Sistemas Operativos

Planificación

Armando Arce, Instituto Tecnológico, arce@itcr.ac.cr

Tecnológico de Costa Rica

Planificación de procesos

La planificación de CPU es la base de los sistemas operativos multiprogramados.

- Al conmutar la CPU entre procesos, el sistema operativo puede hacer más productivo al computador.
- En sistemas operativos que soportan hilos, son los hilos a nivel de kernel y no los procesos, los que son planificados por el sistema operativo.
- Sin embargo, el término planificación de procesos y planificación de hilos se usa indistintamente.

Conceptos básicos

En un sistema de único procesador, sólo puede ejecutarse un proceso cada vez; cualquier otro procesador tendrá que esperar hasta que la CPU quede libre y pueda volver a planificarse.

- El objetivo de la multiprogramación es tener continuamente varios proyectos en ejecución, con el fin de maximizar el uso de la CPU.
- La idea es bastante simple: un proceso se ejecuta hasta que tenga que esperar, normalmente porque es necesario completar alguna solicitud de E/S.

Conceptos básicos

- En un sistema computacional simple, la CPU permanece entonces inactiva y todo el tiempo de espera se desperdicia; no se realiza ningún trabajo útil.
- Con la multiprogramación, se intenta usar ese tiempo de forma productiva.
- En este caso, se mantienen varios procesos en memoria a la vez.
- Cuando un proceso tiene que esperar, el sistema operativo retira el uso de la CPU a este proceso y se lo cede a otro proceso.
- Este patrón se repite continuamente y cada vez que un proceso tiene que esperar, otro puede hacer uso de la CPU.

Ráfagas de CPU y de E/S

El éxito de la planificación de la CPU depende de la siguiente propiedad observada de los procesos: la ejecución de un proceso consiste en un ciclo de ejecución en la CPU y espera por E/S.

- Los procesos alternan entre estos dos estados.
- La ejecución de un programa inicia con una ráfaga de CPU (CPU burst), seguida de una ráfaga de E/S (I/O burst), seguida de otra ráfaga de CPU, luego otra ráfaga de E/S, y así sucesivamente.
- Tarde o temprano, la última ráfaga de CPU termina con una solicitud al sistema para terminar la ejecución, no como otra ráfaga de E/S.

Ráfagas de CPU y de E/S

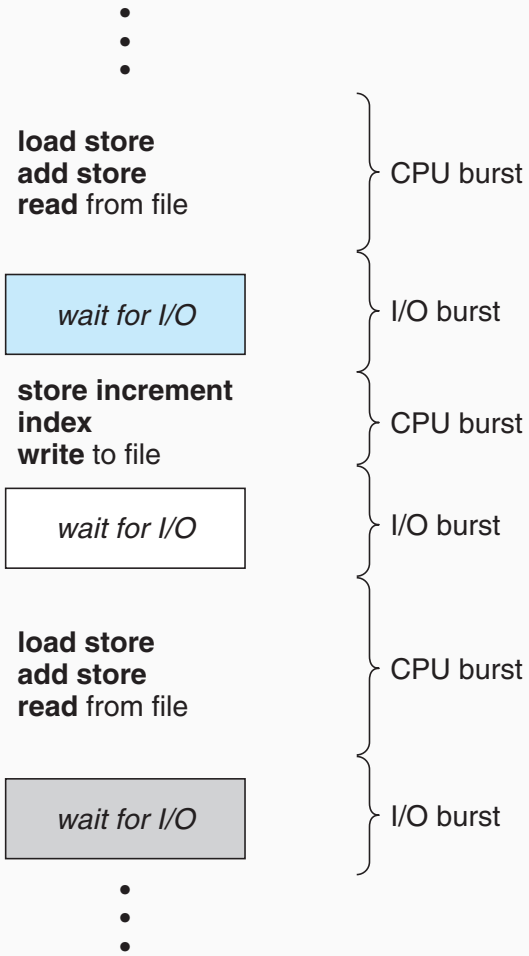


Figure 6.1 Alternating sequence of CPU and I/O bursts.

Figure 1:

Ráfagas de CPU y de E/S

Generalmente, se producen un gran número de ráfagas de CPU cortas, y pocas ráfagas de CPU largas.

- Un programa limitado por E/S (I/O bound) suele tener muchas ráfagas de CPU muy cortas; uno limitado por CPU (CPU bound) podría tener unas cuantas ráfagas de CPU muy largas.
- Esta distribución puede ser importante para la selección de un algoritmo de planificación de la CPU apropiado.

