



HỆ ĐIỀU HÀNH

CHƯƠNG 8: BỘ NHỚ ẢO

Trình bày các khái niệm cơ bản về bộ nhớ ảo, kỹ thuật cài đặt bộ nhớ ảo, một số vấn đề trong bộ nhớ ảo
như cấp phát khung trang và tình trạng trì trệ



Các nội dung đã học

- Chương 1: Tổng quan về hệ điều hành
- Chương 2: Cấu trúc hệ điều hành
- Chương 3: Quản lý tiến trình
- Chương 4: Định thời CPU
- Chương 5: Đồng bộ hóa tiến trình
- Chương 6: Tắc nghẽn
- Chương 7: Quản lý bộ nhớ
- **Chương 8: Bộ nhớ ảo**
- Chương 9: Hệ điều hành Linux và Hệ điều hành Windows



NỘI DUNG

1. Hiểu được các khái niệm tổng quan về bộ nhớ ảo
2. Hiểu và vận dụng kỹ thuật cài đặt bộ nhớ ảo demand paging
3. Hiểu được một số vấn đề trong bộ nhớ ảo: cấp phát frames và thrashing



NỘI DUNG

1. Tổng quan về bộ nhớ ảo
2. Cài đặt bộ nhớ ảo: Demand Paging
3. Các giải thuật thay trang (Page Replacement Algorithms)
4. Vấn đề cấp phát Frames
5. Vấn đề Thrashing



Tổng quan về bộ nhớ ảo

1



Nhắc lại về dynamic loading

- Cơ chế: chỉ khi nào cần được gọi đến thì một thủ tục mới được nạp vào bộ nhớ chính ⇒ tăng độ hiệu dụng của bộ nhớ bởi vì các thủ tục không được gọi đến sẽ không chiếm chỗ trong bộ nhớ.
- Rất hiệu quả trong trường hợp tồn tại khối lượng lớn mã chương trình có tần suất sử dụng thấp, không được sử dụng thường xuyên (ví dụ các thủ tục xử lý lỗi).
- Hỗ trợ từ hệ điều hành:
 - Thông thường, user chịu trách nhiệm thiết kế và hiện thực các chương trình có dynamic loading.
 - Hệ điều hành chủ yếu cung cấp một số thủ tục thư viện hỗ trợ, tạo điều kiện dễ dàng hơn cho lập trình viên.



8.1 Tổng quan về bộ nhớ ảo

- Nhận xét: không phải tất cả các phần của một tiến trình cần thiết phải được nạp vào bộ nhớ chính tại cùng một thời điểm.
- Ví dụ:
 - Đoạn mã điều khiển các lỗi hiếm khi xảy ra.
 - Các arrays, list, tables được cấp phát bộ nhớ (cấp phát tĩnh) nhiều hơn yêu cầu thực sự.
 - Một số tính năng ít khi được dùng của một chương trình.
 - Cả chương trình thì cũng có đoạn code chưa cần dùng.
- Bộ nhớ ảo (virtual memory): Bộ nhớ ảo là một kỹ thuật cho phép xử lý một tiến trình không được nạp toàn bộ vào bộ nhớ vật lý.



8.1 Tổng quan về bộ nhớ ảo

- Ưu điểm của bộ nhớ ảo:
 - Số lượng tiến trình trong bộ nhớ nhiều hơn.
 - Một tiến trình có thể thực thi ngay cả khi kích thước của nó lớn hơn bộ nhớ thực.
 - Giảm nhẹ công việc của lập trình viên.
- Không gian tráo đổi giữa bộ nhớ chính và bộ nhớ phụ (swap space)
- Ví dụ:
 - swap partition trong Linux
 - file pagefile.sys trong Windows



Cài đặt bộ nhớ ảo

8.2.1 Cài đặt bộ nhớ ảo

2



8.2.1 Cài đặt bộ nhớ ảo

- Có hai kỹ thuật:
 - Phân trang theo yêu cầu (Demand Paging)
 - Phân đoạn theo yêu cầu (Demand Segmentation)
- Phần cứng memory management phải hỗ trợ paging và/hoặc segmentation.
- OS phải quản lý sự di chuyển của trang/đoạn giữa bộ nhớ chính và bộ nhớ thứ cấp.
- Trong chương này:
 - Chỉ quan tâm đến paging.
 - Phần cứng hỗ trợ hiện thực bộ nhớ ảo.
 - Chỉ tập trung vào các giải thuật của hệ điều hành.

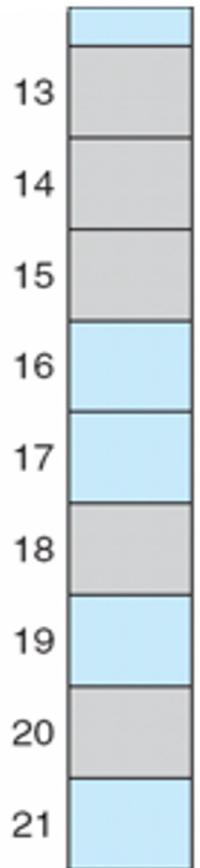
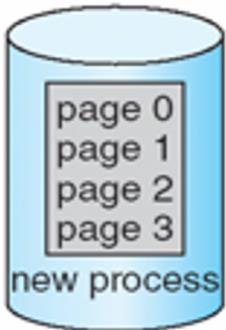


