Planificación de procesos en monoprocesadores

Indice

Necesidad de la planificación

En épocas pasadas de los sistemas de procesamiento por lotes (batch), la idea que existía sobre la planificación era bastante simple y consistía en aplicar un algoritmo secuencial. Esto producía un desaprovechamiento muy importante de las capacidades del procesador ya que la ejecución de un proceso alternaba entre dos estados de ejecución: utilizando la CPU o esperando a que se realice una operación de E/S, por lo que mientras se trabajaba con un dispositivo, el procesador se encontraba inactivo.

Más tarde, surgieron los sistemas multiprogramados, en donde se intentó maximizar la utilización de la CPU. Esto se pudo conseguir manteniendo varios procesos en la memoria, y cuando un proceso tenía que esperar, el sistema operativo le quitaba la CPU y se lo asignaba a otro proceso que se encontraba en dicha memoria. Por lo tanto, la tarea de la planificación cobró gran importancia por su incidencia directa sobre el rendimiento del sistema, ya que el sistema operativo debía decidir qué proceso esperaría y qué proceso continuaría.

Definición

Podemos definir a la planificación como un conjunto de políticas y mecanismos incorporados al sistema operativo, a través de un módulo denominado planificador, que debe decidir cuál de los procesos en condiciones de ser ejecutado conviene ser despachado primero y qué orden de ejecución debe seguirse. Esto debe realizarse sin perder de vista su principal objetivo que consiste en el máximo aprovechamiento del sistema, lo que implica proveer un buen servicio a los procesos existentes en un momento dado. Un buen servicio podría traducirse en tiempo de respuesta aceptable, productividad y eficiencia del procesador.

Niveles de planificación

La planificación se hace en cuatro instantes de tiempo. De estas cuatro, una no la realiza el sistema operativo, sino que es externa al procesamiento, pero tiene una influencia enorme sobre la definición del procesamiento, dado que el sistema operativo queda determinado por las decisiones que se toman en este nivel. A esta instancia le daremos el nombre de extra largo plazo por ser en la escala de tiempo del ser humano.

En la administración del procesador podemos distinguir tres niveles de planificación de acuerdo a la escala de tiempo en que se realiza la misma. El largo plazo en segundos, mediano plazo en milisegundos y el corto plazo en nanosegundos o microsegundos.

Planificación a extra largo plazo

Consiste en una planificación externa que se hace en el centro de cómputos y está estrechamente ligada a las políticas de funcionamiento del sistema, ya que se determina la importancia relativa de los usuarios. A través de procedimientos escritos se fijan las reglas que se aplicarán a los usuarios relativos al uso, seguridad, accesos, prioridades, etc, así como también las reglas en cuanto a modalidad de procesamiento, la operación, la política de backup, etc.

Esta planificación busca satisfacer cuatro objetivos desde el punto de vista de los usuarios:

- Mayor velocidad de respuesta en sus trabajos con lo que disminuye el tiempo de espera de los usuarios.
- Existencia y disponibilidad de recursos que necesitan para ejecutar sus trabajos.

- Importancia de sus tareas.
- Seguridad de que sus trabajos sean completados correctamente.

Por lo tanto, es responsabilidad del profesional de sistemas brindar un adecuado servicio de procesamiento de datos como también ocuparse del orgware y del peopleware para que todo funcione dentro de lo establecido.

Planificación a largo plazo

El planificador a largo plazo, scheduler o planificador de trabajos, es un administrador que se encarga de organizar la ejecución con un adecuado planeamiento de recursos para que el trabajo se ejecute ordenadamente y eficientemente según la modalidad de procesamiento.

El sheduler se ejecuta con poca frecuencia, sólo cuando se necesita crear un nuevo proceso en el sistema, cuando termina un proceso, o ingresa un usuario en el sistema, por lo que tiene prioridad máxima para ejecutar. Es el responsable de controlar el nivel de multiprogramación del sistema y el balance de carga del sistema. Esto último implica la selección cuidadosa de trabajos para mantener un adecuado balance de carga de procesos que hacen uso de E/S intensivo (I/O bound) o uso de CPU intensivo (CPU bound).

Procedamos a describir un poco su accionar ante un nuevo trabajo. Un software del sistema operativo, llamado monitor, recibe al nuevo trabajo y lo carga en la memoria central. Después de haber sido recibido el trabajo, el sheduler se encarga de preparar y crear procesos con sus respectivos bloques de control del proceso (PCB) para poder ejecutarlos. Si los recursos que solicita estuvieran disponibles, se le asignan y se lo ingresa a la cola de listos para ejecutar.

