CT107. Hệ Điều Hành
Chương 4. Định Thời CPU

Giảng viên: Trần Công Án (tcan@cit.ctu.edu.vn)

Bộ môn Mạng máy tính & Truyền thông

Khoa Công Nghệ Thông Tin & Truyền Thông

Đại học Cần Thơ

- Phân CPU cho tiến trình nào.
- Cấp phát bộ nhớ trong tiến trình.
- Câu lệnh nào của tiến trình được chạy / chưa được chạy

2014

Mục Tiêu

Giới thiệu về **tác vụ định thời cho CPU** (CPU scheduling) trong các hệ điều hành đa chương, bao gồm:

- các tiêu chí cho việc định thời CPU
- các giải thuật định thời CPU
- các tiêu chí để lựa chọn 1 giải thuật định thời cho 1 hệ thống

Nội Dung

Các Khái Niệm Cơ Bản

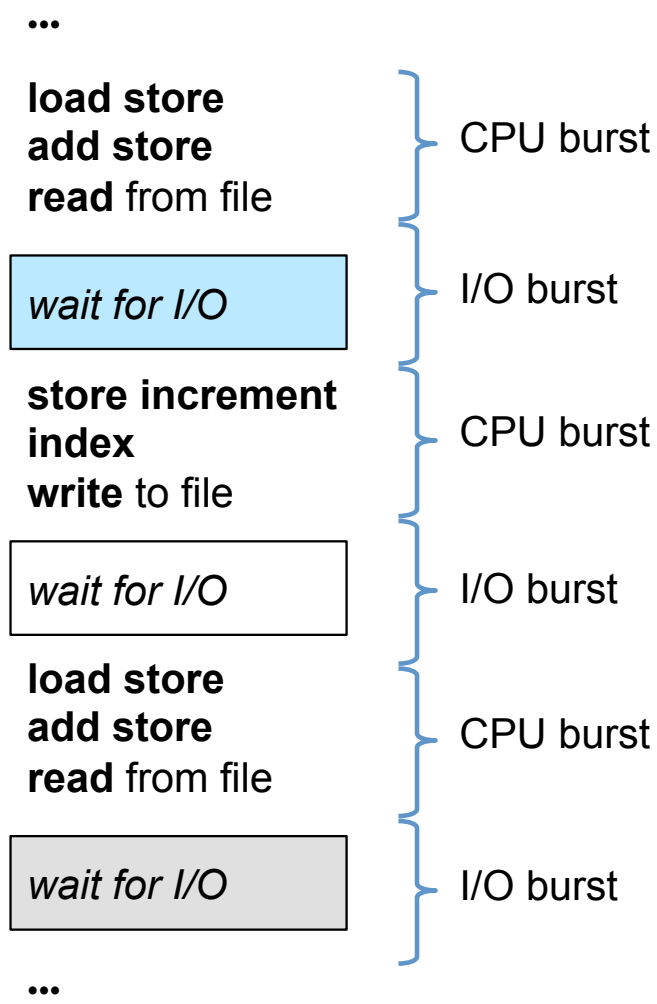
- Định thời biểu CPU là một chức năng cơ bản và quan trọng của các HĐH đa chương.
- **Chức năng**: phân bổ thời gian/thời điểm sử dụng CPU cho các tiến trình trong hệ thống, nhằm:
  - tăng hiệu năng (CPU utilisation) sử dụng CPU
  - giảm thời gian đáp ứng (response time) của hệ thống
- **Ý tưởng cơ bản**: phân bố thời gian rãnh rỗi của CPU (khi chờ đợi tiến trình đang thực thi thực hiện các thao tác nhập xuất) cho các tiến trình khác trong hệ thống.

Thời gian từ yêu cầu chạy tiến trình → bắt đầu chạy tiến

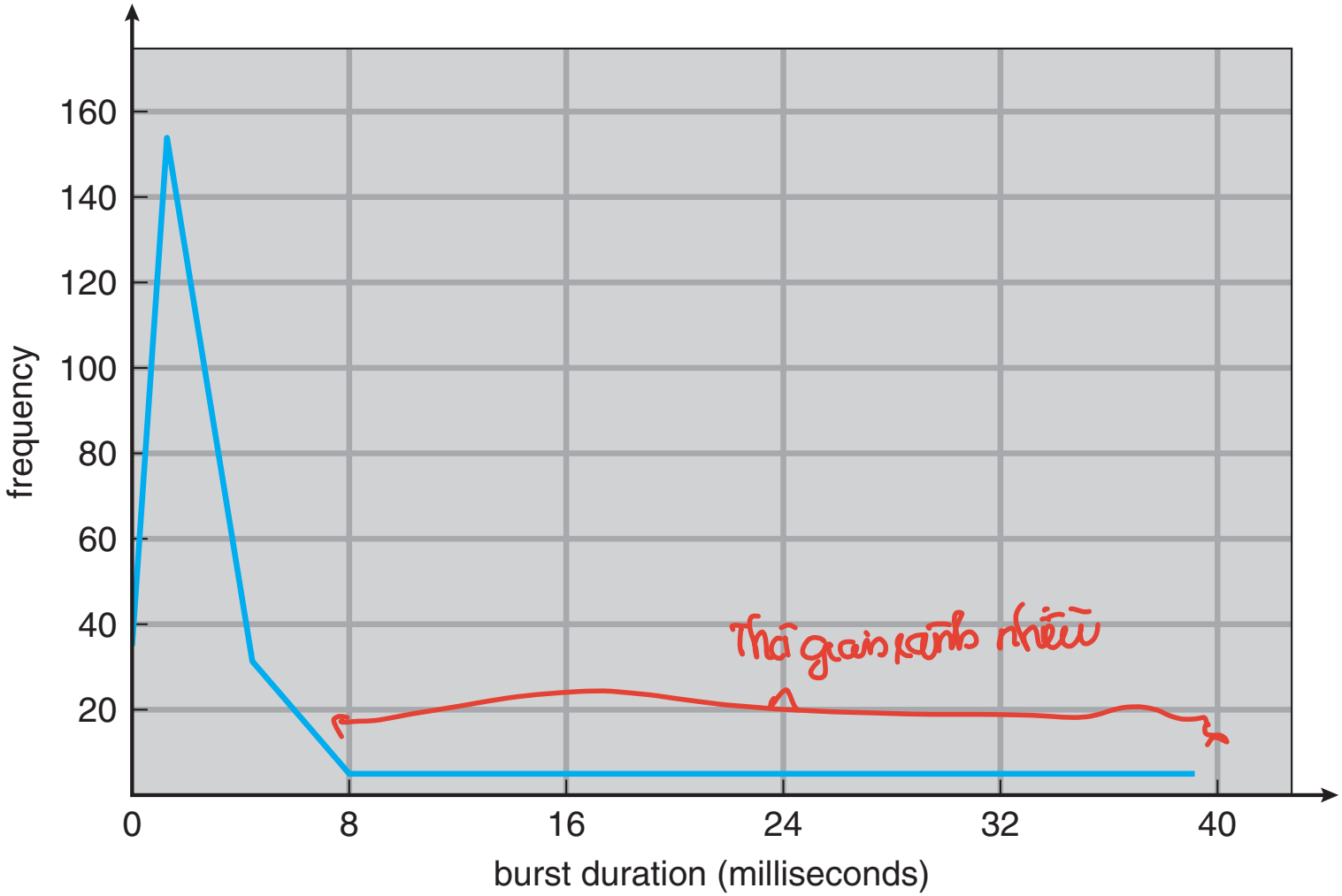
Chu Kỳ CPU–I/O (CPU–I/O Burst)

- **Chu kỳ CPU–I/O**:
  - Sự thực thi của tiến trình bao gồm nhiều chu kỳ CPU–I/O.
  - Một chu kỳ CPU–I/O bao gồm chu kỳ thực thi CPU (CPU burst) và chu kỳ chờ đợi vào/ra (I/O burst).
- **Sự phân bổ sử dụng CPU**:
  - Chương trình hướng nhập xuất (I/O-bound) thường có nhiều chu kỳ CPU ngắn.
  - Chương trình hướng xử lý (CPU-bound) thường có nhiều chu kỳ CPU dài.

