RAG：从检索增强到智能涌现

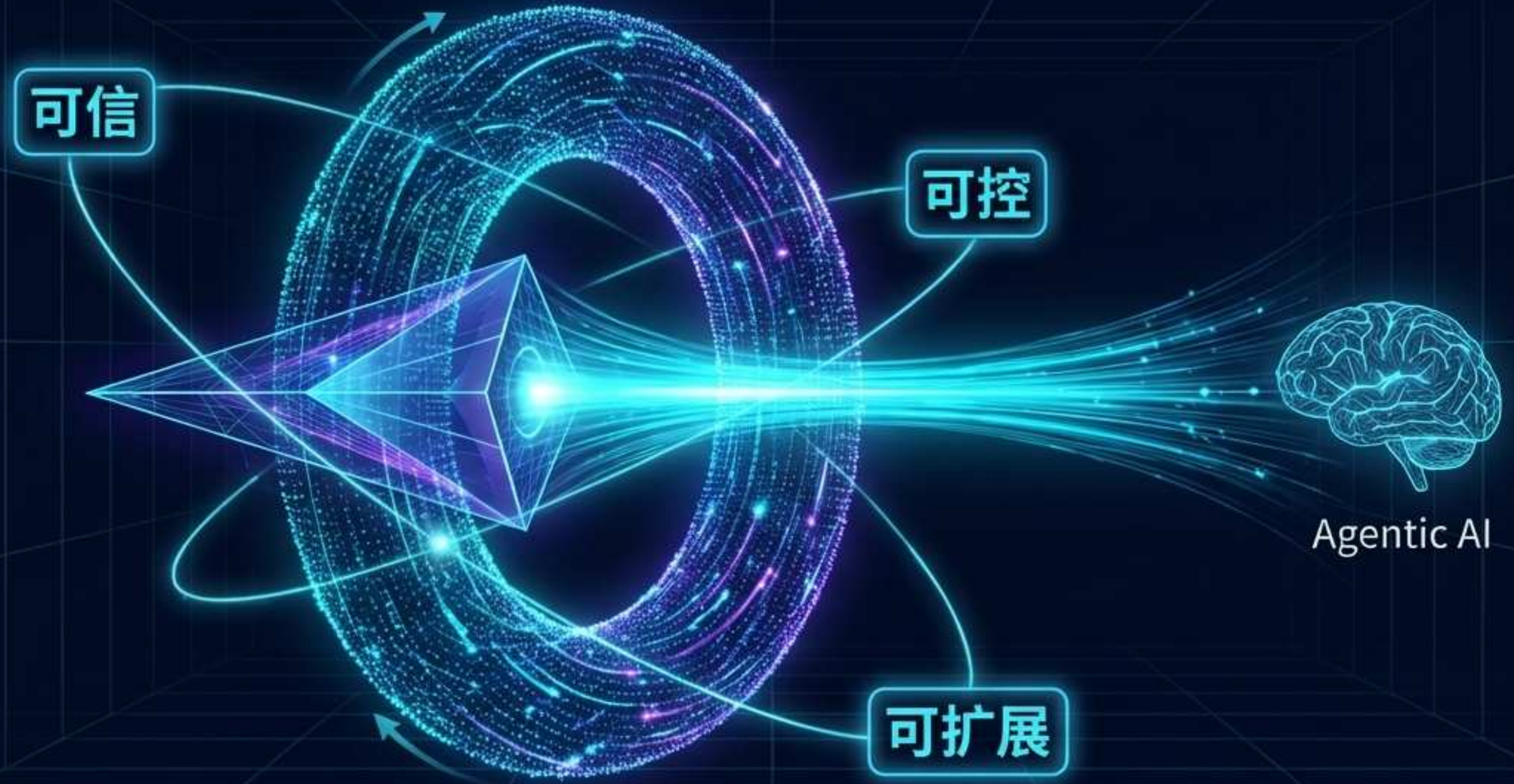
Agentic AI 的认知控制平面与知识基石

深圳市网新新思软件有限公司 金鹏

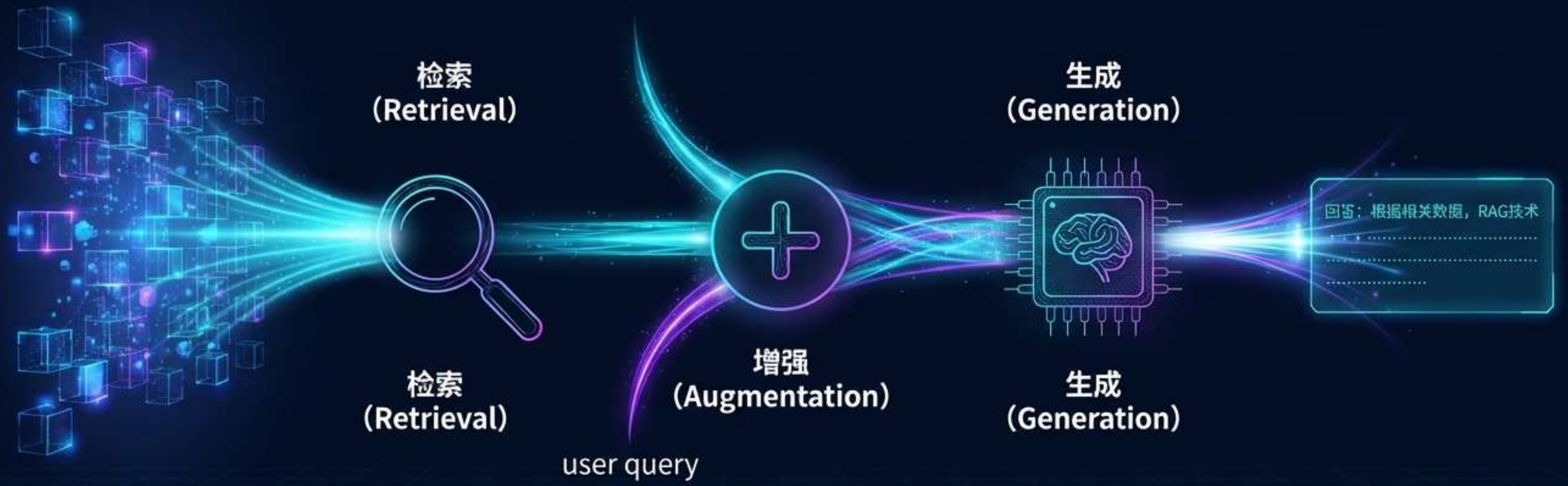
议程：RAG 涌现之路的技术主线

1. **战略价值：** RAG为何是构建未来AI的基石？
2. **挑战与评估：** 我们必须正视的内在困境与科学度量。
3. **检索基石：** 向量数据库的核心技术选型。
4. **边界突破：** 高级RAG算法的工程实践。
5. **智能集成：** RAG如何赋能Agentic AI。

