero el fichero tasks está vacío.

Para mover la *shell* actual al grupo `cg1`, escribe su PID en el fichero `tasks` con:

```
$ echo $$ | sudo tee tasks
```

Ejecuta una tarea intensiva en procesador (por ejemplo, `yes > /dev/null &`) y observa el contenido del fichero `tasks`.

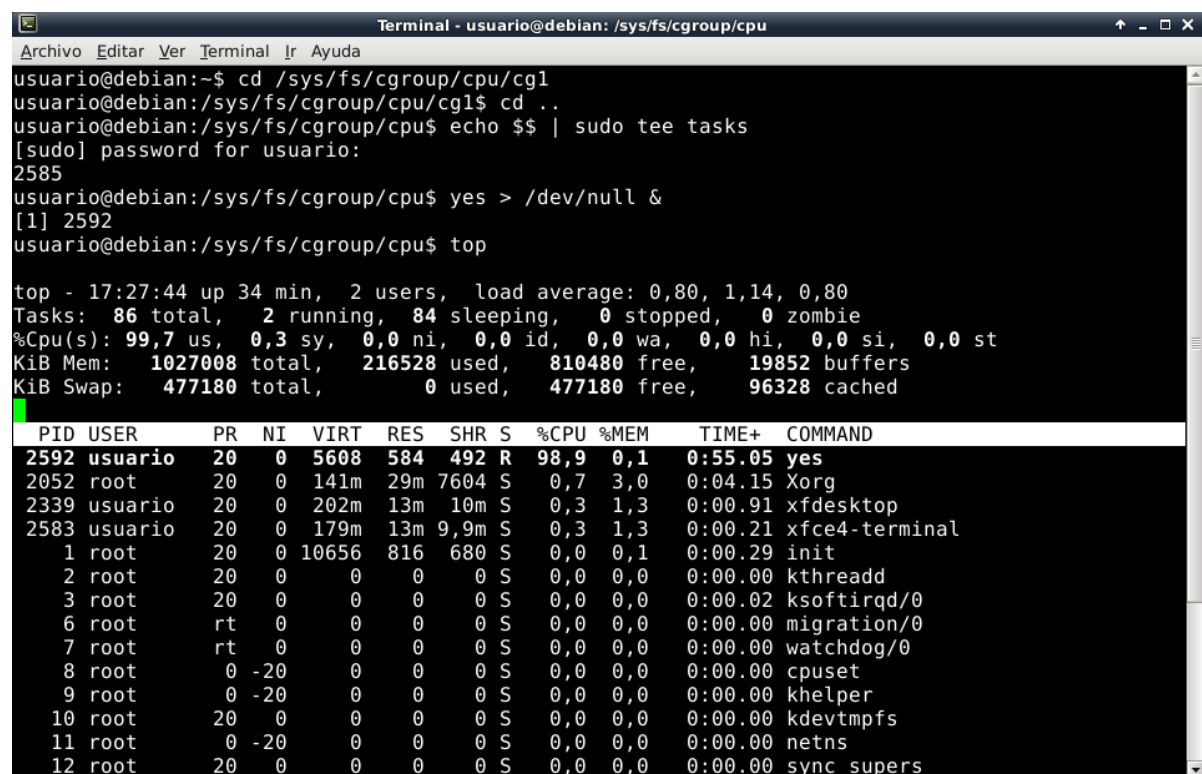
`tasks`: han aparecido la tarea con el pid de la shell y las tareas para realizar el comando introducido.



```
tasks (/sys/fs/cgroup/cpu/cg1) - gedit
Archivo  Editar  Ver  Buscar  Herramientas  Documentos  Ayuda
tasks
2375
2537
2538
2539
2540
Texto plano  Ancho de la tabulación: 8  Ln 1, Col 1  INS
```

Abre otro terminal, cuyo proceso asociado pertenecerá al grupo raíz, y ejecuta otra tarea intensiva en procesador. Observa el resultado con `top`.

