Chương 5
Định thời CPU

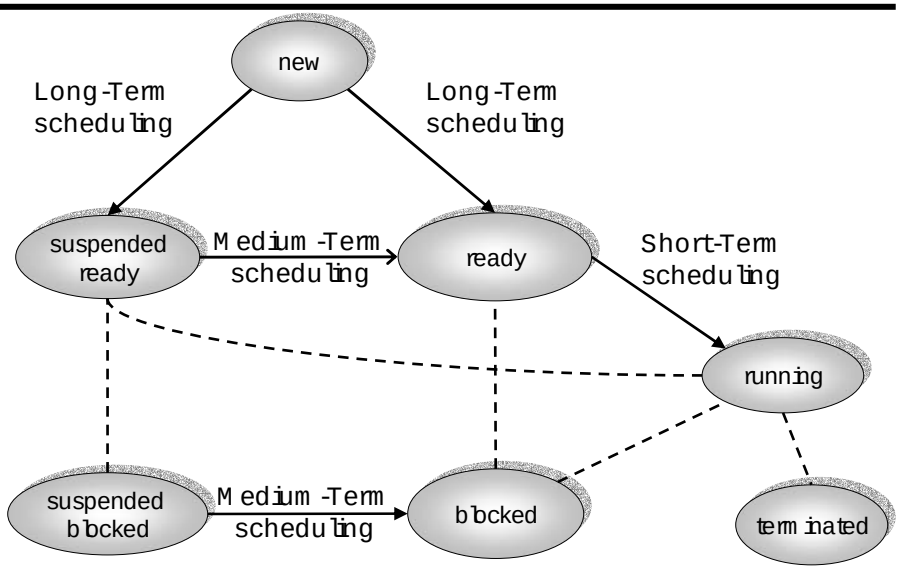
Nội dung

- Khái niệm cơ bản
- Các bộ định thời
  - long-term, mid-term, short-term
- Các tiêu chuẩn định thời CPU
- Các giải thuật định thời
  - First-Come, First-Served (FCFS)
  - Round-Robin (RR)
  - Shortest Job First (SJF)
  - Shortest Remaining Time First (SRTF)
  - Highest Response Ratio Next (HRRN)
  - Multilevel Queue
  - Multilevel Feedback Queue

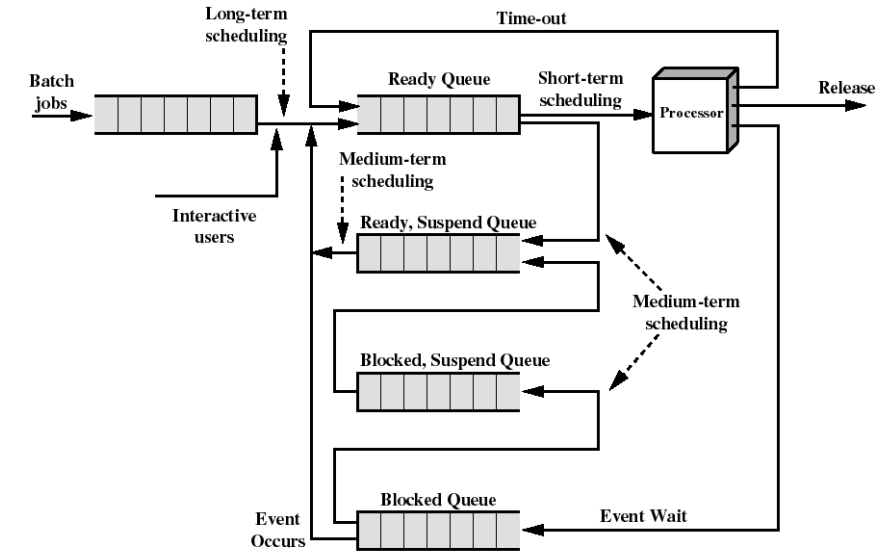
Khái niệm cơ bản

- Trong các hệ thống multi-tasking
  - Thực thi nhiều chương trình đồng thời làm tăng hiệu suất hệ thống.
  - Tại mỗi thời điểm, chỉ có một process được thực thi. Do đó, cần phải giải quyết vấn đề phân chia, lựa chọn process thực thi sao cho được hiệu quả nhất → chiến lược định thời CPU.
- Định thời CPU
  - Chọn một process (từ ready queue) thực thi.
  - Với một multithreaded kernel, việc định thời CPU là do OS chọn kernel thread được chiếm CPU.