8.2.1 Cài đặt bộ nhớ ảo

Cơ chế phân trang

free-frame list

14
13
18
20
15

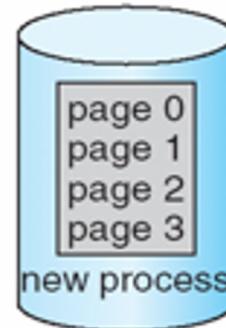


(a)

Before allocation

free-frame list

15



new-process page table

0	14
1	13
2	18
3	20



(b)

After allocation



Cài đặt bộ nhớ ảo

8.2.2 Phân trang theo yêu cầu

2



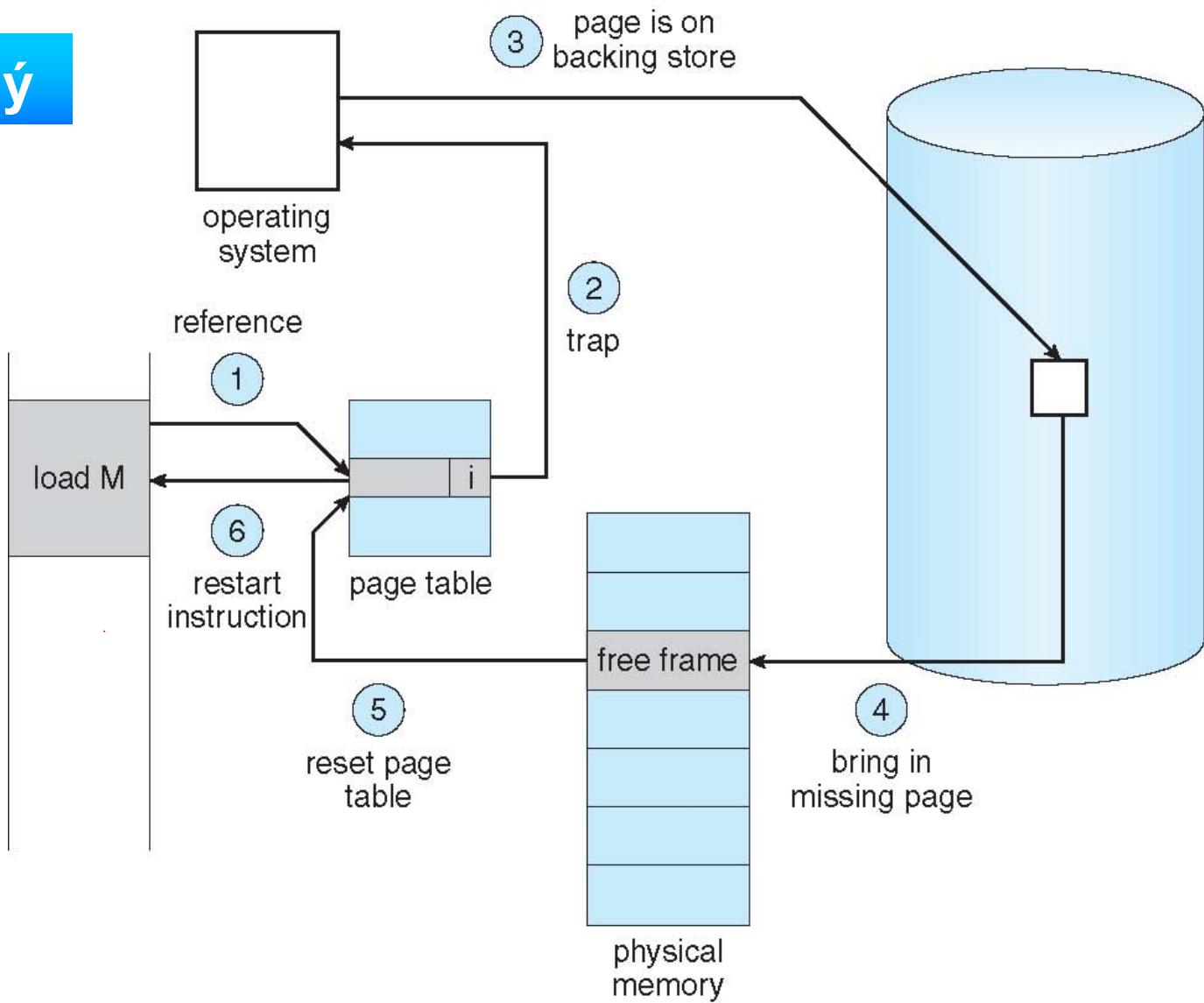
8.2.2 Phân trang theo yêu cầu

- 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 tiến trình 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 tiến trình 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 tiến trình về trạng thái ready.



