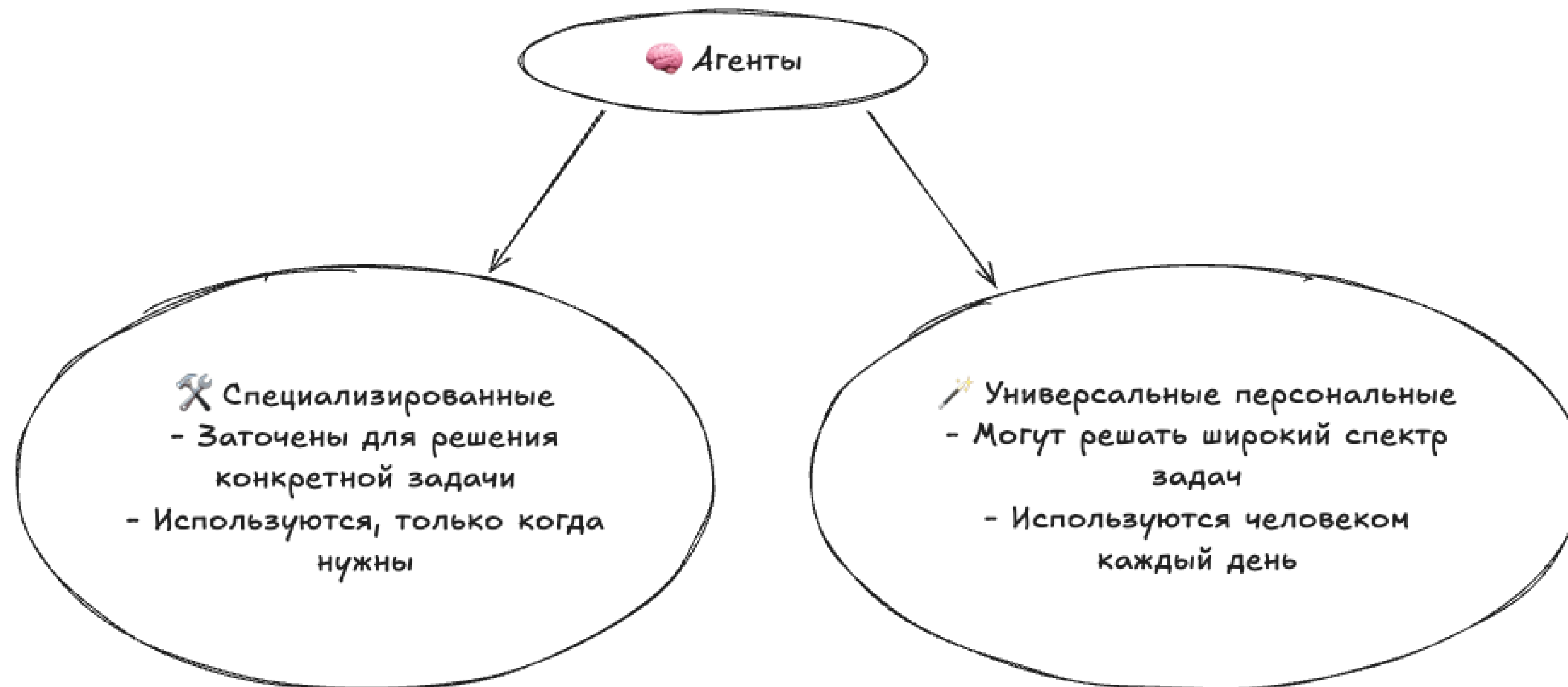




# МСР Think Tool добавляем мышление AI-агенту

# Развилка

Куда всё идёт



Примеры:

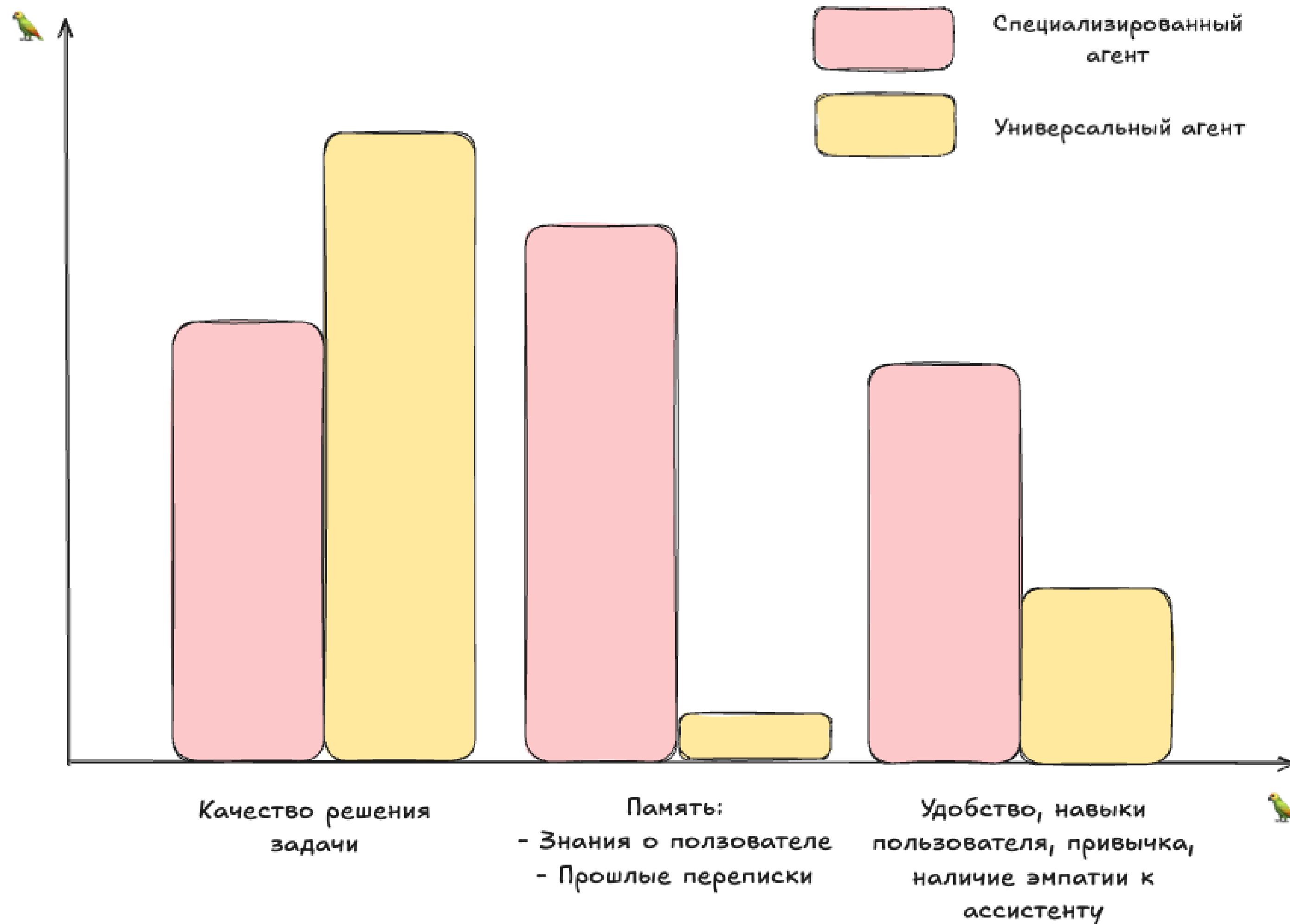
1. Support bot с RAG
2. Deep research mode
3. Поиск по научным статьям (elicit.com)
4. Gamma

