Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang

Nguyên lý hệ điều hành

Nguyễn Hải Châu
Khoa Công nghệ thông tin
Trường Đại học Công nghệ

1

Bộ nhớ ảo (Virtual Memory)

Yêu cầu phân trang
Tạo tiến trình
Thay thế trang
Cấp phát frame
Thrashing

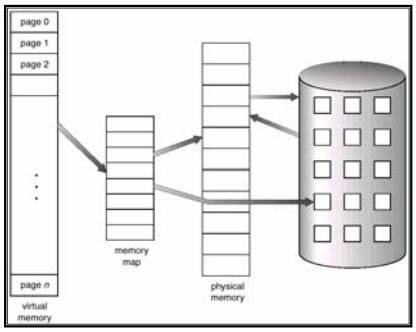
2

Virtual memory (Bộ nhớ ảo)

- **Bộ nhớ ảo** – tách biệt bộ nhớ logic và vật lý.
  - Cho phép tiến trình có cỡ lớn hơn bộ nhớ trong có thể thực hiện được
  - Không gian địa chỉ ảo có thể lớn hơn nhiều so với không gian địa chỉ vật lý (về dung lượng)
  - Cho phép các tiến trình sử dụng chung không gian địa chỉ
  - Cho phép tạo tiến trình hiệu quả hơn
- Bộ nhớ ảo có thể được cài đặt thông qua:
  - Yêu cầu phân trang (demand paging)
  - Yêu cầu phân đoạn (demand segmentation)

3

Minh họa bộ nhớ ảo có dung lượng lớn hơn bộ nhớ vật lý



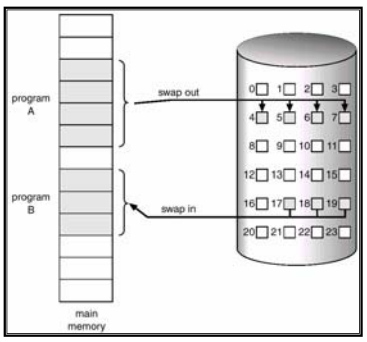
4

Yêu cầu trang (demand paging)

- Chỉ đưa một trang vào bộ nhớ khi cần thiết
  - Giảm thao tác các vào ra
  - Tiết kiệm bộ nhớ
  - Đáp ứng nhanh
  - Tăng được số người sử dụng (tiến trình)
- Khi cần một trang ⇒ tham chiếu đến nó
  - Tham chiếu lỗi ⇒ Hủy bỏ
  - Không nằm trong bộ nhớ ⇒ Đưa trang vào bộ nhớ

5

Chuyển một trang (trong bộ nhớ) ra vùng đĩa liên tục

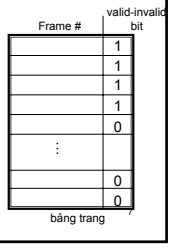


6

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang

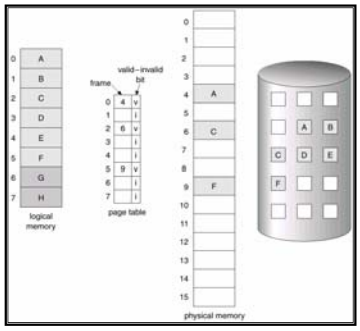
Valid-Invalid Bit

- Mỗi phần tử bảng trang có một bit hợp lệ/không hợp lệ (1: trong bộ nhớ, 0: không trong bộ nhớ)
- Khởi đầu: valid–invalid bằng 0.
- Ví dụ bảng trang+bit invalid/valid:
- Khi tính địa chỉ, nếu valid–invalid ở bảng trang là 0: ⇒ lỗi trang (page-fault trap)



7

Bảng trang với một số trang không nằm trong bộ nhớ



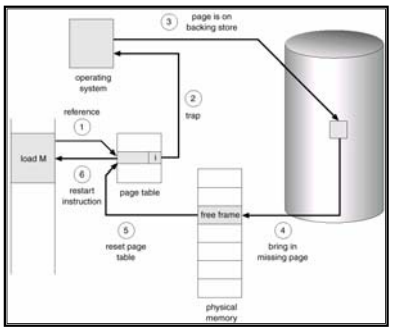
8

Xử lý page-fault (lỗi trang)

