Bài 1:

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

\* Vai trò của Middleware trong hệ phân tán là nó cung cấp các dịch vụ:

- Truy cập trong suốt

- Các phương tiện trao đổi thông tin bậc cao

- Dịch vụ định danh

- Dịch vụ lưu trữ bền vững

- Giao tác phân tán

- Bảo mật

- Các dịch vụ khác

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

- DOS: có tính trong suốt

- NOS: có tính co giãn, tính mở

- Middleware vừa có tính trong suốt vừa có tính co giãn, tính mở nên nó đã kế thừa các ưu điểm của cả DOS và NOS

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

\* HTP có tính trong suốt:

- Hệ thống là duy nhất với NSD: Giao diện giống nhau, Cách thức truy cập giống nhau.

- Trong suốt về quy mô và vị trí

- Che giấu tính phân tán của hệ phân tán

\* Ví dụ cho các kiểu trong suốt khác nhau:

- Truy cập: Che giấu sự khác nhau trong biểu diễn dữ liệu và cách thức truy cập tài nguyên.

- Địa điểm: Che giấu vị trí của tài nguyên

- Di trú: Che giấu việc tài nguyên chuyển đến địa điểm khác

- Chuyển địa điểm: Che giấu việc tài nguyên chuyển đến địa điểm khác trong khi đang được sử dụng

- Sao lưu: Che giấu việc dữ liệu được cung cấp từ nhiều bản sao khác nhau

- Tương tranh: Che giấu việc tài nguyên được truy cập đồng thời bởi nhiều NSD

- Thứ lỗi: Che giấu lỗi và quá trình phục hồi của tài nguyên

- Bền vững: Che giấu việc tài nguyên/dữ liệu được lưu trữ bền vững (disk) hoặc không (RAM)

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

- Khi có lỗi xảy ra sẽ dẫn đến tăng thời gian đáp ứng yêu cầu của NSD. Từ đó NSD sẽ thấy tốc độ giảm, chậm đi, thấy xảy ra lỗi nên sẽ giảm tính trong suốt.

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

- Không phải lúc nào cũng là tốt khi giữ mức độ trong suốt ở mức cao nhất có thể vì:

Đôi khi ta cần phải cân bằng độ trong suốt với hiệu năng của hệ thống.

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

\* HPT mở là:

- Không tạo tính độc quyền cho nhà sản xuất.

- Dễ cải thiện và nâng cấp hệ thống.

- Khi tăng số lượng nhà sản xuất thì hệ thống nhanh chóng phát triển và hoàn thiện.

\* Lợi ích của tính mở đem lại:

- Khả năng phối hợp (interoperability)

- Tính khả chuyển (portability)

- Tính mềm dẻo + mở rộng được (flexibility, extensibility)

- Thực hiện: tách biệt chính sách và cơ chế

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

- Kỹ thuật phân tán

- Kỹ thuật sao lưu

- Kỹ thuật sử dụng bộ đệm

Bài 2

Câu 1: Khi máy client và server đặt xa nhau về mặt địa lý, điều này sẽ dẫn tới việc độ trễ thời gian trao đổi thông tin giữa 2 máy là cao, hậu quả là sẽ làm giảm hiệu năng hệ thống. Bạn có đề xuất gì để giải quyết vấn đề này?

Đề xuất:

- Chuyển từ giao tiếp đồng bộ sang giao tiếp không đồng bộ.

- Sử dụng bộ đệm.

- Sao lưu sang gần phía client hơn.

Câu 2: Kiến trúc client-server 3 tầng là gì?

\* Kiến trúc client-server 3 tầng là:

- Client:

+ gửi yêu cầu, nhận kết quả, hiển thị cho NSD

- Server:

+ lắng nghe, nhận yêu cầu, xử lý, trả lời -Tương tác giữa client và server có thể là hướng kết nối hoặc không hướng kết nối

- Gồm 3 tầng ứng dụng tìm kiếm:

+ User interface

+ Application server

+ Database server

Câu 3: Sự khác nhau giữa phân tán dọc và phân tán ngang là gì?

- Phân tán dọc: Phân chia thep các tầng chức năng

- Phân tán ngang: Phân tán phân tải

Câu 4: Trong một mạng overlay có cấu trúc, các thông điệp được định tuyến dựa theo hình trạng mạng (topology). Nhược điểm quan trọng của hướng tiếp cận này là gì?

