I. CAP THEOREM

1. Tổng quan về hệ thống phân tán

Hệ phân tán là tập hợp các máy tính được kết nối với nhau bởi một mạng máy tính và được cài đặt phần mềm hệ phân tán.

Mục đích:

- Nhu cầu ứng dụng: Tích hợp các ứng dụng riêng rẽ đang có
- Tích hợp các nguồn tài nguyên đang có
- Đưa tin học và các lĩnh vực ứng dụng mới

Hệ thống phân tán là một mạng lưu trữ dữ liệu trên nhiều node (máy vật lý hoặc máy ảo) cùng một lúc. Bởi vì tất cả các ứng dụng cloud (đám mây) đều là hệ thống phân tán, khi thiết kế ứng dụng đám mây, chúng ta cần phải hiểu định lý CAP để có thể lựa chọn hệ thống quản lý dữ liệu với các đặc điểm mà ứng dụng của chúng ta cần nhất

2. Cap Theorem

- ***Consistency (Tính nhất quán)***
  Tính nhất quán có nghĩa là tại cùng một thời điểm, dữ liệu mà tất cả các client nhìn thấy phải là giống nhau, bất kể nó kết nối với node nào. Để điều này xảy ra, bất cứ khi nào dữ liệu được ghi vào một node, nó phải được chuyển tiếp hoặc sao chép ngay lập tức tới tất cả các node khác trong hệ thống trước khi việc ghi được coi là "thành công".
- ***Availability (Tính khả dụng)***
  Tính khả dụng có nghĩa là bất kỳ client nào request dữ liệu đều nhận được phản hồi, ngay cả khi một hoặc nhiều node bị ngừng hoạt động. Hay nói cách khác — đối với bất kỳ yêu cầu nào, tất cả các node đang hoạt động trong hệ thống phân tán phải trả về phản hồi hợp lệ.

- ***Partition tolerance (Dung sai phân vùng)***
  Phân vùng là sự đứt gãy liên lạc trong hệ thống phân tán, hay cụ thể hơn, là việc kết nối giữa hai node bị mất hoặc tạm thời bị trì hoãn. Dung sai phân vùng có nghĩa là cluster phải duy trì được trạng thái hoạt động dù cho có bất kỳ sự cố giao tiếp nào giữa các node trong hệ thống.

=> **Một hệ thống phân tán chỉ có thể có được hai trong ba đặc tính mong muốn**

3. Cơ sở dữ liệu

- ***Cơ sở dữ liệu CP***: Một cơ sở dữ liệu CP có tính nhất quán và dung sai phân vùng, tức hi sinh tính khả dụng. Khi tình trạng phân vùng xảy ra giữa hai node bất kỳ, hệ thống phải shutdown node gặp tình trạng không nhất quán (tức là làm cho nó không khả dụng) cho đến khi việc phân vùng được giải quyết.
- ***Cơ sở dữ liệu AP***: Cơ sở dữ liệu AP cung cấp tính khả dụng và khả năng chịu phân vùng, tức hi sinh tính nhất quán. Khi tình trạng phân vùng xảy ra, tất cả các node vẫn sẽ khả dụng nhưng sẽ có những node chứa phiên bản dữ liệu cũ hơn những node khác. (Khi tình trạng phân vùng được giải quyết,

cơ sở dữ liệu AP thường đồng bộ lại các node để đồng bộ lại tất cả các điểm không nhất quán trong hệ thống.)

- ***Cơ sở dữ liệu CA***: Cơ sở dữ liệu CA cung cấp tính nhất quán và tính khả dụng trên tất cả các node. Tuy nhiên, nó không thể thực hiện được điều này nếu xảy ra tình trạng phân vùng giữa hai node bất kỳ trong hệ thống và do đó không có khả năng chịu lỗi (fault tolerance).

4. Một số Database

- ***MongoDB:***
    - là một hệ quản trị cơ sở dữ liệu NoSQL
    - MongoDB là một kho lưu trữ dữ liệu CP — nó giải quyết các phân vùng mạng bằng cách duy trì tính nhất quán, nhưng không đảm bảo tính khả dụng.

    - MongoDB là một hệ thống single-master — mỗi tập hợp bản sao (replica set) chỉ có thể có một node chính tiếp nhận tất cả các thao tác ghi. Tất cả các node khác trong cùng một tập hợp bản sao là các node thứ cấp log lại nhật ký hoạt động của node chính và apply các hoạt động đó vào tập dữ liệu riêng của chúng. Mặc định, các client cũng đọc từ node chính, nhưng cũng có thể chỉ định tùy chọn đọc để đọc từ các node phụ.

    - Khi node chính không khả dụng, node phụ có nhật ký hoạt động gần đây nhất sẽ được chọn làm node chính mới. Khi tất cả các node phụ khác bắt kịp với node chính mới, cụm sẽ khả dụng trở lại. Vì các máy khách không thể thực hiện bất kỳ yêu cầu ghi nào trong khoảng thời gian này, nên dữ liệu vẫn nhất quán trên toàn bộ mạng.

