SISTEMAS OPERATIVOS – PARCIAL 3

GESTIÓN DE LA MEMORIA

MEMORIA VIRTUAL

HARDWARE Y ESTRUCTURAS DE CONTROL

Las dos características de la paginación y de la segmentación que son la clave de este principio:

- Todas las referencias a memoria dentro de un proceso se realizan a direcciones lógicas, que se traducen dinámicamente en direcciones físicas durante la ejecución. Esto significa que un proceso se puede traer varias veces a memoria principal, ocupando diferentes regiones en distintos instantes de tiempo de la ejecución.
- Un proceso puede dividirse en varias porciones que no tienen que estar localizadas en la memoria de forma contigua.

Supongamos que se tiene que traer un nuevo proceso a memoria. El sistema operativo comienza trayendo solo una o dos porciones, que incluye la porción inicial del programa y la porción inicial de datos.

La parte del proceso que se encuentra en memoria principal se llama **conjunto residente** del proceso.

Si el procesador encuentra una dirección lógica que no se encuentra en memoria principal, generará una interrupción indicando fallo de acceso. El S.O. coloca el proceso en estado bloqueado y toma el control. Para que la ejecución pueda reanudarse, el S.O. debe traer a memoria la porción del proceso que contiene la dirección lógica que ha causado el fallo de acceso. Con este fin, el S.O. realiza una petición de E/S, una lectura a disco.

Mediante esta técnica conseguimos:

- Mantener un mayor número de procesos en memoria principal.
- Un proceso puede ser mayor que toda la memoria principal. El programador no tiene que estar pensando en el tamaño ni en los *overlays*. Con la memoria virtual basada en paginación o segmentación, el trabajo se delega en el S.O. y en el hardware.

Como un proceso solo ejecuta en memoria principal, a esta se la conoce como **memoria real**, pero el programador percibe una memoria mucho más grande (localizada en disco) llamada **memoria virtual** (que permite una multiprogramación muy efectiva).

PROXIMIDAD Y MEMORIA VIRTUAL

Cuando el sistema consume la mayor parte del tiempo enviando y trayendo porciones de procesos a *swap* en lugar de ejecutar instrucciones, estamos ante un **trasiego**

(**thrasing**). Para evitar esto, el S.O. trata de adivinar, en base a la historia reciente, qué porciones es menos probable que sean utilizadas en un futuro cercano.

Este razonamiento se basa en el **principio de proximidad**, que indica que indica que las referencias al programa y a los datos dentro de un proceso, tienden a agruparse, es decir, solo unas pocas porciones del proceso se necesitarán a lo largo de un periodo de tiempo corto.

Para que la memoria virtual resulte práctica y efectiva se necesita:

- Debe existir un soporte hardware para el esquema de paginación y/o segmentación.
- El S.O. debe incluir código para gestionar el movimiento de páginas y/o segmentos entre la memoria secundaria y la memoria principal.

PAGINACIÓN

Normalmente la memoria virtual conlleva sistemas que emplean paginación.

En la paginación con memoria virtual, también tenemos una tabla de páginas para cada proceso, pero sus entradas son más complejas. Ahora necesitamos saber si cada página está presente en memoria principal o no, para lo que necesitamos un **bit de presencia**. También debemos indicar el número de marco de donde se ubica dicha página, de estar en memoria.

La entrada de la tabla de páginas también incluye un **bit de modificado** que indica si los contenidos de la página han sido modificados desde que esta llegó a memoria. Si no ha habido ningún cambio, no es necesario escribir la página cuando llegue el momento de reemplazarla por otra página.

Estructura de la tabla de páginas

Debido a que la tabla de páginas es de longitud variable dependiendo del tamaño del proceso, no podemos suponer que se encuentra almacenada en los registros. En lugar de eso, debe encontrarse en memoria principal para poder ser accedida.

Para evitar tener que guardar tablas de páginas enormes en memoria, almacenamos las tablas en memoria virtual, con lo cual, las tablas de páginas también tendrán paginación.

Cuando un proceso está en ejecución, al menos una parte de la tabla de páginas debe encontrarse en memoria, incluyendo la entrada de tabla de páginas de la página que se encuentra en ejecución.

También existen esquemas de dos niveles donde existe un directorio de páginas, en el cual cada entrada apunta a una tabla de páginas.

Tabla de páginas invertida

Una desventaja del tipo de tablas de páginas que hemos visto es que su tamaño es proporcional al espacio de direccione virtuales.

