Câu 1.

a) Nêu và giải thích hai đặc điểm quan trọng nhất của hệ thống phân tán.
b) Trình bày ba lý do cơ bản khiến các ứng dụng phân tán phức tạp hơn so với các ứng dụng đơn lẻ.
c) Nêu và so sánh ba loại hệ thống phân tán: điện toán phân tán, thông tin phân tán và lan tỏa phân tán.
d) Phân tích các lớp chính (application, middleware, resource) trong kiến trúc điện toán lưới và vai trò của từng lớp.

a) Các định nghĩa hệ thống phân tán đều đề cập đến hai đặc điểm quan trọng:
- Đặc điểm thứ nhất đó là tập hợp các phần tử tính toán có thể hoạt động độc lập với nhau
- Đặc điểm thứ hai là chúng cần phải cộng tác để giải quyết một nhiệm vụ chung.

b) Việc xây dựng các ứng dụng phân tán phức tạp hơn nhiều so với các ứng dụng chạy độc lập trên một máy tính, tuy nhiên đây là xu hướng tất yếu của các hệ thống thông tin.
- Lý do đầu tiên phải kể đến là nhu cầu chia sẻ thông tin và tài nguyên, 17 người sử dụng trao đổi thông tin với nhau thông qua một ứng dụng chạy trên máy tính của mình. Dữ liệu cũng có thể được lưu trữ phân tán, điều này sẽ nảy sinh những vấn đề về phân quyền truy nhập, ví dụ cho phép truy nhập dữ liệu từ xa nhưng không cho phép sao chép để lưu giữ cục bộ.
- Lý do thứ hai là vấn đề hiệu năng, trong nhiều trường hợp bắt buộc tính toán phân tán nhằm tận dụng khả năng tính toán song song hoặc nhằm mục đích khai thác khả năng tính toán của các máy tính chuyên dụng.
- Cuối cùng là xuất phát từ yêu cầu về khả năng chịu lỗi, hệ thống 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 được thực hiện bằng cách sử dụng mọi biện pháp nhằm mục đích phát hiện kịp thời và xử lý lỗi, nếu máy chủ bị hỏng hoàn toàn thì sẽ chuyển sang các máy tính dự phòng khác để xử lý, như vậy dịch vụ của hệ thống không bị gián đoạn.

c) 3 loại hệ thống phân tán:
- Các hệ thống điện toán phân tán: Một trong những vấn đề cơ bản của hệ thống phân tán là phải đảm bảo vấn đề hiệu năng, tính toán phân tán thường được sử dụng trong các tác nghiệp yêu cầu hiệu năng cao. Tùy theo nền tảng hệ thống, điện toán phân tán có thể triển khai theo hình thức cụm hoặc lưới, dù theo hình thức nào thì điểm cốt lõi của nó vẫn là việc phân chia nhiệm vụ để xử lý trên nhiều máy tính. Trong hệ thống điện toán cụm, nền tảng phần cứng là tập hợp các máy tính tương tự nhau, chúng được cài đặt các hệ điều hành giống nhau và kết nối với nhau qua đường truyền tốc độ cao, thông thường là mạng cục bộ. Ngược lại, hệ thống điện toán lưới bao gồm các hệ thống phân tán thuộc về nhiều miền quản lý khác nhau và thường không đồng nhất về phần cứng, phần mềm, hệ điều hành và các hệ thống này có thể sử dụng những công nghệ mạng khác nhau.

- Hệ thống thông tin phân tán Thông tin phân tán là thực tế khá phổ biến trong hệ thống phân tán, dữ liệu được lưu trữ trên nhiều máy tính, xử lý dữ liệu trên một máy có thể sẽ liên quan đến những máy khác. Việc dữ liệu được lưu trữ phân tán xuất phát từ nhiều nguyên nhân, có thể do dung lượng dữ liệu lớn khi qui mô phát triển hoặc triển khai kỹ thuật nhân bản để tăng hiệu năng và đảm bảo tính tin cậy. Các phần mềm ngày nay thường được xây dựng theo mô hình ba bên, tách biệt các thành phần lưu trữ dữ liệu với xử lý nghiệp vụ và tiếp nhận yêu cầu.
- Hệ thống lan tỏa phân tán Các máy chủ thường được đặt ở vị trí địa lý cố định và kết nối với nhau qua kênh truyền chất lượng cao, tính ổn định này tạo điều kiện thuận lợi để đạt được tính trong suốt trong xử lý cũng như truy nhập tài nguyên. Các ký thuật xử lý phân tán và nhân bản dữ liệu không những nâng cao hiệu năng hệ thống mà còn có thể che giấu lỗi, nếu xảy ra trên một máy chủ nào đó. Vấn đề còn lại là các máy khách kết nối đến hệ thống máy chủ như thế nào và cách truyền dữ liệu ra sao.

d) Phân tích các lớp chính (application, middleware, resource) trong kiến trúc điện toán lưới và vai trò của từng lớp.
- Lớp ứng dụng (Application): Đây là lớp gần người sử dụng nhất, chứa các ứng dụng thực tế chạy trên nền điện toán lưới. Lớp này sử dụng các dịch vụ do tầng trung gian cung cấp để thực hiện các nghiệp vụ cụ thể, như: Phân tích dữ liệu lớn, Mô phỏng khoa học, Xử lý giao dịch phân tán, Thực hiện các tính toán phức tạp liên ngành.
- Lớp trung gian (Middleware): Là trái tim của kiến trúc điện toán lưới, đảm nhận các chức năng điều phối và che giấu sự phức tạp của hệ thống tài nguyên phân tán: Quản lý tài nguyên, Thiết lập kết nối & bảo mật, Truy nhập trong suốt.
- Lớp tài nguyên (Resource): Quản lý tài nguyên cục bộ như CPU, bộ nhớ, lưu trữ, dữ liệu, dịch vụ. Cung cấp giao diện truy cập đến tài nguyên thực. Thực hiện các thao tác cấu hình, khởi tạo tiến trình, đọc/ghi dữ liệu, theo dõi trạng thái của tài nguyên. Mỗi tổ chức trong hệ thống điện toán lưới có thể có chính sách quản lý riêng. Lớp tài nguyên cần tương thích với tầng trung gian để chia sẻ tài nguyên vào cơ quan ảo (virtual organization).

