Họ tên	Mã HV
Phạm Minh Dương	B24CHHT064
Ma Công Thành	NCS2024.20
Vũ Quỳnh Anh	B24CHHT056

Câu 1:

- Nêu và giải thích hai đặc điểm quan trọng nhất của hệ thống phân tán.

Hai điểm quan trọng nhất của hệ thống phân tán là:

- Tập hợp các phần tử tính toán có thể hoạt động độc lập với nhau (tính độc lập): mỗi thành phần (ví dụ: máy chủ, tiến trình, dịch vụ) trong hệ thống phân tán có khả năng tự mình thực hiện các tác vụ, xử lý dữ liệu và ra quyết định mà không cần sự can thiệp liên tục từ các thành phần khác. Chúng sở hữu tài nguyên riêng (CPU, bộ nhớ, ổ đĩa) và có thể chạy các quy trình của riêng mình. Sự độc lập này là nền tảng cho việc đạt được **khả năng mở rộng (scalability)** và **khả năng chịu lỗi (fault tolerance)**. Nếu một thành phần gặp sự cố, các thành phần khác vẫn có thể tiếp tục hoạt động, tránh việc toàn bộ hệ thống bị sập.
- Phần tử tính toán cần phải cộng tác để giải quyết một nhiệm vụ chung (Tính cộng tác): Mặc dù các thành phần hoạt động độc lập, mục tiêu cuối cùng của hệ thống phân tán là hoàn thành một nhiệm vụ lớn hơn mà không một thành phần nào có thể giải quyết một mình. Để đạt được điều này, các thành phần phải **giao tiếp và phối hợp** với nhau. Sự cộng tác này có thể bao gồm việc trao đổi dữ liệu, gửi yêu cầu, phản hồi, hoặc đồng bộ hóa trạng thái để đảm bảo rằng tất cả đều đóng góp vào mục tiêu chung một cách mạch lạc.

- Trình bày ba lý do cơ bản khiến các ứng dụng phân tán phức tạp hơn so với các ứng dụng đơn lẻ.

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

- **Nhu cầu chia sẻ thông tin và tài nguyên:** Các ứng dụng phân tán ra đời do nhu cầu chia sẻ thông tin và tài nguyên giữa người sử dụng. Dữ liệu có thể được lưu trữ phân tán, điều này đặt ra các vấn đề về phân quyền truy cập, ví dụ như cho phép truy cập dữ liệu từ xa nhưng không cho phép sao chép để lưu giữ cục bộ.
- **Vấn đề hiệu năng:** Trong nhiều trường hợp, việc tính toán phân tán là bắt buộc nhằm tận dụng khả năng tính toán song song hoặc khai thác khả năng tính toán của các máy tính chuyên dụng để nâng cao hiệu năng hệ thống.
- **Yêu cầu về khả năng chịu lỗi:** Hệ thống phân tán cần phải đảm bảo an toàn tuyệt đối ngay cả khi có sự cố xảy ra trên một máy chủ nào đó thì cũng không làm ảnh hưởng đến hoạt động của toàn bộ hệ thống. Điều này đòi hỏi việc thực hiện các biện

pháp nhằm phát hiện kịp thời và xử lý lỗi, ví dụ như chuyển sang các máy tính dự phòng khác để dịch vụ không bị gián đoạn nếu máy chủ bị hỏng hoàn toàn.

Câu 2:

- Phân tích các yếu tố phần cứng (CPU, bộ nhớ, kênh truyền) và yếu tố mạng (băng thông, topology) ảnh hưởng đến hiệu năng hệ thống phân tán.

Các yếu tố Phần cứng và Mạng ảnh hưởng đến Hiệu năng Hệ thống Phân tán:

- **Yếu tố Phần cứng (CPU, Bộ nhớ, Kênh truyền trên bo mạch chủ):** Tốc độ của bộ vi xử lý trung tâm (CPU), dung lượng bộ nhớ (RAM), và tốc độ kênh truyền trên bo mạch chủ là những yếu tố cốt lõi quyết định khả năng tính toán trên một máy tính. Trong hệ thống phân tán, mỗi nút tính toán có tài nguyên phần cứng riêng, và năng lực của từng nút này đóng góp trực tiếp vào hiệu năng tổng thể khi xử lý các tác vụ cục bộ.
- **Yếu tố Mạng (Băng thông, Topology, Độ trễ):**
    - **Mạng:** Trong hệ thống phân tán, trao đổi thông tin giữa các máy tính được thực hiện hoàn toàn trên môi trường mạng. Mạng đóng vai trò thiết yếu trong việc phân phối dữ liệu đến các thành phần và tập hợp kết quả tính toán từ chúng. Các máy tính không chia sẻ bộ nhớ, do đó thông tin trao đổi giữa chúng phải thông qua cơ chế trao đổi thông điệp qua mạng.
    - **Băng thông:** Đây là tài nguyên chung của hệ thống và là một yếu tố quan trọng cần xem xét. Băng thông mạng quyết định lượng dữ liệu có thể truyền tải trong một đơn vị thời gian, ảnh hưởng trực tiếp đến tốc độ trao đổi thông tin giữa các thành phần phân tán.
    - **Độ trễ và nghẽn mạng:** Hiệu năng của hệ thống yêu cầu thời gian đáp ứng nhanh chóng cho người dùng. Để đạt được điều này, cần có các giải pháp hạn chế tối đa độ trễ trên các kênh truyền vật lý và khắc phục hiện tượng nút cổ chai hoặc nghẽn mạng. Độ trễ chuyển tin trên mạng có thể lên tới vài chục đến hàng trăm mili giây, điều này là một thách thức lớn cần giải quyết trong hệ thống phân tán.

