

# Thuật toán lập lịch cho hệ thống thời gian thực

Dặng Quang Trung

Hệ Điều Hành

Ngày 31 tháng 7 năm 2017

# Nội dung



- 1 Hệ thống thời gian thực
- 2 Phân lớp thuật toán
- 3 Thuật toán EDF
- 4 RM

- Trong thế giới vật lý, mục đích của một hệ thống thời gian thực là có một thực hiện vật lý trong một khung thời gian đã chọn.
- Thông thường, một hệ thống thời gian thực bao gồm một hệ thống điều khiển( máy tính ) và một hệ thống bị điều khiển ( môi trường ).
- Mỗi công việc xảy ra trong một hệ thống thời gian thực có một số thuộc tính thời gian. Các thuộc tính thời gian này cần được xem xét khi lập kế hoạch các nhiệm vụ trên một hệ thống thời gian thực.  
Release time (or ready time), Deadline, Minimum delay, Maximum delay, Worst case execution time, Run time, Weight (or priority).

- Một hệ thống chứa một tập các tác vụ:

$$T = \{\tau_1, \tau_2, \dots, \tau_n\}$$

- Thời gian thực hiện của mỗi tác vụ là  $C_i$  với  $\tau_i \in T$
- Hệ thống được cho là thời gia thực nếu có tồn tại ít nhất tác vụ  $\tau_i \in T$ , tác vụ rơi vào tình trạng:

- 1, Tác vụ  $\tau_i$  là tác vụ **hard real-time**. Thời gian thực hiện tác vụ  $\tau_i$  phải được hoàn thành bởi thời gian hết hạn  $D_i$  ( $C_i \leq D_i$ ).
- 2, Tác vụ  $\tau_i$  là tác vụ **soft real-time**. Tác vụ cuối cùng  $\tau_i$  kết thúc tính toán của nó sau gian gian hết hạn  $D_i$ , sẽ bị phạt nặng hơn. Hàm phạt  $P(\tau_i)$  được định nghĩa cho tác vụ. Nếu  $C_i \leq D_i$  hàm phạt  $P(\tau_i)$  bằng 0 trái lại  $P(\tau_i) > 0$  giá trị này tăng theo  $C_i - D_i$ .
- 3, Tác vụ là **firm real-time**. Tác vụ kết thúc công việc tính toán của nó sớm hơn thời hạn hết hạn  $D_i$ , sẽ nhận được thưởng. Hàm thưởng  $R(\tau_i)$  được định nghĩa cho tác vụ. Nếu  $C_i \geq D_i$ , hàm  $R(\tau_i)$  bằng 0, trái lại  $R(\tau_i) > 0$  giá trị này tăng theo  $D_i - C_i$ .

# Hàm thưởng và phạt của hệ thống



- Một tập các tác vụ thời gian thực  $T = \{\tau_1, \tau_2, \dots, \tau_3\}$  có thể là hỗn hợp của các tác vụ hard, soft, firm thời gian thực.
- $T_s$  là tập tất cả các tác vụ **soft real-time** trong  $T$ , ví dụ  $T_s = \{\tau_{s,1}, \tau_{s,2}, \dots, \tau_{s,l}\}$  với  $\tau_{s,i} \in T$ . Hàm lỗi của hệ thống sẽ là  $P(T)$

$$P(T) = \sum_{i=1}^l P(\tau_{s,i})$$

- $T_f$  là tập tất cả các tác vụ **firm real-time** trong  $T$ , ví dụ  $T_f = \{\tau_{f,1}, \tau_{f,2}, \dots, \tau_{f,k}\}$  với  $\tau_{f,i} \in T$ . Hàm thưởng của hệ thống sẽ là  $R(T)$

$$R(T) = \sum_{i=1}^k R(\tau_{f,i})$$

- Mục đích của lập lịch
  - ▶ Tối ưu hóa hiệu năng của hệ thống, thông lượng.
  - ▶ Làm thế nào? sắp xếp các tiến trình.
  - ▶ Thuật toán lập lịch Cố gắng để đạt được kết quả tốt nhất có thể (trung bình hoặc cho một bộ quy trình nhất định)
- Tiêu chí tối ưu
  - ▶ Thời gian đáp ứng, thời gian chờ đợi, thời gian đáp ứng của quá trình.
  - ▶ Sử dụng bộ vi xử lý.
  - ▶ Thông lượng (quy trình hoàn thành cho mỗi đơn vị thời gian).
  - ▶ Đáp ứng các thời hạn: tính khả thi, khả năng lập lịch