Câu 2.

a) Định nghĩa phần mềm trung gian và nêu vị trí của nó trong ngăn xếp giao thức so với mô hình OSI và TCP/IP.
b) Phân loại các kiểu middleware (mở hoàn toàn, giao diện mở, giao thức mở, riêng) kèm ví dụ tiêu biểu (CORBA, ODBC, DRDA, ActiveX/Hệ thống OLE…).
c) Vai trò của middleware trong việc hỗ trợ giao tiếp giữa các tiến trình ứng dụng phân tán.

a) Định nghĩa phần mềm trung gian và vị trí trong ngăn xếp giao thức
- Định nghĩa phần mềm trung gian (middleware):

- Phần mềm trung gian là thành phần nằm giữa ứng dụng phân tán và nền tảng hệ thống, nhằm che giấu sự phức tạp của giao tiếp mạng và cung cấp các dịch vụ giao tiếp độc lập với hệ điều hành và phần cứng. Nó hỗ trợ các ứng dụng tương tác dễ dàng, hiệu quả trong môi trường phân tán.

- Vị trí trong mô hình OSI và TCP/IP:
  - OSI: Nằm trong tầng trình diễn và tầng phiên, hỗ trợ định dạng dữ liệu, mã hóa, nén và quản lý phiên làm việc.
  - TCP/IP: Không được định nghĩa tầng trình diễn và tầng phiên tách biệt, nhưng middleware được xem là lớp trên tầng vận tải, tức thuộc về tầng ứng dụng mở rộng trong TCP/IP.

b) Phân loại middleware và ví dụ

- Middleware được phân loại theo hai tiêu chí chính: mức độ mở và khả năng tương thích. Có bốn kiểu chính:

Loại Middleware	Đặc điểm chính	Ví dụ tiêu biểu
1. Mở hoàn toàn	Sử dụng giao diện lập trình và giao thức mở, đảm bảo khả năng tương tác tối đa giữa các hệ thống.	CORBA, DCE RPC, OpenDoc
2. Giao diện mở	Các hệ thống dùng chung giao diện lập trình ứng dụng (API), không phụ thuộc nền tảng hệ thống.	ODBC của Microsoft
3. Giao thức mở	Dùng giao thức chung, nhưng giao diện phải chuyển đổi theo từng nền tảng cụ thể.	DRDA của IBM
4. Riêng (đóng)	Chỉ tương thích trong môi trường phần mềm của một nhà cung cấp cụ thể.	ActiveX/OLE của Microsoft

c) Vai trò của middleware trong giao tiếp ứng dụng phân tán

Phần mềm trung gian đóng vai trò cầu nối giúp các tiến trình trong các hệ thống phân tán có thể trao đổi thông tin một cách trừu tượng, an toàn và hiệu quả, cụ thể:

- Ẩn phức tạp truyền thông: Che giấu chi tiết kỹ thuật tầng thấp như kết nối mạng, phân mảnh dữ liệu, mã hóa...
- Tạo môi trường lập trình thống nhất: Giúp lập trình viên phát triển ứng dụng phân tán mà không cần lo về hệ điều hành hoặc phần cứng cụ thể.
- Hỗ trợ tương tác: Cho phép các ứng dụng viết bằng ngôn ngữ khác nhau trên nền tảng khác nhau có thể tương tác (ví dụ CORBA).
- Cung cấp dịch vụ bổ sung: Như bảo mật, định danh, định vị dịch vụ, quản lý phiên, đồng bộ hóa.
- Cải thiện hiệu năng & mở rộng: Cho phép sử dụng các mô hình đồng bộ, không đồng bộ, truyền thông bền bỉ (ví dụ: hàng đợi thông điệp).

Câu 3.

a) Định nghĩa "thông điệp" (message) và "chuyển thông điệp" (message passing) trong hệ thống phân tán.
b) Nêu những ưu nhược điểm cơ bản của phương pháp chuyển thông điệp (so với chia sẻ bộ nhớ) về tính trong suốt, độ tin cậy và hiệu năng.
c) Trình bày mục đích và bối cảnh ra đời của MPI (Message Passing Interface).
d) Nêu và giải thích ngắn gọn 4 hàm nguyên thủy cơ bản trong MPI (ví dụ MPI_Send, MPI_Bsend, MPI_Ssend, MPI_Irecv) về tính đồng bộ, không đồng bộ và cơ chế vùng đệm.

e) Giải thích mô hình Stub–Skeleton trong RPC, nêu rõ vai trò của từng thành phần.
f) Phác họa tuần tự 10 bước của một cuộc gọi RPC (hình 2.9), giải thích ý nghĩa của mỗi bước.

a) Định nghĩa "thông điệp" (message) và "chuyển thông điệp" (message passing) trong hệ thống phân tán:
   - Thông điệp được hiểu là nội dung thể hiện thông tin cần vận chuyển cho đối tượng khác, chuyển thông điệp là phương pháp cơ bản vận chuyển dữ liệu giữa các ứng dụng trên mạng, mặc dù tính trong suốt thấp nhưng nó vẫn được ưa chuộng để phát triển nhiều hệ thống phân tán. Các phương pháp trao đổi thông tin được phát triển sau này đem lại tính trong suốt cao, nhưng bản chất của chúng vẫn sử dụng phương pháp chuyển thông điệp.

b) Ưu nhược điểm cơ bản của phương pháp chuyển thông điệp (so với chia sẻ bộ nhớ) về tính trong suốt, độ tin cậy và hiệu năng:

	Chuyển thông điệp	Chia sẻ bộ nhớ
Tính trong suốt	Tốt hơn về vị trí và truy cập – tiến trình không cần biết nơi lưu trữ dữ liệu, chỉ cần biết nơi gửi thông điệp.	Kém hơn – khó duy trì sự trong suốt về vị trí khi chia sẻ vùng nhớ phân tán.
Độ tin cậy	Có thể thiết kế để đảm bảo độ tin cậy cao qua kiểm tra lỗi, xác nhận, truyền lại…	Dễ xảy ra lỗi do truy xuất đồng thời, điều kiện tranh chấp…
Hiệu năng	Thấp hơn – do chi phí truyền thông qua mạng (độ trễ, băng thông), tuần tự hóa dữ liệu…	Cao hơn trong môi trường cục bộ – truy xuất bộ nhớ nhanh hơn truyền mạng.

c) Mục đích và bối cảnh ra đời của MPI (Message Passing Interface):
   - Chuyển thông điệp nhất thời sử dụng các hàm nguyên thủy có mức độ trừu tượng thấp, nó chỉ đơn thuần cung cấp các hàm cơ bản phục vụ trao đổi thông tin. Hơn nữa, các Socket chỉ đáp ứng yêu cầu truyền thông trong mạng dựa trên mô hình TCP/IP, nó không phù hợp với các giao thức độc quyền dành riêng cho kênh truyền tốc độ cao. Các thiết bị mạng hiệu năng cao thường đính kèm trình điều khiển do nhà sản xuất cung cấp, chúng có thể không tương thích với nhau, nảy sinh vấn đề về tính khả chuyển của hệ thống.

