Planificación de proceso

Temas:

9.1. Tipos de planificación del procesador

9.2. Algoritmos de planificación

10.1. Planificación multiprocesador

10.2. Planificación de tiempo real

9.1. Tipos de planificación del procesador

El objetivo de la planificación de procesos es asignar procesos a ser ejecutados por el procesador o procesadores a lo largo del tiempo, de forma que se cumplan los objetivos del sistema tales como el tiempo de respuesta, el rendimiento y la eficiencia del procesador.

Tipos:

- **Planificación a largo plazo:** La decisión de añadir un proceso al conjunto de procesos a ser ejecutados.

  Determina qué programas se admiten en el sistema para su procesamiento. De esta forma, se controla el grado de multiprogramación. Una vez admitido, un trabajo o programa de usuario se convierte en un proceso y se añade a la cola del planificador a corto plazo. En algunos sistemas, un proceso de reciente creación comienza en la zona de intercambio, en cuyo caso se añaden a la cola del planificador a medio plazo.

- **Planificación a medio plazo:** La decisión de añadir un proceso al número de procesos que están parcialmente o totalmente en la memoria principal.

  Con frecuencia, la decisión de intercambio se basa en la necesidad de gestionar el grado de multiprogramación.

- **Planificación a corto plazo:** La decisión por la que un proceso disponible será ejecutado por el procesador.

  Se ejecuta más frecuentemente para tomar decisiones de intercambio. Se invoca siempre que ocurre un evento que puede conllevar el bloqueo del proceso actual y que puede proporcionar la oportunidad de expulsar al proceso actualmente en ejecución en favor de otro.

  Algunos ejemplos de estos eventos son:

  - Interrupciones de reloj.
  - Interrupciones de E/S.
  - Llamadas al sistema.
  - Señales (por ejemplo, semáforos).

La planificación expulsiva se utiliza cuando un proceso cambia del estado de ejecución al estado listo o del estado de espera al estado listo. Los recursos (principalmente ciclos de CPU) se asignan al proceso durante un período de tiempo limitado y luego se retiran, y el proceso se vuelve a colocar en la cola lista si aún le queda tiempo de ráfaga de CPU. Ese proceso permanece en la cola de espera hasta que tenga la próxima oportunidad de ejecutarse. Algoritmos con este mecanismo son Round Robin, Shortest Remaining Time Next, Priority (Preemptive version), etc.

La planificación no expulsiva se utiliza cuando finaliza un proceso o cuando un proceso cambia del estado de ejecución al estado de espera. En esta programación, una vez que los recursos (ciclos de la CPU) se asignan a un proceso, el proceso retiene la CPU hasta que finaliza o alcanza un estado de espera. En el caso de la programación no expulsiva, no interrumpe un proceso que se ejecuta en la CPU en medio de la ejecución. En cambio, espera hasta que el proceso complete su tiempo de ráfaga de CPU y luego puede asignar la CPU a otro proceso. Algoritmos con este mecanismo son Shortest Process Next, Priority (Non preemptive version).

- **Planificación de la E/S:** La decisión por la que un proceso que está pendiente de una petición de E/S será atendido por un dispositivo de E/S disponible.

