**Nhóm 7**

* Lê Đình Hiếu – B24CHHT070
* Nguyễn Đăng Khoa – B24CHHT080
* Hoàng Tiến Quyết – B24CHHT090

**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.**

* Hai đặc điểm quan trọng nhất của hệ thống phân tán:
* **Các phần tử tính toán có thể hoạt động độc lập:**
  + Các thành phần của hệ thống phân tán (như máy tính, thiết bị) được cài đặt phần mềm riêng, có thể **vận hành một cách tự trị**, tức là không cần phụ thuộc vào các thành phần khác để thực hiện các tác vụ cục bộ của mình.
* **Các phần tử phải phối hợp để giải quyết nhiệm vụ chung:**
  + Mặc dù độc lập, các thành phần này cần **phối hợp thông qua trao đổi thông điệp qua mạng** để thực hiện những chức năng tổng thể như một hệ thống thống nhất. Người dùng sẽ có cảm giác như đang sử dụng một máy tính đơn lẻ.

→ Đây là bản chất của hệ thống phân tán: **độc lập về phần cứng, nhưng thống nhất trong hành vi ứng dụng**.

* **Ba lý do cơ bản khiến các ứng dụng phân tán phức tạp hơn các ứng dụng đơn lẻ:**
* **Chia sẻ thông tin và tài nguyên:**
  + Các máy tính trong hệ phân tán thường **lưu trữ và xử lý dữ liệu ở nhiều nơi khác nhau**, dẫn đến các vấn đề như: **phân quyền truy cập, bảo mật, giới hạn sao chép, đồng bộ hóa dữ liệu từ xa**… điều này không có trong ứng dụng đơn lẻ.
* **Hiệu năng tính toán và xử lý song song:**
  + Hệ thống phân tán cần **tận dụng sức mạnh tính toán song song** từ nhiều thiết bị khác nhau. Việc điều phối và tối ưu hóa phân chia tác vụ khiến việc xây dựng ứng dụng trở nên phức tạp hơn rất nhiều so với chạy đơn lẻ trên một máy.
* **Yêu cầu về khả năng chịu lỗi:**
  + Một hệ phân tán phải đảm bảo **hoạt động liên tục ngay cả khi một số thành phần bị lỗi hoặc hỏng hóc**. Việc phát hiện lỗi, chuyển đổi nhiệm vụ sang máy khác, đảm bảo dữ liệu và dịch vụ không bị gián đoạn là thách thức lớn khi xây dựng hệ thống loại này.

**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?**

**Yếu tố phần cứng:**

* **CPU (Bộ xử lý trung tâm):**
* Trong hệ thống phân tán, mỗi máy tính thường có một hoặc nhiều CPU. Các CPU này đóng vai trò như các nút tính toán trong hệ thống.
* **Số lượng và kiến trúc CPU** ảnh hưởng trực tiếp đến khả năng xử lý song song, tốc độ thực hiện các tác vụ và năng lực mở rộng hệ thống.
* Hệ thống **đa vi xử lý** (multi-processor) có thể nâng cao hiệu năng, nhưng nếu không có cơ chế truy cập bộ nhớ hiệu quả (như dùng bộ nhớ đệm hoặc bộ nhớ riêng), CPU có thể bị **lãng phí thời gian chờ** khi truy xuất bộ nhớ chung.
* **Bộ nhớ:**
* **Bộ nhớ dùng chung (shared memory)** giữa các CPU trong cùng một máy cần có băng thông cao, nếu không sẽ gây tắc nghẽn khi nhiều CPU cùng truy cập.
* **Bộ nhớ đệm và bộ nhớ riêng** giúp giảm xung đột truy cập và nâng cao hiệu suất bằng cách lưu trữ dữ liệu cục bộ cho từng CPU.
* Trong hệ thống phân tán, mỗi máy có bộ nhớ riêng nên **không thể chia sẻ dữ liệu qua biến toàn cục**, điều này làm tăng độ phức tạp khi cần đồng bộ hóa thông tin giữa các nút.
* **Kênh truyền (bus):**
* Trong hệ thống nhiều bộ xử lý (nhiều CPU trong một máy), các CPU chia sẻ một hoặc nhiều **kênh truyền vật lý trên bo mạch chủ** để truy cập bộ nhớ.
* **Băng thông của kênh truyền** là yếu tố then chốt ảnh hưởng đến hiệu năng: nếu quá nhiều CPU dùng chung một bus, sẽ xảy ra tình trạng nghẽn và chờ đợi, làm giảm tốc độ xử lý toàn hệ thống.

**Yếu tố mạng:**

* **Băng thông:**
* Trong hệ thống phân tán nhiều máy tính, **việc truyền thông tin giữa các nút diễn ra qua mạng**, nên **băng thông mạng càng cao thì khả năng trao đổi thông tin càng nhanh**.
* **Mạng cục bộ (LAN)** có băng thông cao hơn mạng diện rộng (WAN), do đó hiệu năng trong các cụm máy tính tại chỗ thường tốt hơn.
* Khi **băng thông hạn chế**, tốc độ trao đổi dữ liệu chậm, thời gian phản hồi kéo dài và dễ xảy ra tắc nghẽn truyền thông, làm giảm hiệu năng chung.
* **Topology (hình trạng mạng):**
* **Topology mạng** ảnh hưởng đến khả năng truyền dữ liệu giữa các nút.
* **Mô hình bus hoặc hub** dễ xảy ra xung đột vì các máy phải chia sẻ cùng một kênh vật lý – tại một thời điểm chỉ có một máy được gửi dữ liệu.
* **Mô hình sao sử dụng switch** tốt hơn vì mỗi cổng là một vùng xung đột riêng, giúp tăng băng thông hiệu dụng.
* Trong mạng diện rộng, **dữ liệu phải đi qua nhiều thiết bị trung gian**, gây ra **độ trễ lớn và không ổn định**, ảnh hưởng tiêu cực đến hiệu năng của hệ thống phân tán.

