Sistemas Operativos

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico 1 - Cachalotes

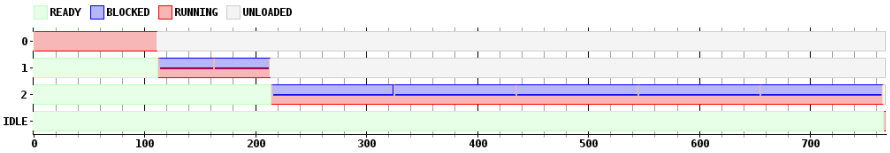
Integrante	LU	Correo electrónico
Ortiz de Zarate, Juan Manuel	403/10	jmanuoz@gmail.com
Kujawski, Kevin	459/10	kevinkuja@gmail.com
Carreiro, Martin	45/10	carreiromartin@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Ejercicio 2



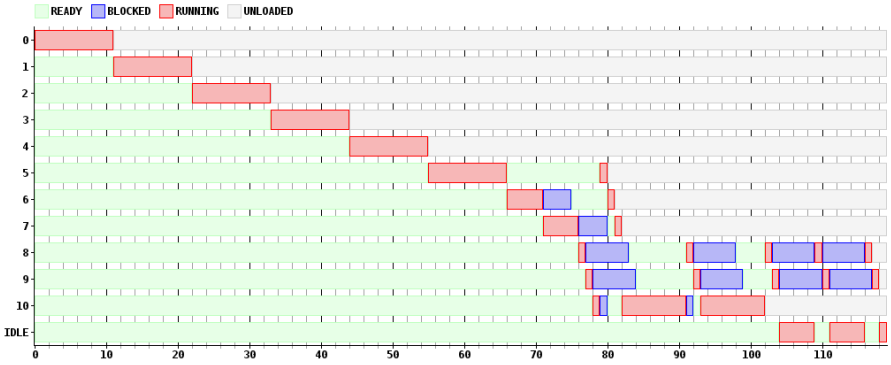
En el diagrama se puede observar como el scheduler ejecuta las tareas en el orden em que entraron. Esto es debido al tipo de scheduler que utilizamos para correr el lote de tareas, este scheduler (FIFO / FCFS) además de ejecutar las tareas en el orden de entrada, no cambia de tarea hasta que la que esta ejecutando termine.

Por esto se puede observar que la primera tarea realiza 110 ciclos de reloj y recién después pasa a la próxima tarea. Esto es porque la primera tarea que se ejecuta es "TaskCPU 110"la cual realiza exactamente 110 ciclos de reloj. Lo mismo sucede con las otras 2 tareas, se ejecutan por comleto y no hace el switch hasta finalizar la tarea acutal.

Finalmente vale destacar que las tareas interactivas cumplen lo pedido ya que la primera tarea ejecuta 2 bloqueos de aproximadenete 25 ciclos y la segunda 5 bloqueos de aproximadamente 110 ciclos cada uno. Lo que se ajusta los parametros de entrada ya que la primera tarea tiene un rango de duracion por ciclo de entre 1 y 50 y la segunda entre 100 y 150.

2. Ejercicio 4

En el siguiente gráfico se puede observar el diagrama de Gantt para el lote LoteEj4.tsk ejecutadas mediante el scheduler Round-Robin con un quantum igual 11.



