Câu 1:

1. Hai đặc điểm quan trọng nhất của hệ thống phân tán

a, Tính minh bạch trong phân tán

Hệ thống phân tán được thiết kế sao cho người dùng hoặc lập trình viên không cần biết chính xác tài nguyên hay dịch vụ họ sử dụng đang nằm ở đâu. Ví dụ, khi ta truy cập một tệp tin từ hệ thống, ta không cần biết tệp đó nằm trên máy nào – hệ thống tự xử lý việc kết nối, truyền dữ liệu. Điều này giúp việc sử dụng hệ thống trở nên đơn giản và liền mạch như khi làm việc với một hệ thống đơn.

b, Tính độc lập về vị trí

Các thành phần trong hệ thống phân tán có thể được đặt ở bất kỳ đâu trong mạng. Việc các máy chủ hay tài nguyên ở xa hay gần không ảnh hưởng tới chức năng của hệ thống. Nhờ đặc điểm này, hệ thống dễ mở rộng, dễ thay đổi, nâng cấp mà không ảnh hưởng lớn đến toàn bộ hệ thống.

2. Ba lý do cơ bản khiến các ứng dụng phân tán phức tạp hơn so với các ứng dụng đơn lẻ

a, Giao tiếp qua mạng phức tạp hơn:

Trong ứng dụng đơn lẻ, các thành phần của chương trình thường giao tiếp trực tiếp trong cùng một máy. Nhưng với ứng dụng phân tán, việc giao tiếp diễn ra qua mạng, điều này kéo theo nhiều vấn đề như mất kết nối, trễ mạng, lỗi truyền dữ liệu.

b, Xử lý lỗi khó hơn:

Khi một phần của hệ thống phân tán gặp sự cố (ví dụ: một máy chủ bị tắt), toàn bộ hệ thống có thể bị ảnh hưởng. Việc phát hiện và xử lý lỗi trong môi trường phân tán phức tạp hơn vì không dễ biết chính xác thành phần nào bị hỏng, và làm sao để tiếp tục hoạt động khi một phần bị lỗi.

c, Đồng bộ hóa dữ liệu khó khăn:

Do các phần của hệ thống phân tán hoạt động độc lập và có thể ở nhiều vị trí khác nhau, việc đồng bộ hóa dữ liệu để đảm bảo tính nhất quán là một thử thách lớn. Ví dụ: nếu hai người cùng chỉnh sửa một tài liệu từ hai nơi khác nhau cùng lúc, hệ thống phải có cơ chế để tránh mất dữ liệu hoặc xung đột.

Câu 2:

1. Phân tích các yếu tố phần cứng và yếu tố mạng ảnh hưởng đến hiệu năng hệ thống phân tán

a, Yếu tố phần cứng

- CPU mạnh sẽ xử lý nhanh hơn các tác vụ được giao trong hệ thống phân tán. Nếu một nút trong mạng có CPU yếu, nó sẽ bị nghẽn khiến cả hệ thống chậm lại. Ngoài ra, sự không đồng đều giữa các CPU trong các nút cũng có thể gây mất cân bằng tải.

- Dung lượng và tốc độ truy xuất của bộ nhớ RAM ảnh hưởng trực tiếp đến khả năng lưu trữ tạm thời và xử lý dữ liệu. Nếu RAM không đủ, hệ thống sẽ phải dùng bộ nhớ phụ (như ổ đĩa), làm giảm hiệu năng.

- Với kênh truyền nội bộ (bus, I/O), tốc độ truy xuất dữ liệu giữa CPU, bộ nhớ và các thiết bị trong cùng một máy cũng ảnh hưởng đến tốc độ xử lý chung. Nếu thiết bị đầu vào/ra (I/O) chậm, các tiến trình cần đọc/ghi dữ liệu sẽ bị chậm lại.

b, Yếu tố mạng

- Băng thông càng lớn thì dữ liệu truyền qua mạng càng nhanh, giảm độ trễ trong giao tiếp giữa các nút. Nếu băng thông thấp, tốc độ truyền dữ liệu sẽ bị nghẽn, ảnh hưởng lớn đến hiệu suất tổng thể.

- Topology (cấu trúc kết nối mạng): Cách các nút trong hệ thống được kết nối với nhau (hình sao, hình vòng, lưới...) ảnh hưởng đến tốc độ truyền dữ liệu và khả năng chịu lỗi. Ví dụ, topology có điểm trung tâm dễ bị tê liệt nếu nút trung tâm gặp lỗi; trong khi topology dạng lưới phân tán đều hơn nhưng phức tạp hơn trong điều phối.

2. Tại sao hệ điều hành phân tán (Distributed OS) và hệ điều hành mạng (Network OS) lại có yêu cầu khác nhau về quản lý tài nguyên?

