**CHƯƠNG 10: BỘ NHỚ ẢO**

1. **Bộ nhớ ảo (Virtual memory):**

* Bộ nhớ ảo là một kỹ thuật cho phép một tiến trình đang được xử lý không cần nạp toàn bộ vào bộ nhớ vật lý. Bộ nhớ ảo mô hình hóa bộ nhớ vật lý như một bảng lưu trữ rất lớn và đồng nhất, tách biệt hẳn khái niệm không gian địa chỉ và không gian vật lý.
* Người sử dụng chỉ nhìn thấy và làm việc trong không gian địa chỉ ảo, việc chuyển đổi sang không gian vật lý do đơn vị quản lí bộ nhớ (memory-management unit) của hệ điều hành với sự trợ giúp của các cơ chế phần cứng cụ thể.
* Lợi ích của việc sử dụng bộ nhớ ảo gồm:
* Dung lượng của chương trình có thể lớn hơn dung lượng của bộ nhớ vật lý
* Một chương trình không cần phải nạp hoàn toàn vào bộ nhớ
* Các tiến trình có thể chia sẻ bộ nhớ
* Các tiến trình có thể được tạo ra một cách hiệu quả hơn

1. **Phân trang theo yêu cầu (Demand paging):**

* Phân trang theo yêu cầu là hệ thống sử dụng kĩ thuật phân trang kết hợp với kĩ thuật swapping. Một tiến trình được xem là tập hợp các trang, thường được lưu trên bộ nhớ trên bộ nhớ thứ cấp. Khi cần xử lý, tiến trình sẽ được nạp vào bộ nhớ chính. Nhưng thay vì nạp toàn bộ chương trình, chỉ những trang cần thiết trong thời điểm hiện tại mới được nạp vào bộ nhớ. Như vậy một trang chỉ được nạp vào bộ nhớ chính khi có yêu cầu.
* Với mô hình này, cần cung cấp một cơ chế phần cứng giúp phân biệt các trang đang ở trong bộ nhớ chính và các trang trong bộ nhớ thứ cấp hoặc sử dụng lại valid-invalid bit với ngữ nghĩa mới
* Cơ chế phần cứng:
* Là sự kết hợp của cơ chế hỗ trợ kỹ thuật phân trang và kỹ thuật swapping.
* Bảng trang: Cấu trúc bảng trang cho phép phản ánh tình trạng của mộ trang là đang nằm trang bộ nhớ chính hay bộ nhớ thứ cấp
* Bộ nhớ thứ cấp: Bộ nhớ phụ lưu trữ những trang không được nạp vào bộ nhớ chính. Vùng không gian thứ cấp được dùng để lưu trữ tạm các trang trong kỹ thuật swapping được gọi là không gian swapping.
* Lỗi trang xảy ra khi một trang được truy cập không khả dụng trong bộ nhớ. Trang nhớ phải được mang từ bộ nhớ thứ cấp vào các khung khả dụng trong bộ nhớ. Nếu nhận thấy trang đang được yêu cầu truy xuất là bất hợp lệ, cơ chế phần cứng sẽ phát sinh một ngắt đồng hồ để báo cho hệ điều hành.

1. **Nguyên tắc sao chép khi ghi (copy-on-write).**

* Nguyên tắc sao chép khi ghi (copy-on-write) là một kỹ thuật quản lý tài nguyên được sử dụng trong lập trình máy tính để thực hiện hiệu quả thao tác nhân đôi và sao chép trên các tài nguyên có thể sửa đổi. Nếu một nguồn tài nguồn được sao chép nhưng không được sửa đổi, tài nguyên có thể được chia sẻ giữa bản sao và bản gốc. Bằng cách chia sẻ tài nguyên theo cách này, có thể giảm đáng kể mức tiêu thụ tài nguyên của các bản sao chưa sửa đổi, đồng thời thêm một chi phí nhỏ cho các hoạt động sửa đổi tài nguyên.

1. **Thay trang (page replacement):**

* Khi bộ nhớ khả dụng sắp hết, giải thuật thay thế trang (page-replacement algorithm) chọn một trang hiện có trong bộ nhớ để thay thế bằng một trang mới.
* Các thuật toán thay thế trang bao gồm FIFO (First-in-first-out), optimal và LRU (Least-remaining used), second-change, counting, …
* Giải thuật thay thế trang toàn cục chọn một trang từ bất kì một tiến trình nào trong hệ thống để thay thế, trong khi đó giải thuật thay thế trang cục bộ chọn một trang từ các tiến trình xảy ra lỗi.