- ***Cassandra***
    - là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở
    - Cassandra là một cơ sở dữ liệu AP — cung cấp tính khả dụng và dung sai phân vùng nhưng không thể cung cấp tính nhất quán mọi lúc. Bởi vì Cassandra không có node chính, tất cả các node phải có sẵn liên tục. Tuy nhiên, Cassandra cung cấp tính nhất quán cuối cùng bằng cách cho phép máy khách ghi vào bất kỳ node nào vào bất kỳ lúc nào và điều chỉnh các conflict càng nhanh càng tốt.

    - Vì dữ liệu chỉ trở nên không nhất quán trong trường hợp mạng bị phân vùng, và sự không nhất quán cũng được giải quyết nhanh chóng, Cassandra cung cấp chức năng “sửa chữa” để giúp các node bắt kịp với các node khác. Tuy nhiên, tính khả dụng liên tục đem lại một hệ thống có hiệu suất cao, và đây là một điều rất có giá trị, đáng để đánh đổi trong nhiều trường hợp.

5. Use Cases

Note:

- **Consistency**: Trong hệ thống giao dịch ngân hàng, tính nhất quán dữ liệu giao dịch là tối quan trọng, vì bất kỳ sai sót nào cũng có thể dẫn đến tổn thất tài chính.
- **Availability**: Đối với một trang web thương mại điện tử, việc ngừng hoạt động trong vài phút có thể dẫn đến tổn thất doanh thu đáng kể.
- **Partition tolerance** : Một ứng dụng phân tán được triển khai trên nhiều quốc gia có thể gặp phải tình trạng phân vùng mạng do gián đoạn kết nối internet.

Ví dụ:

- Ứng dụng cần sự nhất quán của dữ liệu: trong các ứng dụng thương mại điện tử hoặc dịch vụ thanh toán -> chọn PostgreSQL
- Ứng dụng cần khả năng duyệt nhanh chóng mô hình dữ liệu và scale theo chiều ngang, và có thể chấp nhận được tính nhất quán cuối cùng, thì có thể chọn cơ sở dữ liệu AP như Cassandra hoặc Apache CouchDB. For example, Social Networking Applications: Scenario: A social media platform needs to handle a massive amount of data like posts, comments, and user profiles. Users expect high availability to access and share content in real-time.

6. **Summary**

   Định lý CAP (Consistency, Availability, Partition Tolerance) là một nguyên tắc cơ bản trong khoa học máy tính phân tán, khẳng định rằng một hệ thống phân tán chỉ có thể đảm bảo tối đa hai trong ba thuộc tính:

   C (Consistency - Tính nhất quán): Mọi truy vấn đọc đều phải trả về phiên bản dữ liệu mới nhất đã được ghi.
   A (Availability - Tính sẵn sàng): Hệ thống luôn sẵn sàng để phục vụ các yêu cầu đọc và ghi.
   P (Partition Tolerance - Dung sai phân vùng): Hệ thống vẫn hoạt động bình thường ngay cả khi xảy ra lỗi phân vùng mạng, khiến một số nút trong hệ thống không thể giao tiếp với nhau.

7. **Tham khảo**

   https://viblo.asia/p/gioi-thieu-ve-dinh-ly-cap-gDVK2ONmZLj

II. MESSAGE QUEUE

1. Định nghĩa

Microservices là một kiến trúc phần mềm xây dựng ứng dụng từ các dịch vụ nhỏ, độc lập và tương tác với nhau thông qua các giao diện được định nghĩa rõ ràng. Mỗi dịch vụ có trách nhiệm cụ thể, dễ dàng phát triển, triển khai và bảo trì.

Message Queue (Hàng đợi tin nhắn) là một hệ thống trung gian truyền tải tin nhắn bất đồng bộ giữa các ứng dụng hoặc dịch vụ. Nó hoạt động như một hộp thư bưu điện, nơi các dịch vụ gửi tin nhắn và các dịch vụ khác nhận và xử lý chúng.

Message queue là một kiến trúc cung cấp giao tiếp không đồng bộ. Ý nghĩa của queue ở đây chính là 1 hàng đợi chứa message chờ để được xử lý tuần tự theo cơ chế vào trước thì ra trước (FIFO - First In First Out). Một message là các dữ liệu cần vận chuyển giữa người gửi và người nhận.

2. **Cơ chế**
    a) ***Kiến trúc***
    - Message: Thông tin được gửi (có thể là text, binary hoặc JSON)
    - Producer: Service tạo ra thông tin, đưa thông tin vào message queue.
    - Message Queue: Nơi chứa những message này, cho phép producer và consumer có thể trao đổi với nhau
    - Consumer: Service nhận message từ message queue và xử lý
    - Một service có thể vừa làm producer, vừa làm consumer