Ví Dụ Về Chu Kỳ CPU–I/O



Ví Dụ Về Phân Bổ Sử Dụng CPU



Bộ Định Thời CPU (CPU Scheduler)

- Còn gọi là **bộ định thời ngắn kỳ**, chọn một trong các tiến trình trong hàng đợi sẵn sàng và cấp phát CPU cho nó thực thi.
- Quyết định định thời xảy ra khi một tiến trình:
  1. chuyển từ trạng thái đang chạy sang trạng thái chờ đợi
  2. chuyển từ trạng thái đang chạy sang trạng thái sẵn sàng
  3. chuyển từ trạng thái chờ đợi sang trạng thái sẵn sàng
  4. kết thúc

Định Thời Trưng Dụng & Không Trưng Dụng

- **Định thời không trưng dụng** (nonpreemptive scheduling):
  - Tiến trình được phân CPU có quyền sử dụng CPU đến khi sử dụng xong (k/thúc hoặc chuyển sang trạng thái chờ, như trường hợp 1 và 4).
- **Định thời trưng dụng** (preemptive scheduling):
  - Bộ định thời có thể thu hồi CPU của tiến trình bất kỳ lúc nào để phân cho tiến trình khác (trường hợp 2 và 3).
  - Phức tạp hơn định thời không trưng dụng vì nó phải giải quyết:
    - sự cạnh tranh dữ liệu giữa các tiến trình.
    - sự trưng dụng khi tiến trình đang thực thi trong chế độ kernel.
    - dàn xếp giữa sự trưng dụng và xử lý các ngắt của hệ thống.

Bộ Điều Phối (Dispatcher)

- Có nhiệm vụ thực thi việc trao quyền sử dụng CPU cho tiến trình được cấp phát CPU bởi bộ định thời.
- Bao gồm các tác vụ:
  - Chuyển ngữ cảnh
  - Chuyển sang chế độ người dùng
  - Nhảy tới vị trí thích hợp trong chương trình người dùng để khởi động lại chương trình đó.
- **Độ trễ điều phối** (dispatcher latency): thời gian dispatcher cần để ngưng một tiến trình và khởi động một tiến trình khác.

Tiêu Chí Cho Việc Định Thời

Yếu tố quyết định giải thuật đó có tốt hay không

1. **Hiệu suất sử dụng CPU**: tỷ lệ giữa thời gian CPU được sử dụng trên tổng thời gian hoạt động của hệ thống. CPU được sử dụng liên tục
2. **Thời gian đáp ứng** (response time): lượng thời gian từ lúc một yêu cầu được đệ trình cho đến khi bắt đầu được đáp ứng.
3. **Thời gian chờ đợi** (waiting time): tổng thời gian 1 tiến trình nằm trong hàng đợi sẵn sàng (ready queue).
4. **Thời gian xoay vòng** (turnaround time): tổng thời gian để thực thi một t/trình, bao gồm các khoảng t/gian: thực thi, chờ I/O, chờ trong ready queue (= t/điểm kết thúc – t/điểm bắt đầu vào ready queue).
5. **Thông lượng** (throughput): số lượng tiến trình hoàn thành trên một đơn vị thời gian. Càng lớn càng tốt.

Tối Ưu Hóa Các Tiêu Chí

Các giải thuật định thời được đánh giá thông qua khả năng tối ưu hóa các tiêu chí định thời của nó:

1. Hiệu suất sử dụng CPU: càng lớn càng tốt
2. Thông lượng: càng lớn càng tốt
3. Thời gian xoay vòng: càng nhỏ càng tốt
4. Thời gian chờ đợi: càng nhỏ càng tốt
5. Thời gian đáp ứng: càng nhỏ càng tốt

Các Giải Thuật Định Thời

1. First-come, first-served (FCFS): đến trước được phục vụ trước.
2. Shortest-job-rirst (SJF): công việc ngắn nhất trước.
3. Priority: dựa trên độ ưu tiên.
4. Round-robin (RR): xoay vòng.
5. Multilevel scheduling: hàng đợi đa cấp.
6. Multilevel feedback-queue scheduling: hàng đợi phản hồi đa cấp.

First-Come, First Served (FCFS)

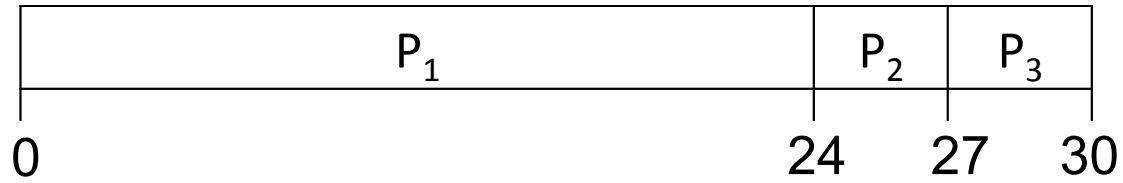
- Là giải thuật định thời đơn giản nhất, dựa trên nguyên tắc đến trước, được phục vụ trước.
- **Cài đặt**: phương pháp đơn giản nhất là dùng hàng đợi FIFO.
- **Ưu điểm**: cài đặt dễ dàng, đơn giản và dễ hiểu.
- **Nhược điểm**:
  - Thời gian chờ đợi trung bình thường là dài.
  - Không thích hợp cho hệ thống phân chia thời gian do đây là giải thuật định thời không trưng dụng (nonpreemptive).

FCFS – Ví Dụ 1

- Cho các tiến trình với thời gian thực thi và thứ tự xuất hiện như sau:

Process	TG sử dụng CPU	Thứ tự xuất hiện
P_1	24	1
P_2	3	2
P_3	3	3

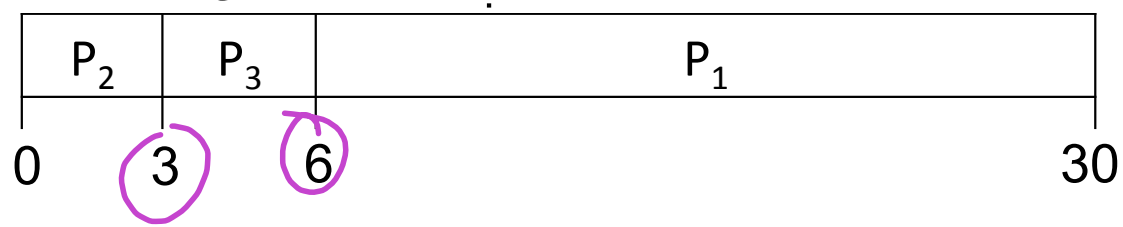
- Biểu đồ Grant cho lịch biểu:



- Thời gian chờ đợi: $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Thời gian chờ đợi trung bình: $(0 + 24 + 27)/3 = 17$

FCFS – Ví Dụ 2

- Giả sử các tiến trình trong ví dụ 1 xuất hiện theo thứ tự P_2, P_3, P_1; biểu đồ Grant của lịch biểu là:



- Thời gian chờ đợi: $P_1 = 6, P_2 = 0, P_3 = 1$
- Thời gian chờ đợi trung bình: $(6 + 0 + 3)/3 = 3$

  $\Rightarrow$ tốt hơn nhiều so với ví dụ 1 (17)
- Tình trạng thời gian chờ đợi dài do tiến trình ngắn nằm sau tiến trình dài được gọi là "hiệu ứng nối đuôi" (convoy effect).

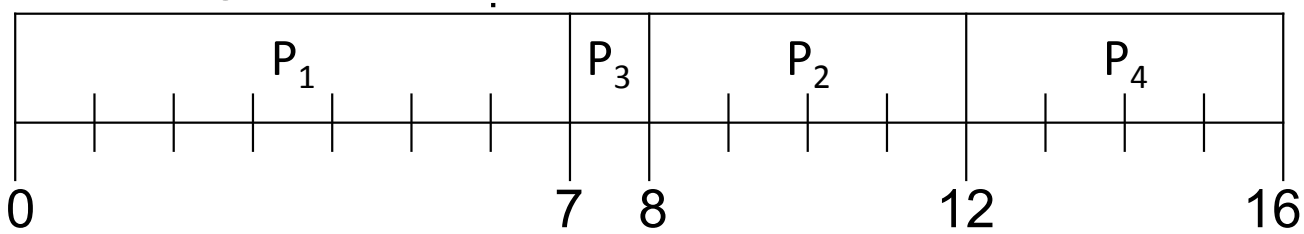
Shortest-Job-First (SJF)

- **Ý tưởng cơ bản**: phân phối CPU cho tiến trình nào có thời gian thực thi CPU (CPU burst) kế tiếp nhỏ nhất (*shortest-next-CPU-burst* alg.)
- Mỗi tiến trình sẽ được gán 1 độ dài thời gian của lần sử dụng CPU kế tiếp (dự đoán).
- Có 2 cách tiếp cận cho việc phân bổ CPU:
  - Không trưng dụng: tiến trình được giao CPU sẽ chiếm giữ CPU đến khi nó thực thi xong CPU burst.
  - Trưng dụng: nếu 1 tiến trình mới đến có CPU burst ngắn hơn thời gian thực thi còn lại của tiến trình đang thực thi, CPU sẽ được lấy lại để giao cho tiến trình mới (*shortest-remaining-time-first* algorithm, SRTF)
- SJF cho thời gian chờ đợi trung bình tối ưu (ngắn nhất).