- Chuyển thông điệp với hiệu năng cao nhưng độc lập với phần cứng đòi hỏi phải được chuẩn hóa, một nhóm các nhà nghiên cứu bắt đầu thảo luận vấn đề này từ năm 1991 và thống nhất tên gọi là giao diện truyền thông điệp MPI. Giao diện truyền thông điệp MPI chưa được bất kỳ tổ chức tiêu chuẩn quốc tế nào xác nhận, nó được nhiều hãng công nghệ lớn hỗ trợ nhưng chưa được tích hợp trong các hệ điều hành, phát triển thư viên giao diện này thường do các tổ chức mã nguồn mở phát triển.

d) 4 hàm nguyên thủy cơ bản trong MPI:
   1. Hàm MPI_bsend hỗ trợ truyền thông không đồng bộ, tiến trình đưa thông điệp vào vùng đệm cục bộ sau đó có thể tiếp tục công việc khác, việc chuyển thông điệp đến bên nhận sẽ do hệ thống MPI thực hiện. Hàm MPI_send thuộc loại phong tỏa, tiến trình gửi phải chờ cho đến khi thông điệp được chuyển đến vùng đệm của bên nhận hoặc vùng đệm tạm thời của hệ thống MPI. Việc tồn tại vùng đệm tạm thời bên trong hệ thống MPI giúp cho thông điệp được lưu trữ an toàn khi nó chưa được chuyển đến bên nhận, tránh hiện tượng ghi đè nếu tiến trình gửi thông điệp khác.
   2. Hàm MPI_ssend hỗ trợ truyền thông đồng bộ, tiến trình gửi bị phong tỏa cho đến khi yêu cầu của nó được chấp nhận, nghĩa là nó có thể tiếp tục chuyển thông điệp khác vào vùng đệm. Hàm MPI_sendrecv thuộc loại truyền thông đồng bộ mạnh nhất, tiến trình chuyển thông điệp đến vùng đệm và chờ cho đến khi thông điệp đã được chuyển thành công đến tiến trình bên nhận.
   3. Hai hàm MPI_send và MPI_ssend đều có các biến thể, chúng sử dụng con trỏ để tránh sao chép thông điệp từ vùng đệm tiến trình sang vùng đệm bên trong hệ thống MPI cục bộ, các biến thể này tương ứng với hình thức truyền thông không đồng bộ.
   4. Hàm MPI_isend, tiến trình gửi chuyển con trỏ đến thông điệp để hệ thống MPI xử lý truyền thông, sau đó ngay lập tức tiếp tục công việc khác, dù cho thông được có thể chưa đến bên nhận. Nhằm ngăn chặn việc ghi đè thông điệp trước khi hoàn thành chuyển thông điệp, thư viện MPI cung cấp các hàm kiểm tra xem đã hoàn thành hoặc thậm chí phong tỏa nếu cần thiết.

e) Mô hình Stub–Skeleton trong RPC:
   - Một thành phần trên máy khách gọi là Stub sẽ mã hóa tên hàm và các tham số vào thông điệp, sau đó sẽ chuyển đến máy chủ. Máy chủ cũng có thành phần tương ứng với Stub gọi là Skeleton, nó giải mã thông điệp và chuyển đến thành phần thực thi như phương pháp gọi thủ tục truyền thống. Kết quả trả về lại được Skeleton mã hóa thành thông điệp để trả về cho máy khách, Stub trên máy khách sẽ giải mã thông điệp và trả về cho chương trình gọi các giá trị hoặc các tham số theo yêu cầu.
   - Thành phần Stub sẽ giúp cho các máy khách xử lý dễ dàng hơn, tiến trình khách sẽ gọi Stub thay vì phải viết đoạn mã triển khai cho hàm đó, các Stub được biên soạn và liên kết với các ứng dụng máy khách trong quá trình phát triển. Thay vì chứa mã đoạn mã để thực hiện gọi thủ tục từ xa, các đoạn mã của Stub sẽ yêu cầu truy vấn những tham số

từ không gian địa chỉ và sau đó chuyển chúng vào thư viện chạy thực trên máy khách. Tương tự như vậy, thành phần Skeleton trên máy chủ giúp cho việc gọi thủ tục của dịch vụ và trả về kết quả cho tiến trình máy khách được thực hiện dễ dàng hơn.

f) Quá trình gọi thủ tục từ xa, quá trình được thực hiện theo mười bước sau:
1. Thủ tục trên máy khách gọi stub như phương pháp gọi thủ tục truyền thống.
2. Stub tạo thông điệp và chuyển đến hệ điều hành của máy khách.
3. Hệ điều hành của máy khách gửi thông điệp đến hệ điều hành của máy chủ.
4. Hệ điều hành của máy chủ chuyển thông điệp đến Skeleton.
5. Skeleton giải mã thông điệp thành các tham số và gọi thủ tục xử lý tương ứng.
6. Máy chủ thực hiện lời gọi thủ tục và trả về giá trị cho Skeleton.
7. Skeleton đóng gói tham số giá trị trả về thành thông điệp và chuyển đến hệ điều hành của máy chủ.
8. Hệ điều hành của máy chủ chuyển thông điệp đến hệ điều hành của máy khách.
9. Hệ điều hành của máy khách nhận thông điệp và chuyển cho Stub.
10. Stub giải mã kết quả và trả về các tham số theo yêu cầu của thủ tục đã gọi.

Câu 4.

a) Trình bày sự khác nhau giữa tên phi cấu trúc (flat name), **tên có cấu trúc (structured name)** và **tên dựa trên thuộc tính (attribute-based name)**.
b) Cho ví dụ minh họa cho từng loại tên trong một hệ thống phân tán.

- Tên phi cấu trúc
  - Tên phi cấu trúc còn gọi là tên phẳng, đơn thuần chỉ gồm chuỗi các bit ngẫu nhiên không phản ánh bất kỳ thông tin nào liên quan tới thực thể. Để tham chiếu đến thực thể thì phải xác định điểm truy nhập của nó, nghĩa là xác định địa chỉ của thực thể trong khi không có bất kỳ manh mối nào, điều đó đòi hỏi xây dựng những giải pháp tìm kiếm phù hợp cho từng hệ thống.
  - Ví dụ: Trong các hệ thống phân tán như Freenet hay IPFS (InterPlanetary File System), các tệp tin hoặc thực thể không được định danh bằng tên có cấu trúc (ví dụ như đường dẫn hoặc tên miền), mà bằng một chuỗi băm (hash).
- Tên có cấu trúc
  - Đặt tên phi cấu trúc phù hợp các máy tính nhưng nó hoàn toàn không thân thiện với người sử dụng, con người cần những tên đơn giản dễ đọc và dễ nhớ. Cách đặt tên có cấu trúc được áp dụng phổ biến trên mạng Internet, cho dù đó là tên của máy tính hay tên của các tập tin. Tên có cấu trúc lại không thuận tiện cho việc xử lý trong các hệ thống máy tính.
  - Ví dụ: đồ thị đặt tên trong hệ điều hành Unix, nút thư mục đại diện cho thư mục chứa tập tin trong khi đó nút lá đại diện cho tập tin cụ thể, nút gốc là tên ổ đĩa.

