Họ tên sinh viên:
Ngô Tiến Đạt – B24CHHT060
Vũ Ngọc Dương – B24CHHT065
Nguyễn Thanh Tùng – B24CHHT096

Môn học: Các hệ thống phân tán
Lớp: M24CQHT02-B

Bài làm

Câu 1:

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

a. Tính minh bạch (Transparency):

- Định nghĩa: Tính minh bạch đề cập đến khả năng ẩn giấu sự phức tạp của hệ thống phân tán với người dùng cuối, khiến họ cảm nhận như đang làm việc trên một hệ thống duy nhất, mặc dù thực tế là nhiều hệ thống cùng phối hợp.
- Các dạng minh bạch:
  - *Minh bạch truy cập:* Truy cập tài nguyên từ xa như thể tài nguyên đó đang ở cục bộ.
  - *Minh bạch vị trí:* Không cần biết tài nguyên được lưu trữ ở đâu.
  - *Minh bạch lỗi:* Hệ thống có thể tiếp tục hoạt động khi có sự cố xảy ra.

b. Tính chịu lỗi (Fault Tolerance):

- Định nghĩa: Là khả năng tiếp tục vận hành một cách chính xác ngay cả khi một số thành phần trong hệ thống bị lỗi.
- Cách hiện thực: Nhờ vào cơ chế phát hiện lỗi, sao lưu dữ liệu (replication), khôi phục tự động và phân phối nhiệm vụ sang các node khác.

2. Ba lý do khiến ứng dụng phân tán phức tạp hơn ứng dụng đơn lẻ:

a. Sự không đồng bộ và độ trễ mạng:

- Trong hệ thống phân tán, các thành phần giao tiếp với nhau qua mạng, vốn có thể chậm trễ, mất gói, hoặc không đảm bảo thứ tự. Điều này dẫn đến khó khăn trong việc đồng bộ và duy trì trạng thái nhất quán giữa các node.

b. Khó khăn trong xử lý lỗi:

- Các lỗi như mất kết nối mạng, node ngừng phản hồi, hoặc mất dữ liệu giữa chừng rất khó phát hiện và xử lý triệt để. Hơn nữa, phân biệt lỗi thực sự với độ trễ là một thách thức lớn.

c. Vấn đề bảo mật và kiểm soát truy cập:

- Vì dữ liệu được truyền qua nhiều hệ thống và mạng khác nhau, cần thiết phải có cơ chế xác thực, phân quyền và mã hóa để tránh truy cập trái phép, tấn công trung gian hoặc rò rỉ thông tin.

Câu 2:

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

A. Yếu tố phần cứng:

- CPU (Bộ xử lý trung tâm):
  - Là trung tâm tính toán của mỗi node trong hệ thống phân tán.
  - CPU mạnh giúp tăng tốc độ xử lý tác vụ, giảm thời gian phản hồi.
  - Tuy nhiên, hiệu năng tổng thể có thể bị ảnh hưởng nếu có sự không đồng đều giữa các CPU trong hệ thống (gây ra hiện tượng "nút cổ chai").
- Bộ nhớ (RAM):
  - RAM đóng vai trò lưu trữ tạm thời dữ liệu và cache, giúp tăng tốc độ xử lý.
  - Trong hệ thống phân tán, thiếu bộ nhớ sẽ khiến node phải sử dụng bộ nhớ ảo (chậm hơn) và ảnh hưởng đến toàn bộ hiệu năng nếu node đó đóng vai trò quan trọng.
- Kênh truyền (I/O Bus, Disk, Storage...):
  - Bao gồm ổ cứng, kênh giao tiếp nội bộ hoặc giữa các thiết bị.
  - Truy xuất dữ liệu chậm từ ổ đĩa hoặc truyền dữ liệu giữa các thiết bị nội bộ yếu kém sẽ gây trễ trong toàn hệ thống.