8.2.2 Phân trang theo yêu cầu

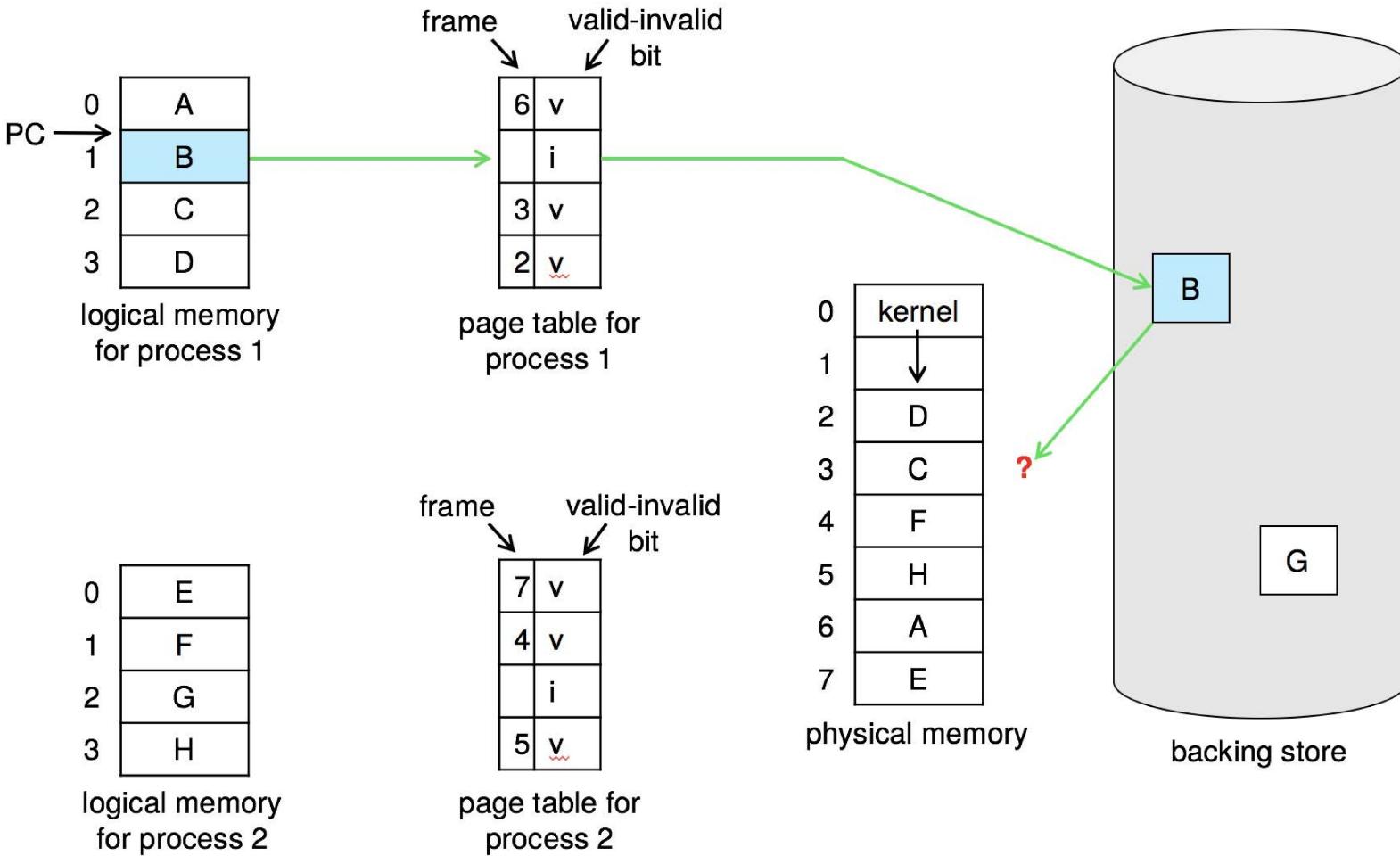
Lỗi trang và các bước xử lý





8.2.2 Phân trang theo yêu cầu

Khi cần thay thế trang





Cài đặt bộ nhớ ảo

8.2.3 Thay thế trang nhớ

2



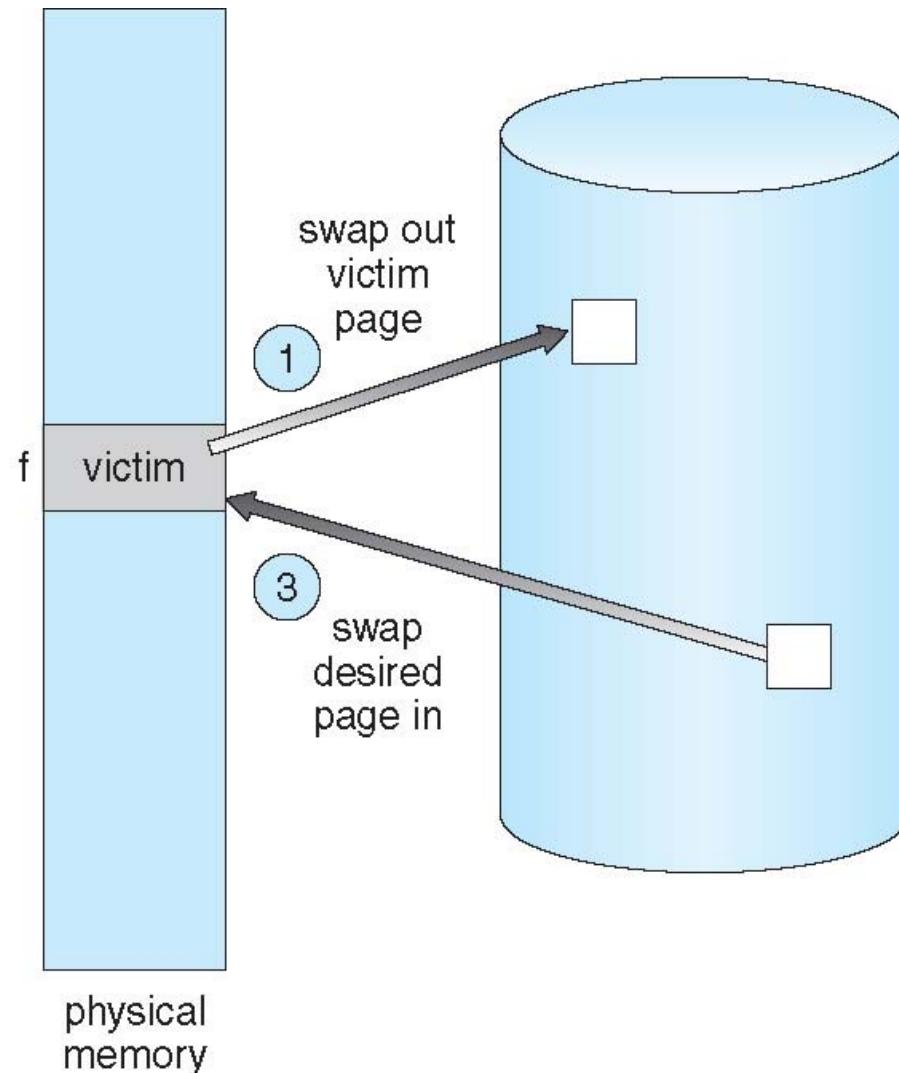
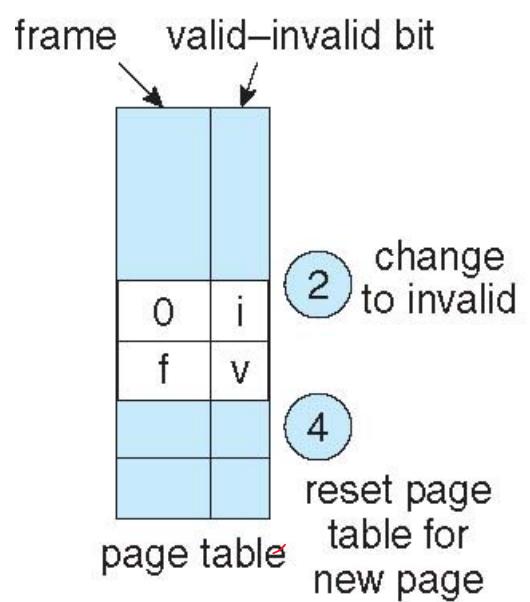
8.2.3 Thay thế trang nhớ

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.



8.2.3 Thay thế trang nhớ





8.2.3 Thay thế trang nhớ

Các vấn đề chủ yếu

- Frame-allocation algorithm
 - Cấp phát cho tiến trình bao nhiêu frame của bộ nhớ thực?
- Page-replacement algorithm
 - Chọn frame của tiến trình sẽ được thay thế trang nhớ.
 - Mục tiêu: số lượng page-fault nhỏ nhất.
 - Được đánh giá bằng cách thực thi giải thuật đối với một chuỗi tham chiếu bộ nhớ (memory reference string) và xác định số lần xảy ra page fault.



8.2.3 Thay thế trang nhớ

Chuỗi tham chiếu bộ nhớ (trang nhớ)

Ví dụ

Thứ tự tham chiếu các địa chỉ nhớ, với page size = 100:

0098, 0432, 0201, 0612, 0302, 0103, 0104, 0101, 0611, 0102, 0103, 0104, 0101,
0610, 0102, 0103, 0104, 0101, 0609, 0102, 0105

các trang nhớ sau được tham chiếu lần lượt = chuỗi tham chiếu bộ nhớ (trang nhớ)

0, 4, 2, 6, 3,

1, 1, 1, 6, 1,

1, 1, 1, 6, 1,

1, 1, 1, 6, 1,

1



Các giải thuật thay trang

8.3.1 Các giải thuật thay trang

3



8.3.1 Các giải thuật thay trang

- Giải thuật thay trang FIFO
- Giải thuật thay trang OPT
- Giải thuật thay trang LRU
- Các dữ liệu cần biết ban đầu:
 - Số khung trang
 - Tình trạng ban đầu
 - Chuỗi tham chiếu



Các giải thuật thay trang

8.3.2 Giải thuật thay trang FIFO

3



8.3.2 Giải thuật thay trang FIFO

- Xét một tiến trình có 8 trang và chuỗi tham chiếu bộ nhớ cần truy xuất như sau:

7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

- Số khung trang là 3
- Ban đầu các khung trang đều trống



8.3.2 Giải thuật thay trang FIFO

Giải thuật thay trang FIFO thay thế trang nhớ có thời gian được nạp vào bộ nhớ sớm nhất trong các trang nhớ.

