Creación y comparación de algoritmos calendarizadores

Carlos Peralta Coto y Jason Salazar Gonzalez

Abstract—En el siguiente documento se muestra la metodología utilizada para modificar el algoritmo calendarizador del sistema operativo pintOS.

Index Terms—PintOS, calendarización, FCFS, SJF, RoundRobin, colas multinivel.

1 Introducción

MEDIANTE la construcción, diseño e implementación de los algoritmos calendarizadores, se desea obtener un conocimiento mayor de su funcionamiento, además de que se desea comparar la eficiencia de los mismos con casos reales, que permitan obtener una referencia de como varía el comportamiento y de las ventajas y desventajas de cada uno.

2 Descripción del problema

El proyecto consiste en la modificación del Sistema Operativo PintOS, para que el mismo sea capaz de ejecutarse utilizando 4 distintos algoritmos calendarizadores. También se desea hacer una comparación de los mismos, por lo que se va a generar una salida en cada algoritmo que presente detalles sobre el tiempo de ejecución y sobre la espera de cada proceso. De esta manera se desean obtener datos reales sobre la diferencia de cada método, se puede conocer si se presentan los problemas de starvation, y si los recursos no se están utilizando de manera óptima.

3 Detalles de la implementación

A continuación se muestran algunas de las decisiones que se tomaron en la implementación de cada uno de los algoritmos.

3.1 FCFS

Para el algoritmo de First Come First Served (FCFS), se decide modificar el código en el archivo threads.c, de la carpeta pintos/scr/threads/. En el mismo lo primero que se hizo es eliminar el tiempo del quantum definido por la variable TIME_SLICE. También se eliminó su uso en la función thread_tick, que cambiaba el hilo cada vez que se llegaba al quantum. Después, en el método kernel_thread, cuando se termina de ejecutar la función, se hace la llamada al schedule para que se pase al siguiente proceso.

3.2 SJF

Para el algoritmo de Shortest Job First (SJF), se utiliza el mismo algoritmo que utiliza el FCFS, pero en la función next_thread_to_run del archivo thread.c, en vez de realizar un list_pop_front, se obtiene el proceso que tenga la menor duración en su información. Para esto también se agrega la variable tiempo a thread.h en la estructura thread.

3.3 Round Robin

El algoritmo del Round Robin, debido a que es el implementado por PintOs por defecto, se decidió sólamente modificar el quantum, con un valor de 20 ticks, y eliminar las interrupciones que se tienen, para hacerlo non-preemptive.

3.4 Colas multinivel

En este algoritmo se decidieron implementar dos colas, una para los procesos que tienen mucho uso del procesador, y otra para procesos que utilizan mucho las entradas y salidas.

4 Justificación

El algoritmo de FCFS se implementa de esa manera debido a que ya se tiene una lista de los distintos procesos en estado ready. La única modificación es evitar que se cambie el proceso antes de que se termine el mismo. También en el caso del SJF, pero en este caso se ocupa agregar la variable tiempo, para tener un parámetro que permita decidir cual es el siguiente proceso a ejecutar. En el caso del Round Robin, el quantum se cambia a 20 ticks, debido que de esta manera el procesador va a pasar menos tiempo desocupado. Si se utilizan 4 ticks, como viene por defecto, y el cambio de contexto utiliza un tick, el procesador va a estar desocupado el 20% del tiempo. Esto es debido a que por cada 5 ticks, solo va a estar funcionando 4, y 1 tick va a estar desocupado, como se puede ver en la fórmula $100/5 = 20\%$. Si se pone un quantum mayor, el algoritmo va a comportarse como un FCFS, lo que tampoco es la idea, por lo tanto se utiliza el 20.

5 Comparación entre los algoritmos

Debido a que no se consiguió generar el programa de comparación, y los algoritmos de calendarización quedaron implementados de manera incompleta, no se pudo hacer la comparación entre los distintos algoritmos. Pero en cuanto a la implementación, es más sencillo notar como el algoritmo de SJF es una modificación al algoritmo de FCFS, con la problemática de que es necesario conocer con anterioridad la duración de los algoritmos. En cuanto a las colas multi-nivel, también puede verse como dos algoritmos de FCFS, debido a la manera en que se deseaba implementar, pero también se puede hacer mediante dos o más algoritmos RoundRobin. La manera de escoger el número de colas, y el criterio para hacerlo, es una de las formas en que se puede decidir entre una u otra forma.

6 Conclusión

Al no haber sido capaces de implementar satisfactoriamente los algoritmos, se puede notar que el trabajo de realizar un algoritmo de calendarización, y en general de implementar un sistema operativo es bastante complejo, y puede llevar a caminos erróneos fácilmente. A pesar de que la complejidad de PintOS no es igual a la de otros sistemas más grandes, y que está ampliamente documentado, el trabajo que conlleva entender y modificar el sistema es muy grande. Debido a que no se tenía conocimiento previo del sistema operativo, y que el factor tiempo estuvo en contra del proyecto, no se llegaron a los resultados deseados.