En esta estrategia, el número de página de la dirección virtual se referencia por medio de un valor hash. El valor hash es un puntero para la tabla de páginas invertida, que contiene las entradas de tablas de página. Hay una entrada de la tabla de páginas invertida por cada marco de página real en lugar de uno por cada página virtual. Debido

a que más de una dirección virtual puede traducirse en la misma entrada de la tabla hash, se usa una técnica de encadenamiento para evitar el desbordamiento.

La entrada en la tabla de páginas incluye la siguiente información:

- Número de página: número de página de la dirección virtual.
- Identificador del proceso: proceso propietario de la página.
- Bits de control: *flags* como válido, referenciado y modificado, e información de protección y cerrojos.
- Puntero de la cadena: valor nulo si no hay más entradas encadenadas en esta entrada. En otro caso, contiene el valor del índice de la siguiente entrada de la cadena.

Buffer de traducción anticipada

Toda referencia a memoria virtual puede causar dos accesos a memoria física: uno para buscar la entrada de la tabla de páginas apropiada y otro para buscar los datos solicitados.

Para solventar el problema de duplicar el tiempo de acceso a memoria, muchos sistemas utilizan una *cache* de alta velocidad (TLB) para entradas de la tabla de páginas.

Esta *cache* contiene aquellas entradas de la tabla de páginas que han sido utilizadas de forma más reciente.

Dada una dirección virtual, el procesador primero examina la TLB, si la entrada de la tabla de páginas solicitada está presente (acierto en la TLB), entonces se recupera el número de marco y se construye la dirección real. Si la entrada no se encuentra (fallo en la TLB), el procesador utilizará el número de página para indexar la tabla de páginas del proceso.

Debido a que la TLB solo contiene algunas de las entradas de toda la tabla de páginas, no es posible indexar simplemente la TLB por medio de número de página. Cada entrada de la TLB debe incluir un número de página, así como la entrada de la tabla de páginas completa.

El procesador proporciona un hardware que permite consultar simultáneamente varias entradas. Esta técnica se llama **resolución asociativa**.

Tamaño de página

Una decisión importante es el tamaño de página a usar. Hay varios factores a considerar. Por un lado, la fragmentación interna. Cuanto mayor es el tamaño de página, menor cantidad de fragmentación interna. Cuanto menor es la página, mayor número de páginas son necesarias para cada proceso. Un mayor número de páginas por proceso, significa también mayores tablas de páginas.

Se puede demostrar que el tamaño de página tiene relación con la posibilidad de que ocurra un fallo de página. Si el tamaño de página es pequeño, habrá un número relativamente grande de páginas disponibles en memoria principal para cada proceso, por lo que la tasa de fallos de página debería ser baja. Según vamos aumentando el tamaño, también aumenta la tasa de fallos de página, pero en algún momento, comenzará a caer a medida que el tamaño se aproxima al tamaño del proceso completo.

Otra complicación adicional es que la tasa de fallos de página también viene determinada por el número de marcos asociados a cada proceso. A tamaño de página

fijo, la tasa de fallos cae a medida que el número de páginas mantenidas en memoria principal crece.

SEGMENTACIÓN

La segmentación permite al programador ver la memoria como si se tratase de diferentes espacios de direcciones o segmentos.

La segmentación tiene un gran número de ventajas para el programador:

- Simplifica el tratamiento de estructuras de datos que pueden crecer. Con un esquema de memoria virtual segmentada, a una estructura de datos se le puede asignar un segmento y el S.O. lo expandirá o reducirá bajo demanda.
- Permite programas que se modifican de forma independiente, sin requerir que el conjunto completo se reenlace y se vuelva a cargar.
- Da soporte a la compartición entre procesos. Se puede situar información útil en un segmento al que se pueda hacer referencia desde varios procesos.
- Soporta los mecanismos de protección.

Organización

El mismo mecanismo que se usaba para la tabla de segmentos en segmentación sencilla se utiliza en el caso de memoria virtual segmentada. Pero cada entrada de la tabla ahora es más compleja, se necesita indicar que el segmento está presente o no en memoria principal y también la dirección de comienzo del segmento y la longitud del mismo. También se incluye un bit de modificado y pueden darse otros bits de control.

El mecanismo básico para la lectura de una palabra de memoria implica la traducción de una dirección virtual o lógica, consistente en un número de segmento y un desplazamiento, en una dirección física, usando la tabla de segmentos.

