模型上下文协议（MCP）深度解析：Agent互操作性的新纪元

2025年4月29日　10 分钟阅读

#AI #Agent #MCP #Protocol

本文介绍了模型上下文协议（MCP），并对其技术原理、主要贡献、当前优劣、生态系统现状，并与Google A2A等相关技术进行比较，展望其未来发展趋势。

I. 引言

人工智能（AI）特别是大型语言模型（LLM）的飞速发展，正深刻改变着人机交互和自动化任务处理的方式[1]。然而，LLM本身如同一个强大的"大脑"，其能力往往受限于训练数据，难以直接、实时地与外部世界丰富的工具、数据和服务进行交互[2]。为了打破这种信息孤岛，使AI Agent能够真正落地并执行复杂任务，迫切需要一种标准化的连接机制。在此背景下，由Anthropic公司于2024年底发起并开源的模型上下文协议（Model Context Protocol, MCP）应运而生[4]。MCP旨在定义一套开放、通用的规范，让LLM应用能够安全、高效地连接并利用外部数据源和工具，从而克服LLM原生知识的局限性，降低集成复杂性，释放AI Agent的全部潜力[2]。本报告将深入剖析MCP协议流行的原因、核心机制、当前优劣、生态系统现状，并与Google A2A等相关技术进行比较，展望其未来发展趋势。 MCP V1版本模型主要解决"如何将上下文引入模型"，未来重点是让模型能够"执行动作"，实现工作流自动化。

关键词： 模型上下文协议（MCP）、AI Agent、互操作性、集成复杂性、标准化、生态系统。

II. MCP的兴起：为何迅速流行?

MCP自推出以来，在短时间内获得了AI社区和业界的广泛关注与采纳 8。究其原因，主要在于它精准地解决了AI Agent发展的核心痛点，并带来了显著的价值。

A. 核心驱动力：标准化与互操作性

解决集成碎片化难题： 在MCP出现之前，将LLM（M个）与各种外部工具或数据源（N个）集成，需要为每对连接开发定制化的适配器，导致所谓的"M×N集成问题" 4。这种方式不仅开发成本高昂、效率低下，而且难以维护和扩展[1]。MCP通过提供一个统一的协议标准，将复杂的M×N问题简化为M+N问题：每个LLM应用（Client）和每个工具/数据源（Server）只需要实现一次MCP协议，即可实现彼此间的互操作 4。这种标准化极大地降低了集成门槛和复杂度，成为MCP迅速流行的首要原因。

"AI的USB-C"： Anthropic等机构形象地将MCP比作AI应用的"USB-C端口" 8。正如USB-C统一了设备的物理连接和数据传输标准，MCP旨在统一AI模型与外

目录

文章信息

字数

阅读时间

发布时间

更新时间

标签

#AI #Agent #MC

部世界的"数字连接"标准，使得任何兼容MCP的客户端都能与任何兼容MCP的服务器无缝对接，无论它们由谁构建 12。这种即插即用的理念对于构建开放、多元的AI生态至关重要。

提升互操作性： MCP定义了统一的通信规则和数据格式（基于JSON-RPC）6，确保了不同模型、不同工具之间交互的一致性。这不仅简化了开发，也增强了系统的健壮性，即使更换底层模型或工具，集成逻辑也能保持相对稳定[4]。

B. 关键推动因素

行业巨头背书与采纳： Anthropic作为发起者，其在AI领域的声誉和影响力为MCP的推广奠定了基础 5。随后，OpenAI、Google DeepMind、百度、微软、GitHub等行业巨头的相继采纳或表示支持，极大地提升了MCP的行业地位，加速其向事实标准乃至行业标准的演进 4。

开发者体验优化： MCP显著减少了开发者在集成工作上投入的时间和精力，使他们能更专注于应用逻辑和创新 4。通过避免碎片化、一次性的集成，降低了设置和持续维护的成本 4。此外，清晰的关注点分离（Agent逻辑 vs. 后端能力）使得代码库更模块化、更易维护 4。甚至可以使用AI来辅助生成MCP代码，进一步简化开发 4。

Agent能力增强： MCP不仅仅是数据传输通道，它支持双向通信，允许AI模型调用外部工具执行操作，并获取实时数据 3。这使得Agent能够基于最新信息做出决策，执行更复杂的任务，例如实时查询航班价格、发送邮件、操作数据库等 4。动态工具发现能力还允许Agent在运行时适应新任务和环境，无需代码修改或模型重训 4。

生态初步形成： 随着协议的推广，社区开始涌现出各种MCP Server实现，连接数据库、API、文件系统、开发工具等 8。同时，Cursor、Cline等AI开发助手以及CAMEL OWL等多Agent框架也集成了MCP Client功能 8。这种生态的初步形成为MCP的普及提供了基础。

III. MCP协议深度解析

理解MCP的核心机制对于评估其潜力和局限性至关重要。MCP协议规范定义了其架构、通信方式和核心交互原语 2。

MCP Spec

A. 协议架构：主机、客户端与服务器
MCP定义了一个清晰的三层架构 1：

主机 (Host): 指的是使用MCP发起连接的LLM应用程序，通常是用户与之交互的界面，例如AI IDE插件（如Cursor）、聊天应用（如Claude Desktop）或其他Agent环境 1。一个主机可以同时连接到多个MCP服务器 24。

客户端 (Client): 作为主机内部的连接器组件，负责管理与**单个特定**MCP服务器的连接 1。每个客户端与其连接的服务器维持一对一的状态化连接，处理主机与服务器之间的双向通信，路由请求、响应和通知 12。

服务器 (Server): 提供上下文信息和各种能力的外部服务 1。服务器可以是本地运行的程序（如访问本地文件系统），也可以是远程服务（如连接云数据库或API）11。服务器向客户端暴露其支持的功能原语。

B. 通信协议与传输

核心协议: MCP的核心通信基于轻量级的**JSON-RPC 2.0**消息格式 6。这是一种使用JSON作为数据格式的远程过程调用协议，定义了请求和响应的标准结构。

状态化连接: MCP连接是**状态化**的，允许在连接的生命周期内进行多次请求和响应，并支持能力协商，让客户端和服务器在通信开始时确定彼此支持的功能 6。

传输机制: 协议规范本身对具体的传输层机制保持一定的灵活性。常见的实现方式包括：

stdio: 用于本地服务器，客户端通过标准输入/输出与服务器进程直接通信 21。这种方式简单直接，但仅限于本地。

HTTP + SSE (Server-Sent Events): 用于本地或远程服务器。客户端通过HTTP连接到服务器的/sse端点，建立连接后，服务器可以通过SSE单向推送实时消息给客户端 14。这是目前较为主流的方式。新版MCP协议对基于HTTP的流式传输机制进行了优化 17。

其他: 协议也允许实现自定义的传输机制 12。

C. 核心原语：交互的基础
MCP定义了一组核心"原语"（Primitives）或称为"特性"（Features），作为客户端与服务器交互的基础构建块 11：

资源 (Resources): 代表服务器可以提供给客户端的结构化数据或上下文信息，供用户或AI模型使用 11。例如代码片段、文档内容、数据库查询结果等。通常由应用程序（Host/Client）控制何时请求资源。