Các bộ định thời



Các hàng đợi định thời



Các bộ định thời

- Long-Term Scheduling
  - Xác định chương trình nào được chấp nhận nạp vào hệ thống để thực thi
  - Điều khiển mức độ multiprogramming của hệ thống
  - Long term scheduler thường cố gắng duy trì xen lẫn CPU-bound và I/O-bound process
- Medium-Term Scheduling
  - Sự chuyển đổi dựa trên sự cần thiết để quản lý multiprogramming
  - Được thực hiện bởi phần quản lý bộ nhớ và được thảo luận ở phần quản lý bộ nhớ.

Short-Term Scheduling

- Xác định process nào trong ready queue sẽ được chiếm CPU để thực thi kế tiếp (do vậy còn được gọi là định thời CPU-CPU scheduling)
- Short term scheduler còn được gọi với tên khác là dispatcher
- Bộ định thời short-term được gọi mỗi khi có một trong các sự kiện/interrupt sau xảy ra:
  - clock interrupt
  - I/O interrupt
  - operating system call, trap
  - signal

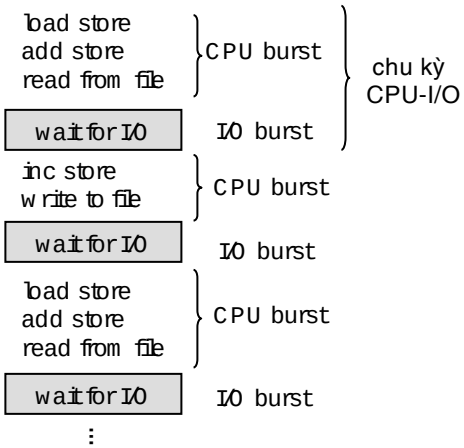
Các tiêu chuẩn định thời CPU

- User-oriented
  - *Response Time*: khoảng thời gian process nhận yêu cầu đến khi yêu cầu đầu tiên được đáp ứng (time-sharing, interactive system) → cực tiểu
  - *Turnaround Time*: khoảng thời gian từ lúc một process được nạp vào hệ thống đến khi process đó kết thúc → cực tiểu
  - *Waiting Time*: tổng thời gian một process đợi trong ready queue → cực tiểu
- System-oriented
  - processor utilization: định thời sao cho CPU càng bận càng tốt → cực đại
  - fairness: tất cả process phải được đối xử như nhau
  - throughput: số process hoàn tất công việc trong một đơn vị thời gian → cực đại.

Hai yếu tố của giải thuật định thời

- Hàm chọn lựa (selection function): dùng để chọn process nào trong ready queue được thực thi (thường dựa trên *độ ưu tiên*, *yêu cầu về tài nguyên, đặc điểm thực thi của process,...),* ví dụ
  - w = tổng thời gian đợi trong hệ thống
  - e = thời gian đã được phục vụ
  - s = tổng thời gian thực thi của process (bao gồm cả "e")
- Chế độ quyết định (decision mode): chọn thời điểm thực hiện hàm chọn lựa để định thời. Có hai chế độ
  - Nonpreemptive
    - Khi ở trạng thái running, process sẽ thực thi cho đến khi kết thúc hoặc bị blocked do yêu cầu I/O
  - Preemptive
    - Process đang thực thi (trạng thái running) có thể bị ngắt nửa chừng và chuyển về trạng thái Ready bởi hệ điều hành
    - Chi phí cao hơn non-preemptive nhưng đánh đổi lại bằng thời gian đáp ứng tốt hơn vì không có trường hợp một process độc chiếm CPU quá lâu.