Cuando solo hay un proceso del grupo se le asigna toda la cantidad de cpu posible.



```
Terminal - usuario@debian: /sys/fs/cgroup/cpu
Archivo  Editar  Ver  Terminal  Ir  Ayuda
usuario@debian:~$ cd /sys/fs/cgroup/cpu/cg1
usuario@debian:/sys/fs/cgroup/cpu/cg1$ cd ..
usuario@debian:/sys/fs/cgroup/cpu$ echo $$ | sudo tee tasks
[sudo] password for usuario:
2585
usuario@debian:/sys/fs/cgroup/cpu$ yes > /dev/null &
[1] 2592
usuario@debian:/sys/fs/cgroup/cpu$ top

top - 17:27:44 up 34 min,  2 users,  load average: 0,80, 1,14, 0,80
Tasks:  86 total,   2 running,  84 sleeping,   0 stopped,   0 zombie
%Cpu(s): 99,7 us,  0,3 sy,  0,0 ni,  0,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem:  1027008 total,  216528 used,  810480 free,  19852 buffers
KiB Swap:  477180 total,    0 used,  477180 free,  96328 cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 2592 usuario    20   0   5608   584   492  R   98,9   0,1   0:55.05  yes
 2052 root       20   0   141m  29m  7604  S    0,7   3,0   0:04.15  Xorg
 2339 usuario    20   0   202m  13m   10m  S    0,3   1,3   0:00.91  xfdesktop
 2583 usuario    20   0   179m  13m   9,9m  S    0,3   1,3   0:00.21  xfce4-terminal
    1 root       20   0 10656   816   680  S    0,0   0,1   0:00.29  init
    2 root       20   0     0     0     0  S    0,0   0,0   0:00.00  kthreadd
    3 root       20   0     0     0     0  S    0,0   0,0   0:00.02  ksoftirqd/0
    6 root       rt    0     0     0     0  S    0,0   0,0   0:00.00  migration/0
    7 root       rt    0     0     0     0  S    0,0   0,0   0:00.00  watchdog/0
    8 root       0 -20     0     0     0  S    0,0   0,0   0:00.00  cpuset
    9 root       0 -20     0     0     0  S    0,0   0,0   0:00.00  khelper
   10 root       20   0     0     0     0  S    0,0   0,0   0:00.00  kdevtmpfs
   11 root       0 -20     0     0     0  S    0,0   0,0   0:00.00  netns
   12 root       20   0     0     0     0  S    0,0   0,0   0:00.00  sync_supers
```

Al añadir otro proceso en el grupo `cg1` (que tiene la misma `cpu.share` asignada) los procesos tienen la misma prioridad así que se tienen que repartir la `cpu` a partes iguales.

```
Terminal - usuario@debian: /sys/fs/cgroup/cpu/cg1
Archivo Editar Ver Terminal Ir Ayuda
top - 17:35:03 up 41 min, 3 users, load average: 1,53, 1,17, 0,92
Tasks: 87 total, 3 running, 84 sleeping, 0 stopped, 0 zombie
%Cpu(s): 99,3 us, 0,7 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 1027008 total, 218124 used, 808884 free, 19908 buffers
KiB Swap: 477180 total, 0 used, 477180 free, 96408 cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 2605 usuario    20   0   5608   584   492  R   49,9   0,1    0:23.56  yes
 2592 usuario    20   0   5608   584   492  R   49,6   0,1    7:49.20  yes
 2583 usuario    20   0   180m   13m    9m  S    0,7   1,4    0:00.35  xfce4-terminal
 2052 root        20   0   142m   29m   7604  S    0,3   3,0    0:04.34  Xorg
 2339 usuario    20   0   202m   13m   10m  S    0,3   1,3    0:00.95  xfdesktop
    1 root        20   0  10656   816   680  S    0,0   0,1    0:00.29  init
    2 root        20   0     0     0     0  S    0,0   0,0    0:00.00  kthreadd
    3 root        20   0     0     0     0  S    0,0   0,0    0:00.02  ksoftirqd/0
    6 root        rt    0     0     0     0  S    0,0   0,0    0:00.00  migration/0
    7 root        rt    0     0     0     0  S    0,0   0,0    0:00.00  watchdog/0
    8 root         0  -20     0     0     0  S    0,0   0,0    0:00.00  cpuset
    9 root         0  -20     0     0     0  S    0,0   0,0    0:00.00  khelper
   10 root        20   0     0     0     0  S    0,0   0,0    0:00.00  kdevtmpfs
   11 root         0  -20     0     0     0  S    0,0   0,0    0:00.00  netns
   12 root        20   0     0     0     0  S    0,0   0,0    0:00.00  sync_supers
   13 root        20   0     0     0     0  S    0,0   0,0    0:00.00  bdi-default
   14 root         0  -20     0     0     0  S    0,0   0,0    0:00.00  kintegrityd
```

Cambia la parte relativa de procesador del grupo `cg1`, escribiendo en el fichero `cpu.shares` la mitad de su valor actual (usa `echo` y `tee` para escribir) y observa el resultado con `top`. Después, escribe el doble del valor original y observa el resultado.

