总结来说，MCP 的工作流程可以概括为：1️⃣ **AI 客户端发现服务器能力** -> 2️⃣ **AI 模型基于用户查询和可用能力决定使用哪些工具/资源/提示** -> 3️⃣ **客户端向服务器发送请求** -> 4️⃣ **服务器执行操作并返回结果** -> 5️⃣ **客户端将结果提供给 AI 模型以生成最终响应**。

整个过程通过标准化的协议进行通信，使得不同的 AI 模型和各种外部工具及数据源能够以一种通用且灵活的方式进行集成。这种标准化降低了集成复杂性，提高了互操作性，并为构建更强大的 AI 应用奠定了基础。

5. MCP 的关键优势

MCP 为 AI 系统开发带来了多项显著优势:

- **降低集成复杂性** : MCP 提供了一个统一的接口，消除了为每个 LLM 驱动的应用程序集成外部功能而实施其自身方法的必要性。通过标准化的协议，开发者可以更轻松地将 AI 模型连接到各种工具和数据源。它将一个 M×N 问题（M 个 AI 模型与 N 个工具/数据源的集成）转化为一个 M+N 的解决方案，大大减少了集成所需的自定义代码和适配器。
- **加快开发速度**: 开发者可以利用预构建的 MCP 客户端和服务器，从而节省开发自定义集成的时间和精力。工具和 API 开发者只需要构建一个 MCP 服务器，就可以被所有兼容的 MCP 客户端使用。
- **提高可扩展性和可靠性**: 通过标准化的架构，MCP 有助于构建更健壮和可扩展的 AI 系统。维护上下文跨工具也变得更加容易，因为交互共享一个通用的框架。
- **实现供应商无关的开发**: MCP 作为一个开放标准，不限制开发者使用特定的 AI 提供商的生态系统或工具链。任何支持 MCP 的 AI 客户端（例如 Claude 或开源 LLMs）都可以使用任何兼容的 MCP 服务器。
- **增强 AI Agent的上下文感知能力**: MCP 的出现能够帮助上下文层实现最好的效果。由于上下文层通常需要开发者自己进行定义，而 MCP 作为一个开源协议，能够最好地发挥社区的力量来加速这个过程。
- **实现“万能应用”的潜力**: 通过合适的 MCP 服务器，用户可以将每个 MCP 客户端变成一个“万能应用”。

6. MCP 与 API 对比

- **定义**
  - **MCP**：MCP 是一种标准化协议，旨在为大型语言模型（LLM）提供上下文信息，以便它们能够更有效地与外部工具和数据源进行交互。它强调数据的上下文环境和语义背景，使得模型能够理解和处理复杂的结构化数据。
  - **API**：API 是一组规则和工具，允许不同的软件应用程序之间进行通信。它定义了请求和响应的格式，使开发者能够集成不同服务的功能，而无需了解其内部实现细节。

- **主要区别**

方面	Model Context Protocol (MCP)	API
核心目标	提供带有上下文的数据传输机制，确保接收方理解信息	提供接口以访问特定功能或数据集，简化应用间互动
通信方式	支持双向、实时、有状态的通信	通常为请求-响应模式，无状态通信
动态发现	支持动态工具发现和适应	通常需要预定义接口，缺乏动态发现能力
集成复杂度	单一标准化集成，简化多工具连接	每个API需单独集成，增加开发复杂度
上下文管理	增强的上下文感知和管理	有限的上下文管理能力
互操作性	模型无关，旨在成为通用标准	通常是供应商特定的

