BỘ THÔNG TIN VÀ TRUYỀN THÔNG

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

# MÔN HỌC: CÁC HỆ THỐNG PHÂN TÁN


| | |
|---|---|
| **Giảng viên:** | **TS. Kim Ngọc Bách** |
| **Lớp:** | **Hệ Thống Thông Tin - M24CQHT02-B** |
| **Nhóm:** | **10** |
| **Học viên:** | **Hoàng Thanh Hằng- B24CHHT068** |
| | **Giáp Thị Huê - B24CHHT075** |
| | **Trần Văn Quyền - B24CHHT089** |
| | **Hoàng Phương Hoa - B24CHHT072** |


**Hà Nội, tháng 07,2025**

## Câu 1:

Nêu và giải thích hai đặc điểm quan trọng nhất của hệ thống phân tán.
Trình bày ba lý do cơ bản khiến các ứng dụng phân tán phức tạp hơn so với các ứng dụng đơn lẻ.

### Đặc điểm quan trọng nhất của hệ thống phân tán

Nhìn chung, các định nghĩa về hệ thống phân tán đều đề cập đến hai đặc điểm quan trọng. Đặc điểm thứ nhất là tập hợp các phần tử tính toán có khả năng hoạt động độc lập với nhau. Đặc điểm thứ hai là các phần tử này cần phải cộng tác để cùng giải quyết một nhiệm vụ chung. Hệ thống phân tán được thống nhất định nghĩa là hệ thống các phần mềm cài đặt trên các máy tính độc lập nhưng được phối hợp hoạt động với nhau như một thể thống nhất. Điều này làm cho người dùng có cảm giác như đang sử dụng một máy tính đơn lẻ, dù phần mềm ứng dụng trên các máy tính trao đổi thông tin với nhau qua mạng bằng cách chuyển thông điệp.

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

Việc xây dựng các ứng dụng phân tán phức tạp hơn đáng kể so với các ứng dụng chạy độc lập trên một máy tính. Dưới đây là ba lý do cơ bản cho sự phức tạp này:

- **Nhu cầu chia sẻ thông tin và tài nguyên**: Các ứng dụng phân tán phát sinh từ nhu cầu chia sẻ thông tin và tài nguyên giữa nhiều người dùng. Dữ liệu có thể được lưu trữ phân tán, dẫn đến các vấn đề phức tạp về phân quyền truy nhập, ví dụ như cho phép truy nhập dữ liệu từ xa nhưng không cho phép sao chép cục bộ.
- **Vấn đề hiệu năng**: Trong nhiều trường hợp, tính toán phân tán là bắt buộc để tận dụng khả năng tính toán song song hoặc khai thác khả năng của các máy tính chuyên dụng nhằm nâng cao hiệu năng hệ thống. Điều này đòi hỏi phải giải quyết các vấn đề liên quan đến quản lý các thành viên và giảm thiểu thời gian trễ do trao đổi thông tin qua môi trường mạng.
- **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, không làm ảnh hưởng đến hoạt động của toàn bộ hệ thống. Điều này đòi hỏi phải có các biện pháp phát hiện và xử lý lỗi kịp thời, bao gồm việc chuyển sang các máy tính dự phòng nếu máy chủ bị hỏng hoàn toàn để dịch vụ không bị gián đoạn.

## 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.
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?

### Các yếu tố ảnh hưởng đến hiệu năng hệ thống phân tán

Hiệu năng của hệ thống phân tán chịu ảnh hưởng từ nhiều yếu tố, bao gồm cả phần cứng và mạng:

- **Yếu tố phần cứng (CPU, bộ nhớ, kênh truyền)**: Tốc độ của bộ vi xử lý trung tâm, bộ nhớ và tốc độ kênh truyền trên bo mạch chủ là những yếu tố then chốt quyết định khả năng tính toán trên từng máy tính trong hệ thống. Trong kiến trúc truy cập bộ nhớ đồng nhất, hiệu năng hệ thống có thể bị hạn chế bởi băng thông của kênh truyền, khiến các đơn vị xử lý trung tâm lãng phí thời gian chờ để đọc/ghi bộ nhớ khi số lượng bộ vi xử lý tăng lên. Việc thêm vùng đệm hoặc bộ nhớ cục bộ riêng cho mỗi bộ vi xử lý có thể tăng hiệu năng sử dụng kênh truyền và mở rộng khả năng xử lý.
- **Yếu tố mạng (băng thông, topology)**: Trong hệ thống phân tán, việc trao đổi thông tin giữa các máy tính được thực hiện trên môi trường mạng, và đây là yếu tố không thể thiếu để thực hiện các nhiệm vụ tính toán.
    - **Băng thông**: Băng thông mạng không phải là vô hạn và dữ liệu di chuyển trên đó cần một khoảng thời gian nhất định, do đó việc giảm thiểu thời gian trễ là công việc quan trọng khi xây dựng các ứng dụng phân tán. Băng thông của các kênh kết nối trên mạng diện rộng (WAN) thường thấp hơn nhiều so với mạng cục bộ (LAN), và chi phí đóng gói/bóc tách dữ liệu cùng với chất lượng kênh truyền thấp làm tăng thời gian trễ.
    - **Topology (hình trạng mạng)**: Tốc độ truyền dữ liệu trong mạng cục bộ phụ thuộc vào hình trạng mạng. Trong mạng cục bộ, nếu sử dụng kênh truyền dạng bus hoặc thiết bị tập trung, sẽ hình thành vùng xung đột, chỉ cho phép một máy tính gửi dữ liệu tại mỗi thời điểm. Giải pháp tốt hơn là đấu nối các máy tính theo hình sao bằng cách sử dụng thiết bị chuyển mạch, khi đó mỗi cổng của bộ chuyển mạch là một vùng xung đột riêng, giúp sử dụng tối đa băng thông. Độ trễ của mạng, bao gồm trễ truyền dẫn và trễ xử lý trên các thiết bị mạng, chắc chắn sẽ cần một khoảng thời gian nhất định cho việc trao đổi thông tin giữa các máy tính.