1. Thuật toán FIFO (First-in-first-out Algorithm):

* Hệ điều hành sẽ 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
* Thực hiện: tổ chức quản lý các trang trong bộ nhớ theo một danh sách FIFO, khi đó trang đầu danh sách sẽ được chọn để thay thế.
* Thuật toán thay thế trang FIFO dễ sử dụng, dễ cài đặt. Tuy nhiên khi hiện thực thuật toán không phải lúc nào cũng cho kết quả tốt. Nếu trang được chọn để thay thế là trang chứa nhiều dữ liệu cần thiết, thường xuyên được sử dụng nên được nạp sớm (first-in), do vậy khi chuyển ra bộ nhớ phụ (first-out) sẽ nhanh chóng gây ra lỗi trang.
* Nghịch lý Belady: Số lượng lỗi trang xảy ra sẽ tăng lên khi số lượng khung trang sử dụng tăng.

1. Thuật toán tối ưu: (Optimal Algorithm)

* Hệ điều hành sẽ thay thế trang sẽ lâu được sử dụng nhất trong tương lai
* Thuật toán này đảm bảo số lượng lỗi trang phát sinh là thấp nhất, và cũng không gánh chịu nghịch lý Belady.
* Tuy nhiên, thuật toán này không khả thi trong thực tế, vì không thể biết trước chuỗi truy xuất của tiến trình.

1. Thuật toán “Lâu nhất chưa sử dụng” (Least-recently-used LRU)

* Với mỗi trang, ghi nhân thời điểm cuối cùng trang được truy cập, trang được chọn để thay thế sẽ là trang lâu nhất chưa được truy xuất
* Thuật toán 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. Có thể cài đặt theo một trong hai cách: sử dụng bộ đếm hoặc sử dụng stack.

1. Các thuật toán xấp xỉ LRU:

* Có ít hệ thống được cung cấp đủ các hỗ trợ phần cứng để cài đặt được thuật toán LRU, do đó nó được trang bị một bit tham khảo (reference).
* Bit reference của một trang được phần cứng đặt giá trị 1 mỗi lần trang tương ứng được truy cập, và được phần cứng gán trở về 0 sau từng chu kỳ quy định trước.
* Sau từng chu kỳ quy định trước, kiểm tra giá trị của các bit reference, có thể xác định được trang nào đã được truy xuất đến và trang nào không, sau khi đã kiểm tra xong, các bit reference được phần cứng gán trở về không
* Với bit reference, có thể biết được trang nào đã được truy xuất, nhưng không biết được thứ tự truy xuất. Thông tin không đầy đủ này dẫn đến nhiều thuật toán xấp xỉ LRU khác nhau.

1. Thuật toán với các bit reference phụ trợ

* Có thể thu thập thêm nhiều thông tin về thứ tự truy xuất hơn bằng cách lưu trữ các bit reference sau từng khoảng thời gian đều đặn.
* Với mỗi trang, sử dụng thêm 8 bit lịch sử trong bảng trang. Sau từng khoảng thời gian nhất định, một ngắt đồng hồ được phát sinh, và quyền điều khiển được chuyển cho hệ điều hành. Hệ điều hành đặt bit reference của mỗi trang vào bit cao nhất trong 8 bit phụ trợ của trang bằng cách dịch phải 1 bit, bỏ bit thấp nhất. Như vậy 8 bit thêm vào sẽ lưu trữ tình hình truy xuất đến trang trong 8 chu kì cuối cùng.
* Số lượng bit lịch sử có thể thay đổi theo phần cứng, và phải được chọn sao cho việc cập nhật là nhanh nhất có thể.

1. Thuật toán “Cơ hội thứ hai” (Second chance Algorithm)

* Sử dụng một bit reference duy nhất, sử dụng thuật toán cơ sở là FIFO, tuy nhiên khi chọn dược một trang tiêu chuẩn theo tiêu chuẩn FIFO, kiểm tra bit reference của trang đó.
* Nếu bit reference = 0, thay thế trang đã chọn. Ngược lại cho trang này một cơ hội thứ hai, chọn trang FIFO tiếp theo
* Một trang đã được cho cơ hội thứ hai sẽ không bị thay thế ngược trước khi hệ thống đã thay thế hết những trang khác. Hơn nữa, nếu trang thường xuyên được sử dụng, bit reference của nó sẽ duy trì được giá trị 1, và các trang hầu như không bao giờ bị thay thế.
* Có thể cài lại thuật toán “Cơ hội thứ hai” với một danh sách liên kết vòng