    b) ***Cách hoạt động***
    Giao tiếp được thực hiện bằng cách gửi tin nhắn có chứa thông tin hoặc lệnh cần được xử lý. Người gửi là Producer. Những tin nhắn này sau đó được đưa vào trong một hàng đợi và được xử lý bởi một microservice khác (Consumer). Sau đó, khi một tin nhắn được xử lý, nó sẽ bị xóa hoặc vô hiệu hóa, điều này đảm bảo rằng nó chỉ được xử lý một lần duy nhất.
3. **Một số loại**
    RabbitMQ: Một Message Queue mã nguồn mở, mạnh mẽ và linh hoạt.
    Kafka: Một Message Queue có khả năng mở rộng cao, phù hợp cho xử lý luồng dữ liệu lớn.
    Amazon SQS: Một Message Queue được quản lý hoàn toàn, dễ sử dụng và tích hợp tốt với các dịch vụ AWS khác.
4. **Use Cases**
    - **E-commerce**: Khi khách hàng đặt hàng, một dịch vụ microservice có thể gửi tin nhắn đến dịch vụ khác để xử lý thanh toán, dịch vụ khác để cập nhật kho hàng, và dịch vụ khác để gửi email xác nhận.
    - **Social Media**: Khi người dùng đăng bài viết mới, một dịch vụ microservice có thể gửi tin nhắn đến dịch vụ khác để lưu trữ bài viết, dịch vụ khác để thông báo cho người theo dõi, và dịch vụ khác để phân tích nội dung bài viết.
    - **Financial Application** : Khi khách hàng chuyển tiền, một dịch vụ microservice có thể gửi tin nhắn đến dịch vụ khác để cập nhật số dư tài khoản, dịch vụ khác để gửi thông báo giao dịch, và dịch vụ khác để kiểm tra gian lận.

III. PROMPTING ENGINEER

1. **Definition**
    Prompt (lời nhắc): là đầu vào hoặc truy vấn được cung cấp cho LLM để thu được một phản hồi cụ thể từ mô hình. Lời nhắc có thể là một câu hoặc câu hỏi bằng ngôn ngữ tự nhiên, hoặc kết hợp thêm cả các đoạn code, tùy thuộc vào lĩnh vực và nhiệm vụ
    Prompt engineering: còn được gọi là Prompt Design, là các phương pháp giao tiếp với LLM để điều khiển hành vi của nó nhằm đạt được kết quả mong muốn mà không cần cập nhật trọng số mô hình.
2. **Basic Techniques**
    a) ***Đưa chỉ dẫn (instruction prompting)***

Đưa ra hướng dẫn những gì muốn LLM thực hiện hoặc một số quy tắc, quy định buộc phải tuân theo.

b) ***Role Prompting***

Gán vai trò (role) cho AI. Ví dụ: Prompt bắt đầu như "You are a doctor" hoặc "Act as a prominent lawyer" và sau đó yêu cầu AI trả lời một số câu hỏi về y tế hoặc pháp lý. -> Cung cấp cho AI một số bối cảnh giúp hiểu câu hỏi tốt hơn và ta cũng phần nào đó định hình được cách mà nó phản hồi

c) ***Cung cấp ví dụ (In-context learning)***

Đưa ra một tập hợp các ví dụ tiêu biểu, chất lượng cao về nhiệm vụ cần được thực hiện, mỗi ví dụ bao gồm cả input và output mong muốn

d) ***Chain of Thought***

Yêu cầu model suy nghĩ theo từng bước, giải thích từng bước cách mà nó đưa ra câu trả lời.

3. **Advance Technique**

a) ***Generated Knowledge Prompting***

Yêu cầu LLM tạo ra nhiều câu hỏi lời cho cùng 1 câu hỏi, sau đó sẽ đưa 10 đáp án nhận được vào context và yêu cầu LLM lựa chọn phương án phù hợp và đúng đắn nhất

b) ***Tree of thoughts***

Một kĩ thuật phát triển từ chain-of-thoughts, nhưng thay vì chỉ có 1 luồng suy nghĩ, bạn sẽ yêu cầu LLM phải suy luật theo nhiều luồng suy nghĩ và loại bỏ những tư duy sai lầm. Điều này sẽ khắc phục được vấn đề khi suy luận của LLM sai trong chain-of-thoughts sẽ khiến toàn bộ kết quả bị sai

Ví dụ:

Yêu cầu GPT tưởng tượng ra một cuộc họp có 10 chuyên gia trong lĩnh vực và yêu cầu họ suy luận và đưa ra cách giải quyết, bất cứ người nào nhận ra sai lầm trong suy luận của họ sẽ phải tự giác rời đi. Cuối cùng, người ở lại cuối cùng sẽ đưa ra đáp án của họ.

c) ***RAG***

Cho phép LLM truy cập vào một lượng lớn dữ liệu bên ngoài mà LLM chưa từng được học trước đó.

4. **Tránh bẫy với prompt engineering**

- Hallucination

  Prompt: “trả lời trung thực dựa trên các thông tin XYZ sau", "nếu không biết thì hãy nói là tôi không biết"

5. **Tham khảo**

https://viblo.asia/p/ban-da-biet-gi-ve-prompt-engineering-tong-hop-cac-tips-tao-loi-nhac-cho-chatgpt-PwlVmxrgL5Z

https://viblo.asia/p/tat-tan-tat-nhung-ki-thuat-prompt-engineering-huu-ich-nhat-cho-chatgpt-bXP4WzmqV7G