**Sự khác biệt về yêu cầu quản lý tài nguyên giữa Hệ điều hành phân tán và Hệ điều hành mạng**

Hệ điều hành phân tán (DOS) và hệ điều hành mạng (NOS) có những yêu cầu khác nhau về quản lý tài nguyên do bản chất và mục tiêu hoạt động của chúng:

- **Hệ điều hành phân tán (Distributed OS)**:
    - **Mô tả và Liên kết**: Các máy tính trong hệ điều hành phân tán liên kết chặt chẽ với nhau, thường được sử dụng cho các hệ thống đồng nhất (ví dụ: cụm máy chủ). Chúng được cài đặt trên các máy tính khác nhau để tạo thành một cụm và vận hành như một máy tính có cấu hình mạnh hơn, với hoạt động gắn kết chặt chẽ để cung cấp các dịch vụ của hệ điều hành.
    - **Mục tiêu quản lý tài nguyên**: Mục tiêu chính của DOS là quản lý và che giấu các tài nguyên phần cứng trên các máy tính khác nhau. Điều này bao gồm việc nâng cao hiệu năng bằng cách phân rã yêu cầu tính toán thành các nhiệm vụ nhỏ hơn để xử lý trên nhiều máy tính (tăng tốc độ) và cân bằng tải (chuyển yêu cầu đến máy tính thích hợp). Đồng thời, DOS còn tăng độ

tin cậy của hệ thống thông qua cơ chế dự phòng nóng, đảm bảo hoạt động liên tục ngay cả khi một máy tính bị hỏng.

- **Hệ điều hành mạng (Network OS)**:
  - **Mô tả và Liên kết**: Liên kết giữa các máy tính trong hệ điều hành mạng tương đối lỏng lẻo, và chúng thường được sử dụng cho các hệ thống không đồng nhất. Mỗi máy tính tạo ra các dịch vụ cung cấp cho các máy tính khác. Các hệ điều hành như Microsoft Server, Linux, Unix được xếp vào loại hệ điều hành mạng.
  - **Mục tiêu quản lý tài nguyên**: Ngoài các chức năng cơ bản của một hệ điều hành, hệ điều hành mạng phải thực hiện việc chia sẻ và bảo vệ tài nguyên của mạng. Máy chủ quản lý tài nguyên của mình để cung cấp dịch vụ cho các máy khách. Mối quan hệ giữa các máy tính không chặt chẽ, và sự gắn kết trong hệ thống do các ứng dụng phân tán đảm nhiệm, với một số dịch vụ chung do hệ điều hành cung cấp cho các ứng dụng phân tán thông qua phần mềm trung gian.

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

### 1- Các loại hệ thống phân tán

Dựa trên các đặc điểm khác nhau, hệ thống phân tán có thể được chia thành ba loại chính: điện toán phân tán, thông tin phân tán và lan tỏa phân tán.

- **Hệ thống điện toán phân tán (Distributed Computing Systems)**: Loại hệ thống này tập trung vào việc đảm bảo hiệu năng cao cho các tác vụ tính toán chuyên sâu. Điểm cốt lõi là phân chia nhiệm vụ để xử lý trên nhiều máy tính.
  - **Điện toán cụm (Cluster Computing)**: Gồm các máy tính tương tự nhau về phần cứng và hệ điều hành, kết nối qua mạng cục bộ tốc độ cao. Mục tiêu là sử dụng kỹ thuật xử lý song song để tăng hiệu năng, tạo thành một máy tính ảo hiệu năng cao cho người dùng.
  - **Điện toán lưới (Grid Computing)**: Bao gồm các hệ thống phân tán thuộc nhiều miền quản lý khác nhau, thường không đồng nhất về phần cứng, phần mềm và hệ điều hành, có thể sử dụng các công nghệ mạng khác nhau. Vấn đề cốt lõi là quản lý tài nguyên từ các cơ quan khác nhau và cho phép cộng tác giữa các nhóm người dùng thông qua các "cơ quan ảo".
- **Hệ thống thông tin phân tán (Distributed Information Systems)**: Thông tin phân tán là hiện thực phổ biến, nơi dữ liệu được lưu trữ trên nhiều máy tính, và việc xử lý dữ liệu trên một máy có thể liên quan đến các máy khác. Nguyên nhân của việc lưu trữ dữ liệu phân tán có thể do dung lượng dữ liệu lớn hoặc để tăng hiệu năng và độ tin cậy thông qua kỹ thuật nhân bản. Các phần mềm hiện nay

thường được xây dựng theo mô hình ba bên, tách biệt các thành phần lưu trữ dữ liệu, xử lý nghiệp vụ và tiếp nhận yêu cầu.

- **Hệ thống lan tỏa phân tán (Pervasive Distributed Systems)**: Thuật ngữ này đề cập đến sự liên kết với môi trường vật lý và được coi là bước phát triển tiếp theo của hệ thống phân tán. Mục tiêu là tích hợp các thiết bị thông minh nhỏ vào môi trường vật lý, cho phép các thiết bị tự thay đổi hành vi và giao tiếp với nhau khi có tương tác của người dùng hoặc thay đổi trạng thái môi trường xung quanh. Các đặc điểm chính bao gồm nhận biết ngữ cảnh, tự thích nghi, tự chủ và khả năng tương tác tự phát. Ví dụ điển hình là nhà thông minh và hệ thống sản xuất thông minh.