工具 (Tools): 代表AI模型可以请求服务器执行的**动作或函数** 11。例如查询数据库、调用API、执行代码、发送邮件等。AI模型通常根据任务需求决定何时调用工具，一般需要用户确认 11。工具是MCP实现Agent能力的关键。

提示 (Prompts): 服务器可以提供的预制指令或模板，用于引导用户与LLM或相关工具进行交互 11。例如，为特定任务（如文本摘要、代码生成）预设的提示模板。通常由用户选择何时使用。

采样 (Sampling): 这是客户端（Host/Client）可能提供给服务器的一种能力，指由**服务器发起**的Agent行为和递归LLM交互 13。这允许服务器在需要时请求LLM进行推理或生成，以支持更复杂、更自主的工作流。用户需要明确批准此类请求 13。

根 (Roots): 在旧版规范中提及 13，但在当前分析的资料中未作为核心原语列出。

这些标准化的原语构成了MCP的"通用语言"，使得任何兼容MCP的客户端都能理解如何向任何兼容MCP的服务器请求数据（Resources）、使用预设指令（Prompts）或执行动作（Tools），从而实现了Agent与外部世界的结构化交互 11。

IV. MCP的优势与挑战

尽管MCP带来了显著的进步，但在实际应用中也面临一些挑战和局限性。

A. 主要优势 (Key Advantages)

标准化与互操作性: 这是MCP最核心的优势。通过统一协议，解决了M×N集成难题，降低了连接AI模型与外部工具/数据的复杂性 1。

开发效率提升: 显著减少了开发和维护定制化集成所需的时间和成本，让开发者聚焦于核心业务逻辑 3。

实时数据与动态能力: 支持实时、双向通信，使Agent能够访问最新数据并动态发现、使用新工具，增强了Agent的上下文感知能力和适应性 3。

可扩展性: 标准化接口使得添加新的工具或客户端变得更容易，促进了生态系统的发展和组件复用 3。

关注点分离: 清晰地划分了Agent逻辑与后端能力实现，有助于构建更模块化、更易于维护的代码库 4。

B. 当前面临的挑战与局限性 (Current Challenges and Limitations)

安全风险: 这是MCP目前最受关注的问题之一。

缺乏内置安全治理: MCP协议本身并未强制规定严格的安全措施，将安全责任主要留给了协议的实现者 11。这意味着不同实现的安全性可能参差不齐。

工具中毒 (Tool Poisoning): 存在恶意行为者通过构造恶意的工具描述或上下文数据，诱导AI执行非预期或有害操作的风险 11。如果客户端缺乏对输入的有效验证和净化，就可能发生此类攻击。

工具间数据泄露: 当Agent同时访问多个工具时，一个低权限或被攻破的工具可能诱骗Agent滥用高权限工具的能力，从而访问或泄露敏感数据 11。

任意代码执行风险: "工具"原语本质上代表了执行任意代码的能力，必须谨慎处理，确保用户充分理解并授权每一次调用 13。

标准化认证缺失: MCP协议规范目前没有定义标准的认证机制 12。这导致各个实现需要自行设计认证方案，增加了集成的复杂性和潜在的安全风险。缺乏统一认证也使得跨组织或跨系统的安全互信难以建立。

生态系统成熟度:

可靠服务器不足: 作为一个相对较新的协议，MCP生态系统仍在发展初期 12。虽然已有不少社区贡献的Server，但高质量、经过充分测试、覆盖广泛应用的官方或可靠第三方Server数量仍有待增加 27。

采用门槛: 对于非技术用户而言，自行部署和管理本地MCP Server仍然存在挑战 27。同时，说服已经投入现有技术栈（如特定API或框架）的开发者转向采用MCP也需要时间和动力 9。

协议复杂性与学习曲线: 虽然MCP旨在简化集成，但协议本身及其安全实现仍包含一定的复杂性，开发者需要投入时间学习和理解 18。特别是安全相关的实现，需要开发者具备相应的知识和经验。

性能与可伸缩性: 尽管协议设计考虑了可伸缩性 3，但在需要处理大规模并发请求或极低延迟的场景下，MCP基于JSON-RPC和可能涉及多层交互的模式是否会引入性能瓶颈，仍有待实际检验 29。

功能局限: MCP主要解决的是Agent与外部工具的**通信协议**问题，它本身并不创造新的应用范式 31。Agent最终能达到的效果，很大程度上仍取决于底层LLM的理解、推理和规划能力 31。此外，虽然MCP相比简单的Function Calling在处理多步任务和上下文方面有优势 10，但面对极其复杂的长期任务或需要精细控制的场景，可能仍有其局限性 10。

C. 安全考量与缓解措施 (Security Considerations and Mitigation)
鉴于上述安全风险，MCP规范本身以及社区实践都强调了实施安全措施的重要性。

核心原则: MCP规范明确提出四大安全原则：**用户同意与控制**（用户必须明确理解并授权所有数据访问和操作）、**数据隐私**（未经同意不得传输用户数据）、**工具安全**（谨慎对待代码执行，明确用户授权）、**LLM Sampling控制**（用户控制采样过程）13。

建议措施:

用户明确授权: 必须为所有敏感操作（特别是工具调用和数据访问）设计清晰、明确的用户授权流程和界面 13。

最小权限原则: 将每个工具或服务器视为独立的安全边界，仅授予其完成任务所必需的最小权限，以限制潜在风险的影响范围 11。

强认证机制: 强制使用可靠的认证机制，如基于PKI的TLS证书、数字签名 18，或针对远程服务器使用OAuth等标准授权框架 11。

输入验证与净化: 对来自服务器的工具描述、参数以及返回的数据进行严格的验证和净化，防范工具中毒等注入攻击 11。

安全开发实践: 遵循通用的安全最佳实践，如定期轮换密钥、安全日志记录、防火墙保护API端点、对高权限账户使用多因素认证等 18。提供清晰的安全文档，帮助用户理解风险 13。

利用安全网关/代理: 采用专门的MCP网关或代理服务，如阿里巴巴的Nacos+Higress 32、Solo.io的Agent Gateway 33 或MetaMCP 35 等。这些中间件可以在协议层面提供统一的认证、授权、访问控制、速率限制、审计日志、协议转换等安全和管理能力，弥补MCP协议自身在安全治理方面的不足。

一个重要的观察是，MCP协议本身在安全方面的"留白"以及企业级应用对安全性的高要求，共同催生了MCP网关、Hub等安全增强层的兴起。这些项目并非要取代MCP，而是在其基础上增加必要的安全、管理和治理能力，以解决MCP在企业环境中落地的关键瓶颈 10。这表明，安全性已成为推动MCP生态演进和满足实际需求的重要驱动力。