RAG的本质：可信、可控的知识接口



RAG核心能力：检索、增强、生成



战略价值（一）：知识时效性与私有性

静态训练数据

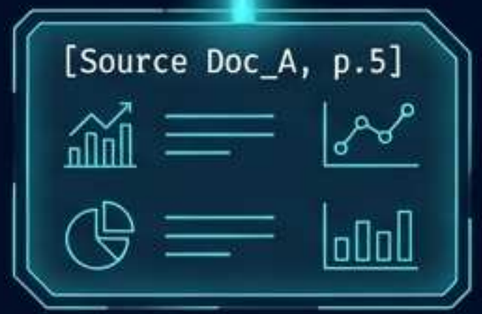
时效性

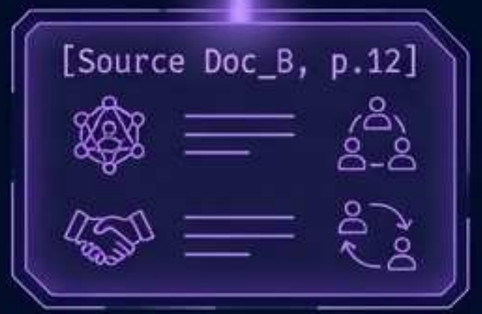
私有性

实时/私有数据

战略价值（二）：可信度与可审计性

关于项目A的市场前景，基于目前的趋势分析，预计在未来三年内将实现显著增长，特别是在亚太地区。根据[Source Doc_A, p.5]的数据，市场规模预计将达到50亿美元。此外，关键技术合作伙伴关系将是成功的关键，如[Source Doc_B, p.12]所述。尽管存在竞争风险，但我们的独特技术优势提供了强大的壁垒，这在[Source Doc_C, p.2]中有详细说明。