Ráfagas de CPU y de E/S

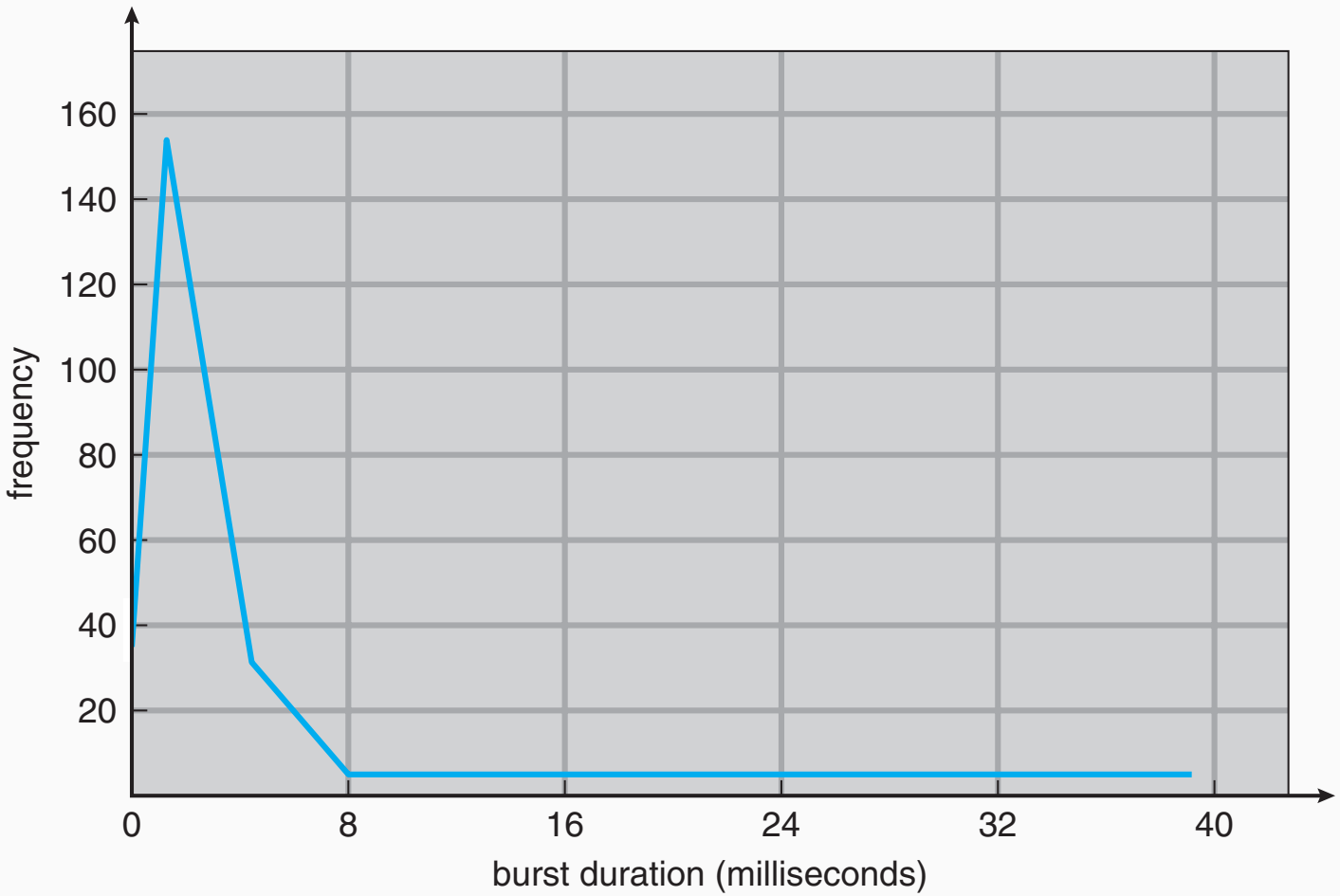


Figure 6.2 Histogram of CPU-burst durations.

Figure 2:

Planificador de CPU

Siempre que la CPU está ociosa, el sistema operativo debe escoger uno de los procesos que están en la cola de procesos listos para ejecutar.

- El proceso de selección corre por cuenta del planificador a corto plazo (o planificador de CPU), el cual escoge uno de los procesos que están en memoria y listos para ejecutarse, y se lo asigna a la CPU.

Planificador de CPU

Es importante indicar que la cola de procesos listos no es necesariamente una cola FIFO (primero que entra, primero que sale).

- Dependiendo del tipo de algoritmo de planificación, una cola de procesos listos puede implementarse como una cola FIFO, una cola de prioridad, un árbol o simplemente una lista enlazada no ordenada.
- Los registros de las colas generalmente son los PCB de los procesos.

Planificación apropiativa

Puede ser necesario tomar decisiones sobre planificación de la CPU en las siguientes cuatro circunstancias:

1. Cuando un proceso cambia del estado de ejecución al estado de espera (solicitud de E/S).
2. Cuando un proceso cambia del estado de ejecución al estado preparado (interrupción).
3. Cuando un proceso cambia del estado de espera al estado preparado (completa E/S).
4. Cuando un proceso termina.

Planificación apropiativa

Cuando las decisiones de planificación sólo tienen lugar en la circunstancias 1 y 4, se dice que el esquema de planificación es sin desalojo o operativo; en caso contrario se trata de un esquema apropiativo.

- En la planificación sin desalojo, una vez que se ha asignado la CPU a un proceso, el proceso se mantiene en la CPU hasta que ésta es liberada bien por la terminación del proceso o bien por la conmutación al estado de espera.

Planificación apropiativa

La planificación cooperativa es el único método que se puede emplear en determinadas plataformas de hardware, dado que no requiere de hardware especial (por ejemplo, un temporizador), necesario para la planificación apropiativa.

Despachador

Otro componente implicado en la función de planificación de la CPU es el despachador. El despachador es el módulo que proporciona el control de la CPU a los proyectos seleccionados por el planificador a corto plazo. Esta función implica lo siguiente:

- Cambio de contexto.
- Cambio de modo de usuario.
- Salto a la posición correcta dentro del programa de usuario para reiniciar dicho programa.

Despachador

El despachador debe ser lo más rápido posible, ya que se invoca en cada computación de proceso.

- El tiempo que tarda el despachador en detener un proceso e iniciar la ejecución de otro se conoce como *latencia de despacho*.

Criterios de planificación

Los diferentes algoritmos de planificación de la CPU tienen distintas propiedades, y la elección de un algoritmo en particular puede favorecer una clase de procesos sobre otros.

- Al escoger qué algoritmo se debe usar en una situación específica, se debe considerar las propiedades del mismo.
- Para comparar los diferentes algoritmos se pueden utilizar los siguientes criterios:

Criterios de planificación

- Utilización de la CPU (aprovechamiento): Se debe mantener la CPU tan ocupada como sea posible. Un sistema ligeramente cargado estaría ocupado un 40% del tiempo y un sistema intensamente utilizado estaría ocupado un 90%.
- Tasa de procesamiento (rendimiento): La tasa de procesamientos es la cantidad de trabajos que se completan por unidad de tiempo. Para procesos largos este valor puede ser un proceso por hora; para transacciones cortas, puede ser de 10 procesos por segundo.
- Tiempo de ejecución (retorno): El intervalo de tiempo que va desde el instante en que se ordena la ejecución de un proceso hasta el instante en que se completa es el tiempo de ejecución.

Criterios de planificación

- Tiempo de espera: El tiempo de espera en la suma de los periodos invertidos en esperar en la cola de procesos listos.
- Tiempo de respuesta: El tiempo de respuesta es el periodo transcurrido desde que se envía una solicitud hasta que se produce la primera respuesta. Este es el tiempo que el proceso tarda en empezar a responder, no el tiempo que tarda en enviar a la salida toda la información de la respuesta.

Es deseable maximizar el aprovechamiento y la productividad de la CPU, y minimizar los tiempos de retorno, espera y respuesta.

Algoritmos de planificación

La planificación de la CPU se ocupa del problema de decidir cuál de los procesos que están en la cola de procesos listos (preparados) debe recibir la CPU.

- Existen muchos algoritmos de planificación de la CPU distintos.

Planificación FCFS

El algoritmo de planificación de la CPU más sencillo es el de servicio por orden de llegada (FCFS, first-come, first-served).