Descripción del diagrama: En la ejecución de las primeras 5 tareas se parece al scheduler FIFO, debido a que puede ejecutarlas enteras en 1 solo Quantum. Pero a partir de la quinta tarea podemos observar el funcionamiento del Round-Robin en todo su esplendor. La tarea numero 5 se ejecuta durante 11 ciclos y antes de ser finalizada el scheduler la saca y pone a la siguiente debido a que se termino su 'tiempo' (llego a los 11 ciclos, es decir alcanzo el quantum). La tarea numero 6 se ejecuta durante 4 cilcos y luego se bloquea durante otros 4 utlizando un cilco extra para llamar a la ejecución de la llamada. Cuando esta tarea procede a bloquearse el scheduler (a diferencia del FIFO) cambia a la tarea siguiente y esta procede de la misma manera. Las 2 tareas siguientes se comportan muy parecido nada mas que lo primero que hacen es bloquearse y por lo tanto el scheduler las deja solo 1 ciclo en ejecucion. Por último la tarea 10 también realiza un bloqueo apenas se ejecuta por una cuestion aleatoria (la tarea 10 es la taskBatch y el momento en que ejecuta el bloqueo es aleatorio). Luego de haber ejecutado lo correspondiente a todas las tareas vuelve a la numero 5 (que no habia sido terminada de ejecutar), una vez que esta termina, cambia a la siguiente (aunque no se haya cumplido el quantum, si no desperdiciarian ciclos).Lo mismo sucede con las dos tareas siguientes. Como la tarea numero 8 sigue bloqueada, no la ejecuta al igual que la tarea 9. Carga la tarea 10 hasta que esta se vuelve a bloquear y la cambia por la tarea 8 (que ya no se encuentra bloqueada) esta bloquea y repite el procedimiento para la tarea 9. Termina de ejecutar la tarea 10 y aca podemos observar que hay un momento en que si bien las tareas 8 y 9 son las unicas tareas cargadas no finalizadas aún no las carga debido a que se encuentran bloqueadas y por eso se corre la tarea IDLE hasta que se desbloquea alguna de las 2. Finalmente repite esa ultima secuencia 1 vez mas y finaliza la ejecución de todas las tareas.

Resumen: A diferencia del FIFO este scheduler no (necesariamente) ejecuta una tarea hasta que esta finalize, sino que hasta que se cumpla el Quantum de ciclos, el proceso se bloquee o efectivamente concluya. Por otro lado también cabe enfatizar que el orden en que las alterna esta dado por una cola en la que las tareas se encuentran ordenandas por orden llegada, donde una vez ejecutada la última vuelve a la primera no acabada y sólo ejecuta la tarea idle en caso que el resto de las tareas se encuentren bloqueadas. Finalmente este scheduler es mas óptimo (en comparación al FIFO) cuando se desea realizar la sensación de que se está ejecutando todo en simulteaneo (multiprogramación), obviamente con un rango de quantum tal qué sea imperceptible para el usuario y hace un mejor uso del cpu ya que cuando una tarea se encuentra bloqueada la quita y pone a otra tarea que este en estado ready"para no desperdiciar ciclos de ejecución.

3. Ejercicio 5

Se indican las simplificaciones tomadas:
Primero queremos aclarar que se decicidió que a cada tarea se le asigna 1 ticket y este número irá incrementándose a medida que gane compensaciones por bloqueo.
Decidimos omitir el hecho de que una tarea pueda subdividirse en threads, junto con el manejo de tickets del usuario. Con esto queremos decir que al no existir usuarios (solo uno) no existe la posibilidad del cambio de cantidad de tickets bajo un mismo usuario.

4. Ejercicio 7

Lote simulado con Round Robin:

TaskBatch 10 1
TaskBatch 20 2
TaskBatch 13 3
TaskBatch 17 3
TaskBatch 27 2
TaskBatch 11 1

Diagrama con 2 ciclos de quantum:

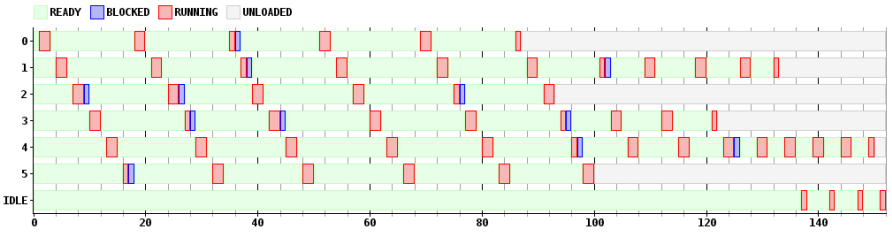


Diagrama con 5 ciclos de quantum:

