MÔN HỌC: HỆ ĐIỀU HÀNH

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 8

1. **Tại sao cần phải có bộ nhớ ảo?**

   Không phải tất cả các phần của một process cần thiết phải được nạp vào bộ nhớ chính tại cùng một thời điểm. Bộ nhớ ảo là một kỹ thuật cho phép xử lý một tiến trình không được nạp toàn bộ vào bộ nhớ vật lý.
   Với bộ nhớ ảo, chúng ta sẽ có thể làm được những điều sau :
   - Thực thi một chương trình có kích thước lớn hơn nhiều so với bộ nhớ chính.
   - Lượng process trong bộ nhớ cũng nhiều hơn.
   - Trừu tượng hoá bộ nhớ chính thành một mảng array lớn. Người dùng chỉ thấy được bộ nhớ thông qua bộ nhớ luận lý (logical memory). Từ đó giúp người lập trình viên đỡ cơ cực hơn.
   - Bộ nhớ ảo cũng giúp cho các tiến trình chia sẻ dữ liệu với nhau dễ dàng, ngoài ra còn hỗ trợ trong việc cài đặt shared memory (bộ nhớ dùng chung).
2. **Có bao nhiêu kỹ thuật cài đặt bộ nhớ ảo? Mô tả sơ lược các kỹ thuật đó?**

   Có hai kỹ thuật chính được sử dụng để cài đặt bộ nhớ ảo:

   - Phân trang theo yêu cầu (Demand Paging): Đây là một kỹ thuật phổ biến trong việc quản lý bộ nhớ ảo. Trong phân trang theo yêu cầu, hệ thống chỉ tải các trang vào bộ nhớ vật lý khi chúng thực sự cần thiết cho việc thực thi. Điều này giúp tiết kiệm bộ nhớ vật lý và tăng hiệu suất hệ thống.

   - Phân trang kết hợp với hoán chuyển (Swapping): Trong kỹ thuật này, hệ thống sẽ hoán chuyển toàn bộ quá trình hoặc phần của

quá trình từ bộ nhớ vật lý sang bộ nhớ ảo khi cần. Điều này giúp giải phóng không gian bộ nhớ vật lý cho các quá trình khác.

Cả hai kỹ thuật trên đều giúp tối ưu hóa việc sử dụng bộ nhớ vật lý và bộ nhớ ảo, giúp hệ thống hoạt động mượt mà hơn.

3. Các bước thực hiện kỹ thuật phân trang theo yêu cầu?

Demand paging: các trang của tiến trình chỉ được nạp vào bộ nhớ chính khi được yêu cầu.

Khi có một tham chiếu đến một trang mà không có trong bộ nhớ chính (valid bit) thì phần cứng sẽ gây ra một ngắt (gọi là page-fault trap) kích khởi page-fault service routine (PFSR) của hệ điều hành.

PFSR:

- Bước 1: Chuyển process về trạng thái blocked
- Bước 2: Phát ra một yêu cầu đọc đĩa để nạp trang được tham chiếu vào một frame trống; trong khi đợi I/O, một process khác được cấp CPU để thực thi
- Bước 3: Sau khi I/O hoàn tất, đĩa gây ra một ngắt đến hệ điều hành; PFSR cập nhật page table và chuyển process về trạng thái ready.

Bước 2 của PFSR giả sử phải thay trang vì không tìm được frame

trống, PFSR được bổ sung như sau:

- Xác định vị trí trên đĩa của trang đang cần
- Tìm một frame trống:
  - Nếu có frame trống thì dùng nó
  - Nếu không có frame trống thì dùng một giải thuật thay trang để chọn một trang hy sinh (victim page)

    - Ghi victim page lên đĩa; cập nhật page table và frame table tương ứng
- Đọc trang đang cần vào frame trống (đã có được từ bước 2); cập nhật page table và frame table tương ứng

4. Mô tả các giải thuật thay thế trang FIFO, OPT, LRU?

1. FIFO:

Cần biết được: Số khung trang, tình trạng ban đầu, chuỗi tham chiếu.

