Bài tập CPU – Scheduling ( điều phối )

1. **What is the difference between preemptive scheduling and non-preemptive scheduling. What is the issue with the latter?**

**( Sự khác biệt giữa lập lịch phân chia (preemptive) và lập lịch không phân chia (non-preemptive) là gì? Vấn đề với lập lịch không phân chia là gì? )**

* **Lập lịch phân chia (preemptive scheduling):** Hệ điều hành có thể tạm dừng một tiến trình đang chạy và chuyển sang tiến trình khác bất kỳ lúc nào, thường dựa trên bộ hẹn giờ hoặc sự kiện ngắt.
* **Lập lịch không phân chia (non-preemptive scheduling):** Tiến trình đang chạy sẽ tiếp tục thực thi cho đến khi hoàn thành hoặc chuyển sang trạng thái chờ, hệ điều hành không can thiệp giữa chừng.

**Vấn đề với lập lịch không phân chia:** Nếu một tiến trình chiếm dụng CPU trong thời gian dài, các tiến trình khác có thể bị chậm trễ hoặc không được thực thi, dẫn đến hiệu suất hệ thống kém và thời gian phản hồi cao.

1. **Describe round robin scheduling. What is the parameter associated with the scheduler? What is the issue in chosing the parameter?**

**( Mô tả lập lịch vòng tròn (Round Robin). Tham số liên quan đến bộ lập lịch là gì? Vấn đề khi chọn tham số này? )**

* **Lập lịch vòng tròn (Round Robin):** 
  + Mỗi tiến trình được cấp phát một lượng thời gian CPU cố định (lượng tử thời gian).
  + Sau khi hết thời gian, tiến trình sẽ bị tạm dừng và chuyển sang tiến trình tiếp theo trong hàng đợi sẵn sàng.
* **Tham số liên quan:** Lượng tử thời gian (time quantum) xác định thời gian mỗi tiến trình được phép sử dụng CPU trước khi chuyển sang tiến trình khác.

**Vấn đề khi chọn tham số:**

Nếu lượng tử thời gian quá nhỏ, chi phí chuyển đổi ngữ cảnh sẽ tăng, làm giảm hiệu suất.

Nếu quá lớn, lập lịch vòng tròn có thể trở nên tương tự như lập lịch FCFS (First-Come, First-Served), dẫn đến hiệu suất kém hơn.

1. **The traditional UNIX scheduler is a priority-based round robin scheduleri also called a multi-level round robin schduler). How does the scheduler go about favouring I/0 bound jobs over long-running CPU-bound jobs?**

**( Bộ lập lịch truyền thống của UNIX là bộ lập lịch ưu tiên dựa trên vòng tròn (multi-level round robin). Bộ lập lịch này ưu tiên các công việc I/O-bound như thế nào so với các công việc CPU-bound dài hạn? )**

Bộ lập lịch UNIX sử dụng cơ chế ưu tiên để phân loại các tiến trình. Các tiến trình I/O-bound thường có mức độ ưu tiên cao hơn, cho phép chúng chiếm dụng CPU nhiều hơn, trong khi các tiến trình CPU-bound có mức độ ưu tiên thấp hơn và bị hạn chế thời gian sử dụng CPU. Điều này giúp hệ thống phản hồi nhanh hơn với các tiến trình I/O-bound.

1. **In a real-time system with a periodic task set, how are priorities assigned to each of the periodic tasks?**

**( Trong hệ thống thời gian thực với tập nhiệm vụ định kỳ, cách thức gán mức độ ưu tiên cho từng nhiệm vụ là gì ? )**

Gán mức độ ưu tiên: Trong hệ thống thời gian thực định kỳ, mức độ ưu tiên thường được gán dựa trên chu kỳ của nhiệm vụ : **nhiệm vụ có chu kỳ ngắn hơn sẽ có mức độ ưu tiên cao hơn.**

1. **What is an EDF scheduler? What is its advantage over a rate monotic scheduler?**

**( Bộ lập lịch EDF là gì? Ưu điểm của nó so với bộ lập lịch rate-monotonic là gì? )**

Bộ lập lịch EDF (Earliest Deadline First): Đây là bộ lập lịch động, trong đó **nhiệm vụ có hạn chót sớm nhất sẽ được ưu tiên thực thi**.