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



Các giải thuật thay trang

8.3.3 Nghịch lý Belady

3



8.3.3 Nghịch lý Belady

Sử dụng 3 khung 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	1	3	3	3
	3	3	3	2	2	2	2	2	4	4	
*	*	*	*	*	*	*		*	*		

Sử dụng 4 khung 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	2	1	1	1	1	5
	3	3	3	3	3	3	3	2	2	2	2
	4	4	4	4	4	4	4	4	3	3	3
*	*	*	*			*	*	*	*	*	*



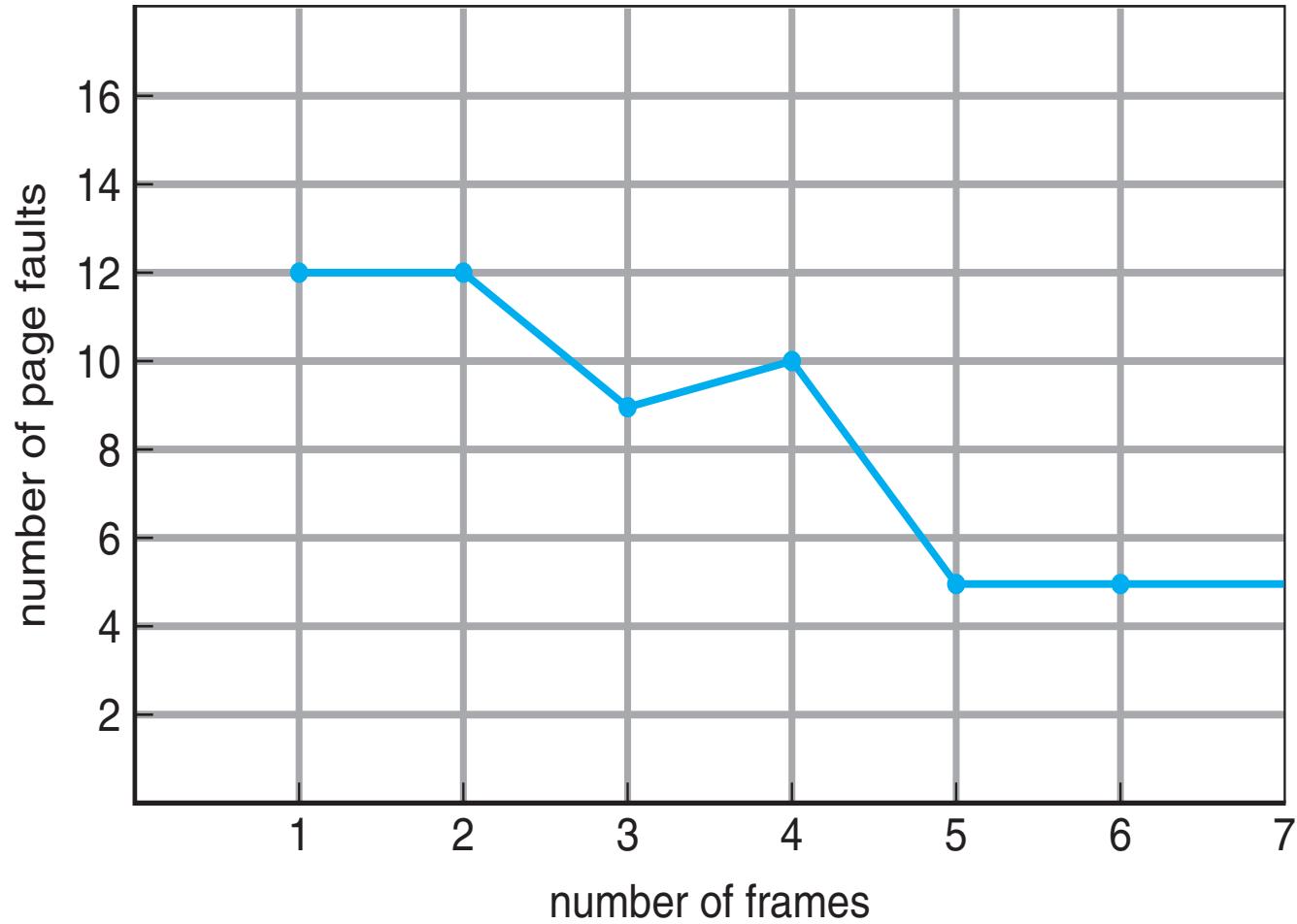
9 lõi trang



10 lõi trang



8.3.3 Nghịch lý Belady



Bất thường (anomaly) Belady: số page fault tăng mặc dù tiến trình đã được cấp nhiều frame hơn.



Các giải thuật thay trang

8.3.4 Giải thuật thay trang OPT

3



8.3.4 Giải thuật thay trang OPT

Giải thuật thay trang OPT thay thế trang nhớ sẽ được tham chiếu trễ nhất trong tương lai \Rightarrow cần phải biết trước các trang sẽ được tham chiếu trong tương lai.

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	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	3	1	1	1	1	1	1	1
*	*	*	*	*	*	*			*			*			*				



Các giải thuật thay trang

8.3.5 Giải thuật thay trang LRU

3



8.3.5 Giải thuật thay trang LRU

- Mỗi trang được ghi nhận (trong bảng phân trang) thời điểm được tham chiếu \Rightarrow trang LRU là trang nhớ có thời điểm tham chiếu nhỏ nhất (OS tốn chi phí tìm kiếm trang nhớ LRU này mỗi khi có page fault).
- Do vậy, LRU cần sự hỗ trợ của phần cứng và chi phí cho việc tìm kiếm. Ít CPU cung cấp đủ sự hỗ trợ phần cứng cho giải thuật LRU.