V. MCP生态系统全景扫描

MCP生态系统正围绕协议标准逐步构建，形成了包括服务器、客户端、Hub和网关等不同角色的参与者网络。

A. 生态系统架构：服务器、客户端、Hub与网关

MCP Servers: 作为生态的基础，提供具体的工具或数据访问能力。社区和企业已经开发了连接各种系统的MCP Server，种类日益丰富。例如：

数据库: 连接PostgreSQL (如通过Supabase PostgREST 19)、Neon Serverless Postgres 19、DataStax Astra DB 8 等。

RAG与知识库: RAGFlow提供检索工具 26，也有针对本地文档目录的RAG Server 36。

开发与代码: 连接GitHub Issues 19、Docker 19、Git 20、Stata 19、Tecton特征工程平台 37、VSCode开发工具 38 等。

API封装与Web服务: 发送邮件 (Resend 19, Mailtrap 19)、Web搜索 (Brave Search 19, Perplexity 20)、天气查询 (Caiyun Weather 19)、Web抓取与自动化 (Browserbase 19, Apify Actors 40)、日志分析 (Axiom 19)、项目管理 (Linear, Jira 18) 等。

协作与通信: Slack 19、Ntfy通知 20。

文件与本地系统: 本地文件系统访问 20、Obsidian笔记 20、手机控制 (PhonePi 19)。

企业内部系统: 连接内部知识库、数据库或特定业务API 11。

MCP Clients: 集成了MCP协议、能够消费MCP Server能力的AI应用或开发工具。主要类别包括：

AI IDE/编码助手: Cursor 8、Cline 22、Continue 22、Windsurf 40。这些工具利用MCP连接代码库、文档、数据库等，提供更智能的代码生成、编辑和问答

能力。

多Agent框架: CAMEL OWL 23、LangChain 5、LangGraph 22、Genkit 22、CrewAI 22、OpenAI Agents SDK 22。这些框架通过集成MCP，使其构建的Agent能够方便地调用外部工具。

桌面/Web应用: Claude Desktop 8、LibreChat 40、Apify Tester MCP Client 40、AgentStudio 22。

其他平台: 微软Copilot Studio 16、GitHub Copilot 15、Flowise 22、MindMac 22、OpenDevin 22、OpenWebUI 22、Portkey 22、Superagent 22。

MCP Hubs: 定位为聚合和管理层，旨在简化对大量MCP Server的访问和使用。

ACI.dev: 提供一个开源平台，预集成了超过600种工具 44。其核心是aci-mcp统一服务器，通过ACI_SEARCH_FUNCTIONS_WITH_INTENT和ACI_EXECUTE_FUNCTION两个元工具，让Agent可以动态发现和执行ACI平台上的任何可用工具，而无需将所有工具都加载到上下文中。同时，ACI平台还负责处理多租户认证和权限管理 44。

MCP Gateways: 在客户端和服务器之间扮演中间件角色，提供安全增强、协议转换、流量管理、可观测性等增值服务。

Alibaba Nacos + Higress: Nacos作为MCP注册中心（MCP Registry）存储和管理工具元数据，Higress作为云原生AI网关，负责将MCP的JSON-RPC请求转换为后端服务的标准HTTP请求，并将现有API（如OpenAPI定义的）零代码暴露为MCP Server 17。Higress还计划推出MCP市场 17。

Solo.io Agent Gateway: 这是一个专门为Agent通信（包括MCP和A2A）设计的开源数据平面 33。它解决了传统API网关难以处理MCP状态化、双向通信的问题，提供了统一的认证授权、多租户隔离、工具聚合（Federation）、可观测性、将现有REST API自动转换为MCP工具等企业级功能 34。

MetaMCP: 一个面向开发者的本地代理MCP服务器 35。它通过一个Web GUI应用（MetaMCP App 50）让用户方便地管理和配置多个实际运行的MCP Server。用户可以创建不同的工作空间（Workspaces），在不同空间内启用不同的Server组合，然后MetaMCP会将当前工作空间激活的Server聚合成一个统一的MCP接口暴露给客户端（如Claude Desktop, Cursor）。这解决了手动编辑配置文件切换Server组合的痛点，并增强了本地操作的隐私性 35。

B. 关键项目分析 (Analysis of Key Projects)

RAGFlow (Server): RAGFlow的MCP Server是其主服务的扩展组件，专注于提供基于RAG的文档检索能力。它通过retrieve工具，利用RAGFlow强大的文档理解和检索技术，为MCP客户端提供高质量的上下文信息。其HTTP+SSE通信模式和两种部署模式（Self-Host/Host）体现了MCP服务器实现的多样性 26。它代表了将特定领域（如RAG）的专业能力通过MCP接口标准化的趋势。

Cursor (Client): Cursor作为AI辅助编程的领军者，深度集成了MCP，使其Agent能够无缝调用数据库、API、文档等外部资源来辅助编码。它对stdio和SSE两种连接方式的支持，以及灵活的配置文件管理（项目级与全局级），展示了成熟MCP客户端应具备的特性。其Agent模式对MCP工具的自动发现、调用审批流程以及结果展示，为MCP在代码生成场景的应用树立了标杆 21。

Cline (Client): Cline的独特之处在于其宣称的"自扩展"能力，即通过自然语言指令或读取文档来创建新的MCP Server 41。如果这一能力得到证实和广泛应用，将极大降低MCP生态的构建门槛，使得Agent本身成为生态扩展的驱动力。它也提供了MCP服务器的管理界面，提升了用户体验 22。

CAMEL OWL (Client/Framework): 作为一个学术界背景的开源多Agent框架，OWL集成MCP（通过MCPToolkit）旨在利用MCP的标准化接口来增强其Agent调用外部工具（如浏览器自动化工具Playwright）的能力 23。这表明MCP不仅适用于商业产品，也为研究和开源社区提供了一种扩展Agent能力的标准化途径。

Manus (Client/Framework?): Manus作为一个备受关注的通用AI Agent产品，其与MCP的关系似乎较为模糊。官网信息并未直接提及MCP 53。相关分析文章或将其与MCP并列讨论 43，或认为两者是互补关系——MCP负责底层工具连接，Manus负责上层Agent部署与管理 43。Manus的案例可能代表了不直接依赖MCP协议，而是构建自身闭环Agent平台的另一种发展路径，或者未来可能通过某种方式接入MCP生态。

ACI.dev (Hub): ACI.dev的核心价值在于解决MCP生态中"工具爆炸"的问题。当可用MCP Server数量庞大时，如何有效发现、管理、授权并避免LLM上下文窗口被过多工具描述撑爆，成为新的挑战。ACI通过统一的MCP Server和动态发现机制（ACI_SEARCH_FUNCTIONS_WITH_INTENT, ACI_EXECUTE_FUNCTION）提供了一个解决方案，同时集成了认证和权限管理，定位于MCP工具的聚合与治理层 44。

Nacos+Higress (Gateway): 阿里巴巴的这套组合拳，精准地瞄准了企业将现有庞大API资产接入新兴MCP生态的需求。Nacos利用其成熟的服务注册与发现能力管理MCP元数据，Higress则发挥其API网关优势进行协议转换和流量管理。这种方式大大降低了企业采用MCP的门槛，为MCP在企业内部的推广铺平了道路 32。

