CHƯƠNG 10: VIRTUAL MEMORY

I – Tổng quát về bộ nhớ ảo:

Bộ nhớ ảo là một kỹ thuật cho phép việc thực thi của các process không cần nằm hoàn toàn trong bộ nhớ, giúp chương trình có dung lượng lớn hơn bộ nhớ vật lý thực có. Bộ nhớ ảo cũng cho phép các process chia sẻ các tập tin, thư viện để thực thi cơ chế chia sẻ bộ nhớ.

Như ta đã tìm hiểu ở chương 9, giải thuật memory-management yêu cầu rằng các lệnh đang thực thi phải được nạp trong bộ nhớ vật lý. Người ta tiếp cận vấn đề này lần đầu tiên bằng cách đặt bộ nhớ luận lý vào bộ nhớ vật lý và sử dụng các liên kết động, nhưng nó thường tạo ra những tiền yêu cầu đặc biệt và tốn nhiều công sức cho các lập trình viên.

Ta cũng quan sát thấy yêu cầu rằng các lệnh phải được nạp trong bộ nhớ vật lý trong nhiều trường hợp thực sự không cần thiết, ví dụ như:

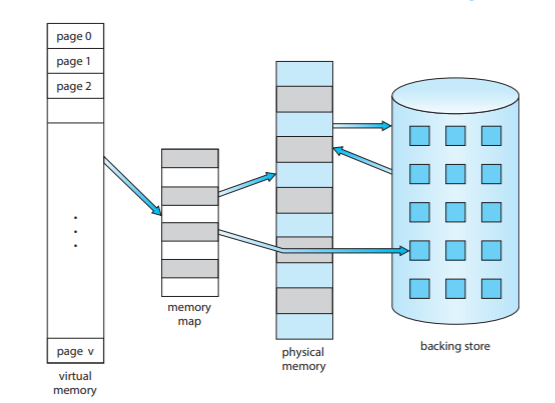
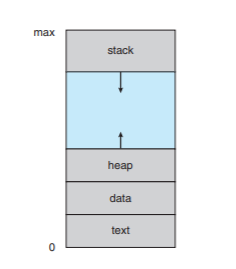
* Chương trình thường có những lệnh để xử lí được những lỗi không thường xảy ra. Do những lỗi đó hiếm khi xảy ra nên trong thực tế những dòng lệnh này hầu như không được thực thi
* Mảng, danh sách, bảng thường được cấp phát nhiều bộ nhớ hơn phần bộ nhớ chúng thật sự cần.
* Những tùy chọn và tính năng đã biết của chương trình hiếm khi được sử dụng

Khả năng thực thi từng phần chương trình trong bộ nhớ sẽ mang lại nhiều lợi ích:

* Một chương trình sẽ không còn bị hạn chế bởi dung lượng bộ nhớ vật lí có sẵn. Người dùng có thể viết một chương trình với một lượng lớn bộ nhớ luận lí để đơn giản việc lập trình.
* Bởi vì mỗi chương trình sử dụng ít bộ nhớ vật lý hơn, nhiều chương trình có thể chạy cùng lúc, giúp tăng CPU utilization và thông năng nhưng không làm tăng response time or turnaround time.
* Giảm thiểu việc sử dụng I/O để nạp và tải từng phần của chương trình vào bộ nhớ, nên mỗi chương trình sẽ chạy nhanh hơn.

II – Không gian địa chỉ ảo:

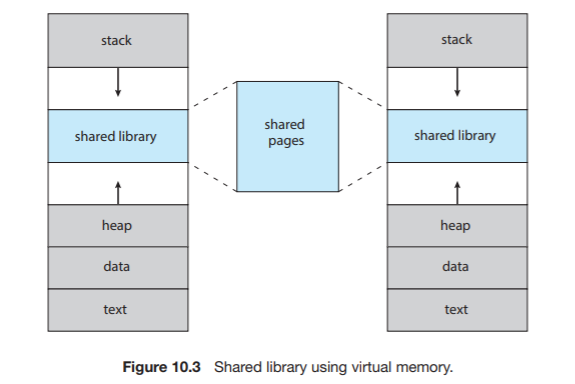
Không gian địa chỉ ảo: diễn tả cái nhìn luận lý về cách mà process được lưu trữ trong bộ nhớ. Thông thường một process bắt đầu ở địa chỉ 0, trải dài liên tục cho đến khi đến điểm kết thúc của không gian. Trong khi đó bộ nhớ vật lý được tổ chức thành các page frames và các page frames được gán cho các process có thể không trải dài liên tục. Nó phụ thuộc vào đơn vị quản lí bộ nhớ (memory-management unit – MMU) để ánh xạ địa chỉ luận lí vào địa chỉ vật lý.