**Lý do 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 phân tán (Distributed OS):**
* **Môi trường:** Các máy tính **liên kết chặt chẽ**, thường là **hệ thống đồng nhất** (cùng cấu hình, cùng hệ điều hành, cùng phần mềm nền).
* **Mục tiêu:**
  + **Quản lý toàn bộ tài nguyên phần cứng và phần mềm trên tất cả các máy tính** trong hệ thống như thể đây là một **máy tính thống nhất duy nhất**.
  + **Che giấu sự phân tán** khỏi người dùng và ứng dụng – người dùng có cảm giác như chỉ đang dùng một máy tính mạnh duy nhất.
* **Yêu cầu quản lý tài nguyên:**
  + Cần **phân phối, điều phối và cân bằng tải tài nguyên giữa các nút**.
  + **Tự động phát hiện lỗi, phục hồi, đồng bộ và dự phòng nóng**.
  + Cần đảm bảo các tiến trình hoạt động trên nhiều máy có thể **chia sẻ tài nguyên một cách minh bạch và hiệu quả**.
* **Hệ điều hành mạng (Network OS):**
* **Môi trường:** Các máy tính **liên kết lỏng lẻo**, thường là **hệ thống không đồng nhất** (khác cấu hình, khác hệ điều hành).
* **Mục tiêu:**
  + Mỗi máy chủ yếu **quản lý tài nguyên của chính nó** và **cung cấp dịch vụ qua mạng cho máy khác truy cập từ xa**.
  + Không cố gắng che giấu sự phân tán mà **cho phép truy cập từ xa có kiểm soát**.
* **Yêu cầu quản lý tài nguyên:**
  + Tập trung vào **chia sẻ, bảo vệ và kiểm soát truy cập tài nguyên** trên từng máy.
  + Không yêu cầu đồng bộ toàn hệ thống, mà để **ứng dụng phân tán tự gắn kết và phối hợp**.
  + Quản lý tài nguyên theo kiểu **máy khách – máy chủ**, mỗi máy tự quản lý và cung cấp dịch vụ khi được yêu cầu.

**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.**

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

* **Hệ thống điện toán phân tán (Distributed Computing Systems):**
* **Khái niệm:**
* Hệ thống điện toán phân tán là một tập hợp các máy tính độc lập (nodes) được kết nối với nhau qua mạng, cùng phối hợp thực hiện các tác vụ điện toán một cách đồng thời hoặc song song. Mặc dù các thành phần phần cứng và phần mềm nằm rải rác ở nhiều nơi khác nhau, người dùng vẫn có thể nhìn nhận hệ thống như một thể thống nhất, giống như một máy tính đơn lẻ mạnh mẽ.
* Mục tiêu chính của hệ thống này là chia nhỏ một bài toán phức tạp thành nhiều phần nhỏ, giao cho nhiều máy xử lý song song để tăng hiệu suất xử lý, tiết kiệm thời gian và tài nguyên. Ngoài ra, hệ thống còn hỗ trợ tính mở rộng cao, tức là có thể dễ dàng thêm hoặc bớt các nút mà không ảnh hưởng đến toàn hệ thống.
* **Đặc điểm:**
* Tập trung vào chia sẻ và quản lý dữ liệu.
* Gồm các thành phần như cơ sở dữ liệu phân tán, dịch vụ web, middleware.
* Nhấn mạnh vào tính nhất quán và độ tin cậy của dữ liệu.
* **Ví dụ:**
* Hệ thống ngân hàng, nơi thông tin tài khoản được lưu tại nhiều chi nhánh.
* Hệ thống thương mại điện tử có nhiều cơ sở dữ liệu ở các quốc gia khác nhau.
* **Hệ thống thông tin phân tán (Distributed Information Systems):**
* **Khái niệm:**
* Hệ thống thông tin phân tán là loại hệ thống trong đó các tài nguyên thông tin (dữ liệu, tài liệu, cơ sở dữ liệu, dịch vụ xử lý) được lưu trữ và quản lý tại nhiều địa điểm khác nhau, nhưng vẫn hoạt động đồng bộ để phục vụ người dùng một cách thống nhất. Người dùng tương tác với hệ thống thông qua các giao diện hoặc ứng dụng trung gian mà không cần biết dữ liệu đang nằm ở đâu.
* Điểm nổi bật của hệ thống này là khả năng chia sẻ và tích hợp dữ liệu một cách hiệu quả trong môi trường phân tán, đảm bảo tính sẵn sàng, toàn vẹn, bảo mật và nhất quán. Các hệ thống này thường ứng dụng trong lĩnh vực tài chính, thương mại điện tử, logistics, chính phủ điện tử và các tổ chức có mạng lưới rộng lớn.
* **Đặc điểm:**
* Tập trung vào chia sẻ và quản lý dữ liệu.
* Gồm các thành phần như cơ sở dữ liệu phân tán, dịch vụ web, middleware.
* Nhấn mạnh vào tính nhất quán và độ tin cậy của dữ liệu.
* **Ví dụ:**
* Hệ thống ngân hàng, nơi thông tin tài khoản được lưu tại nhiều chi nhánh.
* Hệ thống thương mại điện tử có nhiều cơ sở dữ liệu ở các quốc gia khác nhau.
* **Hệ thống lan tỏa phân tán (Pervasive Distributed Systems):**
* **Khái niệm:**
* Hệ thống lan tỏa phân tán là một mô hình hiện đại của hệ thống phân tán, trong đó các thiết bị điện tử và cảm biến được gắn vào môi trường sống, làm việc hoặc di chuyển của con người một cách gần như vô hình. Các thiết bị này có khả năng giao tiếp, thu thập dữ liệu, xử lý ngữ cảnh và phản hồi một cách thông minh mà không cần sự can thiệp trực tiếp của người dùng.
* Đây là bước tiến của điện toán nhằm tạo ra một môi trường "thông minh", nơi các thiết bị được tích hợp chặt chẽ vào cuộc sống hàng ngày, từ nhà ở, xe hơi, đến cơ sở hạ tầng đô thị. Hệ thống lan tỏa phân tán giúp tăng cường khả năng tương tác giữa con người và công nghệ một cách tự nhiên, liên tục và thích nghi với ngữ cảnh cụ thể.
* **Đặc điểm:**
* Tập trung vào tính “ẩn” và trải nghiệm người dùng liền mạch.
* Gồm các thiết bị cảm biến, IoT, thiết bị đeo, thiết bị nhà thông minh.
* Tự động thu thập, xử lý và phản hồi dữ liệu theo ngữ cảnh.
* **Ví dụ:**
* Hệ thống nhà thông minh (smart home).
* Mạng cảm biến trong nông nghiệp, môi trường hoặc y tế (health monitoring).

