Tổng quan

Câu hỏi 1: Vai trò của Middleware trong Hệ Phân Tán là gì?

- Trả lời:

\* Cài đặt các dịch vụ cơ bản để thực hiện, phát triển các ứng dụng, cung cấp 1 số dịch vụ như: truy cập trong suốt, các phương tiện trao đổi thông tin bậc cao, dịch vụ định danh, dịch vụ lưu trữ bền vững.

\* Giúp cải thiện tính trong suốt của hệ thống, giúp cho việc cấu hình là duy nhất.

\* Kết nối các thành phần phần mềm hoặc các ứng dụng với nhau .

Câu hỏi 2: Tại sao nói Middleware thừa kế các ưu điểm của cả DOS và NOS?

- Trả lời:

\* Trong 1 Hệ Phân Tán có Middleware, các Middleware services sẽ đảm nhiệm nhiệm vụ của Multicomputer OS

- Giúp chia sẻ tài nguyên, trong suốt với ứng dụng

\* Middleware thừa kế tính mở của NOS.

Câu hỏi 3: Giải thích tính trong suốt của HPT, đưa ra các ví dụ cho các kiểu trong suốt khác nhau

- Trả lời:

\* Giải thích tính trong suốt:

+ Che giấu tính phân tán: khiến người dùng cảm giác như chỉ đang tương tác trực tiếp với 1 hệ thống

+ Trong suốt về vị trí: khiến người dùng cảm giác như đang tương tác với 1 hệ thông ở 1 vị trí duy nhất

+ Tính duy nhất với NSD: giao diện của các hệ thống trong hệ phân tán là giống nhau, cách thức sử dụng giống nhau khiến người dùng cảm giác là chỉ đang tương tác với 1 hệ thống duy nhất

\* VD:

+ Truy cập: ví dụ khi người dùng click vào tên người dùng trên facebook thì sẽ hiện ra trang cá nhân của người dùng đó, nhưng cách thức mà facebook truy cập vào tài nguyên người dùng đó là gửi 1 GET request lên cho server yêu cầu trả vê dữ liệu liên quan đến người dùng đó, render ra trang cá nhân của người dùng chứ không chỉ đơn thuần là click vào 1 link - khá là đơn giản như ta tưởng

+ Địa điểm: khi sử dụng các tài nguyên của facebook như các video, hay xem các status thì người dùng không hề biết các dữ liệu đó được lưu trữ ở trung tâm dữ liệu nào của facebook, ở đâu

+ Di trú: người dùng youtube không biết được các dữ liệu của youtube đang được di chuyển đến địa điểm khác

+ Chuyển địa điểm: khi xem video trên youtube thì video đó đã được sao lưu ra 1 bản để người dùng xem, còn bản gốc có thể được đem đến 1 trung tâm dữ liệu khác của google

+ Sao lưu: facebook có các trung tâm dữ liệu đặt tại các khu vực khác nhau trên thế giới để sao cho người dùng ở các khu vực có thể truy cập dữ liệu nhanh nhất có thể nên từ 1 bộ dữ liệu gốc, facebook sao lưu nó ra các bản khác nhau và cung cấp cho các trung tâm dữ liệu của họ trên thế giới.

+ Tương tranh: ví dụ khi xem video trên youtube người dùng không thể biết được có bao nhiêu người cũng đang xem video đó

+ Thứ lỗi: khi một thành phần trong hệ thống bị lỗi sẽ có các thành phần dự phòng khác được sử dụng thay thế để người dùng không nhận ra lỗi trong thời gian fix lỗi, khôi phục tài nguyên

+ Bền vững: người sử dụng youtube không thể biết được video của youtube được lưu trên disk hay 1 hệ thống cloud nào cả.

Câu 4: Tại sao đôi khi rất khó để có thể che giấu sự xuất hiện lỗi và sự phục hồi lỗi trong Hệ Phân Tán?