Hệ điều hành mạng network OS là hệ điều hành chạy trên từng máy riêng biệt trong mạng. Mỗi máy tự quản lý tài nguyên của nó, và người dùng phải biết rõ mình đang truy cập tài nguyên từ máy nào. Có việc chia sẻ tài nguyên nhưng theo cách thủ công hoặc qua các dịch vụ cụ thể như chia sẻ file hoặc máy in.

⇒ Quản lý tài nguyên phân tán ở mức thấp, chủ yếu dựa vào người dùng và các giao thức mạng.

Hệ điều hành phân tán Distributed OS tạo cảm giác như toàn bộ hệ thống phân tán là một máy duy nhất. Người dùng không cần quan tâm tài nguyên đang nằm ở đâu, hệ điều hành sẽ tự động phân phối, cân bằng và tối ưu hóa việc sử dụng tài nguyên trên nhiều nút.

⇒ Quản lý tài nguyên tự động, đồng bộ và thông minh hơn, đòi hỏi hệ điều hành phải có khả năng giám sát, điều phối và xử lý lỗi trong toàn hệ thống.

Câu 3:

1. Nêu và so sánh ba loại hệ thống phân tán: điện toán phân tán, thông tin phân tán và lan tỏa phân tán

Loại hệ thống	Đặc điểm chính	Mục tiêu
Điện toán phân tán	Chia nhỏ một bài toán lớn thành các phần nhỏ để xử lý song song trên nhiều máy tính	Tăng tốc độ xử lý, hiệu suất sử dụng tài nguyên
Thông tin phân tán	Dữ liệu được lưu trữ và truy cập từ nhiều vị trí khác nhau trong mạng	Đảm bảo truy cập dữ liệu nhanh, an toàn và hiệu quả
Lan tỏa phân tán (ubiquitous/pervasive)	Các thiết bị nhỏ (cảm biến, thiết bị IoT) hoạt động liên tục, âm thầm trong môi trường	Tạo ra môi trường thông minh, phản ứng theo ngữ cảnh

2. Phân tích các lớp chính trong kiến trúc điện toán lưới (Grid Computing)

Kiến trúc điện toán lưới thường chia thành ba lớp chính:

a, Lớp tài nguyên (Resource Layer)

Vai trò: Quản lý các tài nguyên vật lý và logic như CPU, bộ nhớ, ổ đĩa, máy chủ, cảm biến...

Chức năng:

- Theo dõi trạng thái tài nguyên (đang rảnh hay bận).

- Quản lý truy cập tài nguyên (phân quyền, lịch trình sử dụng,...).

- Cung cấp thông tin cho tầng middleware về năng lực và tình trạng tài nguyên.

b, Lớp trung gian (Middleware Layer)

Vai trò: Là cầu nối giữa tài nguyên và ứng dụng. Tầng này giúp che giấu sự phức tạp của hệ thống phân tán.

Chức năng:

- Lập lịch (scheduling) các tác vụ.

- Cân bằng tải giữa các nút tính toán.

- Đảm bảo bảo mật, quản lý người dùng.

- Xử lý lỗi và phục hồi hệ thống.

c, Lớp ứng dụng (Application Layer)

Vai trò: Là tầng người dùng tương tác trực tiếp để chạy các ứng dụng trên hệ thống điện toán lưới.

Chức năng:

- Gửi yêu cầu tính toán, xử lý dữ liệu.

- Nhận kết quả từ hệ thống lưới.

- Có thể là phần mềm khoa học, mô phỏng, render đồ họa,...

Câu 4:

1. Giải thích tại sao "tính sẵn sàng" (availability) được xem là mục tiêu quan trọng nhất của hệ thống phân tán

Tính sẵn sàng là khả năng của hệ thống cung cấp dịch vụ liên tục và ổn định, ngay cả khi một số thành phần gặp lỗi. Trong hệ thống phân tán, các tài nguyên, dịch vụ, và ứng dụng được phân tán trên nhiều máy khác nhau. Nếu một phần hệ thống không hoạt động, nhưng phần còn lại vẫn duy trì dịch vụ, thì hệ thống được xem là sẵn sàng.

Tính sẵn sàng được coi là quan trọng nhất vì:

- Người dùng luôn kỳ vọng hệ thống hoạt động không bị gián đoạn.

- Các lỗi phần cứng, mạng, hoặc phần mềm là không thể tránh khỏi, nên hệ thống phải biết cách chịu lỗi và duy trì dịch vụ.

- Hệ thống có thể có hiệu năng thấp hoặc dữ liệu bị trễ một chút, nhưng nếu không sẵn sàng, thì người dùng không thể sử dụng hệ thống – điều này nghiêm trọng hơn.

