Evaluación de configuraciones

Práctica 4 – Ajuste

FDI - UCM

Iván Aguilera Calle – Daniel García Moreno

# Gestión de procesos

## 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 en 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

1. **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

1. **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 un mayor alboroto entre ellas. 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 matrx1.c ejecutando el siguiente comando:

while true; do ./matrix1; done

En otra pestaña, ejecutamos el programa perf, obteniendo lo 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 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 |

/**/cuadro con los resultados**

**Podemos apreciar una disminución del número de cambios de contexto. Por otro, a medida que aumentamos el varo de *sched\_min\_granurality­\_ns*, el tiempo de ejecución aumenta hasta llegar a un punto en el que comienza a disminuir. Esto puede indicar que se ha puesto un perdió mayor que lo que dura la tarea en finalizar.**

## 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 utilizdo.

**//Colocar cuadro resultados**

Observamos, que el proceso que tienen el valor de nice por defecto (0), al tener un 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 &

**//Colocar cuadro resultados**

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 20, 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.

# Gestión de la memoria virtual

## Parámetros de la memoria virtual

Comenzamos este apartado investigando la utilidad de los siguientes 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

1. **dirty\_background\_ratio:** este parámetro es un porcentaje del total de la memoria libre que tiene “free pages” y “reclaimable pages”, el numero de paginas a partir del cual el sistema comienza a limpiar páginas sucias

sysctl vm.dirty\_background\_ratio

1. **dirty\_ratio: …**

sysctl vm.dirty\_ratio

1. **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

1. **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

1. **vfs\_cache\_pressure: …**

sysctl vm.vfs\_cache\_pressure

1. **laptop\_mode: …**

sysctl vm.laptop\_mode

1. **drop\_caches: …**

sysctl vm.drop\_caches

# Gestión de la E/S de disco

## 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

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

**//cuadro resultados**

Con el valor de *dirty\_expire\_centisecs* sin modificar (3000), observamos con la ayuda de vmstat, que la escritura se produce 30 segundos después de ejecutar la orden. **¿¿¿?¿?¿?**

**//cuadro resultados**

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 10 segundos después de ejecutar la orden. **¿¿?¿¿?¿?¿?¿?**

## Parámetros de los discos

Consultando el manual de blockdev, se observa que con esta orden podemos llamar al dispositivo ioctls 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

//poner cuadro resultado

# Gestión de la E/S de red

## 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

//poner cuadro de resultados