- Trả lời: Rất khó để che giấu sự xuất hiện lỗi và phục hồi lỗi trong HPT vì khi che giấu lỗi và phục hồi lỗi sẽ khiến cho người dùng quá trong suốt với lỗi, khiến cho khi 1 tài nguyên đã bị lỗi nhưng vẫn được nhiều người dùng truy cập khiến khó phục hồi lỗi, qua đó giảm đi hiệu năng của hệ thống

Câu 5: Tại sao không phải lúc nào cũng là tốt khi giữ độ trong suốt ở mức cao nhất có thể?

- Trả lời: Bởi vì độ trong suốt tỷ lệ nghịch với hiệu năng. Chính vì vậy nếu giữ độ trong suốt ở mức cao nhất thì hiệu năng sẽ thấp nhất. Ví dụ khi để cho độ trong suốt về mặt tương tranh quá cao thì sẽ có quá nhiều người sử dụng truy cập vào 1 tài nguyên có thể gây trễ cho người dùng trong việc truy cập tài nguyên qua đó giảm đi hiệu năng của hệ thống

Câu 6: Thế nào là một Hệ Phân Tán mở? Các lợi ích của tính mở đem lại là gì?

- Trả lời:

\* Hệ phân tán được gọi là mở nếu nó cung cấp các dịch vụ theo các quy tắc mô tả cú pháp và ngữ nghĩa của các dịch vụ, gọi là giao diện. Thường mô tả bằng IDL

\* Các lợi ích của tính mở đem lại:

+ Cho phép các thành phần có thể được sản xuất bởi các nhà sản xuất khác nhau

+ Khả chuyển, mềm dẻo, khả năng phối hợp, mở rộng được, dễ dàng nâng cấp hệ thống. Hệ thống sẽ nhanh chóng hoàn thiện

Câu 7: Những kỹ thuật nào được sử dụng để mở rộng hệ thống?

- Trả lời:

\* Phân tán dữ liệu.

\* Sao lưu dữ liệu.

\* Sử dụng cache

- VD: khi số lường người dùng sử dụng 1 file quá cao, nên để tránh tình trạng tắc nghẽn, tương tranh giữa các người dùng với nhau cần nhân bản file đó lên nhiều hệ thống con trong một hệ phân tán, hoặc có thể cache dữ liệu cho người dùng nhằm tránh việc quá nhiều truy cập vào 1 file từ 1 phía người dùng có mục đích không tốt.

TH - Tổng quan

Câu hỏi 1: Chỉ ra đường dẫn đến file html chứa nội dung của trang web mặc định của apache.

- Trả lời: /var/www/html/index.html

Câu hỏi 2: Cổng mặc định của giao thức http mà web server lắng nghe là cổng nào?

- Trả lời: Cổng mặc định của giao thức http mà web server lắng nghe là cổng 80

Câu hỏi 3: Phân quyền 755 nghĩa là thế nào?

- Trả lời: phân quyền 755 nghĩa là - chủ của file/directory sẽ có đủ 3 quyền: r(đọc), w(ghi), x(thực thi)

- các thành viên khác có thể r(đọc) và x(thực thi) file

Câu hỏi 4: Sau khi gõ 2 địa chỉ trên thì bạn thấy điều gì? giải thích.

- Trả lời: + Với example.com -> hiển thị ra 1 title in đậm:

\* Success! The example.com virtual host is working!

+ Với test.com -> hiển thị ra 1 title in đậm:

\* Success! The test.com virtual host is working!

+ Giải thích: Trong file /etc/hosts khi thêm 2 dòng:

\* 127.0.0.1 example.com

\* 127.0.0.1 test.com

thì khi đó 2 domain này sẽ cùng trỏ về 1 IP là localhost, trước đó do đã khởi động apache2 cùng với việc config 2 file .conf cho 2 domain kia và dùng câu lệnh a2ensite ta đã cho phép 2 domain kia hoạt động và cùng trỏ về địa chỉ IP localhost

Câu hỏi 5: Bây giờ muốn cho các máy khác trong cùng mạng LAN với máy tính vừa làm việc cũng truy cập được vào 2 địa chỉ example.com và test.com thì chúng ta phải làm gì? giải thích.