- Con este esquema el proceso que primero solicita la CPU la recibe primero.
- La implementación de la política FCFS es fácil con una cola FIFO.
- Cuando un proceso ingresa en la cola de procesos listos, su PCB se enlaza al final de la cola.
- Cuando la CPU queda libre, se asigna al proceso que está a la cabeza de la cola.
- Acto seguido, el proceso de ejecución se saca de la cola.

Planificación FCFS

- El código para la planificación FCFS es fácil de escribir y de entender.
- Sin embargo, el tiempo de espera promedio cuando se adopta la política FCFS suele ser muy largo.
- FCFS rinde mejor con procesos largos que con procesos cortos.
- Por ejemplo, considere los siguientes procesos y sus ráfagas de CPU:

Process	Burst Time
P_1	24
P_2	3
P_3	3

Figure 3:

Planificación FCFS

Si los procesos llegan en el orden: p1,p2,p3 el resultado es:

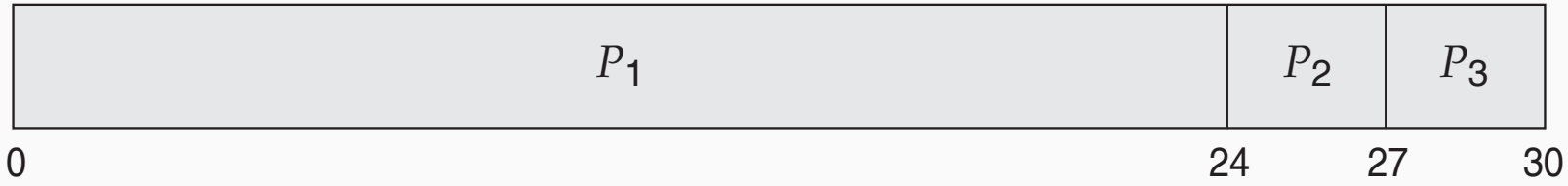


Figure 4:

Si los procesos llegan en el orden: p2,p3,p1 el resultado es:

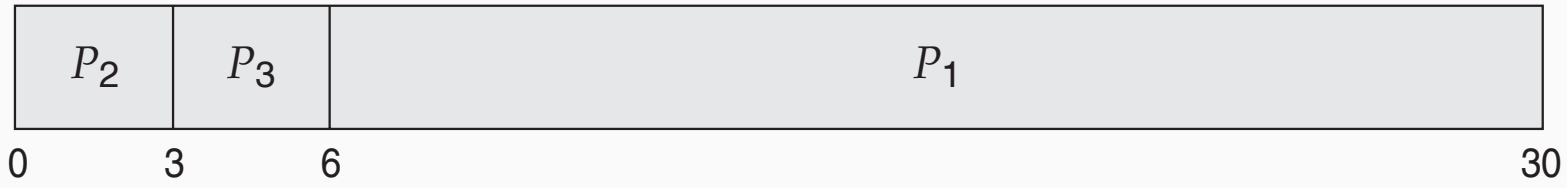


Figure 5:

Planificación SJF

Otro método de planificación de la CPU es el algoritmo de planificación con selección del trabajo más corto (SJF, shortest-job-first).

- Este algoritmo asocia con cada proceso la duración de la siguiente ráfaga de CPU del proceso.
- Cuando la CPU está disponible, se asigna al proceso que tiene la siguiente ráfaga de CPU más corta.
- Si las siguientes ráfagas de CPU de los procesos son iguales, se usa la planificación FCFS para romper el empate.

Planificación SJF

Se debe observar que un término más apropiado para este método de planificación sería el de algoritmo de la siguiente ráfaga de CPU más corta, ya que la planificación depende de la duración de la siguiente ráfaga de CPU de un proceso, en lugar de depender de su duración total.

- Por ejemplo, considere los siguientes procesos y sus ráfagas de CPU:

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Figure 6:

Planificación SJF

La ejecución bajo este algoritmo sería de la siguiente forma:

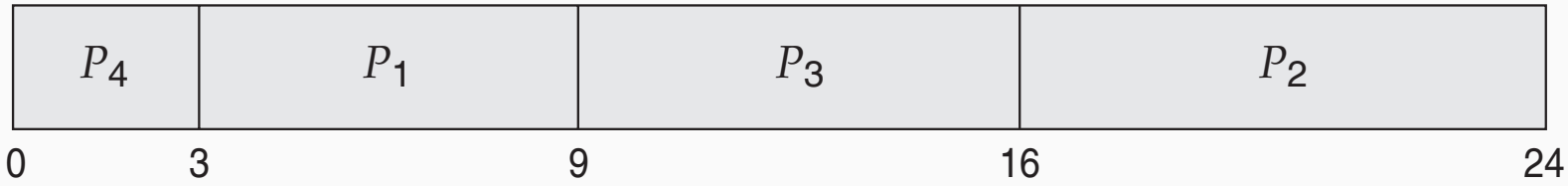


Figure 7:

Planificación SJF

El algoritmo de planificación SJF es probablemente óptimo, en el sentido de que proporciona el tiempo promedio de espera mínimo para un conjunto de procesos dados.

- Anteponer un proceso corto a uno largo disminuye el tiempo de espera del proceso corto en mayor medida de lo que incrementa el tiempo de espera del proceso largo.
- Consecuentemente, el tiempo medio de espera disminuye.

Planificación SJF

Un riesgo que existe en SJF es la posibilidad de inanición (espera indefinida) para los proceso largos mientras exista un flujo continuo de procesos cortos.

- Por otro lado, aunque SJF reduce el sesgo favorable a los procesos largos, no es conveniente para entornos de tiempo compartido o de procesamiento de transacciones, debido a la ausencia de apropiación.

Estimación de tiempos

Aunque el algoritmo SJF es óptimo, no se puede implementar en el nivel de planificación de la CPU a corto plazo, ya que no hay forma de conocer la duración de la siguiente ráfaga de CPU.

- Un método consiste en intentar aproximar la planificación SJF: se puede no conocer la duración de la siguiente ráfaga de CPU, pero se puede predecir su valor, por el procedimiento de confiar en que la siguiente ráfaga de CPU será similar en duración a las anteriores.

