BÀI 3: LẬP LỊCH

Nội dung

- Tiêu chuẩn lập lịch.
- Mức ưu tiên.
- Các giải thuật lập lịch.
- Đa xử lý.
- Các mô hình kết nối – thiếu.
- Hệ điều hành trong hệ thống đa xử lý.

Mục tiêu

- Nắm được các kiến thức liên quan đến mục tiêu của việc lập lịch.
- Biết được các thuật toán lập lịch cơ bản.
- Trình bày được một thuật toán lập lịch.

Thời lượng học

- 8 tiết.

TÌNH HUỐNG DẪN NHẬP

Tình huống

Trong một máy tính, ta có thể chạy đồng thời cùng lúc nhiều ứng dụng như:

- Nghe nhạc, xem phim.
- Chương trình Word, PowerPoint.
- Xem đồng hồ.
- Xem lịch
- Vào Internet.
- …..

Câu hỏi

Như ta đã biết thì tại một thời điểm, bộ xử lý của máy tính chỉ phục vụ được một ứng dụng. Vậy tại sao cùng một lúc ta có thể chạy và xem được nhiều ứng dụng như vậy, cơ chế nào của máy tính giúp ta làm được điều đó?

3.1. Tiêu chuẩn lập lịch

Các mức lập lịch:

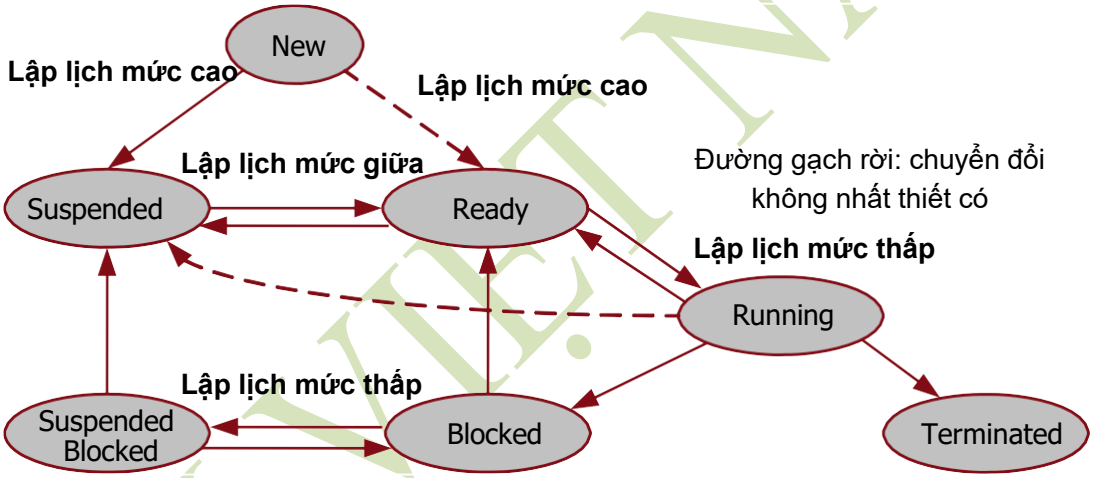
Có thể chia thành 3 mức lập lịch khác nhau:

- Lập lịch mức cao.
- Lập lịch mức giữa.
- Lập lịch mức thấp.

Lập lịch mức cao, hay lập lịch cho các task: các công cụ ở mức này xác định bài toán (chương trình) nào được đưa vào hệ thống, nghĩa là tạo ra tiến trình tương ứng với chương trình đó.

Lập lịch mức giữa: mức này xác định các tiến trình được sử dụng bộ xử lý. Bộ lập lịch ở mức này phản ứng với các thay đổi của hệ thống. Nó sẽ dừng hoặc kích hoạt các tiến trình để đảm bảo hệ thống hoạt động bình thường, đạt các thông số kỹ thuật đề ra.

Lập lịch mức thấp: công cụ ở mức này xác định Ready Process nào tiếp theo sẽ được quyền sử dụng bộ xử lý, do đó thường được gọi là Dispacher.



Các mục tiêu của việc lập lịch:

Cơ chế lập lịch cần đạt được các mục tiêu sau:

- Đúng đắn, nghĩa là cơ chế lập lịch cần phục vụ các tiến trình "công bằng", tránh tình huống có tiến trình bị rơi vào tình trạng chờ vô hạn.
- Đảm bảo khả năng thông qua lớn nhất, tức là tiến tới phục vụ số lượng tiến trình nhiều nhất có thể trong một đơn vị thời gian.
- Thời gian phản ứng chấp nhận được với tất cả các tiến trình tối thiểu chi phí, tài nguyên hệ thống.
- Cân đối việc sử dụng tài nguyên, cần cố gắng nâng cao hiệu suất sử dụng tài nguyên, theo đó cần ưu tiên tiến trình sử dụng tài nguyên giá thành thấp.
- Đảm bảo cân đối giữa thời gian trả lời và hiệu suất sử dụng tài nguyên. Cách tốt nhất để giảm thời gian trả lời là có đủ tài nguyên dự trữ để khi có yêu cầu có thể cấp phát ngay lập tức, nhưng điều đó cũng dẫn tới lãng phí tài nguyên.

- Ngăn ngừa tình huống chờ vô hạn.
- Cần quan tâm các tiến trình đang sử dụng tài nguyên quan trọng, tránh tình trạng tiến trình có mức ưu tiên thấp chiếm tài nguyên mà tiến trình mức ưu tiên cao hơn cần. Nếu tài nguyên đó là không chia sẻ thì hệ điều hành cần tạo điều kiện để tiến trình giải phóng tài nguyên nhanh nhất.

Chúng ta thấy rằng nhiều yêu cầu, mục tiêu trái ngược nhau, do đó việc lập lịch cho các tiến trình là bài toán phức tạp.

Tiêu chuẩn lập lịch

Để đạt được các mục tiêu ở trên, cơ chế lập lịch cần chú ý các yếu tố sau:

- Tiến trình có thực hiện yêu cầu thao tác I/O không?
- Tiến trình có sử dụng bộ xử lý hết lượng tử thời gian (Quantum) hay không?
- Yêu cầu về thời gian trả lời hệ thống cần đạt được.
- Mức ưu tiên của từng tiến trình.
- Tần suất ngắt Missing Page Fault.
- Thời gian tổng cộng tiến trình được sử dụng bộ xử lý.