战略价值（三）：上下文长度与成本优化

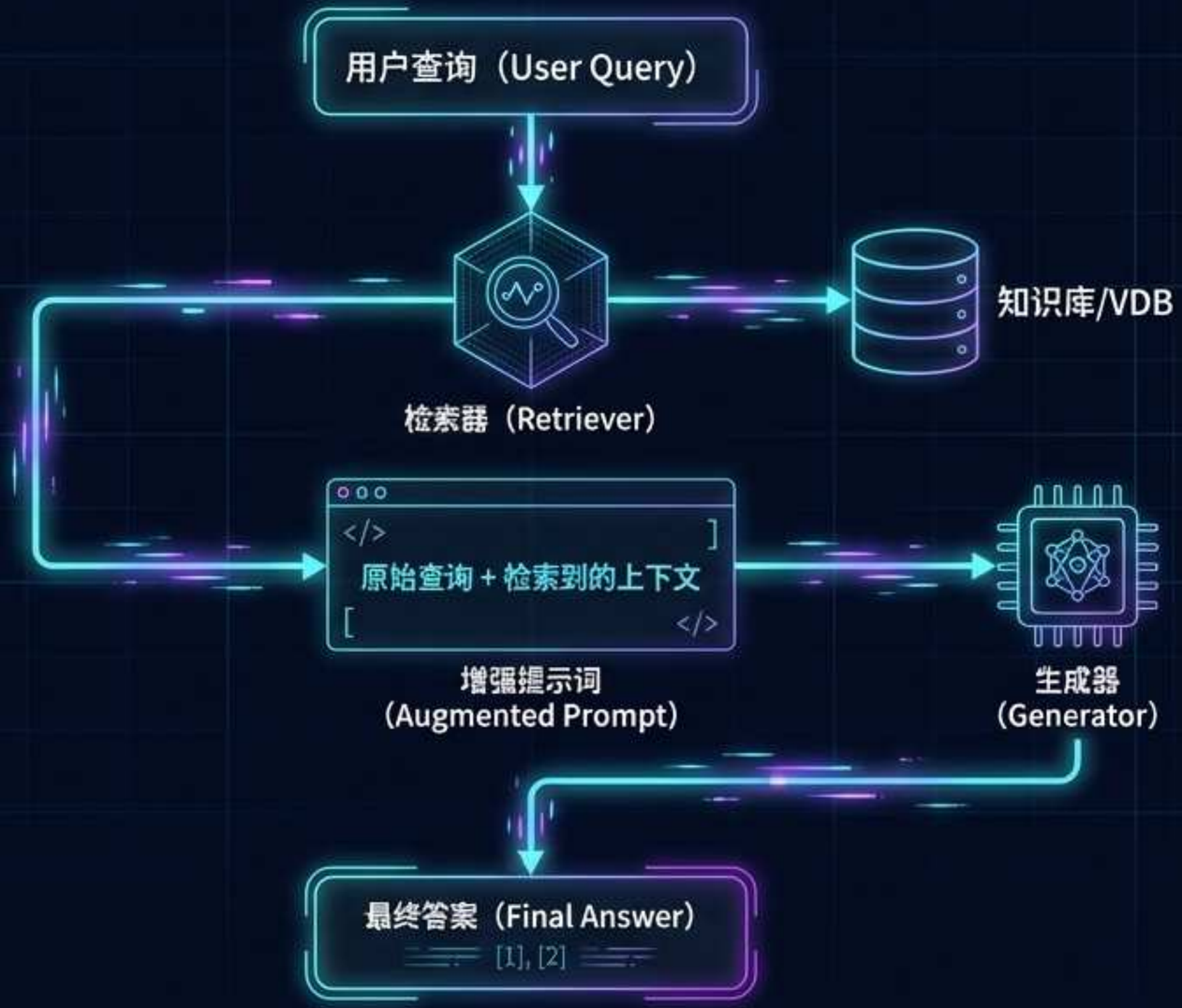
长上下文方法

$$$

RAG方法

$

RAG的基础架构：R+G 工作流解构



RAG的架构边界：它不是Agent，不能做推理

核心应用场景：高合规性与知识依赖

客户支持

ChatBI

法律研究

事实证据

可审计性

RAG的内在劣势：三维困境

效果困境：“垃圾进，垃圾出”的质量悖论

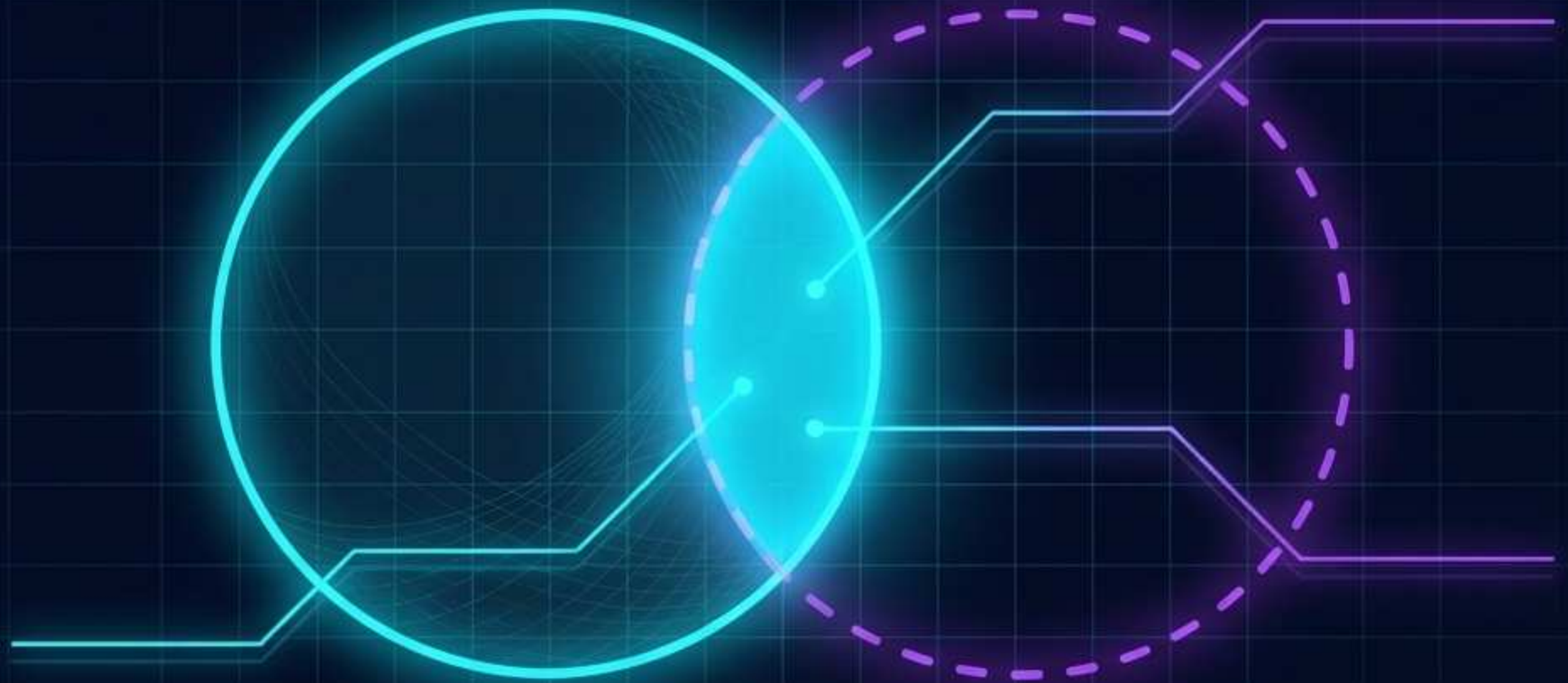
评估基石：RAGAS框架与科学度量

深圳市网新新思软件有限公司 金鹏

评估输入：RAGAS的四要素

检索器评估（一）：上下文精度（Context Precision）

检索器评估（二）：上下文召回率（Context Recall）



生成器评估：忠实度是杜绝幻觉的生命线

检索挑战：语义鸿沟与词汇不匹配

效率瓶颈（一）：Chunking的权衡悖论

小块 / Small Chunks

大块 / Large Chunks

优点：精度高

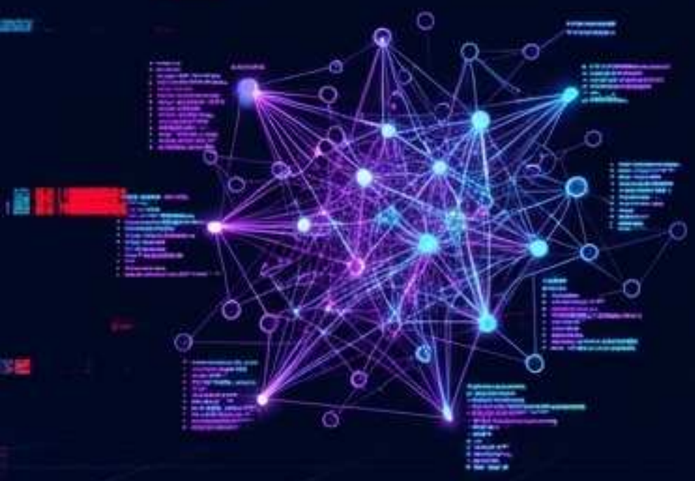
缺点：上下文缺失

优点：上下文完整

缺点：噪声干扰，成本高

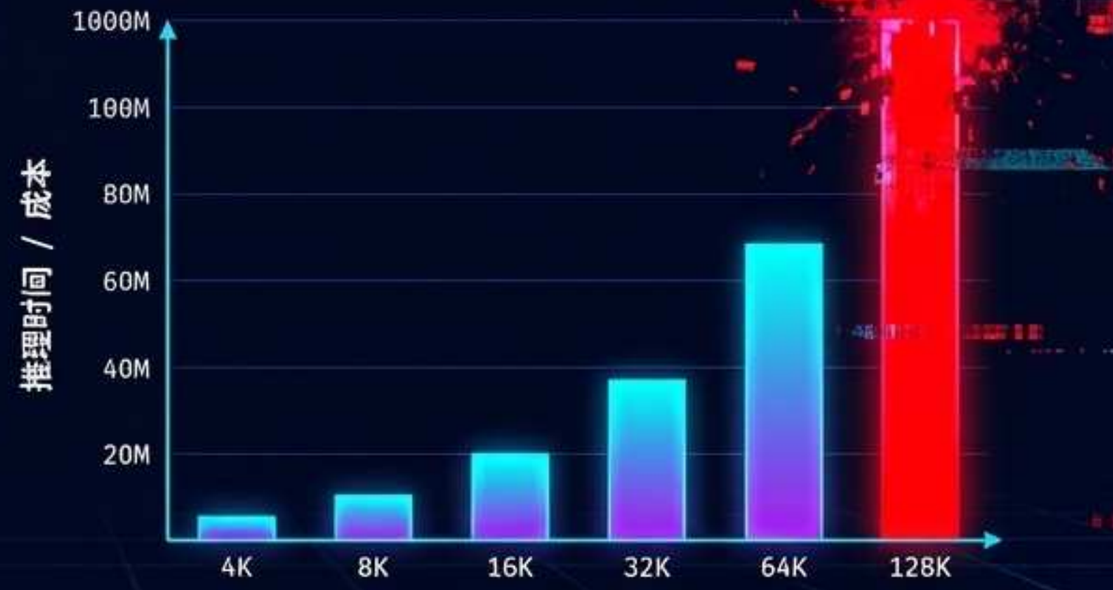
效率瓶颈（二）：算力激增与长上下文代价

算力激增



System Load: CRITICAL
Resource Utilization: >95%
CPU/GPU: MAX CAPACITY

长上下文代价



Exponential Cost & Latency
Budget Exceeded
Unscalable

检索基石：向量数据库作为智能体的长期记忆支柱

向量数据库（VDB）是RAG架构的绝对基石，它超越了传统数据库的角色，成为AI智能体（Agent）的长期记忆系统（Long-Term Memory，LTM）。通过将知识、对话历史和经验转化为高维向量，VDB实现了基于语义相似度的瞬时检索，为智能体提供了持续学习和调用海量记忆的能力。