- HĐH sẽ kiểm tra nguyên nhân lỗi:
  - Lỗi từ tiến trình: kết thúc tiến trình, hoặc
  - Trang không nằm trong bộ nhớ: Thực hiện tiếp:
- Tìm một frame rỗi và đưa trang vào bộ nhớ
- Sửa lại bảng trang (bit = valid)
- Thực hiện lại lệnh tham chiếu trang
- Vấn đề hiệu năng: Nếu tại một thời điểm có yêu cầu nhiều trang (ví dụ: Một trang cho lệnh và vài trang cho dữ liệu)?

9

Các bước xử lý page-fault



10

Nếu không có frame rỗi?

- Thực hiện thay thế trang – swap out một số trang đang ở trong bộ nhớ nhưng hiện tại không được sử dụng
  - Thuật toán nào tốt?
  - Hiệu năng: Cần một thuật toán có ít page-fault nhất để hạn chế vào/ra
- Nhiều trang có thể được đưa vào bộ nhớ tại cùng một thời điểm.
- Thuật toán thay thế trang: FIFO, Optimal, LRU, LRU-approximation

11

Hiệu năng của yêu cầu trang

- Tỷ lệ page-fault là p: $0 \le p \le 1.0$
  - nếu $p = 0$: Không có page-fault
  - nếu $p = 1$, mọi yêu cầu truy cập đến trang đều gây ra page-fault
- Effective Access Time (EAT)

  EAT = (1 – p) x memory access
  + p (page fault overhead
  + [swap page out]
  + swap page in
  + restart overhead)

12

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang

Tạo tiến trình

- Bộ nhớ ảo có ưu điểm khi khởi tạo một tiến trình mới:

  - Copy-on-Write (Chỉ tạo copy của trang khi có thay đổi)

  - Memory-Mapped Files (Các file ánh xạ bộ nhớ)

13

Copy-on-Write

- Copy-on-Write (COW) cho phép tiến trình cha và con dùng chung trang trong bộ nhớ khi mới khởi tạo tiến trình con
- Chỉ khi nào một trong hai tiến trình sửa đổi trang dùng chung, thì trang đó mới được copy một bản mới
- COW làm cho việc tạo tiến trình hiệu quả hơn: Chỉ các trang bị sửa đổi mới được copy
- Các trang rỗi được cấp phát từ một tập hợp (pool) các trang được xóa trắng với số 0

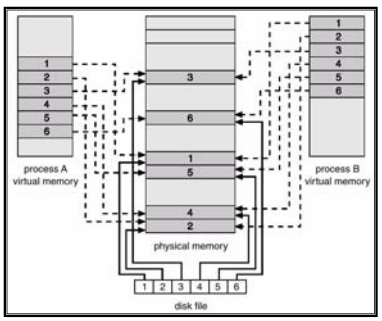
14

Các file ánh xạ bộ nhớ

- Các file được xem như một phần bộ nhớ trong bằng các ánh xạ một khối đĩa vào một trang trong bộ nhớ
- Khởi đầu các file được đọc khi có yêu cầu trang: Một phần của file (cỡ=cỡ trang) được đọc vào bộ nhớ
- Các thao tác đọc/ghi trên file sau đó được xem như đọc/ghi trong bộ nhớ
- Đơn giản hóa việc truy cập file thông qua bộ nhớ hơn là sử dụng các hàm hệ thống **read()** và **write()**.
- Cho phép các tiến trình có thể ánh xạ chung một file, do đó cho phép các trang dùng chung trong bộ nhớ