3.2. Mức ưu tiên

Trong hệ thống, nói chung các tiến trình có vai trò quan trọng khác nhau. Mức độ quan trọng của tiến trình được thể hiện qua mức ưu tiên (Priority) của nó. Mức ưu tiên của tiến trình được gán bởi hệ điều hành, và phụ thuộc kiến trúc của hệ điều hành mà mức ưu tiên đó có thể là động hoặc tĩnh, có thể được gán theo các tiêu chuẩn xác định hoặc ngẫu nhiên (trong trường hợp hệ điều hành không phân biệt được tiến trình nào cần mức ưu tiên cao hơn).

Trong hệ thống sử dụng mức ưu tiên tĩnh, mức ưu tiên của tiến trình được gán ngay khi nó được tạo ra và không thay đổi trong suốt quá trình tồn tại của tiến trình. Sơ đồ mức ưu tiên tĩnh dễ dàng thiết kế và cài đặt hơn. Tuy nhiên chúng không có khả năng điều chỉnh để phù hợp với sự thay đổi của môi trường.

Ngày nay trong hệ điều hành đều sử dụng sơ đồ mức ưu tiên động. Theo đó mức ưu tiên của tiến trình có thể thay đổi khác với mức ưu tiên khởi tạo ban đầu. Cơ chế này cho phép hệ thống thích nghi với sự thay đổi của môi trường để đạt chỉ tiêu tốt hơn, tuy nhiên nó cũng khó khăn hơn trong xây dựng và cài đặt.

Khoảng lượng tử thời gian, ngắt thời gian

Như ta đã biết, tiến trình chỉ thực sự hoạt động khi nó sử dụng bộ xử lý. Nếu tiến trình là tiến trình hệ thống thì lúc đó hệ điều hành thực sự hoạt động. Để tránh tình trạng độc quyền chiếm giữ bộ xử lý, hệ điều hành có các cơ chế cho phép lấy lại quyền kiểm soát bộ xử lý.

Hệ điều hành thiết lập đồng hồ hệ thống, xác định khoảng thời gian gọi là lượng tử thời gian, theo đó sinh ra các tín hiệu ngắt thời gian. Khi đó bộ xử lý chuyển sang phục vụ tiến trình tiếp theo. Như thế, tiến trình có thể chiếm bộ xử lý đến khi nó tự giải phóng hoặc khi có ngắt tiếp theo. Khi bộ xử lý được giải phóng, hệ điều hành sẽ xác định tiến trình nào tiếp theo được chiếm bộ xử lý.

Ngắt thời gian giúp hệ thống đảm bảo thời gian trả lời chấp nhận được với tất cả tiến trình, tránh tình trạng chờ vô hạn, đồng thời cho phép hệ thống phản ứng với các sự kiện phụ thuộc thời gian.

3.3. Các giải thuật lập lịch

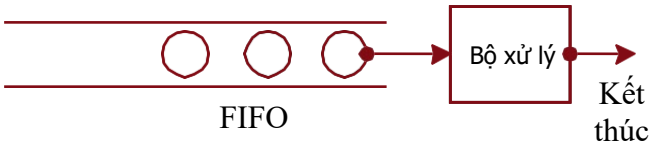
3.3.1. Lập lịch theo thời gian kết thúc

Khi áp dụng giải thuật này, hệ thống sử dụng tất cả khả năng hiện có để một ứng dụng nào đó có thể kết thúc trong thời hạn định trước. Ví dụ, trường hợp điều khiển tên lửa, các kết quả tính toán chỉ có ý nghĩa trước thời điểm nào đó. Lập lịch theo cơ chế này vấp phải các khó khăn:

- Người dùng cần chỉ rõ các tài nguyên cần thiết phục vụ cho ứng dụng và điều này không phải luôn dễ dàng thực hiện.
- Hệ thống một mặt phải thực hiện ứng dụng đúng hạn, mặt khác không được làm
  ảnh hưởng "quá nhiều" đến các ứng dụng khác.
- Rất có thể xảy ra việc tranh chấp tài nguyên giữa các ứng dụng.
- Nếu đồng thời có nhiều yêu cầu kết thúc các ứng dụng đúng thời hạn thì vấn đề lập lịch có thể rất phức tạp.
- Việc phức tạp trong lập lịch thường kéo theo chi phí tài nguyên lớn hơn và làm
  ảnh hưởng đến cả hệ thống.

3.3.2. Lập lịch theo nguyên tắc FIFO

Có lẽ đây là nguyên tắc lập lịch đơn giản nhất. Theo đó bộ xử lý phục vụ các tiến trình theo thứ tự trong danh sách các Ready Process.



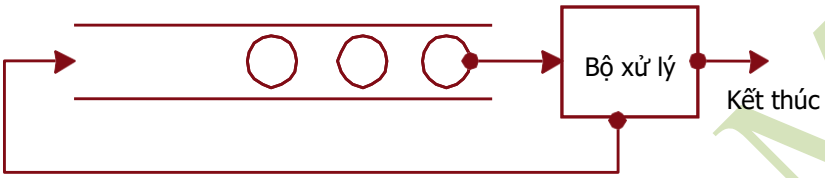
Sau khi tiến trình được quyền sử dụng bộ xử lý, nó được thực hiện đến khi kết thúc. Nguyên tắc FIFO là không hoán đổi, nghĩa là bộ xử lý không thực hiện phục vụ quay vòng lần lượt các Ready Process mà phục vụ từng tiến trình đến khi kết thúc.

Nguyên tắc FIFO có tính xác định cao, có thể dự đoán tương đối chính xác thời gian thực hiện các bài toán. Tuy nhiên vì nó là không hoán đổi nên dễ xảy ra trường hợp tiến trình quan trọng hơn phải chờ các tiến trình khác đứng trước trong danh sách kết thúc mới được thực hiện. Vì thế hiện nay nguyên tắc này không được áp dụng đơn thuần mà thường kết hợp với các phương pháp khác trong các biện pháp tổ hợp.

3.3.3. Nguyên tắc quay vòng (Round robin-RR)

Trong nguyên tắc này, việc điều hành thực hiện các tiến trình thực hiện theo nguyên tắc FIFO nhưng có hoán đổi, nghĩa là mỗi tiến trình trong mỗi lần được sử dụng bộ xử

lý không được vượt quá khoảng thời gian lượng tử (Quantum). Nếu nó không tự giải phóng bộ xử lý sau khoảng thời gian đó thì hệ điều hành sẽ lấy lại quyền điều khiển bộ xử lý và chuyển sang phục vụ tiến trình tiếp theo trong danh sách. tiến trình bị tước quyền được đưa vào cuối danh sách.



