打造 MCP Server

释放 AI 无限潜力

张晋涛

Microsoft MVP

Kong Senior Software Engineer

AI 爆发期的典型应用

ChatGPT

"Chat" refers to the conversational interface—you talk to it like you would with a person.

"GPT" stands for Generative Pre-trained Transformer, which is the underlying AI model. It’s trained on a huge dataset of text from the internet, books, articles, and more.

限制

- 有知识截至时间
- 无法调用工具
  - 联网
  - 执行外部操作
  - ...

用户预期

自动化完成需求

- 一句话完成指令/ Zero-shot prompt

解決之道

- Function calling - 2023.06
- Model Context Protocol (MCP) - 2024.11

什么是 MCP?

Model Context Protocol (MCP) 是一个开放标准，用于促进 AI 模型与外部工具和数据源之间的通信。

核心理念

- 🔌 像 USB-C 一样的**通用接口**
- 🤖 让 AI 应用程序连接到多样化的外部服务
- 🌐 超越训练数据，执行现实世界任务

MCP 的解决方案

- ✅ **标准化**通信协议
- ✅ **一次开发，处处可用**
- ✅ 降低集成复杂度
- ✅ 促进生态系统发展

协议技术规范

基于 JSON-RPC 2.0

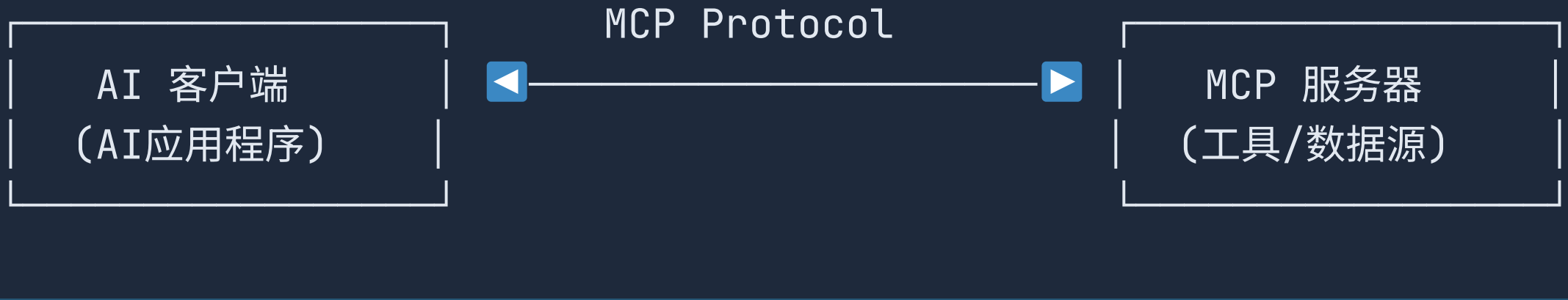
- 🔄 **有状态会话**: 维持客户端-服务器会话
- 📨 **消息类型**:
  - **请求 (Request)**: 双向通信消息
  - **响应 (Response)**: 请求的回复
  - **通知 (Notification)**: 无需响应的消息

协议技术规范

传输方式

- 🖥️ **stdio**: 本地进程间通信
- 🌐 **HTTP SSE**: 网络通信 (Server-Sent Events)
- 🔌 **传输无关**: 支持任意双向通信通道

MCP 架构



客户端-服务器模式

- **客户端**: AI 应用程序，发送请求
- **服务器**: 提供工具或数据源访问
- **协议**: 标准化的通信接口

主要功能特性

🔄 标准化协议

- 统一的通信格式
- 消除自定义集成需求
- 跨平台兼容性

🚀 可扩展性

- 开发者可创建自定义 MCP 服务器

数据类型与架构设计

三大核心原语 (Primitives)

- 🛠️ **工具 (Tools)**: AI 可执行的函数
- 📚 **资源 (Resources)**: 为 AI 提供上下文的数据源
- 📝 **提示 (Prompts)**: 可重用的语言模型交互模板