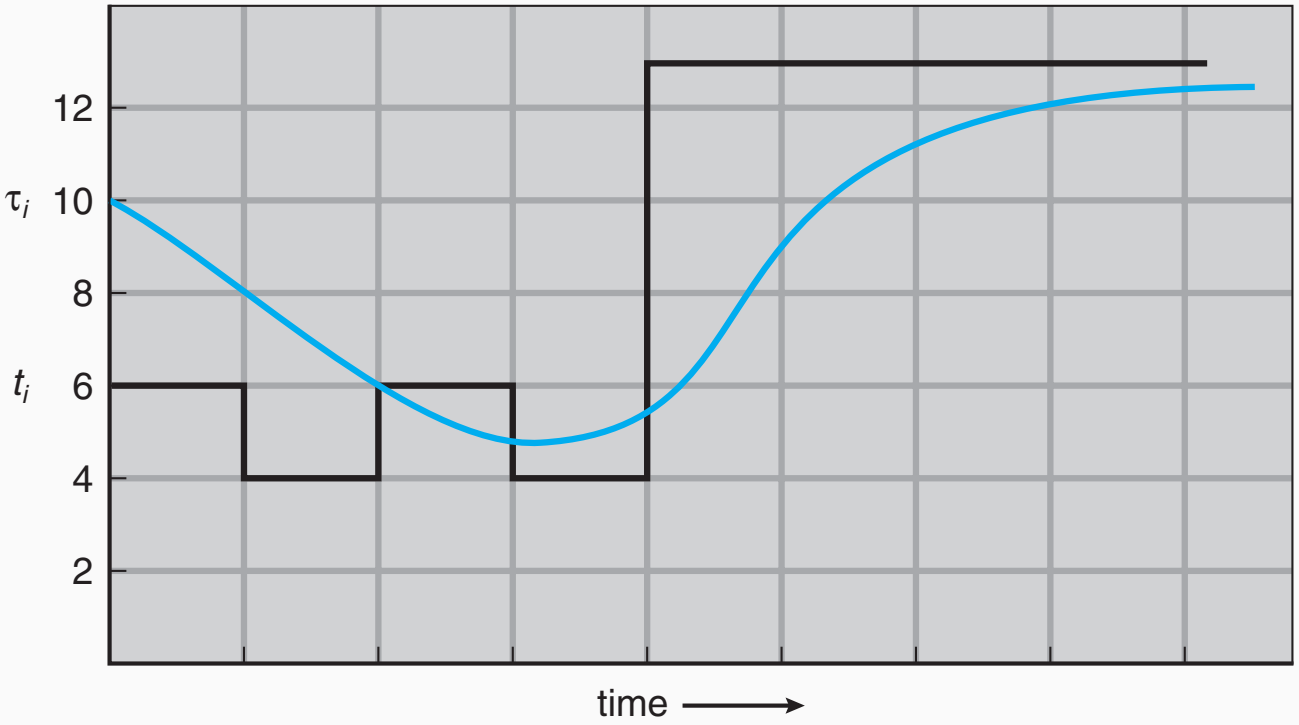
Estimación de tiempos

Por ello, una técnica habitual de predicción de los valores futuros a partir de una serie de valores pasados es usar un promedio exponencial:

```
Tn+1 = a*tn + (1-a)*Tn
```

- Donde *Tn+1* es la predicción de la duración de la próxima ráfaga y *tn* es la duración real de la ráfaga actual.
- Empleando un valor constante de a ($0<a<1$), independiente del número de observaciones pasadas, se llega una situación en la que se tienen en cuenta todos los valores pasados, pero los más distantes reciben un peso menor.
- Frecuentemente, $a=1/2$, en cuyo caso, el historial reciente y pasado tienen el mismo peso.

Estimación de tiempos



CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Figure 6.3 Prediction of the length of the next CPU burst.

Figure 8:

Planificación SRT

La política de menor tiempo restante (SRT, shortest-remaining-time) es una versión apropiativa del SJF, en la que el planificador elige el proceso que le queden menos tiempo esperando en ejecución.

- Considere ahora los siguientes procesos y sus ráfagas de CPU:

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Figure 9:

Planificación SRT

La ejecución bajo este algoritmo sería de la siguiente forma:

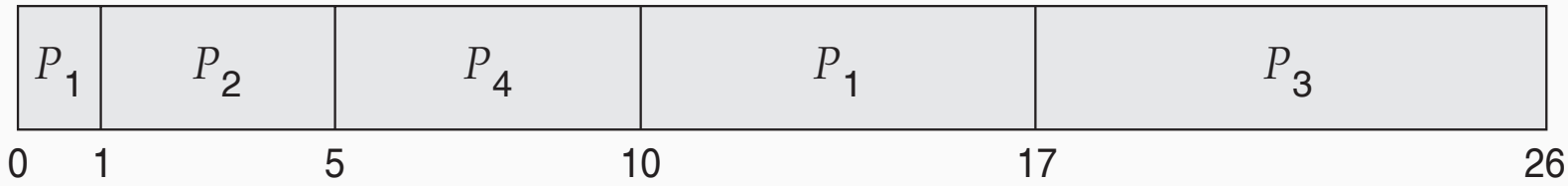


Figure 10:

Planificación SRT

Cuando se añade un nuevo proceso a la cola de listos, puede quedarle un tiempo esperado de ejecución menor que al proceso que está ejecutándose en ese momento.

- Por consiguiente, el planificador puede apropiarse del procesador siempre que un proceso nuevo esté listo.
- Como en el SJF, el planificador debe disponer de una estimación del tiempo del proceso para poder llevar acabo la función de selección, existiendo el riesgo de inanición para procesos largos.

Planificación por prioridades

En este tipo de algoritmo de planificación a cada proceso se le asocia una prioridad y la CPU se asigna al proceso que tenga la prioridad más alta.

- Los procesos con la misma prioridad se planifican en orden FCFS.

Planificación por prioridades

Sin embargo, algunos sistemas usan los números bajos para representar una prioridad baja; otros, emplean números bajos para especificar una prioridad alta; esta diferencia puede llevar a confusión.

- Considere los siguientes procesos y sus ráfagas de CPU:

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

Figure 11:

Planificación por prioridades

La ejecución bajo este algoritmo sería de la siguiente forma:

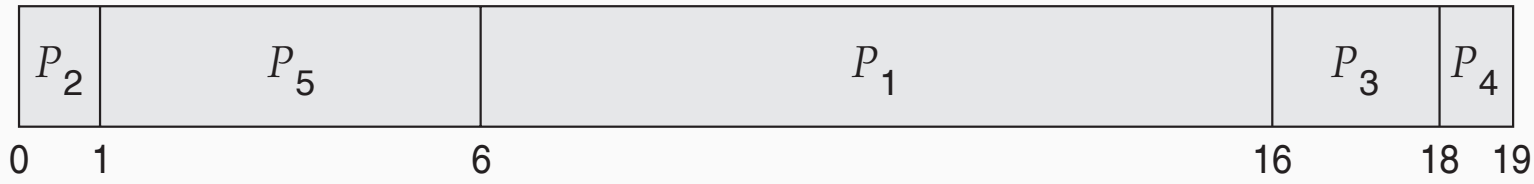


Figure 12:

Planificación por prioridades

Las prioridades pueden definirse interna o externamente.