- Topology dựa trên các liên kết Logic nên khi 2 node trên mạng tưởng chừng rất gần nhau nhưng thực ra chúng lại ở rất xa nhau.

Câu 5: Xét một chuỗi các tiến trình P1, P2, ..., Pn triển khai một kiến trúc client-server đa tầng. Cơ chế hoạt động của tổ chức đó như sau: tiến trình Pi là client của tiến trình Pi+1, và Pi sẽ trả lời Pi-1 chỉ khi đã nhận được câu trả lời từ Pi+1.

Vậy những vấn đề nào sẽ nảy sinh với tổ chức này khi xem xét hiệu năng yêu cầu-trả lời tới P1?

Làm:

- n càng lớn thì càng ảnh hưởng đến hiệu năng hệ thống.

- 1 tiến trình hỏng sẽ dẫn đến ảnh hưởng tới hiệu năng toàn hệ thống.

Câu 6: Xét mạng CAN như trong hình. Giả sử tất cả các node đều biết node hàng xóm của mình. Một giải thuật định tuyến được đưa ra đó là gửi các gói tin cho node hàng xóm gần mình nhất và hướng đến đích. Giải thuật này có tốt không? Giải thích

- Giải thuật không tốt.

- Lý do:

Giả sử cần tìm định tuyến từ (0.2,0.3) đến (0.9,0.6)

Theo giải thuật định tuyến thì đường đi là: (0.2,0.3) -> (0.7,0.2) -> (0.9,0.6)

Nhưng đây không phải con đường ngắn nhất, con đường ngắn nhất là: (0.2,0.3) -> (0.6,0.7) -> (0.9,0.6)

=> Giải thuật này không tốt.

**Tuần 3 :**

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

- Cần thiết phải giới hạn số lượng các luồng trong 1 tiến trình server.

Vì: Người lập trình cần phải tính toán đến sự an toàn và hợp lý của tương tác các luồng do nó ảnh hưởng trực tiếp đến hiệu năng và chi phí lập trình.

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

- Nên chỉ gắn một luồng đơn duy nhất với một tiến trình nhẹ.

Vì: Sự trục trặc của luồng này sẽ không làm ảnh hưởng đến luồng và tiến trình khác.

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

- Không nên chỉ có một tiến trình nhẹ đơn găn với một tiến trình

Câu 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.

cứ 3 request có 1 request cần sử đụng đến ổ đĩa :

+ Trong trường hợp xử lí đơn luồng : Thời gian thực hiện 3 request là : 15 + 15 + 15 + 75 = 120 ms

Số tiến trình thực hiện được là : (1000/120)\*3 = 25 request

+ Trong trường hợp xử lí đa luồng :

- Giả sử request sử dụng ổ đĩa là request đầu tiên :

Thời gian thực hiện 3 request là : 15 + 15 + 15 + 75 -15 -15 = 90 ms

Số tiến trình thực hiện được là : (1000/90)\*3 = 33 request

Giả sử request sử dụng ổ đĩa là request thứ 2 :

Thời gian thực hiện 3 request là : 15 + 15 + 15 + 75 -15 = 105ms

Số tiến trình thực hiện được là : (1000/105)\*3 = 28 request

Câu 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.

Có thể làm được. Đó là, các máy chủ điều khiển phần cứng và các ứng dụng có thể gửi yêu cầu để thao tác phần cứng. Từ đó các cửa sổ của máy chủ X thực sự phải lưu trên máy của người sử dụng, có các ứng dụng hoạt động như khách hàng của mình.

Câu 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?

Có hai vấn đề khả năng mở rộng. Đầu tiên, việc có nhiều băng thông là cần thiết. Bằng cách sử dụng kỹ thuật nén, băng thông có thể được giảm đáng kể. Thứ hai, có một vấn đề là mở rộng địa lý như một ứng dụng và màn hình hiển thị thông thường cần phải đồng bộ hóa quá nhiều. Bằng cách sử dụng các kỹ thuật cachinng mà hiệu quả trạng thái của màn hình được duy trì ở mặt ứng dụng, giao thông đồng bộ nhiều có thể tránh được các ứng dụng có thể kiểm tra bộ nhớ cache của địa phương để vây ra những trạng thái của màn hình.

