BÀI TẬP TRÊN LỚP

MÔN HỌC: HỆ PHÂN TÁN

CHƯƠNG 2: TIẾN TRÌNH VÀ TRAO ĐỔI THÔNG TIN TRONG HPT

HỌ TÊN SV: Phạm Thị Quyên **MSSV:** 20187195

MÃ LỚP: 121760 **MÃ HỌC PHẦN:** IT4610Q

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

Cần giới hạn số lượng các luồng trong một tiến trình server vì càng tạo nhiều luồng càng tốn bộ nhớ chính

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. Khi 1 luồng có thao tác vào ra tiến trình nhẹ tương ứng bị treo, tuy nhiên các tiến trình nhẹ khác vẫn tiếp tục được thực hiện

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

Không nên chỉ có một tiến trình nhẹ đơn gắn với 1 tiến trình. Việc đồng bộ giữa các tiến trình nhẹ không cần mức kernel. 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 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: Hệ thống X chỉ định máy của user chưa server, trong khi các ứng dụng lại được coi như client. Điều đó có vô lý không? Giải thích.

Không vô lý

Câu hỏi 6: Giao thức thiết kế cho hệ thống X gặp phải vấn đề về tính mở rộng. Chỉ ra các giải pháp để giải quyết vấn đề đó?

Giải pháp:

- Cải thiện giao thức X protocol

- Nén thông điệp

Câu hỏi 7: 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 tốn tài nguyên hơn so với luồng

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

Nếu muốn các collection trao đổi với nhau thì nên tạo ra luồng.

Câu hỏi 8: 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)?

Server có trạng thái stateful vì nó có lưu lại dữ liệu của client

Câu hỏi 9: So sánh Docker và Virtual Machine.

Máy ảo	Docker container
Kích thước (dung lượng) lớn.	Kích thước (dung lượng) nhỏ.
Hiệu suất hạn chế.	Hiệu suất gốc (native).
Mỗi máy ảo sẽ có một hệ điều hành riêng.	Container sẽ sử dụng hệ điều hành của host.
Ảo hóa về mặt phần cứng	Ảo hóa về mặt hệ điều hành
Thời gian khởi động tính theo phút	Thời gian khởi động tính theo mili giây
Phân bổ bộ nhớ theo nhu cầu cần thiết	Yêu cầu ít dung lượng bộ nhớ hơn
Hoàn toàn bị cô lập và an toàn hơn	Cô lập ở mức tiến trình, có thể kém an toàn hơn

Câu hỏi 10: 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 nếu làm vậy thì hệ thống sẽ không còn ý nghĩa phân tầng, các tầng cần độc lập với nhau, việc thay thế, cải tiến sẽ làm mất đi tính trong suốt của hệ thống.

Câu hỏi 11: Xét 1 thủ tục incr với 2 tham số nguyên. Thủ tục làm nhiệm vụ là cộng 1 đơn vị vào

mỗi tham số. Bây giờ xét trường hợp chúng ta gọi thủ tục đó với cùng một biến, ví dụ incr(i, i).

Nếu biến i được khởi tạo giá trị 0, vậy giá trị của i sẽ là bao nhiêu sau khi gọi thủ tục này trong 2 trường hợp sau:

- Lời gọi tham chiếu

- Phương pháp sao chép-phục hồi được sử dụng.?

- Trường hợp 1: Lời gọi tham chiếu

Mỗi con trỏ đều trỏ -> i, nếu i tăng 1 đơn vị -> i tăng 2 lần -> i = 2

- Trường hợp 2: Phương pháp sao chép phục hồi được sử dụng:

Sao lưu không làm tăng giá trị.

Ghi đè giá trị 2 lần -> i =1

Câu hỏi 12: Một kết nối socket cần 4 thông tin nào? Tại sao phải cần đủ 4 thông tin đó

Câu hỏi 13: Tại sao giao thức yêu cầu-trả lời (request-reply) lại được coi là đồng bộ và tin cậy?

Câu hỏi 14: Hai vấn đề chính đối với giao thức RPC là gì?