- Tên dựa trên thuộc tính
  - Đặt tên phi cấu trúc hay đặt tên có cấu trúc đều bảo đảm tham chiếu đến một thực thể duy nhất một cách độc lập với vị trí của thực thể đó. Trong thực tế, hai tiêu chí thân thiện với người sử dụng và độc lập vị trí là chưa đủ, người dụng muốn tìm kiếm các thực thể dựa trên các thuộc tính của chúng, mô tả càng nhiều thuộc tính càng tốt. Có nhiều cách mô tả thực thể nhưng cách phổ biến là mô tả thực thể theo cặp thuộc tính và giá trị, phương pháp này gọi là đặt tên theo thuộc tính. Một thực thể bao gồm nhiều thuộc tính liên quan với nhau, mỗi thuộc tính thể hiện đặc điểm của thực thể, bằng cách xác định giá trị của các thuộc tính sẽ hạn chế được tập các thực thể cần tìm.
  - Ví dụ: người sử dụng với thuộc tính tên là <A>, thuộc tính tổ chức là <FOT> và thuộc tính quốc gia là <VN> thì có thể tạo ra thành một thuộc tính tổng hợp là <A.FOT.VN>

Câu 5.

- **Định nghĩa: Hãy định nghĩa khái niệm phối hợp và đồng bộ trong các hệ thống phân tán. Nêu sự khác nhau chính giữa hai khái niệm này.**
- **Tầm quan trọng: Tại sao việc đồng bộ hóa thời gian giữa các thành viên trong hệ thống phân tán lại cần thiết? Hãy giải thích thông qua một ví dụ cụ thể (ví dụ hai lập trình viên cùng biên dịch chung một tập tin mã nguồn) để minh họa hậu quả khi đồng hồ của các máy không khớp nhau.**
- **Giải thuật Cristian: Trình bày nguyên tắc hoạt động của giải thuật đồng bộ thời gian Cristian. Giải thích cách thuật toán ước lượng độ trễ mạng và tính toán độ lệch giữa đồng hồ máy khách và máy chủ để hiệu chỉnh đồng hồ máy khách.**
- **Giải thuật Berkeley: Mô tả cách thức hoạt động của giải thuật đồng bộ Berkeley. So sánh giải thuật Berkeley với giải thuật Cristian về vai trò của máy chủ và máy khách trong quá trình đồng bộ, cũng như ưu nhược điểm của mỗi giải thuật.**
- **Trình bày cách hoạt động của thuật toán đồng hồ Lamport trong việc gán nhãn thời gian cho các sự kiện: mỗi tiến trình cập nhật bộ đếm của mình như thế nào khi xảy ra sự kiện nội bộ, khi gửi thông điệp và khi nhận thông điệp. Hãy giải thích cơ chế hiệu chỉnh nhãn thời gian khi một thông điệp đến với nhãn thời gian nhỏ hơn nhãn của thông điệp đã được gửi (ví dụ minh họa theo Hình 4.6).**

a. Khái niệm phối hợp và đồng bộ trong các hệ thống phân tán.

- Phối hợp (Coordination) trong hệ thống phân tán là sự kết nối hoạt động của các tiến trình nhằm giải quyết một nhiệm vụ chung. Mục tiêu của nó là quản lý các tương tác và sự phụ thuộc lẫn nhau giữa các tiến trình.
- Đồng bộ (Synchronization) là những hoạt động có cùng chu kỳ hoặc cùng tốc độ, được tiến hành trong cùng một thời gian, nhằm tạo ra sự phối hợp nhịp nhàng giữa các tiến trình. Đồng bộ đòi hỏi một tiến trình phải chờ tiến trình khác hoàn thành thì mới có thể thực hiện công việc của mình.
- Sự khác nhau chính giữa hai khái niệm:

- Phối hợp là khái niệm rộng hơn, bao gồm việc quản lý sự tương tác và phụ thuộc giữa các tiến trình để đạt được mục tiêu chung.
- Đồng bộ là một phần trong phối hợp, tập trung vào thời điểm thực hiện đảm bảo các tiến trình thực thi một cách nhịp nhàng hoặc theo thứ tự phụ thuộc, chẳng hạn một tiến trình phải chờ tiến trình khác.

Tóm lại: Phối hợp là chiến lược tổng thể để các tiến trình làm việc cùng nhau, còn đồng bộ là cơ chế cụ thể để đảm bảo các tiến trình thực thi đúng thời điểm theo sự phối hợp đó.

b. Tầm quan trọng của việc đồng bộ hóa thời gian trong hệ thống phân tán

- Trong hệ thống phân tán, mỗi máy tính có đồng hồ riêng và thường không đồng bộ hoàn toàn với nhau. Sự khác biệt nhỏ giữa các đồng hồ này có thể dẫn đến những hậu quả nghiêm trọng, đặc biệt là trong các hệ thống đòi hỏi xử lý theo thứ tự thời gian chính xác.
- Ví dụ minh họa hai lập trình viên cùng biên dịch một tập tin mã nguồn
  - Bối cảnh là hai lập trình viên làm việc trên cùng một dự án phần mềm, mỗi người sử dụng một máy tính riêng biệt:
    + Lập trình viên A trên máy tính biên dịch, đồng hồ hệ thống đang là 2144.
    + Lập trình viên B trên máy tính cá nhân, đồng hồ đang là 2143.
  - Quy trình:
    + Tập tin output.c được chỉnh sửa và lưu tại máy của lập trình viên B vào thời điểm 2143 (theo đồng hồ máy của B).
    + Sau đó, máy của lập trình viên A biên dịch tập tin và tạo ra output.obj tại thời điểm 2144 (theo đồng hồ máy A).
    + Khi lập trình viên B tiếp tục muốn biên dịch lại output.c, trình biên dịch sẽ so sánh: Thời gian output.c là 2143. Thời gian output.obj là 2144.
  - Vấn đề phát sinh do đồng hồ không đồng bộ:
    + Trình biên dịch cho rằng output.c cũ hơn output.obj.
    + Không thực hiện biên dịch lại, mặc dù trên thực tế output.c đã được chỉnh sửa sau đó.
- Hậu quả khi đồng hồ không khớp: Tập tin mã nguồn đã thay đổi không được biên dịch lại, gây ra:
  - Chạy chương trình với mã cũ.
  - Lỗi logic hoặc bug khó phát hiện.
  - Hiểu lầm và khó khăn trong việc gỡ lỗi, lập trình viên không biết vì sao thay đổi của mình không được phản ánh.

c. Giải thuật Cristian

- Nguyên tắc hoạt động của giải thuật đồng bộ thời gian Cristian: Giải thuật Cristian được đề xuất nhằm giúp máy khách đồng bộ thời gian với máy chủ một cách chính xác hơn, bằng cách

ước lượng độ trễ mạng hai chiều và tính toán độ lệch thời gian giữa đồng hồ của máy khách và máy chủ.