Thông thường, việc thiết kế không gian địa chỉ luận lý cho vùng stack bắt đầu từ địa chỉ max và bắt đầu được cấp phát xuống trong khi vùng heap được cấp phát lên. Điều này giúp tối đa hóa việc sử dụng không gian địa chỉ. Vùng không gian trống giữa heap và stack được gọi là hole, và không có không gian vật lí cần yêu cầu cấp phát để cung cấp một trang mới cho đến khi 2 vùng stack và heap tăng lên. Vùng không gian địa chỉ ảo có chứa hole được gọi là sparse address space. Sử dụng sparse address space có rất nhiều điểm lợi vì hole có thể được lấp đầy khi vùng stack và heap dần tăng lên hoặc chúng ta muốn liên kết động thư viện hoặc những đối tượng chia sẻ chung trong thời gian thực thi chương trình.

Ngoài ra việc tách biệt bộ nhớ luận lý khỏi bộ nhớ vật lý, bộ nhớ ảo cho phép tập tin và dữ liệu có thể được chia sẻ bởi nhiều process thông qua cơ chế page sharing. Điều này dẫn tới những lợi ích như sau:

* Thư viện của hệ thống như thư viện C chuẩn có thể được chia sẻ bởi nhiều process thông qua việc ánh xạ của các đối tượng được chia sẻ vào không gian địa chỉ ảo.
* Tương tự, các process có thể chia sẻ memory
* Pages có thể được chia sẻ trong quá trình tạo ra process thông qua lệnh gọi hệ thống fork(), giúp tăng tốc độ khởi tạo process.

III – Phân trang:

Xem xét cách mà một chương trình thực thi có thể được nạp từ bộ lưu trữ thứ cấp vào bộ nhớ. Lựa chọn thứ nhất là nạp toàn bộ chương trình vào bộ nhớ vật lý tại thời điểm thực thi chương trình. Tuy nhiên, một vấn đề với cách tiếp cận này là chúng ra không cần toàn bộ chương trình vào ban đầu. Một chiến lược thay thế khách là nạp trang chứ những gì chương trình cần. Kỹ thuật này được gọi là phân trang và thường được sử dụng trong hệ thống bộ nhớ ảo.

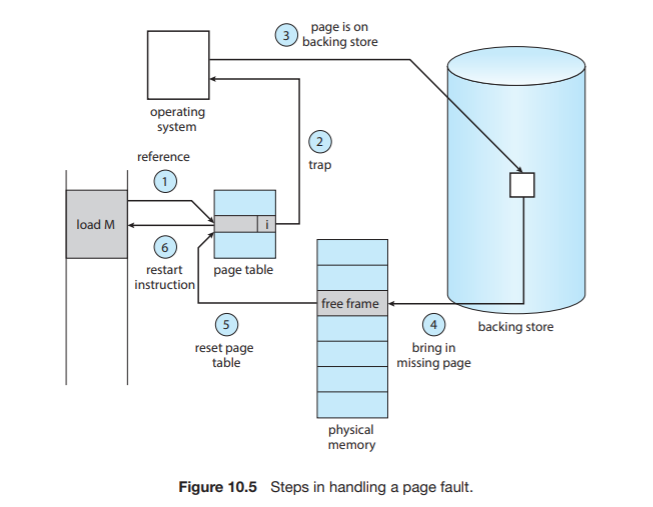
Kỹ thuật phân trang có thể mang toàn bộ các quá trình vào bộ nhớ tại thời điểm nạp hoặc mang từng trang mà chương trình cần vào bộ nhớ. Kỹ thuật này mang lại nhiều lợi ích:

* Giảm thiểu yêu cầu về I/O
* Giảm thiểu yêu cầu về bộ nhớ
* Thời gian phản hồi nhanh hơn
* Có nhiều chương trình cùng thực thi trong một thời gian.