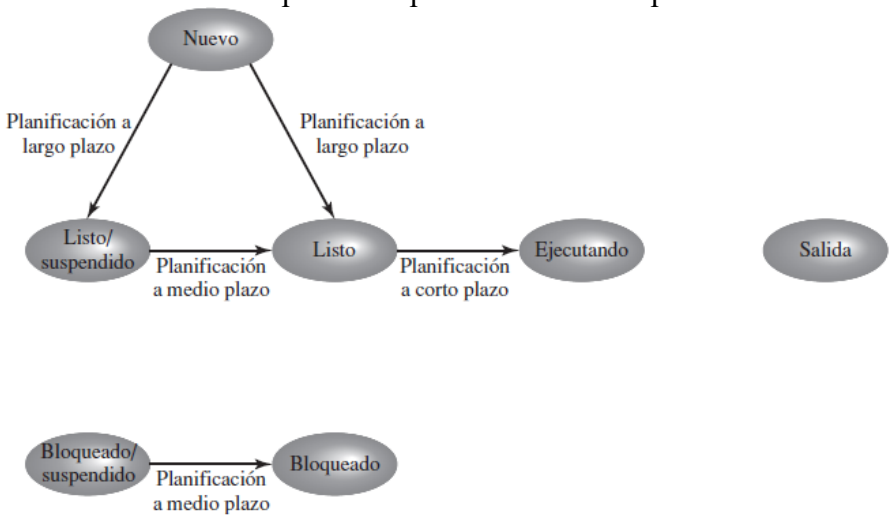


Figura 9.1. Planificación y transiciones de estado de los procesos.

9.2 Algoritmos de planificación

Criterios de la planificación a corto plazo

- **Tiempo de estancia (*turnaround time*)** Tiempo transcurrido desde que se lanza un proceso hasta que finaliza.
- **Tiempo de respuesta (*response time*)** Para un proceso interactivo, es el tiempo que trascurre desde que se lanza una petición hasta que se comienza a recibir la respuesta. La planificación debe intentar lograr bajos tiempos de respuesta y maximizar el número de usuarios interactivos con tiempos de respuesta aceptables.

- **Fecha tope (*deadlines*)** Cuando se puede especificar la fecha tope de un proceso, el planificador debe subordinar otros objetivos al de maximizar el porcentaje de fechas tope conseguidas.

Orientados al usuario, otros

- **Previsibilidad** Un trabajo dado debería ejecutarse aproximadamente en el mismo tiempo y con el mismo coste a pesar de la carga del sistema.

Orientados al sistema, relacionados con las prestaciones

- **Rendimiento** La política de planificación debería intentar maximizar el número de procesos completados por unidad de tiempo. Es una medida de cuánto trabajo está siendo realizado. Esta medida depende claramente de la longitud media de los procesos, pero está influenciada por la política de planificación, que puede afectar a la utilización.
- **Utilización del procesador** Es el porcentaje de tiempo que el procesador está ocupado. Para un sistema compartido costoso, es un criterio significativo. En un sistema de un solo usuario y en otros sistemas, tales como los sistemas de tiempo real, este criterio es menos importante que algunos otros.

Orientados al sistema, otros

- **Equidad** En ausencia de orientación de los usuarios o de orientación proporcionada por otro sistema, los procesos deben ser tratados de la misma manera, y ningún proceso debe sufrir inanición.
- **Imposición de prioridades** Cuando se asignan prioridades a los procesos, la política del planificador debería favorecer a los procesos con prioridades más altas.
- **Equilibrado de recursos** La política del planificador debería mantener ocupados los recursos del sistema. Los procesos que utilicen poco los recursos que en un determinado momento están sobre utilizados, deberían ser favorecidos. Este criterio también implica planificación a medio plazo y a largo plazo.

El uso de prioridades

Un problema de los esquemas de planificación con prioridades es que los procesos con prioridad más baja pueden sufrir inanición. Esto sucederá si hay siempre un conjunto de procesos de mayor prioridad listos para ejecutar.

Políticas de planificación alternativas

A medida que se describan las políticas de planificación se utilizará el conjunto de procesos de la Tabla 9.4 como ejemplo de ejecución. Podemos pensar en estos procesos como trabajos por lotes, con un tiempo de servicio igual al tiempo de ejecución requerido. También los podemos considerar como procesos en curso que requieren un uso alternativo del procesador y de la E/S de forma repetitiva. En este último caso, los tiempos de servicio representan el tiempo de procesador requerido en un ciclo. En cualquier caso, en términos de un modelo de colas, esta cantidad se corresponde con el tiempo de servicio.

Tabla 9.4. Ejemplo de planificación de procesos.