Примеры:

1. ChatGPT
2. Anthropic Desktop
3. Cursor / Github copilot



# Персональный агент



## Прогноз

1

Пользователи будут предпочитать универсальных агентов вместо специализированных

Память и эмпатия важнее процентов качества

2

У каждого человека будет два агента на каждый день

- Личный агент
- Рабочий агент

3

Универсальный агент = новый интерфейс к миру

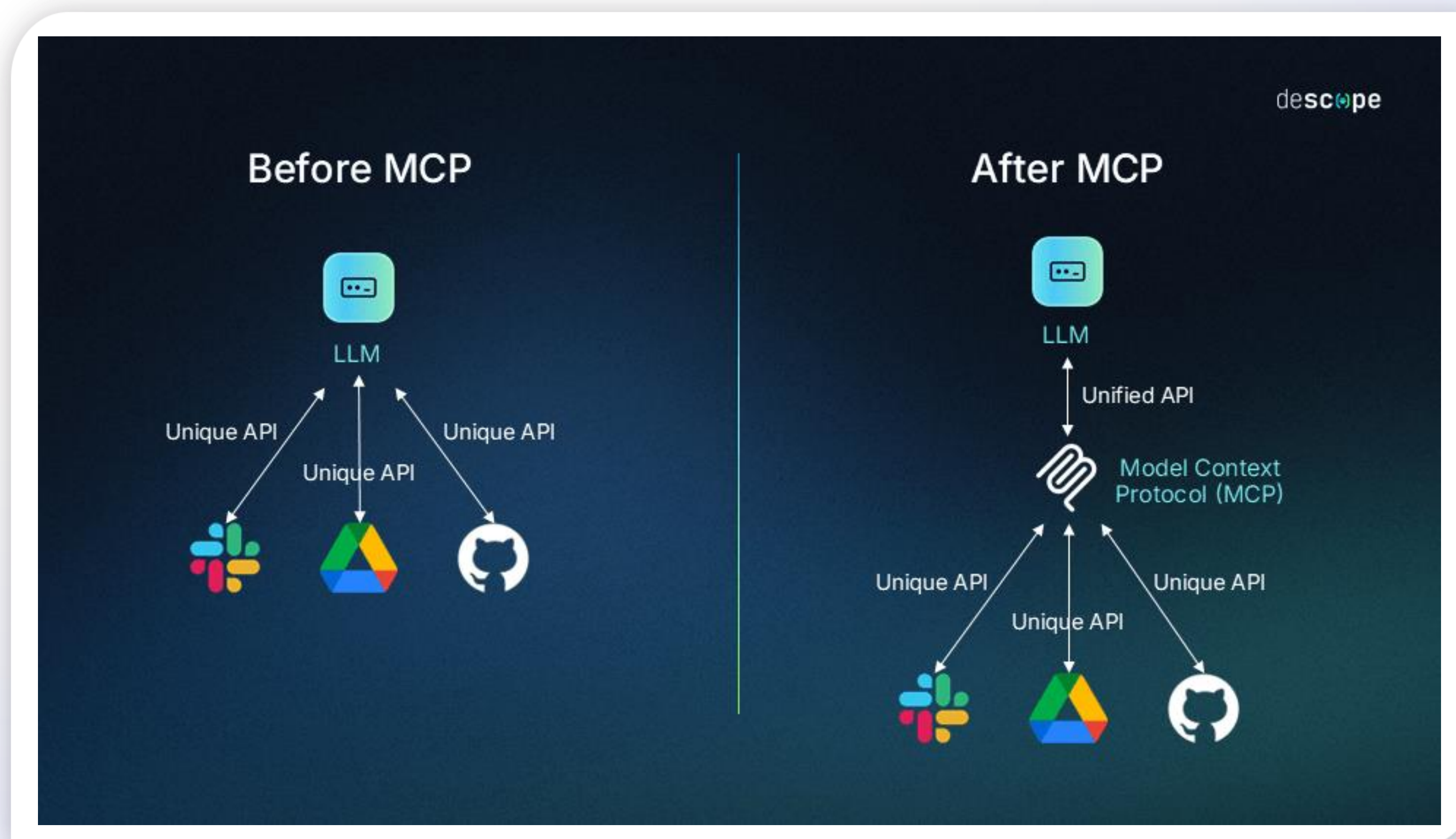
4

Специализированные агенты для агентов-помощников

Специализированные агенты будут существовать, но общаться не с людьми, а с другими агентами



# Что такое MCP



## MODEL CONTEXT PROTOCOL (MCP)

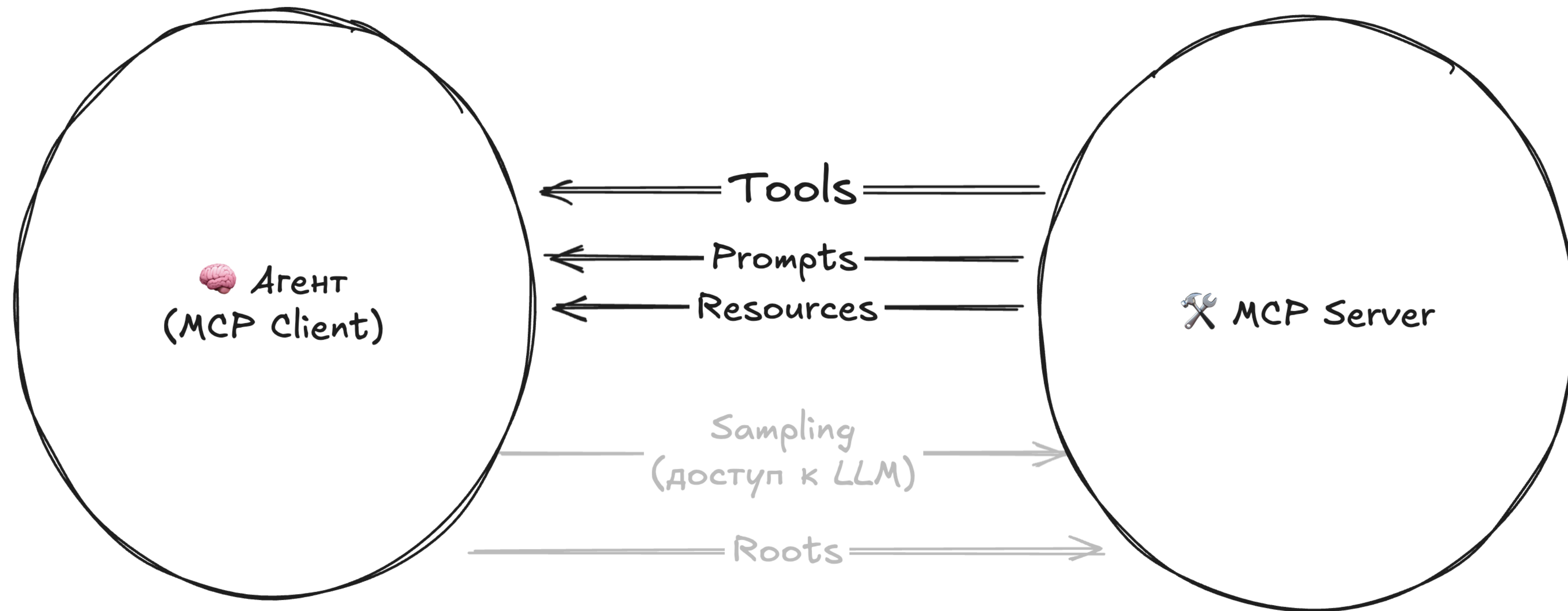
### Model Context Protocol (MCP)

MCP is an open protocol that standardizes how applications provide context to LLMs. Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect your devices to various peripherals and accessories, MCP provides a standardized way to connect AI models to different data sources and tools.