数据处理特点

- ✅ **强类型验证**: TypeScript 定义的严格模式

错误处理与调试

错误类型

- 🌐 **协议错误**: 通信层错误（工具未找到、参数无效等）
- ⚙️ **工具执行错误**: 工具内部错误（API 失败、业务逻辑错误等）

调试工具

- 🔍 **MCP Inspector**: 交互式服务器测试和调试工具
- 💻 **Claude Desktop 开发者工具**: 集成测试和日志收集

SDK 与开发支持

官方 SDK 支持

- 🐍 **Python SDK**: 完整的客户端和服务器开发支持
- 📜 **TypeScript SDK**: 原生支持，类型安全
- ☕ **Java SDK**: 企业级应用开发

开发工具生态

- 📖 **参考实现**: GitHub 上的完整示例代码

MCP 生态系统

支持的服务类型

- 📊 **数据库**: SQL, NoSQL, 时序数据库
- 🌐 **API 服务**: REST, GraphQL, 第三方服务
- 📁 **文件系统**: 本地文件, 云存储
- 🛠️ **开发工具**: Git, CI/CD, 监控系统
- 📧 **通信工具**: 邮件, 聊天, 通知系统

具体服务器示例

文件系统 MCP 服务器

```
// 文件读写操作
server.setRequestHandler(CallToolRequestSchema, async (request) ⇒ {
  if (request.params.name ═══ "read_file") {
    const content = await fs.readFile(request.params.arguments.path);
    return { toolResult: content };
  }
});
```

协议对比分析

MCP vs LSP (Language Server Protocol)

特性	MCP	LSP
目标	AI 与工具集成	IDE 与语言服务
范围	通用工具访问	语言特定功能
协议	JSON-RPC 2.0	JSON-RPC 2.0
状态	有状态会话	有状态会话

MCP vs 传统 Webhook/API

性能特性与限制

性能优势

- 🚀 **高效通信**: JSON-RPC 2.0 协议开销低
- 🔄 **连接复用**: 长连接和连接池管理
- ⚡ **异步支持**: 非阻塞 I/O 操作
- 📊 **流式处理**: 对大数据集的支持

存在的限制

实际应用场景

1. 智能数据分析

AI 助手 → MCP → 数据库 → 实时业务报告

2. 自动化运维

AI 系统 → MCP → 监控工具 → 故障诊断和修复

开发者体验

传统方式 😞

```
# 为每个服务写自定义代码
database_client = DatabaseClient(config)
api_client = APIClient(credentials)
file_client = FileClient(path)
# ... 更多自定义集成
```

使用 MCP 😊

技术优势

 架构优势

- **解耦**: AI 逻辑与工具实现分离
- **模块化**: 可独立开发和部署
- **可测试**: 每个组件可单独测试

 安全性

- 标准化的认证机制

行业采用与集成案例

现有集成实例

- 🤖 **Claude Desktop**: Anthropic 的桌面应用用作主要集成平台
- 💼 **GitHub MCP Server**: 代码仓库管理和协作
- 📁 **Google Drive Server**: 云存储文件访问
- 💬 **Slack Integration**: 团队协作和通信

企业级应用

当前状态 (2024-2025)

🏛️ 由 Anthropic 主导开发

- 已开源并成为行业标准
- 活跃的开发社区
- 持续的功能扩展

📊 生态系统增长

- 越来越多的工具支持

开始使用 MCP

1. 安装 MCP 客户端

```
npm install @modelcontextprotocol/sdk
```

2. 创建简单的服务器

```
import { Server } from '@modelcontextprotocol/sdk/server/index.js';
```

未来展望

🌟 发展方向

- 更多工具和服务的原生支持
- 增强的安全和权限管理
- 跨语言 SDK 完善
- 云原生集成优化

🚀 潜在影响

谢谢观看！

Model Context Protocol

让 AI 真正连接世界

了解更多: https://github.com/anthropics/mcp

Q&A

问题与讨论

📧 Contact: zhangjintao@apache.org

🐙 GitHub: https://github.com/tao12345666333