- Static

- ▶ Tất cả các thông số về độ ưu tiên được biết trước.
- ▶ Lập lịch có thể được hoàn thành trong thời gian thiết kế, do đó hiệu quả cao trong thời gian chạy.
- ▶ Áp dụng cho các hệ thống nhúng đóng.
- ▶

- Dynamic

- ▶
- ▶ Không phải tất cả các tham số đều được biết trước
- ▶ Lập lịch trực tuyến.
- ▶ Mục tiêu tối ưu thường có thể đạt được chỉ khoảng.
- ▶ Lập lịch yêu cầu hiệu năng tính toán.
- ▶ Hoạt động hệ điều hành thông thường cho hệ thống tương tác

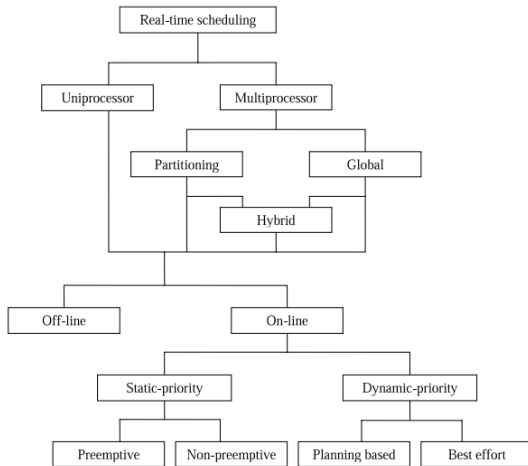


# Phân lớp thuật toán lập lịch (2)



- Lập lịch không trưng dụng.
  - ▶ Hệ điều hành không thu hồi bộ xử lý từ tiến trình đang chạy, nó chạy cho đến khi nó kết thúc hoặc trả lại bộ xử lý.
  - ▶ Ưu điểm: đơn giản thực hiện thay đổi quy trình.
  - ▶ Nhược điểm: không hữu ích cho các hệ thống thời gian thực, nơi mà phản ứng nhanh với các sự kiện bên ngoài là cần thiết
- Lập lịch trưng dụng
  - ▶ Hệ điều hành sẽ thu hồi bộ xử lý của tiến trình đang chạy bất cứ khi nào có nhu cầu lập lịch lại.

# Sơ đồ cấu trúc phân lớp tổng quát



Hình: Sơ đồ lập lịch

# Lập lịch hệ thống thời gian thực



- Earliest Deadline First(EDF)
- Rate-monotonic (RM)

# Earliest Deadline First(EDF)



- Lập lịch dựa trên thời gian hết hạn
  - ▶ Chỉ sử dụng thời gian hết hạn của tiến trình lên kế hoạch, không dùng thời gian tính toán.
  - ▶ Bộ xử lý được gán cho tác vụ với thời gian hết hạn gần nhất (thời gian hết hạn sớm nhất, EDF).
  - ▶ Không trưng dụng: Lập lịch lại khi tác vụ hiện tại thực hiện xong.
  - ▶ Trưng dụng: Lập lịch lại khi có 1 tiến trình mới sẵn sàng.

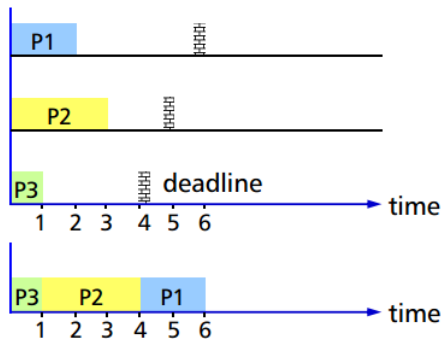
- Static EDF cùng thời gian sẵn sàng.
  - ▶ Sắp xếp các tiến trình với thời gian hết hạn tăng dần.
  - ▶ Bất cứ khi nào một tiến trình được lập lịch, kiểm tra xem thời hạn của nó có được đáp ứng hay bị vi phạm.
  - ▶ Cho k tiến trình nó thỏa mãn.

$$\sum_{i=1}^k C_i \leq D_k \quad k = 1 \dots n$$

# Earliest Deadline First



- Ví dụ:



- EDF là tối ưu thời gian sẵn sàng giống nhau.