- Las prioridades definidas internamente utilizan algún valor medible para calcular la prioridad de un proceso.
- Por ejemplo, para calcular las prioridades se han usado en diversos sistemas magnitudes tales como los límites de tiempo, los requisitos de memoria, el número de archivos abiertos y la relación entre la ráfaga de E/S promedio y la ráfaga de CPU promedio.
- Las prioridades definidas externamente se establecen en función de criterios externos al sistema operativo, como por ejemplo la importancia del proceso, el costo monetario del uso de la computadora, el departamento que patrocina el trabajo y otros factores, a menudo de carácter político.

Planificación por prioridades

La planificación por prioridades puede ser apropiativa o cooperativa. Cuando un proceso llega a la cola de procesos preparados, su prioridad se compara con la prioridad del proceso actualmente en ejecución.

- Un algoritmo de planificación por prioridades apropiativo, expulsará de la CPU al proceso actual si la prioridad del proceso que acaba de llegar es mayor.
- Un algoritmo de planificación por prioridades cooperativo simplemente pondrá el nuevo proceso al principio de la cola de procesos preparados.

Planificación por prioridades

Un problema importante de los algoritmos de planificación por prioridades es el bloqueo indefinido o la muerte por inanición.

- Un proceso que está preparado para ejecutarse pero está esperando acceder a la CPU puede considerarse bloqueado; un algoritmo de planificación por prioridades puede dejar algunos procesos de baja prioridad esperando indefinidamente.
- En un sistema computacional con una carga de trabajo grande, un flujo estable de procesos de alta prioridad puede impedir que un proceso de baja prioridad llegue a la CPU.

Planificación por prioridades

Una solución al problema de bloqueo indefinido de los procesos de baja prioridad consiste en aplicar mecanismos de envejecimiento.

- Esta técnica consiste en aumentar gradualmente la prioridad de los procesos que estén esperando en el sistema durante mucho tiempo.

Planificación RR

El algoritmo de planificación por turnos circular (Round-Robin) se diseñó especialmente para los sistemas de tiempo compartido y es similar a la planificación FCFS, pero con la adición de expropiación para conmutar entre procesos.

- Se define una unidad de tiempo pequeña, llamada cuanto de tiempo o porción de tiempo, que generalmente es de 10 a 100 milisegundos. La cola de procesos listos se trata como una cola circular.
- El planificador de la CPU recorre la cola de procesos listos, asignando la CPU a cada proceso durante un intervalo de tiempo de hasta un cuanto.

Planificación RR

Para implementar la planificación RR, mantenemos la cola de procesos listos como estructura FIFO.

- Los procesos nuevos se añaden al final de la cola. El planificador de la CPU escoge el primer proceso de la cola, ajusta un temporizador de modo que interrumpa después de un cuanto de tiempo, y despacha el proceso.
- Ahora, considere los siguientes procesos y sus ráfagas de CPU:

Process	Burst Time
P_1	24
P_2	3
P_3	3

Figure 13:

Planificación RR

La ejecución bajo este algoritmo sería de la siguiente forma:

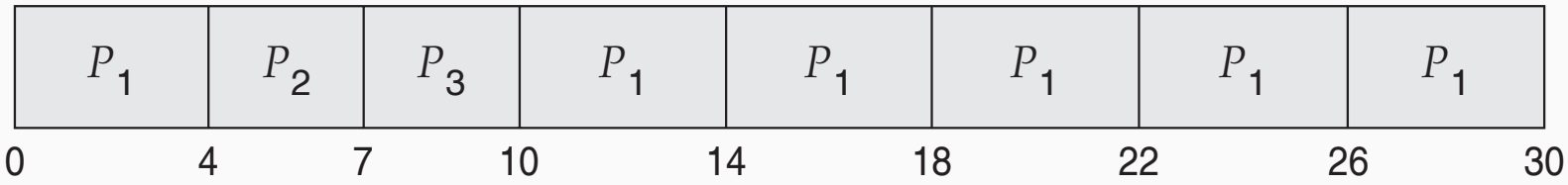


Figure 14:

Planificación RR

Puede ocurrir una de dos cosas.

- El proceso puede tener una ráfaga de CPU cuya duración sea menor que un cuanto de tiempo; en este caso, el propio proceso liberará voluntariamente la CPU. El planificador continuará entonces con el siguiente proceso de la cola de procesos listos.
- En caso contrario, si la ráfaga de CPU del proceso actualmente en ejecución tiene una duración mayor de un cuanto de tiempo, se producirá un fin de cuenta del temporizador y éste enviará una interrupción al sistema operativo; entonces se ejecutará un cambio de contexto y el proceso se colocará al final de la cola de procesos preparados. El planificador de la CPU seleccionará a continuación el siguiente proceso de la cola de procesos listos.

Planificación RR

El tiempo promedio de espera en los sistemas por turnos es, con frecuencia, largo.

- En el algoritmo de planificación por turnos, a ningún proceso se le asigna la CPU por más de un cuanto de tiempo en cada turno (a menos que sea el único proceso ejecutable).
- Si una ráfaga de CPU de un proceso excede un cuanto de tiempo, dicho proceso se desaloja y se coloca de nuevo en la cola de procesos listos. El algoritmo de planificación por turnos incluye, por tanto, un mecanismo de desalojo.

Elección del quantum

El rendimiento del algoritmo de planificación por turnos depende enormemente del tamaño del cuanto de tiempo.

- Por un lado, si el cuanto de tiempo es extremadamente largo, la planificación por turnos es igual a la FCFS.
- Si el cuanto de tiempo es relativamente pequeño por ejemplo, un milisegundo, el método por turnos se denomina compartición del procesador y en teoría crea la apariencia de que cada uno de los N procesos tiene su propio procesador ejecutándose a 1/n de la velocidad del procesador real.