SJF Không Trưng Dụng – Ví Dụ

Process	TG sử dụng CPU kế tiếp	Thời gian xuất hiện
P_1	7	0
P_2	4	2
P_3	1	4
P_4	4	5

- Biểu đồ Grant cho lịch biểu:

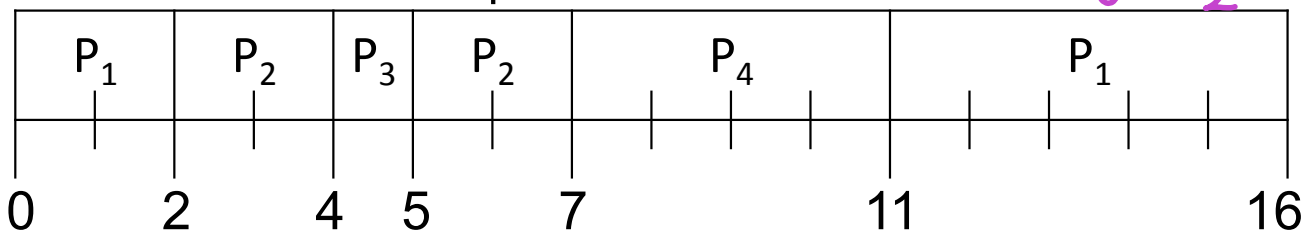


- Thời gian chờ đợi trung bình: $(0 + 6 + 3 + 7)/4 = 4$

SJF Trưng Dụng – Ví Dụ

Process	TG sử dụng CPU kế tiếp	Thời gian xuất hiện
P_1	7	0
P_2	4	2
P_3	1	4
P_4	4	5

- Biểu đồ Grant cho lịch biểu:



- Thời gian chờ đợi trung bình: $(9 + 1 + 0 + 2)/4 = 3$

Thời Gian Sử Dụng CPU Lần Kế Tiếp

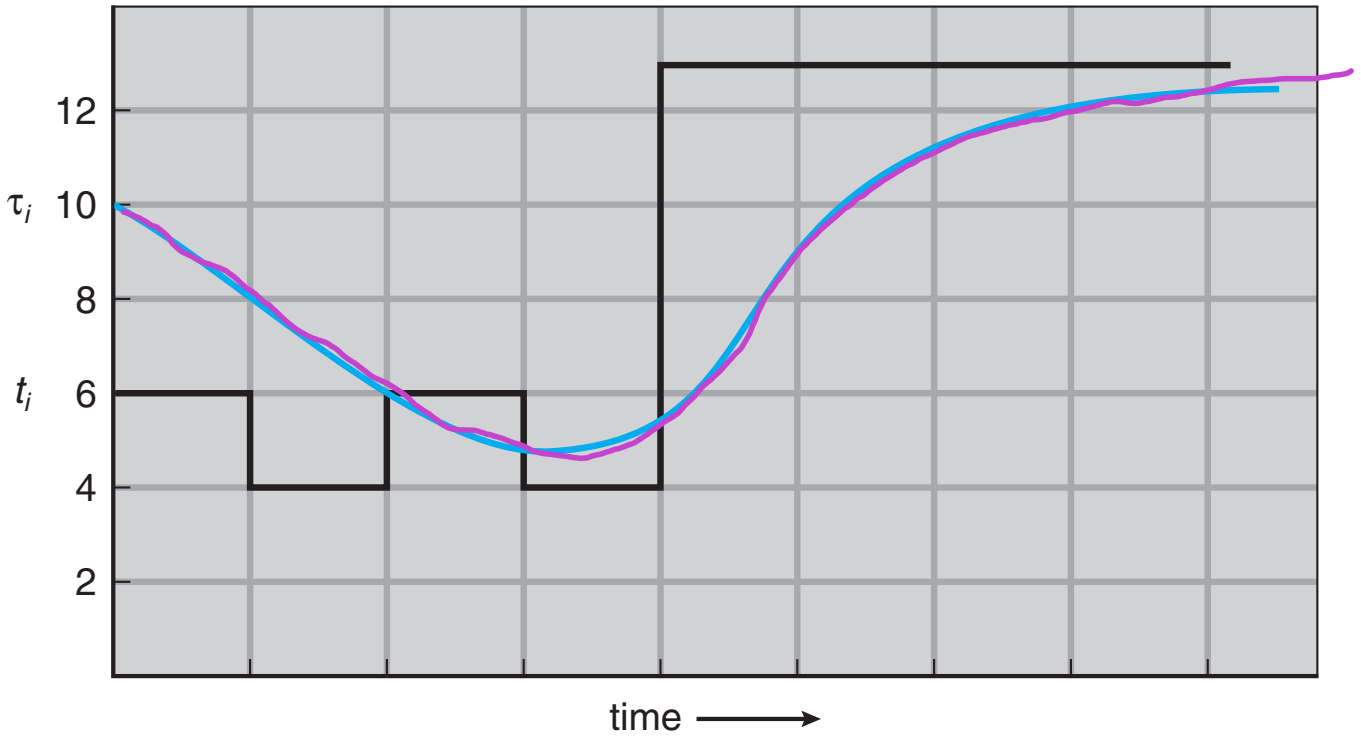
- Chỉ có thể ước lượng, dựa vào lịch sử của những lần sử dụng CPU trước đó.
- Thời gian sử dụng CPU kế tiếp (công thức trung bình mũ):

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$$

  - τ_{n+1}: ước lượng thời gian sử dụng CPU lần $n + 1$
  - t_n: thời gian sử dụng CPU thực tế lần thứ n
  - $\alpha \in [0, 1]$: hệ số trung bình mũ, dùng để điều chỉnh trọng số cho các giá trị lịch sử (thông thường được gán giá trị $1/2$)

Thời Gian Sử Dụng CPU Lần Kế Tiếp

- Ví dụ: ước lượng thời gian sử dụng CPU lần kế tiếp, với $\alpha = 1/2, \tau_0 = 10$



CPU burst (t_i)		6	4	6	4	13	13	13 ...
"guess" (τ_i)	10	8	6	6	5	9	11	12 ...

Tùy Biến Hệ Số Trung Bình Mũ

- $\alpha = 0 \Rightarrow \tau_{n+1} = \tau_n = \ldots = \tau_0$

  $\Rightarrow$ không tính đến lịch sử: tình trạng/sự thực thi "hiện thời" được coi như nhất thời, không có ý nghĩa.

- $\alpha = 1 \Rightarrow \tau_{n+1} = t_n$

  $\Rightarrow$ chỉ tính đến thời gian sử dụng CPU thực tế gần nhất.

- $\alpha = 1/2$: các giá trị lịch sử thực tế và dự đoán có trọng số tương đương.

- Nếu mở rộng công thức, ta có:

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\alpha t_{n-1} + \ldots + (1-\alpha)^j \alpha t_{n-j} + \ldots (1-\alpha)^{n+1}\tau_0$$

- Vì α và $(1-\alpha) \leq 1$, trọng số của giá trị lịch sử càng xa thì càng nhỏ.

Giải Thuật Định Thời Có Ưu Tiên (Priority)

- **Ý tưởng**:
  - Mỗi tiến trình sẽ được gán một chỉ số ưu tiên (priority number, int) .
  - CPU sẽ được cấp phát cho tiến trình có chỉ số ưu tiên cao nhất, thông thường là nhỏ nhất
- SJF là trường hợp đặc biệt của giải thuật này, trong đó thời gian thực thi CPU kế tiếp đóng vai trò là chỉ số ưu tiên.
- Có thể cài đặt theo phương pháp trưng dụng hay không trưng dụng.
- Có thể xảy ra tình trạng "chết đói" (starvation): các tiến trình độ ưu tiên thấp không bao giờ được thực thi.
  $\Rightarrow$ Giải pháp: dùng sự "lão hóa" (aging) – các tiến trình đang chờ đợi trong hệ thống sẽ được tăng dần độ ưu tiên theo thời gian chờ đợi.