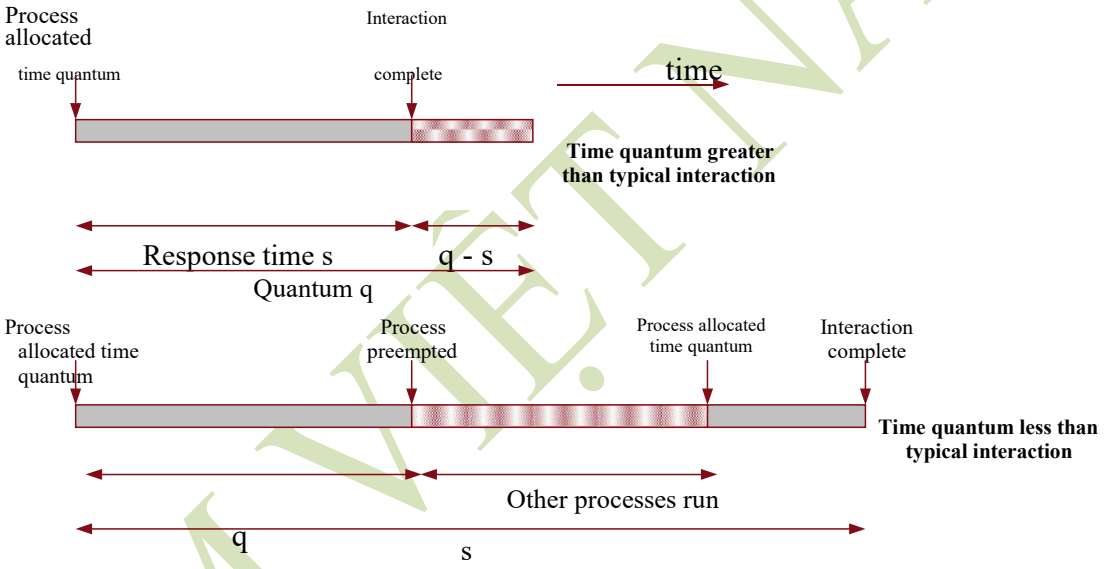
Hết thời gian lượng tử

Nguyên tắc quay vòng có hiệu quả trong các hệ thống phân chia thời gian và cần đảm bảo thời gian trả lời chấp nhận được với tất cả các tiến trình. Chi phí tài nguyên có thể giảm xuống nhờ cơ chế chuyển ngữ cảnh và dung lượng bộ nhớ đủ lớn để đồng thời nạp nhiều ứng dụng.

3.3.4. Giá trị của lượng tử

Trong những nguyên tắc như RR ở trên, việc xác định giá trị của lượng tử có ảnh hưởng đến các chỉ số hoạt động của hệ thống. Giá trị đó là lớn hay nhỏ? cố định hay thay đổi? Với các tiến trình, nó có giá trị như nhau hay khác nhau?

Nếu giá trị đó quá lớn thì có thể lớn hơn cả thời gian thực hiện ứng dụng, nghĩa là trở thành nguyên tắc FIFO. Và như thế không đảm bảo đa nhiệm tốt. Còn nếu giá trị đó quá nhỏ thì chi phí cho việc chuyển đổi ngữ cảnh chiếm phần lớn thời gian và năng lực tính toán của cả hệ thống, chỉ số hoạt động của hệ thống sẽ quá thấp. Quan hệ giữa giá trị của lượng tử và hiệu suất của hệ thống được biểu diễn qua đồ thị sau:



Có thể thấy rằng giá trị tối ưu không phải cố định, nó thay đổi theo từng hệ thống và theo tải của hệ thống, nó cũng có thể khác nhau với từng tiến trình.

3.3.5. Lập lịch theo nguyên tắc SJF (Shortes Job First)

Nguyên tắc SJF là nguyên tắc không hoán đổi, theo đó bài toán có thời gian thực hiện ngắn nhất theo dự đoán sẽ được thực hiện trước.

Nguyên tắc này ưu tiên các bài toán nhỏ, vì nói chung việc tạo điều kiện cho các bài toán nhỏ thực hiện và kết thúc dễ dàng hơn. Từ đó hàng đợi các bài toán giảm đi nhanh chóng. Tuy nhiên, nguyên tắc này không tính đến mức ưu tiên, độ quan trọng của các bài toán.

Theo nguyên tắc này bài toán nhỏ được thực hiện trước do đó hàng đợi nhanh chóng giảm đi và thời gian chờ trung bình giảm. Tuy nhiên, việc xác định chính xác thời gian thực hiện bài toán là việc khó khăn và nhiều trường hợp không thể dự đoán chính xác được.

3.3.6. Lập lịch theo nguyên tắc SRT

Nguyên tắc SRT cũng tương tự nguyên tắc SJF, nhưng SRT là có hoán đổi. Theo nguyên tắc này, tiến trình được đánh giá là có khoảng thời gian còn lại đến khi kết thúc là ngắn nhất hoặc tiến trình mới được tạo sẽ được ưu tiên.

Tiến trình được đưa vào thực hiện sẽ được chạy đến khi nó kết thúc, hoặc khi có một tiến trình mới được đưa vào hệ thống mà có thời gian hoạt động nhỏ hơn thời gian còn lại của tiến trình đang được thực hiện.

Để thực hiện nguyên tắc này, chúng ta lại gặp phải vấn đề dự đoán chính xác thời gian còn lại của tiến trình. Nguyên tắc này đòi hỏi chi phí tính toán lớn hơn so với nguyên tắc SJF. Cơ chế SRT đòi hỏi luôn phải theo dõi thời gian thực hiện của các bài toán để có thể xử lý các tình huống ngắt. Các tiến trình ngắn nói chung được thực thi ngay. Trong khi đó, những tiến trình thực thi lâu sẽ phải chờ lâu hơn thời gian trung bình so với cơ chế SJF.