Copia los resultados y escribe un breve análisis de los mismos.

Para reducir a la mitad el valor de `cpu.shares`: `echo $((1024/2)) | sudo tee cpu.shares`

Al haber reducido a la mitad el valor de `cpu.shares`, todos los procesos de `cg1` tendrán su uso de `cpu` limitado a la mitad con respecto al resto de grupos. Por eso el ejecutar el comando `top` podemos ver como el proceso de grupo `cg1` está usando mucha menos `cpu` que el proceso del grupo raíz.

```
Terminal - usuario@debian: /sys/fs/cgroup/cpu/cg1
Archivo Editar Ver Terminal Ir Ayuda
top - 17:40:40 up 47 min, 3 users, load average: 2,00, 1,73, 1,25
Tasks: 87 total, 3 running, 84 sleeping, 0 stopped, 0 zombie
%Cpu(s): 99,7 us, 0,3 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 1027008 total, 218248 used, 808760 free, 20176 buffers
KiB Swap: 477180 total, 0 used, 477180 free, 96432 cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 2592 usuario    20   0   5608   584   492 R  66,0   0,1   10:45.35 yes
 2605 usuario    20   0   5608   584   492 R  32,9   0,1    3:02.87 yes
 2052 root        20   0   142m   29m  7604 S   0,7   3,0    0:04.53 Xorg
 2339 usuario    20   0   202m   13m   10m S   0,3   1,3    0:00.97 xfdesktop
    1 root        20   0  10656   816   680 S   0,0   0,1    0:00.29 init
    2 root        20   0     0     0     0 S   0,0   0,0    0:00.00 kthreadd
    3 root        20   0     0     0     0 S   0,0   0,0    0:00.02 ksoftirqd/0
    6 root        rt    0     0     0     0 S   0,0   0,0    0:00.00 migration/0
    7 root        rt    0     0     0     0 S   0,0   0,0    0:00.00 watchdog/0
    8 root         0 -20     0     0     0 S   0,0   0,0    0:00.00 cpuset
    9 root         0 -20     0     0     0 S   0,0   0,0    0:00.00 khelper
   10 root        20   0     0     0     0 S   0,0   0,0    0:00.00 kdevtmpfs
   11 root         0 -20     0     0     0 S   0,0   0,0    0:00.00 netns
   12 root        20   0     0     0     0 S   0,0   0,0    0:00.00 sync_supers
   13 root        20   0     0     0     0 S   0,0   0,0    0:00.00 bdi-default
   14 root         0 -20     0     0     0 S   0,0   0,0    0:00.00 kintegrityd
   15 root         0 -20     0     0     0 S   0,0   0,0    0:00.00 kblockd
```