Elección del quantum

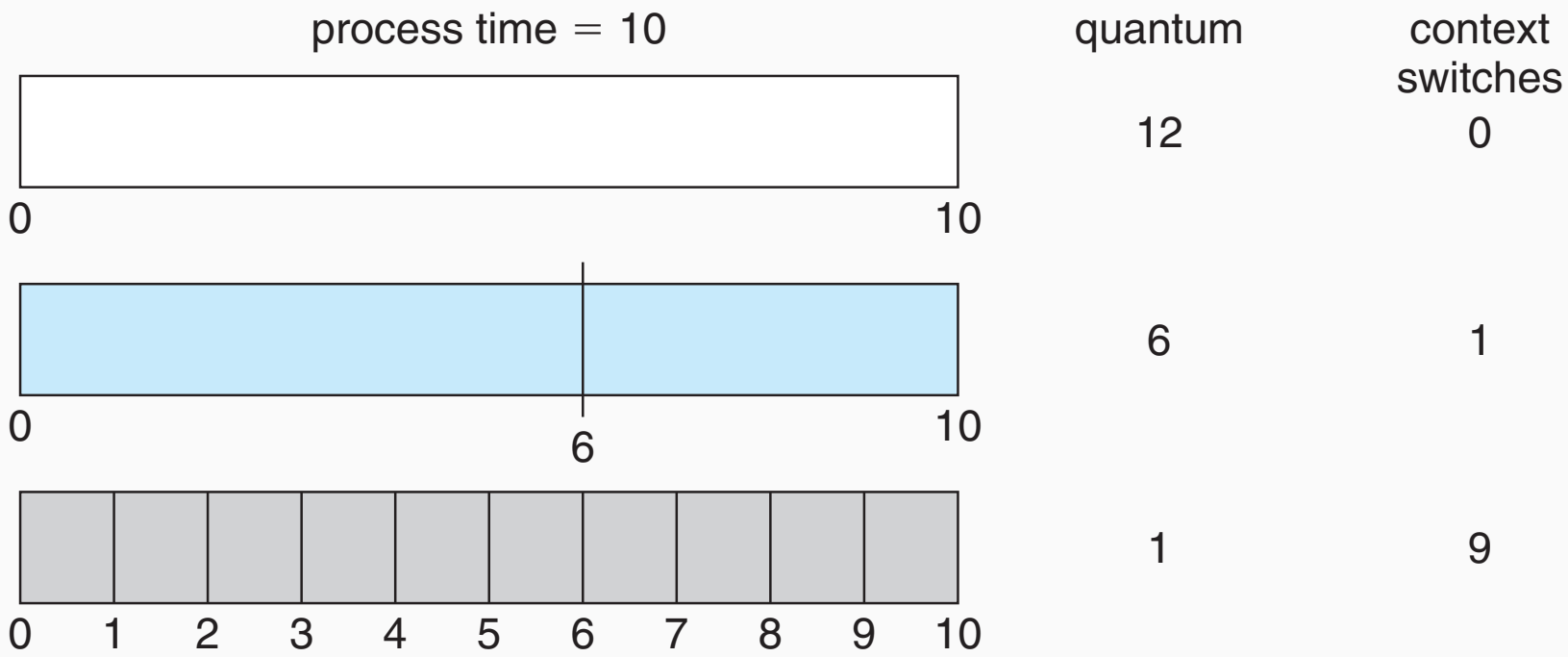


Figure 6.4 How a smaller time quantum increases context switches.

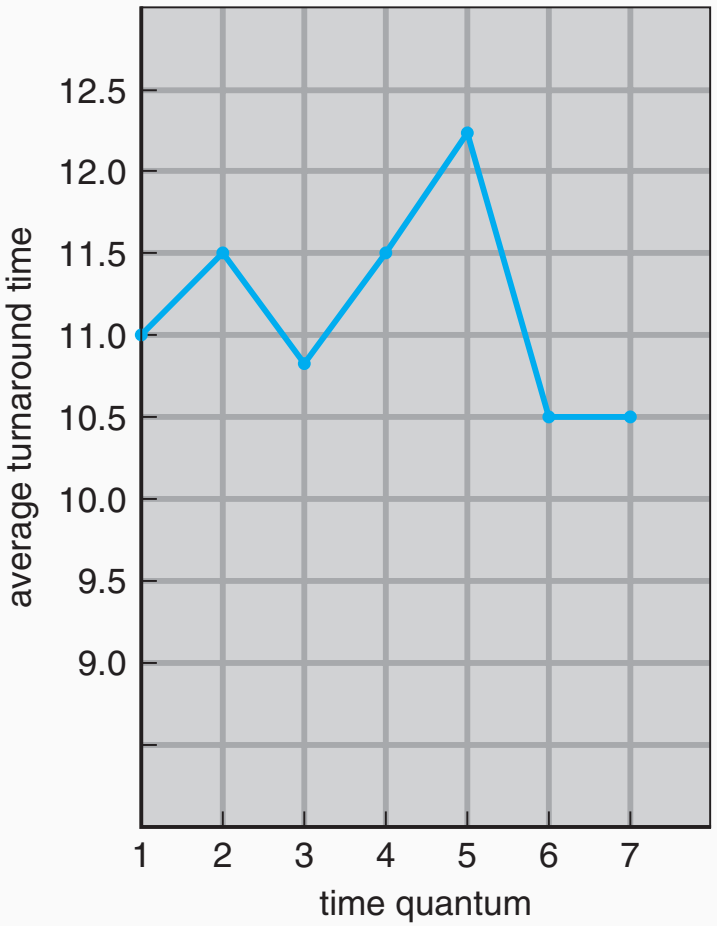
Figure 15:

Elección del quantum

Sin embargo, conviene que cuanto de tiempo sea grande con respecto al tiempo requerido por un cambio de contexto.

- Si el tiempo de cambio de contexto es, aproximadamente, el 10% del cuanto de tiempo, entonces un 10% del tiempo se invertirá en cambio de contexto.
- En la práctica, los sistemas modernos emplean un cuanto de tiempo en el rango de 10 a 100 milisegundos y el tiempo requerido para un cambio de contexto es, normalmente, menor de 10 microsegundos; por tanto, el tiempo de cambio de contexto es una fracción pequeña del cuanto de tiempo.

Elección del quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7

Figure 6.5 How turnaround time varies with the time quantum.

Figure 16:

Planificación con colas multinivel

Otra clase de algoritmo de planificación es la que se ha desarrollado para que las situaciones en que los procesos pueden clasificarse fácilmente en grupos diferentes.

- Por ejemplo, una clasificación habitual consiste en diferenciar entre progresos de primer nivel interactivos y procesos de segundo plano, por lotes.
- Estos dos tipos de procesos tienen requisitos diferentes de tiempo de respuesta y, por tanto, pueden tener distintas necesidades de planificación.
- Además, los procesos de primer plano pueden tener prioridad sobre los poderes de segundo plano.