Theo nguyên tắc SRT, hệ thống cần ghi lại thời gian thực hiện của các tiến trình và điều này làm tăng chi phí tính toán. Về lý thuyết, nguyên tắc SRT đảm bảo thời gian chờ cực tiểu, tuy nhiên do thao tác chuyển đổi ngữ cảnh mà điều đó không phải luôn đúng.

Giả sử tiến trình đang được thực hiện đến lúc gần kết thúc thì xuất hiện tiến trình mới có thời gian thực hiện ngắn hơn. Câu hỏi đặt ra là có ngắt tiến trình đang thực hiện hay không? vì có thể thời gian thực hiện thao tác chuyển đổi ngữ cảnh còn lớn hơn bản thân thời gian thực hiện tiến trình. Để khắc phục nhược điểm này, trong một số hệ thống người ta đặt ra ngưỡng thời gian, theo đó thời gian thực hiện tiến trình hiện tại nhỏ hơn ngưỡng đó thì sẽ không thực hiện thao tác chuyển đổi ngữ cảnh.

Một trường hợp khác cần để ý, đó là khi bài toán mới xuất hiện có thời gian thực hiện dự đoán xấp xỉ thời gian còn lại của bài toán hiện tại, khi đó nếu thực hiện chuyển đổi ngữ cảnh thì chi phí lớn hơn lợi ích thu được.

Chúng ta phân tích các trường hợp trên với mục đích cho thấy khi thiết kế hệ thống cần phải xem xét kỹ hiệu quả thu được với chi phí bỏ ra.

3.3.7. Lập lịch theo nguyên tắc HRN

Brinch Hansen đã đề xuất nguyên tắc HRN (Highest response Ratio Next) để khắc phục một số nhược điểm trong nguyên tắc SJF, đặc biệt sự quá ưu tiên các bài toán có thời gian thực hiện ngắn. HRN là nguyên tắc không hoán đổi và mức ưu tiên động, theo đó mức ưu tiên của tiến trình phụ thuộc không chỉ thời gian cần thực hiện nó mà còn cả thời gian nó phải chờ được phục vụ. Công thức tính toán như sau:

$$p = (t_w + t_s)/t_s$$

trong đó:

- t_w thời gian chờ,
- t_s thời gian thực thi.

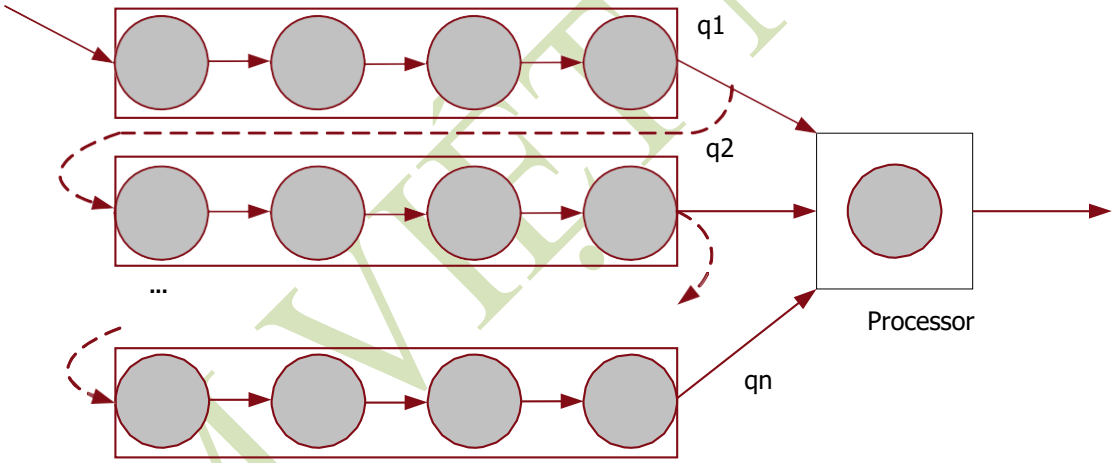
Dễ thấy mức ưu tiên càng cao khi thời gian thực hiện càng ngắn hoặc khi thời gian chờ càng lớn.

3.3.8. Hàng đợi nhiều mức với mối liên hệ ngược (Call Back)

Khi tiến trình chiếm bộ xử lý, nói chung ta khó có thể dự đoán trước về hành vi của nó, thời gian nó cần bộ xử lý. Tiến trình có thể chỉ cần bộ xử lý trong khoảng thời gian ngắn rồi thực hiện yêu cầu vào/ra, hoặc nó cũng có thể cần bộ xử lý tính toán trong khoảng thời gian dài nếu hệ điều hành không lấy lại quyền điều khiển. Do đó cơ chế lập lịch cần:

- Chú ý đến các bài toán ngắn.
- Chú ý các bài toán thường thực hiện thao tác vào/ra để sử dụng thiết bị vào/ra hiệu quả.
- Nhanh chóng xác định đặc điểm của bài toán để có thể lập lịch tối ưu.

Cơ chế hàng đợi nhiều mức với mối liên hệ ngược (hình 3.3.8) là cơ chế cho phép đạt được các yêu cầu trên. Khi tiến trình mới được khởi tạo, nó được đưa vào cuối hàng đợi mức trên. Nó dần dịch chuyển lên phía đầu hàng đợi do các tiến trình khác lần lượt được sử dụng bộ xử lý. Khi tiến trình đang chiếm bộ xử lý kết thúc, hoặc thực hiện yêu cầu vào/ra, hoặc hết thời gian lượng tử hoặc có ngắt,… bộ xử lý được giải phóng và đến tiến trình tiếp theo trong hàng đợi được sử dụng bộ xử lý.



Hình 3.3.8

Nếu tiến trình chưa kết thúc nhưng hết thời gian lượng tử, nó bị tước quyền sử dụng bộ xử lý và được đưa vào cuối hàng đợi mức thấp hơn. Nếu nó thực hiện yêu cầu vào/ra thì nó sẽ được chuyển xuống cuối hàng đợi cùng mức. Nó sẽ được quyền sử dụng bộ xử lý nếu nó chuyển dịch đến đầu hàng đợi và không còn tiến trình nào trong các hàng đợi mức trên. Thường trong hệ thống có một vài hàng đợi mức thấp nhất, và hoạt động theo nguyên tắc quay vòng, theo đó các tiến trình được thực hiện quay vòng đến khi nó kết thúc.

