**Model Context Protocol (MCP)**

Giao thức Ngữ cảnh Mô hình (MCP) được phát triển bởi Anthropic và ra mắt vào cuối năm 2024, là một giao thức mở nhằm kết nối các mô hình trí tuệ nhân tạo (AI) với các nguồn dữ liệu và công cụ một cách dễ dàng và hiệu quả. MCP giúp đơn giản hóa quy trình tích hợp AI bằng cách tiêu chuẩn hóa các tương tác, hỗ trợ giao tiếp hai chiều và khả năng mở rộng dễ dàng hơn so với các API truyền thống. Ngoài ra, MCP mở ra nhiều tiềm năng ứng dụng trong các lĩnh vực như quản lý công việc, robot học và hỗ trợ cá nhân hóa trong các hệ thống AI.

**Lý do tạo ra MCP:**

MCP (Model Context Protocol) được phát triển nhằm cải thiện khả năng truy cập và tích hợp của các mô hình AI, đặc biệt là các mô hình ngôn ngữ lớn (LLMs), với các công cụ và cơ sở dữ liệu bên ngoài. Hiện tại, việc tích hợp AI với hệ thống bên ngoài gặp nhiều khó khăn, bao gồm:

* **Tích hợp API tùy chỉnh:** Mỗi công cụ (như CRM, lưu trữ đám mây, hệ thống bán vé) cần một tích hợp API riêng biệt.
* **Xác thực phức tạp:** Cần thiết lập xác thực qua API như OAuth và khóa API.
* **Định dạng dữ liệu thủ công:** Phải xử lý dữ liệu để nó có thể sử dụng với các mô hình AI.
* **Quản lý giới hạn tốc độ và lỗi:** Khó khăn trong việc quản lý nhiều dịch vụ khác nhau.

Cách tiếp cận hiện tại không mở rộng được, vì mỗi tích hợp mới đều đòi hỏi lập trình tùy chỉnh và bảo trì phức tạp. MCP hướng tới việc xác định một giao thức chung, giúp các mô hình AI truy cập dữ liệu một cách hiệu quả hơn mà không cần phát triển cầu nối API tùy chỉnh cho mỗi hệ thống.

**MCP hoạt động như thế nào?**

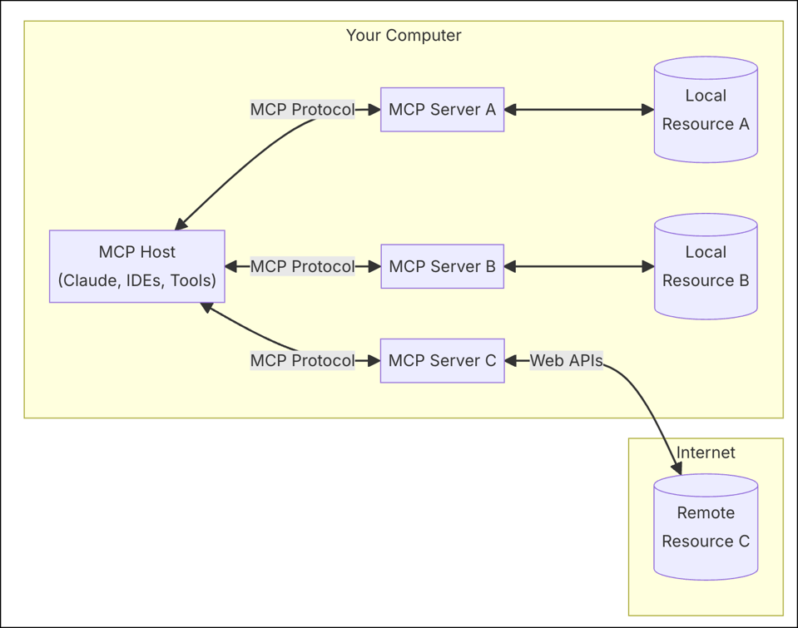
Ngày nay, các tác nhân AI dựa vào các lệnh gọi API tùy chỉnh, xác thực theo từng dịch vụ và phân tích phản hồi thủ công, tạo ra một mạng lưới tích hợp mong manh, khó mở rộng.

Thay vì buộc các tác nhân AI phải tương tác với API một cách riêng biệt, MCP thiết lập một giao thức thống nhất giúp trừu tượng hóa sự phức tạp của xác thực, thực hiện yêu cầu và định dạng dữ liệu, cho phép các hệ thống AI tập trung vào lý luận thay vì logic tích hợp cấp thấp.