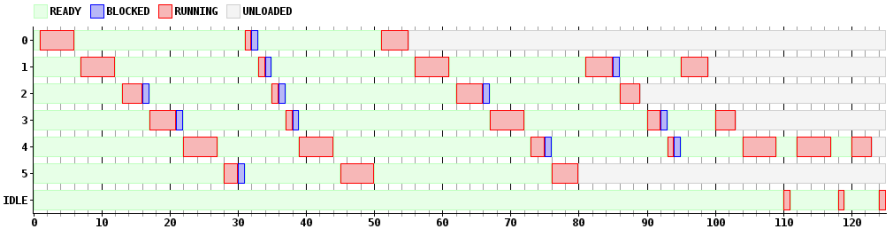


Diagrama con 7 ciclos de quantum:

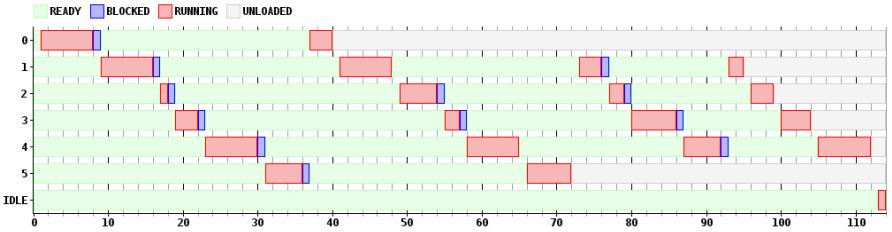


Diagrama con 9 ciclos de quantum:

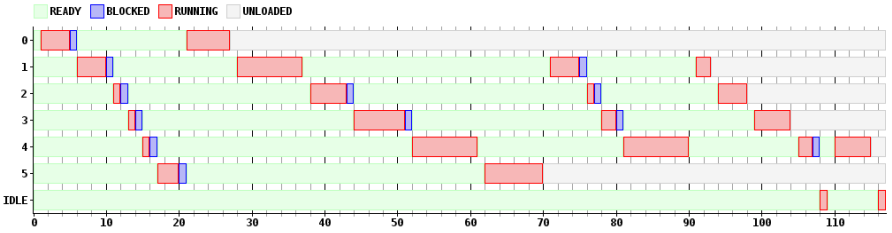


Diagrama con 12 ciclos de quantum:

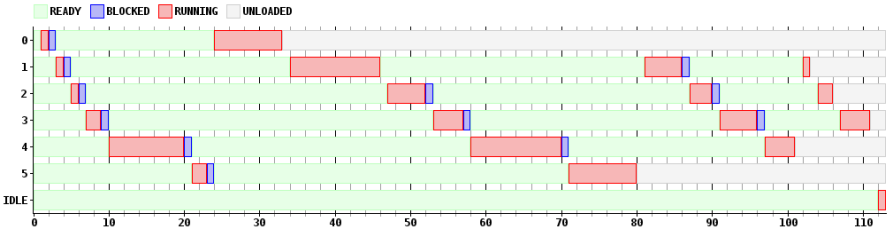


Diagrama con 17 ciclos de quantum:

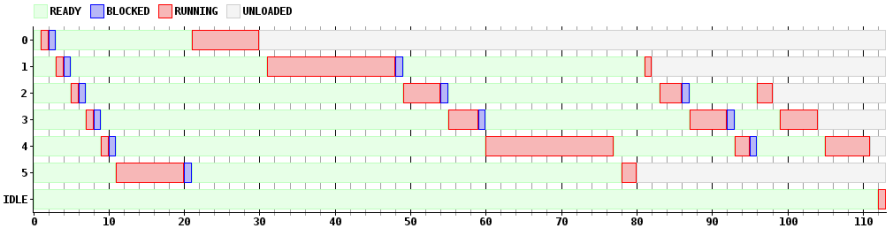
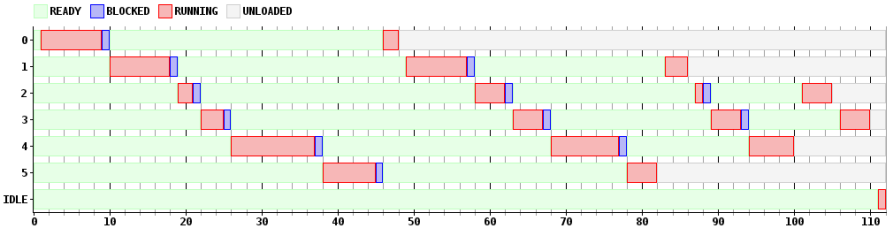