Solo.io Agent Gateway (Gateway): Solo.io敏锐地指出传统API网关在处理MCP/A2A这类状态化、双向协议时的不足，并推出了专门的Agent Gateway。它不仅解决了协议适配问题，还集成了安全、可观测性、多租户、API自动转换等企业级特性，并同时支持MCP和A2A，旨在成为下一代Agent通信基础设施的核心组件 33。

MetaMCP (Gateway/Proxy): MetaMCP则从开发者个人体验出发，解决了在本地管理和切换多个MCP Server配置的繁琐问题。通过一个本地代理和一个Web GUI，实现了对底层Server的灵活组合与管理，并兼容所有MCP客户端。它代表了提升MCP易用性的工具层创新 35。

C. 生态参与者及其关系
MCP生态的快速发展吸引了不同类型的参与者：

协议发起者: Anthropic 4。

主要采纳者/支持者: OpenAI, Google, 百度, 微软, GitHub等大型科技公司，它们的加入极大地推动了MCP的标准化进程 4。

Client开发者: 包括AI IDE（Cursor, Cline）、多Agent框架（CAMEL, LangChain）、桌面应用（Claude Desktop）等，它们是MCP能力的最终消费者和应用场景的实现者 22。

Server开发者: 广大社区开发者和各类SaaS公司，它们为具体的工具、API或数据源开发MCP Server，是生态内容的主要贡献者 8。

Hub/Gateway提供商: ACI.dev, 阿里巴巴, Solo.io, MetaTool AI等，它们在Client和Server之间提供增值服务，解决管理、安全、聚合、转换等问题。

这些参与者之间形成了相互依存的关系：Client依赖Server提供功能；Server依赖Client带来用户和流量；Hub/Gateway通过解决生态痛点连接Client和Server；而协议发起者和主要采纳者则引领着标准的发展方向。观察这些参与者的角色和互动，可以发现MCP生态正呈现出明显的分层发展趋势。在底层的MCP协议规范和基础

的Server/Client实现之上，涌现出了更高层次的管理、安全和聚合层（Hubs/Gateways）。这并非偶然，而是生态走向成熟的必然结果。随着Server数量的激增和企业级应用的深入，直接管理大量异构Server并确保其安全可靠变得日益复杂 34。因此，提供统一访问入口、集中管理认证授权、实现协议转换、增强可观测性等能力的中间层应运而生，以满足更复杂的应用需求和企业级治理要求 32。这种从"点对点连接"到"分层管理与治理"的演变，标志着MCP生态正在从基础建设阶段向应用深化和价值提升阶段迈进。

VI. MCP与相关技术的比较分析

为了更清晰地理解MCP的定位和价值，有必要将其与现有及新兴的相关技术进行比较。

A. MCP vs. 传统API集成 MCP相较于传统的、为每个服务定制API集成的方式，具有显著的优势。下表总结了两者在关键特性上的差异：

特性 (Feature)	MCP (模型上下文协议)	传统API集成 (Traditional APIs)
集成方式 (Integration Method)	单一、标准化协议，实现M+N集成 4	为每个服务定制集成，导致M*N复杂性 1
通信风格 (Communication Style)	支持实时、双向通信 4	通常为单向请求-响应模式 4
工具发现 (Tool Discovery)	支持动态、自动发现 4	通常需要手动配置或硬编码 4
上下文感知 (Context Awareness)	内置上下文处理机制 18	有限或无内置上下文支持 18
可伸缩性 (Scalability)	标准化简化扩展，即插即用 4	集成复杂度随服务数量线性或指数增长 4
开发维护 (Development Effort)	降低开发时间与维护成本 4	开发和维护成本较高 4

可以看出，MCP通过标准化大幅简化了AI Agent与外部世界的连接，降低了开发门槛，提高了系统的灵活性和可维护性，这是其相较于传统API集成的核心竞争力。

B. MCP vs. Function Calling & LangChain

与Function Calling的关系: Function Calling是LLM本身具备的一种能力，允许模型根据上下文理解并决定调用预定义的外部函数 10。它通常由LLM提供商（如OpenAI, Google）在其模型API中实现，具体实现方式各异。虽然Function Calling解决了模型执行外部动作的问题，但它更侧重于模型自身的能力扩展，且在处理多轮对话、复杂依赖或大量工具时可能面临代码维护困难等问题 10。MCP则是一个**独立于特定模型**的**通信协议标准**，旨在规范Agent与任意外部工具/资源之间的交互方式 10。Function Calling可以被视为**实现**MCP中"Tool"调用的一种机制，即Agent通过其Function Calling能力来触发对某个MCP Server提供的Tool的调用。但MCP的范畴更广，它定义了更丰富的交互原语（如Resources, Prompts, Sampling）和标准的通信流程，旨在取代零散的、依赖特定模型实现的Agent代码集成 10。

与LangChain等框架的关系: LangChain、LlamaIndex等是流行的Agent开发框架，它们提供了一系列用于构建Agent应用的组件和抽象，包括对工具使用的封装 5。LangChain定义了其内部的Tool类接口，开发者需要按照这个接口来集成工具到Agent代码中 5。MCP与这类框架的关系在于，MCP提供了一个**模型/Agent运行时**的标准接口。这意味着，一个遵循MCP标准的工具（MCP

Server）可以被任何支持MCP的Agent（无论使用何种框架构建）在运行时发现和使用，而无需开发者在代码层面进行特定于框架的集成 5。因此，MCP可以看作是LangChain等框架中工具层的一种**标准化实现**。许多框架（如LangChain, Genkit, CrewAI）已经开始支持将MCP Server作为其工具来源 22。MCP的开放标准特性也旨在避免某些框架可能存在的代码抽象混乱或过度商业化的问题，促进更健康的生态发展 10。

因此，MCP并非要取代Function Calling或LangChain这类技术。相反，它提供了一个更底层、更通用的**互操作层**。Function Calling是模型执行动作的内在机制，LangChain是构建Agent应用的开发框架，而MCP则是连接Agent与外部世界的标准化通信协议。三者可以协同工作：Agent框架（如LangChain）可以使用模型的Function Calling能力来调用遵循MCP标准的外部工具/资源，从而构建出功能强大且具备良好互操作性的AI Agent系统。

C. MCP vs. Google A2A协议
在MCP崭露头角的同时，Google也推出了Agent2Agent (A2A)协议，引发了关于两者关系的广泛讨论。

A2A协议概述: A2A是由Google发起并联合超过50家技术伙伴（如Salesforce, MongoDB, Langchain等）共同推动的开放协议 54。其核心目标是解决**不同AI Agent之间**的通信和互操作性问题，使由不同供应商、不同框架构建的Agent能够安全地交换信息、协调行动，共同完成复杂任务 42。A2A建立在HTTP, SSE, JSON-RPC等现有Web标准之上 25，定义了一套包括Agent Card（用于能力发现）、Task（工作单元与生命周期管理）、Message/Part（支持文本、文件、结构化数据的通信内容）等核心概念的交互框架 25。它强调Agent之间的协作能力，支持多模态交互（文本、音视频），并内置了安全考量 55。