**Bảng so sánh**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí** | **Điện toán phân tán** | **Thông tin phân tán** | **Lan tỏa phân tán** |
| **Mục tiêu chính** | Tăng hiệu năng xử lý | Quản lý và truy cập  dữ liệu phân tán | Tính ẩn, tiện lợi và trải  nghiệm người dùng |
| **Yếu tố trọng tâm** | Tính toán và xử lý  song song | Nhất quán dữ liệu, dịch  vụ phân tán | Tính thích ứng, cảm  ngữ cảnh |
| **Thiết bị tham gia** | Máy chủ, máy tính hiệu năng cao | Máy chủ, cơ sở dữ liệu, API | Thiết bị IoT, cảm biến,  thiết bị di động |
| **Tính chất** | Phối hợp xử lý một  bài toán lớn | Cộng tác thông tin từ nhiều nguồn | Tự động, ngữ cảnh, thân  thiện người dùng |
| **Ví dụ phổ biến** | Hadoop, Spark, Cluster Systems | Hệ thống ngân hàng, ERP phân tán | Smart Home,  Healthcare IoT |

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

**Application**

* **Khái niệm và chức năng:**
* Tầng ứng dụng bao gồm các ứng dụng vận hành bên trong cơ quan ảo và sử dụng môi trường điện toán lưới. Tầng tiếp nhận xử lý các yêu cầu truy nhập đến nhiều tài nguyên khác nhau, thường cung cấp các chức năng như: thăm dò, định vị, lập lịch truy nhập , nhân bản tài nguyên,... Các giao thức thuộc tầng này khá nhiều, để đảm bảo cung cấp dịch vụ theo yêu cầu của tầng ứng dụng nên chúng thường không phải là những giao thức đã được chuẩn hóa**.**
* **Vai trò:**
* **C**ung cấp giao diện người dùng (UI): Lớp này thường cung cấp giao diện trực quan, giúp người dùng dễ dàng thao tác, thiết lập thông số và nhận kết quả mà không cần biết đến các chi tiết kỹ thuật bên dưới.
* Tối ưu hóa hiệu suất: Ứng dụng cần được thiết kế sao cho có thể phân chia công việc, chạy song song và khai thác được tối đa sức mạnh tính toán của hệ thống lưới.
* Tích hợp dịch vụ: Ứng dụng trong lớp này có thể gọi đến nhiều dịch vụ từ lớp middleware để thực hiện các tác vụ như gửi yêu cầu tính toán, truy xuất dữ liệu, hoặc giám sát tài nguyên.
* **Middleware**
* **Khái niệm và chức năng:**
* Tầng kết nối bao gồm các giao thức truyền thông để hỗ trợ cho các giao tác lưới bao trùm toàn bộ các tài nguyên, ví dụ các giao thức truy nhập để di chuyển tài nguyên hoặc đơn giản chỉ là truy nhập tài nguyên từ một vị trí nào đó. Tầng kết nối sẽ phải bao gồm các các giao thức bảo mật, tính năng bảo mật có thể cho một tài khoản và cũng có thể cho một ứng dụng.
* **Vai trò:**
* Trừu tượng hóa tài nguyên: Middleware giúp ứng dụng "nhìn thấy" một hệ thống đồng nhất thay vì nhiều máy chủ rải rác. Ví dụ, các tài nguyên CPU, bộ nhớ hay dữ liệu sẽ được thể hiện dưới dạng dịch vụ thống nhất.
* Lập lịch và phân phối tác vụ: Middleware có nhiệm vụ phân tích yêu cầu của ứng dụng, lập lịch thực thi (scheduling), và phân phối công việc đến các nút phù hợp.
* Bảo mật và xác thực: Quản lý người dùng, phân quyền truy cập, mã hóa dữ liệu và giám sát các hoạt động nhằm đảm bảo tính bảo mật của toàn hệ thống.
* Phục hồi lỗi và giám sát: Middleware phát hiện lỗi (node hỏng, mất kết nối) và có cơ chế phục hồi tự động. Đồng thời cung cấp khả năng giám sát tình trạng hệ thống và tài nguyên.
* **Resource**
* **Khái niệm và chức năng:**
* Tầng tài nguyên quản lý tài nguyên đơn lẻ, nó sử dụng các chức năng do tầng kết nối cung cấp và gọi trực tiếp các giao diện tầng kết cấu cung cấp để thực hiện các chức năng điều khiển truy nhập, ví dụ các chức năng thiết lập cấu hình tài nguyên, khởi tạo tiến trình đọc/ghi dữ liệu. Tầng thực hiện cung cấp giao diện để truy nhập tài nguyên cục bộ, các giao diện này được xây dựng để có thể thích ứng với việc cho phép chia sẻ tài nguyên bên trong một cơ quan ảo, nó thường cung cấp các chức năng để truy vấn trạng thái và khả năng của tài nguyên, các chức năng quản lý tài nguyên thực.
* **Vai trò:**
* Cung cấp sức mạnh tính toán: Đây là nơi thực sự thực hiện các công việc xử lý dữ liệu, chạy chương trình hoặc lưu trữ thông tin mà các lớp trên yêu cầu.
* Quản lý tài nguyên cục bộ: Mỗi tài nguyên được quản lý bởi hệ điều hành hoặc phần mềm riêng của nó. Các hệ thống này cần phải "mở" và cho phép middleware truy cập theo các chuẩn đã định.
* Tham gia vào môi trường lưới: Dù tài nguyên thuộc nhiều tổ chức khác nhau, lớp này cho phép chia sẻ một cách an toàn, minh bạch và hiệu quả thông qua các chính sách liên kết và cơ chế kiểm soát truy cập.