# Earliest Deadline First



## • Chứng minh

- ▶ Bộ xử lý luôn được sử dụng vì  $R_i = 0$ .
- ▶ Nếu lập lịch đã được tiến hành đúng thời điểm trước khi thêm  $P_k$ , sau đó sẽ không thêm được bất kì gì nữa:

$$\sum_{i=1}^k C_i > D_k$$

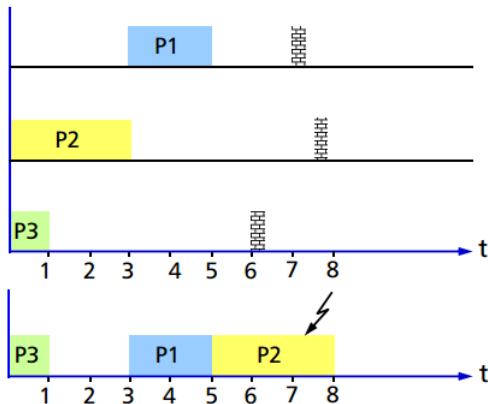
- ▶ Khả năng duy nhất là sau đó trao đổi  $P_k$  với một trong các tiến trình đã được lập lịch  $P_j$  ( $j < k$ ).
- ▶ Tổng thời gian thực hiện vẫn không thay đổi.
- ▶ Chúng ta biết rằng  $D_j \leq D_k$  và do đó:

$$\sum_{i=1}^k C_i > D_j$$

- ▶ Do đó sự trao đổi làm cho mọi thứ tồi tệ hơn.
- ▶ Nếu thuật toán không tìm được giải pháp nào khác.

# Earliest Deadline First

- Static EDF với thời gian sẵn sàng khác nhau
  - ▶ Không trưng dụng là không tối ưu

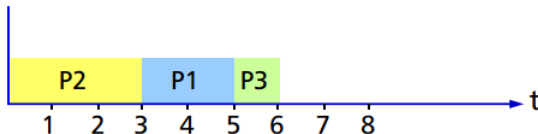




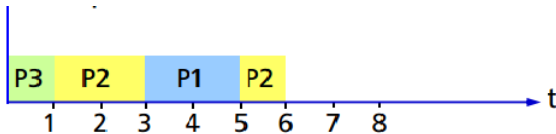
# Earliest Deadline First



- Có lập lịch tối ưu khác ví dụ:



- Thuật toán EDF sẽ tối ưu cho trường hợp này nếu có trưng dụng
- Ví dụ:



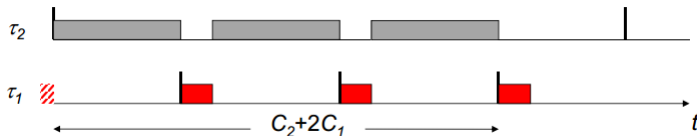
- Giả thiết
  - ▶ Độ ưu tiên của tác vụ được gán cho các tác vụ trước khi thực hiện và không thay đổi theo thời gian (độ ưu tiên là tĩnh)
  - ▶ Trưng dụng, tác vụ đang thực hiện ở hiện tại bị trưng dụng bởi một tác vụ có mức độ ưu tiên cao hơn.
  - ▶ Thời gian hết hạn bằng với chu kì  $D_i = T_i$
- Thuật toán: Mỗi tác vụ được gán một mức độ ưu tiên. Tác vụ có tỷ lệ yêu cầu cao hơn (có chu kì ngắn hơn) sẽ có những ưu tiên cao hơn. Các tác vụ với các tác vụ ngắt ưu tiên cao hơn với mức độ ưu tiên thấp hơn

- Tối ưu: RM là tối ưu trong số tất cả các thuật toán có độ ưu tiên cố định tức là không có thuật toán khác có độ ưu tiên cố định có thể lập lịch một tập công việc mà không thể được lập lịch bởi RM.
- Chứng minh:
  - ▶ Một thời điểm bất kỳ tác vụ nào được sẵn sàng đồng thời với các tác vụ có độ ưu tiên cao hơn. Các tác vụ lập lịch có thể dễ dàng kiểm tra trường hợp quan trọng. Nếu tất cả các tác vụ đều khả thi ở những thời điểm quan trọng của nó, thì tập các tác vụ có thể lập lịch trong mọi điều kiện khác.
  - ▶ Với hai tác vụ định kỳ, nếu lập lịch là khả thi theo phân bổ ưu tiên tùy ý, thì nó cũng có thể thực hiện được bởi RM
  - ▶ Mở rộng kết quả cho tập  $n$  tác vụ định kỳ.