- Quy trình hoạt động:
    1. Máy khách gửi yêu cầu đồng bộ thời gian đến máy chủ thời gian (thường là máy chủ NTP) vào thời điểm T1.
    2. Máy chủ nhận yêu cầu tại thời điểm T2, sau đó trả về phản hồi vào thời điểm T3, bao gồm thời gian nhận yêu cầu và thời gian gửi phản hồi.
    3. Máy khách nhận được phản hồi tại thời điểm T4.
- Cách ước lượng độ trễ mạng và hiệu chỉnh đồng hồ:
    1. Công thức tính độ lệch giữa đồng hồ máy khách và máy chủ (gọi là θ):

$$\theta = \frac{(T2 - T1) + (T3 - T4)}{2}$$

    Giải thích:
    - (T2 - T1) là độ trễ từ máy khách đến máy chủ.
    - (T3 - T4) là độ trễ từ máy chủ về máy khách (giá trị âm).
    - Trung bình hai giá trị này rồi cộng với T3 để tìm ra thời gian thực tại thời điểm T4, sau đó so sánh với T4 để tính ra θ.
    2. Hiệu chỉnh thời gian máy khách:
    - Nếu $\theta > 0$: máy khách chạy chậm hơn → phải tăng thời gian.
    - Nếu $\theta < 0$: máy khách chạy nhanh hơn → phải giảm thời gian (cần trọng vì không được quay lui thời gian).
- Vấn đề khi điều chỉnh thời gian và cách xử lý:
    - Không được điều chỉnh lùi thời gian một cách đột ngột, vì có thể gây lỗi logic trong hệ thống (ví dụ: một sự kiện sau lại có timestamp nhỏ hơn sự kiện trước).
    - Nếu buộc phải điều chỉnh lùi, phải tăng thời gian theo từng bước nhỏ. Ví dụ: cứ mỗi 10ms thì không tăng thời gian hệ thống, lặp lại cho đến khi đồng hồ đạt thời gian đúng → cách này không quay lui thời gian.
    - Với trường hợp điều chỉnh tiến thời gian ($\theta > 0$), cũng cần kiểm tra xem có tác vụ theo lịch trình đang chờ không để tránh gây lỗi hoặc bỏ sót tác vụ.
- Ước lượng độ trễ truyền thông (δ):
    - Công thức:

$$\delta = \frac{(T4 - T1) - (T3 - T2)}{2}$$

        + (T4 - T1) là tổng thời gian khứ hồi (RTT - round trip time).
        + (T3 - T2) là thời gian xử lý tại máy chủ.
        + Lấy hiệu của hai giá trị trên chia đôi sẽ được ước lượng độ trễ một chiều.
    - Để tăng độ chính xác, tiến trình có thể thực hiện 8 lần đo, sau đó chọn:
        + Giá trị nhỏ nhất của δ làm độ trễ đáng tin cậy nhất.
        + Tương ứng, giá trị θ đi kèm được dùng để hiệu chỉnh đồng hồ máy khách.

d. Giải thuật Berkeley

- Giải thuật Berkeley được thiết kế để đồng bộ thời gian giữa các máy tính trong một hệ thống nội bộ, không yêu cầu có máy chủ thời gian chính xác tuyệt đối như UTC.
- Các bước hoạt động chính:
  1. Chọn thành viên điều phối (coordinator) – một máy tính đóng vai trò máy chủ tạm thời trong quá trình đồng bộ.
  2. Thành viên điều phối gửi thời gian hiện tại của mình cho các máy khách (các máy còn lại trong nhóm).
  3. Mỗi máy khách tính độ lệch thời gian của mình so với thời gian nhận được từ điều phối, rồi gửi lại độ lệch đó cho điều phối.
  4. Điều phối tính trung bình các độ lệch nhận được, bao gồm cả độ lệch của chính nó, và dùng giá trị trung bình này làm cơ sở để:
     - Tính toán thời gian điều chỉnh cho từng máy.
     - Gửi thông điệp yêu cầu các máy điều chỉnh thời gian của mình tương ứng.
  5. Các máy tự điều chỉnh đồng hồ của mình dựa trên chỉ dẫn từ điều phối.
- So sánh giải thuật Berkeley và Cristian

	Berkeley	Cristian
Vai trò máy chủ	Máy chủ chỉ là điều phối viên, không cần có thời gian chính xác tuyệt đối.	Máy chủ chỉ là điều phối viên, không cần có thời gian chính xác tuyệt đối.
Vai trò máy khách	Máy khách bị động nhận yêu cầu và phản hồi độ lệch thời gian.	Máy khách chủ động gửi yêu cầu đến máy chủ thời gian.
Cơ chế hoạt động	Điều phối viên thu thập độ lệch → tính trung bình → gửi chỉ dẫn điều chỉnh về cho từng máy.	Máy khách gửi yêu cầu → nhận phản hồi → tính độ lệch và tự hiệu chỉnh.
Độ chính xác	Trung bình khá, vì giả thiết không có đồng hồ chính xác và không tính kỹ độ trễ mạng nếu không mở rộng.	Cao hơn, vì có thể đồng bộ theo thời gian UTC và tính đến độ trễ mạng.
Độ phụ thuộc	Phụ thuộc vào thành viên điều phối, nhưng bất kỳ máy nào cũng có thể làm điều phối.	Phụ thuộc hoàn toàn vào máy chủ thời gian.
Khả năng chống lỗi	Nếu điều phối viên lỗi → có thể bầu chọn máy điều phối khác.	Nếu mất máy chủ thời gian → không thể đồng bộ.
Tính thích ứng	Phù hợp hệ thống cục bộ, không có truy cập NTP.	Phù hợp hệ thống lớn, có truy cập NTP.
Bảo mật	Dễ bị tấn công hơn: máy độc hại có thể làm sai lệch đồng bộ.	Dễ kiểm soát hơn vì chỉ một máy chủ.
Tính mở rộng	Kém hơn, vì cần phản hồi từ tất cả thành viên, nếu 1 máy không phản hồi thì thất bại.	Tốt hơn, có thể đồng bộ hàng triệu thiết bị.

