Eficiencia de la planificación bajo

Round Robin

INDICE

1. Introducción

Este trabajo consta de una investigación acerca de las características propias del algoritmo de planificación conocido como Round Robin en un sistema monoprocesador. La planificación de un procesador es una de las partes mas importantes dentro del análisis de un sistema operativo. Vale la pena aclarar para aquellos que no estén tan introducidos en el tema, que la planificación de un procesador es la base de los sistemas operativos para la multiprogramación. El concepto de multiprogramación es: teniendo un número de programas en memoria al mismo tiempo, la CPU puede ser compartida entre ellos, haciendo más trabajos en menor tiempo. La planificación del procesador es implementar algoritmos que le permitan decidir al sistema operativo que procesos van a competir por el uso de la CPU, y al despachador, que proceso va a hacer uso de ella en cada

instante; y también la manera o técnicas que se usan para decidir cuanto tiempo de ejecución se le asigna a cada proceso.

El Round Robin es un algoritmo de los mas conocidos dentro de los algoritmos de planificación, y de los mas usados también. Consta básicamente de la asignación de un quantum a cada proceso, como tiempo máximo de uso del procesador. Me propongo con este trabajo evaluar la eficiencia del Round Robin con respecto a los objetivos de la planificación, si cumple o no con estos, y si una vez implementado puede llegar a tener deficiencias. En otras palabras, voy a intentar confirmar que el Round Robin, a pesar de ser muy completo y muy usado, debe valerse de otros algoritmos o estrategias para lograr una buena performance de la CPU. Debe tomarse en cuenta de antemano que el Round Robin asigna quantums de tiempo a los procesos según el orden de llegada a la cola de listos, ya que Round Robin es una modificación del algoritmo FIFO.

1.1 Problemática planteada:

¿Es la planificación en Round Robin lo suficientemente eficiente para lograr sola una buena performance de la CPU, o es necesario que se valga de mas algoritmos para lograr la optimización del procesador?

1.2 Grado de avance en el conocimiento actual del tema seleccionado:

Actualmente varios sistemas operativos utilizan el algoritmo de Round Robin, pero aparte lo fusionan con otros. Por ejemplo, el manejo de procesos en UNIX es por prioridad y round robin. En algunas versiones se maneja también un ajuste dinámico de la prioridad de acuerdo al tiempo que los procesos han esperado y al tiempo que ya han usado el CPU (envejecimiento) para evitar que un proceso pueda ser postergado de forma infinita y nunca ejecutarse. En OS/2, la planificación (la cual es apropiativa) se realiza mediante una calendarización de procesos o `threads' que se hace por prioridad y se les da a cada uno un intervalo de ejecución (Round Robin). Por lo visto, muchos sistemas operativos modernos y otros no tanto, utilizan el Round Robin conjuntamente con otros mecanismos.

1.3 Determinación del problema:

Una mala planificación de la CPU puede provocar diversos errores, algunos mínimos otros graves. Esto puede provocar desde el mal uso de un archivo, hasta la caída propia del sistema, pasando por conflictos como que existan procesos que nunca lleguen a ejecutarse, o se ejecuten mal, o no dispongan de recursos necesarios, o la pérdida de tiempo.

1.4 Enunciado de la Hipótesis:

Hipótesis H:

La planificación en Round Robin debe valerse de otros algoritmos o estrategias para lograr una óptima performance del procesador.

1.5 Enunciado de las variables:

Variable Principal :

Round Robin es un algoritmo de planificación eficiente al particionar el tiempo de ejecución de los procesos, pero surgen problemas que escapan de su alcance y deben ser subsanados por otros algoritmos para lograr un mejor rendimiento de la CPU.

Variables Secundarias :

− ¿La inclusión de un algoritmo útil de planificación permite mejorar considerablemente la performance del procesador?

− ¿Round Robin puede ser considerado un algoritmo útil?

− ¿Un algoritmo de planificación debe garantizar equidad, eficacia, rendimiento, ser predecible y tiempo de respuesta corto para cumplir con los objetivos de planificación mas importantes?

