Universidad Nacional Autónoma de México

Facultad de Ingeniería

Sistemas operativos

Proyecto 2: Planificación por Retroalimentación Multinivel y por Lotería

Nombres:

Rojo López Luis Felipe

Uriarte Ortiz Enrique Yahir

Semestre:

2023-2

Profesor:

Gunnar Wolf

Fecha:

09/05/2023

Planificación por Retroalimentación Multinivel

¿Cómo se compararía este método con los otros abordados?

La comparación más cercana que podría visualizarse es con el planificador Round Robin ya que ambos intentan darles oportunidad a todos los procesos dejándolos ejecutarse y moviéndolos a una cola para darle oportunidad a otro proceso de ejecutarse.

¿Para qué tipo de carga es más apto y menos apto?

Este algoritmo es más apto para cargas de trabajo que tengan procesos con diferentes requisitos de tiempos de ejecución, al asignarle distintas prioridades a los procesos los que tengan alta prioridad podrán tener un uso corto del CPU, pero con más recursos de este por tener prioridad, mientras que los de baja prioridad podrán tener más tiempo de uso del CPU, pero con menos recursos asignados.

¿Qué tan susceptible resulta a producir inanición?

El planificador por retroalimentación multinivel utiliza, como su nombre lo indica, retroalimentación para ajustar dinámicamente las prioridades de los procesos. Si un proceso no se ejecuta durante un tiempo prolongado en una cola de baja prioridad, puede ser promovido a una cola de mayor prioridad para recibir más recursos de CPU, por lo que este algoritmo de planificación es capaz de mitigar este problema.

¿Qué tan justa sería su ejecución?

La ejecución de este planificador en general es justa, ya que se intenta dar a todos los procesos una oportunidad de ejecución y recibir recursos del CPU de manera equitativa. Las colas de prioridad junto con la retroalimentación hacen que estas prioridades puedan ser ajustadas según las necesidades de los procesos.

¿Qué modificaciones requeriría para planificar procesos con necesidades de tiempo real? (aunque sea tiempo real suave)

Podría crearse una cola de “altísima” prioridad para que sea exclusiva de los procesos en tiempo real y esta cola no debería ajustar su prioridad para que siempre tenga la preferencia sobre los procesos que no necesitan ejecutarse en tiempo real.

¿Alguna otra reflexión o inquietud que les provoque?

Este algoritmo me parece que es el que mejor maneja los procesos según sus necesidades y es por ello por lo que actualmente es el más empleado por muchos sistemas operativos.

Planificación por Lotería

La lógica que desarrollamos para el código fue en base al ejemplo que se mostró en la asignación de la actividad.

El código genera una lista de procesos aleatorios con nombres, tickets y duración, comenzando desde 'A', el número es aleatorio entre 5 y 8 y los tickets se generan aleatoriamente entre 0 y 10 veces, repartiendo el número de tickets al azar; y su duración es aleatoriamente entre 80 y 120 ticks Luego se imprime una tabla que muestra los datos generados, calcula la cantidad total de tickets e inicia el ciclo principal hasta que no queden tickets para asignar.

En cada iteración del ciclo, se selecciona un número aleatorio entre 0 y el total de tickets menos 1, para ajustar la selección entre todos

Luego, se busca el proceso correspondiente a ese ticket en la lista de procesos. Si el ticket seleccionado es menor que el número de tickets del proceso actual, ese proceso se considera el ganador y se selecciona para su ejecución. De lo contrario, se resta el número de tickets del proceso actual del ticket seleccionado y se busca en el siguiente proceso hasta que se encuentra el ganador.

Si el tiempo actual es menor que el número de tickets del proceso ganador, significa que el proceso aún no ha llegado. En este caso, se agrega una cadena de guiones al resultado para representar el tiempo que se espera antes de que el proceso

pueda comenzar. El tiempo actual se incrementa en la cantidad de tiempo de espera. Se imprime un mensaje para informar sobre el tiempo de espera.

Si el proceso puede ejecutarse, se agrega una cadena de caracteres de letras del nombre del proceso al resultado para representar el tiempo de ejecución del proceso.

Cuando todos los tickets se han asignado, se imprime el resultado de la lotería y la duración total de la ejecución.

El programa presento una limitación la cual no pudimos resolver, y es que una vez seleccionado un boleto ganador parece que el código se enseña con ese y lo guarda una y otra vez como ganador y muy rara vez en 'Orden de los resultados por loteria …' se imprimen 2 procesos.

¿Cómo se compararía este método con los otros abordados?

Este método es una implementación de lotería, cada uno tiene sus propias ventajas y desventajas, y la elección del algoritmo adecuado depende del tipo de carga y de los requisitos del sistema.

¿Para qué tipo de carga es más apto y menos apto?

Es adecuado para sistemas que tienen una carga de trabajo variada y en los que se requiere una planificación justa y aleatoria. Es menos adecuado para sistemas en los que se requiere una respuesta rápida y para sistemas con procesos críticos en tiempo real.

¿Qué tan susceptible resulta a producir inanición?

Es menos susceptible a la inanición en comparación al algoritmo de prioridades, ya que todos los procesos tienen una probabilidad de ser seleccionados para su ejecución.

¿Qué tan justa sería su ejecución?

Considero que bastante, ya que todos los procesos tienen una oportunidad igual de ser seleccionados para su ejecución. Sin embargo, algunos procesos pueden tener más tickets que otros, lo que significa que tienen más probabilidades de ser seleccionados, esto depende de las necesidades que usuario planifique para cada proceso.

¿Qué modificaciones requeriría para planificar procesos con necesidades de tiempo real? (aunque sea tiempo real suave)

no es adecuado para procesos críticos en tiempo real, ya que su planificación es aleatoria y no garantiza tiempos de respuesta predecibles. Para procesos con necesidades de tiempo real suave, se pueden asignar un número mínimo de tickets para garantizar que se seleccionen con cierta frecuencia. También se pueden implementar otros mecanismos, como la priorización de ciertos procesos o la asignación de tamaños de quantum más cortos para garantizar tiempos de respuesta más rápidos.

¿Alguna otra reflexión o inquietud que les provoque?

Fue un reto bastante divertido, una buena opción para sistemas que requieren una planificación justa y aleatoria, pero también es importante tener en cuenta sus limitaciones y adecuación a las necesidades del sistema, aunque su eficiencia también depende del: tamaño de la carga de trabajo, la cantidad de procesadores y la capacidad del sistema de entrada/salida.