Hai vấn đề chính đối với giao thức RPC:

- Biểu diễn dữ liệu
- Giao thức truyền tải

Câu hỏi 15: Vấn đề đối với truyền tham biến trong RPC là gì? Còn đối với truyền tham chiếu? Giải pháp đưa ra là gì?

- Vấn đề đối với truyền tham biến: các tham số truyền đi là cont trỏ hay biến chứa địa chỉ của nơi chứa giá trị thực của chúng. Các thủ tục được gọi sẽ căn cứ vào địa chỉ này để tham chiếu đến giá trị khi tính toán. Khi các giá trị này bị thay đổi trong khi thực hiện thủ tục thì sẽ được thông báo cho client và các lần gọi sau sẽ dùng giá trị mới đó
- Vấn đề đối với truyền tham chiếu: vô nghĩa nếu không có bộ nhớ chia sẻ

Giải pháp: mở rộng mô hình RPC ra làm hai loại chính: Synchronous RPC và Asynchronous RPC.

- Asynchronous RPC (RPC không đồng bộ)
  - Client gửi tới server lời gọi thủ tục và chờ bản tin chấp nhận từ server.
  - Phía server sẽ gửi một tín hiệu ACK về cho client thông báo đã nhận được yêu cầu và bắt đầu thực hiện yêu cầu RPC đó.
  - Lúc này client sẽ tếp tục thực hiện công việc của mình mà không chờ kết quả từ server như ở RPC truyền thống.
- Synchronous RPC (RPC đồng bộ)
  - Thực hiện hai lời gọi, một từ client và một từ server.
  - Client gửi tới server lời gọi thủ tục và chờ bản tin chấp nhận từ server.

- Server gửi bản tin chấp nhận về cho client thông báo đã nhận được yêu cầu và bắt đầu thực hiện yêu cầu RPC đó.
- Lúc này client sẽ tếp tục thực hiện công việc của mình.
- Khi thực hiện thủ tục xong, server sẽ thực hiện lời gọi tới client báo nhận lấy kết quả.
- Client thực hiện ngắt, nhận kết quả và gửi lại cho server tín hiệu ACK đã nhận kết quả thành công.

Câu hỏi 16: So sánh RMI và RPC. Nhược điểm của RMI so với RPC là gì?

Cơ sở để so sánh	**RPC**	**RMI**
Hỗ trợ	Lập trình thủ tục	Lập trình hướng đối tượng
Thông số	Cấu trúc dữ liệu thông thường được chuyển đến các thủ tục từ xa.	Các đối tượng được truyền cho các phương thức từ xa.
Hiệu quả	Thấp hơn RMI	Hơn RPC và được hỗ trợ bởi phương pháp lập trình hiện đại (ví dụ: mô hình hướng đối tượng)
Chi phí chung	Hơn	Ít so sánh
Tham số ra là bắt buộc.	Có	Không cần thiết
Cung cấp dễ dàng lập trình	Cao	thấp
Nhược điểm	Vấn đề của RPC là bởi nó ẩn đi thực tế phân tán ở mức cú pháp, điều này gây khó khăn hơn cho các lập trình viên để giải quyết đúng đắn các thách thức cố hữu đi kèm với các khía cạnh vật lý của phân tán. • Lời gọi cục bộ: - Chia sẻ/đồng bộ trạng thái: nếu thủ	Việc gọi phương thức của đối tượng từ xa luôn phức tạp hơn gọi phương thức cục bộ: - Việc tham chiếu đến biến, địa chỉ của các đối tượng khác nhau - Các tham số truyền cho phương thức của đối tượng phải được đóng

	tục được gọi (callee) lỗi -> thủ tục gọi (caller) lỗi - Call semantic luôn là exactly once • RPC: không chia sẻ/đồng bộ trạng thái: - Lỗi chỉ xảy ra một phía - Lỗi trong quá trình truyền tin • Các khả năng khi có lỗi: - Thủ tục không thực thi - Thủ tục thực thi 1 lần - Thủ tục thực thi nhiều lần - Thủ tục thực thi một phần	gói và chuyển qua mạng đến phương thức thực sự (local stack) - Lời gọi phương thức từ xa phải thông qua mạng và có thể bị ngắt ngang do mạng gặp sự cố ➔ Phụ thuộc vào kết nối mạng

Câu hỏi 17: Hàm listen được sử dụng bởi TCP server có tham số là backlog. Giải thích ý nghĩa tham số đó.