Existen diferentes filosofías en el procesamiento de un trabajo. Todas ellas responden a ciertos criterios de planificación que se vuelcan en los respectivos algoritmos de planificación. Esto se conoce como la modalidad de ejecución o procesamiento. Los más importantes son:

  - Batch: Apunta estrictamente al exhaustivo uso del procesador en detrimento del usuario. Sus principales caracterísiticas son:
  - La CPU es monoprogramada.
  - No existe diferencia entre trabajo y proceso.
  - El scheduler elige el trabajo, crea el proceso y lo ejecuta.
  - Prácticamente hay un solo nivel de planificación.
- Interactivo: Apunta al servicio del usuario en detrimento de la performance del procesador. Es multiprogramado pues se multiplexa la CPU entre varios programas.
- Multiprocesado: Es un ambiente en el que existen varios procesadores para servir a los procesos en ejecución.
- Procesamiento distribuido o en red: Es una forma de procesamiento en que se le presenta al usuario una máquina virtual y en que el procesamiento se realiza en distintas máquinas diseminadas geográficamente y conectadas por una red.

  En conclusión y siendo un poco más precisos, podríamos decir que las tareas que involucra este nivel de planificación son:

  - Mantener un registro de estado de todos los trabajos en el sistema (JBC).
  - Establecer una estrategia para el pasaje entre las colas de suspendidos y la de listos.
  - Asignar recursos (memoria central, dispositivos, procesadores, etc.) a cada trabajo.
  - Pedir (recuperar) los recursos cuando los trabajos se han completado.
  - Detectar y prevenir los conflictos de abrazo mortal o deadlock.
  - Dar entrada a nuevos trabajos.
  - Asignar prioridades a los procesos. Esto genera el orden de ejecución y viene determinado

básicamente por el orden de procesos en la cola de listos, o sea, el orden en el que el dispatcher los seleccionará de esta cola para ponerlos en ejecución (generalmente el primero de la cola).
    - ♦ Implementar las políticas de asignación de recursos, razón por la que se le otorga la máxima prioridad en el sistema para que el dispatcher lo seleccione primero si está libre el procesador y se ejecuta cuando:
- Se pide o libera un recurso.
- Cuando termina un proceso.
- Cuando llega un nuevo trabajo al pool de procesos (lo ubica en la ready queue)
- Cuando llega un nuevo usuario al sistema.

Planificación a mediano plazo

Es el que decide sacar de memoria central y llevar a disco (swap−out) a aquellos procesos inactivos o a los activos cuyos estados sean bloqueado momentáneamente o temporalmente o los suspendidos y luego, cuando desaparezcan las causas de sus bloqueos, traerlos nuevamente a memoria (swap−in) para continuar su ejecución. Este tipo de planificador se encuentra solo en algunos sistemas especialmente en los de tiempo compartido, ya que permite mantener un equilibrio entre los procesos activos e inactivos.

Este planificador puede ser invocado cuando quede espacio libre de memoria por efecto de la terminación de un proceso o cuando el suministro de procesos caiga por debajo de un límite especificado.

En algunos casos suplanta al planificador de largo plazo y otros lo complementa: Por ejemplo en sistemas de tiempo compartido, el long−term scheduler puede admitir más usuarios de los que pueden caber realmente en memoria. Sin embargo, como los trabajos de estos sistemas están caracterizados por ciclos de actividad y ciclos de ociosidad, mientras el usuario piensa algunos procesos pueden ser almacenados y al recibir respuesta vueltos a poner en la cola de listos.

Este tipo de planificación solo es usado en sistemas con mucha carga de procesos, ya que el procedimiento de swapping produce mucho overhead, haciendo bajar considerablemente el desempeño general.

Planificación a corto plazo

También llamado short−term scheduler o low scheduler, es el responsable de decidir quién, cuándo, cómo y por cuánto tiempo recibe el procesador un proceso que está preparado (ready queue) para ejecutar (los recursos a esta altura ya deben estar todos disponibles para este trabajo). Además en sistemas operativos con esquemas expropiativos (se quita el recurso procesador al proceso) verifica las interrupciones.

El planificador a corto plazo es invocado cada vez que un suceso (interno o externo) hace que se modifique el estado global del sistema. Por ejemplo:

- ♦ Tics de reloj (interrupciones basadas en el tiempo).
- ♦ Interrupciones y terminaciones de E/S.
- ♦ La mayoría de las llamadas operacionales al sistema operativo (en oposición a las llamadas de consulta).
- ♦ El envío y recepción de señales.
- ♦ La activación de programas interactivos.

El low scheduler debe ser rápido y con poca carga para el procesador para que se mantenga el

rendimiento, ya que se le debe sumar además el tiempo que toma el cambio de contexto. El cambio de contexto o context switch consiste en la conmutación de la CPU entre un proceso y otro y es overhead puro, por lo tanto debe ser lo más rápido posible. Algunos valores típicos oscilan entre 1 y 100 seg que se conoce como dispatch latency.

El context switch involucra:

- ♦ Preservar el estado del viejo proceso (guardar en el stack su PCB).
- ♦ Recuperar el estado del nuevo proceso (recuperar su PCB).
- ♦ Bifurcar a la dirección donde había sido suspendido el nuevo proceso.