深圳市网新新思软件有限公司 金鹏

统一知识平台：Elasticsearch

Elasticsearch的定位是一个统一的、企业级的知识检索平台。它并非纯粹的向量数据库，而是将三种核心检索能力无缝整合在一个强大的技术栈中：

- 全文搜索（BM25）：业界领先的关键词匹配能力。
- 向量搜索（kNN）：实现基于语义的相似度检索。
- 结构化分析（Metadata Filtering）：精准的元数据过滤与聚合。

核心优势：原生混合搜索（Hybrid Search）

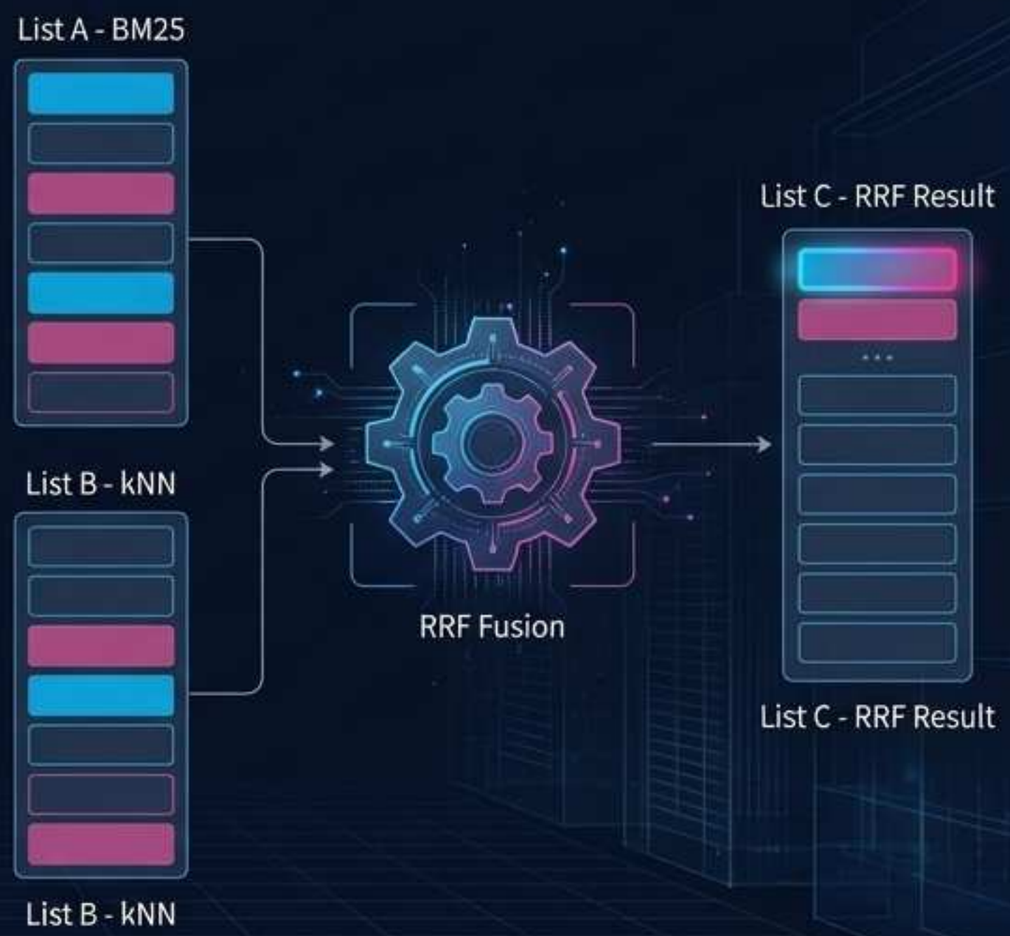
Elasticsearch的关键架构优势在于其原生的混合搜索能力。它允许在一个查询请求中，并行执行两种互补的搜索算法：

- **BM25关键词搜索**：精准匹配特定术语和实体，解决向量搜索在精确性上的短板。
- **kNN向量搜索**：理解查询的深层语义和意图，弥补关键词的词汇鸿沟。

这种“双引擎”模式极大提升了检索的召回率和相关性。

智能融合算法：倒数排名融合（RRF）

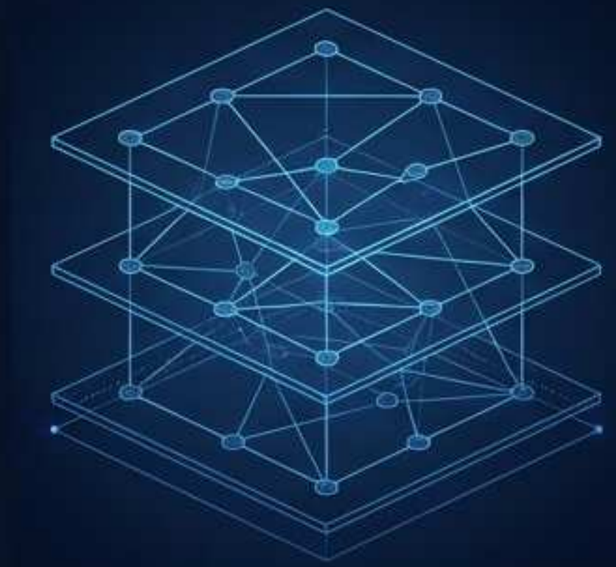
为了智能地合并来自BM25和kNN这两个异构评分系统的结果，Elasticsearch采用了倒数排名融合（Reciprocal Rank Fusion，RRF）算法。RRF的核心优势在于其“零配置”的智能性：它不依赖于复杂的权重调整或分数归一化，而是基于结果的排名（Rank）进行融合。这使得它它在各种场景下都表现得非常稳健和高效，显著降低了工程调优的复杂度。



架构性能优化：HNSW 与 DiskBBQ 量化

Elasticsearch在向量检索性能方面持续演进，采用了多种关键优化策略：

- HNSW算法：采用业界标准的高效近似最近邻（ANN）算法，确保低延迟和高吞吐。
- 向量量化：支持`int8`等量化策略，大幅减少内存占用。
- DiskBBQ（bbq_disk）：实现了基于磁盘的向量检索，极大降低了对昂贵内存资源的依赖，优化了TCO。



