MÔN HỌC: HỆ PHÂN TOÁN

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

HỌ TÊN: LƯU TUẤN HÙNG

MSSV: 20225131

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

Có cần thiết vì khi tạo thêm một luồng thì tốn thêm stack. Nếu tạo quá nhiều không giới hạn thì đến lúc sẽ vượt quá tài nguyên hệ thống

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

Không cần thiết vì tiến trình nhẹ có bảng luồng để tránh các tiến trình dùng chung luồng. Việc truy cập từ một tiến trình nhẹ của các luồng phụ thuộc vào điều khiển của người dùng.

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

Không nên vì:

* Không có ý nghĩa đa tiến trình
* Khi 1 tiến trình nhẹ bị block cả tiến trình bị block
* Các luồng đều sẽ bị block trong LWP khi một luồng xử lý vào ra

**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. Nếu phải truy cập ổ địa thì cần 90ms. Gọi số yêu cầu là trong 1s x.

X \* 2 /3 \* 15+ X\* 1 / 3 \* 90 = 1000

X = 25

Đa luồng. Việc đọc ổ đĩa là riêng biệt

X = 1000 / 15 = 66.7

**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.

Điều này không vô lý vì

* 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?

Cải thiện thông điệp 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.

Giống: Đều xử lý nhiều công việc song song

Khác:

Tiến trình:

+ Chi phí lập trình thấp

+ Làm việc độc lập, không biết sự tồn tại của nhau, được cấp tài nguyên độc lập

+ Chuyển đổi ngữ cảnh giữa các tiến trình tốn kém tài nguyên

Luồng:

+ Chi phí lập trình cao

+ Yêu cầu lập trình viên phải giải quyết xung đột các luồng

+ Chi phí chuyển ngữ cảnh thấp

**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 là stateful vì đã lưu lại thông tin người dùng

**Câu hỏi 9:** So sánh Docker và Virtual Machine.

|  |  |
| --- | --- |
| Docker | Virtual Machine |
| * Kích thước nhỏ * Hiệu suất gốc * Sử dụng hệ điều hành của host * Ảo hóa về mặt hệ điều hành * Khởi động nhanh * Yêu cầu ít dung lượng * Cô lập ở mức tiến trình, kém an toàn | * Kích thước lớn * Hiệu suất hạn chees * Mỗi máy ảo có hệ điều hành riêng * Ảo hóa về mặt phần cứng * Khởi động lâu * Phân bổ bộ nhớ theo dung lượng cần thiết * Hoàn toàn cô lập, 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 do sẽ mất tính phân tầng, mất 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.

Lời gọi tham chiếu

i được tăng lên hai lần

Phương pháp sao chép – phục hồi

i được tăng lên 1 lần,bị ghi đè

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

Cần 4 thông tin: Cổng và port của Server và client

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?

* 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?

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

Hệ thống không đồng nhất: Không gian nhớ khác nhau và các biểu diễn thông tin khác nhau

Khi một trong hai máy bị lỗi thì không triển khai được

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

Vấn đề với truyền tham biến: Biểu diễn dữ liệu hai bên khác nhau

Giải pháp:

* Hai bên phải thống nhất về đặc tả tham số
* Thống nhất định dạng thông điệp,

Vấn đề với truyền tham chiếu: Tham chiếu chỉ có ý nghĩa cục bộ với hai máy tính không đồng nhất

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?

Giống nhau:

* Cùng hỗ trợ lập trình giao diện
* Dựa trên giao thức request/reply
* 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 suất | Thấp hơn RMI | Cao hơn RPC |
| Chi phí chung | Hơn RMI | Thấp |
| Tham số ra là bắt buộc | Có | Không cần thiết |
| Cung cấp dễ dàng lập trình | Cao | Thấp |

Nhược điểm

RPC

Vấn đê: Ẩn đi thực thế 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

RMI:

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.

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

Backlog là số kết nối tối đa trong hàng chờ lúc chưa được accept()

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

QoS có thể được thực thi ở nhiều tầng khác nhau trong mô hình OSI, nhưng chủ yếu tập trung vào tầng mạng (Network Layer - Layer 3) và tầng giao vận (Transport Layer - Layer 4).

Cơ chế thực thi QoS:

* Điều khiển gói tin được xử lý trong hàng đợi
* Điều chỉnh tốc độ truyền