**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.**

1. Tính sẵn sàng (availability) được xem là mục tiêu quan trọng nhất của hệ thống phân tán vì nó ảnh hưởng trực tiếp đến khả năng hoạt động liên tục và hiệu quả của hệ thống, cũng như trải nghiệm của người dùng. Khi hệ thống có tính sẵn sàng cao, người dùng có thể truy cập và sử dụng dịch vụ mọi lúc, mọi nơi, giảm thiểu thời gian chết và đảm bảo hoạt động kinh doanh diễn ra suôn sẻ:

* **Không gián đoạn dịch vụ:**

Hệ thống phân tán, đặc biệt là các hệ thống quan trọng như ngân hàng, thương mại điện tử, hoặc các dịch vụ trực tuyến, cần phải đảm bảo hoạt động liên tục để phục vụ người dùng. Tính sẵn sàng cao giúp hệ thống tránh được các sự cố, lỗi phần cứng, phần mềm, hoặc các cuộc tấn công từ chối dịch vụ, từ đó duy trì hoạt động ổn định.

* **Trải nghiệm người dùng tốt:**

Khi hệ thống có tính sẵn sàng cao, người dùng có thể truy cập và sử dụng dịch vụ một cách liền mạch, không gặp phải tình trạng gián đoạn, lỗi hoặc thời gian chờ đợi lâu. Điều này tạo ra trải nghiệm tích cực cho người dùng, góp phần nâng cao sự hài lòng và lòng trung thành của họ.

* **Giảm thiểu thiệt hại kinh tế:**

Sự cố hoặc gián đoạn hệ thống có thể gây ra thiệt hại kinh tế đáng kể cho các doanh nghiệp, đặc biệt là các doanh nghiệp phụ thuộc vào hệ thống trực tuyến. Tính sẵn sàng cao giúp giảm thiểu rủi ro này bằng cách đảm bảo hệ thống hoạt động liên tục và phục hồi nhanh chóng sau sự cố.

* **Cạnh tranh trên thị trường:**

Trong môi trường cạnh tranh, các doanh nghiệp cần đảm bảo hệ thống của mình có tính sẵn sàng cao để cung cấp dịch vụ tốt hơn so với đối thủ cạnh tranh. Khả năng cung cấp dịch vụ liên tục, ổn định và chất lượng cao là một yếu tố quan trọng để thu hút và giữ chân khách hàng.

1. **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.**

* **Tính trong suốt về truy nhập (Access Transparency)**
* **Khái niệm:** Tính trong suốt về truy nhập đề cập đến khả năng giúp người dùng truy cập tài nguyên từ xa giống như đang truy cập tài nguyên cục bộ, mà không cần quan tâm đến sự khác biệt trong cơ chế truy cập, giao thức, hay vị trí vật lý.
* **Vai trò:**
* Ẩn đi sự khác biệt giữa các loại tài nguyên (file, máy in, dịch vụ web,...).
* Cho phép người dùng sử dụng hệ thống phân tán một cách liền mạch, dễ dàng như khi làm việc với một máy tính cá nhân đơn lẻ.
* **Ví dụ minh họa:** Người dùng mở một file được lưu trữ trên máy chủ mạng qua File Explorer như thể nó nằm trong ổ D của máy mình, mặc dù thực tế file đó được truy xuất qua mạng thông qua giao thức SMB.
* **Tính trong suốt về vị trí (Location Transparency)**
* **Khái niệm:** Tính trong suốt về vị trí có nghĩa là người dùng hoặc chương trình không cần biết tài nguyên đang ở đâu trong hệ thống – địa chỉ thực tế của tài nguyên (trên máy chủ nào, khu vực nào) được hệ thống xử lý và ẩn đi.
* **Vai trò:**
* Cho phép di chuyển tài nguyên mà không ảnh hưởng đến cách người dùng truy cập.
* Giúp cân bằng tải, tối ưu hiệu suất và độ tin cậy của hệ thống phân tán.
* **Ví dụ minh họa:** Người dùng truy cập website thông qua một URL như http://example.com. Họ không cần biết trang web đang được phục vụ từ máy chủ ở Việt Nam, Singapore hay Mỹ – hệ thống DNS và hạ tầng mạng sẽ xử lý chi tiết đó.
* **Tính trong suốt về lỗi (Failure Transparency)**
* **Khái niệm:** Tính trong suốt về lỗi là khả năng của hệ thống giấu đi hoặc tự động xử lý các lỗi phát sinh trong quá trình hoạt động, chẳng hạn như lỗi mạng, lỗi phần cứng hoặc một node bị sập, để người dùng không bị ảnh hưởng hoặc ít bị gián đoạn nhất.
* **Vai trò:**
* Tăng tính tin cậy và khả năng phục hồi của hệ thống.
* Cho phép hệ thống tự động tái khởi động tác vụ, chuyển hướng truy cập, hoặc thay thế tài nguyên khi phát hiện lỗi.
* **Ví dụ minh họa:** Một người đang xem video trực tuyến (streaming), và máy chủ chính đột ngột bị lỗi. Hệ thống ngay lập tức chuyển sang máy chủ dự phòng mà người dùng không nhận ra bất kỳ sự gián đoạn nào.