- 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?

Yêu cầu khác nhau về Quản lý Tài nguyên giữa Hệ điều hành Phân tán và Hệ điều hành Mạng: Sự khác biệt nằm ở mức độ tích hợp và mục tiêu quản lý tài nguyên:

- Quản lý Tài nguyên giữa Hệ điều hành Phân tán hướng tới việc tạo ra một "máy tính lớn ảo" từ nhiều máy vật lý, quản lý tài nguyên một cách tập trung và trong suốt, che giấu sự phân tán của phần cứng đối với người dùng và ứng dụng.

- Hệ điều hành Mạng giữ nguyên tính độc lập của từng máy tính, chỉ tập trung vào việc cung cấp các dịch vụ mạng để các máy tính có thể truy cập tài nguyên từ xa, mà không cố gắng che giấu bản chất phân tán của hệ thống.

Câu 3:

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

- Điện toán phân tán (Distributed Computing)
  - *Khái niệm:* Tập trung vào việc xử lý các tác vụ tính toán phức tạp bằng cách chia nhỏ chúng và phân phối cho nhiều máy tính (nút) trong một mạng lưới.
  - *Mục tiêu:* Tăng cường sức mạnh tính toán, giải quyết các bài toán lớn vượt quá khả năng của một máy tính đơn lẻ.
  - *Đặc điểm nổi bật:*
    - Tập trung vào hiệu suất và khả năng tính toán.
    - Thường sử dụng các thuật toán song song hoặc phân tán.
    - Ít quan tâm đến việc quản lý dữ liệu toàn diện, mà tập trung vào việc xử lý dữ liệu cục bộ hoặc dữ liệu được truyền tải để tính toán.
    - Ví dụ: Điện toán lưới (Grid Computing), điện toán đám mây (Cloud Computing) cho các tác vụ tính toán chuyên sâu (ví dụ: mô phỏng khoa học, phân tích dữ liệu lớn).
- Thông tin phân tán (Distributed Information Systems)
  - *Khái niệm:* Tập trung vào việc quản lý và truy cập dữ liệu được phân tán trên nhiều máy tính hoặc địa điểm khác nhau. Mục tiêu chính là cung cấp một cái nhìn tổng thể và nhất quán về dữ liệu, bất kể vị trí vật lý của chúng.
  - *Mục tiêu:* Quản lý, lưu trữ và truy cập dữ liệu hiệu quả trên các hệ thống phân tán.
  - *Đặc điểm nổi bật:*
    - Tập trung vào tính nhất quán, tính sẵn sàng và tính toàn vẹn của dữ liệu.
    - Thường liên quan đến các hệ quản trị cơ sở dữ liệu phân tán (Distributed Database Management Systems - DDBMS), hệ thống tệp phân tán (Distributed File Systems).
    - Các giao dịch phân tán (distributed transactions) là một khía cạnh quan trọng để đảm bảo tính nhất quán của dữ liệu.
    - Ví dụ: Hệ thống ngân hàng, hệ thống đặt vé trực tuyến, các ứng dụng doanh nghiệp lớn với dữ liệu phân tán.

- Lan tỏa phân tán (Pervasive/Ubiquitous Computing)
  - *Khái niệm:* Là một xu hướng điện toán mà máy tính và các thiết bị tính toán được tích hợp một cách liền mạch vào môi trường hàng ngày của con người, trở nên "vô hình" và cung cấp các dịch vụ một cách tự động và liên tục. Nó tập trung vào sự tương tác tự nhiên và ngữ cảnh giữa con người và công nghệ.
  - *Mục tiêu:* Tích hợp công nghệ vào môi trường sống và làm việc để cung cấp các dịch vụ thông minh, ngữ cảnh và không gián đoạn.
  - *Đặc điểm nổi bật:*
    - Tập trung vào môi trường thông minh, cảm biến, thiết bị nhúng và tương tác ngữ cảnh.
    - Tính di động, tính sẵn sàng và khả năng thích ứng cao với môi trường thay đổi.
    - Thường liên quan đến Internet of Things (IoT), nhà thông minh (smart homes), thành phố thông minh (smart cities), hệ thống y tế điện tử (e-health).
    - Yêu cầu xử lý dữ liệu từ nhiều nguồn khác nhau, thường là dữ liệu thời gian thực và có tính ngữ cảnh.

**** Phân tích các lớp chính (application, middleware, resource) trong kiến trúc điện toán lưới và vai trò của từng lớp.***

- Lớp Tài nguyên (Resource Layer)
  - *Khái niệm:* Là lớp thấp nhất trong kiến trúc điện toán lưới, chứa tất cả các tài nguyên vật lý và logic có sẵn trong lưới. Đây là nơi các tài nguyên được chia sẻ và quản lý một cách cục bộ trước khi được cung cấp cho lưới.
  - *Các thành phần chính:*
    - Tài nguyên vật lý: Máy tính, máy chủ, cụm máy tính (clusters), thiết bị lưu trữ, mạng, thiết bị đo đạc, v.v.
    - Tài nguyên logic/phần mềm: Phần mềm ứng dụng, thư viện tính toán, cơ sở dữ liệu, dịch vụ web cục bộ, v.v.
    - Các giao diện quản lý tài nguyên cục bộ: API hoặc công cụ để quản lý tài nguyên trên từng nút riêng lẻ.
  - *Vai trò:*
    - Cung cấp tài nguyên cơ bản: Đảm bảo rằng các tài nguyên tính toán, lưu trữ và mạng có sẵn và hoạt động.
    - Quản lý tài nguyên cục bộ: Thực hiện các tác vụ quản lý tài nguyên ở cấp độ cục bộ (ví dụ: cấp phát tài nguyên cho các tác vụ đang chạy trên một máy chủ cụ thể).

