Chương 4

QUẢN LÝ BỘ NHỚ

Nội dung chương 4

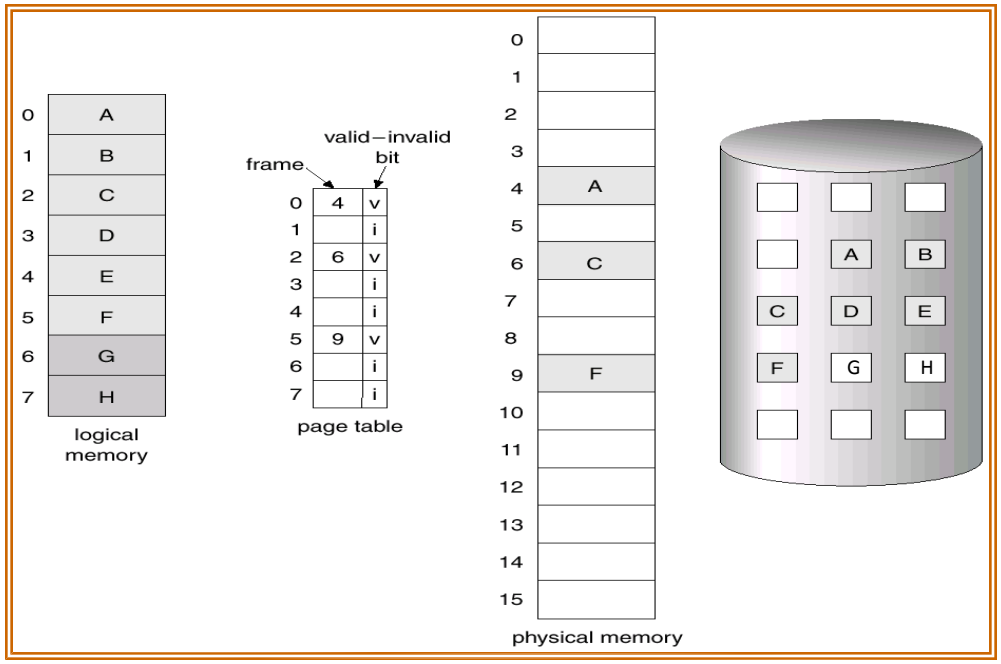
1. Địa chỉ và các vấn đề liên quan.
2. Một số cách tổ chức chương trình.
3. Phân chương bộ nhớ.
4. Phân đoạn bộ nhớ.
5. Phân trang bộ nhớ.
6. Bộ nhớ ảo.
7. Cấp phát khung trang.
8. Tình trạng trì trệ.

BỘ NHỚ ẢO

- Dùng bộ nhớ phụ lưu trữ tiến trình, các phần của tiến trình được chuyển vào-ra giữa bộ nhớ chính và bộ nhớ phụ.
- Phân trang theo yêu cầu (Demand paging).
- Phân đoạn theo yêu cầu (Demand segmentation)

BỘ NHỚ ẢO

- Phân trang theo yêu cầu (Demand paging)