-Trả lời: Muốn cho các máy khác nhau trong cùng mạng LAN với máy tính vừa làm việc cũng truy cập được

vào 2 địa chỉ example.com và test.com thì chúng ta cần đặt cổng mặc định cho giao thức http và web server

--------------------------------------------------------------------------------------------------------------------------

Architect

Câu 1:

- Nó phụ thuộc vào việc client được tổ chức như thế nào

- Ta có thể chia code ở client thành các phần nhỏ hơn để chúng chạy riêng biệt

- Khi đó nếu 1 phần đang chờ server response thì ta có thể lập lịch cho phần khác

- Ta cũng có thể cấu trúc cho client để nó làm việc khác ngay sau khi nó gửi request cho server

- Một giải pháp khác đó là thay thế tương tác client-server từ đồng bộ thành không đồng bộ

Câu 2:

- Là kiến trúc bao gồm 3 tầng logic

- Mỗi tầng được triển khai ở một máy riêng

- Tầng cao nhất: client UI

- Tầng ở giữa: các ứng dụng

- Tầng thấp nhất: triển khai các dữ liệu đang được sử dụng

Câu 3:

- Hệ phân tán dọc: hệ phân tán nhiều tầng trên nhiều máy khác nhau. Về mặt lí thuyết, 1 tầng được triển khai trên 1 máy

- Hệ phân tán ngang: 1 tầng trên nhiều máy khác nhau

Câu 4:

- Ta chỉ đang xem xét đến đường đi logic

- Nó khá tốt khi hai nodes A & B ở cạnh nhau nhưng khoảng cách xa trong một mạng overlay

- Hệ quả là, nếu đường đi từ A - B khá ngắn thì message được định tuyến theo 1 đường đi khá dài trong mạng overlay

Câu 5:

- Hiệu năng khá tồi khi n lớn

- Vấn đề ở đây chính là sự tương tác gĩưa 2 tầng kế tiếp nhau - hoặc có thể hiểu như là 2 máy khác nhau

- Hệ quả là sự tương tác giữa 2 tâng P1, P2 có thể cần tới n - 2 tương tác (request-reply) giữa các tầng khác nhau

- Vấn đề khác đó là khi 1 máy trong chuỗi bị hỏng, hoặc thực thi tồi thì sẽ giảm đi hiệu năng

Câu 6:

- Giải thuật đó chưa tối ưu: VD: Từ nút A gửi cho nút C theo giải thuật, có thể đi từ A->D->C mất thời gian hơn so với A->B->C

TH - Architect

Câu 1:

Vai trò của application server glassfish: đóng vai trò server trong việc gửi nhận tin giữa hai hay nhiều đối tượng, đóng vai trò trung gian, đảm nhận việc phân phối thông tin. Các máy chủ đóng vai trò quan trọng trong việc xử lý minh bạch nhiều vấn đề liên quan đến mạng, bao gồm cả giao thông đáng tin cậy, có thể yêu cầu xử lý giao dịch hoặc lưu trữ tin dai dẳng.

Câu 2: Phải tạo 2 JNDI như vậy vì: Để có thể sử dụng JMS, thì cần có 1 kết nối JNDI vào server JMS. JNDI là API chạy độc lập, được sử dụng chủ yếu trong JMS như 1 dịch vụ đặt tên. Đầu tiên, 1 khách hàng JMS để truy cập vào 1 nhà cung cấp JMS bằng cách tìm một ConnectionFactory. Hay nói cách khác TopicConnectionFactory được tạo trong phần này, để xử lí, tạo ra các kết nối JMS, để sau đó dùng cho việc gửi và nhận thông điệp. Tiếp theo là, Topic - đối tượng đích, đại diện cho các kênh ảo trong JMS, được cung cấp bởi JNDI để sử dụng ở JMS client.