Hướng tiếp cận: Ghi nhận thời điểm một trang được mang vào bộ nhớ chính. Khi cần thay thế trang, trang ở trong bộ nhớ lâu nhất sẽ được chọn.

Lưu ý: Nghịc lý belady: Số lượng lỗi trang sẽ tăng lên mặc dù số khung trang sử dụng tăng lên.

Ví dụ: Xét chuỗi tham chiếu: 1.2.3.4.1.2.5.1.2.3.4.5

Sử dụng 3 khung trang sẽ có 9 lỗi trang

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5	5	5	5	5	5
	2	2	2	1	1	1	1	1	3	3	3
		3	3	3	2	2	2	2	2	4	4
*	*	*	*	*	*	*			*	*	

Sử dụng 4 khung trang sẽ có 10 lỗi trang

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3
*	*	*	*			*	*	*	*	*	*

2. OPT:

Thay thế trang nhớ được tham chiếu trễ nhất trong tương lai.

Ưu điểm: ít xảy ra lỗi trang nhất.

Nhược điểm: Khó hiện thực được.

Ví dụ: cho chuỗi : 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

Sử dụng 3 khung trang với 9 lỗi trang

7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**1**	**2**	**0**	**1**	**7**	**0**	**1**
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
*	*	*	*		*		*			*			*				*		

3. LRU:

Thay thế trang nhớ được tham chiếu trễ nhất trong quá khứ.

Thuật toán FIFO sử dụng thời điểm nạp để chọn trang thay thế, thuật toán tối ưu lại dùng thời điểm trang sẽ được sử dụng, vì thời điểm này không thể xác định trước nên thuật toán LRU phải dùng thời điểm cuối cùng trang được truy xuất – dùng quá khứ gần để dự đoán tương lai. Thuật toán này đòi hỏi phải được cơ chế phần cứng hỗ trợ để xác định một thứ tự cho các trang theo thời điểm truy xuất cuối cùng.

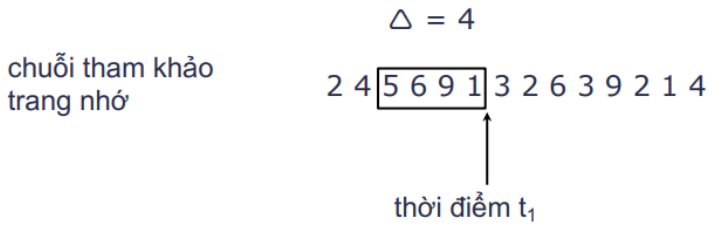
Ví dụ: cho chuỗi : 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

Sử dụng 3 khung trang với 12 lỗi trang

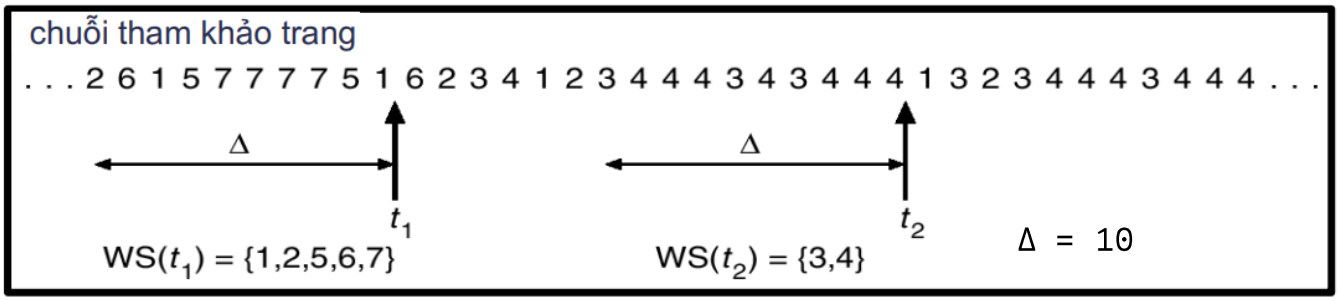
7	**0**	**1**	**2**	**0**	**3**	**0**	**4**	**2**	**3**	**0**	**3**	**2**	**1**	**2**	**0**	**1**	**7**	**0**	**1**
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
*	*	*	*		*		*	*	*	*			*		*		*		