B. Yếu tố mạng:

- Băng thông mạng (Bandwidth):
  - Băng thông quyết định lượng dữ liệu có thể truyền tải trong một đơn vị thời gian.
  - Băng thông thấp sẽ làm giảm tốc độ truyền dữ liệu giữa các node → ảnh hưởng đến tốc độ xử lý, đồng bộ hóa và phản hồi.
- Cấu trúc mạng (Topology):
  - Cấu trúc kết nối vật lý và logic giữa các node ảnh hưởng đến độ tin cậy, hiệu suất và khả năng mở rộng.
  - Một số topology phổ biến:
    - *Star:* Dễ quản lý nhưng nếu trung tâm hỏng sẽ ảnh hưởng toàn hệ thống.
    - *Mesh:* Độ tin cậy cao, nhưng tốn chi phí.
    - *Ring, Bus:* Phù hợp hệ thống nhỏ, nhưng dễ bị nghẽn hoặc mất tín hiệu.

- Topology cũng quyết định tuyến đường dữ liệu, ảnh hưởng đến độ trễ và khả năng cân bằng tải.

2. So sánh Distributed OS và Network OS về quản lý tài nguyên

Tiêu chí	Distributed OS	Network OS
Khái niệm	Là hệ điều hành cho toàn hệ thống phân tán, vận hành như một OS duy nhất	Là tập hợp các OS riêng biệt chạy trên mỗi node, kết nối qua mạng
Quản lý tài nguyên	Trung tâm hoặc phân phối, nhưng đồng bộ và minh bạch	Mỗi máy tự quản lý tài nguyên riêng, chia sẻ qua giao thức mạng
Cách truy cập tài nguyên	Người dùng truy cập tài nguyên từ xa giống như cục bộ	Phải đăng nhập/tương tác cụ thể với từng máy
Yêu cầu đồng bộ	Cao – để bảo đảm tính nhất quán	Thấp hơn – chỉ cần hỗ trợ chia sẻ cơ bản
Ứng dụng	Hệ thống tính toán lưới, siêu máy tính	Mạng LAN văn phòng, chia sẻ máy in, file đơn giản

Tại sao khác nhau?

- Distributed OS cần đảm bảo tính minh bạch và đồng bộ tài nguyên, vì hệ thống vận hành như một thể thống nhất.
- Network OS chỉ cần đảm bảo chia sẻ tài nguyên cơ bản giữa các node, nên không cần đồng bộ sâu hoặc điều phối tài nguyên chặt chẽ.

Câu 3:

1. Nêu và so sánh ba loại hệ thống phân tán

Tiêu chí	Điện toán phân tán (Distributed Computing)	Thông tin phân tán (Distributed Information Systems)	Lan tỏa phân tán (Ubiquitous/Pervasive Computing)
Mục tiêu chính	Tăng năng lực tính toán bằng cách chia nhỏ bài toán	Quản lý, lưu trữ và truy cập thông tin phân tán	Cung cấp dịch vụ số mọi lúc, mọi nơi
Đặc điểm	Tập trung vào xử lý song song và hiệu năng	Tập trung vào tính nhất quán dữ liệu và truy cập thuận tiện	Tập trung vào tính ẩn mình, ngữ cảnh và thích nghi với môi trường
Ví dụ ứng dụng	HPC, xử lý đồ họa, mô phỏng khoa học	Hệ thống quản lý CSDL phân tán, ERP, CRM	Smart home, smart city, thiết bị đeo, IoT
Tính thời gian thực	Không bắt buộc	Không bắt buộc	Có thể có (ví dụ cảm biến môi trường, điều khiển tức thời)
Tính tương tác với người dùng	Thấp hoặc gián tiếp	Trung bình	Rất cao, thường xuyên và tự động

Tóm tắt:

- **Điện toán phân tán** chú trọng hiệu năng.
- **Thông tin phân tán** chú trọng vào lưu trữ và truy vấn dữ liệu.
- **Lan tỏa phân tán** chú trọng tính liền mạch, liên tục, thông minh và "ẩn mình" trong cuộc sống người dùng.

2. Phân tích kiến trúc điện toán lưới (Grid Computing)

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

a. Application Layer (Lớp ứng dụng):

- **Vai trò:** Là nơi triển khai các phần mềm, dịch vụ hoặc chương trình mà người dùng thực thi.