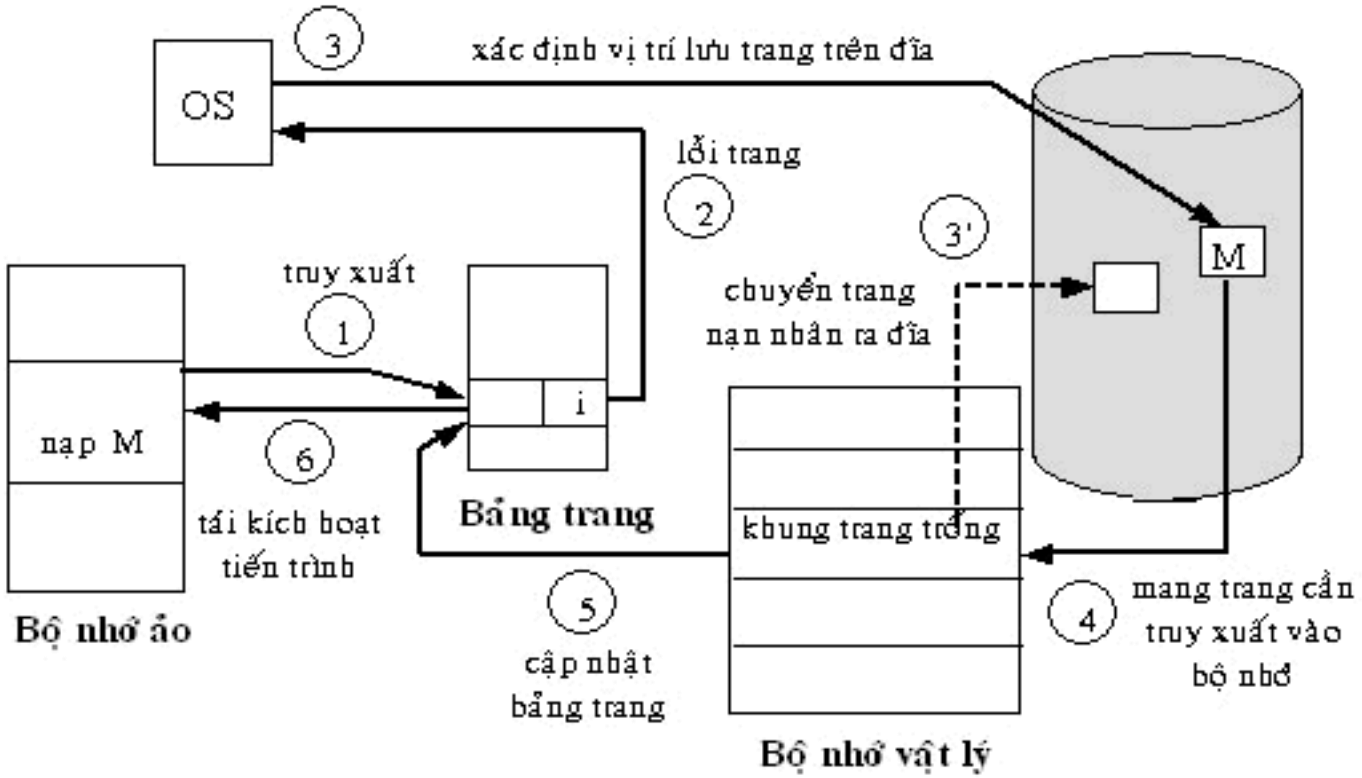


Trường chứa bit "kiểm tra“:

- ✓ 1 (valid) là trang đang ở trong bộ nhớ chính
- ✓ 0 (invalid) là trang đang được lưu trên bộ nhớ phụ hoặc trang không thuộc tiến trình

BỘ NHỚ ẢO

- Chuyển địa chỉ ảo (p,d) thành địa chỉ vật lý



BỘ NHỚ ẢO

- Thay thế trang

  Bit "cập nhật" (dirty bit):

  - ✓ 1: nội dung trang có bị sửa đổi.
  - ✓ 0: nội dung trang không bị thay đổi.

số hiệu khung trang chứa trang hoặc địa chỉ trên đĩa của trang	bit nhận diện trang có trong bộ nhớ (bit valid-invalid)	bit nhận diện trang có thay đổi (bit dirty)

BỘ NHỚ ẢO

- Thời gian thực hiện một yêu cầu truy xuất bộ nhớ
  - P: xác suất xảy ra lỗi trang ($0 \leq p \leq 1$).
  - Memory access (ma): thời gian một lần truy xuất bộ nhớ.
  - Effective Access Time (EAT): thời gian thực hiện một yêu cầu truy xuất bộ nhớ.
  - Page fault overhead (pfo) :thời gian xử lý một lỗi trang.
  - Swap page in (spi): thời gian chuyển trang từ đĩa vào bộ nhớ.
  - Swap page out (spo): thời gian chuyển trang ra đĩa (swap page out có thể bằng 0).

Restart overhead (ro): thời gian tái khởi động lại việc truy xuất bộ nhớ.

$$EAT = (1 - p) \times ma + p\ (pfo + [spo] + spi + ro)$$

Ví dụ: Thời gian một lần truy xuất bộ nhớ là 1 micro second và giả sử 40% trang được chọn đã thay đổi nội dung và thời gian hoán chuyển trang ra/vào là 10 mili second . Tính ETA.

$$EAT = (1 - p) + p\ (pfo + 10000 * 0.4 + 10000 + ro) \text{ micro second}$$

BỘ NHỚ ẢO

❖ Các thuật toán chọn trang nạn nhân

Trang "nạn nhân": trang mà sau khi thay thế sẽ gây ra ít lỗi trang nhất.

