*Simulación de procesos en Java*

Mauricio Torres Osorio

Facultad de Estadística e Informática

Universidad Veracruzana

Xalapa,Ver,México.

Diana Antonio Gómez

Facultad de Estadística e Informática

Universidad Veracruzana

Xalapa, Ver, México.

***Palabras clave — Proceso; Planificación; CPU ; Ráfaga; Algoritmo; Priorización; Optimización de Recursos;***

# Introducción

El presente documento abarca los detalles que rodean el desarrollo de un simulador de algoritmos de planificación de recursos de cómputo de un CPU, el simulador en cuestión comprende los algoritmos *First-Come,First-Served* y *Round-Robin.*

# Antecedentes

La búsqueda por la optimización de recursos y maximización de la productividad de una computadora llevó a la creación de la multiprogramación, el patrón de asignación y retiro descrito en la introducción lleva a la necesidad de planificar el uso del CPU, pues casi todos los recursos de una computadora son planificados antes de ser usados, haciendo que la planificación se vuelva una tarea fundamental del cómputo.

La planificación del CPU se encarga de decidir a cuál proceso dentro de la cola de procesos listos se le será asignado el CPU, para ello existen diferentes algoritmos de planificación de CPU, los usados en la simulación que dio origen a este artículo son: *First-Come,First-Served(FCFS)* y *Round-Robin(RR)****.*** FCFS es un algoritmo no expropiativo en el cual, el proceso solicita el CPU primero es al que se le asigna primero éste, este se implementa con una cola de procesos listos con estructura FIFO, los procesos que van llegando se colocan al final de la cola y cuando el CPU se encuentra libre se asigna al proceso que se encuentre al inicio de la cola. Al ejecutarse el proceso, éste se remueve de la cola, y cuando finaliza se libera el CPU para así asignarlo al siguiente proceso.

El segundo algoritmo(Round-Robin), está diseñado para sistemas que permiten el tiempo compartido y es expropiativo para que el sistema cambie la asignación de CPU entre procesos. Se define un quantum de tiempo que limita el tiempo que se le asigna a cada proceso, y la cola de procesos listos se trata de forma circular, pero utilizando la estructura FIFO nuevamente, es decir, los procesos nuevos se van añadiendo al final de la cola. El planificador de procesos selecciona el primer proceso que se encuentre en la cola y lo ejecuta, y se establece un *timer* que interrumpe el uso del CPU después que pase un quantum, si el tiempo de ejecución del proceso es mayor a un quantum el *timer*  desencadenará una interrupción al sistema operativo, se ejecutará un cambio de contexto y el proceso se incorporará al final de la cola. A pesar que este algoritmo permite mantener al CPU ocupado la mayoría del tiempo, si el quantum es demasiado grande, Round-Robin se convierte en FCFS, y si es muy pequeño, el cambio de contexto se vuelve excesivo.

# Marco Teórico

## El éxito de los diversos algoritmos de planificación depende del ciclo de espera y ejecución de un proceso. Estos algoritmos cuentan con propiedades diferentes, haciendo que en la elección de un algoritmo sea posible favorecer a ciertos procesos en lugar de otros. Además de los algoritmos simulados, existen otros que aplican diferentes criterios de priorización para los procesos, estos son:

*Shortest-Job-First(SJF)*. Este algoritmo asocia el tamaño de la siguiente ráfaga de CPU a cada proceso, de forma que, cuando el CPU se encuentra disponible, se asigna al proceso con la siguiente ráfama más pequeña, en caso que existan procesos con ráfagas del mismo tamaño se utiliza FCFS para poder priorizarlos. A diferencia de FCFS y RR, *Shortest-Job-First* puede llegar a ser expropiativo o no expropiativo, en el primer caso se expropia el proceso en ejecución, en el segundo caso el algoritmo permite al proceso en ejecución terminar su ráfaga de CPU.

*Planificación por Prioridad.* Una prioridad es asociada a cada proceso, y el CPU es asignado al proceso con mayor prioridad, nuevamente, si existen dos o más procesos con la misma prioridad se recurre a FCFS para ordenarlas. Las prioridades asignadas pueden definirse internamente o externamente. Internamente se usan cantidades medibles para la asignación de la prioridad, por ejemplo límites de tiempo, requerimientos de memoria, cantidad de archivos abiertos, etc. Externamente las prioridades se asignan según criterios “fuera” del sistema operativo. Un problema que conlleva el uso de este algoritmo es el bloqueo indefinido o la inanición, pues cuando se realiza la planeación es posible dejar a procesos de baja prioridad esperando indefinidamente la asignación de CPU.