**So sánh ba loại hệ thống phân tán:**

| Đặc điểm | Điện toán phân tán | Thông tin phân tán | Lan tỏa phân tán |
|---|---|---|---|
| **Mục tiêu chính** | Nâng cao hiệu năng tính toán cho các tác vụ phức tạp | Quản lý và truy cập dữ liệu phân tán hiệu quả | Tích hợp thiết bị thông minh vào môi trường vật lý, tự thích nghi |
| **Cấu trúc/Bản chất** | Tập trung vào chia sẻ tài nguyên tính toán (CPU) | Tập trung vào lưu trữ và quản lý dữ liệu phân tán | Tập trung vào tương tác với môi trường vật lý, thiết bị nhỏ |
| **Tính đồng nhất** | Cụm: đồng nhất; Lưới: không đồng nhất | Thường không đồng nhất (nhiều hệ thống DB) | Thường không đồng nhất (nhiều loại thiết bị) |
| **Đặc điểm nổi bật** | Xử lý song song, ảo hóa siêu máy tính | Sao chép dữ liệu, mô hình 3 lớp | Nhận biết ngữ cảnh, tự chủ, tương tác tự phát |

**2- Các lớp chính trong kiến trúc điện toán lưới**

Hệ thống điện toán lưới thường được xây dựng theo mô hình phân tầng, với mỗi tầng thực hiện những chức năng riêng biệt. Trong kiến trúc này, các lớp chính bao gồm:

- **Tầng ứng dụng (Application Layer)**: Tầng này bao gồm các ứng dụng vận hành bên trong các "cơ quan ảo" và sử dụng môi trường điện toán lưới để thực hiện các nhiệm vụ của mình. Đây là lớp cao nhất, nơi người dùng cuối tương tác với hệ thống.
- **Tầng trung gian (Middleware Layer)**: Trong các hệ thống điện toán lưới, tầng trung gian là nơi gộp các chức năng của tầng tiếp nhận, tầng kết nối và tầng tài nguyên. Nó có nhiệm vụ quản lý và cung cấp chức năng truy cập trong suốt đến tất cả các tài nguyên phân bố trên các trang mạng khác nhau.
    - **Tầng tiếp nhận (Collective Layer)**: Xử lý các yêu cầu truy cập đến nhiều tài nguyên khác nhau, cung cấp các chức năng như thăm dò, định vị, lập lịch truy cập và nhân bản tài nguyên. Các giao thức ở tầng này thường không chuẩn hóa để đảm bảo cung cấp dịch vụ theo yêu cầu của tầng ứng dụng.
    - **Tầng kết nối (Connection Layer)**: Bao gồm các giao thức truyền thông hỗ trợ các giao tác lưới, bao trùm toàn bộ các tài nguyên. Ví dụ là các giao

thức truy cập để di chuyển tài nguyên hoặc truy cập tài nguyên từ xa. Tầng này cũng bao gồm các giao thức bảo mật.
    - **Tầng tài nguyên (Resource Layer)**: Quản lý các tài nguyên đơn lẻ. Tầng này sử dụng các chức năng do tầng kết nối cung cấp và gọi trực tiếp các giao diện của tầng thực hiện (Fabric Layer) để thực hiện các chức năng điều khiển truy cập, ví dụ như thiết lập cấu hình tài nguyên hay khởi tạo tiến trình đọc/ghi dữ liệu.
- **Tầng thực hiện (Fabric Layer)**: Tầng này cung cấp giao diện để truy cập tài nguyên cục bộ. Các giao diện này được xây dựng để thích ứng với việc cho phép chia sẻ tài nguyên bên trong một cơ quan ảo. Tầng thực hiện thường cung cấp các chức năng để truy vấn trạng thái và khả năng của tài nguyên, cũng như các chức năng quản lý tài nguyên thực tế. Đây là lớp thấp nhất, tương tác trực tiếp với các tài nguyên vật lý.

## Câu 4

Giải thích tại sao "tính sẵn sàng" (availability) được xem là mục tiêu quan trọng nhất của hệ thống phân tán.
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.
Trình bày mối quan hệ giữa "tính mở" (openness) và khả năng tương tác (interoperability) trong hệ thống phân tán.

**Tính sẵn sàng (Availability) - Mục tiêu quan trọng nhất của hệ thống phân tán**
Tính sẵn sàng được xem là mục tiêu quan trọng nhất của hệ thống phân tán vì mục tiêu chính của hệ thống phân tán là kết nối người sử dụng với tài nguyên hệ thống, sao cho người sử dụng có thể tiếp cận tài nguyên tại bất kỳ thời điểm nào một cách dễ dàng nhất mà không phụ thuộc vị trí địa lý. Tài nguyên ở đây có thể là phần cứng (máy in, máy tính, thiết bị lưu trữ) hoặc dữ liệu cung cấp thông tin hữu ích. Hệ thống phân tán được thiết kế để đảm bảo rằng các dịch vụ luôn có sẵn cho người dùng, ngay cả khi có sự cố xảy ra trên một phần của hệ thống. Khả năng chịu lỗi, được thực hiện bằng cách phát hiện và xử lý lỗi kịp thời, hoặc chuyển sang các máy tính dự phòng khi máy chủ bị hỏng hoàn toàn, nhằm mục đích đảm bảo dịch vụ không bị gián đoạn. Mặc dù hiệu năng, tính linh hoạt, tính nhất quán, và bảo mật cũng là các mục tiêu quan trọng, nhưng tính sẵn sàng là nền tảng để các mục tiêu khác có ý nghĩa trong môi trường phân tán, nơi người dùng mong muốn truy cập liên tục vào các dịch vụ và tài nguyên.
**Ba hình thức "tính trong suốt" (Transparency)**
Tính trong suốt trong hệ thống phân tán là khả năng che giấu sự phức tạp của các thao tác xử lý bên trong và vị trí dữ liệu khi chúng được lưu trữ trên nhiều máy tính, giúp người dùng không cần quan tâm đến vị trí thực sự của tài nguyên và máy tính nào đang xử lý yêu cầu. Nếu hệ thống có thể tạo cảm giác cho người dùng như đang chạy trên một máy tính đơn lẻ thì hệ thống đó đáp ứng yêu cầu về tính trong suốt, góp phần tạo nên sự thân thiện và dễ sử dụng. Tổ chức tiêu chuẩn quốc tế đã đưa ra nhiều khía cạnh của tính trong suốt, trong đó có ba hình thức chính là:

1. **Trong suốt về truy nhập (Access Transparency)**:
    - **Giải thích**: Che giấu sự khác biệt trong việc thể hiện dữ liệu và cách thức người sử dụng có thể truy nhập đến tài nguyên. Ở mức cơ bản, cần che giấu sự khác biệt trong kiến trúc của các máy, nhưng quan trọng hơn là đạt được sự đồng thuận về cách thể hiện trên các nền tảng khác nhau.
    - **Ví dụ**: Trong một hệ thống phân tán không đồng nhất, nơi các máy tính cài đặt hệ điều hành khác nhau (ví dụ: Windows và Linux), quy tắc đặt tên cho các tập tin có thể khác nhau. Tính trong suốt về truy nhập đảm bảo rằng người dùng có thể truy cập một tập tin bằng cùng một cách đặt tên, không cần biết tập tin đó đang được lưu trữ trên máy Windows hay Linux.
2. **Trong suốt về vị trí (Location Transparency)**:
    - **Giải thích**: Che giấu nơi tài nguyên đang trú ngụ đối với người sử dụng. Điều này thường đạt được bằng cách gán các tên thân thiện và công khai cho tài nguyên, mà không tiết lộ vị trí vật lý của nó.
    - **Ví dụ**: Khi truy cập trang web www.ptit.edu.vn, người dùng không cần biết máy chủ vật lý đang lưu trữ trang web này được đặt ở đâu (ví dụ: Hà Nội, TP.HCM, hay nước ngoài). Tên miền www.ptit.edu.vn không chứa bất kỳ thông tin địa lý nào về máy chủ.
3. **Trong suốt về lỗi (Failure Transparency)**:
    - **Giải thích**: Che giấu lỗi và các vấn đề phục hồi sau khi lỗi xảy ra đối với người dùng. Hệ thống có khả năng tự xử lý các trường hợp lỗi, đảm bảo không mất mát thông tin và tự khởi tạo lại trạng thái tốt nhất.
    - **Ví dụ**: Khi một máy chủ cơ sở dữ liệu trong hệ thống phân tán bị lỗi, người dùng vẫn có thể tiếp tục thực hiện các thao tác truy vấn hoặc cập nhật dữ liệu mà không bị gián đoạn hoặc nhận được thông báo lỗi. Hệ thống tự động chuyển đổi sang một bản sao dự phòng của cơ sở dữ liệu và phục hồi dịch vụ mà người dùng không hề hay biết về sự cố đã xảy ra.

**Mối quan hệ giữa "tính mở" (Openness) và khả năng tương tác (Interoperability)**
Trong hệ thống phân tán, "tính mở" và "khả năng tương tác" có mối quan hệ chặt chẽ và bổ trợ lẫn nhau:

- **Tính mở (Openness)**: Tính mở trong hệ thống phân tán liên quan đến khả năng cho phép dễ dàng tích hợp và mở rộng hệ thống bằng cách thêm các thành phần mới hoặc thực hiện sửa đổi mà không ảnh hưởng đến các thành phần hiện có. Một hệ thống mở thường được phân tách thành các thành phần tương đối nhỏ, dễ thay thế và dễ thích nghi, thường được thiết kế dựa trên cách tiếp cận hướng dịch vụ. Điều này đòi hỏi các thành phần không chỉ cung cấp định nghĩa cho các giao diện mức cao nhất mà còn mô tả chính xác cách tương tác với nhau. Tính mở còn thể hiện ở việc các đặc tả phải đầy đủ và trung lập, không quy định cách cài đặt cụ thể cho mỗi giao diện, tạo điều kiện cho các nhà phát triển khác nhau có thể xây dựng và tùy biến dịch vụ.
- **Khả năng tương tác (Interoperability)**: Khả năng tương tác là một đặc tính quan trọng của tính mở, thể hiện việc các thành phần được cài đặt bởi những nhà sản xuất khác nhau có thể cùng làm việc với nhau chỉ dựa trên các dịch vụ đã được mô

tả trong các tiêu chuẩn chung. Để đạt được tính mở và khả năng tương tác, cần có các đặc tả chính xác, đầy đủ và trung lập.