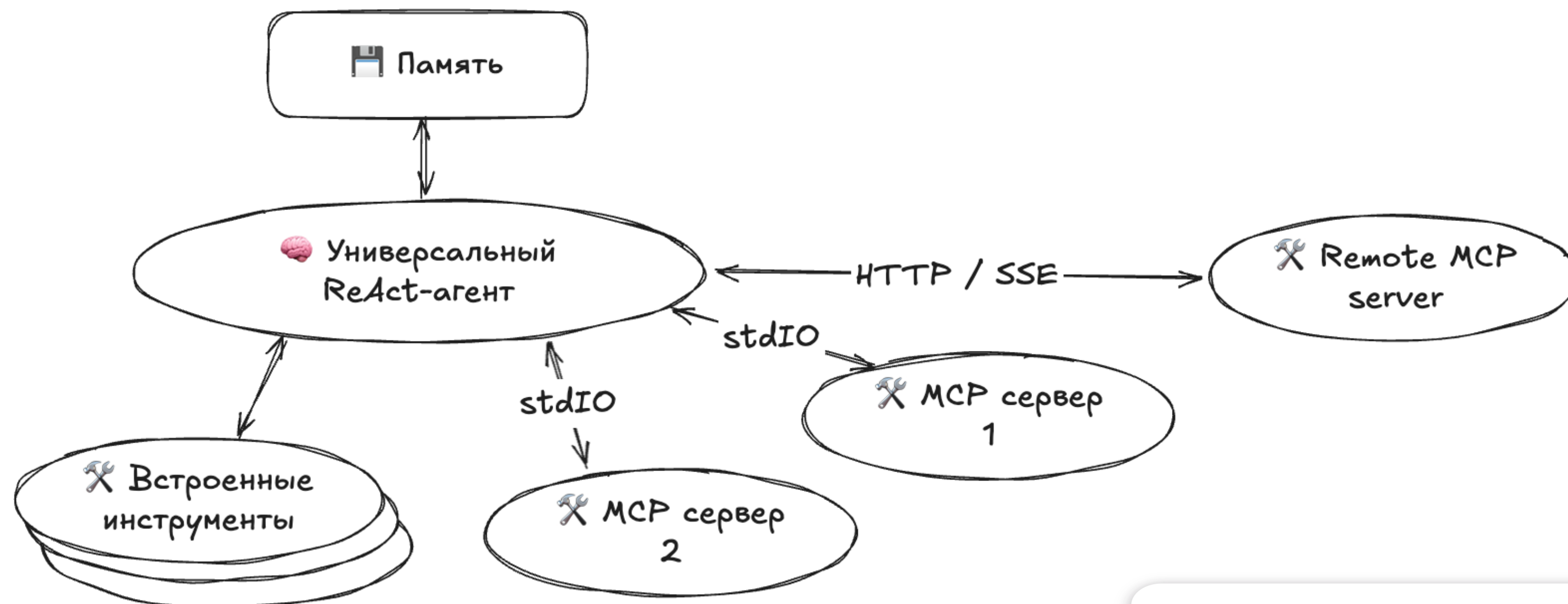
## Anthropic

Today, we're open-sourcing the Model Context Protocol (MCP), a new standard for connecting AI assistants to the systems where data lives, including content repositories, business tools, and development environments. Its aim is to help frontier models produce better, more relevant responses.

# Концепты MCP



# МСР для универсальных агентов



## Важно:

МСР позволяет быстро «собрать» универсального агента под конкретную задачу

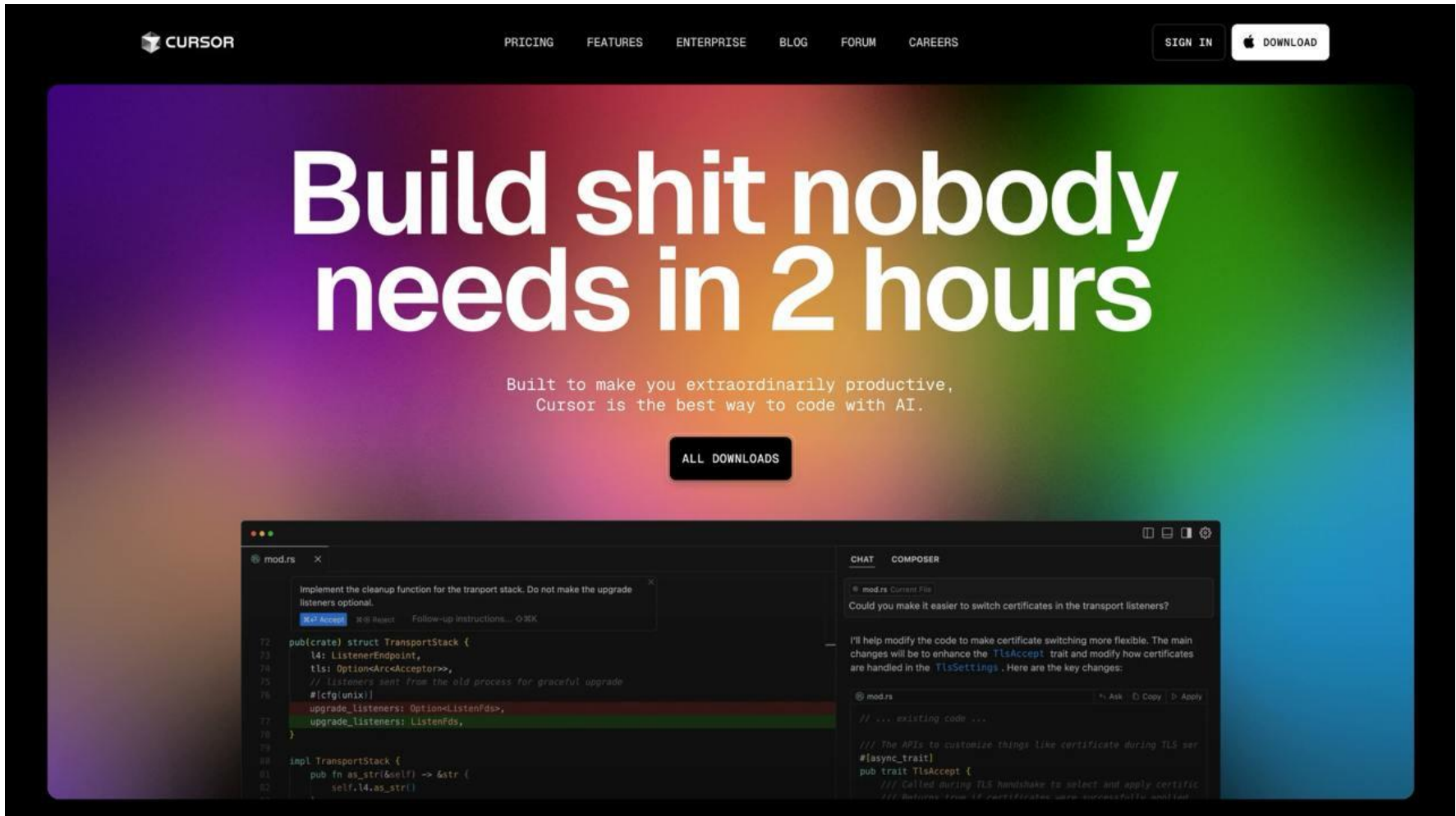


# Как воспользоваться MCP

Для работы с MCP нужен агент!