Trong nhiều hệ thống, người ta thiết kế để lượng tử đối với hàng đợi mức thấp hơn có giá trị lớn hơn. Nhờ đó tiến trình chờ càng lâu thì được chiếm bộ xử lý lâu hơn. Tiến trình ở đầu hàng đợi chỉ được chiếm bộ xử lý nếu tất cả các hàng đợi mức trên là rỗng. Việc thực hiện tiến trình có thể bị ngắt nếu xuất hiện tiến trình mới thuộc hàng đợi mức trên.

Đến đây ta phân tích nguyên tắc này, đối chiếu với các yêu cầu trên. Cơ chế cần ưu tiên các tiến trình thường xuyên thực hiện thao tác vào/ra để đảm bảo hiệu suất sử

dụng thiết bị và thời gian trả lời yêu cầu là ngắn. Lượng tử được chọn đủ để tiến trình thực hiện xong tính toán và yêu cầu thao tác vào/ra trước khi kết thúc lượng tử. tiến trình được đưa vào cuối hàng đợi và vẫn được gán mức ưu tiên cao. Tiến trình sẽ nhanh chóng dịch chuyển lên đầu hàng đợi mức trên và được phục vụ.

Còn với các tiến trình thực hiện tính toán nhiều, đầu tiên tiến trình được đưa vào hàng đợi mức trên. Tiến trình nhanh chóng dịch chuyển lên đầu hàng đợi và được phục vụ. Sau khi kết thúc lượng tử, nó được đưa vào hàng đợi mức dưới với mức ưu tiên thấp hơn. Như thế cơ chế này ưu tiên các tiến trình thường thực hiện thao tác vào/ra. Sau đó, khi không còn tiến trình ở hàng đợi mức trên, tiến trình được phục vụ lần tiếp theo và khi kết thúc lượng tử, nó được chuyển xuống hàng đợi mức thấp hơn. Cuối cùng nó chuyển xuống hàng đợi thấp nhất và được phục vụ quay vòng đến khi kết thúc.

Cơ chế hàng đợi nhiều mức với liên hệ ngược là cơ chế tốt cho phép phân tách các tiến trình theo thời gian sử dụng bộ xử lý. Hệ thống có thể ghi lại mức hàng đợi của tiến trình, như thế khi tiến trình được đưa ngược lại hàng đợi, nó có thể được đưa vào hàng đợi cùng mức.

Ta có thể thấy rằng nếu tiến trình đã nằm ở hàng đợi mức thấp nhất thì hệ thống không thể phản ứng với thay đổi hành vi của tiến trình ví dụ như tiến trình chuyển từ pha tính toán sang pha thực hiện thao tác vào/ra. Vấn đề có thể được giải quyết nếu hệ thống ghi lại thời gian các lần tiến trình chiếm bộ xử lý, nhờ đó khi chuyển pha, hệ thống có thể nhanh chóng phát hiện sự kiện đó và đưa tiến trình vào hàng đợi thích hợp. Hoặc dùng phương án khác, theo đó mỗi khi tiến trình tự giải phóng bộ xử lý trước khi hết lượng tử, hệ thống lại tăng mức ưu tiên cho tiến trình.

Cơ chế này là cơ chế thích nghi – có thể phản ứng với các thay đổi trong hành vi của tiến trình. Thông thường cơ chế thích nghi đòi hỏi chi phí lớn hơn, nhưng nó giúp hệ thống linh động hơn, tự điều chỉnh hoạt động và lợi ích đạt được nói chung cân bằng được với chi phí.

3.4. Đa xử lý

Giới thiệu: một trong những hướng phát triển của kỹ thuật tính toán là việc phổ biến các hệ đa xử lý, tức là các hệ thống có nhiều bộ xử lý. Các kiến trúc đa xử lý đã xuất hiện từ lâu nhưng đến thời gian gần đây mới trở nên phổ biến.

Đa xử lý cho phép xây dựng các hệ thống chứa hàng chục, thậm chí hàng trăm bộ xử lý, đạt được tốc độ cao với chi phí thấp.

3.5. Các ưu điểm của hệ đa xử lý

3.5.1. Tínhs ổn định, sẵn sàng

Một trong các ưu thế chính của các hệ đa xử lý là tính sẵn sàng. Trong trường hợp có 1 bộ xử lý gặp sự cố, hệ thống vẫn có thể tiếp tục hoạt động. Để thực hiện được điều đó, bộ xử lý gặp sự cố cần thông báo cho các bộ xử lý khác về tình trạng sự cố để chuyển giao công việc. Hệ điều hành cần biết tình trạng của các bộ xử lý để xác định lại tài nguyên, điều chỉnh hoạt động của hệ thống.

3.5.2. Tính song song

Một mục đích quan trọng của các hệ đa xử lý là tăng tốc độ tính toán. Tuy nhiên, phần lớn chương trình được thiết kế để hoạt động tuần tự. Điều đó do một số nguyên nhân:

- Con người nói chung suy nghĩ tuần tự.

- Trong ngôn ngữ cũng không có các công cụ thuận lợi để mô tả tính song song mặc dù có các ngôn ngữ lập trình hỗ trợ song song như ADA.
- Bản thân các hệ đa xử lý cũng không được áp dụng tính song song để thiết kế.
- Việc kiểm tra tính đúng đắn của chương trình song song phức tạp hơn so với tuần tự.

3.5.3. Mục tiêu của các hệ đa xử lý

Nhờ việc áp dụng đa xử lý, có thể xây dựng hệ thống tốc độ cao với chi phí thấp.

Các hệ đa xử lý có tính ổn định và sẵn sàng cao.

Các hệ đa xử lý có tính mềm dẻo, cho phép thay thế, mở rộng hệ thống dễ dàng.

3.5.4. Tính toán song song tự động

Các hệ thống đa xử lý cho phép tính toán song song, tuy nhiên phần lớn chương trình được thiết kế cho thực hiện tuần tự. Việc xác định có thể tính toán song song do người lập trình, do trình biên dịch hay hệ điều hành xác định là vấn đề phức tạp.

Tính toán song song có thể do người lập trình chỉ ra, ví dụ với các lệnh tính toán song song. Tuy nhiên việc chỉ ra tính toán tường minh như trên là công việc phức tạp và khó khăn vì người lập trình còn phải xử lý các vấn đề nghiệp vụ, dễ xảy ra lỗi (ví dụ những đoạn không tính song song được thì tính song song và bỏ sót các trường hợp tính toán phức tạp).