El proceso nulo o vacío

Un problema que debe resolver un sistema operativo multitarea es, qué debería hacer el sistema cuando no hay nada que ejecutar. Por ejemplo cuando la cola de listos se encuentra vacía.

Este problema es resuelto en muchos sistemas operativos con el proceso NULO que es creado por el sistema en el momento de arranque. El proceso nulo nunca termina, no tiene E/S y tiene la prioridad más baja en el sistema. En consecuencia la cola de listos nunca está vacía, además la ejecución del planificador puede hacerse más rápida al eliminar la necesidad de comprobar si la cola de listos está vacía o no. Algunas de las tareas que se le pueden dar al proceso nulo, por ejemplo, es realizar estadísticas de uso de procesador, o asistencia de vigilancia de la integridad del sistema, etc.

Algoritmos de planificación

El planificador es el módulo del sistema operativo que decide qué proceso se debe ejecutar, para ello usa un algoritmo de planificación que debe cumplir con los siguientes objetivos:

- Imparcialidad.
- Política justa.
- Eficiencia: mantener la CPU ocupada en lo posible el mayor tiempo con procesos de usuario.
- Minimizar el tiempo de espera de usuarios.
- Maximizar el número de procesos ejecutados. (Rendimiento: trabajos que se procesan por hora).
- Tiempo de respuesta excelente (por ejemplo: minimizar el tiempo de respuesta para los usuarios interactivos).
- Predecibilidad en la ejecución.
- Equilibrio en el uso de los recursos.

Antes de comenzar a describir los respectivos algoritmos de planificación, es importante conocer dos conceptos relacionados. Uno de ellos es la función de selección que determina qué proceso, de entre los listos, se elige para ejecutar a continuación. El otro es el modo de decisión o esquema de planificación, que especifica los instantes de tiempo en que se aplica la función de selección. Hay dos categorías generales:

- **Nonpreemptive scheduling** (apropiativo) También conocido como cooperative multitasking. Una vez que el proceso pasa al estado de ejecución, continúa ejecutando hasta que termina, se bloquean en espera de una E/S o al solicitar algún servicio del sistema. Esta política de ejecución para terminación fue implementada en los primeros sistemas de lote (batch).
- **Preemptive scheduling** (no apropiativo) Generalmente conocida como política de planificación por torneo. El proceso que se está ejecutando actualmente puede ser interrumpido y pasado al estado de listos por el sistema operativo. La decisión de sustituirlos por otro proceso puede llevarse a cabo cuando llega un nuevo proceso, cuando se produce una interrupción que lleva a un proceso bloqueado al estado listo o periódicamente, en función de una interrupción del reloj.

Política vs. Mecanismo

A veces ocurre que un proceso tiene muchos hijos ejecutándose bajo su control y es completamente posible que el proceso principal tenga una idea excelente de cuáles de sus hijos son los más importantes (o críticos respecto al tiempo), y cuáles los menos. Por desgracia, ninguno de los planificadores analizados hasta ahora acepta datos de los procesos del usuario relativos a decisiones de planificación. Como resultado, el planificador pocas veces hace la mejor elección.

La solución a este problema es separar el mecanismo de planificación de la política de planificación. Lo que esto quiere decir es que el algoritmo de planificación queda parametrizado de alguna manera, pero los parámetros pueden ser determinados por medio de procesos del usuario.

Supongamos que el kernel utiliza un algoritmo de planificación, pero que proporciona una llamada al sistema por medio de la cual un proceso puede establecer (y modificar) la prioridad de sus hijos. De esta forma, el padre puede controlar en detalle la forma de planificar sus hijos, incluso aunque él mismo no realice la planificación. En este caso, el mecanismo está en el kernel pero la política queda establecida por el proceso del usuario.

Criterios de la planificación a corto plazo

Generalmente, se fija un conjunto de criterios con los que evaluar las diversas estrategias de planificación. El criterio más empleado establece dos clasificaciones. En primer lugar, se puede hacer una distinción entre los criterios orientados al usuario y los orientados al sistema. Los criterios orientados al usuario se refieren al comportamiento del sistema tal y como lo perciben los usuarios o los procesos individuales. Los criterios orientados al sistema se centran en el uso efectivo y eficiente del procesador.

Otra forma de clasificación es considerar los criterios relativos al rendimiento del sistema y los que no lo son. Los criterios relativos al rendimiento son cuantitativos y, en general, pueden evaluarse fácilmente. Los criterios no relativos al rendimiento son , en cambio, cualitativos y no pueden ser evaluados o analizados fácilmente.

Todos estos criterios son dependientes entre sí y es imposible optimizar todos ellos de forma simultánea.

CRITERIOS ORIENTADOS AL USUARIO, CRITERIOS DE RENDIMIENTO

Tiempo de retorno Es el intervalo de tiempo transcurrido entre el lanzamiento de un proceso y su finalización. Es la suma del tiempo de ejecución real y el tiempo consumido en la espera de los recursos, incluido el procesador. Esta es una medida apropiada para trabajos por lotes.