**Kiến trúc máy khách-máy chủ của MCP**

MCP (Model Context Protocol) được xây dựng dựa trên mô hình máy khách-máy chủ, giúp các mô hình AI tương tác hiệu quả với các nguồn dữ liệu bên ngoài:

* **Máy khách MCP:** Bao gồm các tác nhân AI, ứng dụng hoặc hệ thống cần dữ liệu có cấu trúc.
* **Máy chủ MCP:** Đóng vai trò trung gian, chịu trách nhiệm lấy dữ liệu từ nhiều API, cơ sở dữ liệu và hệ thống doanh nghiệp khác nhau, sau đó trả về dữ liệu theo định dạng thống nhất.



Thay vì để các mô hình AI thực hiện các yêu cầu API trực tiếp, máy chủ MCP xử lý các yếu tố phức tạp như xác thực, truy xuất dữ liệu và chuẩn hóa phản hồi. Điều này giúp các tác nhân AI không còn phải quản lý nhiều thông tin xác thực API, các định dạng yêu cầu khác nhau hoặc các cấu trúc phản hồi không đồng nhất.

**Ví dụ:** Khi một mô hình AI cần thông tin từ Google Drive, Slack và một cơ sở dữ liệu, nó chỉ cần gửi một yêu cầu có cấu trúc duy nhất đến máy chủ MCP. Máy chủ sẽ xử lý yêu cầu, thu thập dữ liệu từ các nguồn cần thiết và trả về phản hồi đã được tổ chức tốt.

**Vòng đời Yêu cầu-Phản hồi của MCP**

Tương tác MCP điển hình tuân theo chu trình yêu cầu-phản hồi có cấu trúc giúp loại bỏ các lệnh gọi API dư thừa và chuẩn hóa việc truy xuất dữ liệu.

1. Tác nhân AI gửi yêu cầu có cấu trúc đến máy chủ MCP. Thay vì tạo các yêu cầu API riêng lẻ, tác nhân xác định dữ liệu cần thiết theo định dạng thống nhất.

2. Máy chủ MCP xử lý yêu cầu bằng cách xác thực, kiểm tra quyền và xác định hệ thống bên ngoài nào cần truy vấn.

3. Các truy vấn được thực hiện song song, nghĩa là dữ liệu từ nhiều dịch vụ được truy xuất cùng lúc thay vì tuần tự, giúp giảm độ trễ tổng thể.

4. Phản hồi từ nhiều nguồn khác nhau được chuẩn hóa thành định dạng có cấu trúc mà mô hình AI có thể dễ dàng xử lý.

Không giống như phản hồi API thô yêu cầu phân tích thủ công, MCP đảm bảo rằng tất cả dữ liệu được truy xuất đều tuân theo định dạng có cấu trúc, có thể dự đoán được, giúp các mô hình AI dễ hiểu và sử dụng hơn.

**Thực hiện truy vấn và tổng hợp phản hồi**

MCP được thiết kế để tối ưu hóa tương tác của các mô hình AI với hệ thống bên ngoài thông qua quy trình thực thi có cấu trúc. Quy trình này bao gồm các bước sau:

1. **Xác thực yêu cầu:** Đảm bảo mô hình AI có đủ quyền truy cập cần thiết trước khi lấy bất kỳ dữ liệu nào.
2. **Định tuyến truy vấn:** Xác định các dịch vụ bên ngoài nào cần được truy cập để thực hiện truy vấn.
3. **Thực thi song song:** Truy xuất dữ liệu từ nhiều nguồn đồng thời, giúp giảm độ trễ so với việc thực hiện các yêu cầu API theo thứ tự.
4. **Tổng hợp phản hồi:** Hợp nhất dữ liệu có cấu trúc từ nhiều nguồn thành một phản hồi duy nhất. Điều này loại bỏ nhu cầu mà các mô hình AI phải xử lý thủ công nhiều đầu ra thô từ API.

A diagram of a service

AI-generated content may be incorrect.

Thông qua việc giảm thiểu các yêu cầu trùng lặp, chuẩn hóa phản hồi và tập trung xử lý xác thực, MCP giúp giảm chi phí API không cần thiết và làm cho việc tự động hóa do AI điều khiển trở nên khả thi hơn và có khả năng mở rộng tốt hơn.

**Hạn chế của Giao thức Ngữ cảnh Mô hình (MCP):**

1. **Giới hạn về bộ nhớ và tài nguyên tính toán:**

Lưu trữ và duy trì ngữ cảnh lâu dài yêu cầu bộ nhớ lớn. Ngữ cảnh quá dài có thể dẫn đến việc mô hình lãng quên thông tin cũ hoặc quá tải bộ nhớ.