**Mối quan hệ**: Tính mở là tiền đề để đạt được khả năng tương tác. Một hệ thống được thiết kế với tính mở, dựa trên các giao diện và giao thức chuẩn hóa, sẽ cho phép các thành phần từ nhiều nguồn khác nhau (thậm chí từ các nhà cung cấp khác nhau) có thể "nói chuyện" và làm việc cùng nhau một cách hiệu quả. Khả năng tương tác chính là minh chứng cụ thể cho việc tính mở đã được hiện thực hóa, giúp hệ thống không bị "khóa" vào một nhà cung cấp hoặc công nghệ cụ thể, từ đó tăng cường tính linh hoạt và khả năng mở rộng. Nói cách khác, tính mở là triết lý thiết kế, còn khả năng tương tác là kết quả của việc áp dụng triết lý đó, cho phép các hệ thống phân tán hoạt động như một thể thống nhất dù được xây dựng từ nhiều thành phần đa dạng.

## Câu 5

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

**So sánh ưu – nhược điểm của kiến trúc phân cấp và kiến trúc ngang hàng**

Hệ thống phân tán có thể được xây dựng dựa trên hai kiến trúc cơ bản: phân cấp và ngang hàng, mỗi loại có những ưu và nhược điểm riêng:

- **Kiến trúc phân cấp (Client-Server Architecture)**:
  - **Ưu điểm**: Đây là hình thức phân cấp phổ biến nhất, dễ cài đặt và quản lý do có sự phân tách rõ ràng vai trò giữa tiến trình khách (yêu cầu dịch vụ) và tiến trình chủ (cung cấp dịch vụ).
  - **Nhược điểm**: Hạn chế chính của kiến trúc này là khả năng chịu lỗi. Nếu máy chủ bị hỏng, dịch vụ sẽ bị gián đoạn hoàn toàn. Hơn nữa, hiệu năng xử lý bị giới hạn bởi khả năng của máy chủ, và khi số lượng truy cập tăng đột biến, toàn bộ hệ thống có thể bị ảnh hưởng nghiêm trọng. Để khắc phục, kiến trúc phân cấp có thể mở rộng thành nhiều lớp (ví dụ: ba lớp hoặc nhiều hơn), bổ sung tầng trung gian để xử lý nghiệp vụ, giảm tải cho cả khách và chủ.
- **Kiến trúc ngang hàng (Peer-to-Peer Architecture)**:
  - **Ưu điểm**: Trong kiến trúc này, các tiến trình có vai trò tương đương nhau, không phân biệt là khách hay chủ, và có thể hoạt động độc lập hoặc cộng tác. Đặc điểm nổi bật của kiến trúc ngang hàng là tính tin cậy cao hơn so với kiến trúc phân cấp; nếu một tiến trình bị hỏng, nó sẽ không làm ảnh hưởng đến hoạt động chung của toàn bộ hệ thống.
  - **Nhược điểm**: Hạn chế chính là kiến trúc ngang hàng thường khó cài đặt và quản lý hơn so với kiến trúc phân cấp do tính phân tán cao và không có điểm điều khiển trung tâm.

**Bốn mô hình hệ thống phân tán**

1. **Mô hình kiến trúc phân tầng (Layered Architecture)**:
    - **Giải thích**: Mô hình này được xây dựng dựa trên ý tưởng các tầng phía trên sẽ gọi dịch vụ của tầng phía dưới, và tầng phía dưới sẽ cung cấp dịch vụ cho tầng phía trên. Mỗi tầng có trách nhiệm riêng và tương tác với các tầng liền kề.
    - **Ví dụ ứng dụng**: Mô hình mạng TCP/IP là một ví dụ điển hình của kiến trúc phân tầng, với các lớp như vật lý, liên kết dữ liệu, mạng, truyền tải và ứng dụng, mỗi lớp cung cấp dịch vụ cho lớp trên và sử dụng dịch vụ của lớp dưới.
2. **Mô hình kiến trúc hướng đối tượng (Distributed Object Model)**:
    - **Giải thích**: Mô hình này xem xét hệ thống phân tán như một tập hợp các đối tượng độc lập và cộng tác với nhau để giải quyết một nhiệm vụ nào đó. Các đối tượng này có thể được phân tán trên các máy tính khác nhau và tương tác thông qua các giao diện đã định nghĩa.
    - **Ví dụ ứng dụng**: Các công nghệ như CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Model) và RMI (Remote Method Invocation) là những công nghệ điển hình được sử dụng để xây dựng các ứng dụng dựa trên mô hình đối tượng phân tán, cho phép các đối tượng phần mềm trên các máy khác nhau gọi phương thức của nhau.
3. **Mô hình kiến trúc dựa trên sự kiện (Event-based Model)**:
    - **Giải thích**: Đây là kiến trúc cho phép các tiến trình gửi thông báo (sự kiện) và các tiến trình khác sẽ tiếp nhận các thông báo này để xử lý. Các tiến trình không cần biết trực tiếp về nhau mà tương tác thông qua việc phát và nhận sự kiện.
    - **Ví dụ ứng dụng**: Một ứng dụng điển hình của mô hình dựa trên sự kiện là hệ thống bản tin (messaging/bulletin board system), nơi những người phát hành (publisher) gửi các thông báo hoặc bài viết, và những người đăng ký (subscriber) sẽ tiếp nhận các thông báo đó dựa trên sở thích của họ.
4. **Mô hình kiến trúc dựa trên tài nguyên (Resource-based Model)**:
    - **Giải thích**: Mô hình này xem xét tất cả các tài nguyên trong hệ thống là một đối tượng, các đối tượng này được định danh thống nhất thông qua các tài nguyên định vị (URL) và có thể được thao tác bằng các phương thức chuẩn (GET, POST, PUT, DELETE).
    - **Ví dụ ứng dụng**: Hệ thống web (World Wide Web) là một ví dụ điển hình của mô hình dựa trên tài nguyên, nơi các trang web, hình ảnh, video (tài nguyên) được định danh bằng URL và có thể được truy cập, hiển thị thông qua các giao thức chuẩn như HTTP.