**Bảng so sánh**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí** | **Truy nhập** | **Vị trí** | **Lỗi** |
| **Mục tiêu**  **chính** | Giúp người dùng không cần biết cách truy cập | Giúp người dùng không  cần biết tài nguyên ở đâu | Giúp người dùng không  cảm nhận được lỗi |
| **Ẩn điều gì?** | Phương thức truy cập  (API, giao thức) | Địa chỉ/tên máy chủ chứa tài nguyên | Sự cố, hỏng hóc của  hệ thống |
| **Lợi ích**  **mang lại** | Dễ dùng, nhất quán  khi làm việc | Di chuyển và phân phối  linh hoạt | Tăng tính ổn định và tin  cậy |
| **Ví dụ**  **đơn giản** | Mở file mạng như file  cục bộ | Truy cập web mà không  biết máy chủ ở đâu | Xem video không gián  đoạn khi máy chủ lỗi |

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

* **Tính mở (Openness):** Tính mở trong hệ thống phân tán được hiểu là khả năng mở rộng, sửa đổi, tích hợp và tương thích với các thành phần khác nhau một cách dễ dàng, thông qua việc sử dụng các tiêu chuẩn công khai. Một hệ thống mở là hệ thống không bị "đóng khung" vào một nhà cung cấp cụ thể hoặc một nền tảng cụ thể, mà có thể tương tác với bên thứ ba nhờ vào các chuẩn mực rõ ràng về giao tiếp, giao thức, định dạng dữ liệu, v.v.
* **Đặc điểm của một hệ thống mở:**
* Sử dụng các giao thức chuẩn hóa (như HTTP, REST, SOAP, TCP/IP).
* Hỗ trợ API công khai, cho phép tích hợp và mở rộng chức năng.
* Có khả năng kết nối với các hệ thống, nền tảng và phần mềm khác một cách dễ dàng.
* Khuyến khích phân tán và tái sử dụng phần mềm thông qua các mô-đun độc lập.
* **Khả năng tương tác (Interoperability):** Khả năng tương tác là mức độ mà các hệ thống, phần mềm hoặc thành phần riêng biệt có thể giao tiếp, hiểu và xử lý thông tin của nhau một cách hiệu quả, bất kể chúng được phát triển bởi các nhà cung cấp khác nhau hay chạy trên các nền tảng khác nhau.
* Khả năng tương tác không chỉ đơn thuần là việc “kết nối được”, mà còn bao gồm:
* Hiểu đúng ý nghĩa dữ liệu (semantic interoperability).
* Đồng thuận về quy trình xử lý (process interoperability).
* Truyền tải và tiếp nhận dữ liệu thành công (syntactic interoperability).
* **Mối quan hệ giữa tính mở và khả năng tương tác**
* Tính mở và khả năng tương tác có mối quan hệ nhân quả hai chiều, trong đó tính mở là điều kiện tiên quyết và cơ sở kỹ thuật quan trọng để đạt được khả năng tương tác trong hệ thống phân tán.
* Tính mở tạo tiền đề cho khả năng tương tác
* Khi một hệ thống được thiết kế theo hướng mở – sử dụng các giao thức chuẩn hóa, định dạng dữ liệu phổ biến (như JSON, XML), và kiến trúc API rõ ràng – nó sẽ dễ dàng tương tác với các hệ thống bên ngoài.
* Ví dụ, một dịch vụ web tuân thủ chuẩn RESTful có thể dễ dàng được gọi từ các hệ thống viết bằng các ngôn ngữ khác nhau như Java, Python, PHP,… mà không cần cấu hình đặc biệt.
* Khả năng tương tác là minh chứng cho tính mở
* Nếu một hệ thống không thể kết nối hay giao tiếp hiệu quả với các hệ thống khác, điều đó cho thấy kiến trúc của nó có thể thiếu tính mở, bị ràng buộc bởi công nghệ đóng, phụ thuộc vào một nhà cung cấp hoặc có giao diện độc quyền.
* Cả hai cùng hướng tới mục tiêu linh hoạt và mở rộng
* Trong môi trường điện toán hiện đại – đặc biệt là điện toán đám mây, điện toán biên và Internet of Things (IoT) – nhu cầu tích hợp các hệ thống dị biệt là rất lớn. Tính mở giúp các thành phần dễ dàng được thay thế, cập nhật hoặc mở rộng, còn khả năng tương tác giúp các thành phần đó hoạt động trơn tru với nhau như một thể thống nhất.
* Tính mở và khả năng tương tác không thể tách rời trong hệ thống phân tán hiện đại. Tính mở cung cấp nền tảng cho các hệ thống dễ dàng kết nối, tích hợp và mở rộng, trong khi khả năng tương tác bảo đảm rằng các thành phần đa dạng có thể hợp tác hiệu quả để cung cấp dịch vụ toàn diện. Trong một thế giới công nghệ ngày càng đa dạng và liên kết chặt chẽ, việc thiết kế hệ thống phân tán theo hướng mở và tương tác cao không chỉ là xu thế mà còn là yêu cầu bắt buộc để đảm bảo hiệu quả và khả năng phát triển bền vững.