**Ưu điểm so với rate-monotonic**:

* EDF có thể đạt được hiệu suất tối ưu hơn trong việc đáp ứng các hạn chót
* Còn rate-monotonic có thể không đáp ứng được tất cả các nhiệm vụ trong một số trường hợp.

1. **Describe why the use of spinlocks might be more appropriate than blocking locks on multiprocessor when compared to a uniprocessor. Is there a trade-off between spinning and blocking on a multiprocessor? Discuss.**

**( Mô tả lý do tại sao việc sử dụng spinlocks có thể phù hợp hơn so với khóa chặn (blocking locks) trên hệ thống đa bộ xử lý so với hệ thống đơn bộ xử lý. Có sự đánh đổi giữa việc quay vòng (spinning) và chặn (blocking) không? )**

* **Spinlocks:** Là loại khóa mà tiến trình sẽ liên tục kiểm tra và chờ đợi cho đến khi khóa được giải phóng, phù hợp khi thời gian chờ đợi ngắn và chi phí chuyển đổi ngữ cảnh cao.
* **Khóa chặn (blocking locks):** Tiến trình sẽ bị chặn và chuyển sang trạng thái chờ cho đến khi khóa được giải phóng, phù hợp khi thời gian chờ đợi dài và chi phí chuyển đổi ngữ cảnh thấp.

**Sự đánh đổi:**

* Trên hệ thống đa bộ xử lý, spinlocks có thể hiệu quả hơn vì tránh được chi phí chuyển đổi ngữ cảnh.
* Tuy nhiên, nếu thời gian chờ đợi dài, việc quay vòng có thể gây lãng phí tài nguyên CPU.
* Ngược lại, trên hệ thống đơn bộ xử lý, khóa chặn có thể hiệu quả hơn do tránh được việc chiếm dụng CPU trong khi chờ đợi.

1. **Is single ready queue on a multiprocesssor a good idea? Why?**

**( Liệu việc sử dụng một hàng đợi sẵn sàng duy nhất trên hệ thống đa bộ xử lý có phải là ý tưởng tốt không? Tại sao? )**

Việc sử dụng một hàng đợi sẵn sàng duy nhất trên hệ thống đa bộ xử lý :

Có thể dẫn đến tình trạng tắc nghẽn khi nhiều bộ xử lý cùng truy cập vào hàng đợi này.

* Gây giảm hiệu suất hệ thống. Điều này xảy ra do cần phải đồng bộ hóa truy cập vào hàng đợi
* Điều này có thể làm tăng chi phí chuyển đổi ngữ cảnh và giảm hiệu suất tổng thể.

1. **What are the advantages and disadvantages of using a global scheduling queue over per-CPU queues ? Under which circumstances would you use the one of the other? What features of a system would ìnluence this decision ?**

**( Ưu điểm và nhược điểm của việc sử dụng hàng đợi lập lịch toàn cục so với hàng đợi lập lịch riêng cho mỗi CPU là gì? Trong trường hợp nào bạn sẽ sử dụng cái này thay vì cái kia? Những đặc điểm nào của hệ thống sẽ ảnh hưởng đến quyết định này? )**

* **Hàng đợi lập lịch toàn cục (Global Scheduling Queue):**
  + ***Ưu điểm:*** Đảm bảo phân phối công bằng tài nguyên CPU cho tất cả các tiến trình.
  + ***Nhược điểm:***
    - Có thể dẫn đến tắc nghẽn khi nhiều bộ xử lý cùng truy cập vào hàng đợi, gây giảm hiệu suất.
    - Khó đạt được sự liên kết bộ xử lý (processor affinity), tức là việc đảm bảo một tiến trình chạy trên cùng một bộ xử lý để tận dụng bộ nhớ đệm.
    - Cần đồng bộ hóa khi truy cập hàng đợi, điều này có thể làm tăng chi phí chuyển đổi ngữ cảnh và giảm hiệu suất.
* **Hàng đợi lập lịch riêng cho mỗi CPU (Per-CPU Queues):**
  + ***Ưu điểm:***
    - Giảm tắc nghẽn do mỗi bộ xử lý có hàng đợi riêng.
    - Dễ dàng đạt được sự liên kết bộ xử lý, cải thiện hiệu suất.
  + ***Nhược điểm:***
    - Có thể dẫn đến mất cân bằng tải giữa các bộ xử lý nếu không được quản lý đúng cách.
    - Khó đảm bảo phân phối công bằng tài nguyên CPU cho tất cả các tiến trình.