- **Đặc điểm:**
  - Cung cấp giao diện người dùng (UI)
  - Triển khai các mô phỏng khoa học, phân tích dữ liệu, xử lý hình ảnh...
- **Ví dụ:** Người dùng chạy một ứng dụng mô phỏng biến đổi khí hậu trên nền lưới tính toán.

b. Middleware Layer (Lớp phần mềm trung gian):

- **Vai trò:** Là cầu nối giữa lớp ứng dụng và lớp tài nguyên vật lý. Chịu trách nhiệm ẩn đi sự phức tạp của hệ thống.
- **Chức năng:**
  - Phát hiện và truy cập tài nguyên (resource discovery)
  - Quản lý tác vụ và lịch trình (job scheduling)
  - Cân bằng tải (load balancing)
  - Quản lý bảo mật (authentication, authorization)
  - Giao tiếp và truyền dữ liệu giữa các node
- **Ví dụ phần mềm:** Globus Toolkit, gLite

c. Resource Layer (Lớp tài nguyên):

- **Vai trò:** Là nơi cung cấp các tài nguyên vật lý và logic cho hệ thống.
- **Bao gồm:**
  - CPU, bộ nhớ, ổ đĩa, GPU
  - Hệ điều hành cục bộ của từng node
  - Hệ thống lưu trữ, mạng, phần cứng tính toán
- **Lưu ý:** Các tài nguyên có thể không đồng nhất và thuộc nhiều tổ chức khác nhau, nên việc quản lý phức tạp hơn nhiều so với hệ thống truyền thống.

Tóm tắt vai trò các lớp:

- **Application:** Giao diện và chức năng người dùng.
- **Middleware:** Quản lý kết nối, bảo mật, lập lịch và phân phối.
- **Resource:** Nền tảng phần cứng để thực hiện các tác vụ.

Câu 4

1. Vì sao "tính sẵn sàng" (Availability) là mục tiêu quan trọng nhất của hệ thống phân tán

Tính sẵn sàng là khả năng của hệ thống trong việc duy trì hoạt động liên tục và đáp ứng yêu cầu từ người dùng bất cứ lúc nào.

Lý do:

- **Người dùng phân tán, yêu cầu 24/7:** Các hệ thống như web, thương mại điện tử, ngân hàng, email,... đều hoạt động trong môi trường phân tán và phục vụ người dùng ở nhiều nơi, mọi thời điểm.
- **Không thể ngừng dịch vụ:** Một node bị lỗi không được làm ngắt toàn bộ hệ thống.
- **Thiệt hại do downtime:** Mỗi phút gián đoạn có thể gây thiệt hại kinh tế, tổn hại uy tín và mất dữ liệu.
- **Thách thức trong hệ phân tán:** Việc đồng bộ, truyền thông và xử lý lỗi giữa các node làm tăng nguy cơ mất kết nối cục bộ, nên cần khả năng phục hồi linh hoạt.

Tóm lại: Trong hệ thống phân tán, duy trì tính sẵn sàng là điều kiện tiên quyết để hệ thống đạt được các yêu cầu khác như hiệu năng, bảo mật hay khả năng mở rộng.

2. So sánh ba hình thức "tính trong suốt"

Loại trong suốt	**Giải thích**	**Ví dụ minh họa**
Trong suốt truy nhập (Access transparency)	Che giấu sự khác biệt giữa cách truy cập dữ liệu nội bộ và từ xa. Người dùng không cần biết cách truy cập là qua mạng hay cục bộ.	Truy cập tập tin qua giao thức NFS giống như mở tập tin trong máy.
Trong suốt vị trí (Location transparency)	Che giấu vị trí vật lý của tài nguyên. Người dùng không cần biết dữ liệu được lưu ở máy nào.	Truy cập http://service.company.com mà không biết thực tế nó chạy ở đâu.

Trong suốt lỗi (Failure transparency)	Che giấu các lỗi tạm thời, cho phép hệ thống tự động phục hồi mà người dùng không nhận ra lỗi đã xảy ra.	Nếu server A bị ngắt, client tự động chuyển sang server B mà không báo lỗi.

Tóm tắt: Tính trong suốt giúp người dùng không phải quan tâm đến đặc điểm kỹ thuật và sự phức tạp của hệ thống – đó là một trong những nguyên lý cốt lõi của hệ thống phân tán.