Câu 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)?

Nó có thể được lập luận rằng đây là một máy chủ không quốc tịch. Các vấn đề quan trọng với thiết kế không quốc tịch không phải là nếu bất kỳ thông tin được duy trì bởi các máy chủ vào khách hàng của mình, nhưng thay vì cho dù thông tin đó là cần thiết cho đúng đắn. Trong ví dụ này, nếu bảng bị mất cho những gì bao giờ lý do, khách hàng và máy chủ vẫn có thể đúng cách tương tác như không có gì xảy ra. Trong một thiết kế trạng thái, một sự tương tác như vậy sẽ có thể chỉ sau khi máy chủ đã phục hồi từ một lỗi có thể.

**Tuần 4 :**

Câu hỏi 2: Xét 1 thủ tục incr với 2 tham số nguyên. Thủ tục làm nhiệm vụ là cộng 2 tham số đó với nhau. Bây giờ xét trường hợp chúng ta gọi thủ tục đó với cùng một biến 2 lần, ví dụ incr(i, j). 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.

Thủ tục *incr* với 2 tham số nguyên. Thủ tục làm nhiệm vụ là cộng 2 tham số đó với nhau. Bây giờ xét trường hợp chúng ta gọi thủ tục đó với cùng một biến 2 lần, ví dụ *incr(i, j).* Nếu biến *i* được khởi tạo giá trị *0*, vậy i sẽ có giá trị như sau trong 2 trường hợp:

- Lời gọi tham chiếu:

+ i=2j.

+ Do lời gọi tham chiếu nên giá trị của i sẽ thay đổi sau mỗi lần được gọi.

Lần 1 i=incr(0,j)=j

Lần 2: i=incr(j,j)=2j.

- Phương pháp sao chép-phục hồi được sử dụng:

+ i=0.

+ Do:

CT gọi copy các dữ liệu vào Stack

CT gọi phục hồi các dữ liệu từ Stack

Sau mỗi lần gọi thì giá trị của i lại được phục hồi về 0.

Câu hỏi 4: Liệu có thể áp dụng phương pháp trao đổi thông tin bất đồng bộ bền vững vào cơ chế RPC được không? Phân tích.

- Có thể áp dụng phương pháp trao đổi thông tin bất đồng bộ bền vững vào cơ chế RPC được.

- Do:

+ RPC: Client yêu cầu server thực hiện và trả lại kết quả

+ Có nhiều trường hợp không cần trả lại kết quả

+ Client sau khi gọi RPC tiếp tục thực hiện, không quan tâm đến kết quả trả lại.

Vì vậy ta có thể áp dụng phương pháp trao đổi thông tin bất đồng bộ bền vững vào cơ chế RPC.

**Tuần 5 :**

**Câu 1: Tại sao không thể lấy địa chỉ của Access Point để sử dụng như địa chỉ của thực thể?**

- Không thể lấy địa chỉ của Access Point để sử dụng như địa chỉ của thực thể vì:

+ Một thực thể có thể thay đổi địa chỉ, từ đó dẫn đến thay đổi Access Point.

+ Một thực thể có thể có nhiều địa chỉ, nên sẽ có thể có nhiều Access Point.

**Câu 2:Với việc sử dụng Định Danh, các vấn đề gì cần phải xem xét?**

- Với việc sử dụng Định Danh, các vấn đề cần phải xem xét là:

+ Khi thực thể không còn tồn tại, liệu còn Định Danh không.

+ Khi sử dụng lại các Định Danh thì có xảy ra trường hợp bị trùng không.

+ Không gian Định Danh không thể là vô hạn, nên sẽ dần cạn kiệt Định Danh.

**Câu 5: Phân giải tên đệ qui có ưu điểm gì so với phân giải tên không đệ qui?**

- Phân giải tên đệ qui có ưu điểm so với phân giải tên không đệ qui:

+ 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.

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

**Tuần 6:**

Câu 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.  
  
Các tiến trình nằm trên các máy khác nhau, do đó các máy có độ lệch về thời gian vật lý.  
  