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



8.3.5 Giải thuật thay trang LRU

So sánh các giải thuật thay trang LRU và FIFO

Page address
stream

	2	3	2	1	5	2	4	5	3	2	5	2																																				
LRU	<table border="1"><tr><td>2</td></tr><tr><td>3</td></tr><tr><td></td></tr></table>	2	3		<table border="1"><tr><td>2</td></tr><tr><td>3</td></tr><tr><td></td></tr></table>	2	3		<table border="1"><tr><td>2</td></tr><tr><td>3</td></tr><tr><td>1</td></tr></table>	2	3	1	<table border="1"><tr><td>2</td></tr><tr><td>3</td></tr><tr><td>1</td></tr></table>	2	3	1	<table border="1"><tr><td>2</td></tr><tr><td>5</td></tr><tr><td>1</td></tr></table>	2	5	1	<table border="1"><tr><td>2</td></tr><tr><td>5</td></tr><tr><td>1</td></tr></table>	2	5	1	<table border="1"><tr><td>2</td></tr><tr><td>5</td></tr><tr><td>4</td></tr></table>	2	5	4	<table border="1"><tr><td>2</td></tr><tr><td>5</td></tr><tr><td>4</td></tr></table>	2	5	4	<table border="1"><tr><td>3</td></tr><tr><td>5</td></tr><tr><td>4</td></tr></table>	3	5	4	<table border="1"><tr><td>3</td></tr><tr><td>5</td></tr><tr><td>2</td></tr></table>	3	5	2	<table border="1"><tr><td>3</td></tr><tr><td>5</td></tr><tr><td>2</td></tr></table>	3	5	2	<table border="1"><tr><td>3</td></tr><tr><td>5</td></tr><tr><td>2</td></tr></table>	3	5	2
2																																																
3																																																
2																																																
3																																																
2																																																
3																																																
1																																																
2																																																
3																																																
1																																																
2																																																
5																																																
1																																																
2																																																
5																																																
1																																																
2																																																
5																																																
4																																																
2																																																
5																																																
4																																																
3																																																
5																																																
4																																																
3																																																
5																																																
2																																																
3																																																
5																																																
2																																																
3																																																
5																																																
2																																																
FIFO	<table border="1"><tr><td>2</td></tr><tr><td>3</td></tr><tr><td></td></tr></table>	2	3		<table border="1"><tr><td>2</td></tr><tr><td>3</td></tr><tr><td></td></tr></table>	2	3		<table border="1"><tr><td>2</td></tr><tr><td>3</td></tr><tr><td>1</td></tr></table>	2	3	1	<table border="1"><tr><td>2</td></tr><tr><td>3</td></tr><tr><td>1</td></tr></table>	2	3	1	<table border="1"><tr><td>5</td></tr><tr><td>3</td></tr><tr><td>1</td></tr></table>	5	3	1	<table border="1"><tr><td>5</td></tr><tr><td>2</td></tr><tr><td>1</td></tr></table>	5	2	1	<table border="1"><tr><td>5</td></tr><tr><td>2</td></tr><tr><td>4</td></tr></table>	5	2	4	<table border="1"><tr><td>5</td></tr><tr><td>2</td></tr><tr><td>4</td></tr></table>	5	2	4	<table border="1"><tr><td>3</td></tr><tr><td>2</td></tr><tr><td>4</td></tr></table>	3	2	4	<table border="1"><tr><td>3</td></tr><tr><td>2</td></tr><tr><td>4</td></tr></table>	3	2	4	<table border="1"><tr><td>3</td></tr><tr><td>5</td></tr><tr><td>4</td></tr></table>	3	5	4	<table border="1"><tr><td>3</td></tr><tr><td>5</td></tr><tr><td>2</td></tr></table>	3	5	2
2																																																
3																																																
2																																																
3																																																
2																																																
3																																																
1																																																
2																																																
3																																																
1																																																
5																																																
3																																																
1																																																
5																																																
2																																																
1																																																
5																																																
2																																																
4																																																
5																																																
2																																																
4																																																
3																																																
2																																																
4																																																
3																																																
2																																																
4																																																
3																																																
5																																																
4																																																
3																																																
5																																																
2																																																



Vấn đề cấp phát Frames

8.4.1 Số lượng frame cấp cho tiến trình

4



8.4.1 Số lượng frame cấp cho tiến trình

- OS phải quyết định cấp cho mỗi tiến trình bao nhiêu frame.
 - Cấp ít frame \Rightarrow nhiều page fault.
 - Cấp nhiều frame \Rightarrow giảm mức độ multiprogramming.
- Chiến lược cấp phát tĩnh (fixed-allocation)
 - Số frame cấp cho mỗi tiến trình không đổi, được xác định vào thời điểm loading và có thể tùy thuộc vào từng ứng dụng (kích thước của nó,...).
- Chiến lược cấp phát động (variable-allocation)
 - Số frame cấp cho mỗi tiến trình có thể thay đổi trong khi nó chạy:
 - Nếu tỷ lệ page-fault cao \Rightarrow cấp thêm frame.
 - Nếu tỷ lệ page-fault thấp \Rightarrow giảm bớt frame.
 - Hệ điều hành phải mất chi phí để ước định các tiến trình.