**Vai trò và tính năng của phần mềm trung gian (Middleware) trong kiến trúc khách-chủ phân tán**

Trong kiến trúc khách-chủ phân tán, đặc biệt là trong các kiến trúc nhiều lớp, phần mềm trung gian đóng một vai trò quan trọng để giải quyết các hạn chế của kiến trúc hai lớp và tăng cường khả năng hoạt động của hệ thống.

- **Vai trò**: Phần mềm trung gian được bổ sung làm tầng giữa tầng khách và tầng chủ. Vai trò chính của nó là xử lý các nghiệp vụ, từ đó giảm tải cho cả tầng khách (ví dụ: máy tính cá nhân của người dùng) và tầng chủ (ví dụ: máy chủ cơ sở dữ liệu). Quan trọng hơn, phần mềm trung gian hỗ trợ kết nối các ứng dụng khác nhau qua môi trường mạng và che giấu các đặc điểm không đồng nhất của hệ điều hành và nền tảng phần cứng mà các máy khách và máy chủ đang chạy. Điều này giúp các ứng dụng phân tán có thể hoạt động liền mạch như một hệ thống thống nhất dù được xây dựng trên các công nghệ khác nhau.
- **Ba tính năng chính**: Phần mềm trung gian cung cấp nhiều dịch vụ để hỗ trợ việc phát triển và vận hành các ứng dụng phân tán. Ba tính năng chính bao gồm:
    1. **Dịch vụ đặt tên và thư mục**: Giúp các thành phần của hệ thống tìm thấy nhau trên mạng bằng cách cung cấp cơ chế đăng ký và tra cứu tên cho các dịch vụ và tài nguyên phân tán.
    2. **Dịch vụ an toàn**: Đảm bảo an ninh cho các giao tiếp và truy cập tài nguyên trong môi trường phân tán, bao gồm xác thực, ủy quyền và mã hóa.
    3. **Dịch vụ giao tác (Transaction Services)**: Hỗ trợ quản lý các giao tác phân tán, đảm bảo tính nhất quán của dữ liệu ngay cả khi giao tác liên quan đến nhiều máy chủ khác nhau, ví dụ thông qua cơ chế giao tác hai pha.

**<u>Câu 6:</u>**

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

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.

**Phân loại các loại dịch vụ trong SOA**

Kiến trúc hướng dịch vụ (SOA) tập trung vào quy trình nghiệp vụ và sử dụng các giao diện chuẩn để che giấu sự phức tạp bên trong, cho phép các ứng dụng được cấu trúc từ các thành phần dịch vụ hoạt động độc lập. Trong SOA, dịch vụ là những hàm chức năng thực hiện theo một quy trình nghiệp vụ nào đó. Có thể phân loại các dịch vụ trong SOA như sau:

1. **Dịch vụ cơ bản (Basic Services)**:
    - **Giải thích**: Đây là những dịch vụ kỹ thuật tái sử dụng, phục vụ cho các mục đích nghiệp vụ và có thể được tái sử dụng trong nhiều dòng dịch vụ khác nhau. Chúng thường là các chức năng cấp thấp, độc lập và cung cấp giao diện đơn giản để khuyến khích việc tái sử dụng.
    - **Ví dụ điển hình**: Các dịch vụ thuộc loại này có thể bao gồm dịch vụ truy xuất dữ liệu, dịch vụ đăng nhập, hoặc dịch vụ quản lý người dùng.

        - Dịch vụ mở tài khoản ngân hàng mới: Kết hợp các dịch vụ cơ bản như: "Tạo thông tin khách hàng", "Tạo tài khoản", "Gán tài khoản cho khách hàng", "Gửi thông báo thành công".
        - Dịch vụ xử lý đơn hàng: Kết hợp các dịch vụ cơ bản như: "Kiểm tra tồn kho", "Xác nhận thanh toán", "Tạo vận đơn", "Gửi thông báo vận chuyển".
        - Dịch vụ đăng ký khóa học: Kết hợp các dịch vụ như: "Kiểm tra đủ điều kiện", "Đăng ký khóa học", "Cập nhật thông tin học phí".

    -
2. **Dịch vụ quy trình (Process Services)**:
    - **Giải thích**: Các dịch vụ này được tổ chức thành tập hợp các dịch vụ hỗ trợ cho nghiệp vụ, nhằm mục đích phục vụ trực tiếp hay gián tiếp cho khách hàng thông qua hệ thống tự động. Chúng thường được định nghĩa thành các miền dịch vụ cụ thể của doanh nghiệp. SOA lấy dịch vụ làm trung tâm để xây dựng các ứng dụng và chú trọng đến quy trình nghiệp vụ.
    - **Ví dụ điển hình**: Các dịch vụ trong những miền dịch vụ như tài chính, bán hàng, quảng cáo, sản xuất, vận chuyển, kỹ thuật, quản lý lợi nhuận, và chăm sóc khách hàng. Các dịch vụ trong những miền khác nhau có thể cần các chính sách truyền dữ liệu khi yêu cầu đồng bộ hóa.

        - Dịch vụ mở tài khoản ngân hàng mới: Kết hợp các dịch vụ cơ bản như: "Tạo thông tin khách hàng", "Tạo tài khoản", "Gán tài khoản cho khách hàng", "Gửi thông báo thành công".
        - Dịch vụ xử lý đơn hàng: Kết hợp các dịch vụ cơ bản như: "Kiểm tra tồn kho", "Xác nhận thanh toán", "Tạo vận đơn", "Gửi thông báo vận chuyển".
        - Dịch vụ đăng ký khóa học: Kết hợp các dịch vụ như: "Kiểm tra đủ điều kiện", "Đăng ký khóa học", "Cập nhật thông tin học phí".