Ví dụ:  
  
Lập trình và dịch compile từ file .c sang file .o  
  
Lập trình viên sửa file .c thì compile không dịch toàn bộ mà chỉ dịch những file .c có thời gian sau thời gian của file .o đã dịch tương ứng, hay là chỉ dịch những file mà lập trình viên sửa.  
  
Giả sử: nằm trên máy khác nhau: 1 máy lập trình, 1 máy dịch:  
  
ở thời điểm người lập trình viên chỉnh sửa file .c có thể xảy ra trước thời điểm file .o tương ứng do độ lêch thời gian giữa máy lập trình và máy dịch do đó dù file .c đã chỉnh sửa nhưng sẽ không được compile lại nữa   
  
Do đó sẽ gây ảnh hưởng tới chương trình, làm chó kết quả chạy không như mong muốn.  
  
Câu 2: Tại sao Lamport lại đề xuất sử dụng đồng hồ logic thay cho đồng hồ vật lý trong hệ phân tán?  
  
Vì: Lamport chỉ quan tâm đến thứ tự các sự kiện xảy ra trong hệ phân tán,  
  
mà không cần hiệu chỉnh tất cả các máy cùng thời gian vật lý.  
  
Chỉ hiệu chỉnh những clock mà thời gian gưir lớn hơn thời gian nhận bằng cách gán cho thời gian nhận bằng Max của( thời gian gửi +1, thời gian nhận)  
  
Câu 3: Đặc điểm gì của mạng không dây (wireless network) khiến cho thiết kế các giải thuật đồng bộ khác các kiểu mạng khác?  
  
Mỗi nút trong mạng hoạt động dựa trên năng lượng của nó.  
  
Do đó cần tìm cơ chế để tiết kiệm năng lượng, tối ưu hóa gửi và nhận  
  
chỉ những nút nhận thông điệp mới tham gia đồng bộ hóa.  
  
Mỗi nút biết được độ lệch của mình so với các nút khác.  
  
Câu 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. **Làm:**   
Nếu có n tiến trình P1,P2, .. Pn;  
  
a, Trong hệ thống sẽ cẫn tất cả:  
  
Mỗi tiến trình sẽ gửi n-1 thông điệp REQUEST tới n-1 tiến trình còn lại do đó có: n(n-1) thông điệp REQUEST  
  
Mỗi tiến trình sẽ gửi n-1 thông điệp REPLY trả lại n-1 tiến trình còn lại do đó có: n(n-1) thông điệp REPLY  
  
khi 1 tiến trình sử dụng xong SR thì sẽ gửi n-1 thông điệp RELEASE tới n-1 tiến trình còn lại;  
  
Do đó sẽ có 2n(n-1) + n-1 thông điệp.  
  
b, <Nếu timestamp nhỏ hơn sẽ không gửi trả REPLY??>  
  
Đúng vì 1 tiến trình muốn dùng SR sẽ cần nhận tất cả các REPLY từ n-1 tiến trình còn lại:  
  
Mỗi tiến trình sẽ gửi n-1 thông điệp REQUEST tới n-1 tiến trình còn lại do đó có: n(n-1) thông điệp REQUEST  
  
tiến trình i sẽ gửi i-1 thông điệp REPLY do cơ chế trên do đó có: n(n-1)/2 thông điệp REPLY  
  
tiến trình khi sử dụng xong SR sẽ gửi lại n-1 thông điệp REPLY tới n-1 thông điệp còn lại  
  
Do đó sẽ có tất cả n(n-1) +n(n-1)/2 + n-1 thông điệp  
  
Câu 5:  
  
2:Chờ tiến trình khác vào phòng chờ  
  
Chờ các tiến trình khác vào phòng chờ và chwof cổng đóng để sử dụng SR  
  
4:Cổng vào được đóng  
  
Các tiến trình ở trong phòng chờ sẽ chờ cho đến khi cổng được đóng để bắt đầu sử dụng SR  
  
1:Tiến trình i đang ở ngoài phòng chờ:  
  
tiến trình i ở ngoài phòng chờ khi: sử dụng xong SR, đang chờ vào phòng chờ . hoặc do cổng đã bị đóng.  
  
0: Rời phòng, mở lại cổng vào nếu không còn ai trong phòng chờ  
  
3: Đứng đợi trong phòng chờ