Cơ chế chung của kỹ thuật phân trang là nạp một trang vào bộ nhớ chỉ khi chương trình cần. Khi process đang được thực thi, một vài trang đã ở sẵn trong bộ nhớ, một vài trang nằm ở bộ nhớ thứ cấp, do đó chúng ta cần sự trợ giúp của phần cứng để phân biệt 2 loại trang này. Valid-bit được sử dụng cho mục đích này. Nếu như bit được thiết lập là valid, trang được liên kết có sẵn trong bộ nhớ. Nếu như bit được thiết lập là invalid, trang được liên kết hoặc không có sẵn trong bộ nhớ luận lý của process hoặc đang nằm trong bộ nhớ thứ cấp. Bảng chỉ mục trang được memory-management unit sử dụng để ánh xạ trang từ bộ nhớ luận lý vào bộ nhớ luận lý. Nếu một process cố gắng truy cập vào một trang chưa được nạp vào bộ nhớ sẽ tạo ra lỗi page fault. Phần cứng khi dịch địa chỉ sang trang, nếu thấy invalid-bit được thiết lập sẽ gây ra trap đến hệ điều hành.

Quy trình để xử lí lỗi page fault gồm:

1. Kiểm tra bảng chỉ mục trang để xác định việc tham chiếu đến trang nhớ có hợp lệ hay không
2. Nếu việc tham chiếu không hợp lệ, ta kết thúc process. Nếu như nó hợp lệ nhưng chưa được mang vào trang nhớ đó thì ta nạp dữ liệu vào trang nhớ đó.
3. Ta tìm một free frame
4. Ta định thời bộ nhớ thứ cấp để đọc trang nhớ được cấp phát mới vào frame
5. Khi việc đọc lưu trữ hoàn thành, ta chỉnh sửa bảng chỉ mục trang để thông báo rằng trang đó đã nhập vào bộ nhớ.
6. Ta thực hiện lại lệnh đã được ngưng bởi trap. Process bây giờ có thể truy cập vào trang tham chiếu đến bởi vì nó đã được nạp vào bộ nhớ.



Trong trường hợp xấu nhất, một process bắt đầu với không có trang được nạp trong bộ nhớ. Khi hệ điều hành thiết lập con trỏ lệnh tới lệnh đầu tiên của process, nó ngay lập tức gây ra page fault. Sau khi trang này được nạp vào bộ nhớ, process tiếp tục được thực thi và việc xảy ra lỗi xảy ra ở tất cả mọi trang cho đến khi tất cả các trang cần được nạp vào bộ nhớ.

Về mặt lý thuyết, một vài chương trình có thể truy cập nhiều trang của bộ nhớ với mỗi lệnh được thực thi dẫn đến gây ra multiple page faults cho mỗi lệnh. Do đó ta cần sự trợ giúp của phần cứng cho việc phân trang

* Bảng chỉ mục trang với valid-invalid bit
* Bộ nhớ thứ cấp với swap device và swap space
* Khởi động lại câu lệnh

Một yêu cầu bắt buộc cho kĩ thuật phân trang là khả năng khởi động lại bắt kì câu lệnh sau khi xảy ra page fault. Bởi vì chúng ta lưu trạng thái của process bị ngưng khi xảy ra page fault, chúng ta phải khởi động lại process chính xác tại đúng nơi và đúng trạng thái, trừ việc trang cần tham chiếu đã được nạp trong bộ nhớ và được quyền truy cập. Trong nhiều trường hợp, yêu cầu dễ dàng được xử lí. Khi process xảy ra page fault khi nạp lệnh, ta khởi động lại bằng cách nạp lại lệnh. Khi process xảy ra page fault khi nạp toán hạng, chúng ta có thể nạp lại và giải mã lệnh rồi nạp lại vào toán hạng.

IV – Free-frame list

Khi xảy ra page fault, hệ điều hành phải nạp trang cần truy xuất từ bộ nhớ thứ cấp và bộ nhớ chính. Để giải quyết page faults, hầu hết các hệ điều hành duy trì một free-frame list – một nhóm các khung miễn phí để giải quyết các yêu cầu (free frames phải được cấp phát khi vùng stack hoặc heap từ các process được mở rộng). Hệ điều hành thường cấp phát free-frame bằng cách sử dụng kĩ thuật được gọi là zero-fill-on-demain. Khi một hệ thống sẵn sàng, tất cả các vùng nhớ khả dụng sẽ được đặt vào các free-frame list. Vì các free-frame, kích thước của các free-frame sẽ được giảm xuống. Tại một số điểm, danh sách hoặc giảm về không hoặc rơi xuống dưới một ngưỡng nhất định, tại thời điểm đó nó phải được hồi sinh.