1. Thuật toán “Cơ hội thứ hai” nâng cao (Not Recently Used)

* Xem các bit reference và dirty bit như một cặp có thứ tự.
* Với hai bit này, có thể có 4 tổ hợp tạo thành 4 lớp sau:
* (0, 0): không truy xuất, không sửa đổi -> đây là trang tốt nhất để thay thế
* (0. 1): không truy xuất gần đây, nhưng bị sửa đổi -> trường hợp này không thật sự tốt, vì trang cần được lưu trữ lại trước khi thay thế.
* (1 ,0): được truy xuất gần đây, nhưng không bị sửa đổi -> trang có thể nhanh chóng tiếp tục được sử dụng
* (1, 1) được truy xuất gần đây, bị sửa đổi -> trang có thể nhanh chóng tiếp tục được sử dụng, và trước khi thay thế cần được lưu trữ trở lại
* Lớp 1 có độ ưu tiên thấp nhất và tăng dần đến lớp 4
* Trang được chọn để thay thế là trang đầu tiên tìm thấy trong lớp có độ ưu tiên thấp nhất và khác rỗng.

1. **Thrashing**

* Thrashing xảy ra khi hệ thống tốn nhiều thời gian phân trang hơn thực thi, điều này xảy ra khi tiến trình không có đủ khung, dẫn đến tỉ lệ lỗi khung tăng và làm giảm CPU utilization.
* Locality: đại diện cho một tập hợp các trang được sử dụng tích cực cùng nhau. Khi một tiến trình được thực thi, nó di chuyển từ locality này đến locality khác. Một tập hợp làm việc dựa trên locality được định nghĩa là tập hợp các trang hiện tại sử dụng bởi một quá trình
* Thrashing xảy ra khi

Tổng dung lượng của locality > Tổng dung lượng bộ nhớ

* Ta có thể giới hạn việc này bằng cách sử dụng thay trang cục bộ hoặc thay trang ưu tiên.

1. **Mô hình tập hợp làm việc (Working-set model)**

* Mô hình tập hợp làm việc dựa trên giả thuyết về locality. Mô hình này sử dụng tham số ∆, để định nghĩa cửa sổ tập hợp làm việc. Tập hợp các trang được tham chiếu gần đây trong tập hợp làm việc. Nếu nó không còn được sử dụng, nó sẽ được rút ra khỏi tập hợp làm việc. Do đó, tập hợp làm việc là một xấp xỉ của locality.
* Mối tương quan giữa cửa sổ tập hợp làm việc và locality:
* Nếu ∆ quá nhỏ sẽ không bao quanh toàn bộ locality
* Nếu ∆ quá lớn sẽ bao quanh một vài locality
* Nếu ∆ → sẽ bao quanh toàn bộ chương trình

1. **Tần số lỗi trang (Page-fault Frequency)**

* Thiết lập tần số lỗi trang và sử dụng chính sách thay thế cục bộ hợp lí
* Nếu tỉ lệ quá thấp, tiến trình giảm khung
* Nếu tỉ lệ quá cao, tiến trình tăng khung

1. **Nén bộ nhớ (Memory compression)**

* Nén bộ nhớ là kĩ thuật quản lí bộ nhớ mà nén nhiều trang vào một trang. Nén bộ nhớ là một cách thay thế phân trang và được sử dụng trong cách hệ điều hành trên điện thoại mà không hỗ trợ phân trang.
* Mặc dù nén bộ nhớ không yêu cầu phân bổ khung miễn phí để giữ các trang nén, có thể thực hiện tiết kiệm bộ nhớ đáng kế, tùy thuộc vào mức giảm đạt được của thuật toán nén bộ nhớ.

1. **Bộ nhớ của nhân.**

* Bộ nhớ của nhân được cấp phát khác với các tiến trình trong chế độ người dùng. Nó được phân bố trong các khối liền kề có kích thước khác nhau.
* Hai kĩ thuật chung để cấp phát bộ nhớ cho nhân là:
* Hệ thống bạn bè
* Hệ thống phân bố bản mỏng.