Самые популярные агенты с поддержкой MCP на сегодня:

- Cursor
- GitHub copilot (VSCode)
- Anthropic desktop
- ChatGPT (анонсировано)



Полный список клиентов от Anthropic:

Client	Resources	Prompts	Tools	Discovery	Sampling	Roots
5ire	✗	✗	✓	?	✗	✗
AgentAI	✗	✗	✓	?	✗	✗
AgenticFlow	✓	✓	✓	✓	✗	✗
Amazon Q CLI	✗	✓	✓	?	✗	✗
Apify MCP Tester	✗	✗	✓	✓	✗	✗
BeeAI Framework	✗	✗	✓	✗	✗	✗
BoltAI	✗	✗	✓	?	✗	✗
Claude.ai	✓	✓	✓	✗	✗	✗
Claude Code	✗	✓	✓	✗	✗	✗
Claude Desktop App	✓	✓	✓	✗	✗	✗
Cline	✓	✗	✓	✓	✗	✗



# Где брать MCP сервера?

README

MIT license

Awesome MCP Servers



Thai Click

English Click

繁體中文 點擊查看

简体中文 点击查看

日本語 クリック

한국어 클릭

discord 1.1k online

subreddit 13k

A curated list of awesome Model Context Protocol (MCP) servers.



• [What is MCP?](#)

• [Clients](#)

• [Tutorials](#)

• [Community](#)

• [Legend](#)

• [Server Implementations](#)

• [Frameworks](#)

• [Utilities](#)

• [Tips & Tricks](#)

What is MCP?

[MCP](#) is an open protocol that enables AI models to securely interact with local and remote resources through standardized server implementations. This list focuses on production-ready and experimental MCP servers that extend AI capabilities through file access, database connections, API integrations, and other contextual services.

## Clients

## Model Context Protocol servers

This repository is a collection of *reference implementations* for the [Model Context Protocol](#) (MCP), as well as references to community built servers and additional resources.

The servers in this repository showcase the versatility and extensibility of MCP, demonstrating how it can be used to give Large Language Models (LLMs) secure, controlled access to tools and data sources. Each MCP server is implemented with either the [Typescript MCP SDK](#) or [Python MCP SDK](#).

Note: Lists in this README are maintained in alphabetical order to minimize merge conflicts when adding new items.

### Reference Servers

These servers aim to demonstrate MCP features and the TypeScript and Python SDKs.

- [AWS KB Retrieval](#) - Retrieval from AWS Knowledge Base using Bedrock Agent Runtime
- [Brave Search](#) - Web and local search using Brave's Search API
- [EverArt](#) - AI image generation using various models
- [Everything](#) - Reference / test server with prompts, resources, and tools
- [Fetch](#) - Web content fetching and conversion for efficient LLM usage
- [Filesystem](#) - Secure file operations with configurable access controls
- [Git](#) - Tools to read, search, and manipulate Git repositories
- [GitHub](#) - Repository management, file operations, and GitHub API integration
- [GitLab](#) - GitLab API, enabling project management
- [Google Drive](#) - File access and search capabilities for Google Drive
- [Google Maps](#) - Location services, directions, and place details
- [Memory](#) - Knowledge graph-based persistent memory system
- [PostgreSQL](#) - Read-only database access with schema inspection
- [Puppeteer](#) - Browser automation and web scraping



# Пример: github copilot

## MCP для документации:

```
"langgraph-docs-mcp": {
  "command": "uvx",
  "args": [
    "--from",
    "mcpdoc",
    "mcpdoc",
    "--urls",
    "LangChain:https://python.langchain.com/llms.txt",
    "--transport",
    "stdio"
  ]
}
```

## MCP SaluteSpeech:

```
"mcp-salutespeech": {
  "command": "uvx",
  "args": [
    "--from",
    "mcp_voice_salute",
    "mcp-salutespeech"
  ],
  "env": {
    "SALUTE_SPEECH": "YTI...Ng=="
  }
}
```

## Базовые инструкции для агента:

```
---
applyTo: '**'
---
Coding standards, domain knowledge, and preferences that AI should follow.

for ANY question about LangGraph, use the langgraph-docs-mcp server to help answer --
+ call list_doc_sources tool to get the available llms.txt file
+ call fetch_docs tool to read it
+ reflect on the urls in llms.txt
+ reflect on the input question
+ call fetch_docs on any urls relevant to the question
+ use this to answer the question

---
Используй синтез речи после каждого ответа, чтобы озвучить его.
```




**GitHub**  
Copilot

# Пример: github copilot

Без тулов ответ не правильный!

С помощью какой абстракции можно реализовать принцип human in the loop в langgraph?

mcp\_config.json:1-7 global.instructions.md

 **GitHub Copilot**

> Used 2 references

В LangGraph принцип human-in-the-loop реализуется с помощью абстракции **Tool** (иногда называемой "tool node" или "tool invocation"). В LangGraph вы можете определить специальный узел-графа, который:

- Останавливает автоматическое выполнение цепочки.
- Передаёт управление человеку (например, через пользовательский интерфейс, форму или ручное подтверждение).
- Получает результат от пользователя и возвращает его обратно в граф для дальнейшей обработки.

Обычно это реализуется через кастомный tool, который ожидает внешнего ввода, либо через узел с типом "human" или "pause", который явно требует вмешательства человека.

Add Context... mcp\_config.json:1-7 Current file

global Instructions

Ask Copilot

@ Ask GPT-4.1

CHAT




## Edit with Copilot

Agent Mode

Ask Copilot to edit your files in **agent mode**. Copilot will automatically use multiple requests to pick files to edit, run terminal commands, and iterate on errors.

Copilot is powered by AI, so mistakes are possible.  
Review output carefully before use.

 or type # to attach context

 Add Context...

global.instructions.md Current file

Прив



Agent

GPT-4.1





# Примеры кода

Эта презентация:





# GigaChain

downloads 26k/month downloads 14k/month downloads 746/month downloads 473/month stars 410

Набор решений для создания LLM-приложений  
и AI-агентов с GigaChat

- [Python](#): LangChain, LangGraph
- [JavaScript / TypeScript](#): LangChainJS, LangGraphJS
- [Java](#): LangChain4j, LangGraph4j

Дополнительно:

- [Gpt2giga](#) — прокси для перенаправления трафика OpenAI<->GigaChat
- [Набор MCP серверов](#)
- [Набор кукбуксов и примеров](#)

GitVerse:



GitHub:



# Код сервера

## Пример MCP-сервера на Anthropic SDK

```
from mcp.server.fastmcp import FastMCP
from pydantic import BaseModel
import sys

mcp = FastMCP("Math")

@mcp.tool()
def add(a: float, b: float) -> float:
    """Add two numbers"""
    print(f">>> Calling add({a}, {b})")
    return a + b

if __name__ == "__main__":
    transport = sys.argv[1] if len(sys.argv) > 1 else "stdio"
    mcp.run(transport=transport)
```