− ¿Round Robin cumple con garantizar equidad, eficacia, rendimiento, ser predecible y tiempo de respuesta corto?

− ¿Existen otros objetivos importantes aparte de los mencionados tal como la aseguración de la prioridad que debe cumplir un algoritmo?

− ¿Round Robin cumple con la aseguración de la prioridad?

− ¿Existen algoritmos que se podrían unir al Round Robin para subsanar sus defectos?

− ¿El algoritmo de prioridades y la estrategia del envejecimiento permiten asegurar la prioridad y evitar la postergación indefinida respectivamente?

1.6 Marco Teórico:

Las fuentes consultadas son básicamente libros especializados en Sistemas Operativos que hagan hincapié en la planificación del procesador, de autores reconocidos tales como Tanenbaum, Deitel, Carretero, etc., y páginas web relacionadas con el tema.

1.7 Metodología demostrativa:

La demostración de la hipótesis está hecha mediante razonamientos lógicos que constan de dos premisas cada uno. Estos razonamientos se van encadenando unos a otros hasta derivar en la confirmación / refutación de la variable principal.

1.8 Importancia del problema, propósitos y objetivos:

La importancia radica en como una planificación insuficiente puede provocar errores de optimización del procesador haciendo que este trabaje lento, o no pueda seleccionar que proceso necesita mas urgentemente ejecutarse, o que algunos ni siquiera lleguen a ejecutarse, entre muchos otros conflictos. Si bien los programadores de sistemas operativos deben estar muy al tanto de todo esto, quizá este trabajo pueda ser una referencia importante para aquellos que recién están empezando a analizar los sistemas operativos. Mi intención es introducir al lector en el funcionamiento básico de un sistema de planificación monoprocesador.

2. Desarrollo

2.1 Marco teórico:

Se utilizó toda la información citada en el punto 1.6 (libros y páginas web).

2.2 Estructura Lógica:

Transformación de variables secundarias en premisas

Premisa 1: variable . La inclusión de un algoritmo útil de planificación permite mejorar considerablemente la performance del procesador. *(ver Anexo 1, punto 5.1 Planificación del procesador)*

Premisa 2: variable . Round Robin puede es considerado un algoritmo útil. *(ver Anexo 2, punto 2.4.1 Planificación tipo round robin, pág. 72,73)*

Premisa 3: variable . Un algoritmo de planificación debe garantizar entre otras, equidad, eficacia, rendimiento, ser predecible y tiempo de respuesta corto para cumplir con los objetivos de planificación mas importantes. *(ver Anexo 1, punto 5.1.3 Objetivos de planificación; Anexo 2 punto 2.4 Planificación de procesos, pág. 71)*

Premisa 4: variable . Round Robin garantiza equidad, eficacia, rendimiento, ser predecible y tiempo de respuesta corto. *(ver Anexo 1, punto 5.1.6 Asignación del turno de ejecución−Round Robin; Anexo 3, punto 5.5.2 Round Robin, pág. 66; Anexo 4, Tabla 8.3, pág. 352)*

Premisa 5: variable . Existen otros objetivos importantes aparte de los mencionados tal como la aseguración de la prioridad. *(ver Anexo 5, punto 2.7 Objetivos de Planificación, pág. 37)*

Premisa 6: variable . Round Robin no cumple con la aseguración de la prioridad. *(ver Anexo 5, punto 2.12.4 Planificación de asignación en rueda, pág. 43)*

Premisa 7: variable . Existen algoritmos que se podrían unir al Round Robin para subsanar sus defectos. *(ver Anexo 1, punto 5.1.6 Asignación del turno de ejecución − por prioridad; Anexo 2, punto 2.4.2 Planificación por prioridad, pág. 73; Anexo 6, punto 2.6.4 Planificación con expropiación basada en prioridades, pág. 84)*