Para duplicar el valor de cpu.shares: `echo $((1024*2)) | sudo tee cpu.shares`

Siguiendo la lógica del caso anterior, al haber duplicado la cantidad de cpu con respecto al resto de grupos, el proceso del grupo `cg1` utiliza el doble de cpu que el del grupo raíz.

```
Terminal - usuario@debian: /sys/fs/cgroup/cpu/cg1
Archivo Editar Ver Terminal Ir Ayuda
top - 17:47:09 up 53 min, 3 users, load average: 2,00, 1,93, 1,51
Tasks: 86 total, 3 running, 83 sleeping, 0 stopped, 0 zombie
%Cpu(s):100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 1027008 total, 218364 used, 808644 free, 20348 buffers
KiB Swap: 477180 total, 0 used, 477180 free, 96432 cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 2605 usuario    20   0   5608   584   492 R  66,9   0,1    5:24.92 yes
 2592 usuario    20   0   5608   584   492 R  33,3   0,1   14:51.06 yes
    1 root        20   0  10656   816   680 S   0,0   0,1    0:00.29 init
    2 root        20   0     0     0     0 S   0,0   0,0    0:00.00 kthreadd
    3 root        20   0     0     0     0 S   0,0   0,0    0:00.02 ksoftirqd/0
    6 root        rt    0     0     0     0 S   0,0   0,0    0:00.00 migration/0
    7 root        rt    0     0     0     0 S   0,0   0,0    0:00.00 watchdog/0
    8 root         0 -20     0     0     0 S   0,0   0,0    0:00.00 cpuset
    9 root         0 -20     0     0     0 S   0,0   0,0    0:00.00 khelper
   10 root        20   0     0     0     0 S   0,0   0,0    0:00.00 kdevtmpfs
   11 root         0 -20     0     0     0 S   0,0   0,0    0:00.00 netns
   12 root        20   0     0     0     0 S   0,0   0,0    0:00.00 sync_supers
   13 root        20   0     0     0     0 S   0,0   0,0    0:00.00 bdi-default
   14 root         0 -20     0     0     0 S   0,0   0,0    0:00.00 kintegrityd
   15 root         0 -20     0     0     0 S   0,0   0,0    0:00.00 kblockd
   17 root        20   0     0     0     0 S   0,0   0,0    0:00.00 khungtaskd
   18 root        20   0     0     0     0 S   0,0   0,0    0:00.20 kswapd0
```

Crea más tareas intensivas en procesador en cada grupo y observa el resultado con `top`. Al terminar, finaliza todas las tareas.

Al crear más tareas, se va repartiendo la cpu entre muchos procesos por lo que se aprecia menos la diferencia, pero aún así se puede ver como los procesos del grupo `cg1` siempre tienen más cpu que los del grupo raíz.

```
Terminal - usuario@debian: /sys/fs/cgroup/cpu/cg1
Archivo Editar Ver Terminal Ir Ayuda
top - 17:50:10 up 56 min, 3 users, load average: 2,94, 2,17, 1,66
Tasks: 91 total, 7 running, 84 sleeping, 0 stopped, 0 zombie
%Cpu(s): 99,0 us, 1,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 1027008 total, 219172 used, 807836 free, 20376 buffers
KiB Swap: 477180 total, 0 used, 477180 free, 96436 cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 2646 usuario    20   0   5608   584   492  R   20,0   0,1    0:04.25  yes
 2647 usuario    20   0   5608   584   492  R   20,0   0,1    0:03.95  yes
 2592 usuario    20   0   5608   584   492  R   19,6   0,1   15:48.54  yes
 2649 usuario    20   0   5608   584   492  R   13,6   0,1    0:01.77  yes
 2605 usuario    20   0   5608   584   492  R   13,3   0,1    7:15.46  yes
 2648 usuario    20   0   5608   580   492  R   13,0   0,1    0:02.56  yes
 2052 root        20   0   142m   29m  7604  S    0,3   3,0    0:04.71  Xorg
    1 root        20   0  10656   816   680  S    0,0   0,1    0:00.29  init
    2 root        20   0      0      0      0  S    0,0   0,0    0:00.00  kthreadd
    3 root        20   0      0      0      0  S    0,0   0,0    0:00.02  ksoftirqd/0
    6 root        rt    0      0      0      0  S    0,0   0,0    0:00.00  migration/0
    7 root        rt    0      0      0      0  S    0,0   0,0    0:00.00  watchdog/0
    8 root         0  -20      0      0      0  S    0,0   0,0    0:00.00  cpuset
    9 root         0  -20      0      0      0  S    0,0   0,0    0:00.00  khelper
   10 root        20   0      0      0      0  S    0,0   0,0    0:00.00  kdevtmpfs
   11 root         0  -20      0      0      0  S    0,0   0,0    0:00.00  netns
   12 root        20   0      0      0      0  S    0,0   0,0    0:00.00  sync supers
```

Opcional:

- Crea más grupos y subgrupos en el subsistema `cpu`.
- Investiga otros subsistemas (el subsistema `memory` no está disponible).
- Prueba los comandos del paquete `cgroup-bin`, que facilitan la gestión de los grupos.

Ajuste del planificador (~20 min.)