MCP与A2A的比较:

方面 (Aspect)	MCP (模型上下文协议)	Google A2A (智能体间协议)
主要焦点 (Primary Focus)	**单个Agent** 与 **工具/数据源** 的连接 24	**不同Agent之间** 的通信与协作 24
交互对象 (Interaction Between)	Agent <-> Tool/Resource Server	Agent (Client) <-> Agent (Remote)
核心抽象 (Core Abstraction)	外部系统是可调用的 **工具/资源** 10	其他Agent是可协作的 **对等实体** 60
关键概念 (Key Concepts)	Host, Client, Server, Resources, Tools, Prompts, Sampling	Agent Card, Task, Message, Part, Artifact, Client/Remote Roles
典型用例 (Typical Use Case)	Agent获取信息、执行具体操作（查数据库、发邮件）4	多Agent协作完成复杂流程（招聘、客服、供应链）55
定位 (Positioning)	连接Agent与外部能力的"接口" 61	Agent之间协作的"语言" 42

* **互补性与潜在竞争关系:** Google官方以及多数分析都将A2A定位为**MCP的补充** [24, 42, 55, 61, 62]。MCP负责武装单个Agent，为其提供访问工具和数据的能力；而A2A则让这些具备了能力的Agent能够相互沟通、协同工作，共同完成更宏大的任务 [24, 54, 58]。一个常见的比喻是：MCP是让工人（Agent）能够使用各种工具（Tools/Resources），而A2A是让工人们能够相互交谈、分配任务、合作完成一个大项目 [58]。甚至可以将一个A2A Agent本身封装为一个MCP Resource供其他Agent发现和交互 [58]。这种"Agent-工具 (MCP)" + "Agent-Agent (A2A)"的模式构成了未来AI系统架构的两个重要层面 [54, 61, 62]。然而，尽管定位互补，两者

在实践中可能存在一定的边界模糊和潜在竞争。例如，一个复杂的外部服务既可以通过MCP Tool直接调用，也可以被封装成一个专门的A2A Agent提供服务。技术选型可能取决于具体场景的复杂度和开发者的偏好。同时，像Solo.io Agent Gateway这样同时支持MCP和A2A的中间件的出现 [33, 34]，也暗示了市场对统一管理这两种协议的需求，这可能在未来促进两者的融合或在某些场景下形成竞争。

MCP和A2A的并存与互补，清晰地揭示了AI Agent系统发展的未来方向：从单一Agent的能力增强，走向由多个具备专业技能、能够相互协作的Agent组成的复杂、分层的智能系统。这类似于软件工程领域从单体应用向微服务架构的演进，预示着AI应用将具备前所未有的协同能力和智能化水平。

VII. MCP的未来发展趋势与展望

作为一项新兴且备受关注的技术标准，MCP的未来发展充满机遇，也面临挑战。

A. 技术演进方向

安全增强: 鉴于当前标准化认证的缺失是主要痛点 12，未来MCP规范极有可能引入标准的认证和授权机制，甚至可能集成更细粒度的权限管理模型，以满足企业级安全需求。

协议优化: 随着应用的深入，可能会对协议本身进行优化，例如进一步提升HTTP+SSE等传输机制的效率和稳定性 17，改进状态管理、错误处理和能力协商等方面。

更丰富的原语: 未来可能根据新的Agent交互模式或能力需求，增加新的协议原语，以支持更复杂的应用场景。

与A2A的协调: 随着MCP和A2A生态的共同发展，两者之间可能会出现更明确的协议层面的协调机制，或者出现更多同时支持两种协议的融合框架和网关，使得开发者能够在一个统一的架构下构建包含两种交互模式的复杂Agent系统 33。

B. 标准完善与生态成熟

标准地位巩固: 随着OpenAI、Google等更多行业领导者的采纳和社区的积极响应，MCP有望从当前的事实标准逐步演变为公认的行业标准 5。

生态工具链完善: 未来的发展将依赖于生态系统的成熟。这包括涌现更多高质量、经过验证、覆盖更广泛应用的MCP Server 12；提供更易用的开发、调试、部署和管理工具，降低使用门槛 27；以及建立成熟的MCP Server发现和分发机制，如MCP市场 17。

治理与维护: 作为一项开放标准，建立清晰、可持续的治理结构和版本迭代机制，对于MCP的长期健康发展至关重要。

C. 新兴应用场景
MCP的标准化接口为AI Agent开辟了广阔的应用前景：

复杂工作流自动化: 实现跨多个系统和服务的端到端任务自动化，如复杂的项目管理、事件策划、客户服务流程等 5。

具身智能与物理世界交互: 作为连接LLM"大脑"与机器人或物联网设备传感器、执行器的桥梁，使Agent能够感知和操作物理环境 5。

协作Agent网络 (Agent Societies): 为多个专用Agent（如研究Agent、规划Agent、执行Agent）提供共享工具和信息交换的标准接口，实现更强大的集体智能 5。

深度个性化AI助手: 通过本地MCP Server安全地连接个人数据（邮件、日历、笔记）和本地应用，创建真正理解用户、保护隐私的个人AI助手 5。

企业级应用与治理: 标准化AI对企业内部工具和数据的访问，不仅提高效率，也便于实现统一的日志记录、安全审计、权限控制和合规管理 3。

垂直行业深度集成: 在医疗、金融、法律、教育、内容创作等领域，通过MCP连接行业特定的数据和工具，构建专业的AI解决方案 3。

D. 面临的机遇与挑战

机遇: MCP有望成为下一代AI应用（特别是Agentic AI）不可或缺的基础设施层，赋能更强大、更自主、更具互操作性的AI系统 4。它将催生一个繁荣的工具（Server）和服务（Hub/Gateway）生态系统，带来巨大的商业价值 9。

挑战:

安全标准的落地: 如何有效建立并推广统一的安全认证和授权标准，并确保其在实践中得到严格执行，是赢得企业信任的关键。

生态培育: 如何激励开发者持续贡献高质量的MCP Server，如何建立有效的发现和信任机制，如何降低用户的使用门槛，是生态能否繁荣的核心 12。

标准协调与演进: 如何与A2A等相关标准协调发展，避免碎片化；如何在快速发展的AI技术浪潮中保持标准的先进性和适应性，避免过早僵化或落后 28。

用户接受度与信任: 最终用户和企业是否愿意将敏感数据和关键操作的权限授予通过MCP连接的AI系统，需要建立充分的信任和透明度 28。

归根结底，MCP的长期成功不仅取决于协议本身的技术优越性，更深层次地依赖于其生态系统的健康发展和有效的治理能力。一个充满活力、值得信赖、易于使用的生态系统，以及一个能够适应变化、保障安全的治理机制，将是MCP从一个有前景的协议演变为无处不在的基础设施的关键所在。