HNSW

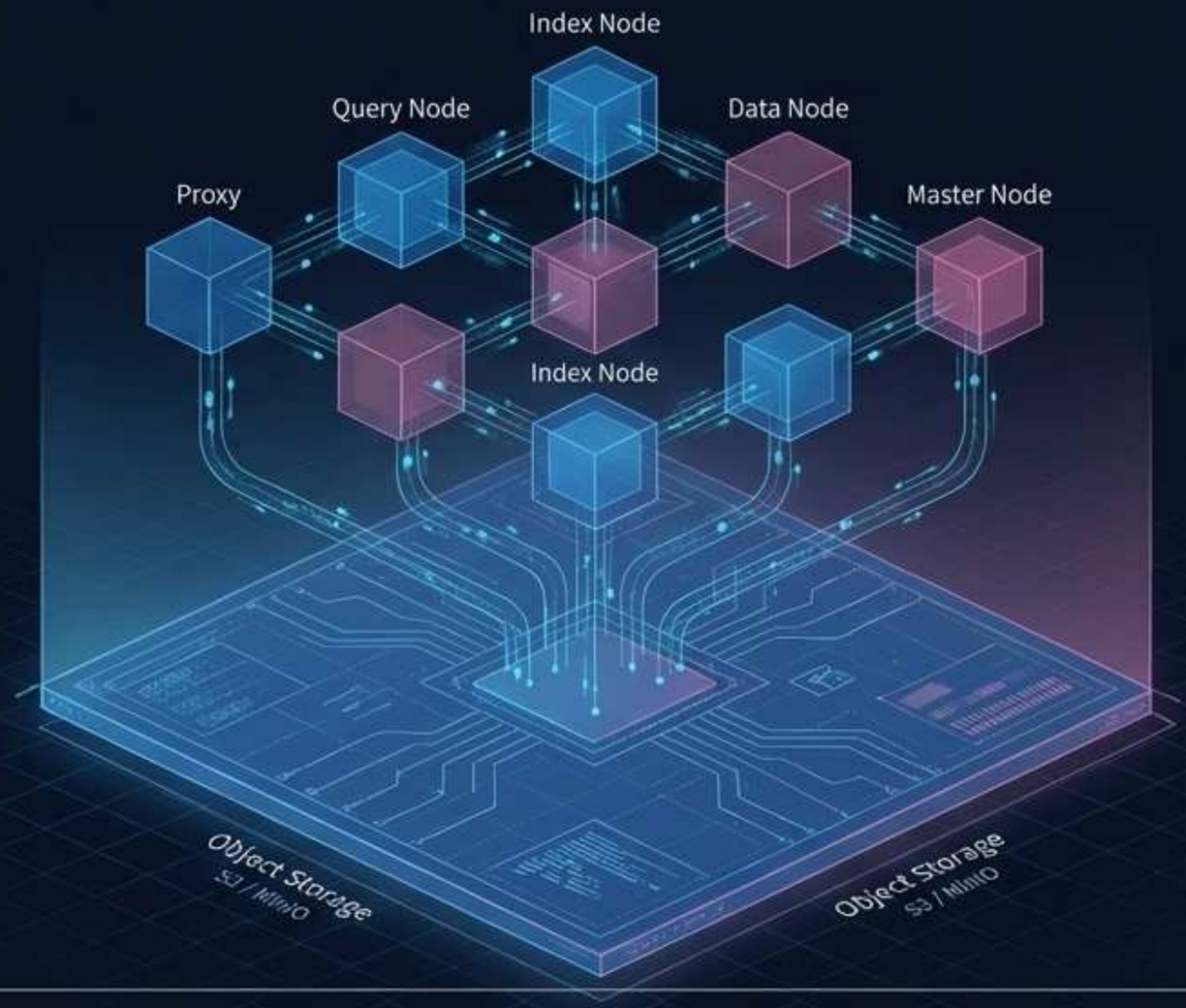
Quantization

DiskBBQ

性能专家：Milvus - 为大规模高并发而生

Milvus是专为大规模向量检索设计的云原生数据库。其核心架构理念是“存算分离”和“微服务化”：

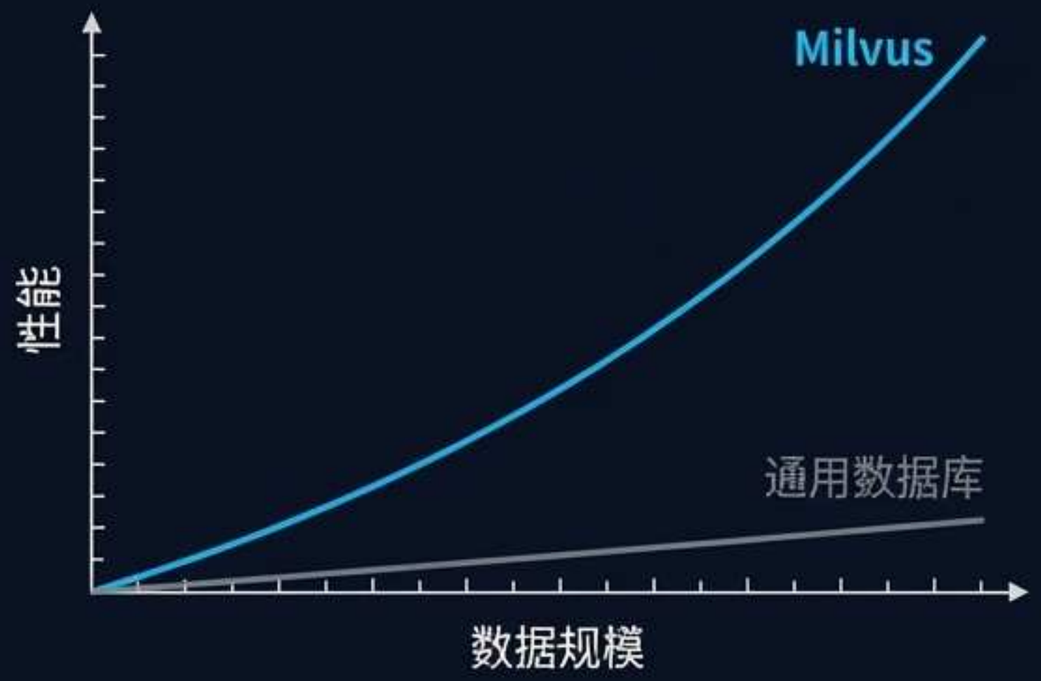
- 分布式微服务：将读、写、索引构建等核心功能解耦为独立的服务，可以按需独立扩展，弹性极佳。
- 存算分离：计算层和存储层分离，为处理千亿级向量数据和应对高并发查询场景提供了架构基础。



