BÀI TẬP TRÊN LỚP

MÔN HỌC: HỆ PHÂN TÁN

CHƯƠNG 2: TIẾN TRÌNH VÀ TRAO ĐỔI THÔNG TIN TRONG HỆ PHÂN TÁN

HỌ TÊN SV: Nguyễn Thanh Hà MSSV: 20210298

MÃ LỚP: 154056 MÃ HỌC PHẦN: IT4611

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

Trả lời:

Cần phải giới hạn số lượng các luồng trong một tiến trình server bởi vì:

- Khi tạo thêm luồng thì tiến trình phải tạo thêm 1 stack.
- Thêm bao nhiêu luồng thì thêm bấy nhiêu stack -> Tốn bấy nhiêu bộ nhớ

➔ Đến một lúc nào đó sẽ vượt quá tài nguyên bộ nhớ của hệ thống.

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

Trả lời:

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?

Trả lời:

Không nên vì:

- Nếu chỉ có 1 tiến trình nhẹ thì không có ý nghĩa trong trường hợp đa luồng.
- Đảm bảo khi log 1 luồng không làm log tất cả.
- Không có ý nghĩa khi 1 luồng block sẽ block tiến trình nhẹ.
- Nếu dùng 1 tiến trình nhẹ map với nhiều luồng ở trên.

- Nó sẽ block các luồng khác.
- Không có ý nghĩa khi giải quyết vấn đề block System Call.

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.

Trả lời:

- Trường hợp máy chủ là đơn luồng:

Thời gian để nhận 1 yêu cầu = $15x\frac{2}{3} + (15+75)x\frac{1}{3} = 40$ (s)

⇨ Máy chủ có thể nhận: $\frac{1000}{40} = 25$ (yêu cầu/giây)

- Trường hợp máy chủ là đa luồng:

Máy chủ có thể nhận: $\frac{1000}{15} \approx 66.67$ (yêu cầu/giây)

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.

Trả lời:

Điều này không vô lý bởi

- Theo quan điểm của Window: X tức là X-Window
- Các máy của user-terminal coi như X-server.
- Các máy application-server được coi như client.
- user-terminal phục vụ cho các application-server với vai trò như X-client.

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?

Trả lời:

Giải pháp giải quyết vấn đề:

- Cải thiện giao thức X-protocol.
- Nén thông điệp lại.

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.

Trả lời:

	Đa tiến trình	Đa luồng
Giống nhau	Xử lý song song nhiều công việc.	
Khác nhau	Chi phí lập trình thấp. Không cần quan tâm xung đột giữa các tiến trình. Mỗi tiến trình được tạo có tài nguyên riêng, có môi trường riêng. Hệ điều hành tự đảm bảo không để xảy ra xung đột. Chuyển đổi ngữ cảnh giữa các tiến trình và tốn kém tài nguyên hơn do hệ điều hành tách riêng tài nguyên.	Chi phí lập trình cao. Lập trình viên phải tự đảm bảo vấn đề lập trình xung đột giữa các luồng. Nhẹ nhàng chuyển ngữ cảnh.

⇨ Đa luồng chạy nhanh hơn đa tiến trình chạy xong song do việc chuyển ngữ cảnh nhanh hơn.

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)?

Trả lời:

Server này là có trạng thái (stateful) vì nó lưu lại dữ liệu của client.

Câu hỏi 9: So sánh Docker và Virtual Machine.

Trả lời:

Docker	Virtual Machine
Kích thước (dung lượng) nhỏ	Kích thước (dung lượng) lớn

Hiệu suất gốc (native) Sử dụng hệ điều hành của host Ảo hóa về mặt hệ điều hành Thời gian khởi động nhanh, tính theo mili giây Yêu cầu ít dung lượng bộ nhớ Cô lập ở mức tiến trình, có thể kém an toàn hơn	Hiệu suất hạn chế Mỗi máy ảo sẽ có 1 hệ điều hành riêng Ảo hóa về mặt phần cứng Thời gian khởi động lâu, tính theo phút Phân bổ bộ nhớ theo dung lượng cần thiết Hoàn toàn bị cô lập và 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.