Averigua la utilidad de los siguientes parámetros del planificador CFS y su valor actual (accediendo a `/proc/sys/kernel` o con `sysctl`):

- `sched_latency_ns`
- `sched_min_granularity_ns`
- `sched_wakeup_granularity_ns`
- `sched_autogroup_enabled`

Escribe la utilidad y el valor actual de los parámetros.

El **Planificador Completamente Justo** (Completely Fair Scheduler - CFS) es un **algoritmo** planificador desarrollado con la meta de **maximizar el uso de la CPU** con las diferentes tareas que se lanzan en un sistema Linux.

Este algoritmo tiene como objetivo el maximizar el uso de la CPU pero permitiendo el uso interactivo de la máquina. Es decir, tratará de que **en ningún momento** un usuario vea una **bajada de rendimiento**

-sched_latency_ns: parámetro que controla el tiempo máximo que un proceso puede permanecer en la cola de espera antes de que el CFS lo programe para la ejecución. El valor predeterminado es 60000000 ns (60ms) (aunque en nuestro caso pone 6000000 ns(6ms)).

-sched_min_granularity_ns: define la cantidad de tiempo mínima que que el CFS asigna a un proceso en una sola planificación. El **valor predeterminado es 4ms (aunque en nuestro caso pone 0.75 ms)**.

-sched_wakeup_granularity_ns: define la cantidad de tiempo mínima que el CFS espera antes de programar un proceso después de que se haya despertado. El valor predeterminado es de 16ms (en nuestro caso es 1ms).

-sched:autogroup_enabled: sirve para habilitar o deshabilitar la función de autogrupos del CFS. Si está habilitado el CFS agrupo automáticamente procesos y asigna a cada grupo una prioridad en función de la carga de trabajo actual. El valor predeterminado es 1, es decir, habilitado (en nuestro caso es 0).

Instala `perf` con:

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

Mide el tiempo de ejecución y el número de cambios de contexto de una ejecución del programa `matrix1.c` (disponible con la práctica) en paralelo con otra ejecución del programa de la siguiente forma:

```
$ perf stat ./matrix1 & ./matrix1
```

Haz varias mediciones y anota la que proporcione menor tiempo de ejecución.


```

usuario@debian:~/Escritorio/laboratorio$ perf stat ./matrix1 & ./matrix1
[1] 2955
Result: 127840000.000000
usuario@debian:~/Escritorio/laboratorio$ Result: 127840000.000000

Performance counter stats for './matrix1':

      938,761025 task-clock                #    0,499 CPUs utilized
           217 context-switches          #    0,000 M/sec
              0 CPU-migrations            #    0,000 M/sec
          3.880 page-faults                #    0,004 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

      1,879857151 seconds time elapsed

```

Repita las mediciones estableciendo el valor del parámetro `sched_min_granularity_ns` a 10 y 100 veces su valor original. Por ejemplo con:

```
$ sudo sysctl kernel.sched_min_granularity_ns=7500000
```

Al terminar, restaura el valor del parámetro.

```
echo $((750000*10)) | sudo tee sched_min_granularity_ns
```

```

<not supported> stalled-cycles-backend
<not supported> instructions
<not supported> branches
<not supported> branch-misses

      1,940231473 seconds time elapsed

^C
[1]+  Hecho                  perf stat ./matrix1
usuario@debian:~/Escritorio/laboratorio$ perf stat ./matrix1 & ./matrix1
[1] 3046
Result: 127840000.000000
usuario@debian:~/Escritorio/laboratorio$ Result: 127840000.000000

Performance counter stats for './matrix1':

      960,326568 task-clock                #    0,504 CPUs utilized
          120 context-switches            #    0,000 M/sec
              0 CPU-migrations            #    0,000 M/sec
          3.880 page-faults                #    0,004 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

      1,903719940 seconds time elapsed

```

```
echo $((750000*100)) | sudo tee sched_min_granularity_ns
```

```

usuario@debian:~/Escritorio/laboratorio$ perf stat ./matrix1 & ./matrix1
[1] 3058
Result: 127840000.000000
usuario@debian:~/Escritorio/laboratorio$ Result: 127840000.000000

Performance counter stats for './matrix1':

    948,032245 task-clock                #    0,563 CPUs utilized
         28 context-switches            #    0,000 M/sec
          0 CPU-migrations               #    0,000 M/sec
    3.879 page-faults                   #    0,004 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

    1,684341886 seconds time elapsed

```

-Al estar aumentando el tiempo mínimo que el CFS asigna al proceso en una planificación, se necesitan muchos menos cambios de contexto para llevar a cabo la operación.