Tiempo de respuesta Para un proceso interactivo, es el intervalo de tiempo transcurrido desde que se emite una solicitud hasta que se empieza a recibir la respuesta. A menudo, un proceso empieza a generar alguna salida para el usuario mientras que continúa procesando la solicitud.

Plazos Cuando se pueden especificar plazos de terminación de un proceso, la disciplina de planificación debe subordinar otras metas a la maximización del porcentaje de plazos cumplidos.

CRITERIOS ORIENTADOS AL USUARIO, OTROS CRITERIOS

Previsibilidad Un determinado trabajo se debe ejecutar aproximadamente en el mismo tiempo y con el mismo coste sin importar la carga del sistema.

CRITERIOS ORIENTADOS AL SISTEMA, CRITERIOS RELATIVOS AL RENDIMIENTO

Productividad La política de planificación debe intentar maximizar el número de procesos terminados por unidad de tiempo. Depende de la longitud media de cada proceso, pero también está influida por la política de planificación, que puede influir en el uso del procesador.

Utilización del procesador Es el porcentaje de tiempo en el que el procesador está ocupado.

CRITERIOS ORIENTADOS AL SISTEMA, OTROS CRITERIOS

Equidad Los procesos deben ser tratados de igual forma y ningún proceso debe sufrir inanición.

Prioridades Cuando se asignan prioridades a los procesos, la política de planificación debe favorecer a los de mayor prioridad.

Equilibrio de recursos La política de planificación debe mantener ocupados los recursos del sistema. Se debe favorecer a los procesos que no utilicen recursos sobrecargados. Este criterio también afecta a la planificación a medio y largo plazo.

Uso de prioridades

(Algoritmo apropiativo) En vez de una simple cola de listos, se ofrece un conjunto de colas en orden de prioridad descendente. Cuando se vaya a realizar una selección de planificación, el planificador comenzará por la cola de listos de mayor prioridad. Si hay uno o más procesos en esta cola, se selecciona uno mediante alguna política de planificación. Si la cola de mayor prioridad está vacía, se examina la cola siguiente y así sucesivamente.

Las prioridades pueden ser:

Internas o dinámicas: modificables por el sistema operativo en ejecución mediante uno o más parámetros medibles.

Externas o estáticas: puestas arbitrariamente por el centro de cómputos de acuerdo a factores externos al sistema.

Un problema de los esquemas puros de planificación por prioridades es que los procesos de prioridad más baja pueden sufrir inanición (starvation). Este problema ocurre si siempre hay un flujo continuo de procesos listos de alta prioridad. Para superar este problema, la prioridad de un proceso puede cambiar en función de su edad o su historial de ejecución (aging).

Primero en llegar, primero en ser servido (FCFS First come first served)

(Algoritmo apropiativo) Con mucha diferencia, es el algoritmo de planificación más sencillo. Esto es, el primer proceso en solicitar la CPU es el primero en recibir la asignación de la misma. La implementación del FCFS se realiza fácilmente mediante una cola FIFO. Cuando un proceso entra en la cola de preparados o listos para la ejecución (ready queue), su PCB se enlaza al final de la cola.

Cuando la CPU queda libre, ésta se le asigna al proceso situado al principio de la cola. Entonces el proceso en ejecución se elimina de la cola. El código para la planificación FCFS es sencillo de escribir y de comprender.

FCFS rinde mucho mejor con procesos largos que con procesos cortos.

Sin embargo, las prestaciones del FCFS son , con frecuencia, bastante pobres.

Los problemas que presenta son:

- El tiempo medio de espera suele ser elevado.
- Bajo nivel de utilización de la CPU.
- Pobre tiempo de respuesta en procesos cortos en esquemas con mucha carga.
- Tiende a favorecer a los procesos con carga de CPU frente a los que tienen carga de E/S.
- Uso ineficiente de los dispositivos de E/S.

Turno rotatorio (RR Round robin)

(Algoritmo no apropiativo) El algoritmo de planificación round−robin fue especialmente diseñado para sistemas en tiempo compartido. Se define una pequeña unidad de tiempo común llamada quantum de tiempo o time slice, que generalmente tiene un valor entre 10 y 100 milisegundos. La cola de listos se trata como una cola circular. El planificador de CPU recorre la cola asignando el procesador a cada proceso durante un intervalo de tiempo de hasta un quantum.

Para implementar la planificación RR, la cola se mantiene como una cola de procesos FIFO. El planificador de la CPU selecciona el primer proceso de la cola, y únicamente puede salir del estado de ejecución por tres motivos: que termine su ejecución, se proceda al llamada a una E/S y el proceso se quede bloqueado o que se genere una interrupción por haber superado un quantum de ejecución del proceso.

