Algoritmos de Reemplazo de Páginas

...

1.1 Algoritmo FIFO

Algoritmo FIFO

Este algoritmo es el más simple de todos, es bastante complicado verlo implementado en esta forma, como su nombre lo dice, el algoritmo primero en entrar, primero en salir, utiliza una cola para ir guardando el orden con el que llegan las referencias, y si ésta no se encuentra dentro de los marcos de página, elimina a la primer dirección que llegó. Este algoritmo presenta un problema principal, llamada anomalía FIFO o anomalía de Belady, que nos dice que, utilizando este algoritmo, si aumentamos el tamaño de marcos, también aumenta el número de fallos de página. A continuación veamos un ejemplo utilizando este algoritmo y el número de fallos de página producidos.

Características

¢Fácil de implementar

Lista de páginas del sistema

•Cuando SO asigna marco a página ponerla al final de la lista

•Cuando SO tiene que reemplazar página, saca la primera de la lista

¢¿Por qué podría ser bueno?

Quizás la página que entró a memoria primero ahora no se necesita

¢¿Por qué podría ser malo?

Quizás página está en uso

No hay información en absoluto

Anomalía de Belady

1.2 Algoritmo Óptimo

Algoritmo óptimo

Este algoritmo tiene como finalidad retirar la página que tenga el tiempo más largo para volver a ser referenciada. Por ejemplo, supongamos que tenemos dos páginas, A y B, la página A será usada en 10,000 instrucciones, y la página B en 5,000, entonces la página A se debe retirar.

En este algoritmo, es necesario que el sistema operativo conozca en cuánto tiempo será utilizada cada página en memoria y elegir cuál es la más distante.

El problema que presenta este algoritmo es que es necesario saber el futuro de las necesidades del sistema, por lo que es imposible de implementar.

Algoritmo óptimo

Al ser imposible, únicamente se utiliza para ser comparado contra otros algoritmos, debido a que de todos, es el que menos fallos de página produce.

Comparación entre FIFO y algoritmo óptimo:

Supongamos la siguiente cadena de referencia: 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2 con tres marcos.

1.3 Algoritmo LRU

Algoritmo LRU (Least Recently Used)

Este algoritmo de reemplazo consiste en reemplazar la página que tenga el mayor tiempo sin haber sido referenciada en memoria y moverla a disco.

A diferencia del algoritmo óptimo este algoritmo es posible de implementar, aunque esta implementación puede tener un gran consumo de recursos dentro del Sistema Operativo.

Algunas formas de realizar la implementación es mediante el uso de listas enlazadas o mediante el uso de un contador en hardware.

Algoritmo LRU (Least Recently Used)

En general el rendimiento de este algoritmo es bueno teóricamente pero como ya ha sido mencionado su implementación implica un gran gasto de recursos, por ello existen diversas variaciones que buscan aproximarse con un uso menor de recursos y tratando de mantener un rendimiento similar.

1.4 Algoritmos de Aproximación a LRU

Algoritmos con bits de referencia

- Las técnicas basadas en LRU, utilizan un bit de referencia puesto por el hardware.
- El hardware enciende el bit de referencia de una página, cada que se hace referencia a ella (Lectura o escritura).
- Examinando este bit no conocemos el orden de uso, pero si conocemos cuáles páginas han sido usadas y cuáles no.
- Es posible obtener información adicional si registramos los bits de referencia en diferentes intervalos.

Algoritmo de segunda oportunidad (Reloj)

- Sencillo y efectivo, examina la página más antigua como posible víctima.
- Comportamiento: FIFO teniendo en cuenta el bit de referencia.
  - Bit de referencia a cero: Reemplazo de página
  - Bit de referencia a uno: Segunda oportunidad, ponemos el bit de referencia a cero y seleccionamos la siguiente página FIFO.
- Se puede implementar como una cola circular, donde un puntero indicará cual es la página a reemplazar a continuación.
- Cuando uno necesita un marco, el puntero avanza hasta encontrar una página cuyo bit de referencia está apagado. Con el avance del puntero, los bits de referencia encendidos se irán apagando.

Algoritmo de segunda oportunidad mejorado

- Bit de referencia + bit de recuperación
- Usando estos dos bits tenemos cuatros posibles situaciones:
  - (0, 0): No se ha usado ni modificado recientemente.
  - (0, 1): No se ha usado recientemente, pero sí modificado.
  - (1, 0): Usada recientemente, pero no modificado.
  - (1, 1): Usada y modificada recientemente.
- Se reemplaza la página que encontremos de clase más baja.
- Se da preferencia a las páginas que han sido modificadas a fin de reducir el número de operaciones de E/S requeridas.