Ajuste de la memoria virtual (~15 min.)

Averigua la utilidad de los siguientes parámetros del sistema de memoria virtual (busca en <https://www.kernel.org/doc/Documentation/sysctl/vm.txt>) y su valor actual (accediendo a /proc/sys/vm o con sysctl):

- min_free_kbytes
- swappiness
- vfs_cache_pressure
- dirty_background_ratio
- dirty_ratio
- dirty_expire_centisecs
- laptop_mode

Escribe la utilidad y el valor actual de los parámetros.

- **min_free_kbytes**: Tamaño de la reserva de las páginas libres. También establece los umbrales min, low e high.
 - Aumento-> Reduce la memoria usable
 - Disminuyo-> Kernel puede no atender las peticiones del sistema
- usuario@debian:/proc/sys/vm\$ cat min_free_kbytes
- 45056

- **swappiness**: [0-100] grado en que el sistema favorece la recuperación de memoria del pool (grupos de memoria)
- usuario@debian:/proc/sys/vm\$ cat swappiness
- 60

<ul style="list-style-type: none"> • vfs_cache_pressure (Prioridad) controla la tendencia del kernel a introducir en memoria RAM bloques de dato • usuario@debian:/proc/sys/vm\$ cat vfs_cache_pressure • 100
<ul style="list-style-type: none"> • dirty_background_ratio (Porcentaje): máximo de memoria RAM disponible que se puede utilizar para cachear datos sucios (dirty) en segundo plano, es decir, aquellos que han sido modificados pero aún no se han escrito en el disco • usuario@debian:/proc/sys/vm\$ cat dirty_background_ratio • 10
<ul style="list-style-type: none"> • dirty_ratio (Porcentaje): Igual que el anterior, pero esta es en primer plano, mientras que la anterior es en segundo plano. Ambos están relacionados • usuario@debian:/proc/sys/vm\$ cat dirty_ratio • 20
<ul style="list-style-type: none"> • dirty_expire_centisecs: Se expresa en cientos y define cuando un dirty data es lo suficientemente viejo como para eliminarlo. • usuario@debian:/proc/sys/vm\$ cat dirty_expire_centisecs • 3000
<ul style="list-style-type: none"> • laptop_mode: Para conservar la batería de los equipos portátiles, reduciendo la energía y mejorando la eficiencia. Se puede activar con: laptop_mode controlled by this knob are discussed in Documentation/laptops/laptop-mode.txt. • usuario@debian:/proc/sys/vm\$ cat laptop_mode • 0

Ajuste del sistema de ficheros (~20 min.)

Consulta la página de manual de `mke2fs` y `tune2fs` y averigua qué parámetros se pueden modificar con cada herramienta.

Obtén el tiempo y la tasa de transferencia con la siguiente orden:

```
$ dd if=/dev/zero of=/var/tmp/prueba count=20K conv=notrunc &
while true; do grep Dirty /proc/meminfo; sleep 5; done;
```

crea un archivo llamado prueba en el directorio /var/tmp y escribe 20K (20,480) bloques de ceros en el archivo
conv=notrunc le dice a dd que no sobrescriba el archivo si ya existe, sino que añada el contenido al final.
while true; do grep Dirty /proc/meminfo; sleep 5; done; crea un bucle infinito que se ejecuta cada 5 segundos En cada interacción, muestra los datos sucios (No se han escrito todavía en disco)
Monitoriza los dirty datas mientras los va copiando en el archivo prueba.