Việc phân tích xử lý song song có thể được giải quyết nhờ việc xác định tự động các tính toán có thể chạy song song. Việc xác định đó có thể do trình biên dịch hay hệ điều hành hay phần cứng đảm nhiệm.

3.5.5. Phân bố vòng lặp

Trong chương trình, nhiều đoạn mã được thực hiện trong vòng lặp. Nhiều trường hợp các đoạn mã trong thân vòng lặp độc lập và do đó có thể tính toán song song.

Ví dụ:

```
for i:= 1 to 10 do
G[i]:= b[i] + c[i]
```

Giảm độ cao cây tính toán.

Trong trường hợp tính toán tuần tự, ta sẽ phải lần lượt tính cho từng lần lặp:

```
G[1]:= b[1] + c[1]
G[2]:= b[2] + c[2]
...
G[10]:= b[10] + c[10]
```

Trong hệ đa xử lý, các phép toán này có thể được thực hiện song song trên nhiều bộ xử lý, nhờ thế giảm thời gian tính toán. Trình biên dịch sẽ biên dịch thành các lệnh tính toán song song:

```
Parbegin
    G[1]:= b[1] + c[1]
    G[2]:= b[2] + c[2]
    ...
    G[10]:= b[10] + c[10]
Parend
```

Phương án này được gọi là phân bố lại vòng lặp. Việc áp dụng tiếp cận này cũng tương đối dễ dàng.

3.5.6. Giảm độ cao cây tính toán

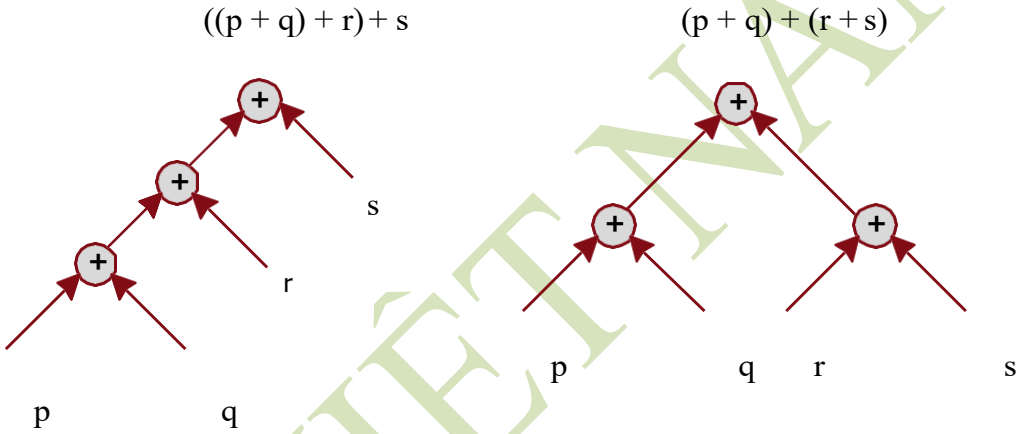
Sử dụng các tính chất kết hợp, phân phối, bắc cầu, … trong số học, trình biên dịch có thể xử lý tính toán song song các biểu thức và sinh mã lệnh tương ứng.

Thông thường, trình biên dịch cần thực hiện phân tích cú pháp theo quy tắc ưu tiên các phép toán, từ đó sinh ra mã lệnh thực thi.

Tuy nhiên, không nhất thiết luôn phải thực hiện theo thứ tự từng bước đó.

Ví dụ 1: Các phép cộng có tính giao hoán:p*q=q*p, sử dụng các tính chất như giao hoán, kết hợp, phân phối, trình biên dịch có thể thay đổi thứ tự tính toán biểu thức thích hợp cho tính toán song song.

Ví dụ 2: Trong ví dụ tiếp theo, ta có thể thấy trình biên dịch xử lý biểu thức như thế nào trong trường hợp tuần tự thông thường và khi biên dịch cho hệ thống tính toán song song.



Áp dụng tính chất kết hợp có thể biến đổi biểu thức ((p + q) + r) + s = (p + q) + (r + s) về dạng thích hợp hơn cho tính toán song song.

Biểu thức ban đầu biểu diễn bằng cây cú pháp có độ sâu 3 còn cây tương ứng với biểu thức đã biến đổi (bên phải) có độ sâu 2 và có thể được thực hiện nhanh hơn trên các hệ đa xử lý.

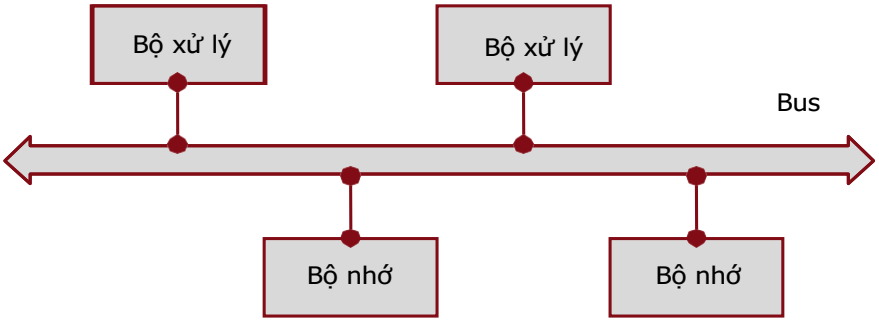
3.6. Các mô hình kết nối

Một trong những vấn đề cơ bản khi thiết kế các hệ đa xử lý là phương án kết nối các bộ xử lý, các module bộ nhớ, các kênh vào/ra. Nói chung, hệ đa xử lý đặc trưng bởi các yếu tố:

- Có từ 2 bộ xử lý.
- Tất cả các bộ xử lý đều có thể truy cập bộ nhớ chung.
- Tất cả các bộ xử lý đều có thể truy cập kênh vào/ra.
- Hệ thống làm việc dưới sự điều khiển của một hệ điều hành.

3.6.1. Đường truyền chung

Tổ chức đường truyền chung sử dụng một kênh kết nối duy nhất để kết nối các cấu thành của hệ thống (bộ xử lý, module bộ nhớ,..). Trong sơ đồ này, đường truyền chung đóng vai trò thiết bị thụ động, các tác vụ trao đổi thông tin giữa các cấu thành được thực hiện dưới sự điều khiển của giao tiếp với đường truyền.