核心优势：极致性能与GPU加速

Milvus的定位非常明确：在极致性能和吞吐量（QPS）方面成为领导者。其性能优势主要体现在：

- 大规模数据集：专为亿级甚至千亿级向量数据集优化。
- 高吞吐量（QPS）：分布式架构轻松应对极高的并发查询请求。
- GPU加速：利用GPU并行计算能力实现毫秒级响应。



功能演进：多向量与混合查询潜力

Milvus不仅在性能上领先，其功能也在快速演进，以适应更复杂的RAG场景：

- **多向量搜索：**原生支持在单个对象上存储和查询多个向量，对多模态RAG至关重要。
- **混合查询潜力：**Milvus 2.4版本已规划引入更强大的混合查询能力，补齐在混合检索场景下的短板。

AI原生创新者：Weaviate

Weaviate的定位是“AI原生”的向量数据库，其设计哲学是最大化地简化RAG应用的开发流程。

- 数据对象与向量统一存储：同时存储原始数据对象和其对应的向量，使数据管理更直观。
- 内置Embedding集成：可直接与OpenAI等模型提供商集成，在数据导入时自动完成向量化，开发者无需自行管理Embedding模型。

核心优势：开箱即用的混合搜索

与Elasticsearch类似，Weaviate也原生支持强大的混合搜索能力，但实现方式更贴近AI开发者的使用习惯。

- **原生BM25F算法：**内置了优化的BM25F算法。
- **简洁的查询接口：**通过一个简单的API调用，同时指定关键词查询和向量查询，并由Weaviate自动完成结果的融合与排序。

这为快速构建高质量的RAG应用提供了“开箱即用”的解决方案。

```
Get {
  MyClass (
    hybrid: {
      query: "关键词",
      nearVector: {...}
    }
  ) {
    _additional {
      score
    }
  }
```

专业选型总结（一）：规模与性能取舍

如果在你的核心需求是：

- **超大规模：**数据集达到亿级甚至千亿级别。
- **极致QPS：**需要应对极高的并发查询请求。
- **高维数据集：**处理的向量维度非常高（如 > 1024）。
- **GPU加速：**希望利用GPU硬件来最大化检索性能。

架构师建议：**Milvus** 是更可靠的选择。

专业选型总结（二）：功能与统一集成

如果你的核心需求是：

- 企业级RAG：需要将向量检索与现有企业搜索、日志分析等系统整合。
- 强大的混合搜索：BM25+kNN+RRF的组合是关键需求。
- 复杂的元数据过滤：需要在检索时进行精细化的权限和属性过滤。
- 统一技术栈：倾向于使用一个成熟、稳定、统一的平台。

架构师建议：Elasticsearch 是理想选择。

专业选型总结（三）：开发体验与模块化

如果你的核心需求是：

- **快速原型与开发：**追求最简化的开发流程和最少的运维负担。
- **内置向量化：**希望数据库能自动处理数据的向量化过程。
- **开箱即用的混合搜索：**需要一个API友好、易于上手的混合搜索方案。
- **多租户隔离：**需要在单一实例中为不同用户或项目提供数据隔离。