La orden anterior también **muestra la cantidad de páginas "sucias", es decir modificadas en memoria pero sin actualizar en disco**. Observa si la actualización de estas páginas en disco

se produce inmediatamente o en diferido y, en este último caso, cuánto tiempo se retrasa.

```
Terminal - usuario@debian: ~
Archivo Editar Ver Terminal Ir Ayuda
[1] 3177
usuario@debian:~$ while true; do grep Dirty /proc/meminfo; sleep 5; done;20480+0 registros leídos
20480+0 registros escritos
10485760 bytes (10 MB) copiados, 0,0365156 s, 287 MB/s
usuario@debian:~$ while true; do grep Dirty /proc/meminfo; sleep 5; done;
Dirty: 10260 kB
[1]+ Hecho dd if=/dev/zero of=/var/tmp/prueba count=20K conv=notrunc
Dirty: 10260 kB
Dirty: 10260 kB
Dirty: 10280 kB
Dirty: 52 kB
Dirty: 0 kB
Dirty: 0 kB
Dirty: 0 kB
Dirty: 0 kB
Dirty: 0 kB
Dirty: 0 kB
Dirty: 0 kB
Dirty: 0 kB
Dirty: 0 kB
```

En **diferido**, ya que se ha tenido que esperar aprox.25 seg (cada it del bucle)para copiar todos los datos sucios.

El tiempo que ha tardado en copiarlo son: 0,0365256s

Repite las mediciones estableciendo el valor del parámetro `dirty_expire_centisecs` a un tercio de su valor original. Por ejemplo con:

```
$ sudo sysctl vm.dirty_expire_centisecs=1000
```

dirty_expire_centisecs *1/3:Por **defecto viene a 3000**, por lo que si lo igualamos a 1000,asignamos un tercio de su valor original.

```
usuario@debian:~$ sudo sysctl vm.dirty_expire_centisecs
[sudo] password for usuario:
vm.dirty_expire_centisecs = 3000
```

(1/3) sudo sysctl vm.dirty_expire_centisecs=1000

Al reducirlo , escribirá los datos sucios en el disco con más frecuencia.

```
Terminal - usuario@debian: ~
Archivo Editar Ver Terminal Ir Ayuda
usuario@debian:~$ sudo sysctl vm.dirty_expire_centisecs=1000
vm.dirty_expire_centisecs = 1000
usuario@debian:~$ dd if=/dev/zero of=/var/tmp/prueba count=20K conv=notrunc &
[1] 3221
usuario@debian:~$ while true; do grep Dirty /proc/meminfo; sleep 5; done;20480+0 registros leídos
20480+0 registros escritos
10485760 bytes (10 MB) copiados, 0,036317 s, 289 MB/s
^C
[1]+ Hecho dd if=/dev/zero of=/var/tmp/prueba count=20K conv=notrunc
usuario@debian:~$ dd if=/dev/zero of=/var/tmp/prueba count=20K conv=notrunc & while true; do grep Dirty /proc/meminfo; sleep 5; done;
[1] 3223
Dirty: 0 kB
20480+0 registros leídos
20480+0 registros escritos
10485760 bytes (10 MB) copiados, 0,027249 s, 385 MB/s
[1]+ Hecho dd if=/dev/zero of=/var/tmp/prueba count=20K conv=notrunc
Dirty: 10240 kB
Dirty: 8216 kB
Dirty: 8200 kB
Dirty: 0 kB
Dirty: 0 kB
Dirty: 0 kB
Dirty: 0 kB
```

Tambien en diferido con una espera de unos 15 segs aprox
El tiempo que tarda es de 0,027249

`sudo sysctl vm.dirty_ratio=1`
Reduzco la memoria sucia a un 1%

```
vm.dirty_ratio = 1
usuario@debian:~$ dd if=/dev/zero of=/var/tmp/prueba count=20K conv=notrunc & while true;
info; sleep 5; done;
[1] 3246
Dirty:                8 kB
20480+0 registros leídos
20480+0 registros escritos
10485760 bytes (10 MB) copiados, 0,0403771 s, 260 MB/s
[1]+  Hecho                  dd if=/dev/zero of=/var/tmp/prueba count=20K conv=notrunc
Dirty:                2044 kB
Dirty:                8 kB
Dirty:                8 kB
Dirty:                16 kB
Dirty:                4 kB
Dirty:                12 kB
Dirty:                4 kB
Dirty:                8 kB
Dirty:                4 kB
Dirty:                12 kB
Dirty:                12 kB
Dirty:                4 kB
Dirty:                12 kB
Dirty:                4 kB
Dirty:                8 kB
Dirty:                4 kB
Dirty:                12 kB
Dirty:                4 kB
Dirty:                8 kB
Dirty:                4 kB
Dirty:                12 kB
Dirty:                4 kB
Dirty:                0 kB
Dirty:                4 kB
Dirty:                12 kB
Dirty:                12 kB
```

En diferido, pero no puedo dar un valor de espera.