1. **Quản lý và cập nhật ngữ cảnh chưa tối ưu:**

Việc cập nhật ngữ cảnh liên tục có thể làm mất thông tin quan trọng. Cập nhật không đúng cách có thể dẫn đến câu trả lời mâu thuẫn hoặc không chính xác.

1. **Độ trễ khi xử lý ngữ cảnh lớn:**

Xử lý một lượng lớn ngữ cảnh có thể gây ra độ trễ cao trong các ứng dụng thời gian thực, ảnh hưởng đến trải nghiệm người dùng.

1. **Giới hạn trong việc hiểu ngữ cảnh phức tạp:**

MCP không phải lúc nào cũng hiểu đúng ý định của người dùng, đặc biệt khi chuyển đổi giữa các chủ đề mà không có phân loại ngữ cảnh rõ ràng.

1. **Khó khăn trong bảo mật và quyền riêng tư:**

Việc lưu trữ ngữ cảnh từ các cuộc hội thoại hoặc truy xuất dữ liệu làm tăng nguy cơ rò rỉ thông tin cá nhân, đặc biệt khi xử lý dữ liệu nhạy cảm.

1. **Thiếu tiêu chuẩn chung:**

MCP vẫn thiếu một chuẩn chung để áp dụng trên nhiều hệ thống khác nhau, dẫn đến sự không tương thích giữa các mô hình và cách triển khai khác nhau.

**So Sánh MCP và API Truyền Thống:**

MCP mang lại một cách tiếp cận mới mẻ và hiệu quả hơn cho việc tích hợp AI so với các API truyền thống. Sự tiêu chuẩn hóa, khả năng khám phá động và giao tiếp hai chiều của MCP giúp đơn giản hóa quá trình phát triển, tăng cường tính linh hoạt và mở ra nhiều tiềm năng ứng dụng sáng tạo

|  |  |  |
| --- | --- | --- |
| **Khía cạnh** | **API Truyền Thống** | **MCP** |
| **Kiến trúc** | **Monolithic** – Một hệ thống lớn xử lý tất cả mọi thứ (ví dụ: đăng nhập người dùng, dữ liệu, thanh toán) trong một đơn vị duy nhất. | **Microservices** – Các dịch vụ nhỏ, độc lập, mỗi dịch vụ thực hiện một nhiệm vụ cụ thể (ví dụ: một cho đăng nhập, một cho thanh toán). |
| **Khả năng mở rộng** | **Khó mở rộng** – Bạn phải mở rộng toàn bộ hệ thống cùng một lúc, ngay cả khi chỉ có một phần cần nhiều tài nguyên hơn. | **Dễ mở rộng** – Bạn có thể mở rộng các dịch vụ cá nhân khi cần mà không ảnh hưởng đến phần còn lại. |
| **Giao thức** | Thường sử dụng **SOAP** – Một giao thức cũ, phức tạp hơn có thể nặng nề và khó sử dụng. | Sử dụng các giao thức hiện đại như **REST** hoặc **GraphQL** – Nhẹ nhàng, nhanh hơn và dễ sử dụng hơn. |
| **Quản lý** | **Quản lý thủ công** – Các lập trình viên phải tự xử lý những nhiệm vụ như bảo mật và định tuyến, điều này tốn thời gian. | **Tự động với các cổng API** – Xử lý bảo mật, định tuyến và các nhiệm vụ khác tự động, tiết kiệm thời gian và công sức. |
| **Linh hoạt** | **Ít linh hoạt** – Những thay đổi có thể ảnh hưởng đến toàn bộ hệ thống, vì vậy cập nhật là rủi ro và cần lên kế hoạch cẩn thận. | **Rất linh hoạt** – Bạn có thể cập nhật một dịch vụ mà không làm ảnh hưởng đến những dịch vụ khác, làm cho việc thay đổi nhanh chóng và an toàn hơn. |
| **Triển khai** | **Cần triển khai toàn bộ ứng dụng** – Ngay cả những cập nhật nhỏ cũng có nghĩa là phải triển khai lại mọi thứ, có thể gây ra thời gian chết. | **Triển khai cập nhật cho các dịch vụ cá nhân** – Bạn có thể cập nhật một phần mà không làm ảnh hưởng đến phần còn lại, giảm thời gian chết. |
| **Cách ly lỗi** | **Một sự cố có thể ảnh hưởng đến toàn bộ hệ thống** – Nếu một phần bị hỏng, nó có thể làm hỏng toàn bộ API. | **Các lỗi được cách ly** – Nếu một dịch vụ bị lỗi, các dịch vụ khác vẫn hoạt động, ngăn ngừa các vấn đề rộng lớn. |