Khảo sát giải thuật định thời

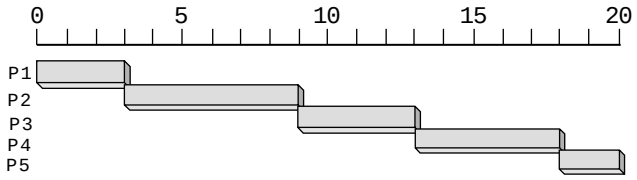


Process	Arrival Time	Service Time
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2

- ***Service time*** = thời gian process cần CPU trong một chu kỳ CPU-I/O
- Process có service time lớn là các CPU-bound process

First-Come First-Served (FCFS)

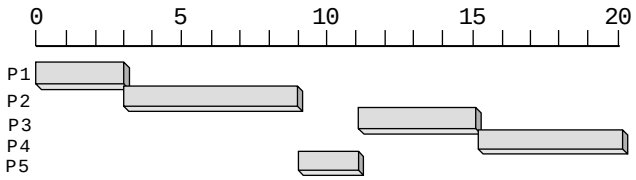
- Hàng đợi Ready là một hàng đợi kiểu FIFO
- Selection function: process nào đợi lâu nhất trong ready queue sẽ được chọn thực thi
- Decision mode: non-preemptive
  - Process sẽ thực thi đến khi kết thúc hoặc bị blocked do I/O



- Process nào không có thực hiện I/O sẽ được độc chiếm CPU.
- Các process I/O-bound (ít CPU-burst) không được ưu tiên bằng các process CPU-bound (nhiều CPU-burst)

Shortest Job First (SJF)

- Selection function: process nào có độ dài *CPU burst kế tiếp* nhỏ nhất sẽ được chọn thực thi.
- Decision mode: non-preemptive
- I/O-bound process sẽ được ưu tiên hơn so với CPU-bound process
- Yêu cầu: phải ước tính được CPU-burst của process



Ước tính độ dài của CPU Burst

- Độ dài của CPU Burst chỉ có thể ước tính xấp xỉ dùng phép tính trung bình hàm mũ (exponential average) dựa trên độ dài của CPU-burst trước.
  1. t_n = độ dài thực sự của CPU burst thứ *n*
  2. τ_{n+1}: giá trị ước đoán của CPU burst (*n + 1*)
  3. α, $0 \le \alpha \le 1$
  4. Độ dài ước tính CPU burst thứ n+1 là:

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

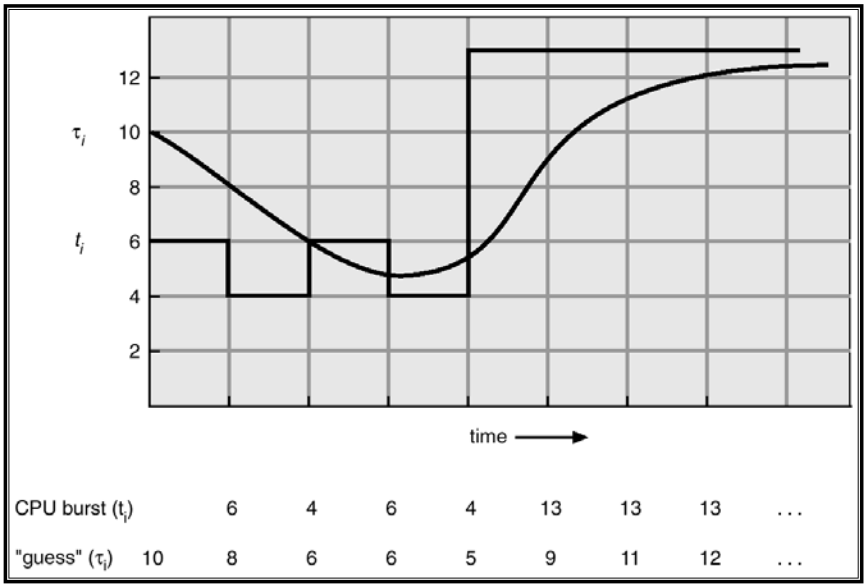
Ước tính độ dài của CPU Burst (t.t)

- Khai triển công thức:

$$\begin{aligned}\tau_{n+1} &= \alpha t_n + (1-\alpha)\tau_n \\ &= \alpha t_n + (1-\alpha)\alpha t_{n-1} + (1-\alpha)^2 \alpha \tau_{n-1} \\ &= \alpha \sum_{i=0}^{n-1} (1-\alpha)^i t_{n-i} + (1-\alpha)^n \tau_1\end{aligned}$$

- Giá trị τ_1 không ước tính mà thường gán = 0
- α=0 $\Rightarrow \tau_{n+1} = \tau_n \Rightarrow$ không quan tâm đến "quá khứ"
- α=1 $\Rightarrow \tau_{n+1} = t_n \Rightarrow$ chỉ quan tâm đến "ngày hôm qua"
- α=1/2