Planificación con colas multinivel

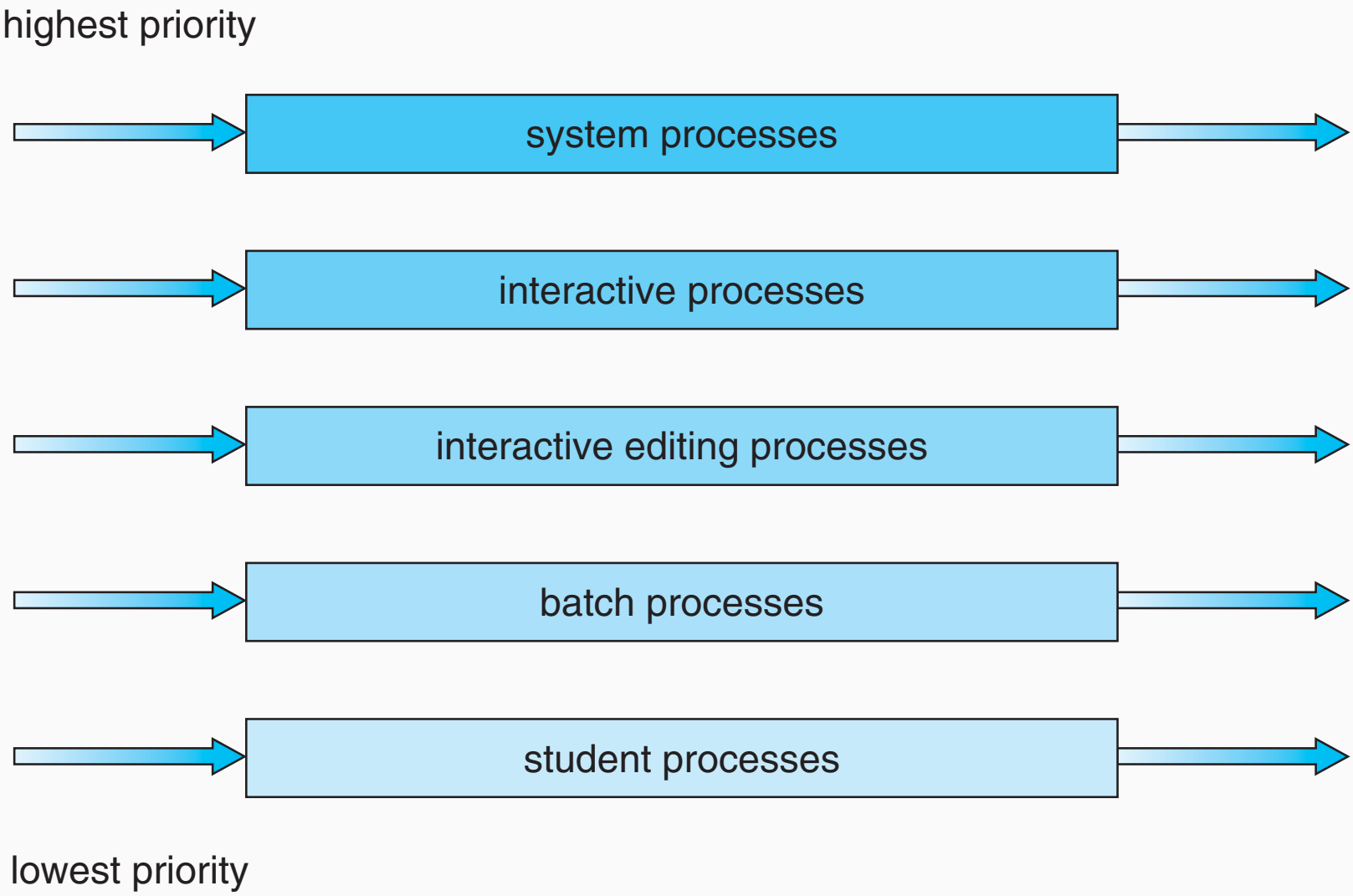


Figure 6.6 Multilevel queue scheduling

Planificación con colas multinivel

Un algoritmo de planificación mediante colas multinivel divide la cola de procesos preparados en varias colas distintas.

- Los procesos se asignan permanentemente a una cola, generalmente en función de alguna propiedad del proceso, como por ejemplo el tamaño de memoria, la prioridad del proceso o el tipo de proceso.
- Cada cola tiene su propio algoritmo de planificación.
- Por ejemplo, pueden emplearse colas distintas para los procesos de primer y de segundo plano.
- La cola de primer plano puede planificarse mediante un algoritmo por turnos, mientras que para la cola de segundo plano puede emplearse un algoritmo FCFS.

Planificación con colas multinivel

Además, debe definirse una planificación entre las colas, la cual suele implementarse como una planificación apropiativa y prioridad fija.

- Por ejemplo, la cola de procesos de primer plano puede tener prioridad absoluta sobre la cola de procesos de segundo plano.
- Otra posibilidad consiste en repartir el tiempo entre las colas. En este caso, cada cola tiene una cierta porción del tiempo de CPU, con la que puede entonces planificar sus distintos procesos.

Planificación con colas multinivel realimentadas

El algoritmo de planificación mediante colas multinivel realimentadas permite mover un proceso de una cola a otra.

- La idea es separar los procesos en función de las características de sus ráfagas de CPU.
- Si un proceso utiliza demasiado tiempo de CPU, se pasa a una cola de prioridad más baja.
- Este esquema deja los procesos limitados por E/S y los procesos interactivos en las colas de prioridad más alta.
- Además, un proceso que esté esperando demasiado tiempo en una cola de baja prioridad puede pasarse a una cola de prioridad más alta. Este mecanismo de envejecimiento evita el bloqueo indefinido.