        - Trừu tượng hóa tài nguyên: Biến các tài nguyên vật lý đa dạng thành một dạng trừu tượng hơn để các lớp trên có thể dễ dàng truy cập và sử dụng.

- Lớp Phần mềm trung gian (Middleware Layer)
    - *Khái niệm:* Là lớp quan trọng nhất trong kiến trúc điện toán lưới, đóng vai trò cầu nối giữa lớp tài nguyên và lớp ứng dụng. Nó cung cấp các dịch vụ và công cụ cần thiết để quản lý, điều phối và truy cập các tài nguyên phân tán một cách hiệu quả, tạo ra một môi trường lưới thống nhất.
    - *Các thành phần chính:*
        - Dịch vụ bảo mật (Security Services): Xác thực, ủy quyền, mã hóa để đảm bảo an toàn cho các giao tiếp và truy cập tài nguyên trên lưới.
        - Dịch vụ khám phá tài nguyên (Resource Discovery Services): Cho phép người dùng hoặc các ứng dụng tìm kiếm và xác định các tài nguyên phù hợp có sẵn trên lưới.
        - Dịch vụ lập lịch và điều phối (Scheduling and Brokering Services): Quyết định tài nguyên nào sẽ được sử dụng cho một tác vụ cụ thể, dựa trên các yếu tố như khả năng sẵn có, hiệu suất, chi phí, v.v.
        - Dịch vụ quản lý tác vụ (Job Management Services): Quản lý vòng đời của các tác vụ, từ khi khởi tạo, thực thi, đến giám sát và báo cáo kết quả.
        - Dịch vụ dữ liệu (Data Services): Quản lý việc lưu trữ, truy xuất và di chuyển dữ liệu giữa các tài nguyên trên lưới.
        - Dịch vụ giám sát (Monitoring Services): Theo dõi trạng thái và hiệu suất của các tài nguyên và tác vụ trên lưới.
    - *Vai trò:*
        - Thống nhất hóa tài nguyên: Tập hợp các tài nguyên phân tán từ nhiều địa điểm khác nhau thành một hệ thống thống nhất.
        - Cung cấp dịch vụ chung: Cung cấp các dịch vụ cốt lõi giúp các ứng dụng và người dùng tương tác với lưới một cách dễ dàng và hiệu quả.
        - Ẩn đi sự phức tạp của phân tán: Che giấu sự phức tạp về vị trí, tính không đồng nhất và sự thay đổi của các tài nguyên bên dưới, cho phép người dùng và ứng dụng xem lưới như một tài nguyên tính toán duy nhất.
        - Đảm bảo tính linh hoạt và mở rộng: Cho phép thêm hoặc bớt tài nguyên một cách linh hoạt mà không ảnh hưởng đến hoạt động của hệ thống.
- Lớp Ứng dụng (Application Layer)
    - *Khái niệm:* Là lớp cao nhất, nơi các ứng dụng thực tế sử dụng các dịch vụ và tài nguyên của lưới để giải quyết các vấn đề cụ thể. Các ứng dụng này được thiết

kế để tận dụng sức mạnh tính toán và khả năng lưu trữ của lưới.

- *Các thành phần chính:*
  - Ứng dụng người dùng: Các chương trình phần mềm được phát triển để chạy trên lưới, thường là các ứng dụng khoa học, kỹ thuật, tài chính, v.v., đòi hỏi nhiều tài nguyên.
  - Cổng lưới (Grid Portals): Giao diện web hoặc ứng dụng đồ họa cung cấp một điểm truy cập thân thiện cho người dùng để gửi tác vụ, giám sát và quản lý ứng dụng của họ trên lưới.
  - API ứng dụng (Application Programming Interfaces - APIs): Các giao diện lập trình cho phép các nhà phát triển tích hợp ứng dụng của họ với các dịch vụ của lớp phần mềm trung gian.
- *Vai trò:*
  - Giải quyết vấn đề thực tế: Sử dụng năng lực của lưới để giải quyết các bài toán phức tạp trong nhiều lĩnh vực khác nhau (ví dụ: mô phỏng khí hậu, phát hiện thuốc mới, phân tích gen).
  - Cung cấp giao diện cho người dùng: Cho phép người dùng tương tác với lưới một cách dễ dàng, gửi và quản lý các tác vụ của họ.
  - Tận dụng hiệu quả tài nguyên: Thiết kế các ứng dụng để khai thác tối đa khả năng song song và phân tán của lưới, đạt được hiệu suất cao.

Câu 4:

**** Tại sao "tính sẵn sàng" (availability) được xem là mục tiêu quan trọng nhất của hệ thống phân tán.***

"Tính sẵn sàng" (availability) được xem là mục tiêu quan trọng nhất của hệ thống phân tán vì mục tiêu quan trọng nhất của hệ thống phân tán là đảm bảo khả năng sẵn sàng của tài nguyên cho các dịch vụ. Tính sẵn sàng có nghĩa là người sử dụng có thể tiếp cận các tài nguyên một cách thuận tiện nhất tại bất kỳ thời điểm nào mà không phụ thuộc vị trí địa lý.