VIII. 结论与建议

模型上下文协议（MCP）作为一项旨在标准化AI Agent与外部工具及数据源交互的开放协议，自问世以来已展现出巨大的潜力，并迅速成为AI领域的热点。

核心价值与当前地位: MCP的核心价值在于通过标准化解决了长期困扰AI Agent发展的集成碎片化难题，显著降低了开发复杂性，提升了系统的互操作性和可扩展性。它使得Agent能够更便捷地获取实时信息、调用外部功能，从而执行更复杂的任务。目前，MCP已获得主要AI公司的支持，生态系统初具规模，涵盖了多样的客户端、服务器以及新兴的Hub和网关，正朝着成为行业基础性协议的方向稳步发展。

关键挑战: 然而，MCP的发展仍面临严峻挑战。**安全性**是其在企业级应用中广泛落地的最大瓶颈，协议本身缺乏内置的安全治理和标准化认证机制，带来了工具中毒、数据泄露等风险。此外，**生态系统的成熟度**仍有待提高，高质量、可靠的服务器数量需要进一步增长，开发和使用的门槛也需降低。

与相关技术的关系: MCP与Function Calling、LangChain等现有技术是互补关系，它提供了一个更底层的标准化互操作层。与Google A2A协议相比，MCP聚焦于Agent与工具的连接，而A2A聚焦于Agent之间的协作。两者共同勾勒了未来复杂、分层、协作的AI Agent系统的蓝图。

建议:

对开发者:

积极关注与评估: 密切关注MCP协议的演进和生态发展，评估在项目中引入MCP Client或开发MCP Server的可行性与收益。

优先采用成熟方案: 在涉及安全、管理等复杂问题时，优先考虑使用成熟的MCP Hub或Gateway解决方案（如ACI.dev, Nacos+Higress, Solo.io Agent Gateway, MetaMCP等），以降低风险和开发成本。

参与生态贡献: 通过开发新的MCP Server、完善文档、参与社区讨论等方式，为MCP生态的繁荣贡献力量。

对企业:

评估应用潜力: 审视自身业务流程，评估引入基于MCP/A2A的AI Agent实现自动化的潜力和价值。

关注安全与治理: 在引入MCP相关技术时，将安全和治理作为首要考量，选择提供相应解决方案的平台或工具。

谨慎技术选型: 考虑生态成熟度、供应商支持、社区活跃度以及与现有技术栈的兼容性。

对标准制定者与社区:

加速安全标准制定: 尽快推出并推广标准化的认证、授权机制，解决当前最大的安全顾虑。

完善工具与文档: 提供更易用的开发、测试、调试工具和更全面的最佳实践文档，降低开发者门槛。

建立治理机制: 建立清晰、开放、可持续的标准治理和版本迭代流程，确保标准的长期活力和适应性。

促进生态合作: 鼓励不同参与者之间的合作，共同建设开放、可信、繁荣的MCP生态系统。

总之，MCP协议为构建更强大、更智能、更具互操作性的AI Agent系统奠定了重要基础。虽然前路仍有挑战，但随着标准的不断完善、生态的日益成熟以及相关技术的协同发展，MCP有望在未来AI应用格局中扮演关键角色，开启Agent互操作性的新纪元。

IX. 参考文献

引用的著作

[1]. Understanding the Model Context Protocol | Frontegg, 访问时间为 四月 29, 2025， https://frontegg.com/blog/model-context-protocol

[2]. Model Context Protocol · GitHub, 访问时间为 四月 29, 2025，https://github.com/modelcontextprotocol

[3]. What is MCP in AI and Its Benefits - Aalpha Information Systems India Pvt. Ltd., 访问时间为 四月 29, 2025， https://www.aalpha.net/blog/what-is-mcp-in-ai/

[4]. What is Model Context Protocol? The emerging standard bridging AI ..., 访问时间为 四月 29, 2025， https://www.zdnet.com/article/what-is-model-context-protocol-the-emerging-standard-bridging-ai-and-data-explained/

[5]. #14: What Is MCP, and Why Is Everyone – Suddenly!– Talking About It? - Hugging Face, 访问时间为 四月 29, 2025，https://huggingface.co/blog/Kseniase/mcp

[6]. 模型上下文协议- 维基百科，自由的百科全书, 访问时间为 四月 29, 2025，

https://zh.m.wikipedia.org/zh-my/%E6%A8%A1%E5%9E%8B%E4%B8%8A%E4%B8%8B%E6%96%87%E5%8D%8F%E8%AE%AE
[7]. 模型上下文协议- 维基百科，自由的百科全书, 访问时间为 四月 29, 2025，https://zh.wikipedia.org/wiki/%E6%A8%A1%E5%9E%8B%E4%B8%8A%E4%B8%8B%E6%96%87%E5%8D%8F%
[8]. Will Model Context Protocol (MCP) Become the Standard for Agentic AI? - Datanami, 访问时间为 四月 29, 2025，https://www.bigdatawire.com/2025/03/31/will-model-context-protocol-mcp-become-the-standard-for-agentic-ai/
[9]. What is MCP? Claude Anthropic's Model Context Protocol - PromptLayer, 访问时间为 四月 29, 2025，https://blog.promptlayer.com/mcp/
[10]. The Ultimate Guide to MCP - Guangzheng Li, 访问时间为 四月 29, 2025，https://guangzhengli.com/blog/en/model-context-protocol
[11]. A Primer on the Model Context Protocol (MCP) - Apideck, 访问时间为 四月 29, 2025，https://www.apideck.com/blog/a-primer-on-the-model-context-protocol
[12]. What is Model Context Protocol (MCP): Explained - Composio, 访问时间为 四月 29, 2025，https://composio.dev/blog/what-is-model-context-protocol-mcp-explained/
[13]. Specification - Model Context Protocol, 访问时间为 四月 29, 2025，https://spec.modelcontextprotocol.io
[14]. 【上线】AI开放能力支持MCP，为智能体装上手和脚 - 百度AI, 访问时间为 四月 29, 2025，https://ai.baidu.com/support/news?action=detail&id=3238
[15]. 使用模型上下文协议(MCP) 扩展Copilot 对话助手- GitHub Enterprise Cloud Docs, 访问时间为 四月 29, 2025，https://docs.github.com/zh/enterprise-cloud@latest/copilot/customizing-copilot/extending-copilot-chat-with-mcp
[16]. 使用模型上下文协议扩展代理（预览版） - Microsoft Copilot Studio, 访问时间为 四月 29, 2025，https://learn.microsoft.com/zh-cn/microsoft-copilot-studio/agent-extend-action-mcp
[17]. Higress Open Source Remote MCP Server Hosting Solution and Upcoming MCP Market, 访问时间为 四月 29, 2025，https://www.alibabacloud.com/blog/higress-open-source-remote-mcp-server-hosting-solution-and-upcoming-mcp-market_602108
[18]. What is MCP (Model Context Protocol)? - Daily.dev, 访问时间为 四月 29, 2025，https://daily.dev/blog/what-is-mcp-model-context-protocol
[19]. MCP Servers for Cursor - Cursor Directory, 访问时间为 四月 29, 2025，https://cursor.directory/mcp
[20]. cyanheads/model-context-protocol-resources: Exploring the Model Context Protocol (MCP) through practical guides, clients, and servers I've built while learning about this new protocol. - GitHub, 访问时间为 四月 29, 2025，https://github.com/cyanheads/model-context-protocol-resources
[21]. Model Context Protocol - Cursor, 访问时间为 四月 29, 2025，https://docs.cursor.com/context/model-context-protocol
[22]. Example Clients - Model Context Protocol, 访问时间为 四月 29, 2025，https://modelcontextprotocol.io/clients
[23]. The New Era of Automation: How OWL, CRAB, and MCP Are ..., 访问时间为 四月 29, 2025，https://www.camel-ai.org/blogs/the-new-era-of-automation-how-owl-crab-and-mcp-are-bridging-the-last-mile
[24]. A Comparative Analysis of Anthropic's Model Context Protocol and ..., 访问时间为 四月 29, 2025，https://securityboulevard.com/2025/04/a-comparative-analysis-of-anthropics-model-context-protocol-and-googles-agent-to-agent-protocol/
[25]. How Google A2A Protocol Actually Works: From Basic Concepts to