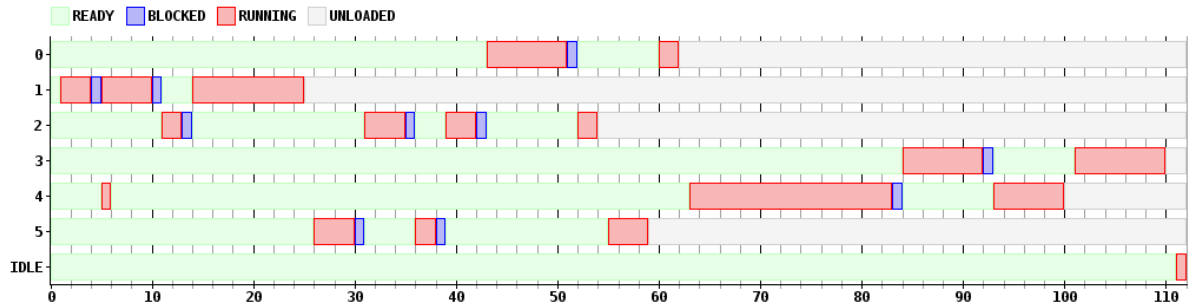
Diagrama con 21 ciclos de quantum:

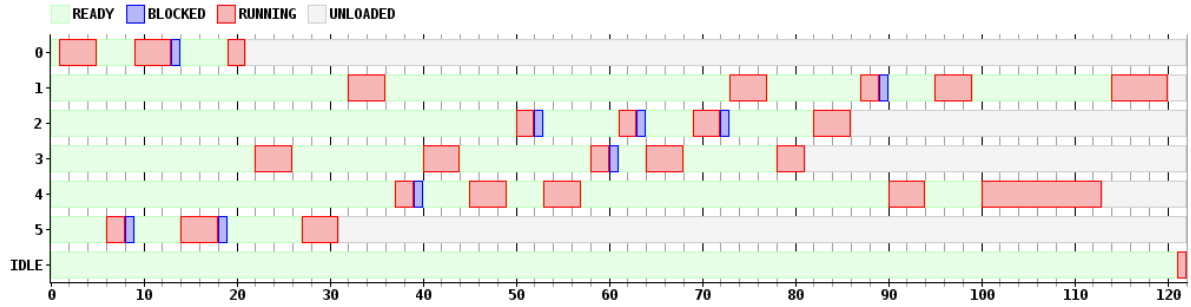


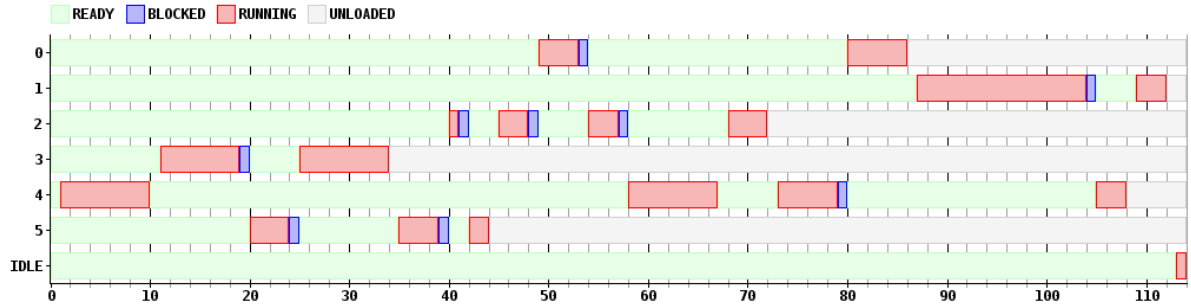
Se puede observar que a medida que los ciclos del quantum aumentan, las tareas van finalizan más rapido, debido a que se le da mas tiempo de ejecución, hay una menor cambio de contexto entre los procesos y ocurren menos bloqueos por intervalo de ejecución del proceso.
Para este lote de tareas en general, suele ser mas eficiente asignarle una cantidad grande de ciclos por quantum, ya que se realizarán menos bloqueos y cada proceso podrá finalizar en la menor cantidad de "cortes" de ejecución, pero no excesiva, porque, como se aprecia en las imágenes, entre 12 y 21 ciclos por quantum todos tardan lo mismo en finalizar y no mejora la eficiencia.
Con un quantum de 21 ciclos, se puede notar que lo que mas influye en cada proceso son los bloqueos que se producen y no tanto el corte porque no le quedan mas ciclos, produciendo cambios de contexto obligatorios y empeorando bastante la optimización del Round Robin frente a otros algoritmos de scheduling.