+ Tài nguyên ở đây bao gồm mọi thứ có thể chia sẻ hoặc cung cấp dịch vụ, từ tài nguyên phần cứng như máy in, máy tính, phương tiện lưu trữ, đến dữ liệu cung cấp thông tin hữu ích cho người sử dụng.
+ Khi nhu cầu kết nối và chia sẻ tài nguyên tăng lên, việc đảm bảo tài nguyên luôn "sẵn sàng" là yếu tố then chốt để đáp ứng nhu cầu kinh tế, trao đổi thông tin và mở rộng các hệ thống thương mại điện tử.

**** Nêu và so sánh ba hình thức "tính trong suốt" (trong suốt về truy nhập, vị trí và lỗi), và ví dụ đơn giản minh họa mỗi loại.***

- Trong suốt về truy nhập (Access Transparency):

- *Giải thích:* Che giấu những điểm khác biệt trong việc thể hiện dữ liệu và cách truy nhập tài nguyên. Mục tiêu là đạt được sự đồng thuận về cách thể hiện trên các nền tảng khác nhau, ẩn đi sự đa dạng của kiến trúc máy và hệ điều hành.
- *Ví dụ:* Trong một hệ thống phân tán không đồng nhất, các máy tính có thể cài đặt những hệ điều hành khác nhau với quy tắc đặt tên cho các tập tin khác nhau (ví dụ: */home/user/file.txt* trên Linux và *C:\Users\user\Documents\file.txt* trên Windows). Tính trong suốt về truy nhập sẽ che giấu sự khác biệt này để người dùng chỉ thấy một cách truy cập thống nhất.

- Trong suốt về vị trí (Location Transparency):

- *Giải thích:* Che giấu nơi tài nguyên thực sự trú ngụ. Việc đặt tên tài nguyên một cách thân thiện và công khai cho người dùng nhưng vẫn đảm bảo tính bí mật vị trí của nó.
- *Ví dụ:* Khi truy cập trang web *www.hanu.edu.vn*, người dùng không cần biết máy chủ vật lý của trang web đó đang đặt ở Hà Nội, TP.HCM hay bất kỳ đâu trên thế giới. Tên miền này không tiết lộ thông tin địa lý của máy chủ.

- Trong suốt về lỗi (Failure Transparency):

- *Giải thích:* Che giấu lỗi và vấn đề phục hồi sau khi lỗi xảy ra. Hệ thống phải có khả năng chịu lỗi, nếu khắc phục kịp thời thì người dùng sẽ cảm thấy như lỗi chưa từng xảy ra. Mục tiêu là hạn chế tối đa những thông báo lỗi và cách phục hồi, đồng thời phát hiện lỗi và thông báo cho người quản trị.
- *Ví dụ:* Khi bạn đang thực hiện giao dịch ngân hàng trực tuyến và máy chủ cơ sở dữ liệu chính gặp sự cố, hệ thống tự động chuyển sang máy chủ dự phòng mà bạn không hề hay biết. Giao dịch của bạn vẫn được hoàn tất mà không bị gián đoạn, hoặc bạn chỉ thấy một thông báo "Hệ thống đang bận, vui lòng thử lại sau" thay vì thông báo lỗi cụ thể về máy chủ.

**** Trình bày mối quan hệ giữa "tính mở" (openness) và khả năng tương tác (interoperability) trong hệ thống phân tán.***

Tính mở (Openness) của một hệ thống phân tán được hiểu là khả năng của nó kết nối đến các hệ thống khác, dễ dàng nâng cấp/mở rộng dịch vụ, và tương thích với các nền tảng phần cứng và hệ điều hành khác nhau. Để đạt được mục tiêu này, hệ thống phải tuân thủ các chuẩn giao tiếp đã được công bố, cho phép sản phẩm của các nhà sản xuất khác nhau có thể tương tác với nhau theo một tập hợp các quy tắc và chuẩn hóa.

Khả năng tương tác (Interoperability) là một khía cạnh quan trọng của tính mở. Nó thể hiện việc cài đặt các thành phần của những nhà sản xuất khác nhau có thể cùng làm việc chỉ dựa trên các dịch vụ của nhau như đã mô tả trong các tiêu chuẩn chung. Nói cách khác, khi một hệ thống là "mở", nó cho phép các thành phần từ nhiều nguồn khác nhau (thông qua việc tuân thủ các chuẩn mở) có thể tương tác (interoperate) với nhau một cách hiệu quả.

Mối quan hệ chặt chẽ giữa hai khái niệm này là: tính mở là điều kiện tiên quyết để đạt được khả năng tương tác. Một hệ thống chỉ có thể tương tác hiệu quả với các hệ thống khác nếu nó được thiết kế theo các chuẩn mở, cho phép các thành phần khác nhau "hiểu" và "giao tiếp" được với nhau mà không gặp rào cản về công nghệ hay nền tảng riêng biệt. Ngược lại, khả năng tương tác là một minh chứng rõ ràng cho thấy hệ thống đó có tính mở cao.