Si hay n procesos en la cola y el quantum de tiempo es q, entonces cada proceso obtiene 1/n del tiempo de CPU en fragmentos de al menos q unidades de tiempo cada vez. Cada proceso tiene que esperar no más de (n−1) x q unidades de tiempo hasta su quantum de tiempo siguiente.

El conflicto surge en el momento de decidir la duración del quantum de tiempo para cada proceso. Si el quantum es muy pequeño, produce mucho overhead por la gran cantidad de cambios de contexto de ejecución que hace el sistema operativo. Si por el contrario, el quantum es muy grande produce un tiempo de reacción muy pobre porque los procesos en cola de listos esperan demasiado y si es infinito se convierte en FCFS. Es decir que para que sea eficiente, la duración del context switch debe ser mucho menor que el time slice.

Una desventaja del turno rotatorio es el tratamiento que hace si existe una mezcla de procesos limitados por CPU y procesos limitados por E/S. En este caso, sucedería lo siguiente: un proceso limitado por E/S utiliza el procesador durante un periodo corto y después se bloquea en la E/S; espera a que se complete la operación de E/S y entonces vuelve a la cola de listos. Por otro lado, un proceso limitado por procesador generalmente hace uso de un cuanto de tiempo completo cuando se ejecuta e inmediatamente retorna a la cola de listos. Así pues, los procesos con carga de procesador tienden a recibir una porción desigual de tiempo de procesador, lo que origina un rendimiento pobre de los procesos con carga de E/S, un mal aprovechamiento de los dispositivos de E/S y un incremento de la variabilidad del tiempo de respuesta.

Para solucionar este problema se implementa un algoritmo llamado VRR (virtual round−robin). La nueva característica consiste en una cola FCFS auxiliar a la que se desplazan los procesos una vez que son liberados de la espera por E/S. Al tomar una decisión sobre el siguiente proceso a expedir, los procesos de la cola auxiliar tienen preferencia sobre los de la cola principal de listos. Cuando se expide un proceso desde la cola auxiliar, no se puede ejecutar más que un tiempo igual al cuanto básico menos el tiempo total de ejecución consumido desde que fue seleccionado por última vez en la cola de listos.

Primero el proceso más corto (SPN Shortest process next / SPF Shortest process first)

(Algoritmo apropiativo) Este algoritmo consiste en seleccionar el proceso con menor tiempo esperado de ejecución. La mejora del rendimiento global es significativa en términos de tiempo de respuesta, sin embargo, se incrementa la variabilidad de los tiempos de respuesta, especialmente para procesos largos, reduciendo así la previsibilidad.

Una dificultad que plantea SPN es la necesidad de conocer o estimar el tiempo exigido por cada proceso. Para ello, generalmente se toma el promedio exponencial que permite predecir valores futuros a partir de una serie de valores pasados.

Sn+1 = Tn + (1 −)Sn

Donde:

Ti = Tiempo de ejecución en el procesador para el i−ésimo caso del proceso (tiempo total de ejecución para un trabajo por lotes; tiempo de ráfaga de procesador para trabajos interactivos).

Si = Valor pronosticado para el caso i−ésimo.

= Factor constante de ponderación. (0 <= <= 1) (generalmente se utiliza 0,5)

determina el peso relativo dado a las observaciones más y menos recientes. Utilizando un valor constante de , independiente del número de observaciones pasadas, se llega a una situación en la que se tienen en cuenta todos los valores pasados, pero los más distantes reciben un peso menor. Para verlo con más claridad, consideremos el siguiente desarrollo de la ecuación anterior:

Sn+1 = Tn + (1 −) Tn−1 + + (1 −)1 Tn−i + ... + (1 −)nS1

S1 = Valor pronosticado para el primer caso; no calculado.

La ventaja de emplear un valor cercano a 1 es que la media reflejará rápidamente los cambios repentinos en la cantidad observada. La desventaja es que si se produce un breve aumento en los valores observados y después se vuelve a estabilizar en algún valor medio, el empleo de un valor grande a generará cambios bruscos en la media.

Un riesgo que existe en SPN es la posibilidad de inanición para los procesos largos mientras exista un flujo continuo de procesos más cortos. Por otro lado, aunque SPN reduce el sesgo favorable a los procesos largos, no es conveniente para entornos de tiempo compartido o de procesamiento de transacciones, debido a que es un algoritmo apropiativo.

Otra observación importante es que se emplea una gran pérdida de tiempo para efectuar este cálculo por lo que no se utiliza este algoritmo.

Menor tiempo restante (SRT Shortest remaining time first)

Esta es la versión no apropiativa del SPN, en la que el planificador siempre elige al proceso que le queda menos tiempo esperado de ejecución. Por lo tanto, el planificador debe disponer de una estimación del tiempo de proceso para poder llevar a cabo la función de selección, existiendo el riesgo de inanición para procesos largos.