Cuando un proceso en particular está en ejecución, un registro mantiene la dirección de comienzo de la tabla de segmentos para dicho proceso. El número de segmento de la dirección virtual se utiliza para indexar esta tabla y para buscar la dirección de la memoria principal donde comienza dicho segmento. Esta es añadida a la parte de desplazamiento de la dirección virtual para producir la dirección real solicitada.

PAGINACIÓN Y SEGMENTACIÓN COMBINADAS

La paginación elimina la fragmentación externa además que debido a que los fragmentos que se mueven entre la memoria y el disco son de tamaño igual y prefijado, es posible desarrollar algoritmos de gestión de memoria más sofisticados. La segmentación permite manejar estructuras de datos que crecen, modularidad y da soporte a la compartición y la protección.

Para combinar las ventajas de ambos, algunos sistemas por medio del hardware del procesador y del soporte del S.O. son capaces de proporcionar ambos.

En un sistema combinado, el espacio de direcciones del usuario se divide en un número de segmentos. Cada segmento se divide en un número de páginas fijo, que son del tamaño de los marcos de la memoria principal.

Desde el punto de vista del programador, una dirección lógica, sigue conteniendo un número de segmento y un desplazamiento dentro de dicho segmento. Desde el punto de vista del sistema, el desplazamiento dentro del segmento es visto como un número de página y un desplazamiento dentro de la página incluida en el segmento.

Asociada a cada proceso existe una tabla de segmentos y varias tablas de páginas, una por cada uno de los segmentos.

Cuando un proceso está en ejecución, un registro guarda la dirección de comienzo de la tabla de segmentos.

A partir de la dirección virtual, el procesador utiliza la parte correspondiente al número de segmento para indexar dentro de la tabla de segmentos del proceso para encontrar la tabla de páginas de dicho segmento. Después, la parte correspondiente al número de página de la dirección virtual original se utiliza para indexar la tabla de páginas y buscar el correspondiente número de marco. Este se combina con el desplazamiento correspondiente de la dirección virtual para generar la dirección real requerida.

PROTECCIÓN Y COMPARTICIÓN

La segmentación proporciona una vía para la implementación de las políticas de protección y compartición.

Como una entrada de la tabla de segmentos incluye la longitud, así como la dirección base, un programa no puede acceder a una posición de memoria principal más allá de los límites del segmento.

SOFTWARE DEL SISTEMA OPERATIVO

El diseño de la parte de gestión de la memoria del S.O. depende de tres opciones fundamentales:

- Si el sistema usa o no técnicas de memoria virtual.
- El uso de paginación o segmentación o ambas.
- Los algoritmos utilizados para los diferentes aspectos de la gestión de la memoria.

Las elecciones posibles para las dos primeras opciones dependen de la plataforma hardware disponible. Las elecciones relativas a la tercera opción entran dentro del dominio del software del S.O.

Se tratará de minimizar la tasa de ocurrencia de fallos de página.

POLÍTICA DE RECUPERACIÓN

La política de recuperación determina cuando una página se trae a la memoria principal. Hay dos alternativas:

- **Paginación bajo demanda**: una página se trae a memoria solo cuando se hace referencia a una posición de dicha página.
- **Paginación adelantada** (**prepaging**): se traen a memoria también otras páginas, además de la que ha causado el fallo de página. Esta técnica es eficiente cuando en el disco se guardan las páginas contiguas, de esta forma se traen juntas en vez de una en una, lo que sería costoso, ya que el tiempo de acceso a disco es largo.

POLÍTICA DE UBICACIÓN

Esta política determina en qué parte de la memoria real van a residir las porciones de la memoria de un proceso.

En sistemas que usan segmentación se usan los algoritmos de primer ajuste, etc.

En los sistemas que usan paginación pura o paginación combinada con segmentación, la ubicación es habitualmente irrelevante debido a que el hardware de traducción de direcciones y el hardware de acceso a la memoria principal pueden realizar sus funciones en cualquier combinación de página-marco con la misma eficiencia.

POLÍTICA DE REEMPLAZO

Esta política trata de la selección de una página en la memoria principal como candidata para reemplazarse cuando se va a traer una nueva página. Hay que tener en cuenta:

- ¿Cuántos marcos de página se van a reservar para cada uno de los procesos activos?
- Si el conjunto de páginas que se van a considerar para realizar el reemplazo se limita a aquellas del mismo proceso que ha causado el fallo de página o, si, por el contrario, se consideran todos los marcos de página de la memoria principal.
- Entre el conjunto de páginas a considerar, qué página en concreto es la que se va a reemplazar.