Ví Dụ

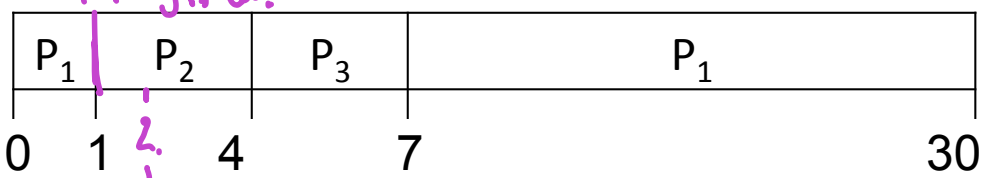
Process	Thời điểm xuất hiện	Độ ưu tiên	Thời gian xử lý
P_1	0	3	24
P_2	1	1	3
P_3	2	2	3

- **Không trưng dụng**: T/gian chờ đợi trung bình $= (0 + 23 + 25)/3 = 16$

  Biểu đồ Grant:

- **Trưng dụng**: T/gian chờ đợi trung bình $= (6 + 0 + 2)/3 = 2.7$

  Biểu đồ Grant:



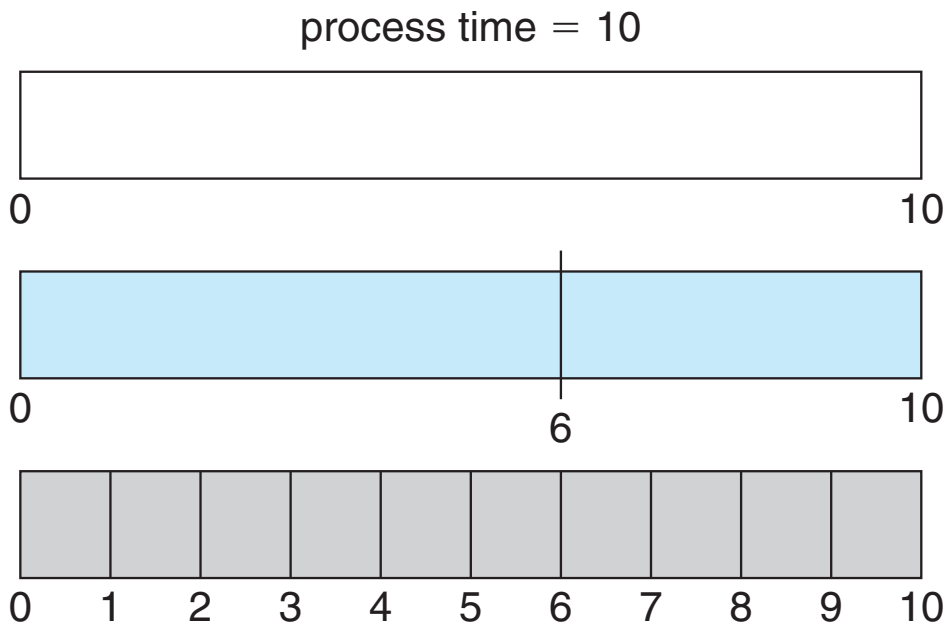
Giải Thuật Định Thời Luân Phiên

Dễ nhớ.

- Bộ điều phối cấp phát xoay vòng cho mỗi tiến trình trong hàng đợi sẵn sàng một đơn vị thời gian, gọi là định mức thời gian (*time quantum*, thường khoảng 10–100ms).
  - Sau khi sử dụng hết t/gian được cấp, CPU bị thu hồi để cấp cho tiến trình khác, tiến trình bị thu hồi CPU sẽ chuyển vào hàng đợi sẵn sàng.
  - Bộ đếm thời gian (timer) sẽ phát ra các ngắt sau mỗi định mức thời gian để xoay vòng cấp phát CPU.
- Nếu hàng đợi sẵn sàng có n tiến trình, định mức thời gian là q:
  - mỗi tiến trình sẽ nhận được $1/n$ tổng thời gian CPU, trong đó thời gian mỗi lần sử dụng tối đa là q
  - không có tiến trình nào chờ đợi quá lượng thời gian $(n-1) \times q$

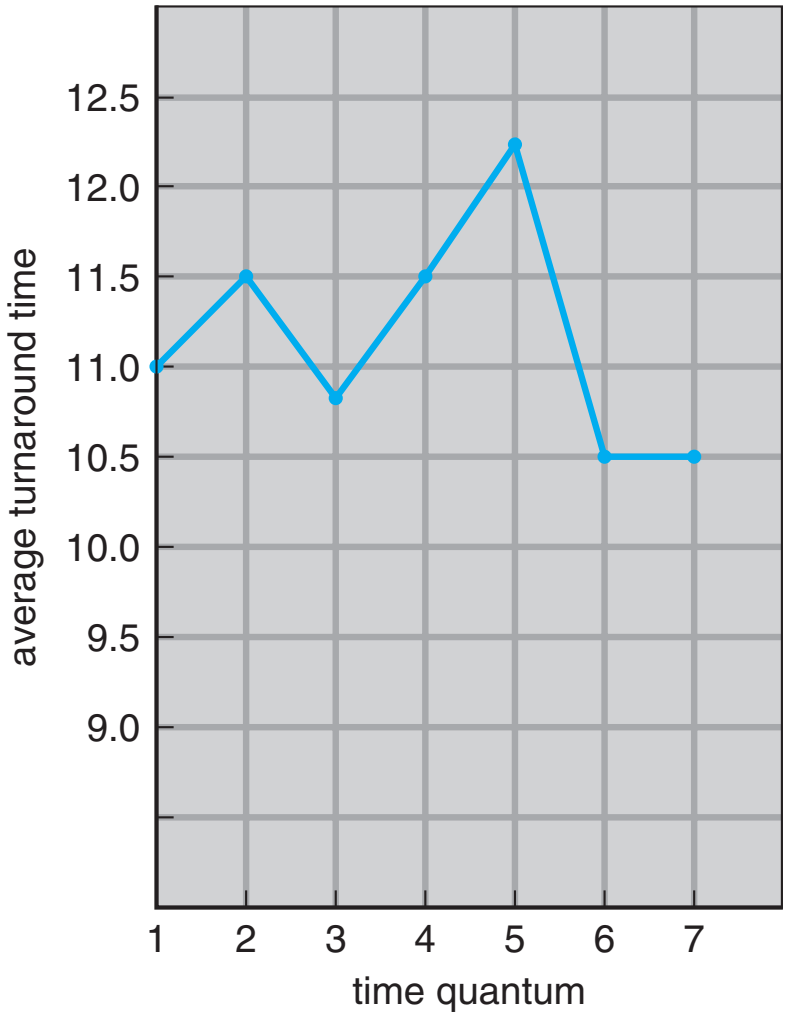
Các Tùy Biến & Hiệu Năng

- *q* lớn: RR trở thành giải thuật FCFS (FIFO)
- *q* nhỏ: *q* phải đủ lớn so với thời gian chuyển ngữ cảnh, nếu không, hao phí chuyển ngữ cảnh sẽ rất cao.



quantum	context switches
12	0
6	1
1	9

Các Tùy Biến & Hiệu Năng

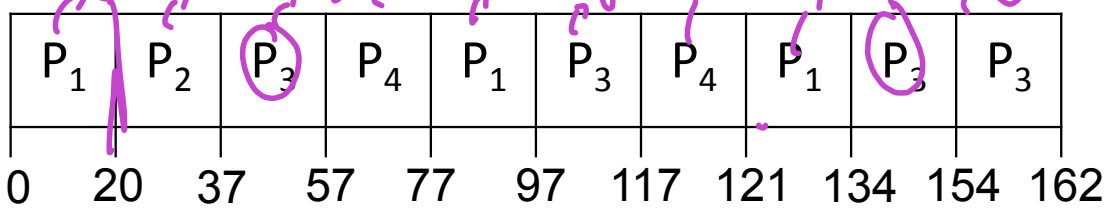


process	time
P_1	6
P_2	3
P_3	1
P_4	7

Ví Dụ

Process	Thời gian sử dụng CPU
P_1	53
P_2	17
P_3	68
P_4	24

- Với time quantum = 20
- Biểu đồ Grant:



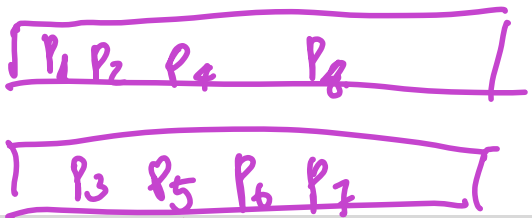
- Thông thường, RR có thời gian chờ đợi trung bình lớn hơn SJF, nhưng đảm bảo thời gian đáp ứng tốt hơn.