## mcp\_config.json

```
{
  "math": {
    "command": "python",
    "args": ["math_server.py"],
    "transport": "stdio"
  }
}
```

## Дополнительный тул для поиска людей в БД:

```
class Query(BaseModel):
    query: str

class Person(BaseModel):
    name: str
    age: int

@mcp.tool()
def find_person(name: Query) -> Person:
    """Find an info about some person by name"""
    print(f">>> Calling find_person({name})")
    return Person(name="John Doe", age=30)
```

# Код агента (MCP-клиент)

## 1. Подключаем GigaChat:

```
from langchain_gigachat import GigaChat

GIGA_CHAT = GigaChat(
    model="GigaChat-2-Max",
    verify_ssl_certs=False,
    streaming=False,
    max_tokens=8000,
)
```

## 2. Загружаем MCP сервера и создаем агента с тулами

```
with open("mcp_config.json", "r", encoding="utf-8") as f:
    MCP_CONFIG = json.load(f)

async with MultiServerMCPClient(MCP_CONFIG) as mcp_client:
    agent = create_react_agent(
        GIGA_CHAT,
        tools=mcp_client.get_tools(),
        prompt="Ты полезный ассистент.",
        checkpoint=MemorySaver()
    )
```

## 3. Ведём диалог с пользователем

```
while True:
    user_input = input("User: ")
    response = await agent.ainvoke(
        {"messages": [{"role": "user", "content": user_input}]}
    )
    print(f"Agent: {response['messages'][-1].content}")
```



# Пример работы агента

Пример работы:

Вызов тула



Тул + память



```
(.venv) % python mcp_react_agent.py
User: Привет
Agent: Здравствуйте! Чем могу быть полезен?
User: Сколько лет Джону Доу
🔧 Tool: find_person | Args: {'name': {'query': 'Джон Доу'}}
Agent: Джон Доу 30 лет.
User: Умножь его возраст на Пи
🔧 Tool: multiply | Args: {'a': 30, 'b': 3.14159}
Agent: Возраст Джона Доу, умноженный на число Пи, примерно равен 94.25.
```

# Think-тул

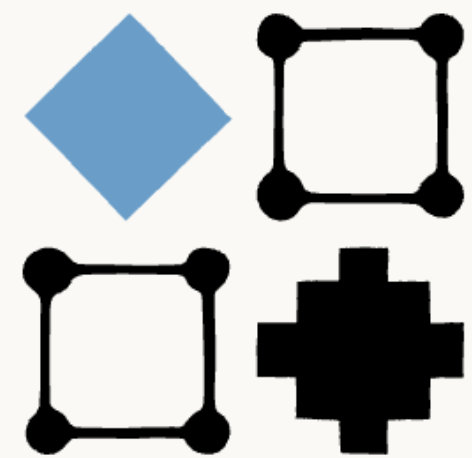
Anthropic предложил подход – добавить агенту функцию-заглушку think, которая помогает модели рассуждать

- Агент сможет самостоятельно принимать решение, когда нужно думать и когда прекратить
- Агент может возвращаться к размышлениям несколько раз за время задачи
- В итоге это приводит к повышению качества

Статья



Engineering at Anthropic



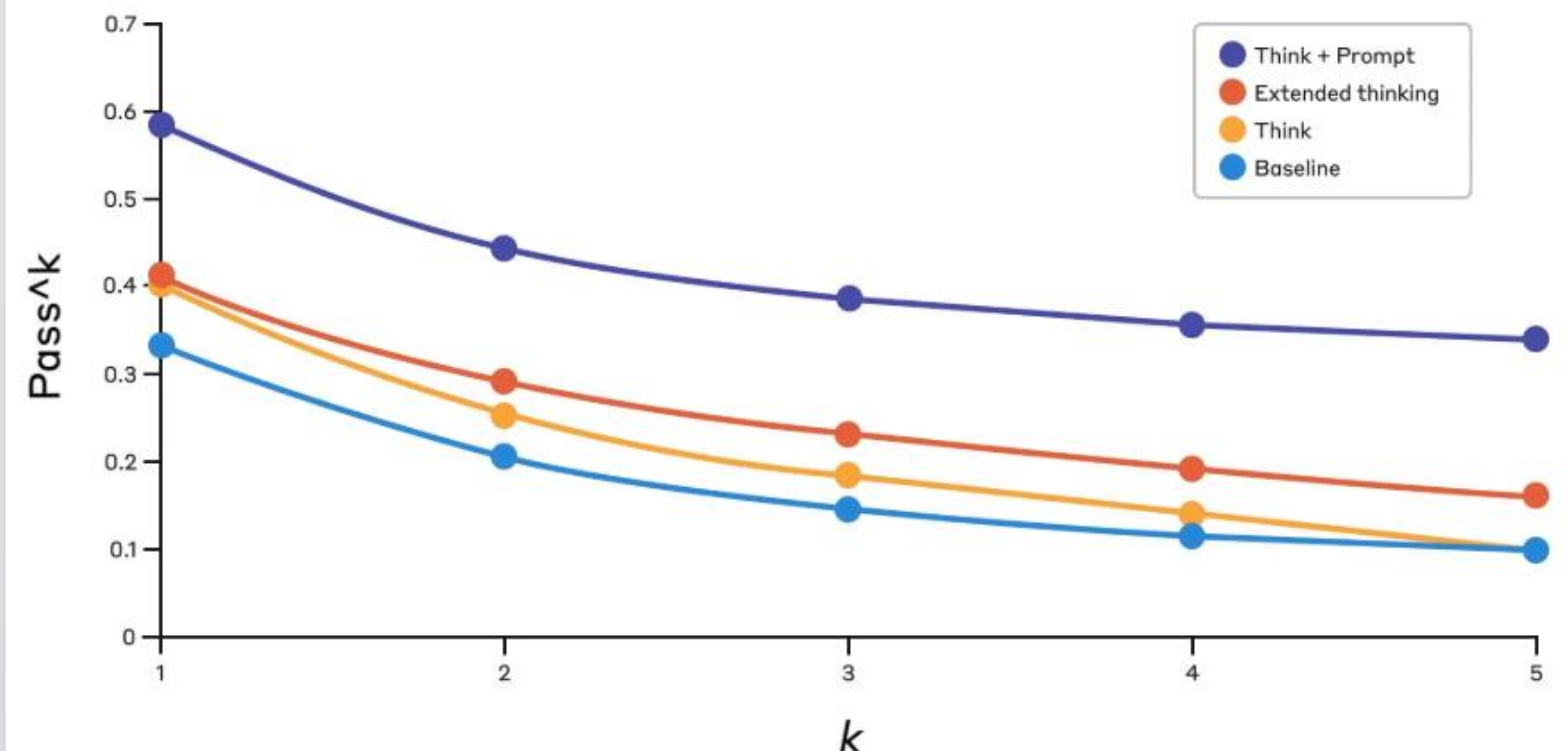
## The "think" tool: Enabling Claude to stop and think in complex tool use situations

Published Mar 20, 2025

A new tool that improves Claude's complex problem-solving performance

As we continue to enhance Claude's complex problem-solving abilities, we've discovered a particularly effective approach: a "think" tool that creates dedicated space for structured thinking during complex tasks.

