Resumen Scheduling

Tomás Felipe Melli

Noviembre 2024

Índice

1 Introducción

Ya anticipamos la importancia del **Scheduler** para el buen funcionamiento de un sistema. Ahora tenemos que hablar de las **Políticas de Scheduling**. Antes vamos a mencionar los pilares fundamentales que se buscan atacar a la hora de definir una de ellas :

- **Liberación de recursos**, es decir hacer que se terminen cuanto antes los proceso que tienen más recursos reservados.
- **Fairness** se refiere a que cada proceso debe recibir un quantum "justo".
- **Throughput** es el rendimiento, se buscará maximizar la cantidad de procesos que terminan por unidad de tiempo.
- **Latency**, el tiempo total que le toma a un proceso ejecutarse por completo. Se buscará minimizarla.
- **Carga del sistema**, es el tiempo de respuesta percibido por los usuarios interactivos. La idea es minimizarla.
- **Eficiencia** se refiere a que la CPU esté siempre haciendo algo útil.

2 Qué es una Ráfaga/ Burst de CPU ?

La ejecución de un proceso consiste en un *ciclo de ejecución de CPU* y espera por I/O. Normalmente, un proceso intensivo en I/O típicamente tiene ráfagas cortas de CPU. Por el contrario, uno intensivo en CPU, tiene ráfagas largas.

3 Cómo es el Scheduling en SMP (Symmetric Multiprocessing) ?

El problema es el CACHE. Si la política de Scheduling hace pasar un proceso a otro core, este llega con el cache vacío, tardando mucho más que si hubiese sido ejecutado en el mismo. El concepto de **afinidad al procesador** es intentar que el proceso utilice el mismo. Puede ser **afinidad dura** o **afinidad blanda** la política.

4 Qué tipos de Scheduler existen ?

Existen dos grupos de Schedulers.

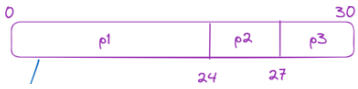
1. El **Preemptive** o apropiativo es un tipo de scheduler en el que, con la iterrupción de reloj, decide si el proceso actual debe continuar o le toca a otro. Este tipo es deseable. Lo que sí, requiere un clock con interrupciones (en general no está disponible en embebidos). Puede no dar garantías de continuidad a los procesos, cosa que es preferible en SO Real Time. Es decir, donde los procesos tienen **deadlines** y si no se cumple algo malo sucede.
   Dato de color : el clock interrumpe entre 50 y 60 veces por segundo.
2. El **No-Preemptive** o cooperativo es un tipo de scheduler en el que los procesos ceden el proce a otros procesos cuando les pinta. Sólo en los momentos de Syscalls el Scheduler se puede dar el lujo de analizar la situación (esto se debe a que el Kernel toma el control)

5 Qué políticas existen entonces ?

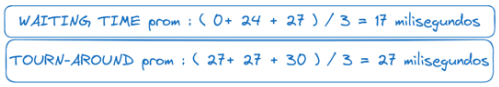
- **First-Come First-Served** es una política que modela una cola FIFO. Se le da el proce al primero en llegar. Supongamos la siguiente tabla :

Proceso	Ráfaga de CPU
P1	24
P2	3
P3	3

Para analizar estos algoritmos usamos el **Diagrama de Gantt** :



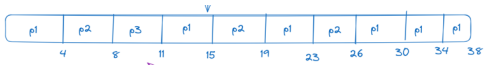
Donde el tiempo se mide en *milisegundos*. Y las magnitudes que queremos analizar son :



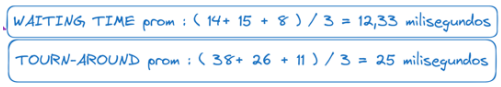
- **Round-Robin** es la política que sigue una cola de procesos Ready **en orden de llegada asignandole a cada proceso el mismo quantum**. En caso de terminar un proceso antes que otro, se desaloja voluntariamente y el scheduler asigna a la CPU al siguiente. En caso de no terminar, el scheduler lo desaloja y lo pone al final de la cola. Supongamos un quantum de 4 milisegundos :

Proceso	Ráfaga de CPU
P1	24
P2	11
P3	3

Si hacemos el diagrama de gantt :

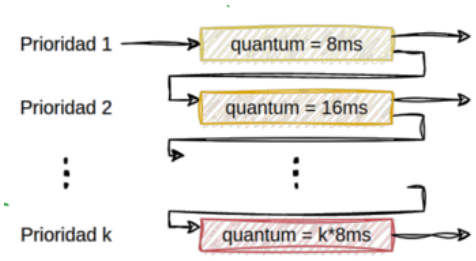


Las métricas que obtenemos son:



Detalles : el p3 termina en 3 miliseg que es menor al quantum.
Qué pasa si el quantum dura mucho ?
Parece que el sistema no responde.
Qué pasa si el quantum dura poco ?
Se va mucho tiempo en context switch. Alto porcentaje de tiempo en "Mantenimiento"
Qué tan largo debería ser ?
Lo suficiente para sacar ventaja respecto al context switch.

- La **Multilevel-Queue** es una política en donde, según disponga el scheduler, los procesos pueden cambiarse de colas. Cómo es esto ? En esta política, los procesos no están en una sóla cola, sino que hay una cantidad acotada en la que cada una tiene asociado un nivel de prioridad. Cuando a un proceso no le alcanza su cuota de CPU se lo suele pasar a la siguiente (disminuyendo su prioridad), pero en su próximo turno, recibe más tiempo de cpu. Como se ve en el diagrama :



Los procesos de **prioridad 0** son aquellos que queremos minimizar el tiempo de respuesta, normalmente, los **inteeractivos**. Ya que los procesos de cómputo largo son menos sensibles a demoras. Esto nos da pié para hablar de los **tipos de procesos** ya que según este, se asignará mayor o menor prioridad.

1. Los **Batch** son procesos periódicos con input predeterminado. Generalmente de gran volumen como las copias de seguridad.
2. Los **Interactivos** son aquellos en los que el **usuario provee el input** y es quién espera respuesta.
3. Los **Real Time** tienen restricciones fijas de tiempo y deben retornar un resultado en ese deadline.

Normalmente, dentro de cada cola se utiliza **round robin**. Podría suceder que siempre lleguen procesos a la cola más prioritaria y el sistema sufra de **Starvation**, esto es, los procesos de menor prioridad nunca reciben la cpu.

- **Shortest-Job-First** es una política de prioridad basada en **el largo de la próxima ráfaga de cpu**. Es decir, si un proceso tiene menor tiempo para terminar de ejecutarse, toma prioridad y se adelanta en la cola de procesos para usar el proce. Normalmente se utiliza para sistemas donde predominan procesos de tipo **batch** para maximizar el throughput. Esto se debe a que muchas veces se puede predecir la **duración** y con este dato, es óptimo. Ese es el tema *saber cuánta cpu va a usar cierto procesos*, si los procesos se comportan de forma regular, se puede usar para predecir.
- **Earliest-Deadline-First** es una política normalmente utilizada en procesos Real Time con deadlines, se busca priorizar aquellos que deban retornar una respuesta primero.