Câu 3: Giải thích cơ chế chuyền và nhận thông điệp của Sender và Receiver: + Phía Sender: Tìm TopicConnectionFactory với tên "myTopicConnectionFactory" đã tạo từ trước, sau đó khởi tạo và start connection. Tiếp theo tạo Topic Session và lấy Topic object (- tìm kiếm các topic với tên "myTopic"). Tạo 1 đối tượng TopicPublisher để gửi những thông điệp được nhập từ bàn phím. + Phía Receiver: cũng tương tụ Sender, tìm TopicConnectionFactory với tên "myTopicConnectionFactory", sau đó khởi tạo và start connection. Tiếp theo tạo Topic Session và lấy Topic object (- tìm kiếm các topic với tên "myTopic"). Tuy nhiên tiếp đến sẽ là tạo TopicSubcriber, sau đó tạo tiếp đối tượng lắng nghe các thông điệp nhận từ phía Sender (MyListener) -> để xử lí thông điệp nhận được (cụ thể là nhận được và in ra console phía Receiver)

Câu 4: So sánh JMS và DDS: + JMS là một tiêu chuẩn gửi tin nhắn cho phép các thành phần ứng dụng dựa trên Java EE để tạo, gửi, nhận và đọc tin nhắn. Nó cho phép truyền thông giữa các thành phần khác nhau của một ứng dụng phân tán được lỏng lẻo, đáng tin cậy, và không đồng bộ. JMS hỗ trợ cả mạng point-to-point và publisher-subcriber. Hạn chế chính của JMS là nó là một API chỉ có tiêu chuẩn Java và không định nghĩa một giao thức. Vì vậy JMS triển khai từ các nhà cung cấp khác nhau sẽ không tương thích, JMS không thể đảm bảo khả năng tương tác giữa người sản xuất và người tiêu dùng sử dụng các ứng dụng JMS khác nhau. + DDS là một tiêu chuẩn trung gian mở cho phép khả năng mở rộng, thời gian thực, đáng tin cậy, hiệu suất cao và trao đổi dữ liệu tương thích giữa các nhà xuất bản và thuê bao. DDS sử dụng các API chuẩn giúp đảm bảo rằng các ứng dụng DDS có thể được chuyển dễ dàng giữa hiện thực nhà cung cấp khác nhau. Các tiêu chuẩn DDS cũng quy định một giao thức dây, DDS Interoperability Wire Protocol, được gọi là DDSI. Một giao thức dây là cần thiết nếu có nhiều hơn một ứng dụng phải tương thích nghĩa là dữ liệu truyền từ điểm-điểm. Đó là điểm khác biệt lớn nhất so với JMS

----------------------------------------------------------------------------------------------------------------------

Process & Thread

Câu 1: Cần giới hạn số lượng các luồng trong một tiến trình server

- Lí do 1: thread cần bộ nhớ cho việc thiết lập stack của chúng, dẫn đến nếu có nhiều threads thì sẽ cần nhiều tài nguyên

- Lí do 2: Việc có quá nhiều threads sẽ tạo ra sự hỗn độn trên hệ điều hành

- Trong hệ thống bộ nhớ ảo, khá khó trong việc tạo ra sự ổn định, dẫn đến tình trạng nhiều pages bị faults, ảnh hưởng đến I/O

- Ảnh hưởng đến hiệu năng của hệ thống

- Kể cả khi bộ nhớ không bị tràn thì việc truy cập bộ nhớ cũng khá hỗn độn, dẫn đến việc cache không được sử dụng

- Có thể hiệu năng khi đó còn kém hơn khi dùng đơn luồng

Câu 2: Không nên làm. Vì khi đó nó sẽ gây ra hiệu ứng chỉ có các threads ở level kernel, khi đó hiệu năng thu được từ các threads ban đầu đều bị mất

