

Autores: Rubén Izquierdo y Rocío García

Parte 1

cgroups es un mecanismo de linux para controlar la asignación de los archivos de la CPU y la RAM

```
usuario@debian:~$ sudo mount -t tmpfs cgroup /sys/fs/cgroup
[sudo] password for usuario:
usuario@debian:~$ sudo mkdir /sys/fs/cgroup/cpu
usuario@debian:~$ sudo mount -t cgroup -o cpu cpu /sys/fs/cgroup/cpu
usuario@debian:~$ cd /sys/fs/cgroup/cpu
usuario@debian:/sys/fs/cgroup/cpu$ cat cpu.shares
1024
usuario@debian:/sys/fs/cgroup/cpu$ sudo mkdir /sys/fs/cgroup/cpu/cg1
usuario@debian:/sys/fs/cgroup/cpu$ cd
usuario@debian:~$ cd /sys/fs/cgroup/cpu/cg1
usuario@debian:/sys/fs/cgroup/cpu/cg1$ echo $$ | sudo tee tasks
2313
```

El programador de Linux se ocupa de las tareas. Cada proceso y subproceso es una tarea están representada por la estructura `task_struct`

```
usuario@debian:/sys/fs/cgroup/cpu$ cat /sys/fs/cgroup/cpu/tasks
1
2
3
4
6
7
8
9
10
11
12
13
14
15
17
18
19
20
21
22
94
104
112
114
115
116
117
145
146
290
351
365
376
404
405
1853
1884
1885
1887
1888
1921
1945
1947
1949
```

```
usuario@debian:/sys/fs/cgroup/cpu/cg1$ echo $$ | sudo tee tasks
[sudo] password for usuario:
2387
```

```
usuario@debian:/sys/fs/cgroup/cpu$ cat /sys/fs/cgroup/cpu/tasks
```

```
1
2
3
4
6
7
8
9
10
11
12
13
14
15
17
18
19
20
21
22
94
104
112
114
115
116
117
145
146
290
351
365
376
404
1853
1884
1885
1887
1888
1921
1945
```

Parte 2

●**sched_latency_ns:**

Latencia de prioridad para tareas ligadas a la CPU. Aumentar esta variable aumenta el intervalo de tiempo de una tarea vinculada a la CPU. La división de tiempo de una tarea es su parte justa ponderada del período de programación:

intervalo de tiempo = período de programación * (peso de la tarea / peso total de las tareas en la cola de ejecución)

El peso de la tarea depende del buen nivel de la tarea y la política de programación. El peso mínimo de tarea para una tarea SCHED_OTHER es 15, que corresponde a nice 19. El peso máximo de tarea es 88761, que corresponde a nice -20.

Los segmentos de tiempo se hacen más pequeños a medida que aumenta la carga.

Especifica la cantidad máxima de tiempo durante el cual se considera que una tarea inactiva se está ejecutando para los cálculos de derechos. Al Aumentar esta variable aumenta la cantidad de tiempo que puede consumir una tarea de activación antes de ser reemplazado, por lo que se aumenta la latencia del programador para las tareas vinculadas a la CPU. El valor predeterminado es 6000000 (ns).

●**sched_min_granularity_ns:**

Granularidad de prioridad mínima para tareas vinculadas a la CPU. Ver sched_latency_ns para más detalles. El valor predeterminado es 4000000 (ns).

●**sched_wakeup_granularity_ns:**

El aumento de esta variable reduce la prioridad de activación, lo que reduce la perturbación de las tareas de cómputo. Se produce una mejora de la latencia de activación y el rendimiento para las tareas críticas de latencia. El valor predeterminado es 2500000 (ns).

●**sched_autogroup_enabled:** un valor de 0 desactiva la función, mientras que un valor de 1 lo habilita. El valor predeterminado en este archivo es 1, a menos que el kernel se haya iniciado con el parámetro noautogroup.

```

usuario@debian:/proc/sys/kernel$ cat sched_latency_ns
6000000
usuario@debian:/proc/sys/kernel$ cat sched_min_granularity_ns
750000
usuario@debian:/proc/sys/kernel$ cat sched_wakeup_granularity_ns
1000000
usuario@debian:/proc/sys/kernel$ cat sched_autogroup_enabled
0
usuario@debian:/proc/sys/kernel$ █

```

Observamos que el sched_min_granularity tiene un valor mayor que el por defecto , wake_up_granularity tiene un valor mayor ya que reduce la perturbación de las tareas de cómputo y autogroup_enabled tiene desactivada su función. También al aumentar la parte relativa de la CPU aumenta el intervalo de una tarea vinculada a esta .La división de tiempo de una tarea es su parte justa ponderada del período de programación:

intervalo de tiempo = período de programación * (peso de la tarea / peso total de las tareas en la cola de ejecución)

El peso de la tarea depende del buen nivel de la tarea y la política de programación. El peso mínimo de tarea para una tarea SCHED_OTHER es 15, que corresponde a nice 19. El peso máximo de tarea es 88761, que corresponde a nice -20.