V – Hiệu năng của kĩ thuật phân trang

Kĩ thuật phân trang làm tăng hiệu năng của hệ thống máy tính một cách đáng kể.

Công thức tính thời gian truy cập hiệu quả:

effective access time = (1 − p) × ma + p × page fault time

Trong mọi trường hợp, có 3 thành phần tác vụ chính của dịch vụ thời gian page-fault:

1. Dịch vụ ngắt trang khi có lỗi
2. Dịch vụ đọc trang
3. Khởi động lại process

Tác vụ thứ nhất và thứ 3 có thể được làm ngắn gọn bằng vài trăm câu lệnh

Thời gian truy xuất hiệu quả tỉ lệ thuận với tỷ lệ lỗi trang.

VI – Tối ưu hóa kĩ thuật phân trang:

Một khía cạnh bổ sung của kĩ thuật phân trang là xử lí và sử dụng tổng thể trao đổi không gian. Nhập/xuất để trao đổi không gian thường nhanh hơn trong hệ thống thông tin, bởi vì trao đổi không gian được cấp phát trong các khối lớn hơn nhiều và tra cứu tệp cũng như các phương pháp cấp phát gián tiếp không được sử dụng. Một lựa chọn cho hệ thống để đạt được thông lượng phân trang tốt hơn đó là sao chép toàn bộ tệp tin hình ảnh vào không gian trao đổi tại thời điểm process được khởi động và tiến hành phân trang từ không gian trao đổi. Một vài hệ thống cố gắng giới hạn dung lượng không gian trao đổi thông qua kĩ thuật phân trang các tệp tin nhị phân thực thi.

Tuy nhiên, các hệ điều hành trên điện thoại thông thường không hỗ trợ việc trao đổi. Thay vào đó, những hệ thống này phân trang từ các file hệ thống và lấy lại các trang chỉ đọc (như mã) từ các ứng dụng nếu bộ nhớ bị ràng buộc. Dữ liệu đó có thể được phân trang theo yêu cầu từ hệ thống tệp nếu nó là cần thiết sau này.

VII – Cơ chế Copy-on-Write

Copy-on-Write là một chiến lược tối ưu hóa được sử dụng trong lập trình máy tính. Ý tưởng cơ bản là nếu nhiều người gọi yêu cầu các tài nguyên đầu không thể phân biệt được, ta có thể cung cấp cho họ các trang nhớ tới cùng một tài nguyên. Chức năng này có thể được duy trì cho đến khi có sự sửa đổi bản sao tài nguyên, tại đó, một bản sao riêng thực sự được tạo ra để ngăn những thay đổi hiển thị cho mọi người khác. Tất cả điều này xảy ra trong suốt quá trình thực thi.

Hệ điều hành có thể thay vào đó sử dụng phương thức thay ra và thay vào tiêu chuẩn, giải phóng các khung và giảm mức độ đa lập trình.

VIII – Thay thế trang:

Ngăn chặn việc phân bổ bộ nhớ quá mức bằng cách sửa đổi dịch vụ lỗi trang thói quen bao gồm thay thế trang. Bằng cách sử dụng modify-bit để giảm chi phí chuyển trang – chỉ các trang sửa đổi được ghi vào đĩa. Thay thế trang hoàn toàn tách biệt giữa bộ nhớ luận lý và bộ nhớ vật lý. Một bộ nhớ ảo lớn có thể được cung cấp bởi một bộ nhớ luận lý nhỏ hơn.

Quy trình thực hiện thay thế trang:

1. Tìm vị trí của trang cần tham chiếu trên ổ đĩa
2. Tìm frame trống: nếu có frame trống, sử dụng nó. Nếu không, sử dụng giải thuật thay thế trang để chọn 1 frame.
3. Mang trang cần tham khảo vào frame mới được tìm thấy, cập nhật trang và bảng chỉ mục trang
4. Tiếp tục process bằng cách khởi động lại câu lệnh gây ra trap

IX – Giải thuật thay thế trang và thay thế khung

Giải thuật cấp phát khung xác định:

* Bao nhiêu khung được cấp phát cho mỗi process
* Khung nào được thay thế

Giải thuật thay thế trang được sử dụng để giảm tỉ lệ lỗi trang khi truy xuất lần đầu và truy xuất lại.