**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)** | **Kiến trúc ngang hàng (Peer-to-Peer - P2P)** |
| --- | --- | --- |
| **Ưu điểm** | - Quản lý tập trung, dễ kiểm soát - Dễ bảo trì, nâng cấp hệ thống - Phân quyền rõ ràng | - Khả năng mở rộng tốt - Phân phối tài nguyên đều - Không phụ thuộc vào máy chủ trung tâm |
| **Nhược điểm** | - Dễ bị nghẽn tại nút trung tâm - Không linh hoạt khi mở rộng - Nếu nút cao bị lỗi, ảnh hưởng toàn hệ thống | - Khó kiểm soát, bảo mật - Việc đồng bộ và tìm kiếm tài nguyên phức tạp hơn - Có thể tiêu tốn băng thông nếu không tối ưu |

**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 hình phân tầng chia hệ thống thành các tầng (layers) riêng biệt, mỗi tầng đảm nhận một chức năng cụ thể và chỉ tương tác với tầng liền kề. Cách tổ chức này giúp tăng tính module, dễ bảo trì và mở rộng hệ thống
* **Ví dụ ứng dụng điển hình:**  
  Hệ thống web ba tầng (3-tier architecture):
* **Tầng 1**: Trình duyệt web của người dùng (giao diện người dùng – UI)
* **Tầng 2**: Máy chủ ứng dụng (xử lý logic nghiệp vụ)
* **Tầng 3**: Máy chủ cơ sở dữ liệu (lưu trữ và truy xuất dữ liệu)
* Ứng dụng thực tế: các nền tảng như **Shopee**, **Lazada**, hay **Facebook** đều sử dụng kiến trúc phân tầng để đảm bảo hiệu quả và khả năng mở rộng.
* **Mô hình đối tượng phân tán (Distributed Object Model)**
* Mô hình này cho phép các tiến trình trong hệ thống giao tiếp với nhau thông qua việc gọi các phương thức trên các đối tượng từ xa (remote objects), giống như khi làm việc với đối tượng cục bộ. Các công nghệ như **Java RMI**, **CORBA**, hoặc **.NET Remoting** được sử dụng phổ biến trong mô hình này
* **Ví dụ ứng dụng điển hình:**
* **Java RMI** trong hệ thống quản lý ngân hàng: các máy trạm tại chi nhánh ngân hàng gọi các phương thức từ xa trên máy chủ trung tâm để xử lý giao dịch, truy xuất thông tin tài khoản, chuyển tiền nội bộ.
* **CORBA** trong hệ thống hàng không: các thành phần đặt vé, thanh toán, tra cứu thông tin chuyến bay được triển khai phân tán nhưng liên kết chặt chẽ qua các đối tượng từ xa.
* **Mô hình kênh sự kiện (Event-based Model)**
* Mô hình này dựa trên cơ chế xử lý sự kiện: các thành phần trong hệ thống có thể phát ra hoặc lắng nghe các sự kiện mà không cần liên kết trực tiếp với nhau. Điều này giúp tăng tính linh hoạt và giảm sự phụ thuộc giữa các thành phần.
* **Ví dụ ứng dụng điển hình:**

**Hệ thống thông báo thời gian thực trong mạng xã hội (như Facebook)**: khi một người dùng đăng bài, bạn bè của họ nhận được thông báo thông qua cơ chế sự kiện.

* **Ứng dụng chat như Zalo hoặc Slack**: các tin nhắn được xử lý và phân phối theo mô hình publish-subscribe giữa các client và server thông qua một message broker như RabbitMQ hoặc Kafka.
* **Hệ thống giám sát thiết bị IoT**: các cảm biến phát sự kiện khi nhiệt độ thay đổi, hệ thống trung tâm lắng nghe và xử lý phản ứng.
* **Mô hình dữ liệu tập trung (Data-Centered Model)**
* Mô hình dữ liệu tập trung xây dựng hệ thống xung quanh một kho dữ liệu trung tâm. Các thành phần khác của hệ thống (các client) thực hiện các thao tác đọc/ghi dữ liệu từ kho trung tâm này. Mô hình phù hợp với các ứng dụng yêu cầu sự nhất quán và quản lý dữ liệu tập trung.
* **Ví dụ ứng dụng điển hình:**
* **Google Drive**: người dùng từ các thiết bị khác nhau truy cập và đồng bộ dữ liệu lên cùng một hệ thống lưu trữ trung tâm.
* **Hệ thống thư viện điện tử**: nhiều người dùng truy cập vào cơ sở dữ liệu tập trung để tìm kiếm, mượn hoặc tải tài liệu.
* **GitHub (trong mô hình centralized)**: các lập trình viên đẩy mã nguồn lên kho chính, mọi người làm việc trên bản trung tâm trước khi phân nhánh hoặc hợp nhất.
* **Mô hình kênh sự kiện (event-based model)** cho phép các thành phần trong hệ thống giao tiếp thông qua việc phát và nhận sự kiện. Các tiến trình trong mô hình này không cần biết trực tiếp đến nhau mà thông qua các kênh trung gian. Kiểu kiến trúc này phù hợp với các hệ thống phản ứng thời gian thực như phần mềm chat, các ứng dụng mạng xã hội, hay hệ thống giám sát cảnh báo sớm, nơi dữ liệu được xử lý theo luồng sự kiện.
* **Mô hình dữ liệu tập trung (data-centered model)** tổ chức hệ thống xung quanh một kho dữ liệu trung tâm. Các thành phần còn lại hoạt động như các client, tương tác với dữ liệu qua các thao tác truy vấn, cập nhật hoặc đồng bộ. Mô hình này phù hợp với các ứng dụng quản lý dữ liệu lớn như hệ thống thư viện điện tử, kho lưu trữ học liệu, hoặc các dịch vụ như Dropbox, Google Drive, nơi dữ liệu được lưu trữ và đồng bộ qua một hệ thống trung tâm.