Ví dụ ước tính độ dài CPU Burst



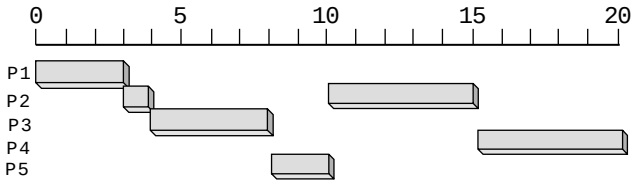
CPU burst (t_i)	6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Nhận xét về giải thuật SJF

- Có thể xảy ra tình trạng "chết đói" (starvation) đối với các process có CPU-burst lớn khi có nhiều process với CPU-burst nhỏ đến hệ thống.
- Cơ chế non-preemptive không phù hợp cho hệ thống time sharing (interractive)
- Giải thuật SJF ngầm định ra độ ưu tiên theo burst time
- Các CPU-bound process có độ ưu tiên thấp hơn so với I/O-bound process, nhưng khi một process không thực hiện I/O được thực thi thì nó độc chiếm CPU cho đến khi kết thúc

Shortest Remaining Time First

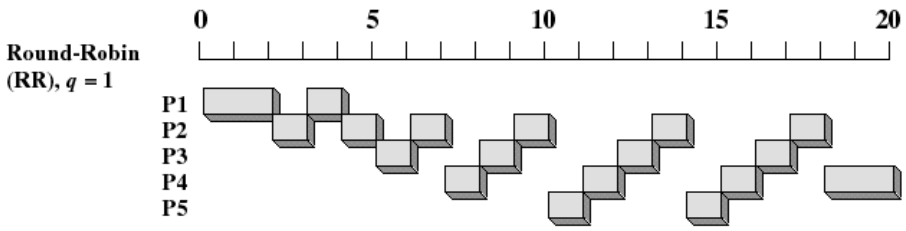
- Tương tự như SJF nhưng decision mode là preemptive
- Yêu cầu: phải ước tính được CPU-burst của process



Round Robin (RR)

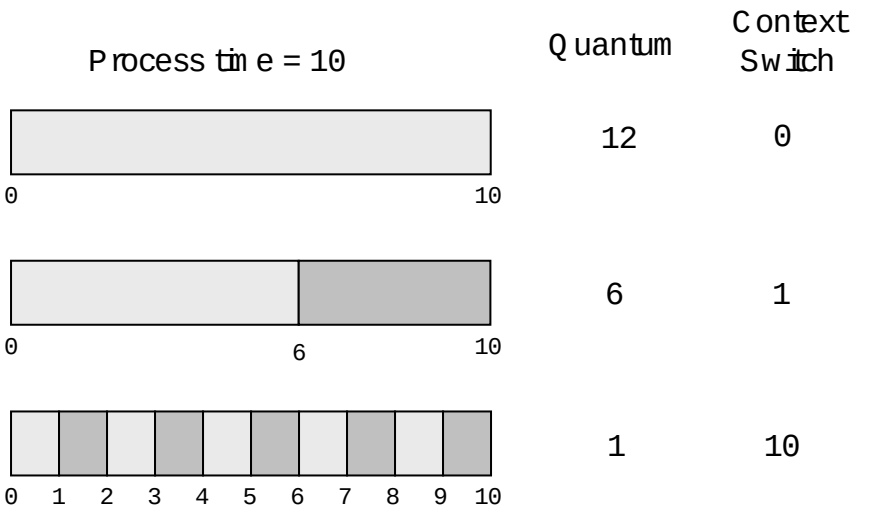
- Mỗi process nhận được một đơn vị nhỏ thời gian CPU (time slice, quantum time), thông thường từ 10-100 msec để thực thi. Sau khoảng thời gian đó, process bị đoạt quyền và trở về cuối hàng đợi ready.
- Nếu có n process trong hàng đợi ready và quantum time = q thì không có process nào phải chờ đợi quá (n-1)*q đơn vị thời gian.
- Hiệu suất
  - Nếu q lớn: RR $\Rightarrow$ FCFS
  - Nếu q nhỏ (q không được quá nhỏ bởi vì phải tốn chi phí chuyển ngữ cảnh)
- Thời gian chờ đợi trung bình của giải thuật RR thường khá lớn nhưng thời gian đáp ứng nhỏ