Claude 3.7 Sonnet performance on airline task





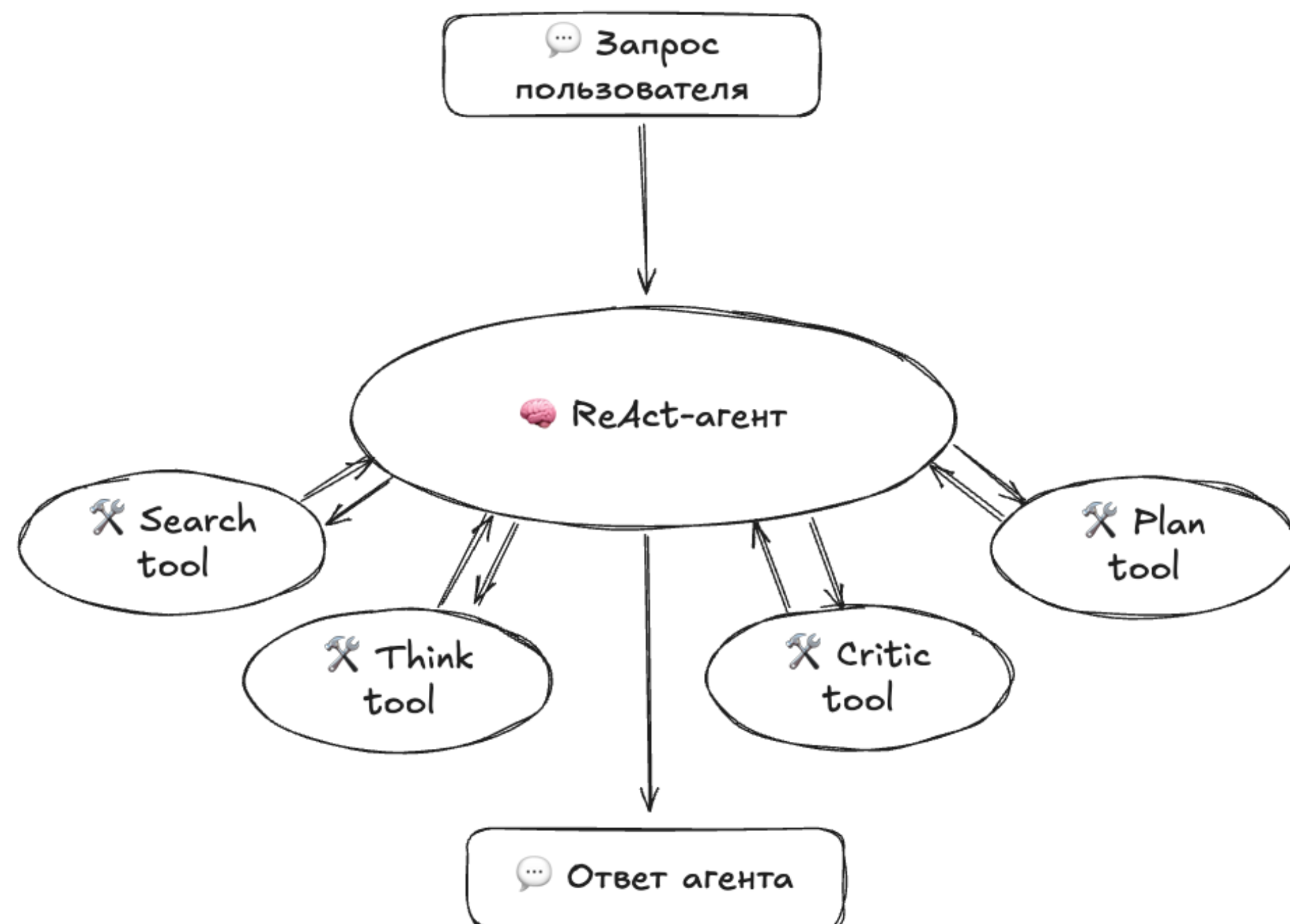
# Пример работы с GigaChat-2-Max

**Запрос:** Сравни чистую прибыль 3 компаний за 2024 год - Сбера, ВТБ и Альфа-Банка

**Ответ:** Чистая прибыль Сбера за 2024 год составила 1580,3 млрд руб., ВТБ - 551,4 млрд руб., Альфа-Банка - 199,61 млрд руб. Таким образом, Сбербанк имеет наибольшую чистую прибыль среди этих трех компаний, за ним следует ВТБ, а затем Альфа-Банк.

**Как действовал агент:**

- **Запрос**
- **Plan**(Сначала найду информацию о чистой прибыли Сбера, ВТБ и Альфа-Банка за 2024 год, а затем сравню полученные данные.)
- **Search**(чистая прибыль Сбера ВТБ Альфа-Банк 2024)
- **Think**(Из результатов поиска я получил информацию о чистой прибыли Сбера за 2024 год, но не нашел данных о чистой прибыли ВТБ и Альфа-Банка. Нужно найти эту информацию.)
- **Search**(чистая прибыль ВТБ 2024)
- **Think**(Теперь у меня есть информация о чистой прибыли Сбера и ВТБ за 2024 год. Осталось найти данные о чистой прибыли Альфа-Банка. После этого я смогу сравнить прибыль всех трех компаний.)
- **Search**(чистая прибыль Альфа-Банк 2024)
- **Think**(Теперь у меня есть информация о чистой прибыли всех трех компаний за 2024 год. Я могу сравнить их. Чистая прибыль Сбера за 2024 год составила 1580,3 млрд руб., ВТБ - 551,4 млрд руб., Альфа-Банка - 199,61 млрд руб. Сбербанк имеет наибольшую чистую прибыль, за ним следует ВТБ, а затем Альфа-Банк.)
- **Ответ**
- ...





# MCP think-tool

Позволяет добавить функцию рассуждений практически к любому не-reasoning агенту

```
"think-mcp": {  
  "command": "uvx",  
  "args": ["think-mcp"]  
}
```

Можно добавить в своих агентов

Можно подключить к коммерческим решениям: **Cursor**,  
**VSCode**, и даже **Anthropic Desktop**

Каталог Awesome MCP Servers (40k 🌟) уже добавил этот тул себе

## Awesome MCP Servers 🕶 awesome

Thai Click English Click 繁體中文 點擊查看 简体中文 点击查看 日本語 クリック 한국어 클릭 Português Brasileiro Clique  
discord 1k online subreddit 15k

A curated list of awesome Model Context Protocol (MCP) servers.

GitHub:



GitVerse:





# Красивый cherry pick

**3 человека хотят поровну разделить между собой 2 яблока. Но у них есть только 1 нож, которым, согласно условию задачи, они могут воспользоваться всего 1 раз. Что нужно сделать для того, чтобы каждый человек получил равную часть яблок?**



## Разделение яблок между тремя лю

Реши задачу

3 человека хотят поровну разделить между собой 2 яблока. Но у них есть только 1 нож, которым, согласно условию задачи, они могут воспользоваться всего 1 раз. Что нужно сделать для того, чтобы каждый человек получил равную часть яблок? 🤔

Задача требует разделить 2 яблока поровну между 3 людьми, используя нож только 1 раз.

Решение: нужно сложить оба яблока вместе (одно на другое) и разрезать их ножом одним движением пополам. В результате получится 4 половинки яблок. Каждому человеку достанется по 1 половинке, а оставшуюся 1 половинку нужно разделить между ними поровну (каждому по  $1/6$  от общего количества яблок).

Альтернативное решение: одно яблоко разрезать на 3 равные части одним движением ножа (поставив яблоко вертикально и разрезав его на 3 сектора), а второе яблоко разделить без использования ножа.