**Quyết định sử dụng hàng đợi nào phụ thuộc vào các yếu tố như:**

* **Tính chất của ứng dụng:**
  + Nếu ứng dụng yêu cầu sự liên kết bộ xử lý cao và có thể chấp nhận mất cân bằng tải, hàng đợi riêng cho mỗi CPU có thể phù hợp.
  + Ngược lại, nếu công bằng và cân bằng tải là quan trọng, hàng đợi toàn cục có thể được ưu tiên.
* **Khả năng mở rộng của hệ thống:** Hệ thống với nhiều bộ xử lý có thể gặp vấn đề với hàng đợi toàn cục do tắc nghẽn, trong khi hàng đợi riêng cho mỗi CPU có thể mở rộng tốt hơn.
* **Chi phí đồng bộ hóa:** Hàng đợi toàn cục yêu cầu đồng bộ hóa khi truy cập, điều này có thể làm tăng chi phí chuyển đổi ngữ cảnh và giảm hiệu suất.

1. **When does spinning on a lock ( busy waiting, as oppossed to bloing on the lock, and being woken up when it’s free ) make sense in a multiprocessor envinroment?**

**( Khi nào việc quay vòng (spinning) trên một khóa (busy waiting), thay vì chặn (blocking) trên khóa và được đánh thức khi nó sẵn sàng, là hợp lý trong môi trường đa bộ xử lý? )**

Việc quay vòng trên một khóa (busy waiting) có thể hợp lý trong môi trường đa bộ xử lý khi:

* **Thời gian chờ đợi ngắn:** Nếu thời gian chờ đợi khóa dự kiến rất ngắn, việc quay vòng có thể hiệu quả hơn so với việc chặn và đánh thức.
* **Tài nguyên hệ thống hạn chế:** Việc chặn và đánh thức có thể tiêu tốn tài nguyên hệ thống, trong khi quay vòng có thể giảm thiểu việc sử dụng tài nguyên này.

Tuy nhiên, việc quay vòng có thể dẫn đến việc chiếm dụng CPU không cần thiết, gây giảm hiệu suất tổng thể.

Do đó, cần cân nhắc kỹ lưỡng giữa việc quay vòng và chặn dựa trên đặc điểm cụ thể của hệ thống và ứng dụng.

**10. Why is preemption an issue with spinlocks?**

**( Tại sao việc ngắt quãng (preemption) lại là vấn đề với spinlocks? )**

Spinlocks là :

1. Cơ chế đồng bộ hóa ở mức thấp
2. Nơi tiến trình chờ đợi khóa bằng cách liên tục kiểm tra trạng thái của nó.
3. Nếu một tiến trình đang giữ spinlock bị ngắt quãng, tiến trình đó có thể không tiếp tục thực thi, dẫn đến tình trạng chờ đợi vô hạn cho tiến trình đang chờ khóa.
4. Điều này gây ra vấn đề nghiêm trọng trong việc đảm bảo tính nhất quán và hiệu suất của hệ thống

**11. What do the terms 1/O bound and CPU bound mean when used to describe a process (or thread)?**

**( Thuật ngữ "I/O bound" và "CPU bound" có nghĩa là gì khi mô tả một tiến trình (hoặc luồng)? )**

* **I/O bound:** 
  + Tiến trình chủ yếu chờ đợi các hoạt động nhập/xuất (I/O) như đọc/ghi đĩa hoặc truyền thông mạng.
  + Những tiến trình này thường dành nhiều thời gian ở trạng thái chờ và ít sử dụng CPU.
* **CPU bound:** 
  + Tiến trình chủ yếu sử dụng CPU để thực hiện các phép toán tính toán nặng, ít phụ thuộc vào các hoạt động I/O.
  + Những tiến trình này thường chiếm dụng CPU trong thời gian dài và ít chờ đợi I/O.