2. Nêu và so sánh ba hình thức "tính trong suốt" (Transparency)

Loại tính trong suốt	**Mô tả**	**Mục tiêu**
Trong suốt về truy nhập	Người dùng truy cập tài nguyên mà không cần biết cách thức truy cập	Làm cho việc truy cập tài nguyên giống như truy cập cục bộ

Trong suốt về vị trí	Người dùng không cần biết tài nguyên ở vị trí địa lý nào	Giảm sự phụ thuộc vào vị trí vật lý của tài nguyên
Trong suốt về lỗi	Hệ thống che giấu hoặc xử lý lỗi mà không ảnh hưởng người dùng	Giúp hệ thống đáng tin cậy hơn

3. Trình bày mối quan hệ giữa "tính mở" (openness) và khả năng tương tác (interoperability)

Tính mở của hệ thống phân tán nghĩa là hệ thống được thiết kế theo các chuẩn công khai, giao diện rõ ràng, cho phép các thành phần từ nhiều nhà cung cấp khác nhau có thể tích hợp và hoạt động chung.

Khả năng tương tác (interoperability) là kết quả trực tiếp của tính mở. Khi một hệ thống sử dụng các chuẩn chung, các thành phần hoặc phần mềm từ nhiều bên có thể giao tiếp, phối hợp và chia sẻ tài nguyên dễ dàng.

Ví dụ: Một hệ thống điện toán đám mây cho phép người dùng sử dụng máy ảo từ nhiều nền tảng (Windows, Linux, macOS). Điều này chỉ có thể thực hiện được nếu hệ thống có giao diện mở, hỗ trợ tương tác đa nền tảng.

Câu 5:

1. So sánh ưu – nhược điểm của kiến trúc phân cấp và kiến trúc ngang hàng trong hệ thống phân tán

Tiêu chí	**Kiến trúc phân cấp (Hierarchical)**	**Kiến trúc ngang hàng (Peer-to-Peer – P2P)**
Ưu điểm	- Quản lý dễ dàng nhờ cấu trúc rõ ràng - Dễ kiểm soát truy cập và bảo mật - Phân công vai trò rõ ràng (máy chủ – máy khách)	- Tăng khả năng mở rộng - Phân tải đều giữa các nút - Không có điểm chết trung tâm
Nhược điểm	- Gây quá tải tại nút trung tâm (server) - Khó mở rộng linh hoạt	- Khó kiểm soát tài nguyên và bảo mật - Đồng bộ dữ liệu phức tạp

	- Dễ bị tê liệt nếu nút chính gặp lỗi	- Giao tiếp có thể chậm nếu không tối ưu
Ví dụ	Web Server – Client (trình duyệt truy cập web)	Torrent, hệ thống chia sẻ tệp như BitTorrent

2. Trình bày bốn mô hình hệ thống phân tán và ví dụ ứng dụng

Mô hình	**Mô tả**	**Ví dụ ứng dụng điển hình**
Mô hình phân tầng (Layered Model)	Hệ thống được chia thành các lớp rõ ràng (giao diện, logic xử lý, dữ liệu). Mỗi lớp chỉ giao tiếp với lớp gần kề.	Ứng dụng Web: trình duyệt (giao diện) ↔ máy chủ ứng dụng ↔ cơ sở dữ liệu
Mô hình đối tượng phân tán (Distributed Object Model)	Các thành phần hệ thống được xây dựng thành các đối tượng có thể giao tiếp từ xa (remote object).	Java RMI, CORBA – gọi phương thức trên đối tượng ở máy khác
Mô hình kênh sự kiện (Event-based Model)	Các thành phần giao tiếp thông qua việc phát và nhận sự kiện. Thường dùng trong hệ thống phản ứng nhanh.	Hệ thống cảnh báo cháy (cảm biến phát sự kiện → trung tâm xử lý nhận)
Mô hình dữ liệu tập trung (Data-Centric Model)	Dữ liệu được lưu trữ tập trung và các máy khác truy cập dữ liệu qua mạng.	Cơ sở dữ liệu trung tâm cho hệ thống bán hàng hoặc ERP

3. Vai trò của phần mềm trung gian (middleware) trong kiến trúc khách–chủ phân tán

Phần mềm trung gian (middleware) là lớp phần mềm nằm giữa ứng dụng (client) và tài nguyên (server) trong hệ thống phân tán. Nó đóng vai trò cầu nối, giúp các thành phần khác biệt có thể giao tiếp, chia sẻ tài nguyên, và làm việc hiệu quả với nhau mà không cần quan tâm tới chi tiết hạ tầng bên dưới.