Bloqueo de marcos

Algunos marcos de la memoria principal pueden encontrarse bloqueados. Cuando un marco está bloqueado, la página actualmente almacenada en dicho marco no puede reemplazarse.

Algoritmos básicos

Existen ciertos algoritmos básicos para la selección de la página a reemplazar:

- Óptimo.
- Usado menos recientemente (LRU).
- FIFO.
- Reloj.

La **política óptima** tomará como reemplazo la página para la cual el instante de la siguiente referencia se encuentra más lejos. Esta política es la que menor número de fallos ocasiona, pero es imposible de implementar.

La **política LRU** selecciona como candidata la página de memoria que no se haya referenciado desde hace más tiempo. La política LRU proporciona resultados casi tan buenos como la política óptima, pero su implementación es realmente difícil. Se puede guardar el instante en el que la página fue referenciada por última vez o se puede usar una pila de referencias a páginas.

La **política FIFO** trata los marcos de página ocupados como si se tratase de un *buffer* circular, y las páginas se reemplazan mediante una estrategia cíclica de tipo *round-robin*. Lo único que se necesita es un puntero que recorra de forma circular los marcos de página del proceso. Reemplaza la página que lleva en memoria más tiempo. Su implementación es bastante sencilla.

La **política de reloj** requiere de la inclusión de un bit adicional a cada marco de página, denominado bit de usado. Cuando la página se trae por primera vez a memoria, el bit se pone a 1 y después de cada vez que se usa cada vez también. Cuando hay que reemplazar una página, un puntero recorre circularmente todos los marcos. Cuando se encuentra un bit de usado a 1, lo pone a 0 y cuando se encuentra un bit a 0, selecciona esa página para reemplazarla. Este algoritmo puede hacerse más potente

incrementando el número de bits que utiliza. En todos los procesadores que soportan paginación, se asocia un bit de modificado a cada una de las páginas de la memoria principal.

Buffering de páginas

Usamos el algoritmo de reemplazo FIFO, porque su implementación es muy sencilla.

Para mejorar el rendimiento, una página reemplazada no se pierde, sino que se asigna a una de las dos siguientes listas: la lista de páginas libres si la página no ha sido modificada o la lista de modificadas si lo ha sido.

La lista de páginas libres es una lista de marcos de páginas disponibles para lectura de nuevas páginas. Cuando una página se va a leer se usa el marco de página en cabeza de esta lista, eliminando la página que estaba. Cuando se va a reemplazar una página que no se ha modificado, se mantiene en la memoria ese marco de página y se añade al final de la lista de páginas libres. De forma similar, cuando una página modificada se va a escribir y reemplazar, su marco de página se añade al final de la lista de páginas modificadas.

El aspecto más importante es que la página que se va a reemplazar se mantiene en la memoria, de forma que, si el proceso hace referencia a esa página, se devuelve al conjunto residente del proceso con un bajo coste.

La lista de páginas modificadas tiene otra función útil: las páginas modificadas se escriben en grupos en lugar de una en una, lo que reduce de forma significativa el número de operaciones de E/S y el tiempo de acceso a disco.

GESTIÓN DEL CONJUNTO RESIDENTE

El S.O. debe saber cuántas páginas de un proceso traerse, es decir, cuánta memoria principal debería reservar para un proceso en particular. Diferentes factores entran en juego:

- Cuanto menor es la cantidad de memoria reservada para un proceso, mayor es el número de procesos que pueden residir en memoria. Esto aumenta la probabilidad de que el S.O. encuentre un proceso listo para ejecutarse.
- Si el conjunto de páginas en memoria de un proceso es pequeño, la posibilidad de un fallo de página es mayor.

La política de **asignación fija** proporciona un número fijo de marcos de memoria principal disponibles para ejecución. Siempre que se produzca un fallo de página, la página que se necesite reemplazará una de las páginas del proceso.

Una política de **asignación variable** permite que se reserven un número de marcos por proceso que pueden variar a lo largo del tiempo de vida del mismo. A un proceso que esté dando muchos fallos de página se le darán más marcos y viceversa.

Ámbito de reemplazo

Una política de **reemplazo local** selecciona solamente entre las páginas residentes del proceso que ha generado el fallo de página.

