Sistemas operativos avanzados

1.3 Algoritmos de planificación del procesador

Parámetros

PROCESOS	EJECUCION	LLEGADA
P1	9	0
P2	4	5
P3	6	7
P4	3	9

Cuando tenemos más de un proceso en condiciones de ejecutar, debemos escoger uno de entre ellos. Para escogerlo empleamos un algoritmo de planificación. Estos algoritmos pueden usar prioridades. En este caso a cada proceso se le asigna una prioridad y los procesos de mayor prioridad tendrán preferencia sobre los de menos. La **prioridad** de un proceso se puede modificar a lo largo de su vida, para evitar que un proceso de baja prioridad nunca llegue a ejecutarse debido a que los de alta prioridad monopolizan el procesador.

Otra característica de un algoritmo de planificación es la **expropiación**. Podemos definir un algoritmo de planificación como expropiativo si podemos retirar un proceso que se está ejecutando para introducir otro nuevo.

Para estudiar la bondad de un algoritmo de planificación se suelen estudiar algunos parámetros:

- Tiempo de espera: Tiempo que el proceso está parado o en espera desde que se lanza hasta que finaliza su ejecución.
- Tiempo de retorno: Tiempo que transcurre desde que el proceso se lanza hasta que finaliza su ejecución. Se puede ver como la suma del tiempo de espera más el tiempo de ejecución.
- Tiempo de respuesta: Tiempo que pasa desde que se manda ejecutar un proceso hasta que se ejecuta por primera vez.
- Productividad: Número de trabajos realizados por unidad de tiempo.
- Uso de la CPU: Porcentaje de tiempo que el procesador pasa ejecutando procesos.

Pasamos a explicar los diferentes algoritmos desarrollando un ejemplo sobre la siguiente tabla de procesos que representa los instantes de llegada de cada proceso y también los tiempos de ejecución respectivamente.

FCFS. First Come, First Served o lo que es lo mismo el primero que llega es el primero en ser atendido. Podemos decir que no es expropiativo y no emplea prioridades. Es un algoritmo muy sencillo de implementar, basta con emplear una cola FIFO, pero corre el peligro de que un proceso muy largo monopolice la CPU durante mucho tiempo generando tiempos de espera mayores de los que serían deseables.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
P1	X	X	X	X	X	X	X	X	X													
P2						E	E	E	E	X	X	X	X									
P3								E	E	E	E	E	E	X	X	X	X	X	X			
P4										E	E	E	E	E	E	E	E	E	E	X	X	X

Tiene tiempos de espera bastantes largos.

---

[5]. J.C. Bezdek *Pattern Recognition with fuzzy Objective Function Algorithms* Plenum Press, NY, 1981.

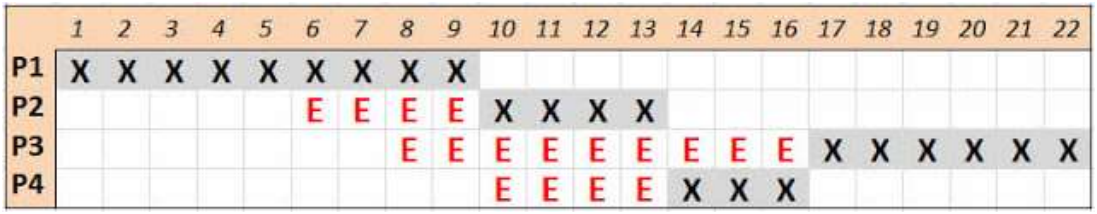
Round-Robin. También conocido como RR, Carrousel o planificación por rondas. Se reparte el tiempo de CPU en quantums o rodajas. El funcionamiento es dar una rodaja a cada proceso de forma secuencial. La selección de entre los procesos activos se gestiona según una cola FIFO o lo que es lo mismo se elige el que más tiempo lleve esperando. Si llega un proceso nuevo y hay otro en ejecución, los ciclos de CPU se distribuyen entre ambos pero se ejecuta un ciclo de CPU para el proceso en ejecución e inmediatamente se le asigna un ciclo al recién llegado. Como

Se puede deducir, este algoritmo es expropiativo y no emplea prioridades.

$\bar{t}w$ = 5,66: Tiempos de proceso y de espera según la planificación RR.
Es adecuado para implementar tiempo compartido.

Si el cuanto es muy pequeño se provocarían constantemente cambios de contexto, disminuyendo el rendimiento.

SJF. Son las siglas de Short Job First, es decir el trabajo más corto primero. En este caso se seleccionará el proceso que requiera menor tiempo de ejecución (si dos tienen el mismo tiempo se decide por FIFO). El problema puede aparecer con procesos muy largos que están siempre bloqueados por procesos más cortos. Este algoritmo puede ser expropiativo o no. En la variante expropiativa denominada **SRTN**(Shortest Remainig Time Next) medimos el tiempo restante que le queda a cada proceso.



Se minimiza el tiempo de espera medio.
Los procesos de larga duración sufren riesgo de inanición.

SJF es un caso especial de planificación por prioridad.

SJF dispone de su versión expulsiva.

---

[5]. J.C. Bezdek *Pattern Recognition with fuzzy Objective Function Algorithms* Plenum Press, NY, 1981.