Vai trò chính:

- Ẩn sự phức tạp của hệ thống phân tán như vị trí, giao thức truyền thông, xử lý lỗi,...

- Cung cấp giao diện đồng nhất để lập trình viên phát triển ứng dụng dễ dàng hơn.

- Đảm bảo kết nối và tương tác giữa các thành phần bất kể hệ điều hành, ngôn ngữ hay kiến trúc.

Ba tính năng chính của middleware:

- Giao tiếp giữa các tiến trình (communication support): Cho phép các tiến trình trên các máy khác nhau giao tiếp (RPC, message passing,...).

- Định vị và truy cập dịch vụ (service discovery and access): Giúp client tìm và truy cập các dịch vụ phù hợp trong hệ thống.

- Quản lý tài nguyên và bảo mật (resource and security management): Kiểm soát truy cập, xác thực người dùng, phân quyền và quản lý tài nguyên mạng.

Câu 6:

1. Phân loại ba loại dịch vụ trong kiến trúc hướng dịch vụ (SOA)

Trong SOA (Service-Oriented Architecture), dịch vụ được chia thành ba loại chính:

Loại dịch vụ	**Mô tả**	**Ví dụ điển hình**
Dịch vụ cơ bản (Basic Services)	Là các dịch vụ nhỏ, thực hiện một chức năng cụ thể và độc lập. Chúng thường tương ứng với thao tác CRUD (Create – Read – Update – Delete).	Dịch vụ "Tra cứu thông tin khách hàng" từ CSDL.
Dịch vụ tích hợp (Composite/Integration Services)	Tích hợp nhiều dịch vụ cơ bản lại để thực hiện một quy trình lớn hơn. Thường dùng để gom nhiều tác vụ nhỏ thành một dịch vụ tổng hợp.	Dịch vụ "Quản lý đơn hàng" gọi các dịch vụ kiểm tra kho, tạo đơn, tính phí vận chuyển.
Dịch vụ quy trình (Process Services)	Dịch vụ ở mức cao nhất, mô tả và thực thi toàn bộ quy trình nghiệp vụ (workflow) bằng cách điều phối các dịch vụ tích hợp và cơ bản.	Dịch vụ "Xử lý giao dịch mua hàng" gồm kiểm tra tồn kho, thanh toán, gửi email xác nhận, cập nhật hệ thống ERP.

2. Vòng đời của một dịch vụ SOA và các thách thức ở mỗi giai đoạn

Vòng đời của một dịch vụ SOA thường gồm 5 giai đoạn chính:

Giai đoạn 1. Phân tích và thiết kế (Analysis & Design)

- Mục tiêu: Xác định dịch vụ cần thiết, thiết kế giao diện (interface), mô tả chức năng và ràng buộc nghiệp vụ.

- Thách thức:

+ Khó xác định ranh giới giữa các dịch vụ.

+ Thiết kế dịch vụ sao cho tái sử dụng và độc lập.

Giai đoạn 2. Phát triển (Implementation)

- Mục tiêu: Xây dựng logic bên trong của dịch vụ (thường dùng SOAP/REST), kiểm tra chức năng cơ bản.

- Thách thức:

+ Tích hợp với hệ thống cũ (legacy).

+ Đảm bảo tuân thủ chuẩn SOA.

+ Kiểm tra lỗi giao tiếp mạng và xử lý ngoại lệ.

Giai đoạn 3. Đóng gói và triển khai (Packaging & Deployment)

- Mục tiêu: Đưa dịch vụ lên môi trường thực thi (application server), đăng ký vào Service Registry (UDDI).

- Thách thức:

+ Tự động hoá triển khai.

+ Đồng bộ giữa các phiên bản dịch vụ.

+ Đảm bảo bảo mật khi triển khai công khai.

Giai đoạn 4. Vận hành và giám sát (Operation & Monitoring)

- Mục tiêu: Theo dõi hiệu suất, xử lý lỗi, điều phối các dịch vụ trong quy trình nghiệp vụ thực tế.

- Thách thức:

+ Phát hiện lỗi khó do hệ thống phân tán.

+ Giám sát cần công cụ chuyên dụng.

+ Khó truy vết nếu thiếu cơ chế logging tốt.

Giai đoạn 5. Bảo trì và cập nhật (Maintenance & Evolution)

- Mục tiêu: Cập nhật dịch vụ (thêm tính năng, sửa lỗi) mà không ảnh hưởng tới người dùng đang sử dụng phiên bản cũ.

- Thách thức:

+ Quản lý nhiều phiên bản cùng lúc.

+ Đảm bảo tương thích ngược (backward compatibility).

+ Cập nhật mà không gây gián đoạn dịch vụ.