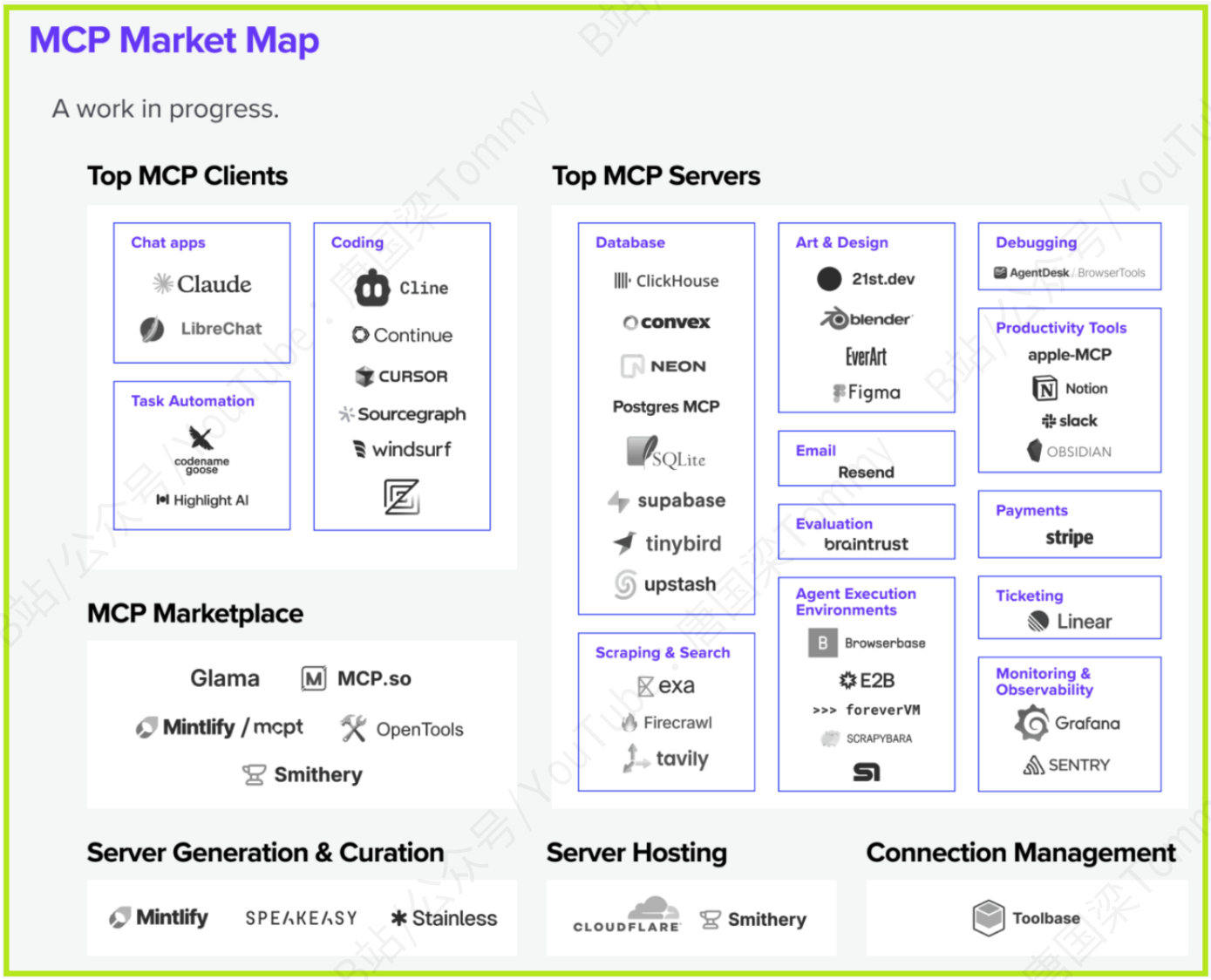
- **应用场景**
  - **MCP 的应用场景**：
    - 适用于需要多种工具和数据源协同工作的复杂 AI 应用，例如智能助手、数据分析平台等。
    - 支持实时数据交互和动态工具调用，使 AI 模型能够根据上下文变化灵活调整行为。
  - **API 的应用场景**：
    - 常用于简单的服务集成，如获取天气信息、支付处理等。
    - 在传统企业系统中广泛使用，用于实现各个模块之间的功能调用。

7. MCP 生态系统

围绕 MCP 正在迅速形成一个生态系统：

- **MCP Clients**: 目前高质量的 MCP 客户端大多以编程为中心。例如，Cline是一款开源的VSCode插件，它通过与MCP协议的结合，使得开发者能够轻松扩展AI的功能，甚至创建完全自定义的智能体。Claude Desktop本身也是一个 MCP 客户端，允许用户连接和使用各种 MCP 服务器。未来预计会出现更多面向业务的 MCP 客户端。
- **MCP Servers**: MCP 服务器的数量正在快速增长，涵盖了各种用例，例如访问本地文件系统、查询数据库、发送电子邮件、生成图像、与设计工具（如 Figma 和 Blender）交互、控制音乐软件（如 Ableton Live）以及进行网络搜索。许多头部数据库公司、编码公司和初创公司都在开发自己的服务器。GitHub 上已经有超过 2000 个 MCP 服务器，其中最常见的使用场景是搜索和数据检索。
- **MCP Marketplace**: 随着 MCP 服务器数量的增加，一个统一的 MCP 市场可能会出现，允许 AI 代理根据速度、成本和相关性等因素选择合适的服务器。

- **MCP Infra**: 围绕 MCP 的基础设施正在发展，包括服务器生成工具（如 Mintlify, Stainless, Speakeasy）以降低创建 MCP 兼容服务的摩擦，以及托管解决方案（如 Cloudflare, Smithery）以解决部署和扩展挑战。连接管理平台（如 Toolbase）也开始简化本地优先的 MCP 密钥管理和代理。目标是让 MCP 更可靠、更易于生产环境部署和更具可扩展性。