Thuật toán FIFO (First In First Out)

Thuật toán tối ưu (Optimal Page Replacement Algorithm)

Thuật toán LRU (Least-recently-used)

Các thuật toán xấp xỉ LRU

- ✓ Thuật toán với các bít history
- ✓ Thuật toán cơ hội thứ hai
- ✓ Thuật toán cơ hội thứ hai nâng cao (Not Recently Used Page Replacement Algorithm: NRU)

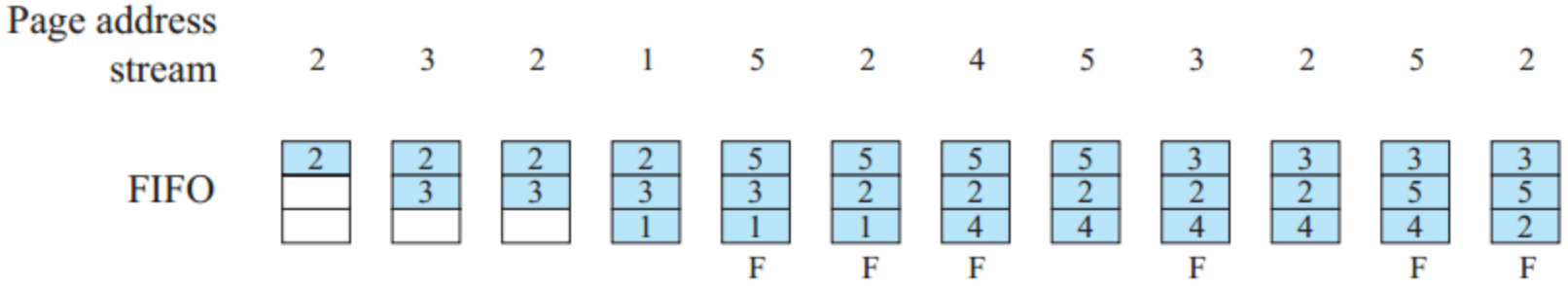
Các thuật toán thống kê

- ✓ Thuật toán LFU (least frequently used)
- ✓ Thuật toán MFU (most frequently used)

Bộ nhớ ảo

Giải thuật thay trang *FIFO*

- Xem các frame được cấp phát cho process như circular buffer
  - ✓ Khi bộ đệm đầy, trang nhớ cũ nhất sẽ được thay thế: FIFO
  - ✓ Một trang nhớ hay được dùng sẽ thường là trang cũ nhất
    $\Rightarrow$ hay bị thay thế bởi giải thuật FIFO
  - ✓ Đơn giản: cần một con trỏ xoay vòng các frame của process



Bộ nhớ ảo

- Nghịch lý Belady

  Xét tiến trình truy xuất chuỗi trang theo thứ tự sau:
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Nếu 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
*	*	*	*	*	*	*			*	*	

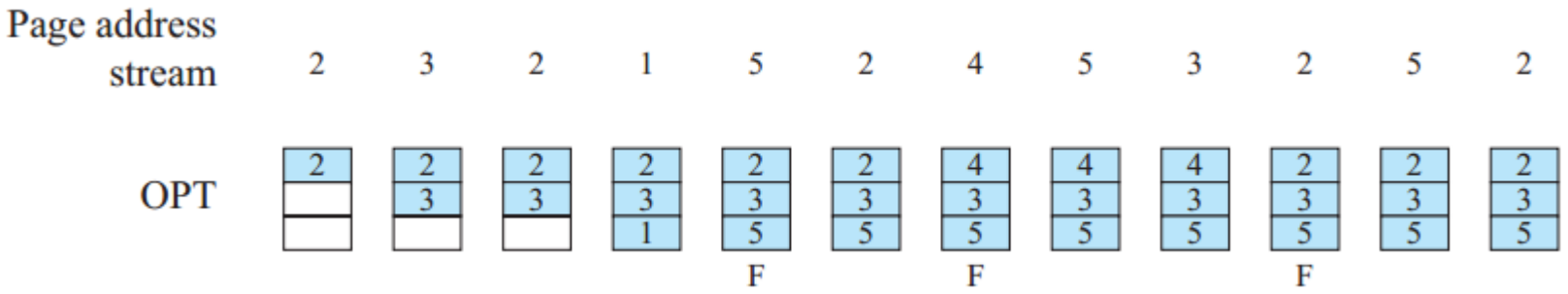
1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3
*	*	*	*			*	*	*	*	*	*