Câu 5:

**** 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 Architecture)	Kiến trúc ngang hàng (P2P Architecture)
Định nghĩa	Kiến trúc phân cấp, còn gọi là kiến trúc dọc, tổ chức các thành phần trong hệ thống dưới dạng hình cây. Trong đó, các nút cao hơn (nút cha) có vai trò trung tâm hoặc quan trọng hơn so với các nút thấp hơn (nút con)	Trong kiến trúc này, mỗi nút (peer) trong hệ thống có thể đồng thời đóng vai trò là cả client và server. Không có máy chủ trung tâm cố định. Các nút tương tác trực tiếp với nhau để chia sẻ tài nguyên, dữ liệu hoặc dịch vụ.
Ưu điểm	• *Quản lý và kiểm soát tập trung:* Dữ liệu và tài nguyên được quản lý tại máy chủ trung tâm, giúp cho việc kiểm soát, bảo mật và sao lưu trở nên đơn giản và hiệu quả hơn. • *Độ tin cậy và nhất quán dữ liệu cao:* Do chỉ có một điểm quản lý dữ liệu trung tâm, việc đảm bảo tính nhất quán và toàn vẹn của dữ liệu sẽ dễ dàng hơn so với kiến trúc ngang hàng. • *Hiệu suất ổn định cho các tác vụ cụ thể:* Các máy chủ thường là các máy tính có cấu hình mạnh, được tối ưu hóa để xử lý các yêu cầu từ nhiều	• *Khả năng mở rộng cao:* Dễ dàng thêm hoặc bớt các nút vào hệ thống. Khi số lượng nút tăng lên, tài nguyên và khả năng xử lý của hệ thống cũng tăng theo. • *Không có điểm lỗi duy nhất:* Do không có máy chủ trung tâm, sự cố của một nút không ảnh hưởng đến toàn bộ hệ thống. Hệ thống có khả năng tự phục hồi tốt hơn. • *Tăng cường khả năng chịu lỗi:* Nếu một nút bị lỗi, các nút khác vẫn có thể tiếp tục hoạt động và cung cấp dịch vụ. • *Tận dụng tài nguyên phân*

	máy khách, do đó hiệu suất thường ổn định và có thể dự đoán được. • *Dễ dàng triển khai và bảo trì:* Mô hình này khá phổ biến và dễ hiểu, việc phát triển ứng dụng, triển khai và bảo trì hệ thống cũng đơn giản hơn cho các nhà phát triển.	*tán:* Tận dụng hiệu quả tài nguyên (băng thông, lưu trữ, CPU) của các nút riêng lẻ, thường là các thiết bị của người dùng cuối. • *Giảm chi phí cơ sở hạ tầng:* Không cần đầu tư lớn vào các máy chủ tập trung mạnh mẽ. • *Chống kiểm duyệt tốt hơn:* Khó bị kiểm soát hoặc tắt bỏ bởi một tổ chức duy nhất.
	• *Điểm lỗi duy nhất (Single Point of Failure):* Nếu máy chủ trung tâm gặp sự cố, toàn bộ hệ thống sẽ ngừng hoạt động. Đây là nhược điểm lớn nhất của mô hình này. • *Khả năng mở rộng hạn chế:* Khi số lượng máy khách tăng lên, máy chủ trung tâm có thể bị quá tải, dẫn đến hiệu suất giảm. Việc nâng cấp và mở rộng hệ thống thường tốn kém và phức tạp. • *Chi phí cao:* Chi phí để xây dựng và duy trì một máy chủ trung tâm mạnh mẽ, có khả năng xử lý cao thường rất tốn kém. • *Phụ thuộc vào máy chủ:* Mọi hoạt động của máy khách đều phải thông qua máy chủ, làm tăng độ trễ và lưu lượng mạng không cần thiết cho các tác vụ có thể thực hiện trực tiếp giữa các máy khách.	• *Quản lý và kiểm soát phức tạp:* Việc quản lý dữ liệu, kiểm soát phiên bản và đảm bảo tính nhất quán trên một hệ thống phân tán phi tập trung là rất khó khăn. • *Bảo mật khó khăn:* Việc đảm bảo an toàn cho dữ liệu và giao tiếp giữa các nút là một thách thức lớn do tính chất phân tán và không đồng nhất. • *Khó tìm kiếm và khám phá tài nguyên:* Tìm kiếm một tài nguyên cụ thể trên một mạng P2P lớn có thể tốn thời gian và không hiệu quả nếu không có cơ chế lập chỉ mục tốt. • *Vấn đề về hiệu suất và độ tin cậy:* Hiệu suất có thể không ổn định vì nó phụ thuộc vào sự sẵn có và chất lượng kết nối của các nút tham gia. Một số nút có thể không đáng tin cậy (ví dụ: tắt đột ngột). • *Khó triển khai và gỡ lỗi:* Do

		tính chất phức tạp và phi tập trung, việc phát triển và gỡ lỗi ứng dụng P2P thường khó hơn.
Ứng dụng	Kiến trúc phân cấp được ứng dụng rộng rãi trong các hệ thống yêu cầu quản lý và kiểm soát tập trung như các máy chủ web (Web Server) phục vụ trang web cho người dùng, và các hệ thống quản trị cơ sở dữ liệu (Database Management Systems) nơi máy chủ quản lý toàn bộ dữ liệu.	Kiến trúc P2P phù hợp với các ứng dụng yêu cầu khả năng mở rộng cao, chịu lỗi tốt, tận dụng tài nguyên phân tán, và ít phụ thuộc vào một điểm trung tâm (ví dụ: Chia sẻ file (BitTorrent), tiền mã hóa (Bitcoin), gọi thoại video (Skype phiên bản cũ)).

