



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA



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

TP1 - Scheduling

Sistemas Operativos

Segundo Cuatrimestre de 2014

Apellido y Nombre	LU	E-mail
Cisneros Rodrigo	920/10	rodricis@hotmail.com
Rodríguez, Agustín	120/10	agustinrodriguez90@hotmail.com
Tripodi, Guido	843/10	guido.tripodi@hotmail.com

Contents

1	Introducción	3
2	Desarrollo y Resultados	4
2.1	Parte I – Entendiendo el simulador simusched	4
2.1.1	Introduccion	4
2.1.2	Ejercicios	4
2.1.3	Resultados y Conclusiones	5
2.2	Parte II: Extendiendo el simulador con nuevos schedulers	6
2.2.1	Introduccion	6
2.2.2	Ejercicios	6
2.2.3	Resultados y Conclusiones	6
2.3	Parte 3: Evaluando los algoritmos de scheduling	7
2.3.1	Introduccion	7
2.3.2	Ejercicios	7
2.3.3	Resultados y Conclusiones	7
3	Conclusión	8

1 Introducción

En este Trabajo Práctico estudiaremos diversas implementaciones de algoritmos de scheduling. Haciendo uso de un simulador provisto por la cátedra podremos representar el comportamiento de estos algoritmos. Implementaremos dos Round-Robin, uno que permite migración de tareas entre núcleos y otro que no y a través de experimentación intentaremos comparar ambos algoritmos. Asimismo, basándonos en un paper implementaremos una versión del algoritmo *Lotery* y mediante experimentos intentaremos comprobar ciertas propiedades que cumple el algoritmo.

2 Desarrollo y Resultados

2.1 Parte I – Entendiendo el simulador simusched

2.1.1 Introduccion

2.1.2 Ejercicios

- **Ejercicio 1** Programar un tipo de tarea TaskConsola, que simulara una tarea interactiva. La tarea debe realizar n llamadas bloqueantes, cada una de una duracion al azar 1 entre bmin y bmax (inclusive). La tarea debe recibir tres parametros: n, bmin y bmax (en ese orden) que seran interpretados como los tres elementos del vector de enteros que recibe la funcion.
- **Ejercicio 2** Escribir un lote de 3 tareas distintas: una intensiva en CPU y las otras dos de tipo interactivo (TaskConsola). Ejecutar y graficar la simulacion usando el algoritmo FCFS para 1, 2 y 3 nucleos.

Grafico algoritmo FCFS con 1 nucleo.

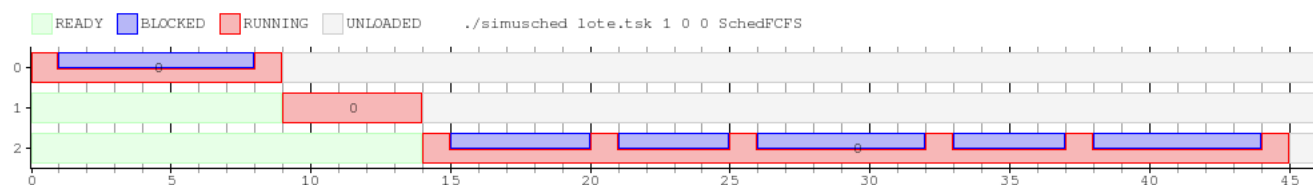


Grafico algoritmo FCFS con 2 nucleos.

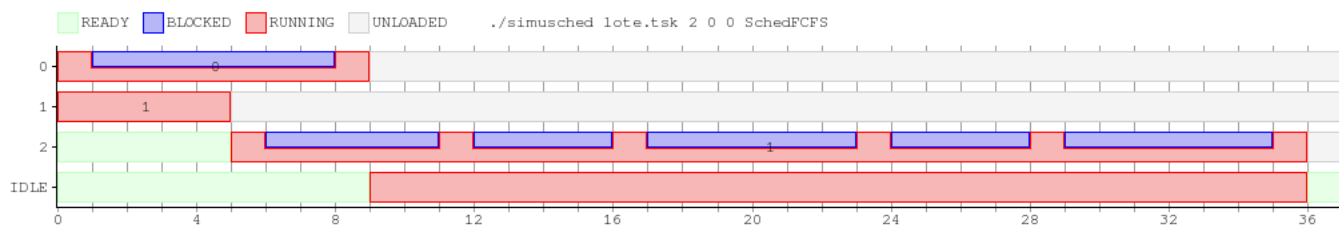
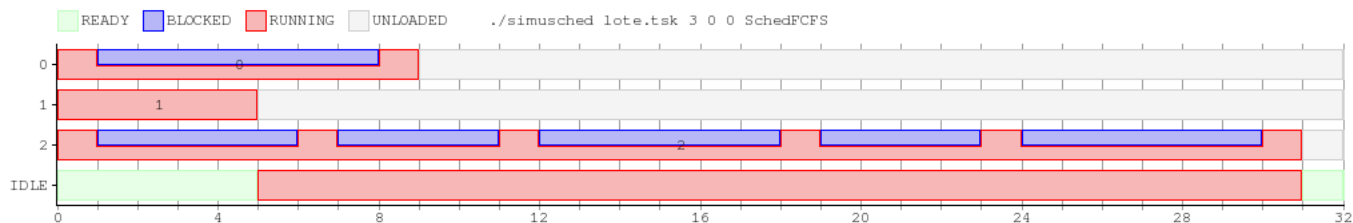


Grafico algoritmo FCFS con 3 nucleos.