El algoritmo SRT no presenta el sesgo favorable a los procesos largos del FCFS. Al contrario que el

turno rotatorio, este algoritmo es más eficiente debido a que no se produce overhead muy frecuente debido a que las interrupciones no son producidos por el reloj del sistema. Por el contrario, se deben tener en cuenta los tiempos de servicio transcurridos, lo que contribuye a la sobrecarga. El SRT también debería producir tiempos de retorno mejores que los del SPN, puesto que los trabajos cortos reciben una atención inmediata y preferente a los trabajos largos.

Primero el de mayor tasa de respuesta (HRRN Highest response ratio next)

(Algoritmo apropiativo) Cuando el proceso actual termina o se bloquea, se elige el proceso listo con un mayor valor de R. Donde R es:

$R = (w + s) / s$

R = tasa de respuesta.

w = tiempo consumido esperando al procesador.

s = tiempo de servicio esperado.

La decisión de planificación se basa en una estimación del tiempo de retorno normalizado. Lo que se intenta es reducir al máximo las proporciones de tiempo R.

Este método es atractivo porque tiene en cuenta la edad del proceso. Aunque se favorece a los trabajos más cortos (un denominador menor produce una razón mayor), el envejecimiento sin que haya servicio incrementa el valor de la razón, de forma que los procesos más largos puedan pasar, en competición con los más cortos. El tiempo esperado de servicio debe estimarse antes de emplear la técnica de la mayor tasa de respuesta.

Planificación con colas de múltiples niveles y realimentación

En el caso en que no se pueda determinar el tiempo de ejecución, puede utilizarse este algoritmo.

La planificación es de tipo no apropiativo por quantum de tiempo y un mecanismo de prioridades dinámico. Cuando llega un proceso nuevo, este es colocado en la cola de mayor prioridad, luego de su primer ejecución éste es colocado en la cola de prioridad siguiente menor y así sucesivamente hasta que llega hasta la última cola en donde se la vuelve a colocar en la misma nuevamente.

Dentro de cada cola se utiliza el algoritmo de planificación FCFS, en el caso de la última se utiliza el algoritmo round robin.

En este algoritmo puede llegar a ocurrir starvation en el caso de que entren frecuentemente procesos nuevos, debido a que al tener mayor prioridad, no llega a ejecutarse los procesos de las últimas colas. Para evitar esto, se utiliza un mecanismo de variación del quantum de tiempo de ejecución de los procesos de acuerdo a la cola en la que se encuentra. A la cola RQi se le asigna 2i quantum de tiempo, de esta forma se trata de que menos procesos lleguen hasta la última cola sin terminar su ejecución.

En el caso de que ocurra starvation, en un proceso que se queda sin ejecución en la última cola, se lo puede enviar nuevamente hasta la cola de mayor prioridad para que continúe su ejecución.

La idea de este algoritmo es separar los procesos con diferentes características en cuanto a sus ráfagas de CPU. Si un proceso gasta demasiado tiempo en CPU, se le pasará a una cola con menor prioridad. Este esquema deja a los procesos limitados por E/S y los procesos interactivos en las colas de más alta

prioridad.

En general, un planificador de colas multinivel con realimentación está definido por los siguientes parámetros:

- El número de colas.
- El algoritmo de planificación para cada cola
- El método empleado para determinar cuándo se debe promover un proceso a una cola de mayor prioridad.
- El método empleado para determinar cuándo se debe promover un proceso a una cola de menor prioridad.
- El método empleado para determinar en cuál cola ingresará un proceso cuando necesite servicio.

La definición de un planificador de colas multinivel con realimentación lo convierte en el algoritmo de planificación de la CPU más general, ya que se puede configurar para adaptarlo a cualquier sistema específico que se este diseñando. Desdichadamente, se requiere alguna forma de seleccionar valores para todos los parámetros de manera que se obtenga el mejor planificador posible.

Aunque este esquema es el más general, también es el más complejo.

Planificación por reparto equitativo (FSS Fair share scheduling)

En un sistema multiusuario, si las aplicaciones o los trabajos de los usuarios pueden organizarse en forma de varios procesos (o hilos), se dispone de una estructura para el conjunto de procesos que no se identifica con ningún planificador tradicional. Desde el punto de vista del usuario, el interés no está en cómo se comporta un proceso en particular, sino en cómo se comporta el conjunto de procesos de usuario que constituyen una aplicación. Así pues, sería interesante poder tomar decisiones de planificación en función de estos grupos de procesos. Este enfoque se conoce generalmente como planificación por reparto equitativo.

El término reparto equitativo hace referencia a la filosofía del planificador. Cada usuario tiene asignado algún tipo de ponderación, que indica la parte de los recursos del sistema para el usuario como una fracción de la utilización total de dichos recursos. En particular, cada usuario dispone de una parte del procesador.

La planificación se lleva a cabo por prioridades, teniendo en cuenta la prioridad básica del proceso, su utilización reciente de la CPU y la utilización reciente de la CPU por parte del grupo al que pertenece. Cuanto mayor es el valor numérico de la prioridad, menor es ésta. Las fórmulas siguientes se aplican al proceso j del grupo k.