**** Trình bày bốn mô hình hệ thống phân tán (phân tầng, đối tượng phân tán, kênh sự kiện, dữ liệu tập trung) và cho ví dụ ứng dụng điển hình cho mỗi mô hình.***

- Mô hình Phân tầng (Layered Model)
  - *Mô tả:* Trong mô hình này, hệ thống được tổ chức thành các lớp (tầng) riêng biệt, mỗi lớp cung cấp một tập hợp các dịch vụ cho lớp phía trên nó và sử dụng các dịch vụ từ lớp phía dưới. Các giao tiếp chỉ diễn ra giữa các lớp liền kề. Điều này giúp quản lý sự phức tạp, tăng tính module hóa và dễ bảo trì.
  - *Đặc điểm:*
    - Mỗi lớp có trách nhiệm và chức năng rõ ràng.
    - Tương tác chỉ diễn ra giữa các lớp kề nhau.
    - Thay đổi một lớp ít ảnh hưởng đến các lớp khác.
  - *Ví dụ ứng dụng điển hình:* Mô hình tham chiếu liên kết các hệ thống mở OSI và TCP/IP trong mạng máy tính. Trong ứng dụng, máy tìm kiếm trên Internet với các tầng giao diện người dùng, xử lý nghiệp vụ và dữ liệu.
- Mô hình Đối tượng phân tán (Distributed Objects Model)
  - Mô tả: Trong mô hình này, toàn bộ hệ thống được xây dựng dựa trên ý tưởng các đối tượng có thể tồn tại và tương tác với nhau trên các máy tính khác nhau trong một mạng. Các đối tượng này có thể gọi các phương thức của nhau từ xa một cách minh bạch, giống như gọi một phương thức cục bộ.
  - Đặc điểm:
    - Minh bạch vị trí: Client không cần biết đối tượng nằm ở đâu.

        - Truyền thông dựa trên gọi phương thức từ xa (Remote Method Invocation - RMI) hoặc gọi thủ tục từ xa (Remote Procedure Call - RPC).
        - Sử dụng giao diện (interface) để định nghĩa các dịch vụ mà đối tượng cung cấp.
    - *Ví dụ ứng dụng điển hình:* CORBA (Common Object Request Broker Architecture): Một tiêu chuẩn cho phép các đối tượng viết bằng các ngôn ngữ lập trình khác nhau trên các nền tảng khác nhau có thể giao tiếp với nhau. Ví dụ, một ứng dụng Java có thể gọi một đối tượng C++ trên một máy tính khác thông qua CORBA.
- Mô hình Kênh sự kiện (Event-based/Publish-Subscribe Model)
    - *Mô tả:* Mô hình này tập trung vào việc giao tiếp thông qua các sự kiện. Các thành phần trong hệ thống (được gọi là "publishers") tạo ra và gửi các sự kiện mà không cần biết ai sẽ nhận chúng. Các thành phần khác (được gọi là "subscribers") đăng ký để nhận các sự kiện quan tâm. Một "broker" hoặc "message queue" thường đóng vai trò trung gian để định tuyến các sự kiện.
    - *Đặc điểm:*
        - Ghép nối lỏng lẻo (loose coupling) giữa các thành phần: Publishers và Subscribers không cần biết về nhau.
        - Bất đồng bộ: Các sự kiện được xử lý một cách bất đồng bộ, giúp tăng hiệu suất và khả năng mở rộng.
        - Khả năng mở rộng cao: Dễ dàng thêm publishers hoặc subscribers mà không ảnh hưởng đến hệ thống hiện có.
    - *Ví dụ ứng dụng điển hình:* IoT (Internet of Things): Các cảm biến (publishers) gửi dữ liệu nhiệt độ, độ ẩm, v.v., và các ứng dụng nhà thông minh (subscribers) nhận dữ liệu này để tự động điều khiển đèn, điều hòa.
- Mô hình Dữ liệu tập trung (Centralized Data/Shared Data Model)
    - *Mô tả:* Trong mô hình này, có một kho dữ liệu trung tâm được chia sẻ bởi tất cả các thành phần trong hệ thống phân tán. Các thành phần này tương tác với nhau bằng cách đọc và ghi vào kho dữ liệu chung. Đây có thể là một cơ sở dữ liệu tập trung, một hệ thống tệp dùng chung, hoặc một bảng thông báo (tuple space).
    - *Đặc điểm:*
        - Dữ liệu là điểm giao tiếp chính giữa các thành phần.
        - Đơn giản hóa việc chia sẻ thông tin.
        - Có thể gặp vấn đề về nút cổ chai và điểm lỗi duy nhất nếu kho dữ liệu không được thiết kế chịu lỗi tốt.

- *Ví dụ ứng dụng điển hình:* Ứng dụng ngân hàng truyền thống: Mặc dù có nhiều chi nhánh và máy ATM phân tán, nhưng tất cả dữ liệu giao dịch và thông tin tài khoản thường được lưu trữ và quản lý bởi một hệ thống cơ sở dữ liệu tập trung (hoặc một cụm CSDL được nhân rộng và quản lý chặt chẽ). Các ứng dụng ở chi nhánh hoặc ATM kết nối đến CSDL trung tâm để thực hiện giao dịch.