Vấn đề cấp phát Frames

8.4.2 Chiến lược cấp phát tĩnh

4



8.4.2 Chiến lược cấp phát tĩnh

- Cấp phát bằng nhau:

Ví dụ, có 100 frame và 5 tiến trình → mỗi tiến trình được 20 frame

- Cấp phát theo tỉ lệ: dựa vào kích thước tiến trình

$$s_i = \text{size of process } p_i$$

$$S = \sum s_i$$

m = total number of frames

$$a_i = \text{allocation for } p_i = \frac{s_i}{S} \times m$$

- Cấp phát theo độ ưu tiên

Ví dụ: $m = 64$

$$s_1 = 10$$

$$s_2 = 127$$

$$a_1 = \frac{10}{137} \times 64 \approx 5$$

$$a_2 = \frac{127}{137} \times 64 \approx 59$$



Vấn đề Thrashing

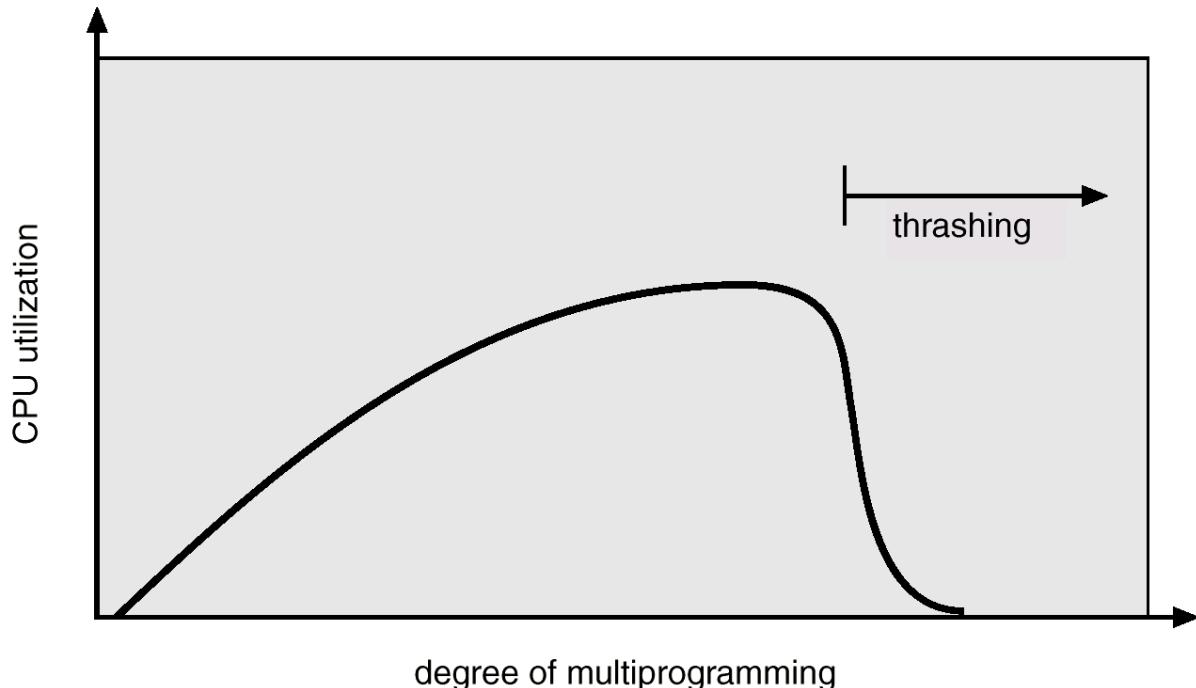
8.5.1 Trì trệ trên toàn bộ hệ thống

5



8.5.1 Trì trệ trên toàn bộ hệ thống

- Nếu một tiến trình không có đủ số frame cần thiết thì tỉ số page faults/sec rất cao.
- Thrashing: hiện tượng các trang nhớ của một tiến trình bị hoán chuyển vào/ra liên tục.





Vấn đề Thrashing

8.5.2 Mô hình cục bộ

5



8.5.2 Mô hình cục bộ

- Để hạn chế thrashing, hệ điều hành phải cung cấp cho tiến trình càng “đủ” frame càng tốt. Bao nhiêu frame thì đủ cho một tiến trình thực thi hiệu quả?
- Nguyên lý locality (locality principle)
 - Locality là tập các trang được tham chiếu gần nhau.
 - Một tiến trình gồm nhiều locality, và trong quá trình thực thi, tiến trình sẽ chuyển từ locality này sang locality khác.
 - Vì sao hiện tượng thrashing xuất hiện?
Khi Σ size of locality > memory size