**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 một hệ thống phân tán kiểu **khách – chủ** (client – server), các máy khách và máy chủ thường nằm ở các vị trí địa lý khác nhau, kết nối với nhau qua mạng (như Internet). Do đó, việc các thành phần này **giao tiếp, chia sẻ dữ liệu và phối hợp xử lý** sẽ phức tạp, liên quan đến nhiều vấn đề kỹ thuật như kết nối mạng, định danh, bảo mật, truyền dữ liệu...

**Phần mềm trung gian (middleware)** chính là **“cầu nối”** giúp các thành phần này **liên kết với nhau một cách dễ dàng và hiệu quả hơn**. Nó ẩn đi các chi tiết kỹ thuật phức tạp, cung cấp các công cụ và dịch vụ để các ứng dụng khách – chủ **có thể hoạt động trơn tru như thể chúng đang nằm cùng một chỗ**.

Nói đơn giản, phần mềm trung gian giúp: Máy khách không cần biết chính xác máy chủ nằm ở đâu, hoạt động thế nào.

* Máy chủ và máy khách có thể **trao đổi dữ liệu, gọi chức năng lẫn nhau một cách minh bạch**.
* Hệ thống dễ mở rộng, bảo trì và xử lý lỗi hơn trong môi trường phân tán.

**Ba tính năng chính của phần mềm trung gian**

## **Tính minh bạch (Transparency):** Phần mềm trung gian che giấu cho lập trình viên và người dùng các yếu tố phức tạp trong hệ thống phân tán như:

## Vị trí của các máy trong mạng (minh bạch về vị trí)

## Sự khác nhau giữa các hệ điều hành hay nền tảng (minh bạch về truy cập)

## Việc xử lý dữ liệu theo thời gian thực hay không (minh bạch về đồng bộ)

## Sự cố mạng hoặc lỗi hệ thống (minh bạch về lỗi)

Nhờ đó, lập trình viên có thể lập trình như thể tất cả thành phần đều nằm trên cùng một máy.

* **Giao tiếp giữa các thành phần (Communication):** Middleware cung cấp các cơ chế để:
* Truyền nhận dữ liệu giữa máy khách và máy chủ
* Gọi các phương thức từ xa (Remote Procedure Call – RPC)
* Hỗ trợ các hình thức truyền thông: đồng bộ (đợi phản hồi), bất đồng bộ (gửi xong không cần chờ)

Nói cách khác, middleware **giúp các thành phần nói chuyện được với nhau**, dù chúng ở xa hay dùng công nghệ khác nhau.

### ****Quản lý dịch vụ (Service Management):**** Middleware còn chịu trách nhiệm quản lý các vấn đề kỹ thuật trong hệ thống, như:

* Định danh và tìm kiếm dịch vụ (ví dụ: biết máy chủ nào xử lý việc gì)
* Bảo mật: phân quyền truy cập, mã hóa dữ liệu
* Phát hiện lỗi, ghi log, và tự động khôi phục nếu có sự cố
* Cân bằng tải: phân phối yêu cầu từ nhiều client đến nhiều server phù hợp

Nhờ những tính năng này, hệ thống phân tán có thể hoạt động ổn định và hiệu quả hơn rất nhiều.