架构师建议：Weaviate 更具优势。

高级检索策略：元数据过滤实现企业级合规

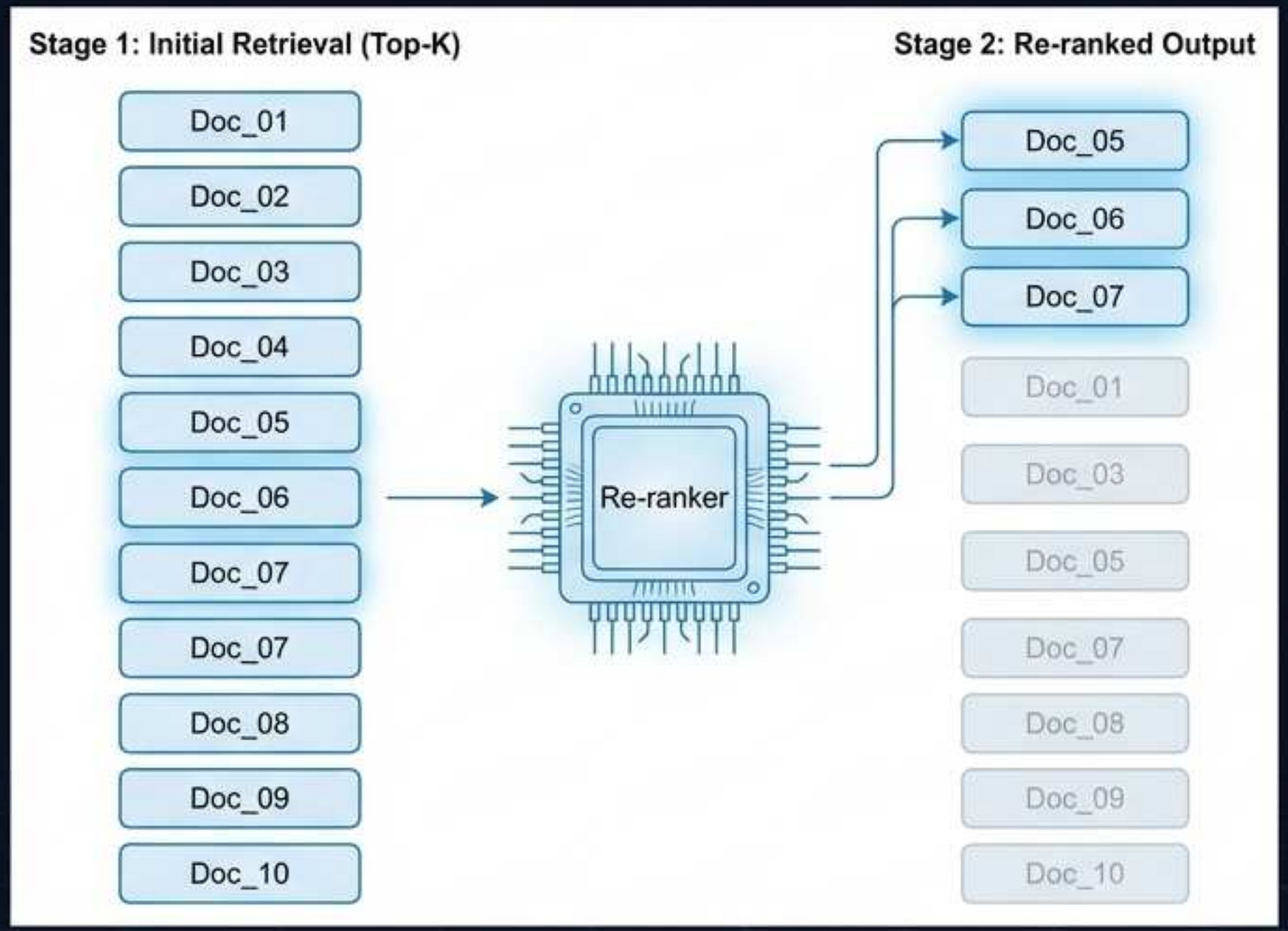
选择数据库只是第一步。高级RAG系统必须实施精细化的检索策略。

元数据过滤（Metadata Filtering）是实现企业级合规的生命线。

在执行向量搜索前后，根据文档的元数据（如访问权限、所属部门）进行严格过滤，确保只有用户被授权访问的数据源能能够进入LLM的上下文，从源头上根绝杜绝数据泄露风险。

高级检索策略：重排（Re-ranking）机制提高精度

初始检索返回的结果集（Top-K）虽然相关，但最优答案不一定排在第一位。LLM在处理长上下文时存在“中间迷失”问题。**重排（Re-ranking）机制**使用更精确的模型对Top-K结果进行二次排序，将最相关的文档片段推到上下文的最顶端，确保LLM能够“第一眼”就看到最关键的证据。