## Равное деление яблок

Подумай, используя think-mcp, и реши задачу

3 человека хотят поровну разделить между собой 2 яблока. Но у них есть только 1 нож, которым, согласно условию задачи, они могут воспользоваться всего 1 раз. Что нужно сделать для того, чтобы каждый человек получил равную часть яблок?

I'll think through this problem step by step. 👍

> Called MCP tool think ✓

Нужно положить яблоки друг на друга и разрезать их одним движением ножа на отметке  $1/3$ . Тогда получим:

- 2 куска по  $1/3$  яблока (меньшие части)
- 2 куска по  $2/3$  яблока (большие части)

Распределение:

- Первый человек получает одну часть размером  $2/3$  яблока
- Второй человек получает одну часть размером  $2/3$  яблока
- Третий человек получает две части по  $1/3$  яблока, что в сумме равно  $2/3$  яблока

Таким образом, каждый получит равную долю - по  $2/3$  яблока.



# Наши MCP серверы

- **Think MCP:** LLM-ризонинг

Помогает в решении сложных reasoning-задач

- **MCP Giga Checker:** проверка на человека

Инструмент для юрлиц, определяющий создан ли текст человеком или сгенерирован с помощью ИИ

- **MCP Kandinsky:** из промпта в PNG за 10 сек

Генерации изображений с помощью API Kandinsky

- **MCP SaluteSpeech:** добавляем голос, синтез и распознавание в 3 строчки кода

Голосовой ассистент на базе LLM

GitVerse:



GitHub:







# Как измерить качество AI-агента

# Инструменты для валидации

Для замеров и экспериментов удобно использовать инструменты:

## — LangSmith

- Платный
- Готовое cloud-решение
- От команды LangChain, максимальный функционал

## — Arize Phoenix

- Бесплатный opensource (elastic license)
- Self-hosted, можно запустить в jupyter notebook
- Содержит все основные функции для замеров

# Загрузка датасета

## 1. Сформировать датасет

```
EXAMPLES = [  
    {"answer": 80, "question": "23+57=?", "topic": "Basic Math"},  
    {"answer": 96, "question": "12×8=?", "topic": "Basic Math"},  
    {  
        "answer": 1070,  
        "question": "1+100−50+1000−500+25−12+7−3+100000",  
        "topic": "Basic Math",  
    },  
    {  
        "answer": 125,  
        "question": "10+((10÷(2×3))+9/(3÷5))×5÷(1+(2÷2))×3",  
        "topic": "Basic Math",  
    },  
    ...  
]
```

## 2. Загрузить датасет

```
from langsmith import Client  
  
client = Client()  
dataset = client.create_dataset(  
    dataset_name="Simple Math Problems",  
    description="Evaluate ability to solve simple math problems.",  
)  
  
client.create_examples(  
    inputs=[{"question": record["question"]} for record in EXAMPLES],  
    outputs=[{"answer": record["answer"]} for record in EXAMPLES],  
    metadata=[{"topic": record["topic"]} for record in EXAMPLES],  
    dataset_id=dataset.id,  
)
```

Пример для начала – [agent evals](#):





# Загрузка датасета

## 3. Результат загрузки

Simple Math Problems

Evaluate ability to solve simple math problems.

ID

Latest

Add Evaluator

+ Experiment

ExperimentsPairwise Experiments**Examples**Few-shot search

Filters

Select split

+ Example

<input type="checkbox"/>	Inputs	Reference Outputs	Created At	Modified At	Splits	{...} topic	
<input type="checkbox"/>	Convert -2/3 pi radians into degrees	-120	17.04.2025, 18:45:22	17.04.2025, 18:45:22	base	Trigonometry	
<input type="checkbox"/>	What is the square root of 256?	16	17.04.2025, 18:45:22	17.04.2025, 18:45:22	base	Intermediate Math Proble	
<input type="checkbox"/>	12×8=?	96	17.04.2025, 18:45:22	17.04.2025, 18:45:22	base	Basic Math	
<input type="checkbox"/>	23+57=?	80	17.04.2025, 18:45:22	17.04.2025, 18:45:22	base	Basic Math	

# Замер

## 1. Функция проверки ответа

```
def correct(inputs: dict, outputs: dict, reference_outputs: dict) -> int:
    actual_float = float(outputs["value"])
    reference_float = float(reference_outputs["answer"])

    is_correct = isclose(actual_float, reference_float, rel_tol=1e-1, abs_tol=1e-1)
    return 1 if is_correct else 0
```

## 2. Выполнение прогона

```
def make_agent_runner():
    agent_graph = create_agent()

    def run_agent(inputs: dict):
        result = agent_graph.invoke(
            {"messages": [{"role": "human", "content": inputs["question"]}]}
        )
        return {
            "value": result['messages'][-1].content
        }

    return run_agent

dataset = client.read_dataset(dataset_name="Simple Math Problems")
run_agent = make_agent_runner()
evaluate(run_agent,
        data=dataset,
        experiment_prefix="GigaCaht_think_tool",
        evaluators=[correct])
```

## 3. Результат прогона

Simple Math Problems

Evaluate ability to solve simple math problems.