Nếu sử dụng 4 khung trang, sẽ có 10 lỗi trang

Bộ nhớ ảo

Giải thuật thay trang OPT - Optimal

- Thay thế trang nhớ sẽ được tham chiếu trễ nhất trong tương lai
- Ví dụ: một process có 5 trang, và được cấp 3 frame



Bộ nhớ ảo

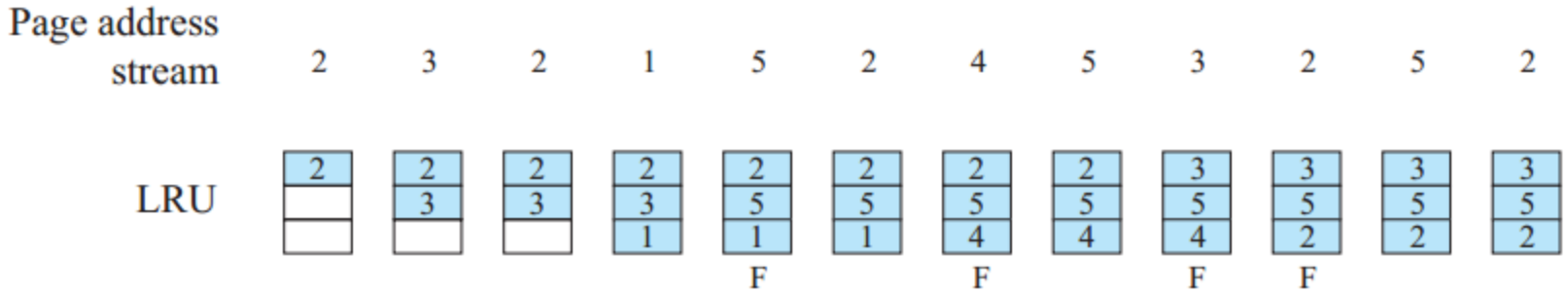
Giải thuật thay trang OPT - Optimal

- ✓ Số lượng lỗi trang phát sinh là thấp nhất.
- ✓ Không bị nghịch lý Belady.
- ✓ Khó cài đặt
- ✓ Phù hợp với hệ điều hành cho thiết bị gia dụng

Bộ nhớ ảo

Giải thuật thay trang *Least Recently Used* (LRU)

- Thay thế trang nhớ không được tham chiếu lâu nhất
- Ví dụ: một process có 5 trang, và được cấp 3 frame



Bộ nhớ ảo

❖ Cài đặt thuật toán LRU

➢ ***Sử dụng bộ đếm***

- ✓ Cấu trúc phần tử trong bảng trang: thêm trường ghi nhận "thời điểm truy xuất gần nhất".
- ✓ Cấu trúc của CPU: thêm một thanh ghi đếm (counter).

số hiệu khung trang chứa trang hoặc địa chỉ trang trên đĩa	bit valid - invalid	bit dirty	thời điểm truy xuất gần nhất

➢ ***Sử dụng danh sách liên kết***

- ✓ Trang ở cuối danh sách là trang được truy xuất gần nhất
- ✓ Trang ở đầu danh sách là trang lâu nhất chưa được sử dụng

Bộ nhớ ảo

❖ Các thuật toán xấp xỉ LRU

➢Mỗi phần tử trong bảng trang có thêm bit reference:

✓ được khởi gán là 0 bởi hđh.

✓ được phần cứng gán là 1 mỗi lần trang tương ứng được truy cập

số hiệu khung trang chứa trang hoặc địa chỉ trang trên đĩa	bit valid-invalid	bit dirty	bit reference

➢ Các thuật toán xấp xỉ LRU:

✓ Thuật toán với các bit history.

✓ Thuật toán cơ hội thứ hai.

