

Administración de Memoria Principal

Explicación de práctica

Introducción a los Sistemas Operativos
Conceptos de Sistemas Operativos

Facultad de Informática Universidad Nacional de La Plata

2025



Memoria Virtual con Paginación

- La técnica de paginación intenta alocar la mayor cantidad de páginas necesarias posibles
- Cada vez que hay que alocar una página en un *marco*, se produce un **fallo de página** (page fault) → *hard page fault*
- ¿Qué sucede si es necesario alocar una página y ya no hay espacio disponible?
 - Se debe seleccionar una p'ágina v'ictima, para lo cual existen diversos algoritmos
 - ¿Cu'ál es el mejor algoritmo?:
 - El que seleccione como p'ágina v'ictima aquella que no vaya a ser referenciada en un futuro pr'oximo
 - La mayoria de los algoritmos predicen el comportamiento futuro mirando el comportamiento pasado



pasado

Memoria Virtual con Paginación

- La técnica de paginación intenta alojar la mayor cantidad de páginas necesarias posibles
- Cada vez que hay que alojar una página en un *marco*, se produce un **fallo de página** (page fault) → *hard page fault*
- ¿Qué sucede si es necesario alojar una página y ya no hay espacio disponible?
- Se debe seleccionar una página vícima, para lo cual existen diversos algoritmos
- ¿Cuál es el mejor algoritmo?:
 - El que seleccione como página vícima aquella que no vaya a ser referenciada en un futuro próximo
 - La mayoría de los algoritmos predicen el comportamiento futuro mirando el comportamiento pasado



pasado

Memoria Virtual con Paginación

- La técnica de paginación intenta alojar la mayor cantidad de páginas necesarias posibles
- Cada vez que hay que alojar una página en un *marco*, se produce un **fallo de página** (page fault) → *hard page fault*
- ¿Qué sucede si es necesario alojar una página y ya no hay espacio disponible?
 - Se debe seleccionar una página víctima, para lo cual existen diversos algoritmos
- ¿Cuál es el mejor algoritmo?:
 - El que seleccione como página víctima aquella que no vaya a ser referenciada en un futuro próximo
 - La mayoría de los algoritmos predicen el comportamiento futuro mirando el comportamiento pasado



- Selecciona la página cuyo próxima referencia se encuentra más lejana a la actual
- Imposible de implementar → no se conoce los futuros eventos

Marco/Página	1	2	1	3	4	1	4	3	5
F1	1	1	1	1	1	1	1	1	?
F2		2	2	2	4	4	4	4	?
F3				3	3	3	3	3	?
PF?	X	X		X	X				X

Continuación de la secuencia: 4 6 3 5 8 1



- El algoritmo *LRU* (Least Recently User) reemplaza la página que no fue referenciada por más tiempo
- Cada página debe tener información del instante de su última referencia → el overhead es mayor

Marco/Página	1	2	1	3	4	1	4	3	5
F1	1	1	1	1	1	1	1	1	?
F2		2	2	2	4	4	4	4	?
F3				3	3	3	3	3	?
PF?	X	X		X	X				X



- El algoritmo **FIFO** (First In First Out) trata a los *frames* en uso como una cola circular
- Simple de implementar
- La página más vieja en la memoria es reemplazada
- Contra? La página puede ser necesitada pronto

Marco/Página	1	2	1	3	4	1	4	3	5
F1	1	1	1	1	4	4	4	4	?
F2		2	2	2	2	1	1	1	?
F3				3	3	3	3	3	?
PF?	X	X		X	X	X			X



Algoritmo FIFO con segunda chance

- Se utiliza un *bit* adicional → *bit de referencia*
- Cuando la página se carga en memoria, el *bit R* se pone a 0
- Cuando la página es referenciada el *bit R* se pone en 1
- La víctima se busca en orden *FIFO*. Se selecciona la primer página cuyo *bit R* esta en 0
- Mientras se busca la víctima cada *bit R* que tiene el valor 1, se cambia a 0



Algoritmo FIFO con segunda chance

Marco/Página	1	2	1	3	4	1	4	3	5
F1	1	1	1*	1*	1	1*	1*	1*	?
F2		2	2	2	4	4	4*	4*	?
F3				3	3	3	3	3*	?
PF?	X	X		X	X				X

* = bit R = 1



- La asignación de *marco* se puede realizar de dos modos:
 - **Asignación Fija:** a cada proceso se le asigna una cantidad arbitraria de *marco*. A su vez para el reparto se puede usar:
 - **Reparto equitativo:** se asigna la misma cantidad de *marcos* a cada proceso → $m \text{ div } p$
 - **Reparto Proporcional:** se asignan marcos en base a la necesidad que tiene cada proceso → $V_p \cdot m / V_T$
 - **Asignación dinámica:** los procesos se van cargando en forma dinámica de acuerdo a la cantidad de marcos que necesiten



- Al momento de seleccionar una página víctima, podemos utilizar:
 - **Reemplazo global:** el fallo de página de un proceso puede reemplazar la página de cualquier proceso
 - **Reemplazo local:** el fallo de página de un proceso solo puede reemplazar sus propias páginas



Descarga asincrónica de páginas

- El sistema operativo reserva uno o varios *marco* para la descarga asincrónica de páginas
- Cuando es necesario descargar una página modificada:
 - La página que provocó el fallo se coloca en un *frame* designado a la descarga asincrónica
 - El SO envía la orden de descargar asincrónicamente la página modificada mientras continúa la ejecución de otro proceso
 - El *frame* de descarga asincrónica pasa a ser el que contenía a la página víctima que ya se descargó correctamente



Descarga asincrónica de páginas (ejemplo)

- Ejemplo de algoritmo
FIFO

Marco/ Página	1	2	1 ^M	3	4	3 ^M	5		6	7	
F1	1	1	1 ^M			7	7				
F2		2	2	2	2	2	2	2	6	6	6
F3				3	3	3 ^M					
F4					4	4	4	4	4	4	4
F5							5	5	5	5	5



- La técnica de paginación por demanda puede generar una degradación de rendimiento del sistema debido a que el reemplazo de páginas es costoso
- Tasa de *page faults* $0 \leq p \leq 1$:
 - Si $p = 0$, no hay *page faults*
 - Si $p = 1$, cada referencia es un *page fault*
- *Effective Access Time*: medida utilizada para medir este costo:
 - **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)

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



- *Thrashing* (hiperpaginación):
 - decimos que un sistema está en *thrashing* cuando pasa más tiempo paginando que ejecutando procesos
- Si un proceso cuenta con todos los *frames* que necesita, no habría *thrashing*. Salvo excepciones como la *anomalía de Belady*
- Existen técnicas para evitarlo → estrategia de *Working Set*



Working Set - Modelo de Localidad

- 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
- En cortos períodos de tiempo, el proceso necesitará pocas “piezas” del proceso (una página de instrucciones y otra de datos)
- Entonces se define una ventana de trabajo o *working set* (Δ) que contiene las referencias de memoria más recientes
- **Working Set:** es el conjunto de páginas que tienen las Δ referencias a páginas más recientes



- Δ chico: no cubrirá la localidad. Toma muy pocas referencias
- Δ grande: puede tomar varias localidades. Toma referencias de la localidad y algunas más, posiblemente viejas
- 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
 - $\sum WSS_i = D$
 - Si $D > m$ habrá *thrashing*



¿Preguntas?