8. MCP 的挑战与限制

尽管 MCP 取得了显著的进展，但在构建和使用 MCP 时仍然存在一些未解决的问题:

- **缺乏内置的工作流概念**: 大多数 AI 工作流程需要按顺序进行多次工具调用，但 MCP 缺乏管理这些步骤的内置工作流概念。
- **标准化客户端体验**: 如何在构建 MCP 客户端时进行工具选择是一个常见问题。目前尚不清楚每个客户端是否需要实现自己的工具 RAG，或者是否存在一个等待标准化的层。
- **安全性和身份验证**: 标准化的安全机制是 MCP 规范中最迫切的需求之一。未来预计会定义一个身份验证层，例如 OAuth 类似的流程或 API 密钥标准，以便客户端可以安全地连接到远程服务器。权限模型也可能被引入。
- **远程 MCP 支持**: 目前大多数 MCP 服务器都是本地优先的，限制了其可扩展性。随着生态系统的发展，将远程 MCP 作为一等公民，并采用可流式的 HTTP 传输，预计将促进 MCP 服务器的更广泛采用。

- **服务器发现与管理**: 随着 MCP 服务器数量的增加，简化服务器的发现、共享和贡献变得越来越重要。类似应用商店的双边平台（MCP Marketplace）可能会出现。
- **多租户支持**: 未来可能会出现 MCP 网关或编排层，作为统一的端点聚合多个 MCP 服务，处理路由甚至关于使用哪个工具的高级决策，并管理多租户和实施策略。
- **优化的 AI 代理**: 未来可能会出现专门为工具使用和 MCP 进行微调的 AI 模型。这些模型将更深入地理解协议，知道如何准确格式化请求，并可能在成功的基于 MCP 的操作日志上进行训练，从而提高效率和可靠性。

9. MCP 与现有解决方案的比较

- **与 API 的比较**: 相对于通常提供细粒度控制和高度特定功能的传统 API，MCP 提供了更广泛、更动态的能力，更适合需要灵活性和上下文感知能力的场景。对于需要精细控制、性能优化和最大可预测性的应用程序，仍然可能更倾向于使用细粒度的 API。
- **与 RAG 和 Agents 的关系** : AI 模型受益于RAG和Agents。MCP 旨在标准化如何将这些信息暴露给语言模型。Agents 在某种程度上可以看作是 RAG 的扩展。MCP 的出现使上下文的价值最大化。
- **与 OpenAI Function Call、GPTs 和 Agent SDK 的比较** : MCP 被认为是现有中间层的集大成者。它借鉴了 OpenAI Function Call 和 GPTs 的思路，但做得更轻量级和开放。相对于仍在发展中的 OpenAI Agent SDK，MCP 更加开源和灵活，但可能在精细度和效率上不如 Agent SDK。
- **与 LangChain 和 LlamaIndex 的比较**: LangChain 和 LlamaIndex 试图构建 Agent 框架，但由于其较高的抽象程度和复杂的框架，许多 LLM 开发者在初步使用后会转向自己开发。MCP 的出现对它们产生了较大的冲击。

10. 结论

模型上下文协议（MCP）代表着 AI 工具集成领域的一个重大进步。它通过提供一个标准化的开放协议，**极大地简化了 LLM 应用与外部世界的连接**，从而加速了 AI 应用的开发，提高了其可扩展性和可靠性。MCP 的成功不仅在于其技术设计，更在于其强大的支持者、借鉴成功的经验以及积极发展的生态系统。尽管仍然面临一些挑战，但 MCP 有望成为 **Agentic AI 的关键中间层**，并为开发者和创业公司带来新的机遇。正如 USB-C 统一了电子设备的连接一样，**MCP 有潜力成为 AI 原生时代的通用语言**，推动人与 AI 通过软件进行更流畅和安全的协作。

💡 参考资料

```
# 1. MCP官方文档
https://modelcontextprotocol.io/introduction
# 2. MCP Servers
https://github.com/modelcontextprotocol/servers.git
```

```
# 3. MCP python-sdk
https://github.com/modelcontextprotocol/python-sdk.git
# 4. awesome-mcp-servers
https://github.com/punkpeye/awesome-mcp-servers.git
# 5. awesome-mcp-clients
https://github.com/punkpeye/awesome-mcp-clients.git
# 6. Smithery 服务托管平台
https://smithery.ai/
#7. claude desktop 下载
https://claude.ai/download
# 8. cherry studio 下载
https://cherry-ai.com/
# 9. cline 插件下载
https://github.com/cline/cline
# 10. cloudflare
https://www.cloudflare.com/products/registrar/
# 11. OpenRoute
https://www.cloudflare.com/products/registrar/
```