Premisa 8: variable . El algoritmo de prioridades y la estrategia del envejecimiento permiten asegurar la prioridad y evitar la postergación indefinida respectivamente. *(ver Anexo 1, punto 5.1.6 Asignación del turno de ejecución − por prioridad; Anexo 2, punto 2.4.2 Planificación por prioridad, pág. 73; Anexo 6, punto 2.6.4 Planificación con expropiación basada en prioridades, pág. 84)* 7

2.3 Estructura Racional Demostrativa:

Razonamiento 1: p1=>p2 " p1 => c1

Premisa 1: La inclusión de un algoritmo útil de planificación permite mejorar considerablemente la performance del procesador

Premisa 2: Round Robin puede es considerado un algoritmo útil

Conclusión 1: Round Robin permite mejorar considerablemente la performance del procesador

Razonamiento 2: p3=>p4 " p3 => c2

Premisa 3: Un algoritmo de planificación debe garantizar entre otras, equidad, eficacia, rendimiento, ser predecible y tiempo de respuesta corto para cumplir con los objetivos de planificación mas importantes

Premisa 4: Round Robin garantiza equidad, eficacia, rendimiento, ser predecible y tiempo de respuesta corto

Conclusión 2: Round Robin cumple con los objetivos de planificación mas importantes

Razonamiento 3: p5=>p6 " ("p6) => c3

Premisa 5: Existen otros objetivos importantes aparte de los mencionados tal como la aseguración de la prioridad

Premisa 6: Round Robin no cumple con la aseguración de la prioridad

Conclusión 3: Round Robin no cumple con uno de los objetivos de planificación

Razonamiento 4: p7=>p8 " p7 => c4

Premisa 7: Existen algoritmos que se podrían unir al Round Robin para subsanar sus defectos

Premisa 8: El algoritmo de prioridades y la estrategia del envejecimiento permiten asegurar la prioridad y evitar la postergación indefinida respectivamente

Conclusión 4: El algoritmo de prioridades y la estrategia del envejecimiento se podrían unir al Round Robin para subsanar sus defectos.

Razonamiento 5: c1 " c2 => c5

Premisa 9 (conclusión 1): Round Robin permite mejorar considerablemente la performance del procesador

Premisa 10 (conclusión 2): Round Robin cumple con los objetivos de planificación mas importantes

Conclusión 5: Round Robin es un algoritmo de planificación eficiente al particionar el tiempo de ejecución de los procesos

Razonamiento 6: c3 " c4 => c6

Premisa 11 (conclusión 3): Round Robin no cumple con uno de los objetivos de planificación

Premisa 12 (conclusión 4): El algoritmo de prioridades y la estrategia del envejecimiento se podrían unir al Round Robin para subsanar sus defectos.

Conclusión 6: El algoritmo de prioridades y la estrategia del envejecimiento se deberían unir al round robin para cumplir el objetivo que el round robin no puede cumplir por si solo

Razonamiento 7: c5 " c6 =>

Premisa 13 (conclusión 5): Round Robin es un algoritmo de planificación eficiente al particionar el tiempo de ejecución de los procesos

Premisa 14 (conclusión 6): El algoritmo de prioridades y la estrategia del envejecimiento se deberían unir al round robin para cumplir el objetivo que el round robin no puede cumplir por si solo

Conclusión 7 ( ) :

Round Robin es un algoritmo de planificación eficiente al particionar el tiempo de ejecución de los procesos, pero surgen problemas que escapan de su alcance y deben ser subsanados por otros algoritmos para lograr un mejor rendimiento de la CPU.

2.4 Estructura formal de demostración:

R1: p1 + p2 = c1

R2: p3 + p4 = c2

R3: p5 + p6 = c3

R4: p7 + p8 = c4

R5: c1 + c2 = c5

R6: c3 + c4 = c6

R7: c5 + c6= (Variable Principal)

Variable Principal => Hipótesis

Se confirma la variable principal , por lo tanto, se confirma la hipótesis planteada.