Production - Trickle AI, 访问时间为 四月 29, 2025, https://www.trickle.so/blog/how-google-a2a-protocol-actually-works
[26]. RAGFlow MCP server overview | RAGFlow, 访问时间为 四月 29, 2025, https://ragflow.io/docs/dev/mcp_server
[27]. 模型上下文协议（MCP） | Technology Radar | Thoughtworks China, 访问时间为 四月 29, 2025, https://www.thoughtworks.com/cn/radar/platforms/model-context-protocol-mcp
[28]. Breaking Data Barriers: Can Anthropic's Model Context Protocol Enhance AI Performance?, 访问时间为 四月 29, 2025, https://www.unite.ai/breaking-data-barriers-can-anthropics-model-context-protocol-enhance-ai-performance/
[29]. Model Context Protocol (MCP) and Its Impact on AI-Driven Startups - Aalpha, 访问时间为 四月 29, 2025, https://www.aalpha.net/blog/model-context-protocol-mcp-and-its-impact-on-ai-driven-startups/
[30]. MCP for Multi-Language Support: Features & Benefits - BytePlus, 访问时间为 四月 29, 2025, https://www.byteplus.com/en/topic/541783
[31]. 跟同济子豪兄一起学MCP - ModelScope, 访问时间为 四月 29, 2025, https://www.modelscope.cn/learn/1121
[32]. Nacos Releases MCP Registry, Achieving a "Zero-Change ..., 访问时间为 四月 29, 2025, https://www.alibabacloud.com/blog/nacos-releases-mcp-registry-achieving-a-zero-change-upgrade-of-existing-application-interfaces-to-mcp-protocol_602156
[33]. Solo.io Launches Agent Gateway and Introduces Agent Mesh for Unified AI Connectivity, 访问时间为 四月 29, 2025, https://www.solo.io/press-releases/solo-io-launches-agent-gateway-and-introduces-agent-mesh
[34]. Solo.io Blog | A new Gateway for AI Agents | Solo.io, 访问时间为 四月 29, 2025, https://www.solo.io/blog/why-do-we-need-a-new-gateway-for-ai-agents
[35]. GitHub - metatool-ai/mcp-server-metamcp, 访问时间为 四月 29, 2025, https://github.com/metatool-ai/mcp-server-metamcp
[36]. MCP Docs RAG Server - UBOS.tech, 访问时间为 四月 29, 2025, https://ubos.tech/mcp/mcp-docs-rag-server/
[37]. Introducing AI-Assisted Feature Engineering with Cursor & MCP | Tecton, 访问时间为 四月 29, 2025, https://www.tecton.ai/blog/introducing-ai-assisted-feature-engineering-with-cursor-mcp/
[38]. C# Lang MCP Server - UBOS.tech, 访问时间为 四月 29, 2025, https://ubos.tech/mcp/c-lang-mcp-server/
[39]. How to Use MCP Servers with Cline - Apidog, 访问时间为 四月 29, 2025, https://apidog.com/blog/cline-mcp-servers/
[40]. What is Anthropic's Model Context Protocol (and why does it matter)? - Apify Blog, 访问时间为 四月 29, 2025, https://blog.apify.com/what-is-model-context-protocol/
[41]. MCP Servers Explained: What They Are, How They Work, and Why ..., 访问时间为 四月 29, 2025, https://cline.bot/blog/mcp-servers-explained-what-they-are-how-they-work-and-why-cline-is-revolutionizing-ai-tools
[42]. google/A2A: An open protocol enabling communication and interoperability between opaque agentic applications. - GitHub, 访问时间为 四月 29, 2025, https://github.com/google/A2A
[43]. AI Agents growing strong: MCP, Manus, and OpenAI Agents API ..., 访问时间为 四月 29, 2025, https://itsg-global.com/ai-agents-growing-strong-mcp-manus-and-openai-agents-api/
[44]. aipotheosis-labs/aci: ACI.dev is the open source platform that connects your AI agents to 600+ tool integrations with multi-tenant auth, granular permissions, and access through direct function calling or a unified MCP server. - GitHub, 访问