Câu 3: Không nên chỉ có một tiến trình nhẹ đơn gắn với 1 tiến trình. Vì trong cấu trúc này chúng ta chỉ có user-level thread, điều đó có nghĩa là bất kì lời gọi hệ thống nào cũng sẽ block toàn bộ các tiến trình

Câu 4:

- Trong trường hợp đơn luồng, cache hit mất 15ms và cache miss mất 90 ms

- Trọng số trung bình 2/3 \* 15 + 1/3 \* 90 nên nhận 1 request mất 40 ms, server nhận được 25 y/c mỗi s

- Trong trường hợp đa luồng, mọi việc chờ cho đĩa đều bị chồng chéo, nên nhận 1 request mất 15ms, server có thể xử lí 200/3 y/c mỗi s

Câu 5: Điều đó không vô lí. Ý tưởng chủ đạo ở đây là server sẽ điều khiển phần cứng còn các ứng dụng có thể gửi request để điều khiển phần cứng

Câu 6:

- Vấn đề 1: mở rộng số, cần rất nhiều băng thông.

=> Giải quyết: Sử dụng công nghệ nén, để tiết kiệm băng thông 1 cách tối đa.

- Vấn đề 2: mở rộng về mặt địa lí, cần đồng bộ hóa giữa các hệ thống khác nhau,

=> Giải quyết: sử dụng kĩ thuật caching,

Câu 7: Tạo tiến trình mới tốn tài nguyên hơn so với luồng

Tạo tiến trình mới -> xử lí tách bạch tài nguyên -> đơn giản hơn

Nếu muốn các collection trao đổi với nhau thì nên dùng luồng

Câu 8: Đây là một server không trạng thái (stateless server). Điều quan trọng nhất đối với 1 server phi trạng thái đó chính là không có bất kì thông tin nào được maintained bởi server lại được lưu trên client của nó, thay vào đó thì các thông tin này cần phải chính xác. Trong ví dụ này nếu như bảng này bị mất bởi bất cứ lí do gì thì client và server vẫn có thể tương tác với nhau.

TH - Process & Thread

Câu 1: Thông điệp này có nghĩa là server có hoạt động, nhận được request từ client, trả về phía client thời gian hiện tại của hệ thống server khi client request lên

Câu 2: Hiện tượng: WorkerRunnable: MultiThreaded Server - timestamp

Giải thích: ban đầu sẽ tạo ra 1 server đa luồng chạy ở cổng 9000

server sẽ chờ tới khi có client connect tới, gọi hàm accept()

server sẽ gửi về cho phía client 1 thông điệp kèm theo time ở bên phía server và ghi lên luồng ra

------------------------------------------------------------------------------------------------------------------------------

Communication

Câu 1. Không nên, vì dữ liệu truyền từ tầng k + 1 xuống k bao gồm: header, data, tuy nhiên tầng k không thể biết được đâu là header, đâu là data. Nếu có chung 1 header thì việc tất cả các layers có thể đọc, ghi nó có thể làm mất đi tính trong suốt, việc thay đổi protocol của 1 layer có thể được biết bởi các layers khác.

Câu 2.

+ Nếu là "lời gọi tham chiếu": một con trỏ tới i sẽ được truyền cho incr, nó sẽ được tăng 2 lần, nên kết quả cuối cùng sẽ là 2

+ Nếu là "Phương pháp sao chép-phục hồi": i sẽ được truyền bởi tham trị 2 lần, mỗi 1 gía trị sẽ khởi tạo là 0. Cả 2 đều được tăng, nên cả 2 bây giờ sẽ là 1, bây giờ chúng được copy lại, nó sẽ lại được ghi đè. Nên gía trị cuối cùng là 1 không phải 2.