Giải Thuật Hàng Đợi Đa Cấp

- Chia hàng đợi s/sàng ra thành nhiều hàng đợi với độ ưu tiên khác nhau, ví dụ:
  - foreground: tương tác, cần ưu tiên cao hơn.
  - background: bó, cần ít ưu tiên hơn.
- Các tiến trình sẽ được phân phối vào các hàng đợi dựa trên các đặc tính như loại tiến trình (foreground/background), độ ưu tiên, ...
- Mỗi hàng đợi sẽ được áp dụng một giải thuật định thời riêng, tùy vào tính chất của hàng đợi; ví dụ:
  - foreground (tương tác): cần thời gian đáp ứng nhanh hơn $\Rightarrow$ RR
  - background (bó): có thể đáp ứng chậm hơn $\Rightarrow$ FCFS

Chiến Lược Định Thời Giữa Các Hàng Đợi

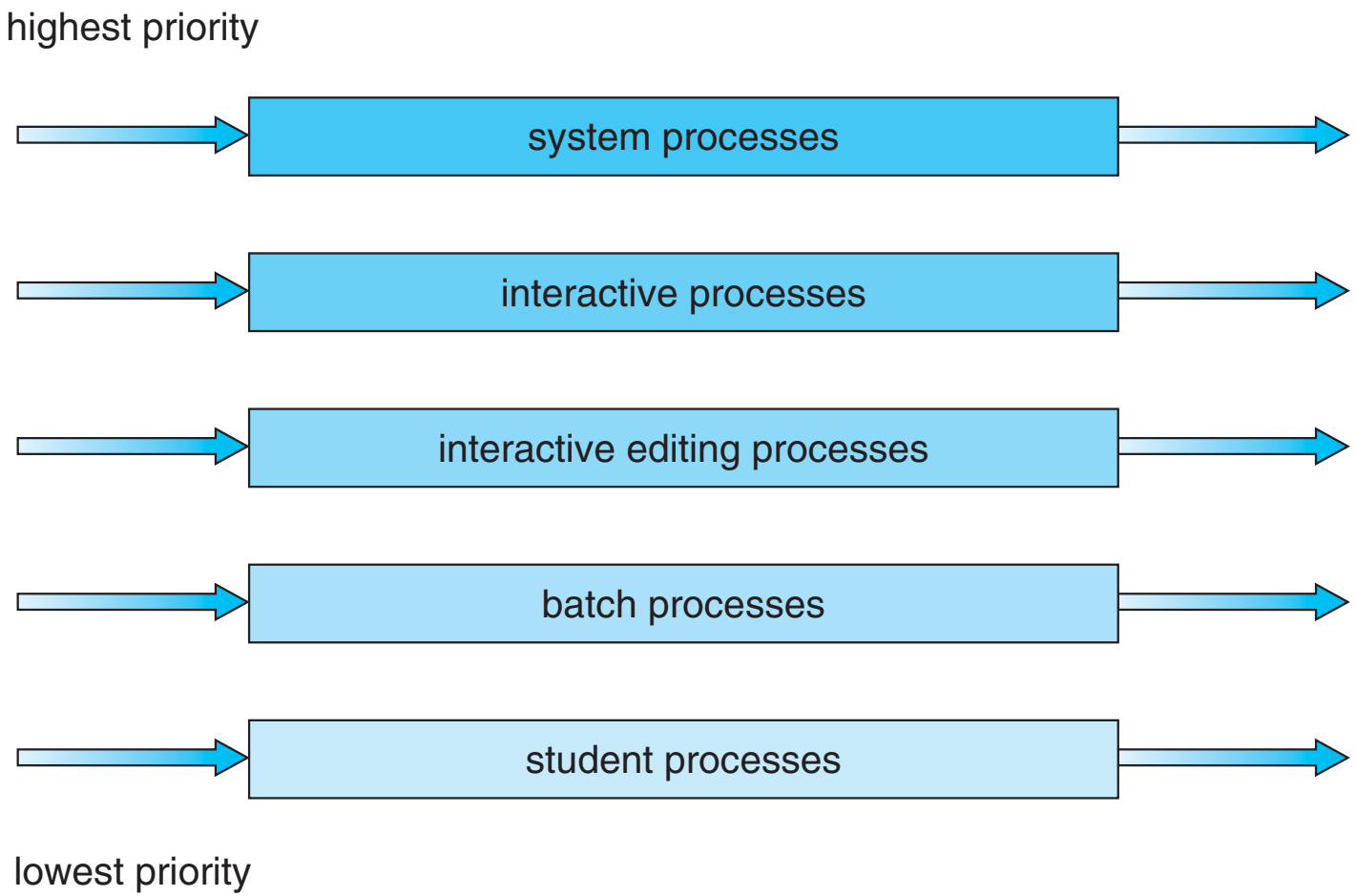
- Định thời với độ ưu tiên cố định: Xong foreground rồi background
  - Phục vụ tất cả các t/trình trong hàng đợi ưu tiên cao (vd: foreground) rồi mới đến hàng đợi có độ ưu tiên thấp hơn (vd: background).
  - Có khả năng dẫn đến tình trạng "chết đói" (starvation) CPU.
- Định thời với phân chia thời gian (time-slice):
  - Mỗi hàng đợi sẽ nhận được một khoảng thời gian nào đó của CPU để định thời cho các tiến trình nằm trong đó.
  - Ví dụ: 80% cho foreground với RR, và 20% cho background với FCFS.



80%

20%

Ví Dụ Hàng Đợi Đa Cấp



Giải Thuật Hàng Đợi Phản Hồi Đa Cấp

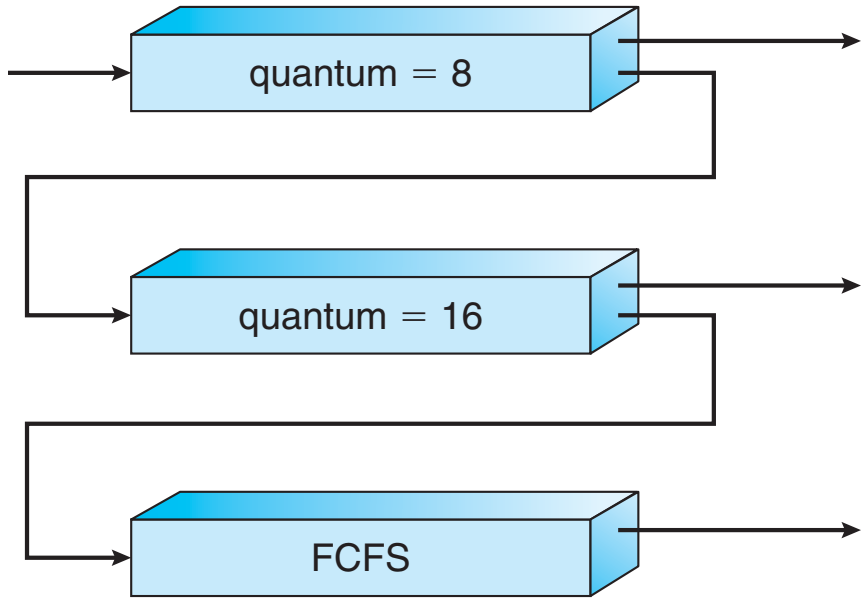
- Hàng đợi sẵn sàng cũng tổ chức giống như trong giải thuật Hàng đợi đa cấp.
- Một tiến trình có thể di chuyển giữa các hàng đợi khác nhau.
- Cơ chế định thời có thể được cài đặt theo cách:
  - Nếu tiến trình dùng quá nhiều thời gian CPU, nó sẽ được di chuyển vào hàng đợi có độ ưu tiên thấp hơn.
  - Nếu tiến trình đã chờ quá lâu trong 1 hàng đợi với độ ưu tiên thấp, nó sẽ được chuyển sang hàng đợi có độ ưu tiên cao hơn (cơ chế "sự lão hóa").

Tham Số Của Bộ Định Thời

- Bộ định thời đa cấp có phản hồi có thể được định nghĩa bằng những tham số sau:
  - Số lượng hàng đợi
  - Giải thuật định thời cho từng hàng đợi
  - Phương thức dùng để quyết định khi nào thì nâng cấp một tiến trình.
  - Phương thức dùng để quyết định khi nào thì hạ cấp một tiến trình
  - Phương thức dùng để quyết định là nên đặt tiến trình vào hàng đợi nào khi tiến trình này cần được phục vụ.

