**BÀI TẬP LÝ THUYẾT**

**MÔN HỌC: CÁC HỆ THỐNG PHÂN TÁN & ỨNG DỤNG CHƯƠNG 1: TỔNG QUAN VÀ KIẾN TRÚC HỆ PHÂN TÁN** HỌ TÊN SV: NGUYỄN NHƯ HIẾU MSSV: 20251063M

MÃ LỚP: MÃ HỌC PHẦN: IT6575-1-25

**Câu hỏi 1:** Em hãy nêu thêm 2 ví dụ về dịch vụ được coi là Hệ Phân Tán (ngoài 2 ví dụ WWW và Email đã trình bày trên lớp). Dựa vào định nghĩa, giải thích tại sao chúng được coi là Hệ Phân Tán.

**Định nghĩa của giáo sư Andrew S. Tanenbaum:** Hệ phân tán là một tập hợp các máy tính độc lập có kết nối, cung cấp dịch vụ cho người dùng như một hệ thống duy nhất

Các ví dụ về hệ phân tán: mạng xã hội (Facebook, Instagram, …), mạng blockchain (Bitcoin, Ethereum)

Mạng xã hội (Social Media Networks):

* Mô hình Client Server
* Thông tin, dữ liệu người dùng lưu trữ trên nhiều máy chủ (Server) khác nhau, không tập trung tại vị trí duy nhất.
* Dữ liệu tại các máy chủ sẽ được đồng bộ với nhau.
* Người dùng được cung cấp giao diện thống nhất (facebook.com hoặc app facebook)

Mạng blockchain:

* Mô hình Peer to peer
* Các node trong mạng blockchain đều được coi là 1 máy chủ, có khả năng tương tác lẫn nhau
* Dữ liệu trên blockchain không tập trung nằm ở nơi nào mà được phân tán tới các node khác nhau trên mạng.
* Cung cấp các API cố định để thực hiện: Giao dịch, kiểm tra thông tin khối, …

**Câu hỏi 2:** Tại sao nói tính chia sẻ tài nguyên của Hệ Phân Tán có khả năng: Giảm chi phí, tăng tính sẵn sàng và hỗ trợ làm việc nhóm? Tuy nhiên lại tăng rủi ro về an toàn thông tin? Giải thích.

### 1. ****Giảm chi phí****

* Việc mở rộng dọc (Vertical Scaling) như nâng Ram, CPU sẽ là cách xử lý nhanh khi hệ thống bị cao tải. Khi một hệ thống tăng tải tới ngưỡng nhất định, cách thức này sẽ tốn chi phí và không khả thi.
* Vì vậy mở rộng theo chiều ngang (Horizontal Scaling) giúp dữ liệu được phân tán sang các node khác nhau với chi phí thấp hơn nhưng yêu cầu cài đặt phức tạp hơn để sao cho các node giao tiếp với nhau một cách hiệu quả.

**2. Tăng tính sẵn sàng và làm việc nhóm**

* Trong hệ thống phân tán, dữ liệu giữa các node có cơ chế sharding, replica
* Khi 1 node bị chết, request sẽ lập tức redirect sang node mới vẫn đảm bảo được việc dữ liệu đầy đủ.

**3. Rủi ro về an toàn thông tin**

* Khi 1 node bị tấn công, node chứa thông tin nhạy cảm của các node khác mà không mã hóa dẫn đến tình trạng toàn bộ các node bị ảnh hưởng.
* Dữ liệu qua đường truyền giữa các node có thể bị nghe lén hoặc thay đổi
* Nếu quản lý tốt, thông tin trao đổi giữa được mã hóa đối xứng và chữ ký mã hóa một chiều sẽ tránh được trường hợp dữ liệu bị nghe lén hay thay đổi trong đường truyền.

**Câu hỏi 3:** Liên quan đến *tính trong suốt*, giải thích tại sao nhà quản trị hệ thống phải xem xét việc cân bằng giữa hiệu năng và độ trong suốt? Đưa ra ví dụ cụ thể để giải thích.

**Khái niệm**: Tính trong suốt (transparency) mô tả mức độ mà hệ phân tán trông như một hệ thống thống nhất với người dùng/ứng dụng, che giấu sự phức tạp của nhiều thành phần, vị trí, và trạng thái sao chép.