Câu 3. Cả client và server đều tạo socket, nhưng chỉ có server là bind socket vào local endpoint, sau đó server sẽ thực hiện lời gọi "blocking read" trong khi nó chờ dữ liệu từ client. Tương tự, sau khi tạo socket, client cũng chỉ thực hiện 1 "blocking call" để ghi dữ liệu lên server. Nên không cần thiết phải có kết nối.

Câu 4. Có thể thực hiện, tuy nhiên chỉ trên các nền tảng hop-to-hop, nơi mà 1 tiến trình quản lí 1 hàng đợi truyền message sang hàng đợi kế tiếp bằng RPC. Service được cung cấp bởi queue manager đó là lưu trữ các message. Việc gọi queue manager được cung cấp bởi việc triển khai giao diện của 1 proxy tới remote queue, đồng thời cũng nhận về trạng thái: thành công/thất bại của mỗi thao tác.

TH - Communication

Câu 1: Trên server hiện thông báo "add function called"

Giải thích: client và server đang giao tiếp.

Câu 2: Thông báo "parameters: 123, 22" chứng minh server lấy được tham số đúng

Câu 3: Bên client hiện thông báo "result = 145"

Câu 4:Trên server hiện thông báo

add function called

parameters: 5, 7

returning: 12

Đây là kết quả của đoạn lệnh

printf("add function called\n");

printf("parameters: %d, %d\n", argp->a, argp->b);

result = argp->a + argp->b;

printf("returning: %d\n", result);

return &result;

------------------------------------------------------------------------------------------------------------------------------

Naming

Câu 1:

- Một thực thể có thể có nhiều địa chỉ => có thể có nhiều Access Point.

- Một thực thể có thể thay đổi địa chỉ => có thể thay đổi Access Point.

Câu 2: Một từ định danh tham chiếu tới nhiều nhất 1 thực thể. Mỗi thực thể được tham chiếu nhiều nhất bởi 1 từ định danh. Một từ định danh luôn luôn tham chiếu tới cùng 1 thực thể. ( nó không được phép sử dụng lại)

Câu 3.

Tổ hợp service phân cấp vị trí với forwarding pointers. Khi 1 thực thể bắt đầu di chuyển, nó sẽ bỏ lại phía sau 1 forwarding pointers ở A tới vị trí tiếp theo của nó (vị trí tạm thời), cứ mỗi khi nó di chuyển thì 1 forwarding pointers bị bỏ lại. Khi đã tới B, thực thể sẽ chèn địa chỉ mới của nó vào service phân cấp vị trí. Chuỗi các forwarding pointers sẽ bị loại bỏ, địa chỉ tại A bị xóa

Câu 4.

Ta không cần thiết. Xem xét đến câu hỏi trước, gỉa sử rằng chỉ có finger table của node 7 được cài đặt, các thành phần khác được gĩư nguyên. Và điều tồi nhất có thể xảy ra là request cho việc tìm kiếm, gỉa sử là tìm kiếm key 5, được định tuyến đến node 9 thay vì node 7. Nhưng node 9 biết là node 7 đã join vào hệ thống và vì thế nó có thể có được hành động chính xác

Câu 5:

+ Chi phí thay đổi thông tin giảm so với phân giải không đệ quy.

+ Sử dụng phương pháp bộ đệm hiệu quả hơn, tức là các server mức trên sử dụng bộ đệm để lưu lại địa chỉ ít thay đổi của các server mức dưới.

+ Khi khoảng cách giữa Client và Server là rất xa thì phân giải đệ quy sẽ hữu ích vì nó ít tốn tài nguyên mạng do chỉ cần gửi đến 1 server duy nhất 1 lần

----------------------------------------------------------------------------------------------------------------------------

Synchronization

Bài tập lý thuyết: Synchronization

Câu hỏi 1: Trình bày 1 ví dụ để mô phỏng vấn đề gặp phải khi các máy tính/tiến trình hoạt động trong hệ thống phân tán mà không có đồng hộ vật lý dùng chung.