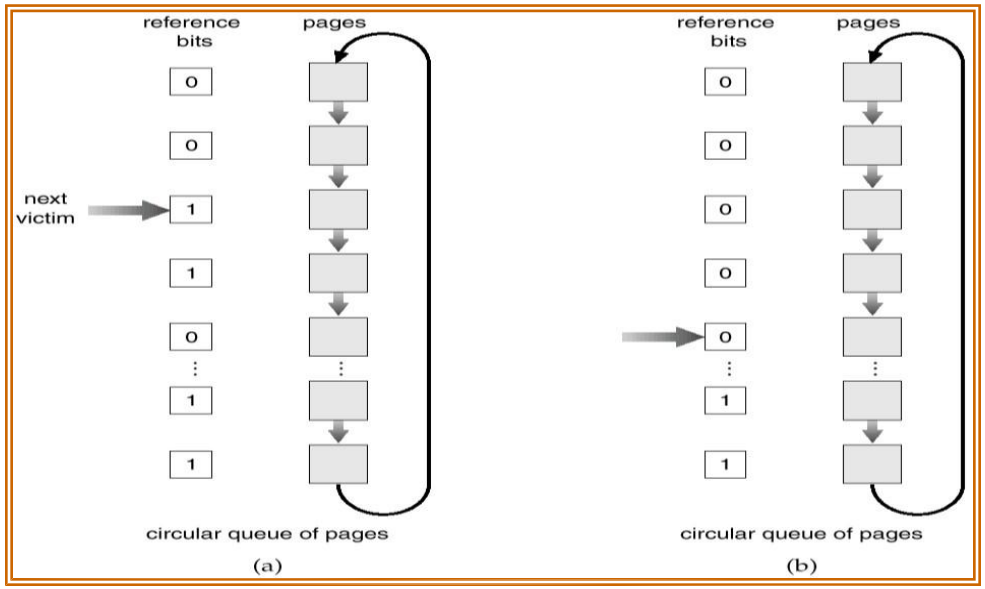
✓ Thuật toán cơ hội thứ hai nâng cao (Not Recently Used Page Replacement Algorithm: NRU)

Bộ nhớ ảo

- Thuật toán với các bit history
  - Mỗi trang sử dụng thêm 8 bit lịch sử (history).
  - Cập nhật các bít history:
    - ✓ dịch các bit history sang phải 1 vị trí để loại bỏ bit thấp nhất.
    - ✓ đặt bit reference của mỗi trang vào bit cao nhất trong 8 bit history của trang đó.
  - 8 bit history sẽ lưu trữ tình hình truy xuất đến trang trong 8 chu kỳ cuối cùng.
  - Trang "nạn nhân" là trang có giá trị history nhỏ nhất.

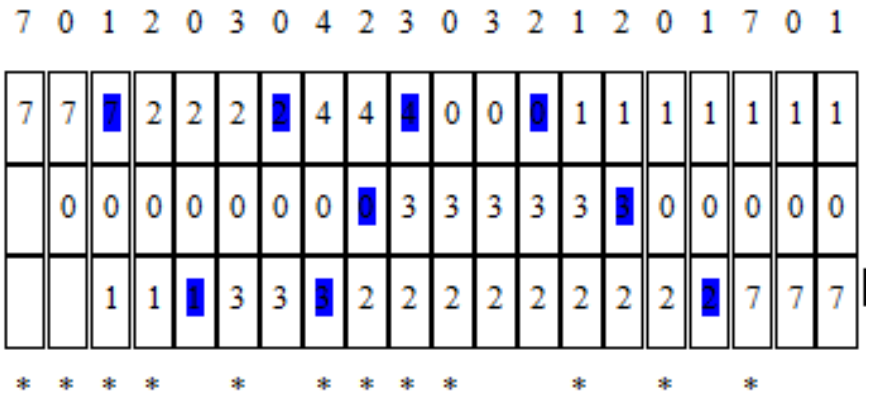
Bộ nhớ ảo

- Thuật toán cơ hội thứ hai:
  - Tìm một trang theo nguyên tắc FIFO.
  - Kiểm tra bit reference của trang đó.
    - ✓ Nếu bit reference là 0, chọn trang này.
    - ✓ Nếu bit reference là 1 thì gán lại là 0 rồi tìm trang FIFO tiếp theo



Bộ nhớ ảo

- Thuật toán cơ hội thứ hai:
  Ví dụ:



7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7(1)	7(1)	7(1)	0(0)	1(0)	2(1)	2(1)	3(0)	0(0)	4(1)	2(0)	2(0)	0(1)	3(0)	3(0)	1(0)	2(1)	0(0)	1(0)	7(1)
	0(1)	0(1)	1(0)	2(1)	0(1)	3(1)	0(0)	4(1)	2(1)	3(0)	0(1)	3(1)	2(0)	1(0)	2(1)	0(1)	1(0)	7(1)	0(1)
		1(1)	2(1)	0(1)	3(1)	0(1)	4(1)	2(1)	3(1)	0(1)	3(1)	2(1)	1(0)	2(1)	0(1)	1(1)	7(1)	0(1)	1(1)

Bộ nhớ ảo

- Thuật toán cơ hội thứ hai nâng cao (Not Recently Used Page Replacement Algorithm - NRU):

➢Lớp 1 (0,0): gồm những trang có (ref,dirty) = (0,0). (độ ưu tiên thấp nhất)

- ✓ Không được truy xuất gần đây và không bị sửa đổi
- ✓ Tốt nhất để thay thế.