Vấn đề Thrashing

8.5.3 Giải pháp tập làm việc

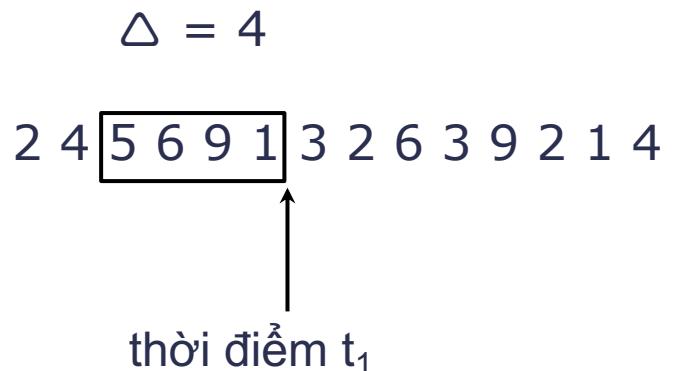
5



8.5.3 Giải pháp tập làm việc

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

chuỗi tham khảo
trang nhớ

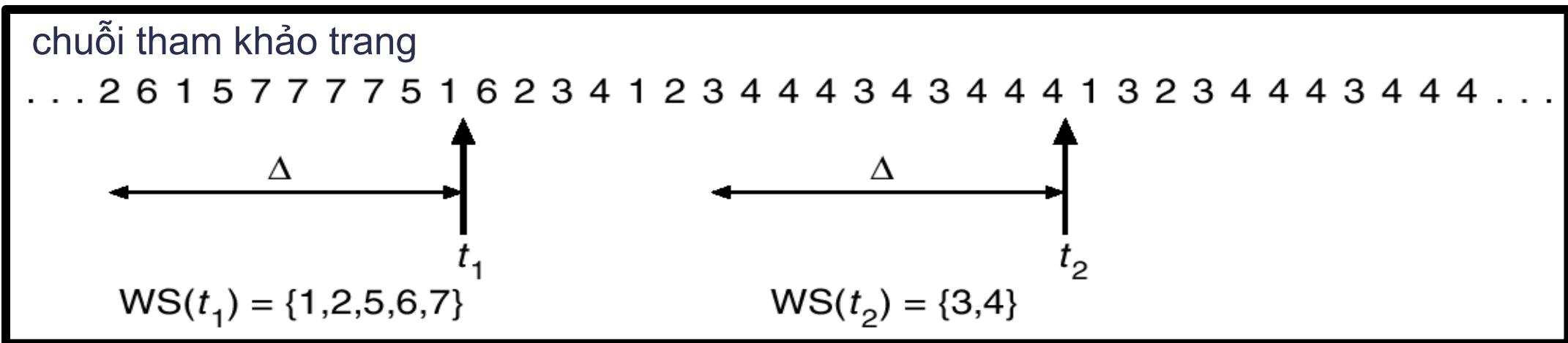




8.5.3 Giải pháp tập làm việc

- Định nghĩa: Working set của tiến trình P_i , ký hiệu WS_i , là tập gồm Δ các trang được sử dụng gần đây nhất.

Ví dụ: $\Delta = 10$ và



- Nhận xét:
 - Δ quá nhỏ \Rightarrow không đủ bao phủ toàn bộ locality.
 - Δ quá lớn \Rightarrow bao phủ nhiều locality khác nhau.
 - $\Delta = \infty$ \Rightarrow bao gồm tất cả các trang được sử dụng.
- Dùng working set của một tiến trình để xấp xỉ locality của nó.



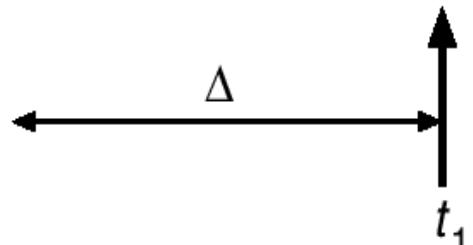
8.5.3 Giải pháp tập làm việc

- Định nghĩa: WSS_i là kích thước của working set của P_i :
 - $WSS_i = \text{số lượng các trang trong } WS_i$

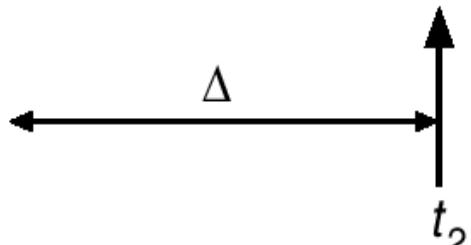
Ví dụ: $\Delta = 10$ và

chuỗi tham khảo trang

... 2 6 1 5 7 7 7 7 5 1 6 2 3 4 1 2 3 4 4 4 3 4 3 4 4 4 1 3 2 3 4 4 4 3 4 4 4 ...



$$WS(t_1) = \{1, 2, 5, 6, 7\}$$



$$WS(t_2) = \{3, 4\}$$

$$WSS(t_1) = 5$$

$$WSS(t_2) = 2$$



8.5.3 Giải pháp tập làm việc

- Đặt $D = \sum WSS_i$ = tổng các working-set size của mọi tiến trình trong hệ thống.
 - Nhận xét: Nếu $D > m$ (số frame của hệ thống) \Rightarrow sẽ xảy ra thrashing.
- Giải pháp working set:
 - Khi khởi tạo một tiến trình: cung cấp cho quá trình số lượng frame thỏa mãn working-set size của nó.
 - Nếu $D > m \Rightarrow$ tạm dừng một trong các tiến trình.
 - Các trang của tiến trình được chuyển ra đĩa cứng và các frame của nó được thu hồi.



8.5.3 Giải pháp tập làm việc

- WS loại trừ được tình trạng trì trệ mà vẫn đảm bảo mức độ đa chương.
- Theo vết các WS? => WS xấp xỉ (đọc thêm trong sách)
- Đọc thêm:
 - Hệ thống tập tin
 - Hệ thống nhập xuất
 - Hệ thống phân tán



Tóm tắt lại nội dung buổi học

- Tổng quan về bộ nhớ ảo
- Cài đặt bộ nhớ ảo: Demand Paging
- Các giải thuật thay trang (Page Replacement Algorithms)
- Vấn đề cấp phát Frames
- Vấn đề Thrashing



Bài tập

Xét chuỗi truy xuất bộ nhớ sau:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1

Có bao nhiêu lỗi trang xảy ra khi sử dụng các thuật toán thay thế sau đây, giả sử hệ thống có 4 khung trang.

- a. LRU
- b. FIFO
- c. Chiến lược tối ưu (OPT)



THẢO LUẬN