3. **Dịch vụ tích hợp (Integration Services)**:
    - **Giải thích**: SOA được sử dụng để tích hợp các ứng dụng và một trong những thách thức đặt ra cho các tổ chức công nghệ thông tin là xây dựng kiến trúc phần mềm có khả năng tích hợp các thành phần mới. Tính độc lập với nền tảng và khả năng giao tiếp thông qua thông điệp dựa trên các giao thức phổ biến (như HTTP, FTP, SMTP) là những yếu tố then chốt cho khả năng tích hợp của SOA. Các dịch vụ trong cùng một miền nên kết nối qua từ điển dữ liệu chung, còn các miền khác nhau có thể cần chính sách truyền dữ liệu để đồng bộ. Điều này cho thấy khả năng tích hợp là một đặc tính nội tại mà SOA mang lại thông qua các dịch vụ của mình, hơn là một loại dịch vụ riêng biệt.
    - **Ví dụ điển hình:**

- Dịch vụ mở tài khoản ngân hàng mới: Kết hợp các dịch vụ cơ bản như: "Tạo thông tin khách hàng", "Tạo tài khoản", "Gán tài khoản cho khách hàng", "Gửi thông báo thành công".
- Dịch vụ xử lý đơn hàng: Kết hợp các dịch vụ cơ bản như: "Kiểm tra tồn kho", "Xác nhận thanh toán", "Tạo vận đơn", "Gửi thông báo vận chuyển".
- Dịch vụ đăng ký khóa học: Kết hợp các dịch vụ như: "Kiểm tra đủ điều kiện", "Đăng ký khóa học", "Cập nhật thông tin học phí".

**Vòng đời của một dịch vụ SOA và những thách thức chính**

Vòng đời của một dịch vụ SOA trải qua nhiều giai đoạn, từ lúc lên ý tưởng cho đến khi dịch vụ được đưa vào vận hành và quản lý. Dưới đây là các giai đoạn chính và những thách thức đi kèm:

**1. Giai đoạn Phát triển (Development)**

- **Mô tả:** Giai đoạn này bao gồm việc thiết kế, cài đặt và kiểm thử dịch vụ. Các nhà phát triển sẽ xác định chức năng, giao diện, các giao thức truyền thông và công nghệ sử dụng.
- **Các bước chính:**
  - **Xác định yêu cầu (Requirements Definition):** Thu thập và phân tích các yêu cầu nghiệp vụ để xác định chức năng mà dịch vụ cần cung cấp.
  - **Thiết kế (Design):** Thiết kế kiến trúc dịch vụ, giao diện (WSDL, REST API), luồng dữ liệu, và các thành phần bên trong.
  - **Cài đặt (Implementation):** Viết mã, xây dựng dịch vụ dựa trên thiết kế.
  - **Kiểm thử đơn vị và tích hợp (Unit and Integration Testing):** Đảm bảo từng thành phần và sự tương tác giữa chúng hoạt động đúng đắn.
- **Thách thức chính:**
  - **Xác định ranh giới dịch vụ:** Quyết định mức độ trừu tượng và chức năng của dịch vụ để đảm bảo khả năng tái sử dụng và quản lý.
  - **Thiết kế giao diện:** Đảm bảo giao diện dịch vụ rõ ràng, nhất quán, dễ sử dụng và có thể mở rộng.
  - **Quản lý sự phụ thuộc:** Giảm thiểu sự phụ thuộc giữa các dịch vụ để tăng tính độc lập.
  - **Kiểm soát phiên bản:** Quản lý các phiên bản dịch vụ khác nhau khi có sự thay đổi.
  - **Khả năng mở rộng và hiệu suất:** Thiết kế để dịch vụ có thể chịu tải cao và hoạt động hiệu quả.

**2. Giai đoạn Triển khai (Deployment)**

- **Mô tả:** Sau khi dịch vụ được phát triển và kiểm thử, nó sẽ được triển khai lên môi trường thử nghiệm và sau đó là môi trường sản xuất.
- **Các bước chính:**
  - **Đóng gói (Packaging):** Đóng gói dịch vụ thành các gói triển khai.
  - **Cấu hình (Configuration):** Cấu hình các tham số môi trường, kết nối cơ sở dữ liệu, bảo mật, v.v.

    - **Triển khai lên máy chủ (Deployment to Servers):** Đặt dịch vụ lên các máy chủ ứng dụng hoặc nền tảng dịch vụ.
- **Thách thức chính:**
    - **Quản lý môi trường:** Đảm bảo tính nhất quán giữa các môi trường phát triển, thử nghiệm và sản xuất.
    - **Tự động hóa triển khai:** Giảm thiểu lỗi thủ công và tăng tốc độ triển khai.
    - **Quản lý cấu hình:** Đảm bảo các cấu hình đúng đắn và an toàn cho từng môi trường.
    - **Khả năng tương thích:** Đảm bảo dịch vụ tương thích với hạ tầng và các hệ thống hiện có.