Debido a las tareas de tipo TaskConsola, se observa que en los tres casos la duración de cada bloqueio es distinta.

A modo de análisis, se puede observar por medio de los gráficos como aumenta el paralelismo a mayor cantidad de núcleos. En este scheduler en particular esto ayuda de gran manera al rendimiento del sistema, puesto que un nucleo podrá ejecutar otra tarea recién cuando haya terminado la anterior. Las

consecuencias de este comportamiento son visibles en los 3 graficos. Agregando un core más, el tiempo que se tarda en ejecutar por completo todas las tareas se reduce casi a la mitad.

2.1.3 Resultados y Conclusiones

2.2 Parte II: Extendiendo el simulador con nuevos schedulers

2.2.1 Introduccion

2.2.2 Ejercicios

- **Ejercicio 3** Completar la implementacion del scheduler Round-Robin implementando los metodos de la clase SchedRR en los archivos sched rr.cpp y sched rr.h. La implementacion recibe como primer parametro la cantidad de nucleos y a continuacion los valores de sus respectivos quantums. Debe utilizar una unica cola global, permitiendo asi la migracion de procesos entre nucleos.
- **Ejercicio 4** Diseñar uno o mas lotes de tareas para ejecutar con el algoritmo del ejercicio anterior. Graficar las simulaciones y comentarlas, justificando brevemente por que el comportamiento observado es efectivamente el esperable de un algoritmo Round-Robin.
- **Ejercicio 5** A partir del articulo Waldspurger, C.A. and Weihl, W.E., Lottery scheduling: Flexible proportional-share re- source management. Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation – 1994. diseñar e implementar un scheduler basado en el esquema de loteria. El constructor de la clase SchedLottery debe recibir dos parametros: el quantum y la semilla de la secuencia pseudoaleatoria (en ese orden). Interesa implementar al menos la idea basica del algoritmo y la optimizacion de tickets compensatorios (compensation tickets). Otras optimizaciones y refinamientos que propone el articulo seran opcionales siempre que, en cada caso, se explique brevemente por que la optimizacion no se considero relevante a los efectos de este trabajo.

2.2.3 Resultados y Conclusiones

2.3 Parte 3: Evaluando los algoritmos de scheduling

2.3.1 Introduccion

2.3.2 Ejercicios

- **Ejercicio 6** Programar un tipo de tarea TaskBatch que reciba dos parametros: total cpu y cant bloqueos. Una tarea de este tipo debera realizar cant bloqueos llamadas bloqueantes, en momentos elegidos pseudoaleatoriamente. En cada tal ocasion, la tarea debera permanecer bloqueada durante exactamente un (1) ciclo de reloj. El tiempo de CPU total que utilice una tarea TaskBatch debera ser de total cpu ciclos de reloj (incluyendo el tiempo utilizado para lanzar las llamadas bloqueantes; no así el tiempo en que la tarea permanezca bloqueada).
- **Ejercicio 7** Elegir al menos dos metricas diferentes, definirlas y explicar la semantica de su definicion. Diseñar un lote de tareas TaskBatch, todas ellas con igual uso de CPU, pero con diversas cantidades de bloqueos. Simular este lote utilizando el algoritmo SchedRR y una variedad apropiada de valores de quantum. Mantener fijo en un (1) ciclo de reloj el costo de cambio de contexto y dos (2) ciclos el de migracion. Deben variar la cantidad de nucleos de procesamiento. Para cada una de las metricas elegidas, concluir cual es el valor optimo de quantum a los efectos de dicha metrica.
- **Ejercicio 8** Implemente un scheduler Round-Robin que no permita la migracion de procesos entre nucleos (SchedRR2). La asignacion de CPU se debe realizar en el momento en que se produce la carga de un proceso (load). El nucleo correspondiente a un nuevo proceso sera aquel con menor cantidad de procesos activos totales (RUNNING + BLOCKED + READY). Diseñe y realice un conjunto de experimentos que permita evaluar comparativamente las dos implementaciones de Round-Robin.
- **Ejercicio 9** Diseñar y llevar a cabo un experimento que permita poner a prueba la ecuanimidad (fairness) del algoritmo SchedLottery implementado. Tener en cuenta que, debido al factor pseudoaleatorio involucrado, cualquier corrida puntual podria ser arbitrariamente injusta; sin embargo, si se repite un mismo experimento n veces y se observan los resultados acumulativos, tales anomalias deberian ir desapareciendo conforme n aumenta. En otras palabras, interesa mostrar en base a evidencia empirica que el algoritmo implementado efectivamente tiende a ser totalmente ecuanime a medida que n tiende a infinito.
- **Ejercicio 10** Los autores del articulo sobre lottery scheduling alegan que la optimizacion de compensation tickets es necesaria para compensar una posible falencia del algoritmo inicialmente propuesto en ciertos escenarios. Diseñar y llevar a cabo un experimento apropiado para comprobar esta afirmacion (provocar un escenario donde se manifieste el problema, comparar simulaciones ejecutadas con y sin compensation tickets y discutir los resultados obtenidos).

2.3.3 Resultados y Conclusiones

3 Conclusión