Cơ chế hoạt động thường dùng tương tự CSMA/CD. Theo đó, trước khi truyền dữ liệu bộ xử lý cần kiểm tra xem đường truyền có trống/rỗi không và cấu thành nhận đã sẵn sàng chưa, sau đó mới truyền dữ liệu. Cấu thành nhận cần xác nhận về việc nhận dữ liệu.

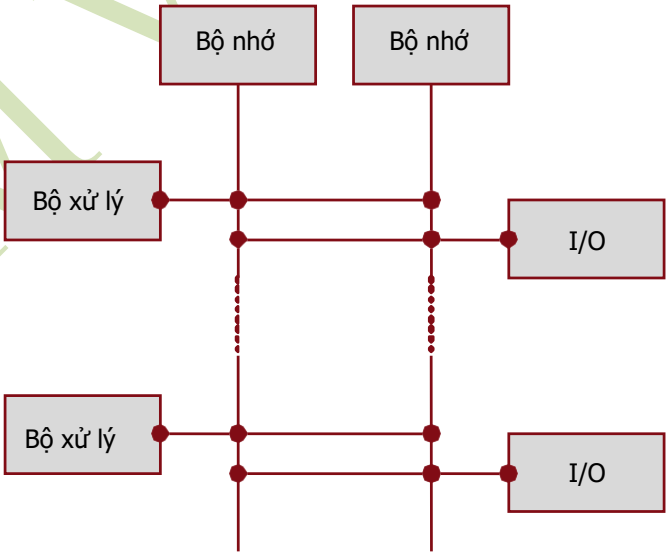
Kiến trúc đường truyền cho phép dễ dàng gắn kết thêm cấu thành mới bằng cách nối chúng với đường truyền.

Nhược điểm cơ bản của hệ thống với một đường truyền:

- Nếu đường truyền gặp sự cố thì ảnh hưởng đến cả hệ thống.
- Tốc độ truyền trong hệ thống thường bị nghẽn tại đường truyền.
- Các xung đột xảy ra trên đường truyền khi số lượng yêu cầu tác vụ truyền lớn có thể dẫn đến giảm đáng kể hiệu suất của đường truyền.

3.6.2. Ma trận chuyển mạch

Trong mô hình đường truyền chung, nếu tăng số đường truyền lên bằng số module bộ nhớ thì ta có mô hình ma trận chuyển mạch. Trong mô hình này, có thể đồng thời diễn ra 2 tác vụ truy cập bộ nhớ mà không bị ảnh hưởng lẫn nhau.



Thiết kế theo mô hình ma trận chuyển mạch có thể rất phức tạp. Ví dụ, bộ chuyển mạch cần phải giải quyết các vấn đề xung đột khi có nhiều yêu cầu truy cập đến cùng module bộ nhớ. Việc có nhiều đường truyền cho phép tốc độ đường truyền cũng như tính sẵn sàng cao hơn.

3.7. Hệ điều hành trong hệ thống đa xử lý

Hệ điều hành đa nhiệm và hệ điều hành đa xử lý có nhiều tính năng tương tự nhau. Dựa trên các kiến trúc phần cứng, có thể chia thành các loại cơ bản sau:

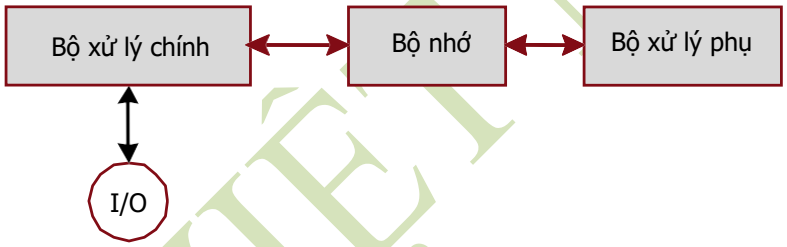
- Hệ điều hành bất đối xứng (chính-phụ).
- Hệ điều hành độc lập trên từng bộ xử lý.
- Hệ điều hành đa xử lý đối xứng.

3.7.1. Hệ điều hành bất đối xứng (chính-phụ)

Có thể nói đây là mô hình dễ cài đặt nhất, có thể mở rộng từ hệ thống đa nhiệm. Trong mô hình này, hệ điều hành (các tiến trình của nó) thực thi trên bộ xử lý chính. Trên các bộ xử lý khác chỉ thực thi các tiến trình của ứng dụng. Khi ứng dụng cần gọi dịch vụ của hệ điều hành, nó gửi tín hiệu ngắt và chờ bộ xử lý chính xử lý.

Việc giải quyết loại trừ tương đối đơn giản vì các tiến trình của hệ điều hành chỉ thực thi trên 1 bộ xử lý.

Mô hình này có tính sẵn sàng thấp hơn so với các mô hình khác do nếu bộ xử lý chính gặp sự cố thì cả hệ thống không vận hành được.



3.7.2. Hệ điều hành độc lập trên từng bộ xử lý

Theo mô hình này, mỗi bộ xử lý có hệ điều hành riêng, hệ điều hành này phục vụ các tiến trình ứng dụng chạy trên cùng bộ xử lý. Với các thông tin chung của hệ thống (ví dụ danh sách tiến trình, ...) cần có cơ chế điều khiển truy cập chặt chẽ, tính đến loại trừ. Mức an toàn của mô hình này cao hơn so với mô hình bất đối xứng ở trên. Một bộ xử lý gặp sự cố ít khi kéo theo sự cố của cả hệ thống.

Mỗi bộ xử lý quản lý các tài nguyên riêng như kênh vào ra (I/O), module bộ nhớ, .. như một máy tính có 1 bộ xử lý.

3.7.3. Hệ thống đa xử lý đối xứng

Hệ đa xử lý đối xứng (Symmetric Multi Processsing – SMP) là mô hình phức tạp nhưng có hiệu quả cao nhất. Trong mô hình này, tất cả các bộ xử lý giống nhau và có vai trò như nhau. Hệ điều hành quản lý và lập lịch cho nhiều bộ xử lý. Các chương trình có thể được thực thi đồng thời trên nhiều bộ xử lý, do đó cần xử lý các tình huống loại trừ. Nhờ đa xử lý đối xứng, hệ thống có thể thực hiện cân bằng tải tốt hơn các mô hình khác. Trong mô hình này, việc xử lý xung đột có vai trò quan trọng. Thường xung đột do các bộ xử lý cùng truy cập 1 module bộ nhớ được giải quyết bằng thiết bị phần cứng, còn xung đột do các tiến trình truy cập thông tin hệ thống thường giải quyết bằng chương trình.