15

Ví dụ file ánh xạ bộ nhớ



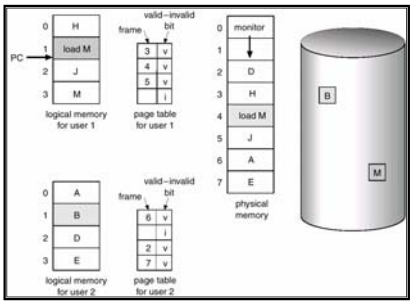
16

Thay thế trang

- Thay thế trang dùng để tránh thực hiện nhiều lần cấp phát mỗi khi có page-fault
- Sử dụng *modify bit* để giảm chi phí (overhead) vào/ra với các trang: Chỉ các trang có thay đổi mới được ghi ra đĩa
- Thay thế trang là một trong các yếu tố xóa đi sự khác biệt của bộ nhớ ảo và thật: Tiến trình lớn hơn dung lượng bộ nhớ trong có thể thực hiện được

17

Ví dụ: Yêu cầu thay thế trang



18

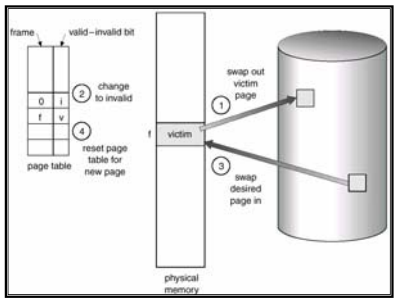
Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang

Cơ sở thay thế trang

1. Tìm vị trí của trang *p* cần thay trên đĩa
2. Tìm một frame rỗi *f*:
   1. Nếu có frame rỗi: Sử dụng frame đó
   2. Nếu không có frame rỗi: Sử dụng thuật toán thay thế trang để đưa một trang trong bộ nhớ ra để sử dụng frame ứng với trang đó
3. Đọc trang *p* vào frame *f* vừa tìm được và cập nhật bảng trang, bảng frame
4. Lặp lại quá trình này

19

Thay thế trang



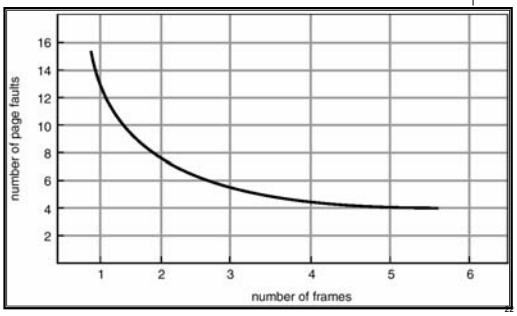
20

Thuật toán thay thế trang

- Cần tỷ lệ page-fault thấp nhất
- Đánh giá thuật toán: Thực hiện trên một danh sách các yêu cầu truy cập bộ nhớ và tính số lượng các page-fault
- Trong tất cả các ví dụ, ta sử dụng danh sách yêu cầu truy cập bộ nhớ:
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

21

Đồ thị page-fault



22

Thuật toán FIFO

- Yêu cầu truy cập: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- Bộ nhớ VL có 3 frame: 9 page-faults
- Bộ nhớ VL có 4 frame: 10 page-faults

9 page faults

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

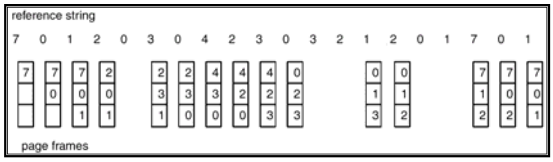
- Thay thế FIFO – Belady's anomaly
  - Có nhiều frame ⇒ ít page-fault