2.5 Transferencia de los resultados obtenidos:

Si bien los algoritmos ya han sido creados y desarrollados y hoy en día se haya llegado a una gran optimización de la CPU, este trabajo puede servir, por un lado, para que aquellos que recién comienzan a investigar como se maneja un sistema operativo, puedan quizás comprender como se reparten el procesador entre los distintos procesos que lo requieren. Y por otro lado, pone en evidencia algunas fallas comunes de algunos algoritmos como el round robin y el de prioridades, para que quienes quieran crear modificaciones de estos, puedan saber sus puntos mas débiles para tratar de optimizar su rendimiento.

3. Conclusión

La hipótesis propuesta:

La planificación en Round Robin debe valerse de otros algoritmos o estrategias para lograr una óptima performance del procesador.

Ha sido formalmente comprobada.

Bibliografía

- Sistemas Operativos 2da. Edición − H.M.Deitel
- Sistemas Operativos, Conceptos y Diseño − M.Milenkovic
- Sistemas Operativos − W.Stallings
- Sistemas Operativos Modernos − A. Tanenbaum
- Introducción a los Sistemas Operativos − MS−DOS,UNIX,OS/2, MVS,VMS,OS/400 − Alcalde, Morera, Perez−Campanera
- Sistemas Operativos − D.W.Barron
- Sistemas Operativos − J.Carretero

- www.tau.org.ar/base/lara.pue.udlap.mx/sistoper
- Diseño de sistemas operativos − Univ. Politécnica de Madrid http://laurel.datsi.fi.upm.es/"ssoo/DSO/download/procesos.pdf

- Dpto. de Informática, Univ. De Valladolid http://www.infor.uva.es/"benja/planificacion.html
- Dpto. de Informática, U. Técnica F. Santa María− Chile http://www.inf.utfsm.cl/"rmonge/so/guia1a.pdf
- Univ. Tecnológica Nacional − Facultad Regional Resistencia ftp://ecomchaco.com.ar/utn/cpu.doc
- Ayudas Temáticas http://server2.southlink.com.ar/vap/sistemas_operativos.htm

La planificación del procesador se refiere a la manera o técnicas que se usan para decidir cuánto tiempo de ejecución y cuando se le asignan a cada proceso del sistema. Obviamente, si el sistema es monousuario y monotarea no hay mucho que decidir, pero en el resto de los sistemas esto es crucial para el buen funcionamiento del sistema.

Uno de los más antiguos, sencillos, justo y de uso más amplio es el round robin. Cada proceso tiene asignado un intervalo de tiempo de ejecución, llamado su quantum. Si el proceso continúa en ejecución al final de su quantum, otro proceso se apropia de la CPU. Si el proceso está bloqueado o ha terminado antes de consumir su quantum, se alterna el uso de la CPU (por supuesto, si el proceso se ha bloqueado). El round robin es muy fácil de implantar. Todo lo que necesita el planificador es mantener una lista de procesos ejecutables. Cuando el quantum de un proceso se consume, se le coloca al final de la lista.

(Anexo 2) Se vienen a la mente varios criterios acerca de los que es un buen algoritmo de planificación. Algunas de las posibilidades más obvias son:

- Equidad: garantizar que cada proceso obtiene su proporción justa de la CPU.
- Eficacia: mantener ocupada la CPU el 100% del tiempo.
- Tiempo de respuesta: minimizar el tiempo de respuesta para los usuarios interactivos.
- Tiempo de regreso: minimizar el tiempo que deben esperar los usuarios por lotes para obtener sus resultados.
- Rendimiento: maximizar el número de tareas procesadas por hora.

(Anexo 1) En general se buscan cinco objetivos principales:

- Justicia o imparcialidad: todos los procesos deben ser tratados de la misma forma, y en algún momento obtienen su turno de ejecución o intervalos de tiempo de ejecución hasta su terminación exitosa.
- Maximizar la producción: el sistema debe de finalizar el mayor numero de procesos en por unidad de tiempo.
- Maximizar el tiempo de respuesta: cada usuario o proceso debe observar que el sistema les responde consistentemente a sus requerimientos.
- Evitar el aplazamiento indefinido: los procesos deben terminar en un plazo finito de tiempo.
- El sistema debe ser predecible: ante cargas de trabajo ligeras el sistema debe responder rápido y con cargas pesadas debe ir degradándose paulatinamente.