**Tình huống**:  
Cần truy xuất thông tin người dùng từ một cơ sở dữ liệu phân tán. Để duy trì tính trong suốt, ứng dụng không trực tiếp tương tác với cơ sở dữ liệu mà thông qua một lớp dịch vụ trung gian:

* **Lớp bảo mật: Kiểm tra xác thực người dùng**
* Lớp kiểm toán: Lưu thông tin các thao tác của người dùng
* Lớp dữ liệu: Truy xuất dữ liệu từ database

Mỗi lớp ở các node khác nhau khi tương tác với nhau có độ trễ nhất định (Round-Trip Time). Nhiều lớp dẫn tới độ trễ tăng cao, giảm hiệu năng. Ngoài ra, nếu một lớp bị cao tải hay gặp lỗi, việc check nguyên nhân giữa các lớp/node trung gian sẽ gặp khó khăn.

Vì vậy, cần cân bằng số lượng lớp trung gian với từng bài toán cụ thể để cân bằng giữa tính trong suốt và hiệu năng.

**Câu hỏi 4:** Trong mô hình kiến trúc phân tầng OSI của Mạng máy tính, hãy trình bày tóm tắt chức năng của từng tầng. Lấy ví dụ cụ thể khi chúng ta thay đổi/cập nhật một tầng bất kỳ thì không ảnh hưởng đến hoạt động của các tầng khác.

|  |  |
| --- | --- |
| Tầng 7: Ứng dụng (Application Layer) | **Chức năng**: Cung cấp giao diện cho người dùng và các ứng dụng. Xử lý các yêu cầu từ ứng dụng.  **Ví dụ**: HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol). |
| Tầng 6: Trình diễn (Presentation Layer) | **Chức năng**: Chuyển đổi dữ liệu thành định dạng mà ứng dụng có thể hiểu. Mã hóa và giải mã dữ liệu.  **Ví dụ**: Mã hóa JPEG, SSL (Secure Sockets Layer). |
| Tầng 5: Phiên (Session Layer) | **Chức năng**: Quản lý phiên làm việc giữa các ứng dụng. Thiết lập, duy trì và kết thúc các phiên.  **Ví dụ**: Giao thức RPC (Remote Procedure Call). |
| Tầng 4: Vận chuyển (Transport Layer) | **Chức năng**: Đảm bảo truyền tải dữ liệu giữa hai thiết bị. Cung cấp dịch vụ kết nối (TCP) và không kết nối (UDP).  **Ví dụ**: Giao thức TCP (Transmission Control Protocol), UDP (User Datagram Protocol). |
| Tầng 3: Mạng (Network Layer) | **Chức năng**: Quản lý định tuyến dữ liệu từ nguồn đến đích qua nhiều mạng. Phân đoạn và định địa chỉ IP.  **Ví dụ**: Giao thức IP (Internet Protocol). |
| Tầng 2: Liên kết dữ liệu (Data Link Layer) | **Chức năng**: Cung cấp giao tiếp tin cậy giữa các thiết bị trên cùng một mạng. Phát hiện và sửa lỗi, điều khiển luồng dữ liệu.  **Ví dụ**: Giao thức Ethernet, PPP (Point-to-Point Protocol). |
| Tầng 1: Vật lý (Physical Layer) | **Chức năng**: Xác định cách truyền tải dữ liệu qua các phương tiện vật lý (cáp, sóng vô tuyến, v.v.).  **Ví dụ**: Cáp Ethernet, tín hiệu điện hoặc ánh sáng. |

**Thay đổi ở Tầng 6: Trình diễn (Presentation Layer)**

* **Tình huống**: Khi đang họp thông qua google meet và muốn nâng cao hiệu suất mạng.
* **Mạng hiện tại**: đang sử dụng kết nối Wi-Fi nhưng gặp vấn đề về độ ổn định.

**Chuyển đổi sang Ethernet**:

* **Thực hiện**: Cắm cáp Ethernet vào máy tính.
* **Cấu hình**: Hệ thống tự động nhận diện và thiết lập kết nối Ethernet thay vì dùng Wifi.

### Tầng liên quan

* **Tầng 1 và 2** (Vật lý và Liên kết dữ liệu): thay đổi phương thức truyền tải từ sóng vô tuyến (Wi-Fi) sang cáp Ethernet. Việc này không ảnh hưởng đến các tầng trên.
* **Tầng 3** (Mạng): Vẫn tiếp tục định tuyến dữ liệu bằng địa chỉ IP mà không thay đổi.
* **Tầng 4** (Vận chuyển) và các tầng khác: Vẫn hoạt động bình thường, cho phép tham gia cuộc họp video mà không gặp trở ngại.