**Định nghĩa:** một thời điểm quan trọng của 1 tác vụ là thời gian sẵn sàng của 1 tác vụ sẽ tạo ra thời gian phản hồi là lớn nhất.

**Bổ đề:** Đối với bất kỳ tác vụ nào, thời điểm quan trọng xảy ra nếu tác vụ đó được sẵn sàng đồng thời với tất cả các tác vụ ưu tiên cao hơn.

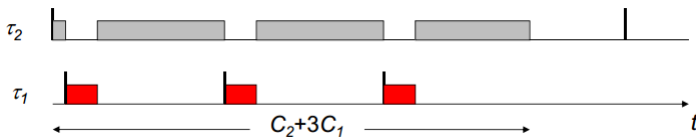
Với 2 tác vụ  $\tau_1$  và  $\tau_2$ . Thời gian phản hồi của  $\tau_2$  bị trì hoãn bởi tác vụ  $\tau_1$  có độ ưu tiên cao hơn.



# Chứng minh



Thời gian delay tăng lên khi  $\tau_1$  bắt đầu sớm hơn.



Thời gian delay tối đa khi  $\tau_1$  và  $\tau_2$  bắt đầu cùng nhau

Sửa lại tham số độ ưu tiên cho các tác vụ có độ ưu tiên cao hơn  $\tau_2$

Trường hợp tồi nhất thời gian đáp ứng của một tác vụ xảy ra khi nó được sẵn sàng đồng thời với tất cả các nhiệm vụ ưu tiên cao hơn.

# Chứng minh

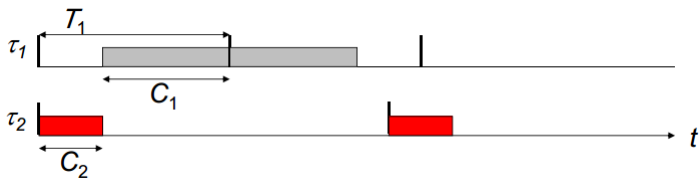


Chúng ta có 2 tác vụ  $\tau_1$  và  $\tau_2$  với chu kì  $T_1 < T_2$

Định nghĩa  $F = \lfloor T_2 / T_1 \rfloor$  số chu kì  $\tau_1$  thực hiện đầy đủ trong  $T_2$ .

Xem xét 2 trường hợp A và B:

A Giả sử không dùng RM độ ưu tiên  $\tau_2$  là lớn nhất.

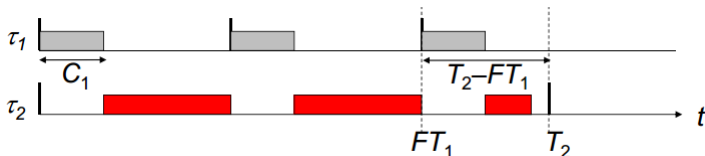


Lập lịch là có khả năng nếu  $C_1 + C_2 \leq T_1$

# Chứng minh



B: Giả sử dùng RM với độ ưu tiên  $\tau_1$  là lớn nhất.



Lập lịch là có thể nếu  $FC_1 + C_2 + \min(T_2 - FT_1, C_1) \leq T_2$  và  $C_1 \leq T_1$

Chúng ta có thể thấy(A) $\Rightarrow$ (B):  $C_1 + C_2 \leq T_1 \Rightarrow C_1 \leq T_1$

$C_1 + C_2 \leq T_1 \Rightarrow FC_1 + C_2 \leq FC_1 + FC_2 \leq FT_1 \Rightarrow FC_1 + C_2 + \min(T_2 - FT_1, C_1) \leq FT_1 + \min(T_2 - FT_1, C_1) \leq \min(T_2, C_1 + FT_1) \leq T_2$

Cho tác vụ  $\tau_1$  và  $\tau_2$  với  $T_1 < T_2$  Nếu lập lịch là khả thi bởi một phân công ưu tiên cố định tùy ý, nó cũng khả thi bởi RM.

- Một tập hợp các tác vụ định kỳ được lập lịch với RM nếu

$$\sum_{i=1}^n \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$$

Điều kiện cần nhưng không phải điều kiện đủ.



$$U = \sum_{i=1}^n \frac{C_i}{T_i}$$

Định nghĩa yếu tố sử dụng bộ vi xử lý



Thank You!

# Tài liệu tham khảo