3. Mối quan hệ giữa "tính mở" (Openness) và khả năng tương tác (Interoperability)

Tính mở (Openness):

- Là khả năng của hệ thống cho phép các thành phần khác nhau (phần mềm, phần cứng) **giao tiếp, mở rộng hoặc tích hợp** một cách dễ dàng.
- Được thể hiện qua:
  - Giao diện lập trình ứng dụng chuẩn (API)
  - Giao thức truyền thông mở (HTTP, SOAP…)
  - Tài liệu kỹ thuật rõ ràng

Khả năng tương tác (Interoperability):

- Là **khả năng các hệ thống khác nhau làm việc cùng nhau** một cách hiệu quả, dù khác nền tảng, ngôn ngữ hay thiết bị.
- Đòi hỏi các thành phần tuân thủ chung một chuẩn và có thể hiểu được lẫn nhau.

Mối quan hệ:

- **Tính mở là điều kiện cần để đạt được khả năng tương tác.**
  - Nếu hệ thống không mở, không có cách nào để các hệ thống khác hiểu và tương tác với nó.
- Trong hệ thống phân tán, tính mở giúp xây dựng một hệ sinh thái linh hoạt, dễ tích hợp, tương thích đa nền tảng (cross-platform).

Ví dụ: Một hệ thống IoT muốn tương tác với nền tảng điện toán đám mây như AWS, cần tuân thủ các giao thức mở như MQTT, HTTPS và định dạng dữ liệu như JSON hoặc XML.

Câu 5

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

Tiêu chí	**Kiến trúc phân cấp (Hierarchical/Client-Server)**	**Kiến trúc ngang hàng (Peer-to-Peer – P2P)**
Cách tổ chức	Có máy chủ (server) trung tâm điều phối tài nguyên	Mỗi node vừa là client vừa là server
Quản lý tài nguyên	Tập trung, dễ kiểm soát	Phân tán, không có trung tâm quản lý
Tính sẵn sàng	Thấp – nếu server lỗi → toàn hệ thống bị ảnh hưởng	Cao – các peer có thể thay thế lẫn nhau
Khả năng mở rộng	Hạn chế – server dễ quá tải	Cao – càng nhiều peer càng tăng hiệu năng hệ thống
Hiệu năng truyền thông	Tốt nếu số lượng client ít	Tốt nếu cơ chế chia sẻ hiệu quả (VD: BitTorrent)
Ví dụ điển hình	Web server, email server	Mạng chia sẻ file (BitTorrent), Blockchain

Tóm lại:

- Kiến trúc phân cấp dễ triển khai và kiểm soát, nhưng dễ bị nghẽn và lỗi trung tâm.
- Kiến trúc P2P phân tán tốt hơn nhưng phức tạp trong đồng bộ và bảo mật.

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

Mô hình	**Đặc điểm chính**	**Ví dụ ứng dụng điển hình**
Phân tầng (Layered Model)	Tổ chức hệ thống thành các lớp logic (UI, logic xử lý, dữ liệu...)	Website 3 lớp: frontend – backend – database (Web App MVC)
Đối tượng phân tán (Distributed Object Model)	Thành phần giao tiếp dưới dạng các đối tượng từ xa thông qua gọi phương thức	Java RMI, CORBA, .NET Remoting

Kênh sự kiện (Event-based Model)	Các thành phần giao tiếp thông qua sự kiện, theo mô hình publish-subscribe	Hệ thống cảnh báo khẩn cấp, IoT Smart Home (MQTT, Kafka)
Dữ liệu tập trung (Centralized Data Model)	Tài nguyên chính là dữ liệu, tập trung tại một hoặc vài node trung tâm	Google Drive, Dropbox, hệ thống lưu trữ đám mây tập trung

3. Vai trò và tính năng của phần mềm trung gian (middleware)

Vai trò của middleware trong kiến trúc khách – chủ phân tán:

Middleware là **lớp phần mềm trung gian** giữa client và server, giúp ẩn đi chi tiết hạ tầng, tạo điều kiện cho các thành phần phân tán giao tiếp hiệu quả.

Các chức năng chính:

- **Ẩn sự phức tạp:** Client không cần biết tài nguyên ở đâu, truy cập như cục bộ.
- **Cung cấp dịch vụ hệ thống:** Như giao tiếp, truyền thông tin, bảo mật, đồng bộ hóa.
- **Quản lý tài nguyên và tương tác:** Quản lý các đối tượng phân tán, định tuyến thông điệp, cân bằng tải.