Trả lời:

Không nên triển khai như thế bởi như vậy thì hệ thống sẽ không còn mang ý nghĩa của kiến trúc phân tầng (các tầng hoạt động độc lập với nhau), việc cải tiến sẽ làm mất tính trong suốt của hệ thống. Vậy nên việc tách mỗi tầng có 1 header riêng để đảm bảo tính trong suốt giữa các tầng.

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.

Trả lời:

Giá trị của i sau lời gọi hàm incr(i, i)

- Trường hợp lời gọi tham chiếu:
  Mỗi con trỏ đều trỏ đến biến i.
  i cộng 1 đơn vị => i tăng gấp 2 lần => Vậy i = 2
- Trường hợp phương pháp sao chép – phục hồi được sử dụng:
  Phương pháp sao lưu không làm thay đổi giá trị
  Ghi đè giá trị 2 lần => Vậy i = 1

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

Trả lời:

- Một kết nối socket cần 4 thông tin:
- 2 địa chỉ IP
- 2 số hiệu cổng (port number)

  của 2 máy phía client và server.
- Cần phải đủ 4 thông tin trên bởi: Socket là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên internet. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều để kết nối 2 process trò chuyện với nhau. Để chúng có thể xác định, kết nối được nhau và thông tin được truyền đúng và tới nơi thì cần có cổng và địa chỉ IP.

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

Trả lời:

Giao thức yêu cầu – trả lời (request - reply) được coi là đồng bộ và tin cậy vì:

- Không cần cơ chế kiểm soát luồng.
- Không cần cơ chế báo nhận.
- Client gửi request, nó sẽ tự block mình và đợi phản hồi từ server.
- Server tiếp nhận phản hồi từ client và gửi lại cho client.
- Sau khi client tiếp nhận trả lời từ server nó mới tiếp tục làm việc khác.

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

Trả lời:

Hai vấn đề chính đối với giao thức RPC:

- Hệ thống không đồng nhất:
- Không gian nhớ khác nhau.
- Cách biểu diễn thông tin là khác nhau.
- Khi 1 trong 2 máy lỗi thì sẽ không thể triển khai các thủ tục được.

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?

Trả lời:

- Vấn đề với truyền tham biến:

- Biểu diễn dữ liệu của 2 hệ thống khác nhau.
- Các dữ liệu không thuộc cùng một kiểu, các kiểu dữ liệu khác nhau được biểu diễn khác nhau.

Giải pháp:

- 2 bên gửi và nhận cùng phải thống nhất về đặc tả tham số (tuân thủ 1 kiểu giao thức)
- Cần thống nhất về định dạng thông điệp, cách biểu diễn cấu trúc dữ liệu cơ bản, kiểu trao đổi thông điệp, triển khai client-sub và server-sub

- Vấn đề với truyền tham chiếu:

- Tham chiếu chỉ có ý nghĩa cục bộ với 2 máy tính không đồng nhất.
- Không thực hiện được tham chiếu tới các dữ liệu có cấu trúc.

Giải pháp:

- Cấm sử dụng các tham chiếu khi không cần thiết.
- Tham chiếu thay bằng copy/restore với cách thức hoạt động là chỉ copy một lần (cho input hoặc output) tránh việc tốn kém băng thông, lưu trữ.

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

- So sánh RMI và RPC:
- Giống nhau:

1. Cùng hỗ trợ lập trình giao diện
2. Dựa trên giao thức request/reply
3. Mức độ trong suốt

- Khác nhau:

	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
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:

RPC	RMI
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 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 1 phần	Việc gọi phương thức của đối tượng từ xa luôn phức tạp hơn gọi phương thức cụ 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 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

⇨ Nhược điểm cần giải quyết một số vấn đề như:

1. Định vị đối tượng từ xa
2. Trao đổi thông tin các đối tượng
3. Gọi các phương thức của đối tượng

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.