(Anexo 1) También llamada por turno, consiste en darle a cada proceso un intervalo de tiempo de ejecución y cada vez que este se vence ese intervalo se copia el contexto del proceso a un lugar seguro y se le da su turno a otro proceso. Los procesos están ordenados en una cola circular. La ventaja es su simplicidad, es justo y no provoca aplazamiento indefinido.

(Anexo 3) Esta política, es una mejora de la FCFS. Trata de ser mas justa en la respuesta tanto de los procesos cortos como de los largos.

Consiste en conceder a cada proceso en ejecución un determinado período de tiempo q (quantum), transcurrido el cual, si el proceso no ha terminado, se le devuelve al final de la cola de procesos preparados, concediéndose el procesador al siguiente proceso por su correspondiente quantum. Esta interrupción continúa

hasta que el proceso termine su ejecución, formando una rueda de procesos que serán ejecutados cíclicamente hasta que terminen.

(Anexo 4) Turno Rotatorio:

- Función de selección: constante
- Modo de decisión: Apropiativo (en los quantums de tiempo)
- Productividad: buena, puede ser baja si el quantum es pequeño.
- Tiempo de respuesta: ofrece un buen tiempo de respuesta para procesos cortos.
- Sobrecarga: baja.
- Efecto sobre los procesos: trato equitativo.
- Inanición: no.

(...) Asegurar la prioridad: los mecanismos de planificación deben favorecer a los procesos con prioridades mas altas.

Dar preferencia a los procesos que mantienen recursos claves: un proceso de baja prioridad podría mantener un recurso clave, que puede ser requerido por un proceso de mas alta prioridad. Si el recurso es no apropiativo, el mecanismo de planificación debe otorgar al proceso un tratamiento mejor del que le correspondería normalmente, puesto que es necesario liberar rápidamente el recurso clave.

En la planificación de asignación en rueda (round robin), los procesos se despachan en FIFO y disponen de una cantidad limitada de tiempo de cpu, llamada división de tiempo o cuanto. *(NOTA: para extender el concepto, el sistema fifo ordena los procesos según el orden de llegada y no según la prioridad o importancia o urgencia que tengan para ser ejecutados. Esta situación hace que el round robin falle en ejecutar primero los procesos que necesitan ejecutarse de manera urgente o previa).*

(Anexo 2) Una hipótesis implícita de la planificación round robin es que todos los procesos tienen la misma importancia. (...) La necesidad de tomar en cuenta los factores externos lleva a la planificación por prioridades. La idea fundamental es directa: cada proceso tiene asociada una prioridad y el proceso ejecutable con máxima prioridad es el que tiene el permiso de ejecución. Para evitar que los procesos de alta prioridad se ejecuten de forma indefinida, el planificador puede disminuir la prioridad del proceso en ejecución en cada instante.

(Anexo 1) La ventaja de este algoritmo es que es flexible en cuanto a permitir que ciertos procesos se ejecuten primero e, incluso, por mas tiempo. Su desventaja es que puede provocar aplazamiento indefinido en los procesos de baja prioridad. También provoca que el sistema sea impredecible para los procesos de baja prioridad.

(Anexo 6) Un problema habitual con la planificación basada en prioridades es la posibilidad es que los procesos de prioridad baja puedan quedar bloqueados de hecho por los de prioridad mas elevada. En general, la terminación de un proceso dentro de un tiempo finito a partir de su creación no puede ser garantizada con esta política de planificación. En sistemas donde la incertidumbre no puede ser tolerada, el remedio habitual lo proporciona la prioridad de envejecimiento, en la cual la prioridad de un proceso aumenta gradualmente en función del tiempo que el proceso lleve en el sistema. Eventualmente, los procesos mas antiguos conseguirán prioridades altas y esto les asegurará su terminación en tiempo finito.

-