Proceso	Tiempo de llegada	Tiempo de servicio
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Primero en llegar, primero en servirse (first-come-first-served).

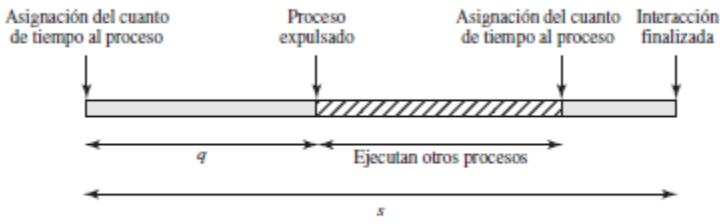
En el momento en que un proceso pasa al estado de listo, se une a la cola de listos.
FCFS no es una alternativa atractiva por sí misma para un sistema uniprocesador. Sin embargo, a menudo se combina con esquemas de prioridades para proporcionar una planificación eficaz. De esta forma, el planificador puede mantener varias colas, una por cada nivel de prioridad, y despachar dentro de cada cola usando primero en llegar primero en servirse.
FCFS funciona mucho mejor para procesos largos que para procesos cortos. Considérese el siguiente ejemplo, basado en [FINK88]:

Proceso	Tiempo de Llegada	Tiempo de Servicio (T_s)	Tiempo de Comienzo	Tiempo de Finalización	Tiempo de Estancia (T_r)	T_r/T_s
W	0	1	0	1	1	1
X	1	100	1	101	100	1
Y	2	1	101	102	100	100
Z	3	100	102	202	199	1,99
Media					100	26

Turno rotatorio (round robin)

Esta técnica es también conocida como cortar el tiempo (time slicing), porque a cada proceso se le da una rodaja de tiempo antes de ser expulsado.

El tema clave de diseño es la longitud del quantum de tiempo, o rodaja, a ser utilizada. Si el quantum es muy pequeño, el proceso se moverá por el sistema relativamente rápido. Por otra parte, existe una sobrecarga de procesamiento debido al manejo de la interrupción de reloj y por las funciones de planificación y activación. De esta forma, se deben evitar los quantums de tiempo muy pequeños.



(b) Cuanto de tiempo menor que la interacción típica

Figura 9.6. Efecto del tamaño del *quantum* de tiempo de expulsión.

Primero el proceso más corto (shortest process next)

Es una política no expulsiva en la que se selecciona el proceso con el tiempo de procesamiento más corto esperado. De esta forma un proceso corto se situará a la cabeza de la cola, por delante de los procesos más largos.

Un problema de la política SPN es la necesidad de saber, o al menos de estimar, el tiempo de procesamiento requerido por cada proceso.
La forma más sencilla de cálculo podría ser la siguiente:

$$S_{n+1} = \frac{1}{n}\sum_{i=1}^{n} T_i$$

donde,
Ti = tiempo de ejecución del procesador para la instancia *i*-ésima de este proceso (tiempo total de ejecución para los trabajos por lotes; tiempo de ráfaga de procesador para los trabajos interactivos)
Si = valor predicho para la instancia *i*-ésima
S1 = valor predicho para la primera instancia; no calculado

Para evitar volver a calcular la suma completa cada vez, podemos reescribir la ecuación como

$$S_{n+1} = \frac{1}{n}T_n + \frac{n-1}{n}S_n$$

Menor tiempo restante (shortest remaining time)

La política del menor tiempo restante (SRT) es una versión expulsiva de SPN. En este caso, a medida que entran procesos nuevos, el planificador escoge el proceso que tiene el menor tiempo restante por ejecución.

Primero el de mayor tasa de respuesta (highest response ratio next)

No podemos saber por adelantado el tiempo de servicio de los procesos, pero lo podemos aproximar, basándonos en la historia anterior, en alguna entrada de usuario o en un gestor de configuración. Considerar la siguiente tasa:

$$R = \frac{(w+s)}{s}$$

donde
R = tasa de respuesta
w = tiempo invertido esperando por el procesador
s = tiempo de servicio esperado

Si se despacha inmediatamente al proceso con este valor, R es igual al tiempo de estancia normalizado. Observar que el valor mínimo de R es 1,0, que sucede cuando un proceso acaba de entrar en el sistema.