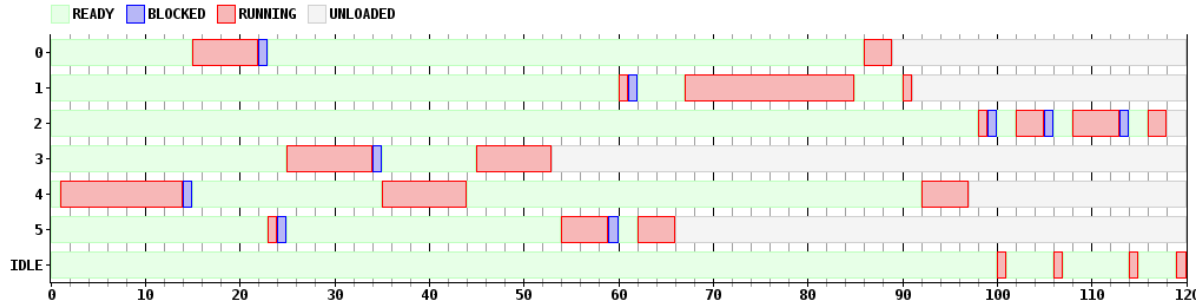
5. Ejercicio 8

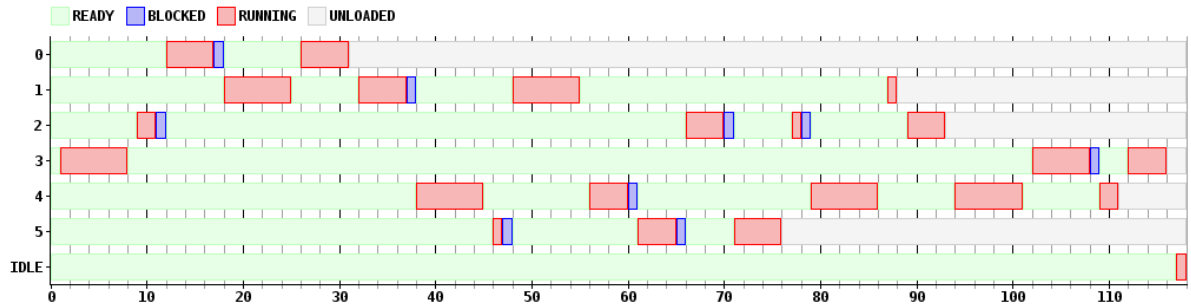
Presentamos el siguiente lote corrido con distintos semillas de aleatoridad y tamaño de quantum.
TaskBatch 10 1
TaskBatch 20 1
TaskBatch 13 3
TaskBatch 17 1
TaskBatch 27 1
TaskBatch 11 2











Se puede notar que aunque se elija el mismo lote, la aleatoridad del algoritmo muestra como aunque se asignen de distinta forma los recursos, todos los procesos tienden a obtenerlos uniformemente.
Utilizamos el mismo ejemplo para notar la "compensación". Notese como en el A el proceso 1 después de haber bloqueado 2 veces obtuvo tantos tickets que casi tuvo exclusividad del procesador. Por el contrario el proceso 4 tuvo que esperar varios sorteos hasta salir ganador.