$CPUj(i) = CPUj(i - 1) / 2$

$GCPUk(i) = GCPUk(i - 1) / 2$

$Pj(i) = Basej + CPUj(i - 1) / 2 + GCPUk(i - 1) / (4 \times Wk)$

Donde:

CPUj(i) = Media ponderada de la utilización de la CPU del proceso j en el intervalo i.

GCPUk(i) = Media ponderada de la utilización de la CPU del grupo k en el intervalo i.

Pj(i) = Prioridad del proceso j al principio del intervalo i; los valores más bajos indican prioridades más altas.

Basej = Prioridad de base del proceso j.

Wk = Peso asignado al grupo k, con la restricción de 0 <= Wk <= 1 y Wk = 1.

Cada proceso tiene asignada una prioridad de base. Esta prioridad desciende a medida que el proceso y el grupo al que pertenece utilizan la CPU. En el caso de la utilización del grupo, la media se normaliza dividiendo por el peso del grupo. Cuanto mayor es el peso asignado al grupo, menos afecta su utilización a la prioridad.

Planificación con múltiples colas fijas

Este algoritmo pertenece a una clase de algoritmos de planificación para situaciones en las que es fácil clasificar los procesos en diferentes grupos.

Un algoritmo de planificación con colas de múltiples nivel divide la cola de procesos listos en varias colas distintas. Los procesos se asignan permanentemente a una cola, casi siempre con base en alguna propiedad del proceso, como ser tamaño de memoria, prioridad y tipo de proceso. Cada cola tiene su propio algoritmo de planificación.

Además debe haber planificación entre las colas, lo cual por lo regular se implementa como una planificación expropiativa de prioridades fijas. Por ejemplo, la cola de primer plano podría tener prioridad absoluta sobre la cola de segundo plano, por lo tanto mientras no se vacía la cola de prioridad superior, los procesos de la cola inferior no se ejecutan.

Otra posibilidad es dividir el tiempo entre las colas. Cada cola obtiene cierta proporción del tiempo de CPU, que entonces puede repartir entre los diversos procesos en su cola.

Este algoritmo tiene la ventaja de que el gasto por planificación es bajo y la desventaja de ser inflexible.

Planificación con múltiples colas dinámicas

Es idéntico al anterior con la diferencia que los procesos se pueden mover de una cola a otra cola.

El planificador se configura usando algunos de los siguientes criterios:

- Número de colas
- Algoritmo de planificación para cada cola.
- Método o criterio para subir / bajar un proceso.
- Criterio para determinar en qué cola se pone inicialmente a un proceso.
- Es muy apropiado para esquemas client − server.

Evaluación de algoritmos

La selección del algoritmo de planificación adecuado comienza por definir los criterios que se utilizarán y ordenarlos de acuerdo al perfil buscado. Una vez definido esto el siguiente paso es evaluar los diversos algoritmos que se estén considerando. Existen varios métodos de evaluación, que se describirán en las secciones siguientes.

Modelos determinísticos

Una clase importante de métodos de evaluación se denomina evaluación analítica. Estos métodos utilizan el algoritmo dado y la carga de trabajo del sistema para producir una fórmula o número que califica el desempeño del algoritmo para esa carga de trabajo.

Un tipo de evaluación analítica es el modelo determinista. Este modelo es simple y rápido. Da una visión precisa permitiendo comparar los algoritmos. Sin embargo, como entrada requiere números exactos y sus resultados se aplican sólo a esos casos, por lo que es demasiado específico para resultar útil en la mayoría de los casos.

Los diagramas de Gantt de los distintos algoritmos nos permiten hacer un análisis del desempeño de cada uno viendo el tiempo en que finalizan la ejecución.

Modelo de colas

En muchos sistemas, los trabajos que se ejecutan varían diariamente, por lo que no hay un conjunto estático de trabajos (y de tiempos) como para utilizar un modelo determinístico. Sin embargo lo que si puede determinarse es la distribución de las ráfagas de CPU y de E/S. Sobre esta distribución pueden tomarse datos y luego aproximarla o simplemente estimarla. El resultado es una fórmula matemática que describe la probabilidad de una ráfaga de CPU concreta. Corrientemente se trata de una distribución exponencial, que puede describirse en términos de su media. Asimismo, debe darse la distribución de los tiempos en que los procesos llegan al sistema. A partir de estas dos distribuciones, es posible calcular el rendimiento promedio, el aprovechamiento, el tiempo de espera y demás para la mayor parte de los algoritmos.

El sistema de computación puede describirse como una red de servidores. Cada servidor tiene una cola de espera. La CPU es un servidor con su cola de listos, así como el sistema de E/S lo es de su cola de dispositivos. Si conocemos los ritmos de llegada y de servicio, podemos calcular la utilización, la longitud media de la cola, el tiempo de espera medio, etc. Esta área de estudio recibe el nombre de análisis de redes de colas.