时间为 四月 29, 2025， https://github.com/aipotheosis-labs/aci
[45]. aipotheosis-labs/aci-mcp: MCP server(s) for Aipolabs ACI.dev - GitHub, 访问时间为 四月 29, 2025， https://github.com/aipotheosis-labs/aci-mcp
[46]. aipotheosis-labs/aipolabs-mcp: MCP server(s) for Aipolabs ACI.dev - GitHub, 访问时间为 四月 29, 2025， https://github.com/aipotheosis-labs/aipolabs-mcp
[47]. What Is Higress?-HigressOfficial Website, 访问时间为 四月 29, 2025， https://higress.cn/en/docs/latest/overview/what-is-higress/
[48]. alibaba/higress - AI Native API Gateway - GitHub, 访问时间为 四月 29, 2025， https://github.com/alibaba/higress
[49]. Solo.io Launches Agent Gateway and Introduces Agent Mesh for Unified AI Connectivity, 访问时间为 四月 29, 2025， https://www.globenewswire.com/news-release/2025/04/24/3067475/0/en/Solo-io-Launches-Agent-Gateway-and-Introduces-Agent-Mesh-for-Unified-AI-Connectivity.html
[50]. metatool-ai/metatool-app - GitHub, 访问时间为 四月 29, 2025， https://github.com/metatool-ai/metatool-app
[51]. I built MetaMCP: a middleware MCP to manage all your MCPs (open source with GUI, multi-client, multi-workspace, including Claude) : r/ClaudeAI - Reddit, 访问时间为 四月 29, 2025， https://www.reddit.com/r/ClaudeAI/comments/1ix1map/i_built_metamcp_a_middleware_mcp_to_manage_all/
[52]. I built MetaMCP: a middleware MCP to manage all your MCPs (open source with GUI, multi-client, multi-workspace, including Claude) - Reddit, 访问时间为 四月 29, 2025， https://www.reddit.com/r/mcp/comments/1ix261z/i_built_metamcp_a_middleware_mcp_to_manage_all/
[53]. Manus, 访问时间为 四月 29, 2025， https://manus.im/
[54]. Google's Agent-to-Agent (A2A) and Anthropic's Model Context Protocol (MCP) - Gravitee.io, 访问时间为 四月 29, 2025， https://www.gravitee.io/blog/googles-agent-to-agent-a2a-and-anthropics-model-context-protocol-mcp
[55]. Announcing the Agent2Agent Protocol (A2A) - Google for Developers Blog, 访问时间为 四月 29, 2025， https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/
[56]. Google's Agent2Agent (A2A) protocol: A new standard for AI agent collaboration - Wandb, 访问时间为 四月 29, 2025， https://wandb.ai/onlineinference/mcp/reports/Google-s-Agent2Agent-A2A-protocol-A-new-standard-for-AI-agent-collaboration--VmlldzoxMjIxMTk1OQ
[57]. Agent2Agent (A2A) 协议发布 - Google for Developers Blog, 访问时间为 四月 29, 2025， https://developers.googleblog.com/zh-hans/a2a-a-new-era-of-agent-interoperability/
[58]. Google A2A协议：MCP的替代还是配合？ - 安全内参, 访问时间为 四月 29, 2025， https://www.secrss.com/articles/77523
[59]. 讓AI Agent 彼此溝通：Google A2A 和Anthropic MCP 深度解析 - iKala, 访问时间为 四月 29, 2025， https://ikala.ai/zh-tw/blog/ikala-ai-insight/an-in-depth-analysis-of-googles-a2a-protocol-and-its-relationship-with-anthropics-mcp-ch/
[60]. Google Announces A2A - Agent to Agent protocol : r/AI_Agents - Reddit, 访问时间为 四月 29, 2025， https://www.reddit.com/r/AI_Agents/comments/1jvbfe8/google_announces_a2a_agent_to_agent_protocol/
[61]. A2A and MCP: Start of the AI Agent Protocol Wars? - Koyeb, 访问时间为 四月 29, 2025， https://www.koyeb.com/blog/a2a-and-mcp-start-of-the-ai-agent-protocol-wars
[62]. Home - Google, 访问时间为 四月 29, 2025， https://google.github.io/A2A/
[63]. blog.promptlayer.com, 访问时间为 四月 29, 2025， https://blog.promptlayer.com/mcp/#:~:text=and%20server%20registration.-,Final%20thoughts,levels%20of%2

[64]. 拼合智能体技术版图：MCP 协议、身份验证和授权以及Durable Objects 免费计划, 访问时间为 四月 29, 2025， https://blog.cloudflare.com/zh-cn/building-ai-agents-with-mcp-authn-authz-and-durable-objects/
[65]. MCP与ANP对比：智能体需要什么样的通信协议.md - GitHub, 访问时间为 四月 29, 2025 , https://github.com/agent-network-protocol/AgentNetworkProtocol/blob/main/blogs/cn/MCP%E4%B8%8EANP%E5%AF%B9%E6%AF%94%EF%B
[66]. Vertex AI Agent Builder | Google Cloud, 访问时间为 四月 29, 2025 , https://cloud.google.com/products/agent-builder
[67]. I Started awesome-a2a for Google's Agent2Agent Protocol - Hoping to Build It with Community Help! : r/AI_Agents - Reddit, 访问时间为 四月 29, 2025 , https://www.reddit.com/r/AI_Agents/comments/1jwq6eh/i_started_awesomea2a_for_googles_agent2agent/
[68]. AI Agent破局： MCP与A2A定义安全新边界, 访问时间为 四月 29, 2025 , https://www.secrss.com/articles/77593
[69]. Vertex AI Agent Builder | Google Cloud, 访问时间为 四月 29, 2025 , https://cloud.google.com/products/agent-builder?hl=zh-CN
[70]. Developers - RAGFlow, 访问时间为 四月 29, 2025 , https://ragflow.io/docs/dev/category/developers
[71]. Compare Model Context Protocol (MCP) vs. RAGFlow in 2025 - Slashdot, 访问时间为 四月 29, 2025 , https://slashdot.org/software/comparison/Model-Context-Protocol-MCP-vs-RAGFlow/
[72]. Features | Cursor - The AI Code Editor, 访问时间为 四月 29, 2025 , https://www.cursor.com/features
[73]. Cline - AI Autonomous Coding Agent for VS Code, 访问时间为 四月 29, 2025， https://cline.bot/
[74]. Starting from Manus and MCP: AI Agent's cross-border exploration of Web3 - PANews, 访问时间为 四月 29, 2025 , https://www.panewslab.com/en/articledetails/7h568o8qz5n8.html
[75]. Manus AI Explained: How Autonomous Agents Are Changing the Game - Design+Code, 访问时间为 四月 29, 2025 , https://designcode.io/agentic-workflows-manus-ai-explained/
[76]. Show HN: Owl and MCP Integration – Plug-and-play agents with external tools, 访问时间为 四月 29, 2025 , https://news.ycombinator.com/item?id=43488255
[77]. We integrated the MCP with OWL— fully autonomous multi-agent workflows using external tools : r/CamelAI - Reddit, 访问时间为 四月 29, 2025, https://www.reddit.com/r/CamelAI/comments/1jkolyx/we_integrated_the_mcp_with_owl_fully_autonomous/
[78]. Seamless Notion automation powered by MCP × OWL : r/CamelAI - Reddit, 访问时间为 四月 29, 2025 , https://www.reddit.com/r/CamelAI/comments/1k5yr4r/seamless_notion_automation_powered_by_mcp_owl/
[79]. AI updates from the past week: Docker MCP Catalog, Solo.io's Agent Gateway, and AWS SWE-PolyBench — April 25, 2025 - SD Times, 访问时间为 四月 29, 2025 , https://sdtimes.com/ai/ai-updates-from-the-past-week-docker-mcp-catalog-solo-ios-agent-gateway-and-aws-swe-polybench-april-25-2025/
[80]. Releases · metatool-ai/mcp-server-metamcp - GitHub, 访问时间为 四月 29, 2025， https://github.com/metatool-ai/mcp-server-metamcp/releases
[81]. How do you register MCP servers with this proxy? · Issue #14 - GitHub, 访问时间为 四月 29, 2025 , https://github.com/metatool-ai/mcp-server-metamcp/issues/14
[82]. metatool-ai - GitHub, 访问时间为 四月 29, 2025 , https://github.com/metatool-ai

分享这篇文章

相关文章推荐

模型上下文协议（MCP）深度解析：...

本文介绍了模型上下文协议（MCP），并对其技术原理、...

Deep Research 深度研究

Deep Research 深度研究

AI Agent Gateway

AI Agent Gateway