Práctica 5 - Memoria

Page • 1 enlace entrante • Tag

La organización y administración de la memoria RAM es uno de los factores más importantes en el diseño de un SO. Los programas y los datos necesitan estar en la RAM para poder ejecutarse y referenciarlos.

Administrador de Memoria

- Lleva un registro de las partes de la memoria que se están utilizando y de aquellas que no
- Asigna espacio a los procesos cuando estos la necesitan
- Libera espacio asignado a procesos que han terminado
- Su uso debe ser lo mas eficiente que se pueda con el fin de alojar la mayor cantidad de procesos posible

Direccionamiento

- **Dirección física** → Dirección con la que se accede efectivamente a memoria. Representa la dirección absoluta en memoria.
- **Dirección lógica** → Dirección que abstrae a una dirección física. Hace referencia a una localidad en memoria, pero se la debe traducir antes de poder ser utilizada (con la *MMU*)
- **Memory Management Unity (MMU)** → Dispositivo de Hardware que mapea direcciones lógicas a físicas (en tiempo de ejecución). Es parte del procesador. Se suma el valor del "registro de realocación" a cada dirección generada al momento de acceder a la memoria.

Técnicas de Asignación de Memoria

Fragmentación

Espacio de la memoria que no puede ser utilizado por no encontrarse de manera contigua

- **Interna** → Espacio no utilizado de una localidad asignada (*partición / página / segmento*)
- **Externa** → Huecos que van quedando en la memoria al terminar los procesos
    - **Solución** → Compactación (costosa)

Particiones fijas

La memoria principal se divide en porciones de tamaño fijo que alojan un solo proceso (tmn_proc <= tmn_part)

- Los procesos son alojados de acuerdo a algún criterio (Best, Worst, First, Next)
- Genera *fragmentación interna*

Particiones dinámicas

Las particiones varían en tamaño y número

- Cada partición se genera en forma dinámica del tamaño justo que necesita el proceso
- Genera *fragmentación externa*

Paginación

La memoria principal se divide en porciones de igual tamaño llamados **marcos**, y el espacio de direcciones de los procesos se dividen en **páginas** (tmn_marco = tmn_pag)

- El SO mantiene una **tabla de páginas** por proceso, que contiene los marcos donde se encuentran sus páginas
- **Obtener la página/marco de una dirección lógica/física** → dir_log/dir_fis DIV tmn_pag/tmn_marc (con / separo opciones)
- **Obtener el desplazamiento dentro de la página/marco** → dir_log/dir_fis MOD tmn_pag/tmn_marc
- **Convertir la dirección lógica/fisica a física/lógica** → (num_pag/num_marc * tmn_pag/tmn_marc) + despl_pag/despl_marc
- Genera *fragmentación interna*
- Puede generar *hiperpaginación* (thrashing)

Segmentación

Los espacios de direcciones de los procesos se dividen en segmentos según su contenido, y estos a su vez se subdividen

- El SO mantiene una **tabla de segmentos** por proceso, con la dirección base del segmento más su longitud
- Si se combina esta técnica con paginación se obtiene la **segmentación paginada (**segmentos con páginas y marcos**)**
    - **Obtener la dir. física de una dir. lógica** → (Dir. base segmento) + (Dir. base página) + desplazamiento

Memoria Virtual

La técnica de páginacion, gracias a la división de porciones del código y a las direcciones lógicas, si se hace "**bajo demanda**" (ir trayendo páginas a medida que se necesitan para seguir ejecutando el proceso), crea la ilusión de que la memoria es mil veces mayor de la que realmente se tiene (*memoria real*) y se pueden ejecutar muchos mas procesos a la vez

- Cada vez que hay que alojar una página en un marco → **fallo de página**
- Si no quedan marcos disponibles, hay que decidir qué página cargada se expulsa → **política de reemplazo**
- También hay que decidir si expulsar la página del mismo proceso (*local*) o de cualquiera (*global*) → **alcance del reemplazo**

Políticas de Reemplazo

- Lo ideal sería reemplazar las páginas menos referenciadas en el futuro próximo → **algoritmo óptimo** → *irrealizable*
- **Least Recently Used (LRU)** → Reemplaza la página que no fue referenciada por mas tiempo (overhead ult. instante)
- **First In First Out** (FIFO) → Va reemplazando las páginas en ronda (implementación simple, rendimiento pobre)
- **FIFO con Segunda Chance** → Se utiliza un bit adicional puesto en 0 cuando la página se carga, y en 1 cuando se referencia. La víctima se busca en orden FIFO, y mientras se busca se cambian los bits de las páginas a 0

Asignación de Marcos

- **Asignación fija** → A cada proceso se le asigna una cantidad arbitraria de marco (incompatible con *pol. reem. global*)
    - **Reparto equitativo** → Se le asigna la misma cantidad de marcos a todos
    - **Reparto proporcional** → Se le asigna una cantidad proporcional a su tamaño/necesidad
- **Asignación dinámica** → Los procesos se van cargando en forma dinámica de acuerdo a la cantidad de marcos que necesiten

Marco para la descarga asincrónica de páginas

El SO reserva uno o mas marcos para la descarga asincrónica de páginas. Cuando se necesita descargar una pag. modificada:

- La página que provoco el fallo se coloca en un marco designado a la descarga asincrónica
- El SO envía la orden de descargar asincrónicamente la página modificada mientras continua la ejecución de otro proceso
- El marco de descarga asincrónica pasa a ser el que contenía a la página víctima que ya se descargó correctamente

Performance

La paginación bajo demanda puede degradar el rendimiento del sistema

- Si el sistema utiliza mas tiempo paginando que ejecutando instrucciones → **Thrashing**
- **Anomalía de Belady** → Situación donde se obtienen mas fallos de páginas al aumentar el número de marcos, utilizando la política de reemplazo FIFO. Esto ocurre porque se van reemplazando las páginas que se tienen que ejecutar próximamente
- Para medir el costo de los fallos de página utilizamos la siguiente fórmula

- **Am** = tiempo de acceso a la memoria real
- **Tf** = tiempo de atención de un fallo de página
- **At** = tiempo de acceso a la tabla de páginas. Es igual al tiempo de acceso a la memoria (*Am*) si la entrada de la tabla de páginas no se encuentra en la *TLB* (cache donde residen las traducciones de direcciones realizadas)

$$\mathbf{TAE = At + (1 - p) * Am + p * (Tf + Am)}$$

Modelo de Localidad

Por principio de proximidad sabemos que las referencias a datos y programas dentro de un proceso tienden a agruparse

- La **localidad de un proceso,** en un momento dado, se da por el conjunto de páginas que son referenciadas en ese momento
- **Working Set** → Conjunto de páginas que tienen las Δ referencias a páginas más recientes. Este tiene que abarcar tantas páginas como para cubrir la *localidad* del proceso y no mas

Para determinar la medida del *working set* debemos tener en cuenta:

- $m =$ cantidad de *frames* disponibles
- $D =$ demanda total de *frames*
- $WSS_i =$ medida del *working set* del proceso$_i$
- $\sum WSS_i = D$
- Si $D \rangle m$ habrá **thrashing**