Câu hỏi 2: Tại sao Lamport lại đề xuất sử dụng đồng hồ logic thay cho đồng hồ vật lý trong hệ phân tán?

Câu hỏi 3: Đặc điểm gì của mạng không dây (wireless network) khiến cho thiết kế các giải thuật đồng bộ khác các kiểu mạng khác?

Câu hỏi 4: Giải thuật Lamport được đưa ra để thực hiện loại trừ lẫn nhau (mutual exclusion). Giải thuật được mô tả như sau:

Hệ thống có n tiến trình: P1, P2, ... Pn. Có 1 tài nguyên chia sẻ dùng chung gọi là SR (Shared Resource). Mỗi tiến trình sẽ lưu trữ một hàng đợi queuei để lưu các yêu cầu của các tiến trình khác khi chưa được thực hiện.

Khi tiến trình Pi muốn truy cập vào SR, nó sẽ quảng bá 1 thông điệp REQUEST(tsi,i) cho tất cả các tiến trình khác, đồng thời lưu trữ thông điệp đó vào hàng đợi của mình (queuei) trong đó tsi là timestamp của yêu cầu.

Khi 1 tiến trình Pj nhận được yêu cầu REQUEST(tsi,i) từ tiến trình Pi thì nó đưa yêu cầu đó vào hàng đợi của mình (queuej) và gửi trả lại cho Pi thông điệp REPLY.

Tiến trình Pi sẽ tự cho phép mình sử dụng SR khi nó kiểm tra thấy yêu cầu của nó nằm ở đầu hàng đợi queueivà các yêu cầu khác đều có timestamp lớn hơn yêu cầu của chính nó.

Tiến trình Pi, khi không dùng SR nữa sẽ xóa yêu cầu của nó khỏi hàng đợi và quảng bá thông điệp RELEASE cho tất cả các tiến trình khác.

Khi tiến trình Pj nhận được thông điệp RELEASE từ Pi thì nó sẽ xóa yêu cầu của Pi trong hàng đợi của nó.

Câu hỏi:

a) Để thực hiện thành công 1 tiến trình vào sử dụng SR, hệ thống cần tổng cộng bao nhiêu thông điệp?

b) Có 1 cách cải thiện thuật toán trên như sau: sau khi Pj gửi yêu cầu REQUEST cho các tiến trình khác thì nhận được thông điệp REQUEST từ Pi, nếu nó nhận thấy rằng timestamp của REQUEST nó vừa gửi lớn hơn timestamp của REQUEST của Pi, nó sẽ không gửi thông điệp REPLY cho Pi nữa.

Cải thiện trên có đúng hay không? Và với cải thiện này thì tổng số thông điệp cần để thực hiện thành công 1 tiến trình vào sử dụng SR là bao nhiêu? Giải thích.

Câu hỏi 5: Giải thuật Szymanski được thiết kế để thực hiện loại trừ lẫn nhau. Ý tưởng của giải thuật đó là xây dựng một phòng chờ (waiting room) và có đường ra và đường vào, tương ứng với cổng ra và cổng vào. Ban đầu cổng vào sẽ được mở, cổng ra sẽ đóng. Nếu có một nhóm các tiến trình cùng yêu cầu muốn được sử dụng tài nguyên chung SR (shared resource) thì các tiến trình đó sẽ được xếp hàng ở cổng vào và lần lượt vào phòng chờ. Khi tất cả đã vào phòng chờ rồi thì tiến trình cuối cùng vào phòng sẽ đóng cổng vào và mở cổng ra. Sau đó các tiến trình sẽ lần lượt được sử dụng tài nguyên chung. Tiến trình cuối cùng sử dụng tài nguyên sẽ đóng cổng ra và mở lại cổng vào.