ExperimentsPairwise ExperimentsExamplesFew-shot search

Q Search by name...Chartsx-axis dataset\_versionFilters

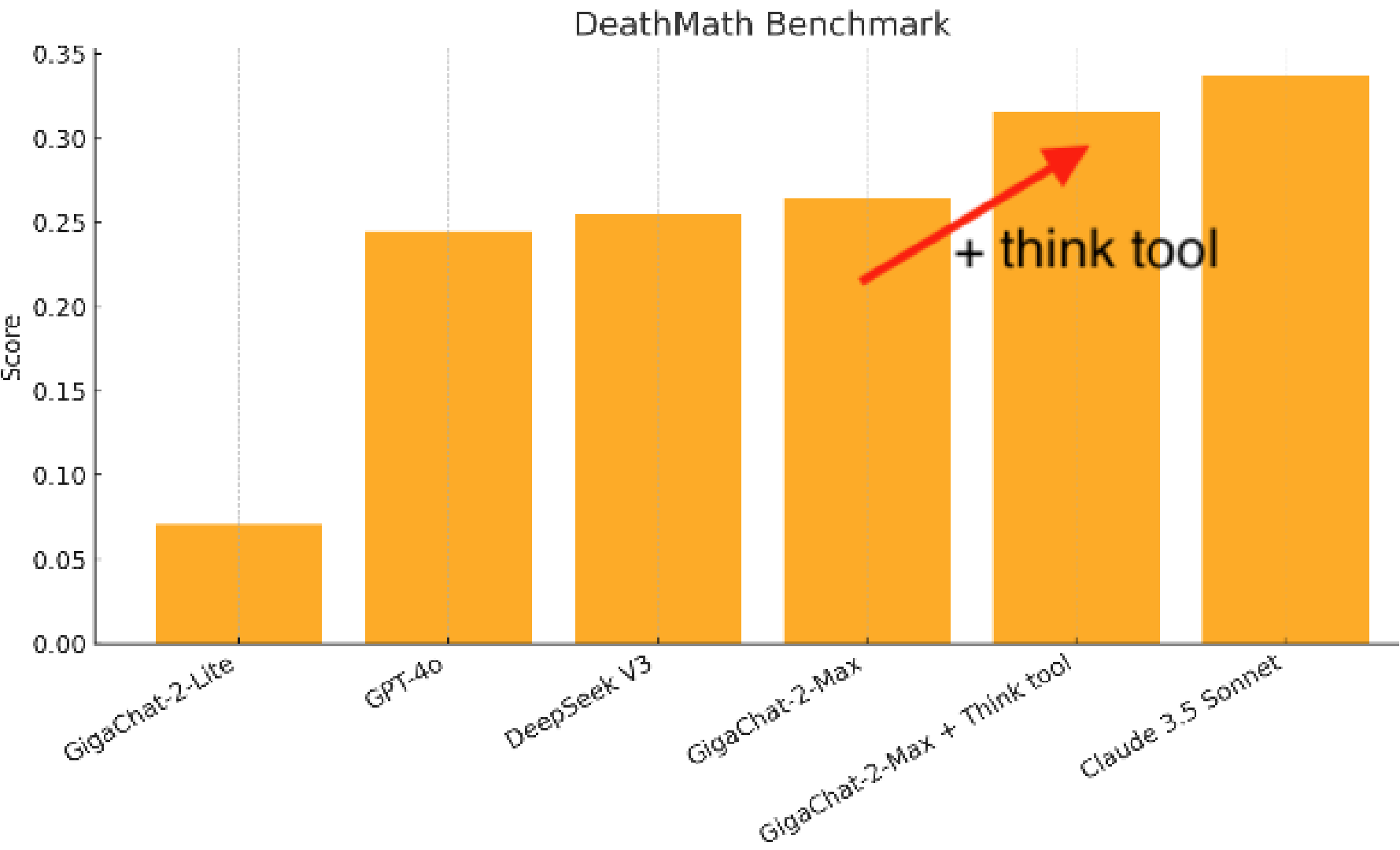
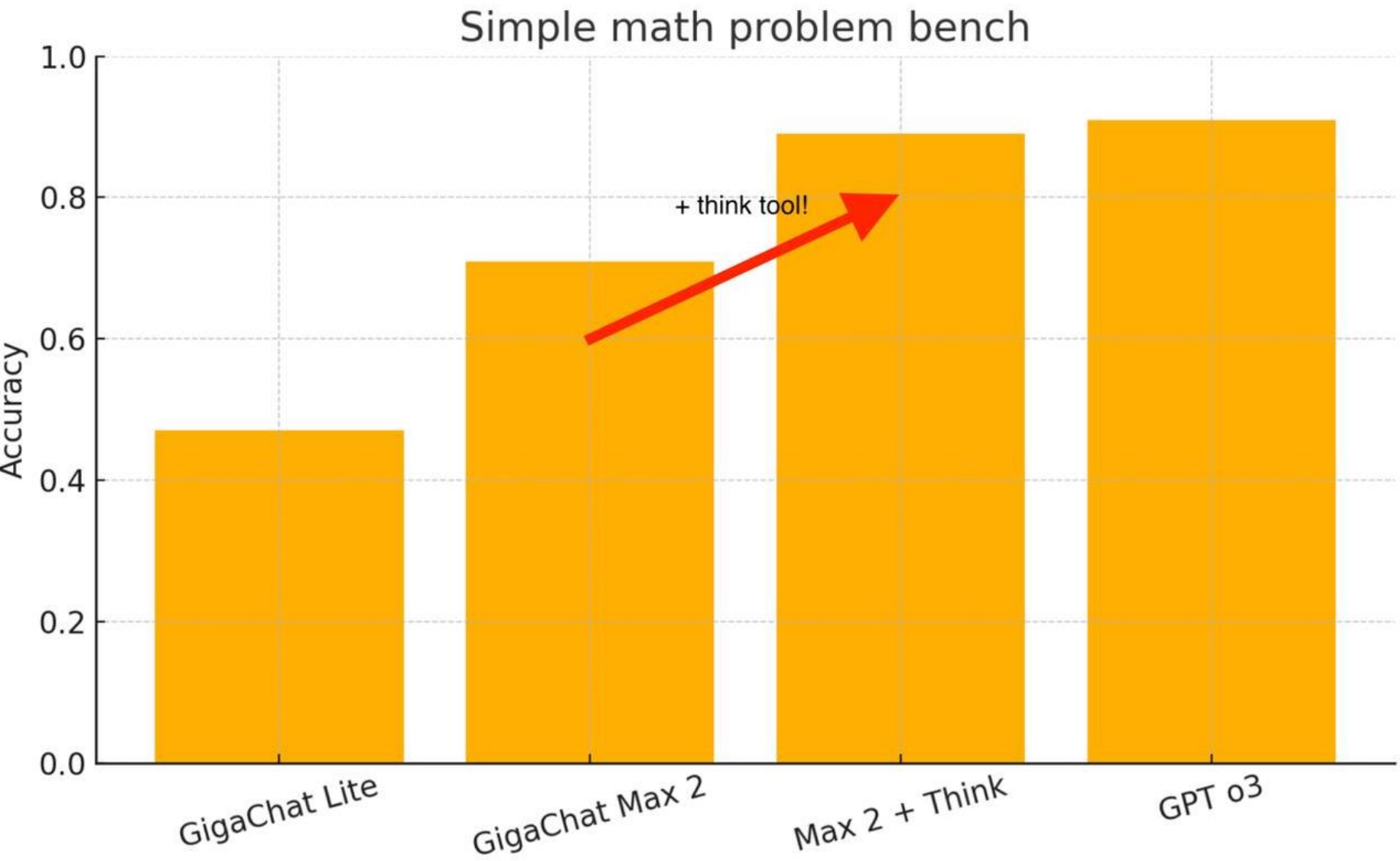
Correct

Experiment	Correct
#23 aching-view-18	0.42
#24 ordinary-tea-70	0.71
#29 scholarly-education-76	0.68
#31 yellow-lip-17	0.92

Experiment	Splits	Repetitions	Correct
#31 yellow-lip-17	base	1	0.92
#29 scholarly-education-76	base	1	0.68
#24 ordinary-tea-70	base	1	0.71
#23 aching-view-18	base	1	0.42



# Результаты замеров



## Выводы:

- На некоторых бенчах явный рост (физика, математика, где нужен счет)
- На других бенчах без изменений (например, Aider benchmark)



## DeathMath benchmark (физика на русском языке)

model	score	math_score	physics_score
Gemini 2.0 Flash	0.514	0.558	0.469
Claude 3.7 Sonnet	0.47	0.542	0.398
gpt-4.1	0.466	0.584	0.347
Gemma 3 27B	0.4	0.474	0.327
Claude 3.5 Sonnet	0.376	0.416	0.337
DeepSeek R1 Distill Qwen 14B	0.346	0.447	0.245
DeepSeek V3 0324	0.343	0.432	0.255
gpt-4o	0.338	0.432	0.245
<b>GigaChat-2-Max-Think-Tool</b>	<b>0.337</b>	<b>0.357</b>	<b>0.316</b>
GigaChat-2-Max	0.314	0.363	0.265
GigaChat-2-Pro	0.27	0.316	0.224
Qwen2.5 72B Instruct	0.189	0.379	0
GigaChat-Max	0.181	0.189	0.173
Gemma 3 4B	0.18	0.258	0.102
GigaChat-2	0.083	0.095	0.071





# Спасибо за внимание!



Константин Крестников  
Техлид команды GigaChain

Эта презентация:



Мой Telegram канал про ИИ  
[@Robofuture](#)

