**BÀI TẬP LÝ THUYẾT**

**MÔN HỌC: HỆ PHÂN TÁN**

**CHƯƠNG 2: Tiến trình và Luồng trong HPT (P1)**

**Họ và tên: Lê Thị Yên**

**MSSV: 20183861**

**Câu hỏi 1:** Có cần thiết phải giới hạn số lượng các luồng trong một tiến trình server?

* Khi 1 tiến trình chạy 1 chương trình nó sẽ sinh ra 1 hay nhiều luồng.
* Sau khi được tiến trình sinh ra mỗi luồng có đoạn mã thực hiện riêng, độc lập với các luồng khác. -> Nếu càng tạo ra nhiều luồng thì càng tốn bộ nhớ chính
* Đồng thời, các luồng không cần phải hoạt động hoàn toàn độc lập và trong suốt với nhau. Vì vậy mỗi luồng chỉ cần lưu trữ thông tin ít nhất để có thể cho phép các luồng chia sẻ CPU
* Cần giới hạn số lượng các luồng trong 1 tiến trình server

**Câu hỏi 2:** Có nên chỉ gắn một luồng đơn duy nhất với một tiến trình nhẹ?

Không cần thiết vì:

* Mỗi tiến trình nhẹ giữ 1 bảng luồng để tránh việc cùng dùng 1 luồng. Việc tránh truy cập cùng lúc vào dữ liệu chia sẻ được đảm đương hoàn toàn bởi mức người dùng.
* Nếu 1 luồng bị dừng nó sẽ thực hiện lời gọi lập lịch. Khi 1 luồng khả chạy khác được tìm thấy nó sẽ chuyển ngữ cảnh sang cho luồng mới đó và tiến trình nhẹ đang chạy luồng đó hoàn toàn không biết việc chuyển ngữ cảnh

**Câu hỏi 3:** Có nên chỉ có một tiến trình nhẹ đơn gắn với 1 tiến trình?

Diagram

Description automatically generated

Viêc sử dụng tiến trình nhẹ là sự kết hợp của việc chạy đa luồng ở chế độ người dùng và nhân.

* Với cơ chế này khi chạy ngữ cảnh ở 1 tiến trình đơn và mỗi tiến trình đơn có nhiều tiến trình nhẹ.
* Khi 1 luồng thực hiện 1 lời gọi hệ thống dừng, việc thực thi được chuyển xuống cho kiểu nhân nhưng vẫn ở trong ngữ cảnh của tiến trình nhẹ hiện tại. HĐH sẽ chuyển ngữ cảnh sang cho tiến trình nhẹ khác, dẫn tới việc chuyển ngữ cảnh chuyển ngược lại về cho kiểu người dùng. => lời gọi hệ thống dùng không làm dừng cả hệ thống lại
  + Như vậy, khi 1 tiến trình có đủ các tiến trình nhẹ, 1 lời gọi hệ thống dừng không làm dùng cả hệ thống mà chỉ làm dừng 1 tiến trình nhẹ.

**Câu hỏi 4:** Bài toán này yêu cầu bạn so sánh thời gian đọc một tệp (file) của một máy chủ tập tin (file server) đơn luồng và một máy chủ đa luồng. Phải mất tổng cộng 15 ms để nhận 1 yêu cầu (request) và thực hiện quá trình xử lý, giả định rằng các dữ liệu cần thiết nằm ở bộ nhớ đệm trong bộ nhớ chính. Nếu cần thiết phải thực hiện một thao tác truy cập ổ đĩa thì cần thêm 75 ms, biết rằng việc phải thực hiện thao tác này có xắc suất là 1/3. Hỏi máy chủ có thể nhận bao nhiêu yêu cầu/giây trong 2 trường hợp: máy chủ là đơn luồng và máy chủ là đa luồng (ngoài luồng nhận và xử lý request, sẽ có thêm 1 luồng để truy cập ổ đĩa nếu cần thiết)? Giải thích.

- Đơn luồng:

Thời gian để nhận 1 yêu cầu: 15 x 2/3 + (15+75) x 1/3 = 40 (ms)

=> Số yêu cầu/giây: 1000/40 = 250 (yêu cầu)

- Đa luồng: 1000/15 (yêu cầu)

**Câu hỏi 5:** Với việc xây dựng một *server đồng thời*, hãy so sánh việc server này tạo một luồng mới và tạo một tiến trình mới khi nhận được yêu cầu từ phía client.

* Tạo tiến trình mới -> Xử lý tách biệt tài nguyên -> Đơn giản hơn.

Nhưng việc chuyển CPU giữa các tiến trình rất tốn kém tài nguyên của hệ thống, đồng thời khi có quá nhiều tiến trình sẽ cho hệ điều hành phải kiểm soát chặt chẽ độ bảo mật và an toàn.  Dựa trên nguyên tắc: các thao tác lỗi của 1 tiến trình rất khó có thể ảnh hưởng đến 1 tiến trình khác không phụ thuộc

* Các ứng dụng đa luồng thường giúp chương trình có hiệu năng cao hơn
* Nếu muốn các collection trao đổi với nhau thì nên tạo ra luồng

**Câu hỏi 6:** Nếu bây giờ một webserver tổ chức lưu lại thông tin về địa chỉ IP của client và trang web client đó vừa truy cập. Khi có 1 client kết nối với server đó, server sẽ tra xem trong bảng thông tin, nếu tìm thấy thì sẽ gửi nội dung trang web đó cho client. Server này là có trạng thái (stateful) hay không trạng thái (stateless)?

* Stateless là thiết kế không lưu dữ liệu của client trên server. Có nghĩa là sau khi client gửi dữ liệu lên server, server thực thi xong, trả kết quả thì “quan hệ” giữa client và server bị “cắt đứt” – server không lưu bất cứ dữ liệu gì của client.
* Stateful là một thiết kế ngược lại, chúng ta cần server lưu dữ liệu của client, điều đó đồng nghĩa với việc ràng buộc giữa client và server vẫn được giữ sau mỗi request (yêu cầu) của client. Data được lưu lại phía server có thể làm input parameters cho lần kế tiếp.
* Server này có trạng thái stateful, lưu thông tin về địa chỉ IP và trang web client đó vừa truy cập. Thông tin này là tham số đầu vào cho lần request tiếp theo

**Câu hỏi 7:** Trong các giao thức phân tầng, mỗi tầng sẽ có một header riêng. Vậy có nên triển khai một hệ thống mà tất cả các header của các tầng đưa chung vào một phần (gọi là header chung), gắn vào đầu mỗi thông điệp để có thể xử lý chung? Giải thích.

Không nên triển khai 1 hệ thống như vậy vì:

+ khó bảo trì, nâng cấp hệ thống

+ phá vỡ tính độc lập tương đối với nhau thực hiện các chức năng riêng biệt

+ Thay đổi chức năng hoặc giao thức trong 1 tầng thì sẽ làm ảnh hưởng đến tầng khác

* + Mất đi tính trong suốt của hệ thống

+ tại một tầng, cách đóng gói dữ liệu khác nhau theo các giao thức khác nhau -> khó khăn trong việc đóng gói chung vào 1 header

+ khó khăn trong việc bóc tách, xử lý gói tin khi chuyển tiếp