1. **Những vấn đề liên quan:**

* Tiền phân trang:
* Để giảm số lượng lỗi trang sinh ra khi quá trình được khởi động
* Tiền phân trang tất cả hoặc một vài trang của tiến trình sẽ cần thiết, trước khi nó được tham chiếu
* Nhưng nếu tiền phân trang không được sử dụng, chi phí về nhập/xuất và bộ nhớ sẽ bị lãng phí
* Kích thước trang
* Việc lựa chọn kích thước trang phải xem xét đến các yếu tố như: phân mảnh, kích thước bảng phân trang, độ phân giải, chi phí nhập/xuất, số lượng lỗi trang, locality, kích thước và hiệu suất của TLB
* TLB reach: dung lượng của bộ nhớ được truy cập từ TLB và bằng với số lượng đầu vào TLB bởi kích thước trang
* Một cách lý tưởng, tập hợp làm việc của mỗi tiến trình được lưu trong TLB, nếu không sẽ gây ra nhiều khả năng xảy ra lỗi trang.
* Một kĩ thuật để tăng TLB reach là tăng kích thước của trang.
* Khóa liên động nhập/xuất:
* Khóa liên động nhập/xuất: Trong một vài trường hợp trang phải bị khóa trong bộ nhớ.
* Cân nhắc nhập/xuất: Trang được sử dụng để sap chép tệp tin từ thiết bị phải được khóa khỏi việc được chọn làm mục tiêu bởi giải thuật thay trang.

**Câu hỏi trắc nghiệm:**

1. **Đâu không phải là lợi ích của việc sử dụng bộ nhớ ảo (virtual memory):**
2. Dung lượng của chương trình lớn hơn dung lượng của bộ nhớ vật lý.
3. Chương trình không cần nạp hoàn toàn vào bộ nhớ.
4. Nhiều chương trình có thể được thực thi tại cùng một thời điểm
5. Làm giảm hiện tượng phân mảng nội của bộ nhớ

* **Đáp án: D**

1. **Cơ chế phần cứng (hardware support) hỗ trợ hệ điều hành trong cơ chế phân trang theo yêu cầu (demand paging) là:**
2. Giữ cho mọi trang luôn nằm ở trên bộ nhớ chính
3. Cung cấp bảng trang để giải phóng trang nhớ khỏi bộ nhớ chính
4. Cung cấp bộ nhớ thứ cấp để lưu trữ những trang không được nạp vào bộ nhớ chính và vùng không gian dùng trong kỹ thuật swapping được gọi là không gian swapping.
5. Tất cả các câu trên đều đúng.

* **Đáp án: C.**

1. **Chọn câu đúng về lỗi trang (page-fault):**
2. Lỗi trang (page-fault) xảy ra khi tiến trình truy xuất đến một trang không khả dụng trong bộ nhớ.
3. Việc xử lí lỗi trang (page-fault handling) được thực hiện bởi hệ điều hành.
4. Việc khởi động lại câu lệnh gây ra lỗi trang (page-fault) được thực hiện bởi cơ chế phần cứng (hardware support)
5. Tất cả các câu trên đều đúng.

* **Đáp án: D**

1. **Lợi ích của việc sử dụng nguyên tắc sao chép khi ghi (copy-on-write) là:**
2. Giảm mức độ tiêu thụ tài nguyên của các bản sao.
3. Giảm phân mảnh nội
4. Chương trình được thực thi nhanh hơn
5. Giảm số lỗi trang (page-fault) có thể xảy ra.

* **Đáp án: A**

1. **Chọn câu sai về giải thuật thay trang (page-replacement algorithm):**
2. Giải thuật thay trang (page-replacement algorithm) xảy ra khi một trang được truy xuất không khả dụng trong bộ nhớ và trang trống không thỏa mãn việc cấp phát
3. Có hai giải thuật thay trang là: thay trang toàn cục (global page-replacement) và thay trang cục bộ (local page-replacement).
4. Giải thuật thay trang giải quyết vấn đề trang nào nên được thay để giảm thiểu hụt trang (page misses) và cân bằng chi phí của nó.
5. Tất cả các câu trên đều sai

* **Đáp án: D**

1. **Chọn câu sai về** **giải thuật thay trang toàn cục (global page-replacement) và thay trang cục bộ (local page-replacement):**
2. Giải thuật thay trang toàn cục (global page-replacement) chọn bất kì trang thay thế trong bộ nhớ.
3. Giải thuật thay trang cục bộ (local page-replacement) chọn trang thay thế cùng nằm trong một tiến trình.
4. Giải thuật thay trang toàn cục (global page-replacement) cho lỗi trang (page-fault) ít hơn so với giải thuật thay trang cục bộ (local page-replacement).
5. Giải thuật thay thế trang cục bộ (local page-replacement) cho nhiều ưu điểm hơn vì tính mở rộng (scalability) của nó.