Planificación con colas multinivel realimentadas

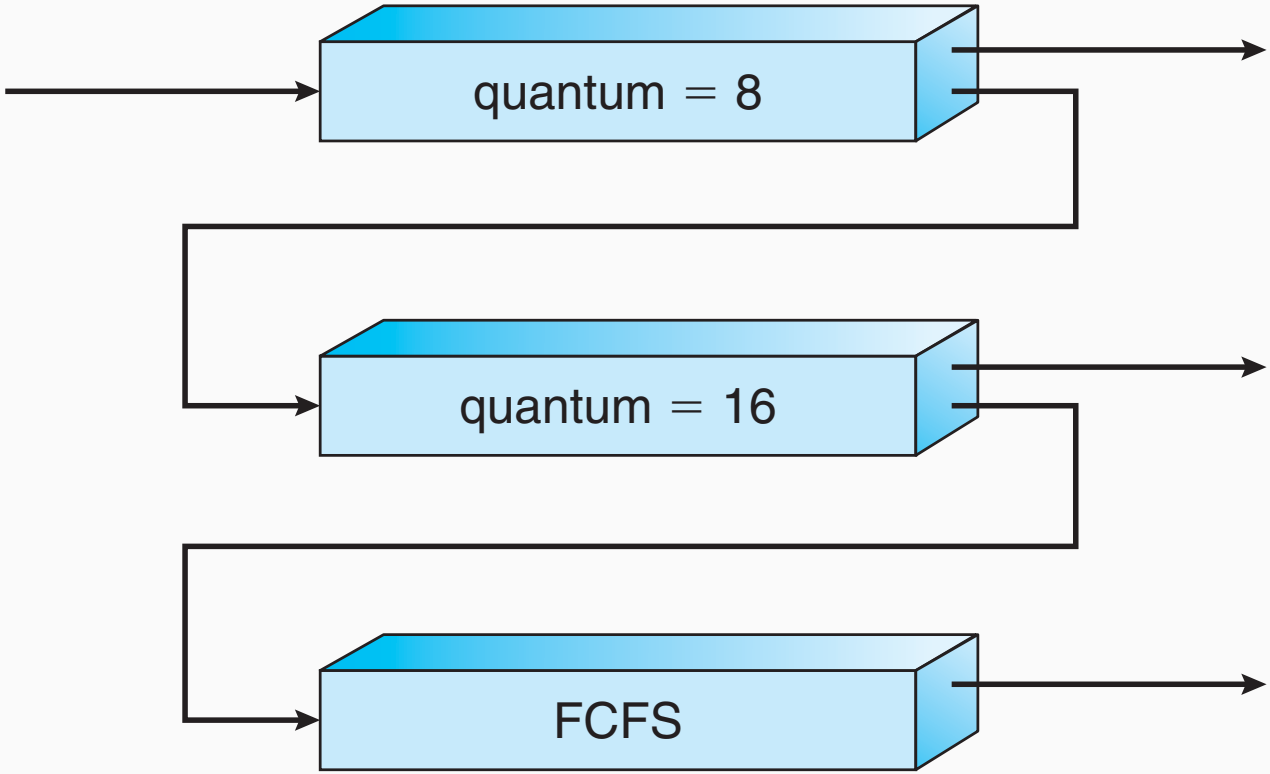


Figure 6.7 Multilevel feedback queues.

Figure 18:

Planificación con colas multinivel realimentadas

Este algoritmo de planificación proporciona la prioridad más alta a todo proceso que tenga una ráfaga de CPU pequeña.

- Tales procesos acceden rápidamente a la CPU, concluyen su ráfaga de CPU y pasan a su siguiente ráfaga de E/S.
- Los procesos que necesitan una ráfaga de tiempo mediana también son servidos rápidamente, aunque con una prioridad más baja que los procesos más cortos.
- Los procesos largos terminan yendo automáticamente a la última cola y se sirven, (posiblemente) siguiendo el orden FCFS, con los ciclos de CPU no utilizados por las colas anteriores

Planificación en multiprocesadores

Si hay varias CPU, el problema de planificación se vuelve más complejo. Un método consiste en que todas las decisiones sobre la planificación, el procesamiento de E/S y otras actividades del sistema sean administradas por un mismo procesador, el servidor maestro.

- Los demás procesadores sólo ejecutan código de usuario. Este multiprocesamiento asimétrico resulta simple, porque sólo hay un procesador que accede a las estructuras de datos del sistema, reduciendo la necesidad de compartir datos.

Planificación en multiprocesadores

El segundo método utiliza el multiprocesamiento simétrico (SMP), en el que cada uno de los procesadores se auto-planifica.

- Todos los procesos puede estar en una cola común de procesos listos, o cada procesador puede tener su propia cola privada de procesos listos.
- Independientemente de esto, la planificación se lleva acabo haciendo que el planificador de cada procesador examine la cola de procesos listos y seleccione un proceso para ejecutar.

Planificación en multiprocesadores

Sin embargo, si se tienen varios procesadores intentando acceder a una estructura de datos común para actualizarla, el planificador tiene que programarse cuidadosamente: se tiene que asegurar que dos procesadores no elegirán el mismo proceso y que no se perderán procesos en la cola.

Afinidad al procesador

Cuando un proceso migra a otro procesador: los contenidos de la memoria caché del procesador de origen deben invalidarse y la caché del procesador de destino debe ser rellenada.

- Debido al alto costo de esto, la mayoría de los sistemas SMP intentan evitar la migración de procesos de un procesador a otro, y en su lugar intentan mantener en ejecución cada proceso en el mismo procesador.
- Esto se conoce con el nombre de *afinidad al procesador*, lo que significa que un proceso tiene una afinidad hacia el procesador en que está ejecutándose actualmente.

Afinidad al procesador

Dependiendo de las características del sistema la afinidad puede ser de dos formas: suave o dura.

- La afinidad suave consiste en que un sistema operativo tiene la política de intentar mantener un proceso en ejecución en el mismo procesador pero no está garantizado que lo haga.
- La afinidad dura permite a un proceso especificar que no debe emigrar a otros procesadores, en este caso el sistema garantiza que la migración no será realizada.

Equilibrado de cargas

Los mecanismos de equilibrado de carga intentan distribuir equitativamente la carga de trabajos entre todos los procesadores de un sistema SMP.

- Normalmente, este mecanismo sólo es necesario en aquellos sistemas en los que cada procesador tiene su propia cola privada de procesos preparados para ejecutarse.

Equilibrado de cargas

Existen dos métodos generales para equilibrar la carga: migración comandada (push migration) y migración solicitada (pull migration).

- Con la migración comandada, una tarea específica comprueba periódicamente la carga en cada procesador y, si encuentra un desequilibrio, distribuye equitativamente la carga moviendo (o cargando procesos) de los procesadores sobrecargados en los menos ocupados o inactivos.

Equilibrado de cargas

La migración solicitada se produce cuando un procesador inactivo extrae de un procesador ocupado alguna tarea que esté en espera.

- Las migraciones comandadas y solicitadas no tienen por qué ser mutuamente excluyentes y, de hecho, a menudo se implementan en paralelo en los sistemas de equilibrado de carga.
- Sin embargo, un problema que se presenta es que el equilibrado de carga a menudo contrarresta los beneficios de la afinidad al procesador.