*Planificación con colas multinivel*.Este algoritmo “parte” la cola de procesos listos en colas separadas según grupos, así cada proceso es asignado permanentemente a una cola, estos grupos se basan en distintas propiedades de un proceso como tamaño de la memoria, prioridad del proceso, o tipo de proceso, y cada cola emplea su propio algoritmo de planificación.

*Multilevel feedback queue.* A diferencia del algoritmo anterior, permite a los procesos moverse de una cola a otra, sin embargo, si un proceso utiliza el tiempo del CPU en exceso, será movido a una cola de más baja prioridad, en cambio, si hay un proceso que haya esperado mucho tiempo en una cola de baja prioridad, éste debe ser transferido a una cola de mayor prioridad, previniendo la inanición.

Para poder lograr el uso de cualquiera de los algoritmos de priorización ya mencionados es necesario hacer uso de uno de dos mecanismos de asignación de memoria: segmentación o paginación. El método utilizado en este proyecto fue el de paginación, ya que es el más usado comúnmente en los sistemas operativos hoy en día.

Aunque la segmentación permite que las direcciones físicas de un procesos no tengan que ser contiguas, la paginación también tiene esta propiedad pero evita la fragmentación externa y la necesidad de compactar programas. La implementación de la paginación requiere dividir la memoria física en bloques de tamaño fijo llamados marcos, y la ruptura de la memoria lógica en fragmentos (del mismo tamaño que los marcos) llamados páginas y cuando un proceso es ejecutado sus páginas son cargadas en los marcos que se encuentren libres.

# Marco práctico

Para la solución del problema a la implementación de los algoritmos de planificación *First-Come, First-Served* y *Round-Robin.* Fueron seleccionadas las tecnologías Java 8 y JavaFX para la programación del comportamiento de cada algoritmo y representación gráfica para representar el comportamiento del algoritmo programado de manera visual y entendible. La principal motivación tras la selección de JavaFX, además de su naturaleza como reemplazo de la tecnología Java Swing; sobre otras soluciones de representación gráfica es su versatilidad para la creación de interfaces de escritorio así como Rich Internet Applications (RIAs), lo que facilitará la reutilización del código generado en otros ambientes de desarrollo.

En el algoritmo desarrollado, se emplean listas ligadas para simular el comportamiento de colas, pues en la versión 8 de Java, no existe una implementación nativa de colas con manejo de concurrencia. En caso de extender el simulador a un ambiente multiprocesador, esta funcionalidad es de suma importancia, pues permitiría controlar las condiciones de carrera creadas por el uso de recursos compartidos.

Se simula la interacción de los procesos, el procesador y la unidad de administración de memoria, y cómo el cambio de estado en ellos provoca que sean movidos de una lista a otra según las propiedades de cada algoritmo de planificación como el quantum del CPU, tamaño de la memoria o su progreso.

# Conclusiones

La planificación del CPU es una tarea que involucra la selección de un proceso listo para su ejecución y la asignación del CPU a este proceso, para esto se pueden utilizar diversos algoritmos de planificación como; FCFS, que es el algoritmo más simple pero puede causar que procesos cortos tengan que pasar por largos tiempos de espera, SJF, puede llegar a considerarse óptimo según el tiempo de espera promedio más pequeño, pero su implementación es problemática pues implica la predicción de la siguiente ráfaga del CPU, Round-Robin generalmente es el más apropiado para sistemas de tiempo compartido ya que asigna el CPU al primer proceso en la cola de procesos listos por un quantum de tiempo, una vez pasado el quantum, si este tiempo asignado no fue suficiente, se toma la CPU y se asigna a otro proceso nuevo, colocando el anterior al final de la cola de procesos, finalmente, los algoritmos multi-nivel permiten el uso de distintos algoritmos de planificación de forma independiente.

La variedad de estos algoritmos exige el contar con métodos de selección, ya sea mediante el análisis matemático(para determinar su desempeño) o la simulación al imitar un algoritmo en un proceso y que muestre el desempeño realizado, sin embargo, la única técnica de selección confiable es la implementación de un algoritmo y su monitoreo en una situación “real”(Silberschatz,2013).

# Referencias

[1] Silberschatz,A., Galvin,B.,P., Gagne,G. (2013).Conceptos de Sistemas Operativos. Estados Unidos de América: John Wiley & Sons, Inc.