**** Nêu vai trò của phần mềm trung gian (middleware) trong kiến trúc khách-chủ phân tán, và liệt kê ba tính năng chính mà nó cung cấp.***

Trong kiến trúc khách-chủ (Client-Server) phân tán, phần mềm trung gian (middleware) đóng một vai trò cực kỳ quan trọng, là cầu nối giữa các ứng dụng khách và các dịch vụ máy chủ. Nó giúp che giấu sự phức tạp của môi trường phân tán, cho phép các ứng dụng giao tiếp và tương tác một cách hiệu quả và minh bạch. Middleware là xương sống của nhiều hệ thống phân tán, cho phép các ứng dụng hoạt động hiệu quả trong một môi trường phức tạp và không đồng nhất

Vai trò chính: Phần mềm trung gian hoạt động như một lớp trừu tượng giữa hệ điều hành và các ứng dụng, cung cấp một tập hợp các dịch vụ để hỗ trợ việc phát triển và triển khai các ứng dụng phân tán. Nó giúp các nhà phát triển tập trung vào logic nghiệp vụ của ứng dụng mà không cần phải xử lý các vấn đề phức tạp của hệ thống phân tán như:

- Minh bạch vị trí: Client không cần biết dịch vụ nằm ở đâu trên mạng.
- Minh bạch truy cập: Client gọi dịch vụ từ xa giống như gọi dịch vụ cục bộ.
- Quản lý sự không đồng nhất: Xử lý sự khác biệt về phần cứng, hệ điều hành, giao thức mạng và ngôn ngữ lập trình.
- Quản lý lỗi và độ tin cậy: Cung cấp cơ chế để xử lý lỗi mạng, lỗi máy chủ và đảm bảo tính tin cậy của giao tiếp.
- Quản lý tài nguyên và giao dịch: Giúp quản lý việc cấp phát tài nguyên và đảm bảo tính nguyên tử của các giao dịch phân tán.

Ba tính năng chính mà phần mềm trung gian cung cấp:

- *Thiết lập phiên làm việc giữa các tiến trình:* Middleware giúp quản lý các kết nối và phiên giao tiếp giữa các tiến trình máy khách và máy chủ. Nó giao tiếp với các tiến trình qua giao diện lập trình ứng dụng (API), thiết lập liên kết và quản lý việc gửi/nhận thông điệp.
- *Bảo mật dữ liệu:* Middleware cung cấp các tính năng bảo mật cần thiết cho việc trao đổi thông tin trong hệ thống phân tán. Điều này bao gồm việc kiểm tra bảo mật hệ thống và đảm bảo thông tin được an toàn khi truyền qua mạng.
- Xử lý lỗi và đồng bộ hóa: Middleware đảm bảo sự phối hợp và đồng bộ giữa các tiến trình trong hệ thống phân tán. Nó có khả năng giám sát các yêu cầu từ phía máy khách, xử lý tương tranh khi nhận được nhiều yêu cầu đồng thời, và quản lý việc trả

về kết quả cho máy khách. Middleware cũng đóng vai trò quan trọng trong việc xử lý các tình huống lỗi, giúp hệ thống phục hồi và duy trì hoạt động.

Câu 6:

- Phân loại ba loại dịch vụ trong SOA (cơ bản, tích hợp, quy trình) kèm ví dụ điển hình cho mỗi loại.

Trong Kiến trúc hướng dịch vụ (SOA), các dịch vụ được phân loại thành ba nhóm chính, dựa trên mức độ trừu tượng và vai trò của chúng trong quy trình nghiệp vụ:

- **Dịch vụ cơ bản (Basic Services):**
  - *Đặc điểm:* Là các dịch vụ cung cấp những tính năng kinh doanh cơ bản nhất và chưa được phân tách cho các dịch vụ khác. Chúng thường không trạng thái (stateless) và có thời gian chạy tương đối ngắn.
  - *Ví dụ:* Dịch vụ truy xuất dữ liệu (ví dụ: lấy thông tin khách hàng từ cơ sở dữ liệu), dịch vụ đăng nhập, hoặc dịch vụ thay đổi mật khẩu.
- **Dịch vụ tích hợp (Integration Services):**
  - *Đặc điểm:* Được cấu thành từ một hoặc nhiều dịch vụ cơ bản. Các dịch vụ này cũng thường không trạng thái và có thời gian thực hiện tương đối ngắn.
  - *Ví dụ:*
    - Xử lý thanh toán đơn hàng (kết hợp dịch vụ xác minh người dùng + trừ tiền tài khoản + ghi log giao dịch)
    - Đăng ký tài khoản kèm gửi email xác nhận
- **Dịch vụ quy trình (Process Services):**
  - *Đặc điểm:* Là các dịch vụ phản ánh một quy trình kinh doanh hoàn chỉnh, do đó có thời gian thực hiện khá dài và thuộc loại có trạng thái (stateful).
  - *Ví dụ:* Dịch vụ "Xử lý Đơn hàng", có thể bao gồm các bước như kiểm tra thông tin khách hàng, kiểm tra tồn kho, xử lý thanh toán và gửi thông báo xác nhận.

- Trình bày vòng đời của một dịch vụ SOA, từ giai đoạn phát triển đến vận hành sản xuất, và những thách thức chính ở mỗi giai đoạn.

Vòng đời của một dịch vụ SOA là một chu trình lặp lại, từ khi dịch vụ được hình thành ý tưởng cho đến khi nó không còn được sử dụng. Mỗi giai đoạn đều có những hoạt động và thách thức riêng:

a. Giai đoạn Phát triển (Development/Design & Build)

- *Mô tả:* Giai đoạn này bao gồm việc xác định nhu cầu kinh doanh, thiết kế kiến trúc dịch vụ (xác định các chức năng, giao diện, dữ liệu), phát triển mã nguồn cho dịch vụ và kiểm thử đơn vị.