Al reducir tanto el porcentaje de memoria para datos sucios, en cada interacción va copiando muy popco a poco.

Repite las mediciones estableciendo el valor del parámetro `dirty_ratio` a 1.

Copia los resultados y escribe un breve análisis de los mismos.

Ajuste del *hardware* (~10 min.)

Consulta las páginas de manual de `blockdev` y `hdparm` y averigua qué parámetros de los discos se pueden consultar o modificar con cada herramienta.

Obtén las características del disco (virtual) con:

```
$ sudo hdparm -I /dev/sda
```

Copia los resultados y describe las características del disco.

```
/dev/sda:
ATA device, with non-removable media
    Model Number:      VBOX HARDDISK
    Serial Number:     VB1b4b71d9-20f2c631
    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
usuario@debian:~$
```

Model Number: muestra el modelo del disco.

Serial Number: muestra el número de serie del disco.

Firmware Version: muestra la versión del firmware del disco.

Standby Timer: muestra el tiempo que el disco permanece inactivo antes de entrar en modo de espera.

PIO modes: muestra los modos de transferencia de entrada/salida programable compatibles.

DMA modes: muestra los modos de transferencia de acceso directo a memoria compatibles.

Features: muestra una lista de características compatibles del disco.

Instala el paquete `ethtool`. Consulta la página de manual de `ethtool` y averigua qué parámetros de los interfaces de red se pueden consultar o modificar.

```
sudo apt-get install ethtool
```

Obtén la configuración del interfaz de red (virtual) con:

```
$ sudo ethtool -k eth0
```

Copia los resultados y describe la configuración del interfaz de red.

```

usuario@debian:~$ sudo 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: 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]
usuario@debian:~$

```

rx-checksumming: muestra si la verificación de la suma de comprobación del paquete recibido está habilitada o deshabilitada.

tx-checksumming: muestra si la generación de la suma de comprobación del paquete transmitido está habilitada o deshabilitada.

scatter-gather: muestra si el modo de recolección dispersa y reunión (scatter-gather) está habilitado o deshabilitado. Este modo permite que la tarjeta de red transmita y reciba paquetes utilizando múltiples búferes de memoria en lugar de uno solo.

tcp-segmentation-offload: muestra si el apagado de segmentación TCP está habilitado o deshabilitado. Cuando está habilitado, el controlador de la tarjeta de red maneja la segmentación de paquetes TCP/IP en lugar de la CPU del sistema.

udp-fragmentation-offload: muestra si el apagado de fragmentación UDP está habilitado o deshabilitado. Cuando está habilitado, el controlador de la tarjeta de red maneja la fragmentación de paquetes UDP/IP en lugar de la CPU del sistema.

generic-segmentation-offload: muestra si la segmentación genérica está habilitada o deshabilitada. Este modo permite que la tarjeta de red transmita paquetes TCP/IP y UDP/IP utilizando múltiples búferes de memoria en lugar de uno solo.

generic-receive-offload: muestra si la recepción genérica está habilitada o deshabilitada. Este modo permite que la tarjeta de red reciba paquetes TCP/IP y UDP/IP utilizando múltiples búferes de memoria en lugar de uno solo.

large-receive-offload: muestra si la recepción de paquetes grandes está habilitada o deshabilitada. Este modo permite que la tarjeta de red reciba paquetes más grandes de lo normal y los divida en múltiples búferes de memoria antes de pasarlos a la CPU del sistema.

rx-vlan-offload: muestra si el apagado de VLAN de recepción está habilitado o deshabilitado. Este modo permite que la tarjeta de red maneje paquetes con etiquetas de VLAN sin necesidad de que la CPU del sistema se involucre.

tx-vlan-offload: muestra si el apagado de VLAN de transmisión está habilitado o deshabilitado. Este modo permite que la tarjeta de red agregue automáticamente etiquetas de VLAN a los paquetes salientes sin necesidad de que la CPU del sistema se involucre.

ntuple-filters: muestra si el filtrado de paquetes mediante NetFilter está habilitado o deshabilitado. Este modo permite que la tarjeta de red filtre los paquetes en función de ciertas reglas establecidas en el sistema operativo.