* **Đáp án: C**

1. **Giải thuật thay trang nào đảm bảo số lỗi trang (page-fault) là thấp nhất và cũng không chịu ảnh hưởng bởi nghịch lý belady ?**
2. FIFO
3. Optimal
4. Second chance
5. LRU

* **Đáp án: B**

1. **Giải thuật thay trang nào chọn trang thay thế là trang lâu nhất chưa được truy xuất**
2. FIFO
3. Optimal
4. Second chance
5. LRU

* **Đáp án: D**

1. **Trong thuật toán thay thế trang Second chance nâng cao (Not Recently Used), trang được chọn là trang:**
2. Trang đầu tiên được tìm thấy trong lớp có độ ưu tiên cao nhất và khác rỗng
3. Trang đầu tiên được tìm thấy trong lớp có độ ưu tiên thấp nhất và khác rỗng
4. Trang cuối cùng được tìm thấy trong lớp có độ ưu tiên thấp nhất và khác rỗng
5. Trang cuối cùng được tìm thấy trong lớp có độ ưu tiên cao nhất và khác rỗng.

* **Đáp án: B**

1. **Reference bit và dirty bit được dùng để phân lớp tập hợp trong thuật toán phân trang “Second change” nâng cao** **(Not Recently Used). Cho các lớp có (reference bit, dirty bit) như sau:**
2. **(1 , 1) (2) (1 , 0) (3) (0 , 1) (4) (0 , 0).**

**Mức độ ưu tiên của các lớp trong thuật toán được sắp xếp theo thứ tự tăng dần là:**

1. (1) > (2) > (3) > (4)
2. (1) > (3) > (2) > (4)
3. (4) > (3) > (2) > (1)
4. (4) > (2) > (3) > (1)

* **Đáp án: C**

1. **Có thể thiết lập lại thuật toán Second change, thay nền FIFO bằng danh sách liên kết vòng ?**
2. Đúng
3. Sai

* **Đáp án: A**

1. **Sự trì trệ (thrashing) là hiện tượng ?**
2. Tiến trình sử dụng nhiều thời gian để thay thế trang hơn là để thực thi.
3. Truy xuất đến trang không khả dụng trong bộ nhớ
4. Dung lượng của locality < Dung lượng của bộ nhớ
5. Một tiến trình bị chờ vô hạn định

* **Đáp án: A.**

1. **Sự trì trệ (thrasing) có thể được giải quyết bằng cách ?**
2. Sử dụng giải thuật thay thế trang cục bộ (local page-replacement)
3. Sử dụng giải thuật thay thế trang ưu tiên (priority page-replacement)
4. Cả 2 phương án trên đều đúng
5. Cả 2 phương án trên đều sai

* **Đáp án: C**

1. **Chọn phản biểu đúng về tập hợp làm việc (working set) ?**
2. Khi chọn ∆ đủ lớn, tập hợp làm việc (working set) chính là tập hợp của tất cả các trang mà tiến trình đã truy xuất đến
3. Là một tập hợp số trang có thể thay đổi được mà tiến trình đã truy xuất đến
4. Là tập hợp các trang mà tiến trình thường xuyên truy xuất đến
5. Tất cả đều sai.

* **Đáp án: D**

1. **Chọn câu sai về tần số lỗi trang (page-fault frequency)**
2. Giải thuật tìm tần số lỗi trang (Page-fault frequency algorithm) là cách xấp xỉ mô hình làm việc (working set) để xác định số khung (frame) mà một tiến trình cần.
3. Nếu tần số lỗi trang thấp, tiến trình sẽ giải phóng khung (lose frame).
4. Nếu tần số lỗi trang cao, tiến trình sẽ cấp phát thêm khung (gain frame)
5. Mối quan hệ giữa tần số lỗi trang và số khung là quan hệ tuyến tính.

* **Đáp án: D.**

1. **Nén bộ nhớ (Memory compression) ?**
2. Giúp lưu trữ được nhiều hệ thống trên bộ nhớ.
3. Không được ứng dụng trong các hệ điều hành trên điện thoại
4. Không tiêu tốn tài nguyên CPU
5. Chi phí cho việc nén dữ liệu trên bộ nhớ lớn hơn chi phí của việc truy xuất dữ liệu.

* **Đáp án: A.**