La planificación contribución justa (fair-share scheduling)
El término *contribución justa* indica la filosofía que hay detrás de este planificador. A cada usuario se le asigna una prima de algún tipo que define la participación del usuario en los recursos del sistema como una fracción del uso total de dichos recursos. En particular, a cada usuario se le asigna una rodaja del procesador. Este esquema debería operar, más o menos, de forma lineal. Así, si un usuario A tiene el doble de prima que un usuario B, en una ejecución larga, el usuario A debería ser capaz de hacer el doble de trabajo que el usuario B.

El objetivo del planificador contribución justa es controlar el uso para dar menores recursos a usuarios que se han excedido de su contribución justa y mayores recursos a los que no han llegado.

10.1. Planificación multiprocesador

Podemos clasificar los sistemas multiprocesador como sigue:

- **Débilmente acoplado o multiprocesador distribuido, o cluster.** Consiste en una colección de sistemas relativamente autónomos; cada procesador tiene su propia memoria principal y canales de E/S.
- **Procesadores de funcionalidad especializada.** Un ejemplo es un procesador de E/S. En este caso, hay un procesador de propósito general maestro y procesadores especializados que son controlados por el procesador maestro y que le proporcionan servicios.
- **Procesamiento fuertemente acoplado.** Consiste en un conjunto de procesadores que comparten la memoria principal y están bajo el control integrado de un único sistema operativo.

Granularidad

Una buena manera de caracterizar los multiprocesadores y de situarlos en contexto respecto de otras arquitecturas, es considerar la granularidad de sincronización, o frecuencia de

sincronización entre los procesos del sistema. Podemos distinguir cinco categorías de paralelismo que difieren en el grado de granularidad.

Tabla 10.1. Granularidad de sincronización y procesos.

Tamaño del Grano	Descripción	Intervalo de sincronización (Instrucciones)
Fino	Paralelismo inherente en un único flujo de instrucciones	<20
Medio	Procesamiento paralelo o multitarea dentro de una única aplicación	20-200
Grueso	Multiprocesamiento de procesos concurrentes en un entorno multiprogramado	200-2000
Muy grueso	Procesamiento distribuido entre nodos de una red para conformar un único entorno de computación	2000-1M
Independiente	Múltiples procesos no relacionados	(N/D)

Planificación de procesos

En los sistemas multiprocesador más tradicionales, los procesos no se vinculan a los procesadores. En cambio, hay una única cola para todos los procesadores o, si se utiliza algún tipo de esquema basado en prioridades, hay múltiples colas basadas en prioridad, alimentando a un único colectivo de procesadores. En cualquier caso, el sistema puede verse como una arquitectura de colas multiservidor.

Planificación de hilos

En un monoprocesador, los hilos pueden usarse como una ayuda a la estructuración de programas y para solapar E/S con el procesamiento. Dada la mínima penalización por realizar un cambio de hilo comparado con un cambio de proceso, los beneficios se obtienen con poco coste.

Entre las muchas propuestas para la planificación multiprocesador de hilos y la asignación a procesadores, destacan cuatro enfoques generales:

- **Compartición de carga.** Los procesos no se asignan a un procesador particular. Se mantiene una cola global de hilos listos, y cada procesador, cuando está ocioso, selecciona un hilo de la cola.
- **Planificación en pandilla.** Un conjunto de hilos relacionados que se planifica para ejecutar sobre un conjunto de procesadores al mismo tiempo, en una relación uno-a-uno.
- **Asignación de procesador dedicado.** Esto es lo opuesto al enfoque de compartición de carga y proporciona una planificación implícita definida por la asignación de hilos a procesadores.

- **Planificación dinámica**. El número de hilos de un proceso puede cambiar durante el curso de su ejecución.