RR với Time Quantum = 1



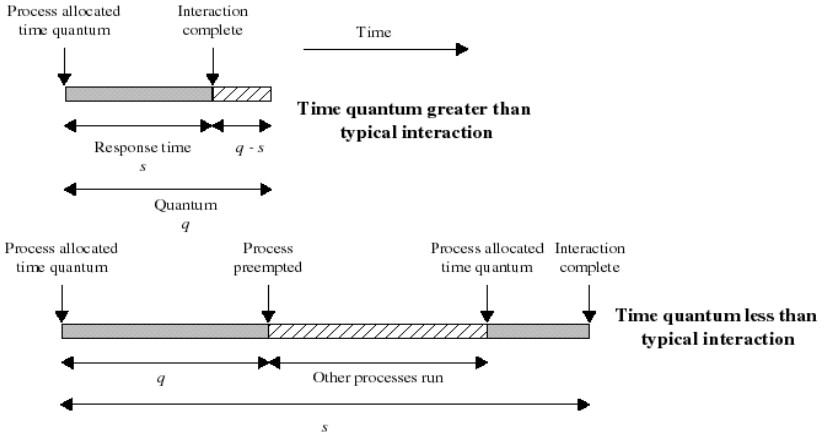
- Thời gian turn-around trung bình cao hơn so với SJF nhưng có thời gian đáp ứng trung bình tốt hơn.
- Ưu tiên CPU-bound process
  - ✓ I/O-bound process thường sử dụng rất ít thời gian của CPU, sau đó phải blocked đợi I/O
  - ✓ CPU-bound process tận dụng hết quantum time, sau đó quay về ready queue ⇒ được xếp trước các process bị blocked

Time Quantum và Context Switch

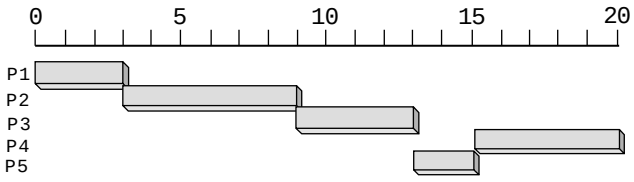


Quantum và response time

- Quantum time phải lớn hơn thời gian dùng để xử lý clock interrupt (timer) và thời gian dispatching
- Nên lớn hơn thời gian tương tác trung bình (typical interaction)



Highest Response Ratio Next



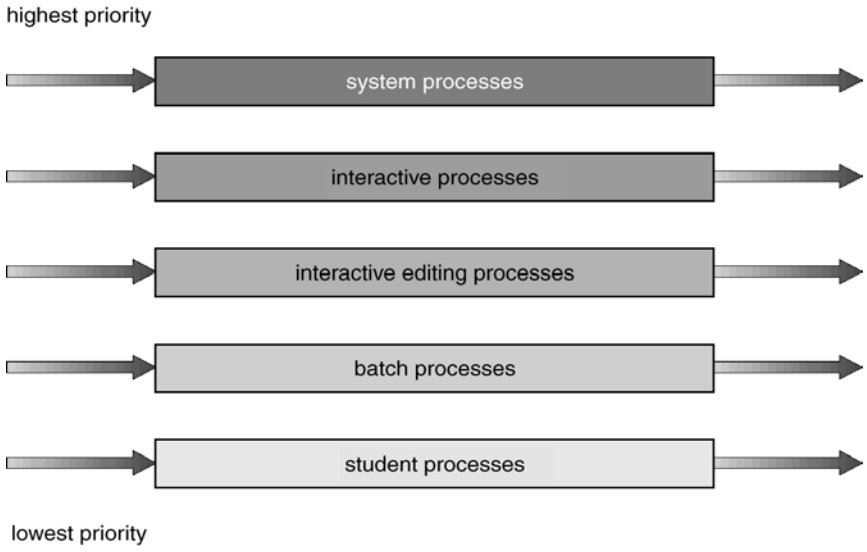
$$RR = \frac{\text{time spent waiting} + \text{expected service time}}{\text{expected service time}}$$

- Chọn process kế tiếp có giá trị RR (Response Ratio) lớn nhất.
- Các process ngắn được ưu tiên hơn (vì service time nhỏ)