**12. What is the difference between cooperative and pre-emptive multitasking?**

**( Sự khác biệt giữa đa nhiệm hợp tác (cooperative multitasking) và đa nhiệm ngắt quãng (preemptive multitasking) là gì? )**

* **Đa nhiệm hợp tác:** 
  + Tiến trình tự nguyện nhường CPU cho tiến trình khác khi hoàn thành công việc hoặc khi ở trạng thái chờ.
  + Nếu một tiến trình không nhường CPU, nó có thể chiếm dụng tài nguyên hệ thống, gây ảnh hưởng đến các tiến trình khác.
* **Đa nhiệm ngắt quãng:** 
  + Hệ điều hành có thể tạm dừng một tiến trình đang chạy và chuyển sang tiến trình khác bất kỳ lúc nào
  + Đảm bảo công bằng và hiệu quả trong việc phân phối tài nguyên hệ thống.

**13. Consider the multilevel feedback queue scheduling algorithm used in traditional Unix systems. It is designed to favour IO bound over CPU bound processes. How is this achieved? How does it make sure that low priority, CPU bound background jobs do not suffer starvation?**

**( Thuật toán lập lịch hàng đợi phản hồi đa cấp (multilevel feedback queue) trong hệ thống Unix truyền thống được thiết kế để ưu tiên các tiến trình I/O bound hơn các tiến trình CPU bound. Điều này được thực hiện như thế nào? Làm thế nào để đảm bảo rằng các tiến trình nền CPU bound với độ ưu tiên thấp không bị đói (starvation)? )**

* Thuật toán này sử dụng nhiều hàng đợi với mức độ ưu tiên khác nhau.
* Các tiến trình I/O bound, thường có thời gian chờ đợi I/O dài, được chuyển đến hàng đợi ưu tiên cao hơn, cho phép chúng chiếm dụng CPU nhiều hơn.
* Ngược lại, các tiến trình CPU bound được chuyển đến hàng đợi ưu tiên thấp hơn.
* Để tránh tình trạng đói, hệ thống có thể nâng độ ưu tiên của các tiến trình CPU bound sau một thời gian chờ đợi, đảm bảo chúng có cơ hội thực thi.

**14. Why would a hypothetical OS always schedule a thread in the same address space over a thread in a different address space? Is this a good idea?**

**( Tại sao một hệ điều hành giả thuyết lại luôn lập lịch một luồng trong cùng không gian địa chỉ thay vì một luồng trong không gian địa chỉ khác? Đây có phải là một ý tưởng tốt không? )**

* Việc lập lịch luồng trong cùng không gian địa chỉ giúp :
  + Giảm chi phí chuyển đổi ngữ cảnh và truy cập bộ nhớ, vì các luồng chia sẻ cùng không gian bộ nhớ.
  + Tuy nhiên, điều này có thể dẫn đến vấn đề về bảo mật và ổn định hệ thống, vì một luồng có thể ảnh hưởng đến bộ nhớ của luồng khác.
  + Do đó, việc lập lịch luồng trong cùng không gian địa chỉ có thể không phải là ý tưởng tốt trong mọi trường hợp.

**15. Why would a round robin scheduler NOT use a very short time slice to provide good responsive application behaviour?**

**( Tại sao bộ lập lịch vòng tròn (round robin) không sử dụng lượng tử thời gian rất ngắn để cung cấp hành vi ứng dụng phản hồi tốt? )**

* Việc sử dụng lượng tử thời gian quá ngắn có thể dẫn đến chi phí chuyển đổi ngữ cảnh cao, làm giảm hiệu suất hệ thống và tăng độ trễ.
* Điều này có thể gây ảnh hưởng xấu đến trải nghiệm người dùng, đặc biệt đối với các ứng dụng yêu cầu phản hồi nhanh.

**16. What is the average waiting time for the following processes with non preemptive SJF (Shortes Job First).**

**( Thời gian chờ đợi trung bình cho các tiến trình sau với lập lịch SJF không ngắt quãng (non-preemptive Shortest Job First) là bao nhiêu? )**

Để tính toán thời gian chờ đợi trung bình, bạn cần biết thời gian thực thi của từng tiến trình và thứ tự thực thi của chúng.

Công thức tính thời gian chờ đợi trung bình là :

**Thời gian chờ trung bình = tổng thời gian chờ của mỗi tiến trình / số lượng tiến trình**