**Câu hỏi 5:** Cho ví dụ và phân tích một mô hình kiến trúc thuê bao/xuất bản (publish/subscribe).

Ví dụ 1: Thư từ viết tay. Thư được đóng gói vào phong bì (message) kèm theo thông tin địa chỉ người gửi và nhận (metadata). Sau đó gửi đến bưu điện (Message Broker). Người gửi ở đây là Publisher.

Tính bất đồng bộ thể hiện qua việc người gửi biết chắc chắn thư mình sẽ đến người kia bằng không cần quan tâm ông bưu điện (Message Broker) làm gì.

Trong khi bưu điện phải check: địa chỉ ở đâu, đóng gói ship tiếp qua đường nào (đường hàng không, đường thủy, đường bộ, …), qua rất nhiều bước, rồi cuối cùng đặt thư vào đúng địa chỉ hòm thư người nhận. Người nhận thư là Subscriber.

Ví dụ 2: Người dùng subscribe 1 kênh Youtube nào đó, sau khi chủ kênh đăng video mới lên hay thực hiện live stream người dùng sẽ nhận được thông báo.

Ví dụ trong lĩnh vực công nghệ: MessageQueue (RabbitMQ, Kafka), Redis Pub/sub, …

**Câu hỏi 6:** Phân tích ưu nhược điểm của kiến trúc tập trung và kiến trúc không tập trung.

|  |  |  |
| --- | --- | --- |
|  | Ưu điểm | Nhược điểm |
| Kiến trúc tập trung | **Quản lý dễ dàng**: Tất cả dữ liệu và ứng dụng được lưu trữ tại một vị trí, giúp dễ dàng quản lý, bảo trì và sao lưu.  **Bảo mật cao**: Dữ liệu tập trung giúp dễ dàng thực hiện các biện pháp bảo mật và kiểm soát truy cập.  **Hiệu suất cao**: Tất cả tài nguyên có thể được tối ưu hóa và phân phối hiệu quả từ một trung tâm, giảm thiểu độ trễ. **Chi phí thấp**: Ít cần thiết bị phần cứng hơn vì không cần nhiều máy chủ hoặc thiết bị lưu trữ. | **Điểm thất bại duy nhất**: Nếu máy chủ trung tâm gặp sự cố, toàn bộ hệ thống có thể bị gián đoạn.  **Tải nặng lên máy chủ**: Tất cả yêu cầu từ người dùng phải được xử lý bởi một máy chủ, có thể dẫn đến quá tải.  **Khó mở rộng**: Khi số lượng người dùng tăng lên, việc mở rộng hệ thống có thể trở nên khó khăn và tốn kém.  **Phụ thuộc vào kết nối mạng**: Người dùng cần kết nối liên tục với máy chủ để truy cập dịch vụ, điều này có thể gây ra vấn đề trong trường hợp mất kết nối. |
| Kiến trúc phi tập trung | **Độ tin cậy cao**: Không có điểm thất bại duy nhất; nếu một phần của hệ thống gặp sự cố, các phần khác vẫn hoạt động.  **Khả năng mở rộng tốt**: Dễ dàng thêm các nút mới vào hệ thống mà không làm gián đoạn hoạt động của các nút khác.  **Tính linh hoạt**: Các phần của hệ thống có thể hoạt động độc lập, cho phép tùy chỉnh và phát triển riêng biệt.  **Tăng cường hiệu suất**: Tải có thể được phân phối giữa nhiều máy chủ, giảm thiểu áp lực lên từng máy chủ đơn lẻ. | **Quản lý phức tạp**: Quản lý nhiều nút và dịch vụ có thể trở nên phức tạp và khó khăn hơn.  **Bảo mật khó khăn hơn**: Phân tán dữ liệu có thể làm cho việc bảo mật trở nên khó khăn hơn và cần nhiều biện pháp bảo vệ hơn.  **Chi phí cao**: Có thể cần nhiều phần cứng và phần mềm hơn để duy trì một hệ thống phân tán.  **Độ trễ cao hơn**: Giao tiếp giữa các nút có thể tạo ra độ trễ, đặc biệt trong các ứng dụng yêu cầu thời gian thực. |

**Câu hỏi 7:** Xét một chuỗi các tiến trình P1, P2, ..., Pn triển khai một kiến trúc client-server đa tầng. Cơ chế hoạt động của tổ chức đó như sau: tiến trình Pi là client của tiến trình Pi+1, và Pi sẽ trả lời Pi-1 chỉ khi đã nhận được câu trả lời từ Pi+1.