Các tiến trình của hệ điều hành có thể thực hiện trên nhiều bộ xử lý. Mô hình đa xử lý đối xứng cho phép sử dụng tài nguyên hiệu quả hơn, dễ dàng phân tải trên các bộ xử lý.

TÓM LƯỢC CUỐI BÀI

Các bạn đã được học về lập lịch trong hệ thống.

Ta cần ghi nhớ các vấn đề sau:

- Mục tiêu của lập lịch.
- Tiêu chuẩn để lập lịch.
- Tiêu chuẩn đánh giá.
- Các giải thuật lập lịch:
    - Định thời hạn chót.
    - FIFO.
    - SJF.
    - SRT.
    - RR.
    - HRN.
    - Hàng đa mức hồi quy.

Giải được bài tập về tìm hiểu và mô tả các bài toán lập lịch.

CÂU HỎI TỰ LUẬN

Câu 1. Một hệ thống có 3 tiến trình với thời điểm đến và thời gian sử dụng CPU như sau:

Tiến trình	Thời điểm đến (ms)	CPU-Burst (ms)
P1	3	3
P2	10	20
P3	24	14

Dùng thuật giải Round - Robin với thời lượng 10 ms để điều phối CPU:

a) Thể hiện bằng biểu đồ Gantt.

b) Tính thời gian chờ trung bình của các tiến trình.

Câu 2. Xét một tập hợp các tiến trình đến tại thời điểm 0, với chiều dài thời gian của CPU - Burst được tính bằng ms:

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Các tiến trình được giả định đến theo thứ tự P1, P2, P3, P4, P5.

a) Vẽ biểu đồ Gantt minh họa việc thực thi các tiến trình sử dụng các cơ chế lập lịch FCFS, SJF, cơ chế định thời không trưng dụng (Non-Preemptive Priority), RR (quantum=1)

b) Thời gian hoàn thành mỗi tiến trình ứng với từng giải thuật trên bằng bao nhiêu?

c) Thời gian chờ cho mỗi tiến trình ứng với từng giải thuật trên bằng bao nhiêu?

d) Từ kết quả trên, giải thuật nào cho thời gian chờ trung bình nhỏ nhất.

Câu 3. Xét một hệ thống đang chạy 10 thao tác hướng nhập/xuất và 1 thao tác hướng xử lý. Giả sử rằng thao tác của hướng nhập xuất gồm: một toán tử vào/ra cho mỗi ms của tính toán CPU và mỗi toán tử vào ra thì cần 10ms để hoàn thành. Và giả sử chuyển đổi ngữ cảnh lúc ban đầu là 0.1ms và tất cả các quá trình là thao tác thực thi dài. Việc sử dụng CPU với phương pháp lập lịch RR là gì khi:

a) Thời gian lượng tử là 1ms.

b) Thời gian lượng tử là 10ms?

Câu 4. Một hệ thống có 4 tiến trình:

Tiến trình	Thời điểm đến(ms)	Burst time(ms)
P1	0	7
P2	2	4
P3	4	1
P4	5	4

a) Vẽ biểu đồ Gantt định thời theo SJF.

b) Thời gian đợi trung bình là bao nhiêu?

Câu 5. Giả sử các tiến trình đến theo thứ tự P1, P2, P3:

Tiến trình	Burst time (ms)
P1	24
P2	3
P3	3

a) Vẽ biểu đồ Gantt cho giải thuật lập lịch theo nguyên tắc FIFO?

b) Thời gian đợi cho mỗi tiến trình là bao nhiêu?

c) Thời gian trung bình là bao nhiêu?

BÀI TẬP TRẮC NGHIỆM

1. Khi hệ thống phải truy xuất dữ liệu khối lượng lớn thì thuật toán lập lịch nào sau đây hiệu quả?

a) FCFS. b) SCAN.

c) C-SCAN. d) Cả đáp án a, b, c là đúng.

2. Khi hệ thống phải truy xuất dữ liệu có số khối liên tục thì thuật toán lập lịch nào sau đây hiệu quả nhất?

a) FCFS. b) SSTF. c) SCAN. d) C - SCAN.

3. Cần đọc khối sau: 98, 183, 37, 122, 14, 122, 65, 67 đầu đọc tại vị trí 53, dùng thuật toán lập lịch FCFS thì đầu đọc sẽ lần lượt qua các khối có thứ tự nào sau đây?

a) 53, 37, 14, 65, 67, 98, 122, 183. b) 53, 65, 67, 98, 122, 124, 183, 37, 14.

c) 53, 14, 37, 65, 67, 98, 122, 124, 183. d) 53, 98, 183, 37, 122, 14, 124, 65, 67.

4. Ví dụ cần đọc các khối sau 98, 183, 37, 122, 14, 122, 65, 67 đầu đọc tại vị trí 53, dùng thuật toán lập lịch SSTF thì đầu đọc sẽ lần lượt qua các khối có thứ tự nào sau đây?

a) 53, 37, 14, 65, 67, 98, 122, 124, 183. b) 53, 65, 67, 37, 14, 98, 122, 124, 183.

c) 53, 14, 37, 65, 67, 98, 122, 124, 183. d) 53, 183, 124, 122, 98, 67, 65, 37, 14.

5. Ví dụ cần đọc các khối sau 98, 183, 37, 122, 14, 122, 65, 67 đầu đọc tại vị trí 53, dùng thuật toán lập lịch SCAN thì đầu đọc sẽ lần lượt qua các khối có thứ tự nào sau đây?

a) 53, 37, 14, 65, 67, 98, 122, 124, 183. b) 53, 65, 67, 98, 122, 124, 183, 37, 14.

c) 53, 183, 124, 122, 98, 67, 65, 37, 14. d) Đáp án a, b là đúng.

6. Kỹ thuật nhập/xuất nào sau đây làm CPU ít bận rộn nhất?

a) Busy_waitting. b) Interrup. c) DMA. d) Không có đáp án đúng.

7. Giải thuật lập lịch nào mà kết quả có thể bị đói CPU hay nghẽn không hạn định?

a) FCFS. b) CAN. c) RR. d) Priority.