Ví Dụ Về Hàng Đợi Phản Hồi Đa Cấp

- **Các hàng đợi**:
  - Q_0: FCFS + quantum-time 8ms
  - Q_1: FCFS + quantum-time 16ms
  - Q_2: original FCFS
- **Định thời**:
  - Một tiến trình mới P sẽ được phân vào Q_0 với giải thuật định thời FCFS. Khi có được CPU, nó sẽ được sử dụng tối đa 8ms.



  - Nếu sau 8ms, P chưa hoàn thành thì CPU sẽ bị thu hồi để phân phối cho tiến trình khác và P sẽ được chuyển sang Q_1.
  - Tại Q_1, việc định thời diễn ra tương tự, với quantum-time là 16ms. Nếu P vẫn chưa hoàn thành thì nó sẽ được chuyển sang Q_2 với giải thuật FCFS.

Chính Sách, Cơ Chế & Định Thời

- Chính sách (policy) và cơ chế (mechanism) có liên hệ mật thiết đến sự lựa chọn giải thuật định thời.
  - Policy: Cần làm điều gì (what)?
  - Mechanism: Làm sao (how) để làm điều đó?
- **Ví dụ**:
  - Policy: tất cả các người dùng cần được công bằng. RR Công bằng nhất.

    $\Rightarrow$ Mechanism: sử dụng Robin-round.
  - Policy: Các tiến trình người dùng trả tiền có độ ưu tiên cao hơn các tiến trình người dùng miễn phí.

    $\Rightarrow$ Mechanism: sử dụng các giải thuật trưng dụng (preemptive). Trưng dụng

Định Thời Đa Xử Lý (Multi-Processor)

- Hệ thống đa xử lý cho phép cân bằng tải (load sharing) nhưng giải thuật định thời CPU sẽ phức tạp hơn.
- Các CPU trong hệ thống đa xử lý thường đồng nhất (homogeneous).
- Định thời đa xử lý đối xứng (symmetric multi-processing):
  - Mỗi CPU sẽ tự định thời.
  - Hàng đợi sẵn sàng có thể dùng chung hay riêng.
  - Cần quản lý sự cạnh tranh giữa các CPU khi chọn tiến trình và khi cập nhật cấu trúc dữ liệu hệ thống.
  - Là hệ thống hiện tại được dùng rộng rãi.

Định Thời Đa Xử Lý (Multi-Processor)

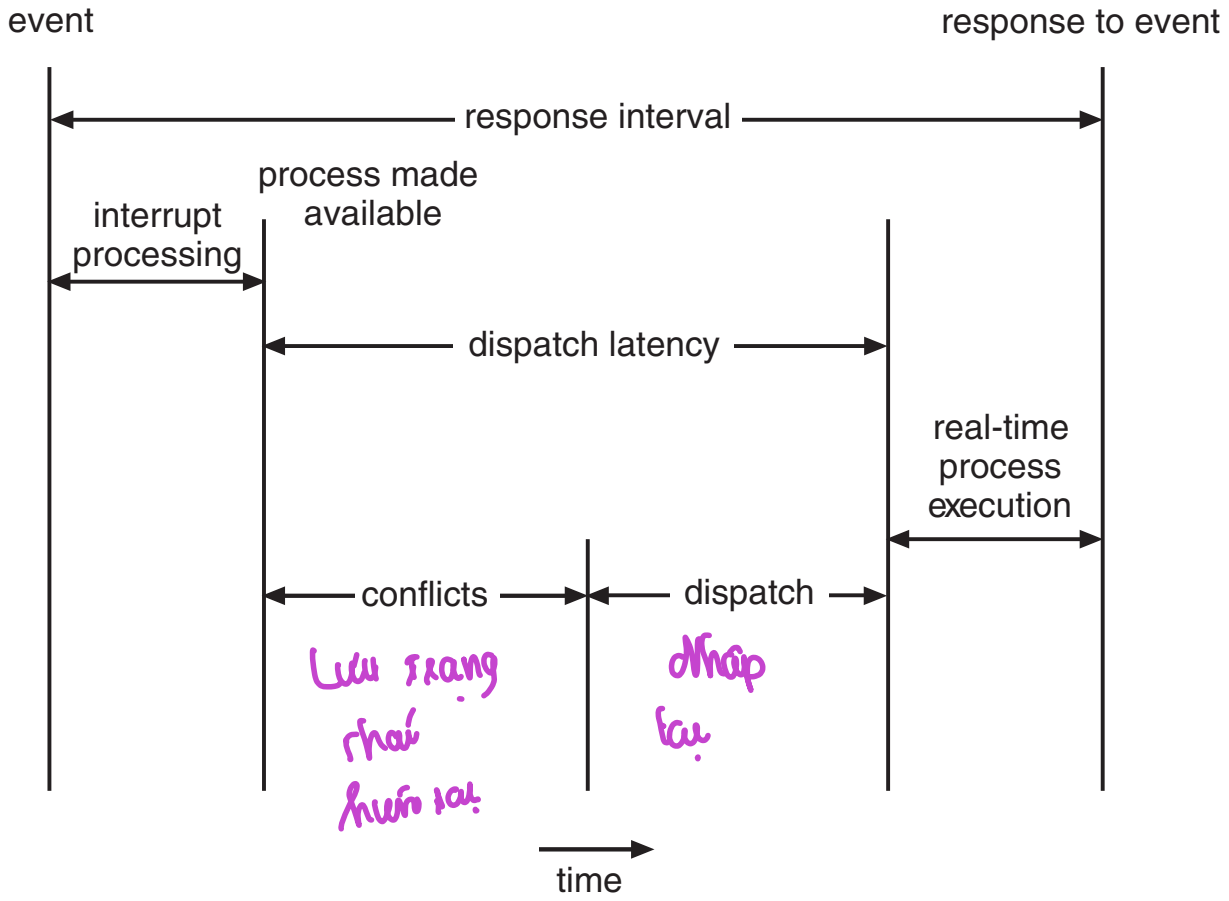
- Định thời đa xử lý bất đối xứng (asymmetric multi-processing):
  - Việc định thời được thực hiện bởi 1 CPU (master).
  - Master CPU sẽ quản lý cấu trúc dữ liệu dùng chung ⇒ giảm nhu cầu quản lý chia sẻ dữ liệu.
  - Có thể dẫn đến bottle-neck tại master CPU.
  - Ít được dùng hơn định thời đối xứng.
- Có thể dùng chiến lược "bám CPU" (processor affinity) – một tiến trình sẽ được thực thi trên cùng 1 CPU trong suốt sự thực thi – để tránh tổn phí di dời tiến trình sang CPU khác.
  - Soft-affinity và hard-affinity.

Định Thời Thời Gian Thực

- Hệ thống thời gian thực cứng:
  - Phải biết chính xác thời gian mà chức năng hệ điều hành cần cho mỗi thao tác.
  - Chỉ được thực hiện với các phần mềm có mục đích chuyên biệt trên nền phần cứng tận hiến.
- Hệ thống thời gian thực mềm:
  - Hệ thống phải có định thời trưng dụng.
  - Quá trình thời thực phải có độ ưu tiên cao nhất và không giảm, chấp nhận tình trạng không công bằng và đói tài nguyên.
  - Độ trễ điều phối phải nhỏ, bảo đảm đáp ứng nhanh cho tiến trình thời thực.

Độ Trễ Điều Phối (Dispatch Latency)

- Là khoảng thời gian cần thiết để bộ định thời chuẩn bị cho sự phục vụ 1 tiến trình.
- Giảm độ trễ điều phối:
  - đưa điểm trưng dụng vào lời gọi hệ thống dài
  - cho phép trưng dụng nhân



Đánh Giá Giải Thuật (đọc cho biết)

Phương pháp đánh giá và chọn giải thuật định thời thích hợp cho hệ thống.

1. Mô hình xác định (deterministic modeling)
2. Mô hình hàng đợi (queueing model)
3. Mô phỏng (simulation)
4. Cài đặt thực nghiệm (implementation)

Các chỉ tiêu đánh giá giải thuật
Tg chờ
Tg đáp ứng
Tg xoay vòng
Thông lượng:
Hiệu suất sử dụng CPU.

Mô Hình Xác Định

- Dựa vào tải công việc (workload) được xác định trước và tính toán hiệu năng của mỗi giải thuật (như thời gian chờ trung bình, . . .).
- **Ưu điểm**: Đơn giản, nhanh.
- **Khó khăn**: đòi hỏi tập hợp thông tin chính xác thông tin đầu vào $\Rightarrow$ khó thực hiện trong thực tế.