Como no nos funciona, sabemos que perf es una herramienta para el análisis del rendimiento y sabemos que perf stat permite la ejecución de dicho comando y guarda el análisis del rendimiento

```

usuario@debian:~/Descargas$ perf stat ./matrix1 & ./matrix1
[1] 2804
/usr/bin/perf: línea 24: exec: perf_3.2: no se encontró
E: linux-tools-3.2 is not installed.
Result: 127840000.000000
[1]+  Salida 1                  perf stat ./matrix1
usuario@debian:~/Descargas$ perf stat ./matrix2 & ./matrix2
[1] 2807
/usr/bin/perf: línea 24: exec: perf_3.2: no se encontró
E: linux-tools-3.2 is not installed.
Result: 127840000.000000
[1]+  Salida 1                  perf stat ./matrix2
usuario@debian:~/Descargas$ man perf
usuario@debian:~/Descargas$ █

```

Parte 3

- **min_free_kbytes**

Esto se usa para forzar la VM de linux a mantener un n° mínimo de KB libres, se utiliza para calcular una marca de agua para cada zona inferior del sistema, cada una de estas tiene reservadas una serie de páginas en función de su tamaño.

Se necesita una cantidad mínima de memoria para satisfacer asignaciones Si configura esto a menos de 1024 KB, su sistema lo se rompe sutilmente, y es propenso a un bloqueo bajo cargas elevadas.

- **dirty_background_ratio:**

Contiene el porcentaje total disponible en memoria que incluye el n° de páginas libres y reclamables, el n° de páginas las cuales tiene hilos del "background kernel flusher" que serán escritos a la salida de los datos sucios

- **dirty_ratio:**

Contiene el porcentaje total disponible en memoria que incluye el n° de páginas libres y reclamables, el n° de páginas por el que un proceso genera escrituras de disco y comenzará a escribir en los datos de sucio

- **dirty_expire_centisecs:**

se usa para definir cuándo los datos sucios son lo suficientemente antiguos para ser elegibles

Para escritura por los hilos del kernel. Se expresa en 100'ths. de un segundo. Datos que han estado sucios en la memoria por más tiempo que este el intervalo se escribirá la próxima vez que se despierte un hilo de enjuague.

- **swappiness**

Este control se usa para definir qué tan agresivo cambiará el kernel páginas de memoria. Los valores más altos aumentarán la agresividad, los valores más bajos

Disminuir la cantidad de swap. Un valor de 0 le indica al núcleo qué no iniciar el intercambio hasta que la cantidad de páginas gratuitas y respaldadas por archivos sea menor que la marca de agua alta en una zona.

- **vfs_cache_pressure**

este valor porcentual controla la tendencia del kernel a reclamar. la memoria que se utiliza para el almacenamiento en caché de objetos de directorio e inodo.

- **laptop_mode**

laptop_mode

es un mando que controla el "modo portátil". Todas las cosas que son controlados por este mando se describen en Documentation / laptops / laptop-mode.txt.

```

usuario@debian:/proc/sys/vm$ cat min_free_kbytes
45056
usuario@debian:/proc/sys/vm$ cat dirty_background_ratio
10
usuario@debian:/proc/sys/vm$ cat dirty_ratio
20
usuario@debian:/proc/sys/vm$ cat dirty_expire_centisecs
3000
usuario@debian:/proc/sys/vm$ cat swappiness
60
usuario@debian:/proc/sys/vm$ cat vfs_cache_pressure
100
usuario@debian:/proc/sys/vm$ cat laptop_mode
0
usuario@debian:/proc/sys/vm$

```

1 Parte opcional

make2fs crea un archivo de ficheros del tipo ext2/ext3/ext4 normalmente en las particiones de disco, el archivo correspondiente a los dispositivos dev/hdXX contiene el nº de bloque que hay en el si se omite se configura llamando a make3fs creandola con la opción -j más lo que sea especificado

tune2fs permite que el administrador del sistema sintoniza los parámetros en los ficheros de ext2/ext3/ext4 , algunas de las opciones que ofrece linux pueden ser desactivadas usando la opción -l

The image shows two terminal windows. The top window displays the output of a disk benchmarking command using 'dd'. The bottom window shows the output of the 'vmstat' command, which provides system statistics.

```

Terminal - usuario@debian: ~
Archivo Editar Ver Terminal Ir Ayuda
usuario@debian:~$ dd if=/dev/zero of=/var/tmp/prueba count=10K conv=notrunc
10240+0 registros leídos
10240+0 registros escritos
5242880 bytes (5,2 MB) copiados, 0,0096135 s, 545 MB/s
usuario@debian:~$ dd if=/dev/zero of=/var/tmp/prueba count=10K conv=notrunc
10240+0 registros leídos
10240+0 registros escritos
5242880 bytes (5,2 MB) copiados, 0,00892158 s, 588 MB/s
usuario@debian:~$

```

```

Terminal - usuario@debian: ~
Archivo Editar Ver Terminal Ir Ayuda
usuario@debian:~$ vmstat 5
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 442220  22348 333916    0    0   65   50   47  205  2  1 97  1
0   0       0 442064  22348 333916    0    0    0    1   41  145  1  1 99  0
0   0       0 442072  22356 333916    0    0    0    2   60  334  2  0 97  0

```

Observamos que al ejecutarlo una segunda vez tarda menos pero ocupa más espacio y lo mismo pasa cuando observamos con vmstat