10 page faults

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

23

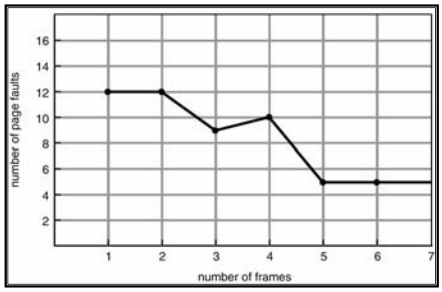
Thay thế trang FIFO



24

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang

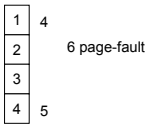
FIFO Illustrating Belady's Anamoly



25

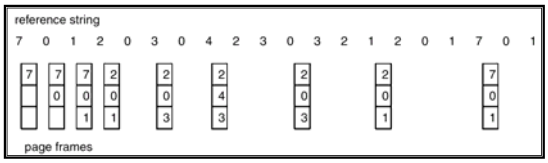
Thuật toán tối ưu

- Thay thế các trang *sẽ không được sử dụng* trong khoảng thời gian dài nhất
- Danh sách yêu cầu truy cập: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5; có 4 frame



26

Thay thế trang tối ưu



27

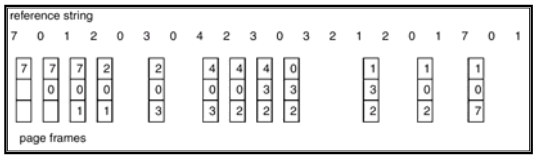
Thuật toán LRU (Least Recently Used)

- Thay thế trang *không được sử dụng lâu nhất*
- Danh sách yêu cầu truy cập: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

- Cài đặt sử dụng biến đếm
  - Mỗi trang có một biến đếm; mỗi khi trang được truy cập, gán giá trị đồng hồ thời gian cho biến đếm.
  - Khi một cần phải thay thế trang, căn cứ vào giá trị các biến đếm của trang: Cần tìm kiếm trong danh sách các trang

28

Thay thế trang LRU



29

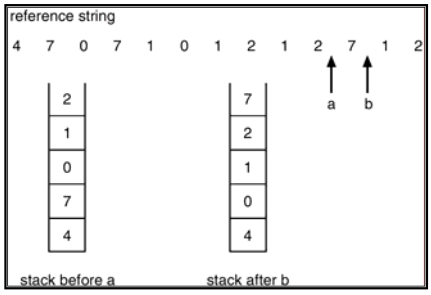
Thuật toán LRU (tiếp)

- Cài đặt sử dụng ngăn xếp: Sử dụng một ngăn xếp (stack) lưu các số hiệu trang ở dạng danh sách móc nối kép:
  - Khi trang được tham chiếu đến:
    - Chuyển trang lên đỉnh ngăn xếp
    - Cần phải thay đổi 6 con trỏ
  - Khi thay thế trang không cần tìm kiếm

30

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang

Sử dụng ngăn xếp để ghi lại trang vừa mới được sử dụng



31

Thuật toán LRU xấp xỉ

- Thuật toán bit tham chiếu
- Sử dụng bit tham chiếu đánh dấu các trang đã được sử dụng/chưa được sử dụng
  - Mỗi trang có 1 bit được khởi tạo bằng 0
  - Khi trang được tham chiếu đến, đặt bit bằng 1
  - Không biết thứ tự sử dụng các trang

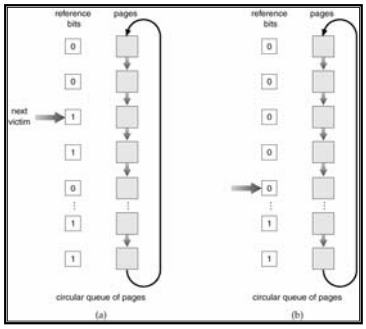
32

Thuật toán LRU xấp xỉ (tiếp)

- Thuật toán "Second-chance"
  - Là một thuật toán kiểu FIFO
  - Cần sử dụng bit tham chiếu
  - Thay thế theo thứ tự thời gian
  - Nếu trang cần được thay thế (theo thứ tự thời gian) có bit tham chiếu là 1 thì:
    - Đặt bit tham chiếu bằng 0, để trang đó trong bộ nhớ, chưa thay thế ngay (*second chance*)
    - Thay thế trang tiếp theo (theo thứ tự thời gian) tuân theo qui tắc tương tự
    - Có thể cài đặt bằng buffer vòng