En el caso de la política de **reemplazo global**, se consideran todas las páginas en la memoria principal que no se encuentren bloqueadas, como candidatos para el reemplazo.

Asignación fija, ámbito local

Se parte de un proceso que se encuentra en ejecución en la memoria principal con un número de marcos fijo. Cuando se da un fallo de página, el S.O. debe elegir una página entre las residentes del proceso para reemplazar.

En esta política es necesario decidir por adelantado la cantidad de espacio reservado para un proceso determinado.

La desventaja es que, si las reservas resultan ser demasiado pequeñas, habrá que soportar una alta tasa de fallo de página. Y en el caso contrario, habrá muy pocos programas en memoria principal.

Asignación variable, ámbito global

Existen un número de procesos determinado en la memoria principal, cada uno de los cuales tiene una serie de marcos asignados. También se guarda una lista de marcos libres.

Cuando sucede un fallo de página, se añade un marco libre al conjunto residente de un proceso y se trae una página a dicho marco. Un proceso que sufra diversos fallos de página, crecerá gradualmente en tamaño.

Asignación variable, ámbito local

Cuando se carga un nuevo proceso, se le asigna un número de marcos. Para cubrir esta reserva se usa la paginación adelantada o la paginación bajo demanda.

Cuando ocurra un fallo de página, la página que se seleccionará para reemplazar pertenecerá al conjunto residente del proceso que causó el fallo.

De vez en cuando, se reevaluará la asignación proporcionada a cada proceso, incrementándose o reduciéndose para mejorar al rendimiento.

Una estrategia para determinar el tamaño del conjunto residente y la periodicidad de los cambios es la denominada **estrategia del conjunto de trabajo**.

POLÍTICA DE LIMPIEZA

La política de limpieza es la opuesta a la política de recuperación. Se encarga de determinar cuándo una página modificada se debe escribir en memoria secundaria. Hay dos alternativas.

Con la **limpieza bajo demanda**, una página se escribe a memoria secundaria solo cuando se ha seleccionado para el reemplazo.

En la **limpieza adelantada**, se escribe las páginas modificadas antes de que sus marcos de páginas se necesiten, de forma que las páginas se pueden escribir en lotes.

CONTROL DE CARGA

El control de carga determina el número de procesos que residirán en la memoria principal, eso se denomina el grado de multiprogramación.

Si hay muy pocos procesos residentes a la vez, habrá muchas ocasiones en las cuales todos los procesos se encuentren bloqueados y gran parte del tiempo se gastará haciendo *swapping*. Por otro lado, si hay demasiados procesos residentes, entonces se producirán muchos fallos de página. El resultado es el trasiego.

Grado de multiprogramación

Algunos algoritmos como el del conjunto de trabajo o el de frecuencia de fallos de página, incorporan control de carga. Solo aquellos procesos cuyo conjunto residente es suficientemente grande se les permite ejecutar.

Otra estrategia es el **criterio L = S**, que ajusta el nivel de multiprogramación de forma que el tiempo medio entre fallos de página se iguala al tiempo medio necesario para procesar un fallo de página.

Suspensión de procesos

Si se va a reducir el grado de multiprogramación, uno o más de los procesos actualmente residentes deben suspenderse (enviarlos a *swap*). Hay seis posibilidades para escoger que procesos enviar a *swap*:

- **Procesos con baja prioridad**: implementa una decisión de la política de activación y no tiene relación con cuestiones de rendimiento.
- **Procesos que provocan muchos fallos**: es muy probable que la tarea que causa fallos no tenga su conjunto de trabajo residente y el rendimiento sufrirá menos si dicha tarea se suspende.
- **Proceso activado hace más tiempo**: proceso que tiene menor probabilidad de tener su conjunto de trabajo residente.
- **Proceso con el conjunto residente de menor tamaño**: es el que menor esfuerzo requerirá al cargarse de nuevo.
- **Proceso mayor**: es el que proporciona un mayor número de marcos libres en una memoria que se encuentra sobrecargada.
- **Procesos con la mayor ventana de ejecución restante**: en la mayoría de los esquemas de activación de procesos, un proceso solo puede ejecutarse durante una determinada rodaja de tiempo antes de recibir la interrupción y situarse al final de la cola de listos. Esta estrategia se aproxima a la disciplina de activación de primero el proceso con menor tiempo en ejecución.