MCP

MCP基本原理

本质

本质是对functioncalling进行了一层封装，对外暴露特定装饰器，方便调用

MCP目的

旨在统一LLM应用与外部数据源和不同语言的工具之间的通信协议，为AI开发提供了标准化的上下文交互方式。

选择MCP的原因

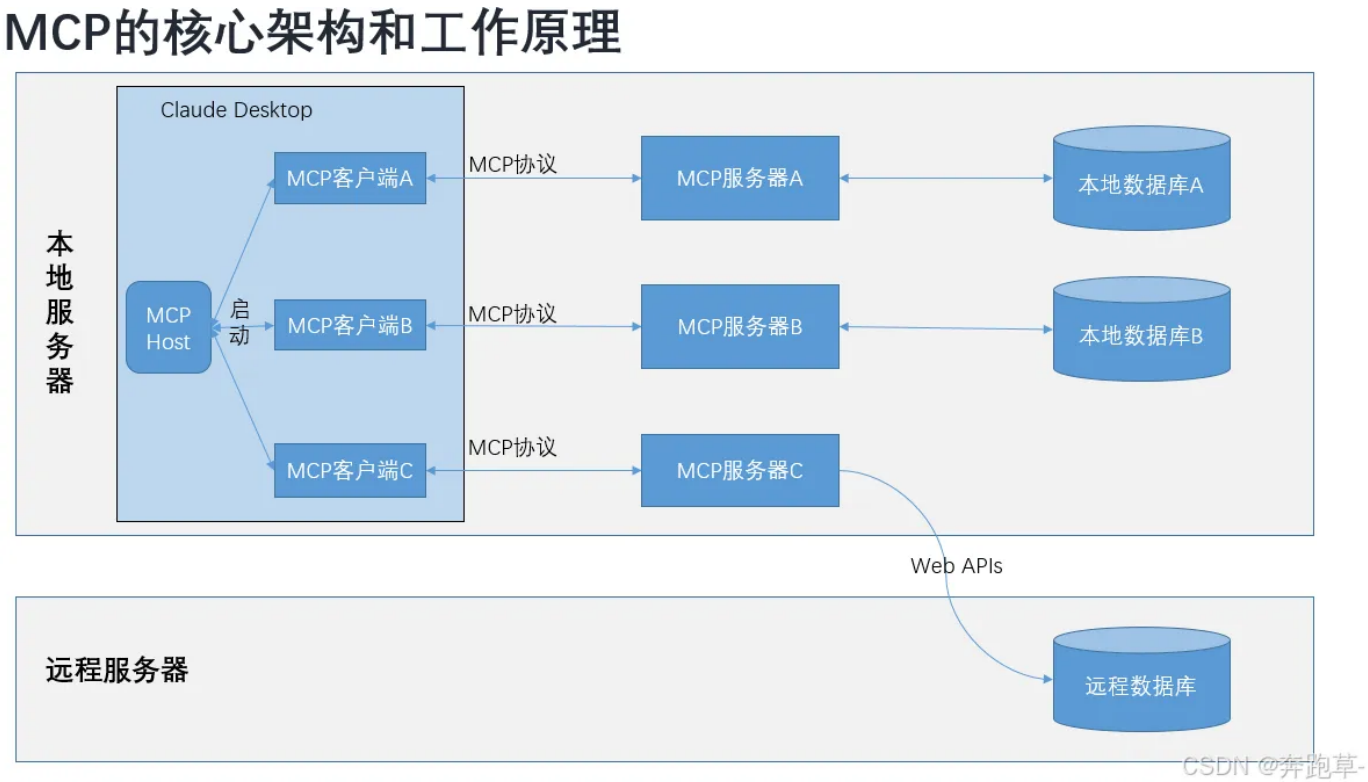
MCP 可帮助您在 LLM 之上构建代理和复杂的工作流。LLM 通常需要与数据和工具集成，而 MCP 可提供：

- 您的 LLM 可以直接插入不断增加的预构建集成列表，就是可以调用现有的mcpserver，并且更利于扩展功能
- 在 LLM 提供商和供应商之间切换的灵活性，接口统一，所以替换模型比较方便
- 保护基础架构内数据的最佳实践
- 实现不同语言工具的统一集成

架构与工作原理

MCP的工作原理可以概括为三个步骤：

- 首先，通过调用聊天完成API，将函数和用户输入传递给MCP服务器；
- 其次，使用模型的响应来调用API或函数；
- 最后，再次调用聊天完成API，包括从函数获得的响应，以得到最终响应。



- **MCP 主机**：希望通过 MCP 访问数据的程序，例如 Claude Desktop、IDE 或 AI 工具
- **MCP 客户端**：与服务器保持 1:1 连接的协议客户端
- **MCP 服务器**：轻量级程序，每个程序都通过标准化模型上下文协议公开特定功能
- **本地数据源**：MCP 服务器可以安全访问的您的计算机文件、数据库和服务
- **远程服务**：MCP 服务器可通过互联网（例如通过 API）连接到的外部系统

工作流程

MCP的工作流程通常包括以下步骤：

初始化：主机应用程序启动并初始化客户端，每个客户端与一个服务器建立连接。

功能协商：客户端和服务器之间进行功能协商，确定它们可以相互提供哪些功能和服务。

请求处理：客户端根据用户请求或AI模型的需要，向服务器发送请求。服务器处理这些请求，并可能与本地或远程资源进行交互。

响应返回：服务器将处理结果返回给客户端，客户端再将信息传递回主机应用程序。

MCP通信原理

MCP的连接机制遵循客户端-服务器架构。在这种架构中，MCP Clients与MCP Servers之间建立一对一的连接。

这种设计允许MCP Hosts通过MCP Clients与一个或多个MCP Servers进行通信，以获取数据和执行任务。

MCP支持两种类型的通信机制：

标准输入输出（Stdio）：

适用于本地进程间通信，其中Client启动Server程序作为子进程，消息通讯通过stdin/stdout进行，消息格式为JSON-RPC 2.0。

服务器发送事件（SSE）：

用于基于HTTP的通信，允许服务器向客户端推送消息，而客户端到服务器的消息传递则使用HTTP POST，同样采用JSON-RPC 2.0格式进行消息交换

MCP 提供了一个用于 HTTP 的授权框架。使用基于 HTTP 的传输的实现**应**遵循此规范，而使用 STDIO 传输的实现**不应**遵循此规范，而是从环境中检索凭据。

MCPServer&Client

现在已经有很多开发完善的Server，在不同的IDE上可以直接下载使用，比如cursor内部支持MCP协议，VScode可以下载插件Cline，其中也支持很多现有的Server

自定义Server

SDK

主要通过SDK（开发工具包）来构建自己的server，在MCP官网上提供了**Python、TypeScript、Java、Kotlin** 四种语言的 SDK（软件开发工具包），但是目前并没有一个完善的PHP-SDK，不过存在一个开源的PHP-SDK不过比较冷门不够完善，是基于python的sdk改的

https://github.com/logiscape/mcp-sdk-php

https://modelcontextprotocol.io/quickstart/server

https://github.com/modelcontextprotocol/python-sdk

自定义Client

同样SDK一般包含两者的开发，只要client和server端的接口都服从mcp的统一标准，就可以实现不同语言代码的调用。现在最常见的是client支持连接 Python 和 Node.js 服务器，不过比如很多开源项目在实现不同语言的服务器，比如上面的php-sdk就是基于SDK改的