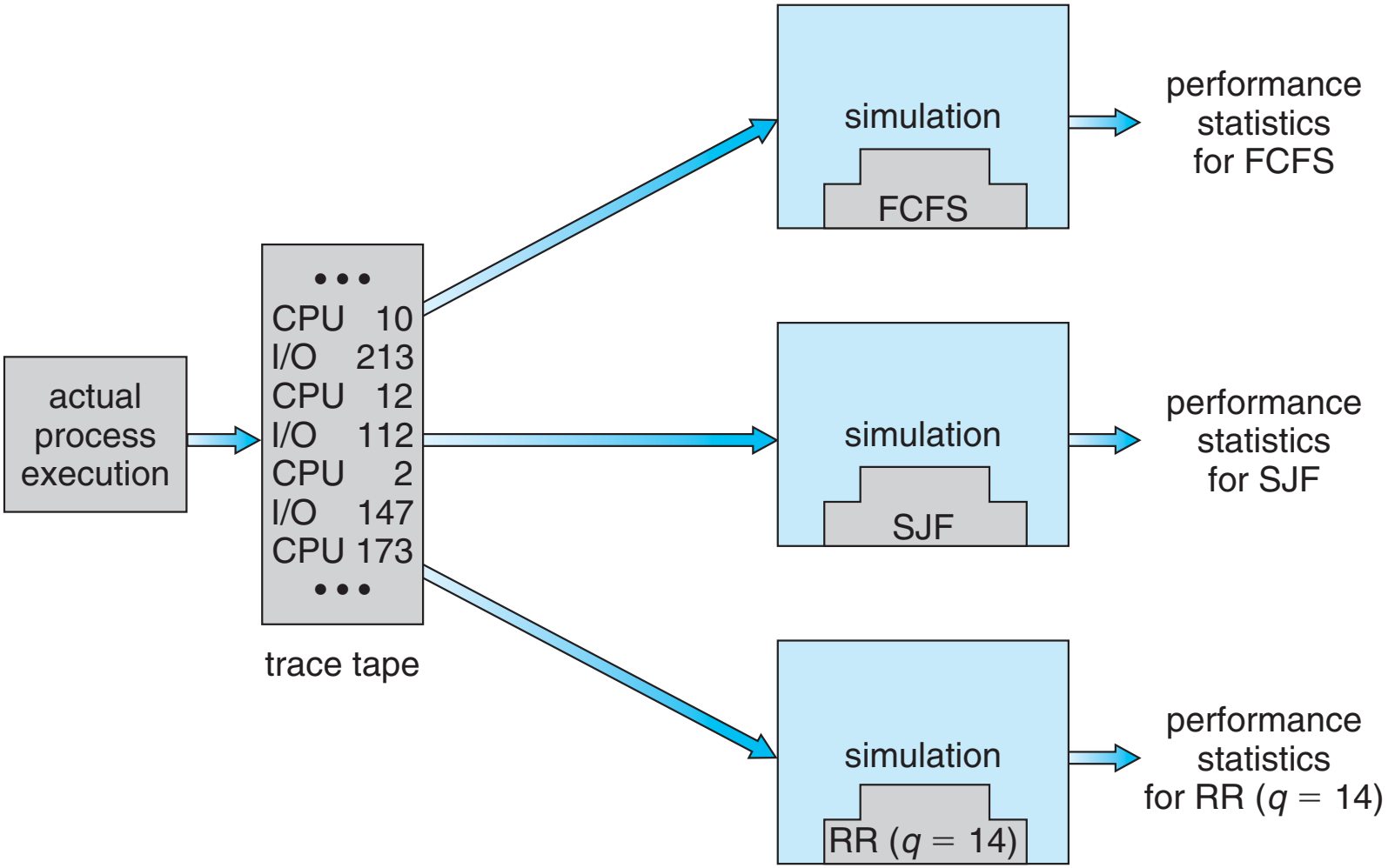
Mô Hình Hàng Đợi

- Dựa vào sự phân bổ thời gian thực thi CPU và I/O (CPU, I/O burst) và thời gian đến (arrival-time).
  - Thường phân phối theo 1 hàm nào đó (thường là hàm mũ – exponential distribution).
  - Dựa vào các phân phối trên, ta có thể tính hiệu năng của mỗi giải thuật (hiệu năng CPU, t/gian chờ trung bình, chiều dài hàng đợi trung bình)
- Chiều dài hàng đợi trung bình (công thức Little):

$$n = \lambda \times W$$

  - λ: tỉ lệ đến trung bình của các tiến trình mới (tiến trình/s)
  - W: thời gian chờ trung bình trong hàng đợi.

Mô Phỏng


actual process execution
• • •
CPU 10
I/O 213
CPU 12
I/O 112
CPU 2
I/O 147
CPU 173
• • •
trace tape
simulation
FCFS
simulation
SJF
simulation
RR (q = 14)
performance statistics for FCFS
performance statistics for SJF
performance statistics for RR (q = 14)

Cài Đặt Thực Nghiệm

- Cài đặt giải thuật vào hệ điều hành thật cùng với các phương tiện đo lường.
- Thực thi các tiến trình và đo lường hiệu năng của giải thuật trên hệ thống thật.
- **Ưu điểm**: Độ chính xác cao.
- **Khó khăn**:
  - Đòi hỏi chi phí cao
  - Người dùng có thể phản ứng với sự thay đổi liên tục của hệ thống.

Định Thời Trong Một Số HĐH

1. Windows 7
2. Linux
3. Solaris

Định Thời Trong Windows

- Đơn vị cấp phát CPU trong các HĐH hiện nay là luồng (thread) thay vì tiến trình.
- HĐH Windows dùng giải thuật định thời dựa trên độ ưu tiên (32 mức) và định mức thời gian (RR), sử dụng chiến lược trưng dụng.
- Mỗi độ ưu tiên sẽ có 1 hàng đợi riêng.
- Một luồng được chọn thực thi bởi bộ điều phối sẽ t/thi cho đến khi:
  - bị trưng dụng bởi luồng có mức ưu tiên cao hơn, hoặc
  - kết thúc (terminate), hoặc
  - hết định mức thời gian (time quantum), hoặc
  - thực hiện lời gọi I/O.

Định Thời Trong Windows

- Độ và nhóm ưu tiên của các luồng được chia như sau:

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

- Ngoài ra, một số cơ chế để nâng/hạ độ ưu tiên cho cá luồng cũng được sử dụng (tương tự ý tưởng của g/thuật hàng đợi phản hồi đa cấp)

Chọn Lựa Tiến Trình Trong Windows

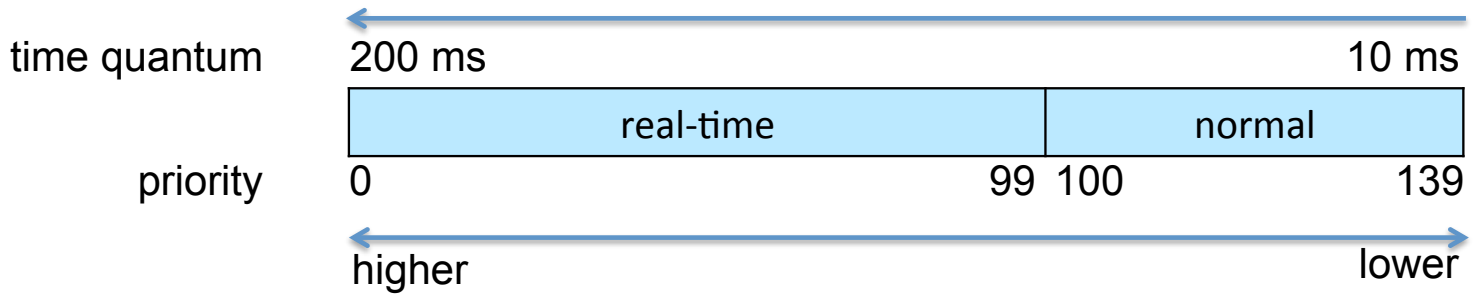
- Bộ định thời duyệt qua các hàng đợi theo độ ưu tiên từ cao đến thấp.
- Nếu không có tiến trình nào sẵn sàng, bộ định thời khởi động 1 tiến trình đặc biệt gọi là idle process.
- Độ ưu tiên của t/trình bị trưng dụng có thể bị thay đổi.
  - Nếu sử dụng hết time quantum: độ ưu tiên giảm, nhưng không dưới mức "normal".
  - Chuyển từ trạng thái chờ (waiting): độ ưu tiên tăng, mức độ tăng phụ thuộc vào t/trình đã chờ cái gì: đĩa – tăng nhiều, I/O – tăng ít hơn.