Planificación en pandilla

El concepto de planificar un conjunto de procesos simultáneamente sobre un conjunto de procesadores es anterior al uso de hilos.

- Si se ejecutan en paralelo procesos estrechamente relacionados, puede reducirse el bloqueo por sincronización, pueden necesitarse menos cambios de proceso y las prestaciones aumentarán.
- La sobrecarga de planificación puede reducirse dado que una decisión única afecta a varios procesadores y procesos a la vez.

Asignación de procesador dedicado

Pueden realizarse dos observaciones en defensa de esta estrategia:

- En un sistema altamente paralelo, con decenas o cientos de procesadores, cada uno de los cuales representa una pequeña fracción del coste del sistema, la utilización del procesador deja de ser tan importante como medida de la eficacia o rendimiento.
- Evitar totalmente el cambio de proceso durante la vida de un programa debe llevar a una sustancial mejora de velocidad de ese programa.

Planificación dinámica

Cuando un trabajo solicita uno o más procesadores (bien cuando el trabajo llega por primera vez o porque sus requisitos cambian),

- Si hay procesadores ociosos, utilizarlos para satisfacer la solicitud.
- En otro caso, si el trabajo que realiza la solicitud acaba de llegar, ubicarlo en un único procesador quitándoselo a cualquier trabajo que actualmente tenga más de un procesador.
- Si no puede satisfacerse cualquier parte de la solicitud, mantenerla pendiente hasta que un procesador pase a estar disponible, o el trabajo rescinda la solicitud (por ejemplo, si dejan de ser necesarios los procesadores extra). Cuando se libere uno o más procesadores (incluyendo la terminación de un trabajo),
- Examinar la cola actual de solicitudes de procesador no satisfechas. Asignar un único procesador a cada trabajo en la lista que no tenga actualmente procesadores (por ejemplo, a todos los recién llegados en espera). Luego volver a examinar la lista, volviendo a asignar el resto de los procesadores siguiendo la estrategia FCFS.

Compartición de carga

La compartición de carga es posiblemente el enfoque más simple y que se surge más directamente de un entorno monoprocesador.

[LEUT90] analiza tres versiones diferentes de compartición de carga:

- **Primero en llegar, primero en ser servido (FCFS).** Cuando llega un trabajo, cada uno de sus hilos se disponen consecutivamente al final de la cola compartida. Cuando un procesador pasa a estar ocioso, coge el siguiente hilo listo, que ejecuta hasta que se completa o se bloquea.
- **Menor número de hilos primero.** La cola compartida de listos se organiza como una cola de prioridad, con la mayor prioridad para los hilos de los trabajos con el menor número de hilos no planificados. Los trabajos con igual prioridad se ordenan de acuerdo con qué trabajo llega primero. Al igual que con FCFS, el hilo planificado ejecuta hasta que se completa o se bloquea.
- **Menor número de hilos primero con expulsión.** Se les da mayor prioridad a los trabajos con el menor número de hilos no planificados. Si llega un trabajo con menor número de hilos que un trabajo en ejecución se expulsarán los hilos pertenecientes al trabajo planificado.

10.2. Planificación de tiempo real

Características de los sistemas operativos de tiempo real

Los sistemas operativos de tiempo real pueden ser caracterizados por tener requisitos únicos en cinco áreas generales [MORG92]:

- Determinismo
- Reactividad
- Control del usuario
- Fiabilidad
- Operación de fallo suave

Para cumplir los requisitos precedentes, los sistemas operativos de tiempo real incluyen de forma representativa las siguientes características [STAN89]:

- Cambio de proceso o hilo rápido.
- Pequeño tamaño (que está asociado con funcionalidades mínimas).
- Capacidad para responder rápidamente a interrupciones externas.
- Multitarea con herramientas para la comunicación entre procesos como semáforos, señales y eventos.