Multilevel Queue Scheduling

- Hàng đợi Ready được chia thành nhiều hàng đợi riêng biệt theo một số tiêu chuẩn như
  - Đặc điểm và yêu cầu định thời của process
  - Foreground (interactive) và background process...
- Process được gán cố định vào một hàng đợi, mỗi hàng đợi sử dụng giải thuật định thời riêng
- Hệ điều hành cần phải định thời cho các hàng đợi.
  - *Fixed priority scheduling:* phục vụ từ hàng đợi có độ ưu tiên cao đến thấp. Vấn đề: có thể có starvation.
  - *Time slice* : mỗi hàng đợi được nhận một khoảng thời gian chiếm CPU và phân phối cho các process trong hàng đợi khoảng thời gian đó. Ví dụ: 80% cho hàng đợi foreground định thời bằng RR và 20% cho hàng đợi background định thời bằng giải thuật FCFS

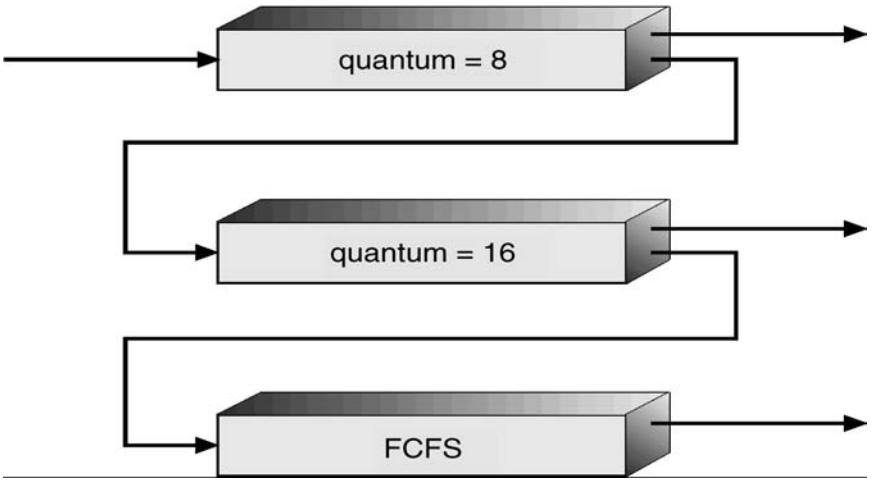
Multilevel Queue Scheduling



Multilevel Feedback Queue

- Vấn đề của multilevel queue
  - process không thể chuyển từ hàng đợi này sang hàng đợi khác ⇒ khắc phục bằng cơ chế feedback: cho phép process di chuyển một cách thích hợp giữa các hàng đợi khác nhau.
- Multilevel Feedback Queue
  - Phân loại processes dựa trên các đặc tính về CPU-burst
  - Sử dụng decision mode preemptive
  - Sau một khoảng thời gian nào đó, các I/O-bound process và interactive process sẽ ở các hàng đợi có độ ưu tiên cao hơn còn CPU-bound process sẽ ở các queue có độ ưu tiên thấp hơn .
  - Một process đã chờ quá lâu ở một hàng đợi có độ ưu tiên thấp có thể được chuyển đến hàng đợi có độ ưu tiên cao hơn (cơ chế niên hạn - aging)

Multilevel Feedback Queue (t.t)



Multilevel Feedback Queue (t.t)

- Định thời bằng Multilevel-feedback-queue đòi hỏi phải giải quyết các vấn đề sau
  - Số lượng hàng đợi bao nhiêu là thích hợp?
  - Dùng giải thuật định thời nào ở mỗi hàng đợi?
  - Làm sao để xác định thời điểm cần chuyển một process đến hàng đợi cao hơn hoặc thấp hơn?
  - Khi process yêu cầu được xử lý thì đưa vào hàng đợi nào là hợp lý nhất?

So sánh các giải thuật

- Giải thuật định thời nào là tốt nhất ?
- Câu trả lời phụ thuộc các yếu tố sau:
  - System workload
  - Sự hỗ trợ của phần cứng đối với dispatcher
  - Sự tương quan về trọng số của các tiêu chuẩn định thời như response time, hiệu suất CPU, throughput,...
  - Phương pháp định lượng so sánh