Ba tính năng chính mà middleware cung cấp:

1. **Truy cập minh bạch:** Giúp người dùng truy cập tài nguyên mà không biết/quan tâm đến vị trí thực tế.
2. **Bảo mật:** Cung cấp cơ chế xác thực, mã hóa, kiểm soát truy cập.
3. **Giao tiếp liên tiến trình:** Hỗ trợ gọi phương thức từ xa (Remote Procedure Call), truyền thông không đồng bộ (message queue).

Ví dụ middleware phổ biến: CORBA, Java EE, Apache Kafka, gRPC, RabbitMQ.

Câu 6:

1. Phân loại ba loại dịch vụ trong SOA

SOA (Kiến trúc hướng dịch vụ) tổ chức phần mềm thành các **dịch vụ độc lập**, có thể tái sử dụng và kết hợp linh hoạt. Các dịch vụ thường được phân thành ba loại:

Loại dịch vụ	Đặc điểm	Ví dụ điển hình
Dịch vụ cơ bản (Basic Services)	Cung cấp một chức năng đơn lẻ, độc lập, có thể được gọi trực tiếp bởi các dịch vụ khác hoặc client	Dịch vụ kiểm tra tồn kho, dịch vụ xác minh email, dịch vụ tính phí vận chuyển
Dịch vụ tích hợp (Composite Services)	Kết hợp nhiều dịch vụ cơ bản để thực hiện một tác vụ phức tạp hơn	Dịch vụ đặt hàng: kiểm tra tồn kho + tính toán phí + xử lý thanh toán
Dịch vụ quy trình (Process Services / Business Process Services)	Điều phối các dịch vụ cơ bản và tích hợp theo một quy trình nghiệp vụ xác định	Dịch vụ xử lý đăng ký tín dụng ngân hàng: đi qua nhiều bước xác thực, chấm điểm, duyệt hồ sơ

Tóm tắt: Càng lên cao, dịch vụ càng phức tạp và gắn với quy trình kinh doanh.

2. Vòng đời của một dịch vụ SOA và thách thức chính

Vòng đời của dịch vụ SOA có thể chia thành **5 giai đoạn chính**:

a. Phân tích & thiết kế (Analysis & Design)

- **Mục tiêu:** Xác định nhu cầu nghiệp vụ, đặc tả giao diện dịch vụ (WSDL, REST API...).
- **Thách thức:**
  - Hiểu rõ nghiệp vụ để thiết kế đúng mức trừu tượng.
  - Đảm bảo dịch vụ được thiết kế **tái sử dụng được**, không bị cứng nhắc.
  - Thiết kế giao diện ổn định, không bị thay đổi thường xuyên.

b. Phát triển (Implementation)

- **Mục tiêu:** Lập trình logic dịch vụ, kết nối với cơ sở dữ liệu hoặc hệ thống kế thừa.
- **Thách thức:**

- Tích hợp với hệ thống cũ hoặc hệ thống không theo chuẩn.
- Quản lý phiên bản dịch vụ trong môi trường nhiều nhóm phát triển.

c. Triển khai (Deployment)

- **Mục tiêu:** Đưa dịch vụ lên môi trường máy chủ thật hoặc đám mây.
- **Thách thức:**
  - Đảm bảo tính tương thích giữa môi trường phát triển và môi trường triển khai.
  - Kiểm tra hiệu suất và tính sẵn sàng.

d. Đăng ký & khám phá (Publish & Discover)

- **Mục tiêu:** Đưa dịch vụ vào thư viện dịch vụ (service registry) như UDDI để các ứng dụng khác có thể tìm kiếm và sử dụng.
- **Thách thức:**
  - Đảm bảo thông tin metadata của dịch vụ luôn chính xác và đồng bộ.
  - Quản lý phiên bản dịch vụ trong registry.

e. Vận hành & giám sát (Run & Monitor)

- **Mục tiêu:** Theo dõi hiệu suất, bảo trì và cập nhật dịch vụ đang hoạt động.
- **Thách thức:**
  - Phát hiện lỗi sớm, bảo trì mà không gây gián đoạn.
  - Kiểm soát hiệu năng khi số lượng người dùng tăng đột biến.
  - Đảm bảo tính bảo mật khi dịch vụ được gọi từ nhiều bên.

Lưu ý: SOA không chỉ là thiết kế phần mềm mà là cả một chiến lược quản lý vòng đời dịch vụ liên tục.