Mỗi tiến trình Pi sẽ có 1 biến flagi, chỉ tiến trình Pi mới có quyền ghi, còn các tiến trình Pj (j ≠ i) thì chỉ đọc được. Trạng thái mở hay đóng cổng sẽ được xác định bằng việc đọc giá trị flag của các tiến trình khác. Mã giả của thuật toán đối với tiến trình i được viết như sau:

#Thực hiện vào phòng đợi

flag[i] ← 1

await(all flag[1..N]∈{0,1,2})

flag[i] ← 3

ifany flag[1..N]=1:

flag[i] ← 2

await(any flag[1..N]=4)

flag[i] ← 4

await(all flag[1..i-1]∈{0,1})

#Sử dụng tài nguyên

#...

#Thực hiện giải phóng tài nguyên

await(all flag[i+1..N]∈{0,1,4})

flag[i] ← 0

Giải thích ký pháp trong thuật toán:

await(điều\_kiện): chờ đến khi thỏa mãn điều\_kiện

all: tất cả

any: có bất kỳ 1 cái nào

Câu hỏi:

flag[i] sẽ có 5 giá trị trạng thái từ 0-4. Dựa vào giải thuật trên, 5 giá trị đó mang ý nghĩa tương ứng nào sau đây (có giải thích):

- Chờ tiến trình khác vào phòng chờ

- Cổng vào được đóng

- Tiến trình i đang ở ngoài phòng chờ

- Rời phòng, mở lại cổng vào nếu không còn ai trong phòng chờ

- Đứng đợi trong phòng chờ

Answers

Câu 1:

Giả sử, trong qúa trình biên dịch chương trình trong một ngôn ngữ nào đó, khi không có sự đồng bộ về mặt thời gian vật lý của trình biên dịch và file code mình đang chỉnh sửa thì rất có thể file code mới nhất của mình sẽ không được biên dịch mà trình biên dịch vẫn biên dịch file code trước đó.

Câu 2:

Người ta quan niệm rằng bất kỳ một tiến trình đơn, một biến cố nào đều có thể sắp xếp một cách duy nhất theo thời gian của địa phương. Tuy nhiên, vì chúng ta không thể đồng bộ hóa một cách hoàn hảo các đồng hộ cục bộ trong hệ phân tán, Lamport đã đề nghị là không nên sử dụng thời gian vật lý để tìm ra trật tự của cặp biến cố bất kỳ nào xẩy ra trong nó. Lamport đề xuất sử dụng đồng hồ logic thay cho đồng hồ vật lý.

Câu 3:

Đặc điểm của mạng không dây khiến giải thuật đồng bộ khác các dạng khác:

- địa chỉ đích không đồng nghĩa với vị trí đích: Trong các mạng khác một địa chỉ tương đương với một địa chỉ vậy lý. Trong mạng không day thì đơn vị được đánh địa chỉ là một trạm (STA). STA là một đích nhận bản tin nhưng nó không có vị trí cố định.

- Môi trường ảnh hưởng tới việc thiết kế: các lớp vật lý sử dụng trong mạng không dây về cơ bản là khác với môi trường truyền hữu tuyến.

- ảnh hưởng của việc giám sát trạm di động: một trong những yêu cầu của mạng không dây là giám sát các trạm di động cũng như các trạm dễ mạng.Trạm dễ mang là trạm có thể di chuyển từ vị trí này sang vị trí khác nhưng chỉ hoạt động trong khi cố định ở một vị trí. còn trạm di động là tạm hoạt động trong khi đang ở trang tháu di chuyển.

- Tương tác với các lớp IEEE 802 khác: mạng không dây phải làm việc với các lớp cao hơn giống như là các mạng khác, điều này đòi hỏi mạng không dây phải xử lý khả năng di động của các trạm trong phân lớp MAC. để đáp ứng độ tin cậy mà lớp LLC đòi hỏi. mạng không dây cần phải phối hợp với các chức năng hoàn toàn mới trong các lớp MAC.