**Câu 6:**

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

* **Dịch vụ cơ bản (Basic Services)**
* **Khái niệm:** Đây là những dịch vụ thực hiện các chức năng đơn lẻ, độc lập, thường là các thao tác trực tiếp với dữ liệu như truy xuất, thêm, xóa, hoặc cập nhật. Chúng là những "viên gạch" nền tảng của SOA.
* **Đặc điểm:**
* Có thể tái sử dụng nhiều nơi
* Ít phụ thuộc vào dịch vụ khác
* Dễ triển khai và kiểm thử
* **Ví dụ điển hình:**
* Dịch vụ lấy thông tin sinh viên từ cơ sở dữ liệu: getStudentByID()
* Dịch vụ kiểm tra tồn kho sản phẩm trong hệ thống bán hàng: checkInventory()
* Dịch vụ xác thực tài khoản người dùng: authenticateUser()
* **Dịch vụ tích hợp (Composite/Integration Services)**
* **Khái niệm:** Dịch vụ tích hợp là các dịch vụ có nhiệm vụ **tập hợp và phối hợp** nhiều dịch vụ cơ bản khác để cung cấp một chức năng phức tạp hơn cho hệ thống. Đây là "cầu nối" giữa các dịch vụ riêng lẻ.
* **Đặc điểm:**
* Gồm nhiều dịch vụ cơ bản hoặc kết hợp với các hệ thống bên ngoài
* Xử lý logic trung gian, có thể thực hiện tính toán, xử lý dữ liệu, chuyển đổi định dạng
* Tăng khả năng tái sử dụng ở mức cao hơn
* **Ví dụ điển hình:**
* Dịch vụ placeOrder() trong hệ thống thương mại điện tử: kết hợp các dịch vụ checkInventory(), calculateShipping(), processPayment() và sendConfirmationEmail().
* Dịch vụ getStudentTranscript(): tổng hợp điểm từ nhiều học kỳ, các môn học, từ các hệ thống quản lý khác nhau.
* **Dịch vụ quy trình (Process/Business Services)**
* **Khái niệm:** Đây là dịch vụ quản lý toàn bộ **luồng xử lý nghiệp vụ**, được xây dựng từ các dịch vụ tích hợp và cơ bản, sắp xếp theo một trình tự logic để hoàn thành một quy trình hoàn chỉnh. Có thể dùng công nghệ như **BPEL (Business Process Execution Language)** để mô hình hóa.
* **Đặc điểm:**
* Thể hiện logic nghiệp vụ ở mức cao
* Có thể gồm điều kiện rẽ nhánh, vòng lặp, xử lý lỗi
* Gần với góc nhìn người dùng/doanh nghiệp
* **Ví dụ điển hình:**
* Quy trình **xử lý đơn vay ngân hàng**: kiểm tra thông tin khách hàng → kiểm tra tín dụng → đánh giá khoản vay → phê duyệt → giải ngân.
* Quy trình **đăng ký môn học trực tuyến**: xác thực sinh viên → kiểm tra số tín chỉ → hiển thị môn học phù hợp → cho phép đăng ký → 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 trong SOA (Service-Oriented Architecture) bao gồm nhiều giai đoạn, từ khi dịch vụ được thiết kế và phát triển, đến khi được triển khai, vận hành, và bảo trì trong môi trường thực tế. Mỗi giai đoạn đóng vai trò quan trọng và đều có những thách thức riêng cần giải quyết.

* **Giai đoạn phân tích và thiết kế**
* Mục tiêu:
* Xác định nhu cầu nghiệp vụ cần được số hóa dưới dạng dịch vụ.
* Phân tích các yêu cầu chức năng và phi chức năng.
* Thiết kế cấu trúc dịch vụ, giao diện (API), định dạng dữ liệu (XML, JSON…), và các ràng buộc.

#### Thách thức:

#### Hiểu không đúng hoặc không đầy đủ yêu cầu nghiệp vụ.

#### Khó xác định mức độ tái sử dụng của dịch vụ (có nên chia nhỏ không?).

#### Thiếu tiêu chuẩn chung khi thiết kế dịch vụ (dẫn đến khó tích hợp sau này).

* **Giai đoạn phát triển (Develop)**

#### Mục tiêu:

#### Lập trình và xây dựng dịch vụ theo bản thiết kế.

#### Đảm bảo giao tiếp tuân theo chuẩn SOA (thường dùng SOAP hoặc REST).

#### Kiểm thử đơn vị (unit test) cho các chức năng cơ bản.

#### Thách thức:

#### Đồng bộ giữa nhóm phát triển và nhóm tích hợp dịch vụ.

#### Phát triển không tuân thủ chặt chẽ các quy tắc thiết kế dịch vụ (dẫn đến khó bảo trì).

#### Thiếu công cụ hỗ trợ kiểm thử toàn diện các dịch vụ phân tán.

* **Giai đoạn triển khai và tích hợp (Deploy & Integrate)**
* Mục tiêu:
* Đưa dịch vụ lên môi trường triển khai (dev/test/prod).
* Đăng ký dịch vụ vào **Service Registry** để các hệ thống khác có thể phát hiện và sử dụng.
* Tích hợp dịch vụ với các thành phần khác của hệ thống.

#### Thách thức:

#### Khó khăn trong việc tích hợp với các hệ thống cũ (legacy systems).

#### Vấn đề tương thích giao diện dịch vụ giữa các nhóm phát triển khác nhau

#### Rủi ro an ninh nếu triển khai mà không cấu hình đúng về xác thực và phân quyền.

* **Giai đoạn vận hành và giám sát (Operate & Monitor)**
* Mục tiêu:
* Đảm bảo dịch vụ chạy ổn định, hiệu suất cao
* Giám sát log, cảnh báo lỗi, theo dõi trạng thái dịch vụ.
* Phân tích dữ liệu sử dụng để tối ưu dịch vụ trong tương lai.

#### Thách thức:

#### Khó phát hiện nguyên nhân lỗi trong môi trường phân tán.

#### Cần có hệ thống giám sát chuyên biệt cho dịch vụ (như Prometheus, ELK stack).

#### Việc thay đổi cấu hình dịch vụ trong môi trường sản xuất tiềm ẩn nhiều rủi ro nếu không được kiểm soát chặt.

* **Giai đoạn bảo trì và tái sử dụng (Maintain & Reuse)**
* Mục tiêu:
* Cập nhật, sửa lỗi, hoặc nâng cấp chức năng dịch vụ.
* Tái sử dụng dịch vụ ở các quy trình nghiệp vụ hoặc hệ thống khác.

#### Thách thức:

#### Dịch vụ cập nhật có thể ảnh hưởng đến các hệ thống đang dùng nếu không có versioning.

#### Khó kiểm soát vòng đời các phiên bản dịch vụ nếu không có chiến lược quản lý tốt.

#### Việc loại bỏ (deprecate) dịch vụ cũ cần thông báo và chuyển đổi phù hợp.