5. Giải pháp tập làm việc hoạt động như thế nào?

- Được thiết kế dựa trên nguyên lý locality.
- Xác định xem process thực sự sử dụng bao nhiêu frame.
- Định nghĩa:
  - WS(t) – các tham chiếu trang nhớ của process gần đây nhất cần được quan sát.
  - r – khoảng thời gian tham chiếu
- Ví dụ:

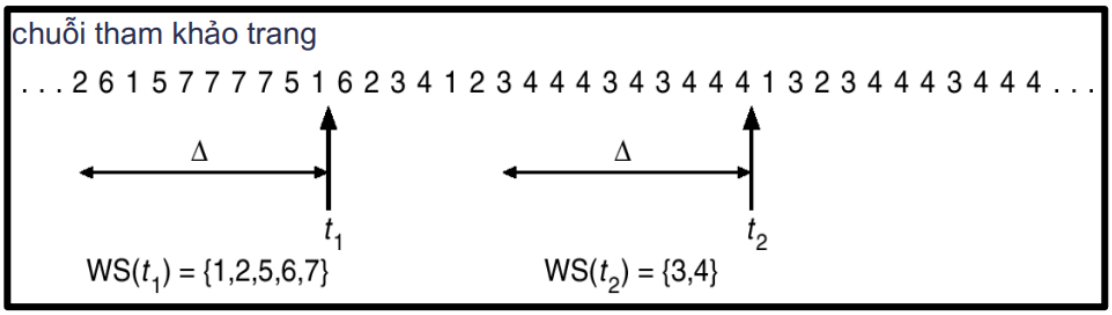


- Định nghĩa: Working set của process Pi , ký hiệu WSi, là tập gồm Δ các trang được sử dụng gần đây nhất.



- Nhận xét:
  - Δ quá nhỏ ⇒ không đủ bao phủ toàn bộ locality.
  - Δ quá lớn ⇒ bao phủ nhiều locality khác nhau.
  - Δ = ∞ ⇒ bao gồm tất cả các trang được sử dụng
- Dùng working set của một process để xấp xỉ locality của nó
- WSSi là kích thước của working set của Pi:WSSi = số lượng các trang trong Wsi

Ví dụ: Δ = 10 và



WSS(t_1) = 5 WSS(t_2) = 2

- Đặt D = Σ WSSi = tổng các working-set size của mọi process trong hệ thống.

Nhận xét: Nếu D > m (số frame của hệ thống) ⇒ sẽ xảy ra thrashing.

- Giải pháp working set:
  - Khi khởi tạo một quá trình: cung cấp cho quá trình số lượng frame thỏa mản working-set size của nó.
  - Nếu D > m ⇒ tạm dừng một trong các process. Các trang của quá trình được chuyển ra đĩa cứng và các frame của nó được thu hồi.

6. (Bài tập mẫu) Xét chuỗi truy xuất bộ nhớ sau: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6 Có bao nhiêu lỗi trang xảy ra khi sử dụng các thuật toán thay thế sau đây, giả sử có 4 khung trang.

a. LRU

b. FIFO

c. Chiến lược tối ưu (OPT)

Trả lời:

a. LRU: Có 10 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	1	1	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3
			4	4	4	4	6	6	6	6	6	7	7	7	7	1	1	1	1
*	*	*	*			*	*				*	*	*			*			

b. FIFO: Có 14 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	5	5	5	5	5	3	3	3	3	3	1	1	1	1
	2	2	2	2	2	2	6	6	6	6	6	7	7	7	7	7	7	3	3
		3	3	3	3	3	3	2	2	2	2	2	6	6	6	6	6	6	6
			4	4	4	4	4	4	1	1	1	1	1	1	2	2	2	2	2
*	*	*	*			*	*	*	*		*	*	*		*	*		*	

c. OPT: Có 8 lỗi trang

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6
*	*	*	*			*	*					*				*			