Sebastian Butorovic - Gabriel Della Maggiora

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2333 — Sistemas Operativos y Redes

Tarea 1: Scheduling de threads en Pintos

La solución está diseñada en la base de que los threads y las prioridades de los threads se van almacenando en un minHeap y asi se van ordenando por sus prioridades. El uso que se la de al minHeap es que cada vez que se agrega o se remueve un elemento a la ready_list de threads, se pasan cada uno de los elementos de la ready_list al minHeap, se ordenan según la prioridad y cola, y se pasan de vuelta a la ready_list para asi mantenerla ordenada. El método que se encarga de esto es heapSort(), y es llamado cada vez que se ejecutan los metodos thread_set_priority(), thread_yield() y thread_unblock().

Para incluir el concepto de multicolas, lo que se hace es calcular una "nueva prioridad" a todos los threads al momento de pasarlos al minheap en función de la cola en la que se encuentra y la prioridad de ese thread. Posteriormente se ordena los threads en una lista al pasarlos de vuelta desde el minHeap. La función usada es:

$$data = 1000 - (queue + 1) * 63 + priority$$

Para implementar aging, lo que se hace es que cada vez que un thread completa un quantum, se incrementa la cantidad total de quantums completados por ese thread. La cantidad total de quantums completados por un thread es usada al momento de calcular a que cola pertenece, en el método get_queue(), con la fórmula:

$$queue = quantums_completados/A$$

Con respecto a la expropiación, se modificó el método create para que en el caso en que el thread creado tenga mayor prioridad que el thread en ejecución, este haga un yield y el nuevo thread se ejecute.

Los criterios para obtener los valores recomendados de A y N fueron dos:

- Disminuir la cantidad de ticks que cada proceso espera.
- Disminuir la diferencia de ticks esperados por el proceso que más espera con el que menos espera.

Para esto se corrieron algunos test probando diferentes combinaciones de los parámetros A y N. El test que entregó mejores resultados para evaluar la eficiencia del scheduler fue el test-n el cual consiste en crear 100 threads con la misma prioridad y ponerlos a dormir (asignarles prioridades diferentes no cambia mucho el resultado del test ya que de igual forma tienen que dormir). Al despertarse, los threads se van acumulando en las colas y se puede medir cuanto espera y cuantas veces cambia de cola cada uno mediante métodos implementados en la parte del accounting. Los parámetros recomendados son A=1 y N=3. La sintaxis para correr los test y entregar como parámetros A y N desde la consola es:

./pintos run test-n A N