Sistemas Operativos

TP Scheduling

Grupo 20

Daniel Grosso	694/08	dgrosso@gmail.com
Mariano De Sousa Bispo	389/08	marian_sabianaa@hotmail.com

Septiembre 2010

1. Ejercicio 4

1.1. Análisis de los resultados de `ts1`

El valor de *Waiting Time Promedio (WTP)* obtenido tanto para `FCFS` como para `SJF` fue de 87 unidades de tiempo. Esto se debe a que los procesos tienen duración creciente a medida de que llegan a la cola, por lo tanto no difiere el orden según duración al orden de llegada. Esto implica que el tiempo de espera en ambos casos es el óptimo, ya que cada proceso nuevo que llega debe a lo sumo esperar a que terminen todos los procesos incluídos anteriormente, que son los de menor duración.

1.2. Análisis de los resultados de `ts2`

En este caso, el *WTP* obtenido con `FCFS` fue de 192 unidades de tiempo, mientras que con `SJF` fue de 122 unidades de tiempo. Esta diferencia se debe a que los procesos tienen duración decreciente a medida de que están listos, por lo tanto el orden según su duración es inverso al orden de llegada. Como la primera tarea del *taskset* tiene una duración de 80 y el resto de las tareas tiene un tiempo de liberación menor a 80, todas las tareas restantes están listas para ser ejecutadas antes de que la primera tarea termine. Esto implica que el tiempo de espera con `SJF` para las tareas es el óptimo, porque ejecuta primero las de menor duración. En el caso de `FCFS`, esto sucede al revés, es decir, el tiempo de espera es el peor posible.

1.3. Análisis de los resultados de `ts3`

Para `ts3`, el *WTP* obtenido con `FCFS` fue de 127 unidades de tiempo, mientras que para `SJF` fue de 105 unidades de tiempo. Todas las tareas del *taskset* llegan en el mismo instante, por lo que `FCFS` podría establecer cualquier orden. Esto hace que, en este caso, el *WTP* no dependa de la política de `FCFS` y que pierda sentido el valor obtenido.

1.4. Análisis de los resultados de `ts4`

Con `FCFS`, el *WTP* es de 77 unidades de tiempo, mientras que para `SJF` fue de 58 unidades de tiempo. La mejora de tiempo por parte de `SJF` viene dada porque se ordena tanto en el instante 0 como en el instante 100 los procesos

que pasan a estar en estado *ready* de menor a mayor (sobre la variable de la cantidad de tiempo que necesita el `CPU` para terminar de procesar). Al igual que en `ts3`, el orden en el que se agregan las tareas que ingresan en un mismo instante en `FCFS`, hace que el análisis de la diferencia de *WTP* obtenida pierda sentido.

1.5. Análisis de los resultados de `ts5`

El task set cinco, modela los procesos entre un servidor, y varios clientes. Los valores de *WTP* son para `FCFS` de 144 unidades de tiempo y para `SJF` de 56. Si bien a primera vista consideramos que conviene usar el scheduling implementado con el algoritmo de `SJF`, al analizar el output otorgado por la corrida del programa notamos que, el orden en el que ejecutan los procesos no es un orden que modele la relación entre un servidor y clientes. La base de datos como el servidor son apagados previo al encendido, y finalmente se corren los clientes. Podemos decir entonces que no es correcto, ya que el servidor, una vez que todos los clientes son procesados debería apagarse y esto no ocurre. En `FCFS` no ocurre este problema, ya que debido a su implementación inicialmente los procesos son ingresados a la cola en el orden correcto. Si no supiéramos en qué momento ha de ingresar un nuevo proceso, podría llegar a pasar una situación similar al caso que tenemos en `SJF` con este task set. Es importante saber cuál es el entorno en el que los procesos van a correr para poder decidir entonces qué tipo de *scheduling* se usa, intentando maximizar el rendimiento sin obtener resultados inconsistentes.

1.6. Análisis de los resultados de `ts6`

El sexto *taskset* modela también un servidor con varios clientes, en particular con dos clientes más que el *taskset* anterior. El *WTP* de `FCFS` es de 176 unidades de tiempo, contra 57 unidades de `SJF`. Si bien uno tendería a decir que en materia de performance el *taskset* con el *scheduling* `SJF` es mejor, pero incurriríamos en el mismo error que con el `ts5`. En este, se ejecutan clientes antes de que se inicialice la base de datos y el servidor. Como conclusiones, cabe destacar las mismas consideraciones que en el `ts5`.

2. Ejercicio 5

bla bla bla

3. Ejercicio 6

bla bla bla