- Ưu điểm và nhược điểm

	Berkeley	Cristian
Ưu điểm	Không yêu cầu đồng hồ chính xác ở bất kỳ máy nào. Phù hợp với mạng cục bộ, nội bộ không cần chuẩn UTC. Có thể lựa chọn lại điều phối nếu bị lỗi.	Đơn giản, dễ triển khai. Đồng bộ theo chuẩn thời gian quốc tế (UTC) nếu có kết nối NTP. Độ chính xác cao (sai số thấp, khoảng 50ms).
Nhược điểm	Dễ bị ảnh hưởng bởi độ trễ truyền thông, gây sai số. Yêu cầu phản hồi từ tất cả các máy, nếu một máy không trả lời → thất bại. Bảo mật yếu, dễ bị phá hoại nếu có máy giả mạo tham gia.	Phụ thuộc vào một máy chủ thời gian. Không phù hợp với hệ thống nội bộ không có kết nối ngoài. Đồng hồ máy khách có thể phải quay lui gây lỗi nếu không xử lý đúng.

e. Thuật toán đồng hồ Lamport – Gán nhãn thời gian
Mỗi tiến trình giữ một bộ đếm logic *L* (ban đầu là 0).
Cách cập nhật:
- Sự kiện nội bộ: Khi tiến trình thực hiện sự kiện nội bộ: L := L + 1
- Gửi thông điệp: Khi gửi thông điệp *m* m, gửi kèm giá trị thời gian hiện tại *L*.
  L := L + 1, sau đó gắn timestamp(m) := L
- Nhận thông điệp: Khi nhận thông điệp *m* m có thời gian *Tm*:
  L := max(L, T_m) + 1

Câu 6.

- **Hãy định nghĩa khái niệm** luồng (thread) **và nêu sự khác biệt cơ bản giữa luồng và tiến trình (process) trong hệ điều hành.**
- **Giải thích tại sao việc chia tiến trình thành nhiều luồng có thể cải thiện hiệu năng trên máy tính đa bộ vi xử lý. Lấy ví dụ minh họa từ bài toán tính toán và nhập liệu song song như trong Microsoft Excel.**
- **Hãy định nghĩa ảo hóa (virtualization) và nêu vai trò chính của kỹ thuật ảo hóa trong các hệ thống phân tán.**
- **Phân loại các hình thức ảo hóa dựa trên lớp giao diện (instruction-set virtualization vs. system-level virtualization):**
  - **Máy ảo tiến trình (process-level VM, ví dụ Java Runtime).**
  - **Giám sát máy ảo (hypervisor-based VM).**
    Mô tả cơ chế hoạt động và điểm khác biệt cơ bản giữa hai hình thức này.

a. Định nghĩa khái niệm luồng (thread):
- Luồng là một đơn vị nhỏ hơn của tiến trình, thực hiện đoạn mã chương trình của mình một cách độc lập với các luồng khác. Luồng không cần đạt đến độ trong suốt cao nếu thao tác

đó làm suy giảm hiệu năng, vì vậy hệ thống luồng thường chỉ duy trì thông tin tối thiểu nhằm chia sẻ đơn vị xử lý trung tâm.

- Sự khác biệt cơ bản giữa luồng và tiến trình:
  - Tiến trình là một chương trình đang chạy trên bộ xử lý ảo của hệ điều hành, có ngữ cảnh riêng bao gồm bộ nhớ, các thanh ghi, tập tin đang dùng, thông tin tài khoản...
  - Luồng thì chia sẻ tài nguyên của tiến trình, không tạo không gian bộ nhớ riêng mà dùng chung không gian của tiến trình cha.
  - Hệ điều hành phải tạo không gian bộ nhớ riêng cho mỗi tiến trình, trong khi các luồng chỉ cần thông tin tối thiểu và không phải thay đổi MMU hay TLB khi chuyển đổi.
  - Tạo và hủy luồng nhẹ hơn so với tiến trình do thao tác chủ yếu nằm ở mức người dùng, không cần sao chép toàn bộ dữ liệu như tiến trình.

b. Việc chia tiến trình thành nhiều luồng có thể cải thiện hiệu năng trên máy tính đa bộ vi xử lý vì:
  - Mỗi luồng có thể được gán cho một bộ xử lý khác nhau để thực thi song song, từ đó tận dụng tối đa tài nguyên của hệ thống. Thay vì một tiến trình xử lý tuần tự từng tác vụ, nhiều luồng có thể thực hiện các công việc độc lập cùng lúc, giảm thời gian chờ và tăng hiệu quả xử lý.

- Ví dụ minh họa:
  - Microsoft Excel Trong bảng tính Excel, các ô hoạt động độc lập với nhau. Nếu người dùng thay đổi giá trị của một ô thì những ô có liên quan sẽ được cập nhật. Nếu chỉ dùng một luồng:
    + Khi đang tính toán, người dùng không thể nhập liệu.
    + Khi đang nhập dữ liệu, quá trình tính toán bị dừng.
  - Giải pháp là sử dụng đa luồng:
    + Một luồng đảm nhiệm giao tiếp với người dùng.
    + Một luồng khác xử lý tính toán cập nhật giá trị ô.
    + Có thể thêm luồng thứ ba để sao lưu dữ liệu vào ổ đĩa.

    Nhờ đó, các hoạt động có thể diễn ra đồng thời, giúp chương trình mượt mà và hiệu quả hơn.

c. Định nghĩa ảo hóa (virtualization):
  - Ảo hóa là kỹ thuật tạo ra một phiên bản trừu tượng (ảo) của tài nguyên vật lý như phần cứng, hệ điều hành, thiết bị lưu trữ hay tài nguyên mạng. Mục tiêu của ảo hóa là cho phép nhiều hệ thống hoặc tiến trình sử dụng cùng một tài nguyên vật lý mà không xảy ra xung đột, bằng cách mô phỏng hoặc che giấu chi tiết thực của tài nguyên bên dưới.

- Vai trò chính của kỹ thuật ảo hóa trong các hệ thống phân tán:
  - Tăng tính linh hoạt và khả chuyển (portability): Ảo hóa cho phép phần mềm có thể chạy trên nhiều nền tảng phần cứng khác nhau mà không cần thay đổi mã nguồn.