Ahora se hará uso de fórmulas básicas de la teoría de colas para poder estudiar los diferentes algoritmos, para ello haremos la suposición de que las llegadas siguen una Poisson y los tiempos de servicio una función exponencial.

En primer lugar, hay que observar que cualquier disciplina de planificación que elija el siguiente elemento a servir independientemente del tiempo de servicio cumple la siguiente relación:

$Tr / Ts = 1 / (1 -)$

Donde:

Tr = Tiempo de retorno o tiempo de estancia; tiempo total en el sistema, espera más ejecución.

Ts = Tiempo medio de servicio; tiempo medio consumido en el estado de ejecución.

 = Utilización del procesador.

Más concretamente, un planificador basado en prioridades, en el que la prioridad de cada proceso se asigna independientemente del tiempo esperado de servicio, proporciona el mismo tiempo medio de retorno y tiempo medio de retorno normalizado que un simple FCFS. Es más, la presencia o ausencia de apropiación no marca diferencia entre las medias.

De las diversas disciplinas de planificación vistas hasta ahora, gran parte de ellas hacen elecciones en función del tiempo esperado de servicio. Por desgracia, resulta bastante difícil construir modelos analíticos fiables para estas disciplinas. Sin embargo, es posible hacerse una idea del rendimiento relativo de dichos algoritmos de planificación en comparación con el FCFS, considerando una planificación por prioridades donde la prioridad está en función del tiempo de servicio.

Si la planificación se hace en función de prioridades y si los procesos se asignan a una clase de prioridad según su tiempo de servicio, entonces aparecen diferencias.

Fórmulas para colas de dos prioridades con un solo servidor:

Suposiciones:

- La llegada sigue una Poisson.
- Los elementos de prioridad 1 se sirven antes que los de prioridad 2.
- Los elementos de igual prioridad se expiden según FIFO.
- Los elementos no son interrumpidos durante el servicio.
- Ningún elemento abandona la cola (se retrasan las pérdidas).

Fórmulas generales

$= +$

$= Ts1; \quad = Ts2$

$= +$

$Ts = (/) Ts1 + (/) Ts2$

$Tr = (/) Tr1 + (/) Tr2$

(= tasa de llegada)

Sin interrupciones; tiempo de servicio exponencial

$Tr1 = Ts1 + (Ts1 + Ts2) / (1 -)$

$Tr2 = Ts2 + (Tr1 - Ts1) / (1 -)$

Disciplina de colas de reanudación preferente; tiempo de servicio exponencial

$Tr1 = 1 + Ts1 / (1 -)$

$Tr2 = 1 + (1 / (1 -)) (Ts1 + Ts / (1 -))$

Nótese que las fórmulas son diferentes para la planificación no preferente y preferente. En el último caso, se supone que un proceso de menor prioridad es interrumpido inmediatamente cuando uno de mayor prioridad está listo.

Modelos de simulación

Se utilizan si se busca obtener una evaluación más exacta de los algoritmos de planificación. Una

simulación implica programar un modelo del sistema de computación. Estructuras de datos en software representan los principales componentes del sistema. El simulador tiene una variable que representa un reloj; cuando se incrementa el valor de esta variable, el simulador modifica el estado del sistema de modo que refleje las actividades de los dispositivos, los procesos y el planificador. Conforme se ejecuta la simulación, se recopilan e imprimen datos estadísticos que indican el desempeño del algoritmo.

Las simulaciones pueden ser costosas, y a menudo requieren horas de tiempo de computador. Finalmente, el diseño, codificación y depuración del simulador no es una tarea trivial.

Implementación

Incluso las simulaciones tienen una exactitud limitada. La única forma exacta de evaluar un algoritmo de planificación es codificarlo, colocarlo en el sistema operativo, y ver cómo funciona.

El principal problema de este enfoque es el costo, ya que además de los gastos de codificación, adaptación del sistema operativo para que lo soporte, etc. existirá una reacción adversa de los usuarios, ya que estos no quieren el mejor sistema, sino que quieren que sus procesos se ejecuten y obtener sus resultados. El otro problema de cualquier evaluación de algoritmos es que el entorno en el que se usa el algoritmo cambiará. El cambio no será solo normal, a medida que se escriben nuevos programas y los tipos de problemas cambian, sino también el causado por el desempeño del planificador. Si se da prioridad a los procesos cortos, los usuarios tal vez dividan sus procesos grandes en grupos de pequeños procesos. Si se da prioridad a los procesos interactivos sobre los no interactivos, los usuarios podrían cambiar al uso interactivo.

Bibliografía

Sistemas Operativos − William Stallings

Notas Sobre Sistemas Operativos − Carlos Neetzel

Operating System Concepts − Silberschatz Galvin

Sistemas Operativos − Andrew S. Tanembaum