- Utilización de ficheros secuenciales especiales que pueden acumular datos a alta velocidad.
- Planificación expulsiva basada en prioridades.
- Minimización de los intervalos durante los cuales se deshabilitan las interrupciones.
- Primitivas para retardar tareas durante una cantidad dada de tiempo y para parar/retomar tareas.
- Alarmas y temporizaciones especiales.

Planificación de tiempo real

En base a estas consideraciones los autores identifican las siguientes clases de algoritmos:

- **Enfoques estáticos dirigidos por tabla.** En éstos se realiza un análisis estático de la factibilidad de la planificación. El resultado del análisis es una planificación que determina cuando, en tiempo de ejecución, debe comenzar a ejecutarse cada tarea.
- **Enfoques estáticos expulsivos dirigidos por prioridad.** También se realiza un análisis estático, pero no se obtiene una planificación. En cambio, el análisis se utiliza para asignar prioridades a las tareas, y así puede utilizarse un planificador expulsivo tradicional basado en prioridades.
- **Enfoques dinámicos basados en un plan.** La factibilidad se determina en tiempo de ejecución (dinámicamente) en vez de antes de comenzar la ejecución (estáticamente). Una nueva tarea será aceptada como ejecutable sólo si es posible satisfacer sus restricciones de tiempo. Uno de los resultados del análisis de factibilidad es un plan que se usará para decidir cuándo poner en marcha la tarea.
- **Enfoques dinámicos de mejor esfuerzo.** No se realiza análisis de factibilidad. El sistema intenta cumplir todos los plazos y aborta la ejecución de cualquier proceso cuyo plazo haya fallado.

Planificación por plazos

Generalmente, las aplicaciones de tiempo-real no se preocupan tanto de la velocidad de ejecución como de completar (o comenzar) sus tareas en los momentos más adecuados, ni muy pronto ni muy tarde, a pesar de la demanda dinámica de recursos u otros conflictos, sobrecarga de procesamiento y fallos hardware o software.

En su forma más general, puede utilizarse la siguiente información de cada tarea:

- **Tiempo de activación.** Momento en el cual la tarea pasa a estar lista para ejecutar. En el caso de una tarea repetitiva o periódica se tratará de una secuencia de tiempos conocida de antemano.

- **Plazo de comienzo.** Momento en el cual la tarea debe comenzar.

- **Plazo de conclusión.** Momento para el cual la tarea debe estar completada. Las aplicaciones de tiempo real típicas tendrán plazos de comienzo o bien plazos de conclusión, pero no ambos.

- **Tiempo de proceso.** Tiempo necesario para ejecutar la tarea hasta su conclusión.

- **Recursos requeridos.** Conjunto de recursos (distintos del procesador) que la tarea necesita
- durante su ejecución.

- **Prioridad.** Mide la importancia relativa de la tarea. Las tareas de tiempo real duro pueden tener una prioridad «absoluta», provocando que el sistema falle si algún plazo no se cumple.

- **Estructura de subtareas.** Una tarea puede ser descompuesta en una subtarea obligatoria y una subtarea opcional. Sólo la subtarea obligatoria posee un plazo duro.

Planificación de tasa monótona

Uno de los métodos más prometedores para resolver los conflictos de planificación multitarea para tareas periódicas es la planificación de tasa monótona (RMS). El esquema fue propuesto por primera vez en [LIU73] pero sólo recientemente ha ganado popularidad [BRIA99, SHA94]. RMS asigna prioridades a las tareas en base a sus periodos.

Para el RMS la tarea de mayor prioridad es aquélla con el periodo más breve, la segunda tarea de mayor prioridad es aquélla con el segundo periodo más breve, y así sucesivamente. Cuando hay más de una tarea disponible para su ejecución, aquella con el periodo más breve se sirve primero.

Bibliografía

Stallings, W. (s.f.). Sistema Operativo 5 edición. En Pearson, *Aspectos internos y principio de diseño.*

https://www.geeksforgeeks.org/preemptive-and-non-preemptive-scheduling/