**Bài tập Hệ phân tán**

**Chương 2: Tiến trình và Trao đổi thông tin**

|  |  |
| --- | --- |
| Họ và tên: Mai Xuân Ngọc  Mã lớp: 141344 | MSSV: 20204769  Mã học phần: IT4611 |

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

**TL:**

Chúng ta cần phải giới hạn số lượng các luồng ở một mức nhất định trong một tiến trình server, Vì:

* Để dễ dàng quản lý các tiến trình, biết được khả năng phục vụ của mỗi tiến trình gồm bao nhiêu luồng.
* Giữ chi phí để đồng bộ và giải quyết tương tranh giữa các luồng không quá lớn.
* Kiểm soát tài nguyên, chi phí chuyển cảnh giữa các tiến trình.
* Tránh việc một luồng gọi hệ thống dừng dẫn đến dừng nhiều các luồng khác, nên lượng luồng song song không được quá nhiều.

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

**TL:**

Việc gắn một luồng đơn nhất duy nhất với một tiến trình nhẹ sẽ làm giảm hiệu quả, tốc độ của đa luồng vì tiến trình nhẹ được điều phối trong tầng nhân (kernal), một luồng đơn chỉ chạy trong một tiến trình nhệ có thể coi như không còn khả năng đa luồng ở tầng sử dụng nữa và chi phí để luồng chính kiểm soát các luồng nhẹ sẽ tăng lên tỉ lệ thuạn với số luồng nhẹ, thậm chí nhiều luồng nhẹ quá sẽ không kiểm soát được. Vì vậy, không nên chỉ gắn một luồng đơn duy nhất vào một tiến trình nhẹ.

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

**TL:**

- Việc gắn tiến trình nhẹ đơn với 1 tiến trình chính sẽ làm mất đi tính chịu lỗi của tiến trình đó. Nếu một luồng thực hiện lời gọi hệ thống dừng sẽ kéo theo tất cả các luồng song song với nó trong tiến trình nhẹ bị dừng theo, sẽ không có sự khác biệt với việc tất cả các luồng này chạy trong một luồng chính. Vì vậy, không nên chỉ có một tiến trình nhẹ đơn gắn với 1 tiến trình chính, sẽ làm phức tạp hệ thống mà không thu được lợi ích nào.

**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.

**TL:**

**Đơn luồng**:

Một yêu cầu:

Giải thích: Mỗi yêu cầu sẽ cần thời gian nhận, thực hiện xử lý yêu cầu là là cố định, đôi khi cần phải thêm cho các thao tác truy nhập ổ đĩa nữa nên thời gian trung bình được tính bao gồm cả thời gian nhận, xử lý và thời gian truy nhập ổ đĩa

**Đa luồng:**

Một yêu cầu:

Giải thích: Mỗi yêu cầu có 1 luồng nhận và xử lý request riêng và có 1 luồng để truy cập ổ đĩa riêng. Nên thời gian tối đa cho một yêu cầu là giá trị lớn hơn giữa tích thời gian nhận, xử lý với xác suất xảy ra và tích cần thêm thời gian truy cập dữ liệu với xác suất xảy ra.

**Câu hỏi 5:** hệ thống X chỉ định máy của user chưa sever, 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 Windows: X tức là X – Windows.

* Các máy của user – terminal coi như là 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.

**TL:**

Việc tạo một tiến trình mới để xử lý yêu cầu tử phía client sẽ cho chất lượng dịch vụ tốt hơn ở phía Client vì mỗi tiến trình được cung cấp tài nguyên riêng để hoạt động. Tuy nhiên, việc tạo ra nhiều tiến trình sẽ làm gia tăng chi phí cho máy chủ. Vì tài nguyên máy chủ là hữu hạn nên việc tạo ra mỗi tiến trình để phục vụ riêng cho 1 client khi gửi yêu cầu tới sẽ dẫn tới việc Overload rất nhanh. Trong trường hợp không đòi hỏi chất lượng dịch vụ quá cao thì sử dụng cơ sể đa luồng ở máy chủ (sử dụng chung tài nguyên) sẽ tăng khả năng phục vụ của máy sever.

|  |  |  |
| --- | --- | --- |
|  | 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)?

**TL:**

Server này không có trạng thái (stateless) vì chỉ tổ chức lại thông tin về địa chỉ IP và trang web client vừa truy cập, không lưu bất kỳ dữ liệu nào từ phía client. Tức là nếu có lỗi trong việc tra bảng thì người dùng hoàn toàn có thể tự truy cập lại dến trang web cũ và không gặp lỗi vì không có dữ liệu riêng nào cho client cần được cung cấp.

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

**Trả lời:**

|  |  |
| --- | --- |
| Docker | Virtual Machine |
| * Kích thước (dung lượng) nhỏ * 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 | * Kích thước (dung lượng) lớn * 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.

**TL:**

Không nên, vì trong kiến trúc phân tầng, mỗi tầng là độc lập, chứa dữ liệu và header riêng và giao tiếp với nhau qua các interface và giao thức. Việc chia header riêng để đảm bảo tính trong suốt trong quá trình trao đổi thông tin giữa các tầng với nhau, nếu gộp chung thành một phần (gọi là header chung) là đã làm mất đi tính chất quan trọng của kiến trúc. Nếu sử dụng header chung tức là tất cả các tầng đều có thể thay đổi, chỉnh sửa phần header có thể dẫn đến không toàn vẹn dữ liệu, gây lỗi lên các tầng khác.

**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 sau lời gọi hàm

* Trường hợp lời gọi tham chiếu:

Mỗi con trỏ đều trỏ đến biến i.

cộng 1 đơn vị tăng gấp 2 lần Vậy

* 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

**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 của 2 máy phía client và server.
* 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 đồi 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.
* Bộ nhớ phân tán.

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