33

Thuật toán second-chance



34

Các thuật toán đếm

- Sử dụng biến đếm để đếm số lần tham chiếu đến trang
- Thuật toán LFU (Least Frequently Used): Thay thế các trang có giá trị biến đếm nhỏ nhất
- Thuật toán MFU (Most Frequently Used): Ngược lại với LFU, dựa trên cơ sở: Các trang ít được sử dụng nhất (giá trị biến đếm nhỏ nhất) là các trang vừa được đưa vào bộ nhớ trong

35

Cấp phát các frame

- Mỗi tiến trình cần một số lượng tối thiểu các trang để thực hiện được
- Ví dụ: IBM 370 cần 6 để thực hiện lệnh SS MOVE:
  - Lệnh dài 6 bytes, có thể chiếm 2 trang.
  - 2 trang để thao tác **from**.
  - 2 trang để thao tác **to**.
- Hai cách cấp phát:
  - Cấp phát cố định
  - Cấp phát ưu tiên

36

Tài liệu chỉ xem được một số trang đầu. Vui lòng download file gốc để xem toàn bộ các trang

Cấp phát cố định

- Cấp phát bình đẳng: nếu cấp phát 100 frame cho 5 tiến trình, mỗi tiến trình có 20 frame.
- Cấp phát tỷ lệ: Dựa theo cỡ tiến trình.
- Cấp phát tỷ lệ:

s_i = size of process p_i
$S = \sum s_i$
m = total number of frames
a_i = allocation for $p_i = \frac{s_i}{S} \times m$

$m = 64$
$s_i = 10$
$s_2 = 127$
$a_1 = \frac{10}{137} \times 64 \approx 5$
$a_2 = \frac{127}{137} \times 64 \approx 59$

37

Cấp phát ưu tiên

- Sử dụng cấp phát tỷ lệ căn cứ vào độ ưu tiên của tiến trình, không căn cứ vào cỡ tiến trình
- Nếu tiến trình P_i sinh ra page-fault:
  - Thay thế một trong các frame của tiến trình đó
  - Thay thế một trong các frame của tiến trình khác có độ ưu tiên thấp hơn

38

Cấp phát tổng thể và cục bộ

- Thay thế tổng thể – Các trang/frame có thể được chọn để thay thế từ tập hợp tất cả các frame/trang. Tiến trình này có thể dùng lại frame/trang của tiến trình khác
- Thay thế cục bộ – Mỗi tiến trình chỉ thay thế trong các trang/frame của chính nó đã được cấp phát

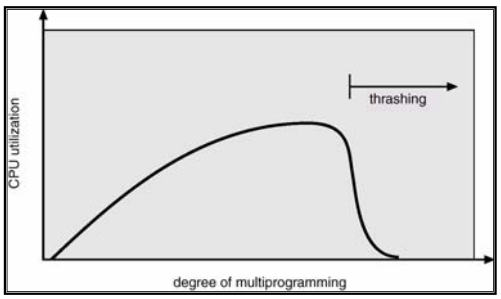
39

Thrashing

- Nếu tiến trình không có đủ trang, tỷ lệ page-fault rất cao, điều đó dẫn tới:
  - Khả năng tận dụng CPU thấp, do đó
  - HĐH có thể tăng mức độ đa chương trình →
  - Các tiến trình được tiếp tục đưa vào hệ thống
- → Hiện tượng thrashing
- Tiến trình thrashing: Một tiến trình luôn bận để swap-in và swap-out

40

Minh họa thrashing



41

Cách hạn chế/ngăn chặn thrashing

- Sử dụng thuật toán thay thế trang cục bộ hoặc thay thế trang ưu tiên để hạn chế thrashing: Chưa giải quyết tốt
- Sử dụng mô hình cục bộ: mô hình working-set
  - Khi tiến trình thực hiện, nó chuyển từ điều kiện cục bộ này sang điều kiện cục bộ khác
  - Điều kiện cục bộ được xác định dựa trên cấu trúc của chương trình và cấu trúc dữ liệu

42