Backlog là chiều dài hàng đợi chờ xử lý cho các kết nối đã được thiết lập

Câu hỏi 18: Trong trao đổi thông tin hướng dòng, những cơ chế thực thi QoS được thực hiện ở tầng nào? Giải thích. Trình bày một số cơ chế thực thi QoS để chứng minh điều đó.

Việc thực hiện QoS được dựa trên kiến trúc dịch vụ phân biệt (Diff-Serv), kiến trúc này chỉ định rằng mỗi gói tin được phân loại khi nhập vào mạng. Các gói tin sẽ được đánh dấu như Class of Service (CoS), DSCP, IP Precedence… Nên đánh dấu gói tin càng sớm càng tốt, khi được ưu tiên sớm thì luồng đi qua thiết bị tiếp theo sẽ trơn tru.

Cơ chế thực thi QoS được thực thi ở 2 tầng là tầng liên kết dữ liệu (data-link layer) và tầng mạng (network layer)

- Ở tầng mạng:
  - sử dụng 3 bit đầu tiên trong trường loại dịch vụ (Service Type - ToS) trong phần mào đầu của gói dữ liệu IP và 3 bits đầu tiên (P2 đến P0) dùng để quy định các giá trị đánh dấu độ ưu tiên của packet và các giá trị này được gọi là IP Precedence
  - Giá trị IP precedence nằm trong khoảng từ 0 đến 7.
    - 3 bits đầu tiên (P2 đến P0): IP Precedence. Do sử dụng 3 bits nên sẽ có 8 giá trị (000 đến 111) định ra độ ưu tiên của gói tin từ thấp đến cao. Giúp router xử lý các gói tin này theo chất lượng dịch vụ
    - 3 bits tiếp theo (T2 đến T0):
      bit T2 (T2=1): Yêu cầu truyền gấp.
      bit T1 (T1=1): Yêu cầu truyền với đường truyền chất lượng cao.
      bit T0 (T0=1): Yêu cầu truyền đảm bảo.
    - 2 bit cuối (CU1-CU2): Không dùng tới (Currently and Unused).
  - Trong trường hợp cần thiết phải phân chia nhiều hơn 8 lớp lưu lượng, chúng ta có thể sử dụng 6 bit đầu tiên của trường ToS gọi là trường DSCP
- ở tầng liên kết dữ liệu:

- Trong phần mào đầu của khung dữ liệu ở lớp liên kết dữ liệu không có trường nào phục vụ cho việc phân lớp lưu lượng. Tuy nhiên ta có thể phân lưu lượng dựa vào việc chèn thêm các thẻ định danh VLAN gọi là tag. Mỗi tag gồm 4 byte trong đó trường CoS gồm 3 bit được dùng để phân lớp lưu lượng. Như vậy tại mức liên kết dữ liệu chúng ta cũng có thể phân chia lưu lượng thành 8 lớp với các mức ưu tiên tăng dần tương tự như khi sử dụng IP Precedence tại lớp mạng của gói tin IP.

1 số cơ chế thực thi của QoS:

- Cấu trúc Best-Effort: dữ liệu đi vào mạng đều tuân theo quy tắc FIFO. Không có sự đối xử nào của QoS đối với dữ liệu
- Cấu trúc Guaranteed Services: dữ liệu đi qua mạng được dành riêng 1 băng thông chắc chắn cho dữ liệu. Thực hiện thông qua cơ chế RSVP và CBWFQ của QoS.
- Cấu trúc Differentiated Services: dữ liệu đi vào mạng được phân loại thành các lớp khác nhau để phân loại cách đối xử của mạng đối với dữ liệu. Thực hiện thông qua các tool QoS là PQ, CQ, WFQ và WRED.