Vậy những vấn đề nào sẽ nảy sinh với tổ chức này khi xem xét hiệu năng yêu cầu-trả lời tới P1?

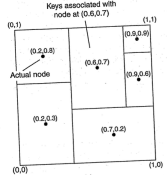
Vấn đề 1: Tăng độ trễ: Mỗi yêu cầu từ P1​ phải đi qua tất cả các tiến trình từ P2​ đến Pn​ trước khi nhận được câu trả lời. Điều này tạo ra một độ trễ tổng thể, vì mỗi tiến trình phải chờ đợi câu trả lời từ tiến trình tiếp theo.

Vấn đề 2: Đồng bộ và sự phụ thuộc: Xử lý đồng bộ dẫn đến tình trạng chờ đợi không cần thiết, nếu Pi trung gian bị treo, thì P1 đến Pi-1 phải đợi response. Nếu 1 tiến trình lỗi, tất cả tiến trình sẽ bị ảnh hưởng.

Vấn đề 3: Sự mở rộng: Khi thêm tiến trình, độ trễ và tải trọng sẽ tăng, gây ảnh hưởng tới hiệu suất.

**Câu hỏi 8:** Xét mạng CAN như trong hình. Giả sử tất cả các node đều biết node hàng xóm của mình. Một giải thuật định tuyến được đưa ra đó là gửi các gói tin cho node hàng xóm gần mình nhất và hướng đến đích. Giải thuật này có tốt không? Giải thích.

1



#### Ưu điểm

1. **Đơn giản**: Giải thuật này dễ hiểu và dễ triển khai. Các nút chỉ cần biết vị trí của các nút hàng xóm và có thể quyết định nhanh chóng nút nào để gửi gói tin.
2. **Tối ưu hóa băng thông**: Bằng cách chỉ gửi gói tin đến nút hàng xóm gần nhất, có thể giảm thiểu số lượng gói tin được truyền tải qua mạng, từ đó tiết kiệm băng thông.
3. **Giảm độ trễ**: Nếu các nút hàng xóm gần nhau, việc gửi gói tin đến nút gần nhất có thể dẫn đến việc truyền tải nhanh hơn.

#### Nhược điểm

1. **Không đảm bảo đường đi tối ưu**: Giải thuật này có thể không tìm được đường đi ngắn nhất đến đích. Việc gửi gói tin đến nút gần nhất không đảm bảo rằng gói tin sẽ đến đích trong thời gian ngắn nhất, vì có thể có những nút khác có đường đi ngắn hơn nhưng không được chọn.
2. **Vấn đề vòng lặp**: Nếu mạng có cấu trúc phức tạp, giải thuật này có thể dẫn đến việc gói tin bị gửi vòng quanh mà không đến được đích, đặc biệt trong trường hợp các nút hàng xóm không có đường đi trực tiếp đến đích.
3. **Tính khả thi trong mạng biến đổi**: Trong các mạng động, nơi các nút có thể tham gia hoặc rời khỏi mạng, giải thuật này có thể không hoạt động hiệu quả do thay đổi cấu trúc mạng.
4. **Chưa tối ưu cho các trường hợp phức tạp**: Trong các trường hợp mà các nút có nhiều kết nối và nhiều hướng đi có thể, giải thuật này có thể không đủ linh hoạt để tìm ra đường đi tốt nhất.

### Kết luận

Giải thuật định tuyến gửi gói tin cho nút hàng xóm gần nhất có những ưu điểm nhất định như đơn giản và tiết kiệm băng thông, nhưng cũng có nhiều nhược điểm nghiêm trọng như không đảm bảo đường đi tối ưu, có thể gây vòng lặp và không khả thi trong các mạng động. Do đó, để cải thiện hiệu suất, nên xem xét các giải thuật định tuyến phức tạp hơn, chẳng hạn như Dijkstra hoặc A\*, những giải thuật này có thể tìm ra đường đi tối ưu hơn trong mạng.