➢Lớp 2 (0,1):

- ✓ Không truy xuất gần đây nhưng đã bị sửa đổi.
- ✓ Trường hợp này không thật tốt, vì trang cần được lưu trữ lại trước khi thay thế.

Bộ nhớ ảo

- Thuật toán cơ hội thứ hai nâng cao (Not Recently Used Page Replacement Algorithm - NRU):
  - Lớp 3 (1,0):
    - ✓ Được truy xuất gần đây, nhưng không bị sửa đổi.
    - ✓ Trang có thể nhanh chóng được tiếp tục được sử dụng.
  - Lớp 4 (1,1): (độ ưu tiên cao nhất)
    - ✓ trang được truy xuất gần đây, và bị sửa đổi.
    - ✓ Trang có thể nhanh chóng được tiếp tục được sử dụng và trước khi thay thế cần phải được lưu trữ lại.
- Trang "nạn nhân": trang đầu tiên tìm thấy trong lớp có độ ưu tiên thấp nhất.

Bộ nhớ ảo

- Các thuật toán thống kê
  - Biến đếm: lưu số lần truy xuất đến một trang.
  - Thuật toán LFU (least frequently used):
    - ✓ Thay thế trang có giá trị biến đếm nhỏ nhất, nghĩa là trang ít được sử dụng nhất.
  - Thuật toán MFU (most frequently used):
    - ✓ Thay thế trang có giá trị biến đếm lớn nhất, nghĩa là trang được sử dụng nhiều nhất.

Bộ nhớ ảo

- Chiến lược cấp phát khung trang
  - **Cấp phát ngang bằng:**
    - ✓ m khung trang và n tiến trình.
    - ✓ Mỗi tiến trình được cấp m/n khung trang.
  - **Cấp phát theo tỷ lệ kích thước:**

$$a_i = \frac{s_i}{S} \times m$$

s_i: kích thước của tiến trình p_i

$S = \sum s_i$ là tổng kích thước của tất cả tiến trình

m: số lượng khung trang có thể sử dụng

a_i: số khung trang được cấp phát cho tiến trình p_i

Ví dụ: Tiến trình 1 = 10K, tiến trình 2 = 127K, có 62 khung trang trống. Khi đó có thể cấp cho
tiến trình 1: 10/137 x 62 ~ 4 khung
tiến trình 2: 127/137 x62 ~ 57 khung

  - **Cấp phát theo tỷ lệ độ ưu tiên**

Bộ nhớ ảo

- Thay thế trang
  - **Thay thế toàn cục**
    - ✓ Chọn trang "nạn nhân" từ tập tất cả các khung trang trong hệ thống.
    - ✓ Có nhiều khả năng lựa chọn hơn.
    - ✓ Số khung trang cấp cho một tiến trình có thể thay đổi.
    - ✓ Các tiến trình không thể kiểm soát được tỷ lệ phát sinh lỗi trang của mình.
  - **Thay thế cục bộ**
    - ✓ Chỉ chọn trang thay thế trong tập các khung trang được cấp cho tiến trình phát sinh lỗi trang.
    - ✓ Số khung trang cấp cho một tiến trình sẽ không thay đổi

Tình trạng trì trệ (thrashing)

- Là hiện tượng các trang nhớ của một process bị hoán chuyển vào/ra liên tục
- Nếu một process không có đủ số frame cần thiết thì tỉ số page faults/sec rất cao. Điều này khiến giảm hiệu suất CPU rất nhiều.
- Mô hình tập làm việc (working set).

Tình trạng trì trệ (thrashing)

- Mô hình tập làm việc (working set)
  - WSSi(Δ, t): số phần tử của tập working set của tiến trình Pi tại thời điểm t.
    - ✓ Tập các trang được tiến trình truy xuất đến trong Δ lần truy cập cuối cùng tính tại thời điểm t.
  - m: số khung trang trống.
  - D = ∑WSSi: tổng số khung trang yêu cầu cho toàn hệ thống.
  - Tại thời điểm t: cấp cho Pi số khung trang bằng WSSi(Δ, t-1).
  - D>m: Trì trệ hệ thống