- *Thách thức chính:*
    - Thiết kế dịch vụ phù hợp: Làm sao để định nghĩa các dịch vụ với mức độ trừu tượng và kích thước (granularity) đúng đắn, đảm bảo tính độc lập, khả năng tái sử dụng và phù hợp với yêu cầu nghiệp vụ.
    - Quản lý phiên bản (Versioning): Khi các yêu cầu thay đổi, làm thế nào để cập nhật dịch vụ mà không ảnh hưởng đến các ứng dụng hoặc dịch vụ khác đang sử dụng phiên bản cũ.
    - Đảm bảo chất lượng (Quality Assurance): Thiết kế và thực hiện các bộ kiểm thử tự động toàn diện để đảm bảo dịch vụ hoạt động đúng đắn và ổn định.
    - Cân bằng giữa tái sử dụng và đặc thù: Đảm bảo dịch vụ đủ tổng quát để tái sử dụng nhưng vẫn đáp ứng được các yêu cầu đặc thù của nghiệp vụ.

b. Giai đoạn Triển khai (Deployment)

- *Mô tả:* Sau khi dịch vụ được phát triển và kiểm thử nội bộ, nó được triển khai lên các môi trường khác nhau: môi trường kiểm thử tích hợp, môi trường tiền sản xuất (staging) và cuối cùng là môi trường sản xuất.
- *Thách thức chính:*
    - Phức tạp về cơ sở hạ tầng: Triển khai dịch vụ lên các máy chủ, cấu hình môi trường chạy, đảm bảo kết nối và khả năng giao tiếp giữa các dịch vụ.
    - Quản lý phụ thuộc (Dependency Management): Đảm bảo rằng tất cả các dịch vụ mà dịch vụ mới phụ thuộc vào đều đã được triển khai, sẵn sàng và tương thích về phiên bản.
    - Gián đoạn dịch vụ: Giảm thiểu thời gian ngừng hoạt động (downtime) trong quá trình triển khai, đặc biệt là trong môi trường sản xuất.
    - Tự động hóa triển khai: Phát triển các script hoặc công cụ để tự động hóa quá trình triển khai nhằm giảm lỗi thủ công và tăng tốc độ.

c. Giai đoạn Vận hành và Giám sát (Operation & Monitoring)

- *Mô tả:* Dịch vụ đang hoạt động trong môi trường sản xuất và cung cấp chức năng cho người dùng và các hệ thống khác. Giai đoạn này tập trung vào việc theo dõi hiệu suất, tài nguyên, phát hiện lỗi, và đảm bảo tính khả dụng.
- *Thách thức chính:*
    - Giám sát và cảnh báo: Thiết lập hệ thống giám sát toàn diện để theo dõi các chỉ số hiệu suất (latency, throughput), tài nguyên (CPU, RAM), và phát hiện lỗi, sau đó đưa ra cảnh báo kịp thời.
    - Khắc phục sự cố (Troubleshooting): Khi có vấn đề, nhanh chóng xác định nguyên nhân gốc rễ, đặc biệt đối với các luồng dịch vụ phức tạp đi qua nhiều

dịch vụ con.

- Quản lý tài nguyên và khả năng mở rộng: Đảm bảo dịch vụ có đủ tài nguyên để đáp ứng tải tăng lên và có thể mở rộng (scaling) một cách linh hoạt.
- An toàn và bảo mật: Liên tục giám sát và bảo vệ dịch vụ khỏi các cuộc tấn công, quản lý quyền truy cập và lỗ hổng bảo mật.

d. Giai đoạn Tối ưu hóa và Cải tiến (Optimization & Improvement)

- *Mô tả:* Dựa trên dữ liệu thu thập được từ giai đoạn giám sát, phản hồi từ người dùng và các yêu cầu kinh doanh mới, dịch vụ được phân tích để tìm ra các cơ hội cải thiện hiệu suất, độ tin cậy, khả năng mở rộng, hoặc bổ sung tính năng mới.
- *Thách thức chính:*
  - Xác định điểm nghẽn và cơ hội cải tiến: Phân tích dữ liệu lớn và các log để tìm ra nguyên nhân của hiệu suất kém hoặc các lĩnh vực cần tối ưu hóa.
  - Thực hiện thay đổi mà không gây gián đoạn: Đảm bảo các cải tiến được triển khai mà không ảnh hưởng đến hoạt động hiện tại của các ứng dụng hoặc dịch vụ phụ thuộc.
  - Đo lường hiệu quả: Đánh giá xem các cải tiến có mang lại lợi ích mong muốn hay không.

e. Giai đoạn Ngừng sử dụng (Retirement)

- *Mô tả:* Khi một dịch vụ không còn cần thiết, lỗi thời, hoặc đã được thay thế hoàn toàn bởi một dịch vụ mới, nó sẽ được loại bỏ khỏi hệ thống một cách có kế hoạch.
- *Thách thức chính:*
  - Xác định các bên phụ thuộc: Tìm ra tất cả các ứng dụng và dịch vụ khác vẫn đang sử dụng dịch vụ này để tránh gây hỏng hóc khi ngừng hoạt động.
  - Chuyển đổi các bên phụ thuộc: Hỗ trợ các hệ thống phụ thuộc chuyển đổi sang dịch vụ hoặc phiên bản mới.
  - Lập kế hoạch ngừng hoạt động: Thực hiện quá trình ngừng hoạt động một cách cẩn thận và có kiểm soát để tránh gây ra sự cố bất ngờ cho hệ thống tổng thể.
  - Lưu trữ dữ liệu: Đảm bảo dữ liệu liên quan đến dịch vụ được lưu trữ an toàn hoặc chuyển đổi sang các hệ thống khác nếu cần.