- Tăng hiệu quả sử dụng tài nguyên: Cho phép chia sẻ tài nguyên vật lý (như máy chủ, bộ nhớ, ổ cứng) cho nhiều tiến trình hoặc người dùng, tối ưu hiệu suất và giảm chi phí phần cứng.
- Tăng tính bảo mật và cách ly: Mỗi máy ảo hoạt động độc lập. Nếu một máy ảo bị tấn công hoặc gặp lỗi, nó không ảnh hưởng đến các máy ảo khác hay hệ thống vật lý.
- Hỗ trợ khả năng mở rộng và phục hồi: Dễ dàng tạo, sao chép, di chuyển hoặc khôi phục máy ảo để đáp ứng nhu cầu thay đổi trong hệ thống phân tán.
- Nâng cao tính trong suốt của hệ thống: Người dùng hoặc tiến trình không cần biết tài nguyên thực sự nằm ở đâu, mà vẫn có cảm giác như đang sử dụng tài nguyên cục bộ.
- Nền tảng cho điện toán đám mây: Ảo hóa là công nghệ cốt lõi của các dịch vụ đám mây như IaaS (Infrastructure as a Service), giúp người dùng có cảm giác sở hữu máy chủ riêng dù đang chia sẻ tài nguyên vật lý.

d. Phân loại các hình thức ảo hóa dựa trên lớp giao diện

- Máy ảo tiến trình (Process-level Virtual Machine)
  - Ví dụ: Java Virtual Machine (JVM), .NET CLR
  - Lớp giao diện được ảo hóa: Instruction Set (tập lệnh giả lập)
  - Cơ chế hoạt động: Máy ảo tiến trình cung cấp một môi trường thực thi cho chương trình, trong đó toàn bộ mã nguồn được biên dịch thành mã trung gian (bytecode). Sau đó, máy ảo chuyển đổi hoặc thông dịch bytecode này thành mã máy phù hợp với hệ thống hiện tại.
  - Đặc điểm:
    + Hoạt động bên trên hệ điều hành.
    + Thường chỉ phục vụ cho một tiến trình tại một thời điểm.
    + Không trực tiếp quản lý tài nguyên phần cứng.
    + Tập trung vào tính di động (portability) và độc lập nền tảng.
- Giám sát máy ảo (Hypervisor-based Virtual Machine)
  - Ví dụ: VMware, Xen, Microsoft Hyper-V
  - Lớp giao diện được ảo hóa: System Level (giao diện hệ điều hành – phần cứng)
  - Cơ chế hoạt động: Hypervisor hoạt động ở mức thấp (gần với phần cứng), đóng vai trò trung gian giữa phần cứng vật lý và các hệ điều hành chạy trên máy ảo. Nó cung cấp môi trường độc lập để nhiều hệ điều hành có thể chạy song song trên cùng một phần cứng.
  - Đặc điểm:
    + Có thể hoạt động trực tiếp trên phần cứng (bare-metal) hoặc trên hệ điều hành hiện có (hosted).
    + Quản lý toàn bộ tài nguyên: CPU, bộ nhớ, I/O…
    + Nhiều máy ảo (OS) chạy độc lập, có thể khác nhau về hệ điều hành.
    + Hỗ trợ phân vùng, cách ly, sao lưu, di chuyển máy ảo.
- Điểm khác biệt cơ bản:
  - Máy ảo tiến trình tạo môi trường chạy cho một ứng dụng cụ thể, không can thiệp đến phần cứng.

- Giám sát máy ảo cho phép chạy nhiều hệ điều hành như các máy riêng biệt, chia sẻ phần cứng chung.

Câu 7.

- **"Tương tranh" là gì? Hãy định nghĩa khái niệm tương tranh theo Edsger Dijkstra và nêu hai hiện tượng phổ biến (mất cập nhật, đọc không nhất quán) kèm theo mô tả ngắn về nguyên nhân xảy ra của mỗi hiện tượng.**
- **Dựa trên ví dụ chuyển tiền giữa các tài khoản A, B, C, hãy giải thích tại sao trình tự thực thi không đồng bộ của các lệnh GetBalance/SetBalance/Withdraw dẫn đến hiện tượng mất cập nhật và đọc không nhất quán.**

a. Định nghĩa "Tương tranh" theo Edsger Dijkstra và mô tả hai hiện tượng phổ biến

- Tương tranh là hiện tượng tại một thời điểm có nhiều yêu cầu sử dụng tài nguyên dùng chung, theo Edsger Dijkstra: "Tương tranh xảy ra khi nhiều hơn một luồng thực thi có thể chạy đồng thời".
- Hai hiện tượng phổ biến:
  - Mất cập nhật:
    + Mô tả: Khi nhiều giao dịch cập nhật cùng một dữ liệu mà không được đồng bộ đúng cách, một số cập nhật có thể bị ghi đè, làm mất dữ liệu trước đó.
    + Nguyên nhân: Do trình tự thực hiện các câu lệnh bị xáo trộn, các lệnh cập nhật được thực thi dựa trên giá trị không mới nhất, hoặc giá trị chưa được cập nhật của các giao dịch khác.
  - Đọc không nhất quán:
    + Mô tả: Khi một giao dịch đang đọc dữ liệu mà cùng lúc có giao dịch khác đang thay đổi dữ liệu đó, khiến giá trị đọc được không phản ánh trạng thái thực sự.
    + Nguyên nhân: Do một giao tác đang cập nhật dữ liệu trong khi giao tác khác đọc dữ liệu đó ở thời điểm chưa cập nhật hoàn toàn.

b. Hiện tượng mất cập nhật & đọc không nhất quán

- Giả sử tài khoản B có 200$, tài khoản A và C lần lượt chuyển cho B một số tiền bằng 10% số dư hiện tại của B. Nếu hai giao dịch A → B và C → B xảy ra cùng lúc và thực hiện không đồng bộ, thì cả hai đều lấy số dư B là 200$, tính 10% là 20$, và mỗi giao dịch cộng 20$ vào tài khoản B.
- Kết quả là tài khoản B chỉ được cộng 20$ hai lần, thành 220$, trong khi đáng ra nếu cập nhật lần lượt, số dư sẽ là 220$ → 10% tiếp theo là 22$, tổng cộng là 242$. Như vậy, B bị thiếu 22$ − 20$ = 2$, do hai giao dịch lấy cùng một số dư cũ để tính toán → gây mất cập nhật.

Câu 8.

