Bài 1:

Câu 1:

\* 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:

- 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:

\* 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:

- 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:

- 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:

\* 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:

- Kỹ thuật phân tán

- Kỹ thuật sao lưu

- Kỹ thuật sử dụng bộ đệm

Bài 2

Câu 1:

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:

- 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:

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

- Phân tán ngang: Phân tán phân tải

Câu 4:

- 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:

- 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:

- 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:

- 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:

- 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:

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

Câu 4 : 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:

**Tuần 4 :**

Câu hỏi 2:

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:

- 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:**

- 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 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 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.