Planificación por prioridad. En este tipo de algoritmos el proceso de mayor prioridad es el que se ejecuta. Para evitar que los procesos con baja prioridad no lleguen a ejecutarse, a lo largo de la vida de un proceso se puede incrementar dicha prioridad de acuerdo a diferentes criterios:

- Según la categoría del usuario.
- Según el tipo de proceso.
- Según la ocupación de CPU de los procesos.

Planificación con colas de múltiples niveles: Se usan diferentes colas, donde cada cola puede tener diferentes algoritmos de planificación y también se pueden clasificar los procesos. Se trata de repartir el tiempo de la CPU entra las diferentes colas según la carga que tenga cada una. Otra idea útil es migrar los procesos de una cola a otra cuando la situación lo requiera. Este algoritmo es uno de los más completos, pero también es de los más difíciles de implementar.

2 niveles. Hasta ahora se ha supuesto que todos los procesos están en memoria, pero qué pasa cuando hay muchos procesos, o poca memoria y no podemos almacenarlos todos allí. Una de las soluciones más elegantes consiste en establecer dos niveles, uno para planificar a largo plazo, donde se sitúan los procesos que no están en memoria y otro nivel a corto plazo donde se ponen los procesos que están en memoria.

Ejemplo

Cierto SO posee un algoritmo de planificación de CPU basado en 3 colas multinivel realimentadas. La forma en la que los trabajos se alojan en cada una de las colas es la siguiente:

- Todos los trabajos, cuando llegan al sistema, son colocados en la cola 1, la cual se planifica de acuerdo con un algoritmo Round-Robin con cuanto de tiempo igual a 2ms. en esta cola un trabajo permanecerá si después de ejecutar su primera ráfaga de CPU, le queda por ejecutar ráfagas inferiores a 5 ms. en caso contrario pasaría a la cola 2 o a la cola 3.
- Un trabajo pasará a la cola 2 en caso de que le quede por ejecutar una ráfaga mayor o igual a 5 ms. Este trabajo permanecerá en esta cola hasta que termine su ejecución y se planifica según Round-Robin con cuanto igual a 3ms.
- Un trabajo pasará a la cola 3, en caso de que le quede por ejecutar una ráfaga de CPU superior o igual a 8 ms. Este trabajo permanecerá en esta cola hasta que termine su ejecución y se planifica según SJF.

Los procesos de prioridad más baja tienen riesgo de inanición que podría ser solventado aumentando de forma progresiva la prioridad de los procesos en espera (prioridades dinámicas).

La política de prioridades puede ser o no expulsiva.

---

[5]. J.C. Bezdek *Pattern Recognition with fuzzy Objective Function Algorithms* Plenum Press, NY, 1981.

1.4 Evaluación de algoritmos

Criterios de evaluación de un algoritmo:

• Grado de utilización de la CPU.

• Tiempo de respuesta.

• Rendimiento.

„ Evaluación de políticas:

• Modelado determinista.

• Modelos de colas (estadísticos).

• Simulaciones.

• Implementación.

Modelado determinista:

„ Evaluación analítica de algoritmos:

Se calcula el desempeño de un algoritmo teniendo en cuenta la carga de trabajo del sistema:

• 1. Se definen los criterios de rendimiento.

• 2. Se buscan los algoritmos candidatos.

• 3. Se establece una carga de trabajo representativa del sistema.

• 4. Para cada algoritmo:

• Sometemos la carga de trabajo a su planificación.

• Evaluamos su rendimiento en función de los criterios de 1.

• 5. Seleccionamos el que mejor se comporte.

Modelo determinista:

„ Características:

• Cómodo de realizar.

• Proporciona magnitudes exactas con las que comparar los algoritmos.

• Limitación de su validez en cuanto a que se le somete a una carga concreta de trabajo.

Modelos de colas:

„ Metodología:

• Determinación de la distribución (estadística) de ráfagas de CPU y de E/S.

• Distribución de los tiempos de llegada al sistema.

„ Resultado: probabilidad de una ráfaga de CPU dada.

• Distribución exponencial que se describe en términos de su media.

Cálculos de:

• Rendimiento promedio.

• Tiempo de espera.

„ Sistema informático como:

• Red de servidores, cada servidor con su cola de procesos en Espera.

Simulaciones:

Programación de un modelo del sistema de computación.

Generación de datos: generador de números aleatorios modificado para generar procesos.

• Tiempo de ráfagas de CPU.

• Llegadas, partidas.

Implementación:

„ Método más fiable, forma exacta de evaluar un algoritmo de planificación.

„ Metodología:

• Codificarlo.

• Colocarlo en el SO.

• Probar su funcionamiento.

„ Problema:

• Coste elevado:

• Modificación del SO.

• Dificultar el trabajo de los usuarios, puesto que el SO está en continuo cambio.

Problemas:

• Simulaciones costosas, requieren mucho tiempo y recursos.

• Cintas de rastreo, registran secuencias de sucesos reales, requieren mucho espacio de almacenamiento.

• Tarea compleja de diseño, codificación y depuración del simulador.

Solución:

„ Plantear un esquema híbrido del tipo:

• 1. Análisis preliminar de las políticas candidatas mediante modelos deterministas.

• 2. Simulación de la opción u opciones más ventajosas.

• 3. Implementación de la opción óptima:

• Primero en un sistema de desarrollo (pruebas).

• Finalmente en un sistema de producción.

„ Problema:

• Entorno dinámico y variable (nuevos programas, nuevos problemas).

• Característica deseada: Planificación flexible, separación clara entre mecanismos y políticas.