- **Hãy nêu định nghĩa phục hồi lùi và phục hồi tiến, đồng thời liệt kê ưu – nhược điểm cơ bản của mỗi phương pháp.**
- **Giải thích vai trò của điểm kiểm tra (checkpoint) và ghi nhật ký thông điệp (message logging) trong cơ chế phục hồi lùi, và tại sao kết hợp cả hai lại giúp giảm chi phí lưu trạng thái.**

a. Định nghĩa phục hồi lùi và phục hồi tiến, kèm ưu – nhược điểm

- Phục hồi lùi (Backward Recovery): Đưa hệ thống trở lại trạng thái trước khi xảy ra lỗi. Dựa vào các điểm kiểm tra (checkpoint) được ghi nhận trước đó.
  - Ưu điểm:
    + Đơn giản, phổ biến.
    + Phù hợp với lỗi nhất thời, dễ tích hợp vào hệ thống trung gian.
  - Nhược điểm:
    + Có thể mất dữ liệu gần thời điểm lỗi.
    + Chi phí hiệu năng cao nếu checkpoint thường xuyên.
    + Có thể xảy ra vòng lặp phục hồi nếu lỗi cứ lặp lại sau checkpoint.
- Phục hồi tiến (Forward Recovery): Không quay lại mà chuyển hệ thống đến trạng thái mới hợp lệ. Phải biết trước các lỗi có thể xảy ra và chuẩn bị biện pháp sửa lỗi.
  - Ưu điểm:
    + Không mất trạng thái gần nhất.
    + Phù hợp với các hệ thống có khả năng sửa lỗi (ví dụ: mã sửa lỗi trong truyền tin).
  - Nhược điểm:
    + Khó triển khai, cần dự đoán lỗi.
    + Không áp dụng chung cho mọi hệ thống.

b. Vai trò của checkpoint và message logging trong phục hồi lùi

- Checkpoint:
  - Lưu lại trạng thái hệ thống tại một thời điểm xác định.
  - Khi xảy ra lỗi → khôi phục lại trạng thái từ checkpoint gần nhất.
- Message Logging:
  - Ghi lại thông điệp gửi/nhận sau mỗi checkpoint.
  - Sau khi phục hồi từ checkpoint → dùng log để phát lại thông điệp và khôi phục trạng thái hiện tại.
- Kết hợp cả hai giúp:
  - Giảm số lượng checkpoint (do dùng message để tái tạo).
  - Tối ưu hiệu năng: ít checkpoint → tiết kiệm thời gian/băng thông.
  - Vẫn đảm bảo phục hồi đầy đủ → chỉ cần phát lại phần đã log.

Câu 9.

- **Hãy định nghĩa nhân bản dữ liệu trong hệ thống phân tán và nêu hai lý do chính mà ta phải triển khai nhân bản.**
- **Giải thích tại sao việc nhân bản dữ liệu có thể dẫn đến vấn đề nhất quán (consistency). Lấy ví dụ kịch bản cache web để minh họa cách người dùng có thể vẫn nhìn thấy "nội dung cũ" sau khi dữ liệu gốc đã được cập nhật.**
- **Hãy liệt kê và định nghĩa ngắn gọn bốn mô hình nhất quán theo thứ tự thao tác được nhắc đến trong phần 8.2.2.**
- **Giải thích sự khác nhau giữa nhất quán nghiêm ngặt (strict consistency) và nhất quán tuần tự (sequential consistency). Tại sao mô hình tuần tự yếu hơn nhưng lại thực tế hơn trong hệ phân tán?**

a. Định nghĩa nhân bản dữ liệu và hai lý do chính

- Nhân bản dữ liệu là quá trình tạo và duy trì nhiều bản sao của cùng một dữ liệu ở nhiều vị trí khác nhau như một cách đảm bảo tính sẵn sàng, tính đáng tin cậy và khả năng phục hồi của hệ thống.
- Hai lý do chính để triển khai nhân bản:
  - Tăng khả năng chịu lỗi: Khi một bản sao bị hỏng, hệ thống vẫn có thể hoạt động bằng cách sử dụng bản sao khác.
  - Cải thiện hiệu năng: Nhờ vào việc phân bố các bản sao tại nhiều vị trí địa lý, yêu cầu truy cập được xử lý bởi máy chủ gần nhất, giảm độ trễ và cân bằng tải.

b. Nhân bản dẫn đến vấn đề nhất quán – Ví dụ cache web

- Việc nhân bản dẫn đến vấn đề nhất quán vì khi một bản sao được cập nhật, các bản sao khác có thể chưa được cập nhật kịp thời, gây ra tình trạng dữ liệu không đồng bộ giữa các bản sao.
- Ví dụ minh họa – cache web: Trình duyệt gửi yêu cầu đến máy chủ web → máy chủ truy vấn CSDL → tạo HTML và trả về → để tối ưu, máy chủ web giữ lại bản sao trong cache. Nếu trang web được yêu cầu lại, nội dung sẽ được lấy từ cache mà không cần truy vấn CSDL.
- Vấn đề: Nếu dữ liệu gốc đã được cập nhật nhưng chưa được làm tươi cache, người dùng vẫn sẽ thấy nội dung cũ.

c. Liệt kê và định nghĩa bốn mô hình nhất quán theo thứ tự thao tác (mục 8.2.2)

1. Nhất quán nghiêm ngặt (Strict Consistency): Mọi thao tác đọc phải trả về giá trị tương ứng với thao tác ghi gần nhất (theo thời gian thực) lên cùng một mục dữ liệu.
2. Nhất quán tuần tự (Sequential Consistency): Mọi tiến trình đều thấy cùng một thứ tự thực hiện các thao tác đọc/ghi, nhưng không cần đúng thời gian thực.
3. Nhất quán nhân quả (Causal Consistency): Nếu thao tác A gây ra hoặc có quan hệ nhân quả với B thì mọi tiến trình phải thấy A trước B. Các thao tác không liên quan nhân quả có thể thấy theo thứ tự khác nhau.

4. Nhất quán hàng đợi (FIFO Consistency): Các thao tác ghi từ một tiến trình phải được các tiến trình khác nhìn thấy theo đúng thứ tự đã xảy ra, nhưng thứ tự giữa các tiến trình khác nhau có thể khác nhau.

d. So sánh Strict và Sequential Consistency

Tiêu chí	Nhất quán nghiêm ngặt (Strict)	Nhất quán tuần tự (Sequential)
Đồng hồ	Phụ thuộc vào thời gian thực	Không phụ thuộc thời gian thực
Đọc dữ liệu	Luôn thấy giá trị mới nhất	Thứ tự hợp lý, không cần là mới nhất
Hiệu năng	Rất thấp do cần đồng bộ toàn cục	Cao hơn, không cần đồng hồ toàn cục
Triển khai thực tế	Gần như không khả thi	Thực hiện được trong nhiều hệ thống

Kết luận: Nhất quán tuần tự yếu hơn nhưng thực tế hơn, vì không cần đồng hồ toàn cục và cho phép đan xen thao tác khi triển khai trên hệ phân tán.