**3. Giai đoạn Kiểm thử và Xác nhận (Testing and Validation)**

- **Mô tả:** Dịch vụ được kiểm tra kỹ lưỡng trong môi trường gần giống với sản xuất để đảm bảo chất lượng, hiệu suất và độ tin cậy trước khi đưa vào sử dụng thực tế.
- **Các bước chính:**
    - **Kiểm thử chức năng (Functional Testing):** Đảm bảo dịch vụ thực hiện đúng các chức năng theo yêu cầu.
    - **Kiểm thử phi chức năng (Non-functional Testing):** Bao gồm kiểm thử hiệu suất, tải, khả năng chịu lỗi, bảo mật.
    - **Kiểm thử hồi quy (Regression Testing):** Đảm bảo các thay đổi không gây ra lỗi ở các chức năng đã có.
    - **Kiểm thử chấp nhận người dùng (User Acceptance Testing - UAT):** Người dùng nghiệp vụ kiểm tra và xác nhận dịch vụ đáp ứng nhu cầu của họ.
- **Thách thức chính:**
    - **Môi trường kiểm thử thực tế:** Khó khăn trong việc tạo ra môi trường kiểm thử giống hệt môi trường sản xuất.
    - **Dữ liệu kiểm thử:** Cần dữ liệu kiểm thử đầy đủ, đa dạng và không làm lộ thông tin nhạy cảm.
    - **Tích hợp với hệ thống khác:** Kiểm thử sự tương tác với các dịch vụ và hệ thống bên ngoài.
    - **Thời gian và chi phí:** Kiểm thử toàn diện có thể tốn kém và mất nhiều thời gian.

**4. Giai đoạn Đăng ký và Tìm kiếm (Registration and Discovery)**

- **Mô tả:** Sau khi dịch vụ sẵn sàng, nó được đăng ký vào một kho lưu trữ dịch vụ (Service Registry) hoặc danh mục dịch vụ (Service Catalog) để các ứng dụng hoặc dịch vụ khác có thể tìm kiếm và sử dụng.
- **Các bước chính:**
    - **Đăng ký (Registration):** Đăng ký siêu dữ liệu của dịch vụ (ví dụ: WSDL, REST API definition, mô tả, chính sách) vào Service Registry.
    - **Tìm kiếm (Discovery):** Các ứng dụng hoặc dịch vụ khách hàng có thể tìm kiếm dịch vụ dựa trên các tiêu chí cụ thể.
- **Thách thức chính:**

- **Quản lý kho lưu trữ dịch vụ:** Đảm bảo kho lưu trữ được cập nhật, dễ tìm kiếm và đáng tin cậy.
- **Mô tả dịch vụ:** Tạo các mô tả dịch vụ rõ ràng, đầy đủ và nhất quán để dễ dàng tìm kiếm và hiểu.
- **Quản lý phiên bản:** Đảm bảo các phiên bản dịch vụ khác nhau được quản lý đúng cách trong kho lưu trữ.

**5. Giai đoạn Vận hành Sản xuất (Production Operation)**

- **Mô tả:** Dịch vụ được đưa vào sử dụng thực tế và được quản lý, giám sát liên tục để đảm bảo hoạt động ổn định và hiệu quả.
- **Các bước chính:**
  - **Giám sát (Monitoring):** Theo dõi hiệu suất, tính sẵn sàng, lỗi và các chỉ số quan trọng khác của dịch vụ.
  - **Quản lý lỗi (Error Management):** Xử lý và khắc phục các lỗi phát sinh.
  - **Quản lý hiệu suất (Performance Management):** Tối ưu hóa để đảm bảo dịch vụ đáp ứng yêu cầu về hiệu suất.
  - **Quản lý bảo mật (Security Management):** Đảm bảo dịch vụ an toàn trước các mối đe dọa.
  - **Quản lý thay đổi (Change Management):** Xử lý các yêu cầu thay đổi, nâng cấp dịch vụ.
- **Thách thức chính:**
  - **Khả năng mở rộng động:** Dịch vụ cần có khả năng mở rộng (scale up/out) linh hoạt để đáp ứng tải thay đổi.
  - **Quản lý SLA (Service Level Agreement):** Đảm bảo dịch vụ đáp ứng các cam kết về thời gian hoạt động, hiệu suất.
  - **Xử lý sự cố:** Nhanh chóng xác định và khắc phục các sự cố để giảm thiểu thời gian ngừng hoạt động.
  - **Bảo mật:** Bảo vệ dịch vụ khỏi các cuộc tấn công và truy cập trái phép.
  - **Quản lý phiên bản và khả năng tương thích ngược:** Đảm bảo các phiên bản mới của dịch vụ không phá vỡ các ứng dụng sử dụng phiên bản cũ.

**6. Giai đoạn Ngừng sử dụng (Decommissioning)**

- **Mô tả:** Khi một dịch vụ không còn cần thiết hoặc được thay thế bằng một dịch vụ mới, nó cần được ngừng sử dụng một cách có kiểm soát.
- **Các bước chính:**
  - **Thông báo (Notification):** Thông báo cho các ứng dụng và dịch vụ phụ thuộc biết về việc ngừng sử dụng.
  - **Chuyển đổi (Migration):** Hỗ trợ các ứng dụng chuyển đổi sang dịch vụ thay thế (nếu có).
  - **Gỡ bỏ (Removal):** Gỡ bỏ dịch vụ khỏi môi trường sản xuất và kho lưu trữ.
  - **Lưu trữ (Archiving):** Lưu trữ các dữ liệu và nhật ký liên quan.
- **Thách thức chính:**
  - **Xác định các bên phụ thuộc:** Tìm ra tất cả các ứng dụng và dịch vụ đang sử dụng dịch vụ để tránh gây gián đoạn.

- **Thời gian chuyển đổi:** Đảm bảo đủ thời gian cho các bên liên quan để chuyển đổi.
- **Tác động ngược:** Tránh gây ra lỗi hoặc sự cố cho các hệ thống khác khi ngừng sử dụng.