Định Thời Trong Linux

- Giải thuật định thời chính cho Linux từ kernel 2.6.23 là giải thuật Completely Fair Scheduler (CFS), dựa trên sự ưu tiên có thay đổi.
- Tiêu chí:
  - Giải thuật định thời phải nhanh, O(1) $\Rightarrow$ hỗ trợ tốt cho các hệ SMP.
  - Cho hiệu năng tốt đối với các interactive processes.
  - Công bằng (fairness)
  - Tối ưu cho trường hợp thông dụng nhất là hệ thống có 1 đến 2 t/trình, nhưng phải scale cho trường hợp có nhiều t/trình hoặc nhiều CPU.
  - Dùng k/niệm tác vụ (task) để chỉ tiến trình (process) và luồng (thread).

Định Thời Trong Linux

- Các t/vụ được phân chia thành 2 nhóm chính: default scheduling class và real-time scheduling class.
  - Default scheduling class: định thời bằng giải thuật CFS (trưng dụng), trong đó tỷ lệ sử dụng CPU được tính dựa trên nhiều tham số của t/vụ như **nice value** (nicer thread, lower priority), virtual run time,...
  - Real-time scheduling class: các t/vụ được gán độ ưu tiên cố định dựa vào nice value của các luồng.



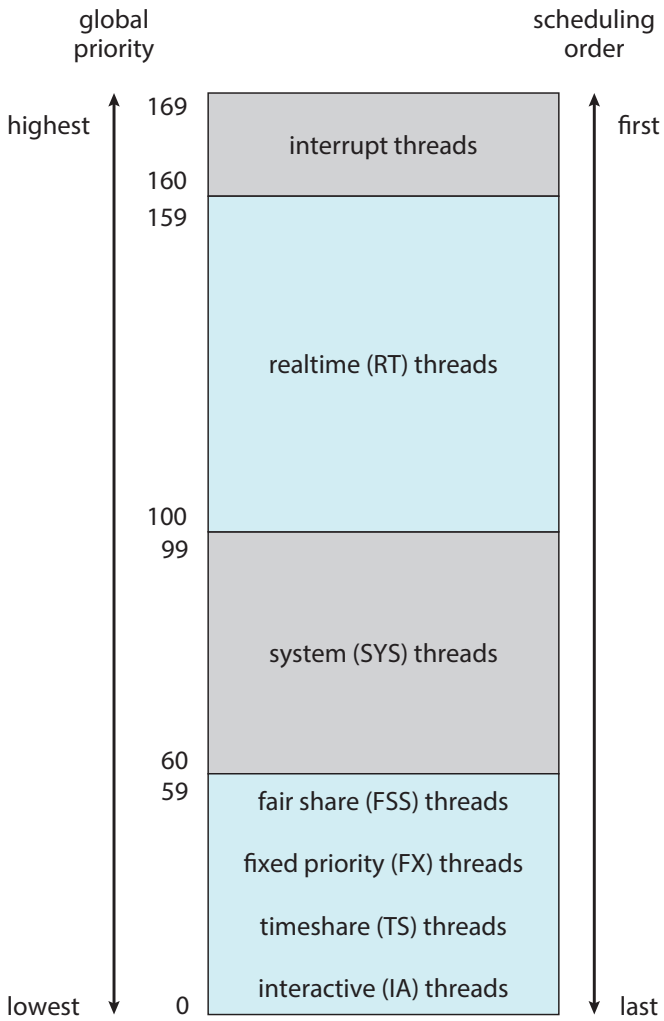
Chọn Lựa Tiến Trình Trong Linux

- Bộ định thời duy trì 2 hàng đợi:
  1. Active queue: chứa các t/vụ sẵn sàng thực thi và còn time slice.
  2. Expired queue: chứa các t/vụ sẵn sàng thực thi nhưng hết time slice.
- Các t/vụ trong cùng hàng đợi được sắp xếp dựa vào độ ưu tiên.
- Bộ định thời sẽ lựa t/vụ có độ ưu tiên cao nhất trong active queue.
  - Nếu một t/vụ bị trưng dụng CPU, nó sẽ trở lại active queue với độ ưu tiên không đổi.
  - Nếu một tác vụ hết time quantum, độ ưu tiên của nó sẽ được tính lại và được đưa vào expired queue.
- Nếu active queue rỗng: swap(expired queue, active queue).

Định Thời Trong Solaris

- Sử dụng định thời dựa trên độ ưu tiên.
- Bộ định thời sử dụng 6 hàng đợi: Time-sharing (TS), Interactive (IA), Real time (RT), System (SYS), Fair share (FSS), Fixed priority (FP).
- Mỗi hàng đợi có độ ưu tiên khác nhau và g/thuật định thời khác nhau.
- Độ ưu tiên của luồng ảnh hưởng đến time slice dành cho luồng và có thể bị thay đổi (hàng đợi phản hồi đa cấp).
- Tính tương tác của luồng (interactive/CPU-bound) cũng ảnh hưởng đến độ ưu tiên.
- Độ ưu tiên của các luồng có giá trị từ 0 (lowest) đến 160 (highest).

Định Thời Trong Solaris



* **Real-time class**:
  - độ ưu tiên cao nhất
  - đảm bảo t/gian đáp ứng
  - một ít kernel process được phân vào h/đợi này
* **System class**:
  - kernel process được phân vào h/đợi này
  - độ ưu tiên được gán cố định
* **Time-sharing**: độ ưu tiên có thể thay đổi.
* **Fixed-priority**: độ ưu tiên không thay đổi.
* **Fair-share**: chia sẻ CPU, không ưu tiên.

Real-Time Scheduling

- Dùng giải thuật FCFS hoặc RR.
- Sử dụng định thời trưng dụng: tiến trình ưu tiên cao có thể trưng dụng CPU của tiến trình ưu tiên thấp.
- Có rất ít tiến trình được vào hàng đợi này.

Time-Sharing Scheduling

- Sử dụng g/thuật hàng đợi phản hồi đa cấp, với độ ưu tiên thay đổi:
  - Mỗi hàng đợi có một độ ưu tiên khác nhau.
  - Time slice phụ thuộc vào độ ưu tiên (higher priority, smaller time slice)
- Solaris sử dụng một bảng điều phối để điều khiển việc định thời:
  - Nếu một luồng s/dụng hết time quantum (CPU-intensive process), độ ưu tiên của nó sẽ giảm $\Rightarrow$ tăng thông lượng (throughput).
  - Nếu một luồng trở lại từ t/thái waiting (e.g. chờ I/O), độ ưu tiên của nó sẽ tăng $\Rightarrow$ giảm t/gian đáp ứng (response time).
  - Chiến lược định thời này giúp các interactive process (như quản lý giao diện đồ họa) có độ ưu tiên cao (giảm t/gian đáp ứng).

Bảng Điều Phối (Solaris Dispatching Table)

priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59

- **time quantum**: t/gian cấp phát cho t/trình tương ứng với độ ưu tiên (tỷ lệ nghịch)
- **time quantum expired**: độ ưu tiên gán lại cho t/trình sau khi s/dụng hết time quantum
- **return from sleep**: độ ưu tiên gán cho t/trình trở về từ t/thái waiting

Tổng Kết

- Sự thực thi của một t/trình: chu kỳ CPU–I/O (CPU–I/O burst).
- Bộ định thời (scheduler): chọn 1 tiến trình từ hàng đợi sẵn sàng.
- Bộ điều điều phối (dispatcher): thực hiện switching.
- Các tiêu chí định thời (có thể xung đột nhau): hiệu suất s/dụng CPU, thông lượng, t/gian xoay vòng, t/gian chờ đợi, t/gian đáp ứng.
- Các giải thuật định thời: FCFS, SJF, Priority, Robin-round, Multilevel [feedback-queue] scheduling.
- Định thời trên hệ thống đa xử lý (multi-processor): đối xứng, bất đối xứng.
- Định thời thời gian thực, độ trễ điều phối (đọc thêm).
- Đánh giá g/thuật định thời: mô hình xác định (deterministic), mô hình hàng đợi (queuing), mô phỏng (simulation), cài đặt thực nghiệm (implementation).

Một số giải thuật tranh giá cao.