第四部分：突破边界：高级RAG算法与工程优化

结构性瓶颈：突破多跳推理的次元壁

传统RAG的局限

难以处理需要连接多个知识点的复杂问题，受限于单跳检索的线性模式。

架构突破：GraphRAG

引入知识图谱（KG）作为核心索引，显式建模实体间的深层关联，将隐性关系显性化。

核心价值：赋予系统进行多步“思考”和“推理”的能力，解答“A和B有什么关系”这类深层次问题。

GraphRAG实现路径：三元组与嵌入查询

Stage 1
知識图谱构建

通过大语言模型从非结构化文本中自动化提取“主-谓-宾”三元组，构建知識网络。

Stage 2

通过大语言文本中自动化擾取瞀的图谱，提升主寷的嵌入種知訳查詢的，提升式的特數字和形式。

Stage 3
查詢范式升級

超越传统关鍵词匹配，利用向量嵌入（Embeddings）进行图查詢，提升語义理解的靈活性與準確性。

- 框架支持：LlamaIndex等主流框架已內置知識图谱索引与查詢工具，简化了GraphRAG的工程落地。

高级算法：RAPTOR优化长文本分块

Chunking的权衡悖论：小块（Small Chunks）精度高但上下文缺失；大块（Large Chunks）上下文完整但噪但噪声过多，效率低下。

RAPTOR的分层策略：采用**聚类算法**将语义相似的文本块分组，并递归地生成更更高层次的摘要块，构建**分层知识树**。

检索优势：在树的不同层级上进行检索，能够同时兼顾细节精度与宏观上下文，有效破解“**迷失在中间**”的难题。

高级算法：Self RAG的Agent式推理

传统RAG的被动性：
无论检索质量如何，都必须进行生成，导致“垃圾进，垃圾出”。

Self RAG的主动决策：
引入“反思令牌”（Reflection Tokens），让LLM主动评估检索结果的相关性，并自主决策下一步行动。

智能体行为模式：

- 判断：是否需要检索？
- 评估：检索内容是否相关？
- 决策：直接生成答案、重新检索或迭代优化。

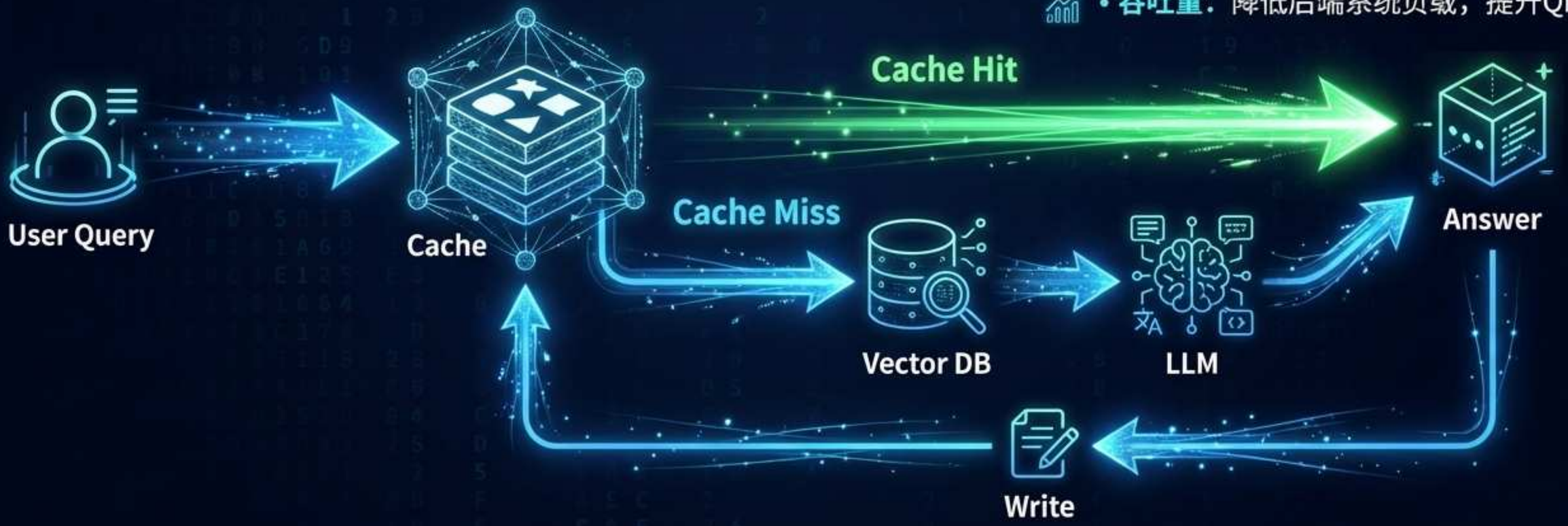
效率优化：分布式缓存机制

性能瓶颈：重复或相似的查询导致对检索系统和LLM的冗余调用，增加延迟与成本。

架构模式：引入高速分布式缓存（如Redis），存储高频查询的检索结果、中间生成内容、甚至最终答案。

核心收益：

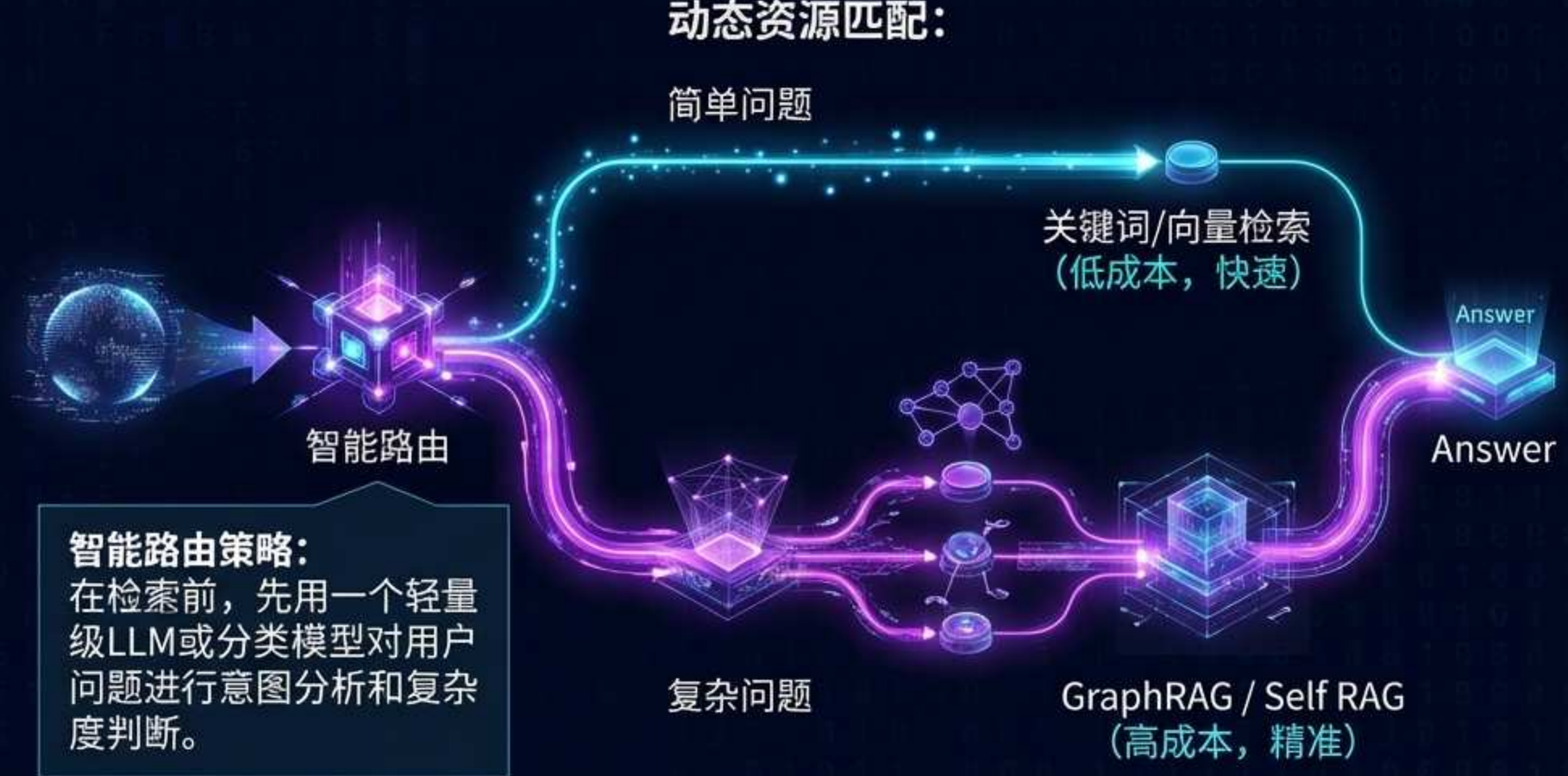
- **延迟：**毫秒级响应常见问题。
- **成本：**显著减少昂贵的LLM Token消耗。
- **吞吐量：**降低后端系统负载，提升QPS。



效率优化：自适应多级检索

资源错配：

用重量级的GraphRAG回答简单问题，或用简单的向量检索应对复杂问题，都造成资源浪费或效果不佳。



智能路由策略：
在检索前，先用一个轻量级LLM或分类模型对用户问题进行意图分析和复杂度判断。

多模态RAG（一）：挑战与知识表示

新边界：非文本数据：
基础RAG原生为文本设计，**无法理解和检索**图像、表格、图表和音视频中蕴含的知识。

架构要求：需要**专门的模型和流程**来来解析多模态数据，并将其嵌入到**统一的向量空间**中。

核心挑战：如何将**异构、多模态的数据源**转化为**统一、可被LLM理解和检索的知识表示**。

多模态RAG（二）：VLM解析与多模态VDB

视觉语言模型（VLM）

利用VLM（如GPT-4V）或专用模型（如RAGFlow DeepDoc）对图像和复杂PDF进行深度解析，识别布局、提取表格、生成图像描述。

多模态嵌入

将文本描述、图像块、表格数据等通过统一的多模态嵌入模型，映射到同一个高维向量空间。

统一检索

在支持多模态向量的数据库中，实现用文本查询图像、用图像查询相关文本的跨模态检索能力。

数据摄取：知识库管道的工程化

管道化架构（Pipeline）：将数据摄取流程（同步、清洗、分块、嵌入）抽象为可编排、可配置、可重用的模块化管道。

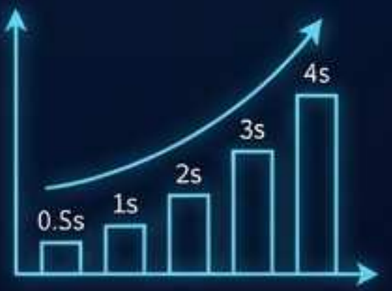
从脚本到平台：手动的、一次性的数据处理脚本无法支撑企业级知识库的持续更新、版本控制和质量监控。

平台级能力：Dify的知识库管道、RAGFlow的数据管线等工具，实现了从数据源到Agent消费的端到端、自动化的知识管理。

工程优化：模型重试中间件

生产环境的脆弱性：网络抖动、API速率限制、模型服务临时不可用等问题，导致LLM调用在生产环境中频繁失败。

韧性设计模式：引入专门的中间件（Middleware），在应用层透明地处理API调用失败。



核心策略：**指数退避**（Exponential Backoff）：在发生瞬时错误时，以递增的时间间隔自动重试，极大提高系统调用的成功率和健壮性。

工程优化：上下文摘要中间件

智能压缩策略： 引入上下文摘要中间件，在历史对话长度超过阈值时，自动调用LLM对最旧的对话部分进行摘要和压缩。

上下文窗口的物理极限： 随着对话轮次增加，上下文窗口（Context Window）会迅速饱和，导致Token成本激增和性能下降。

对话轮次 1

摘要
(轮次1-5)

对话轮次 6

对话轮次 7

...

对话轮次 N

效果： 在保留核心对话记忆的同时，动态管理上下文窗口大小，实现成本与记忆深度的最佳平衡。

案例研究：Dify的Workflow架构升级

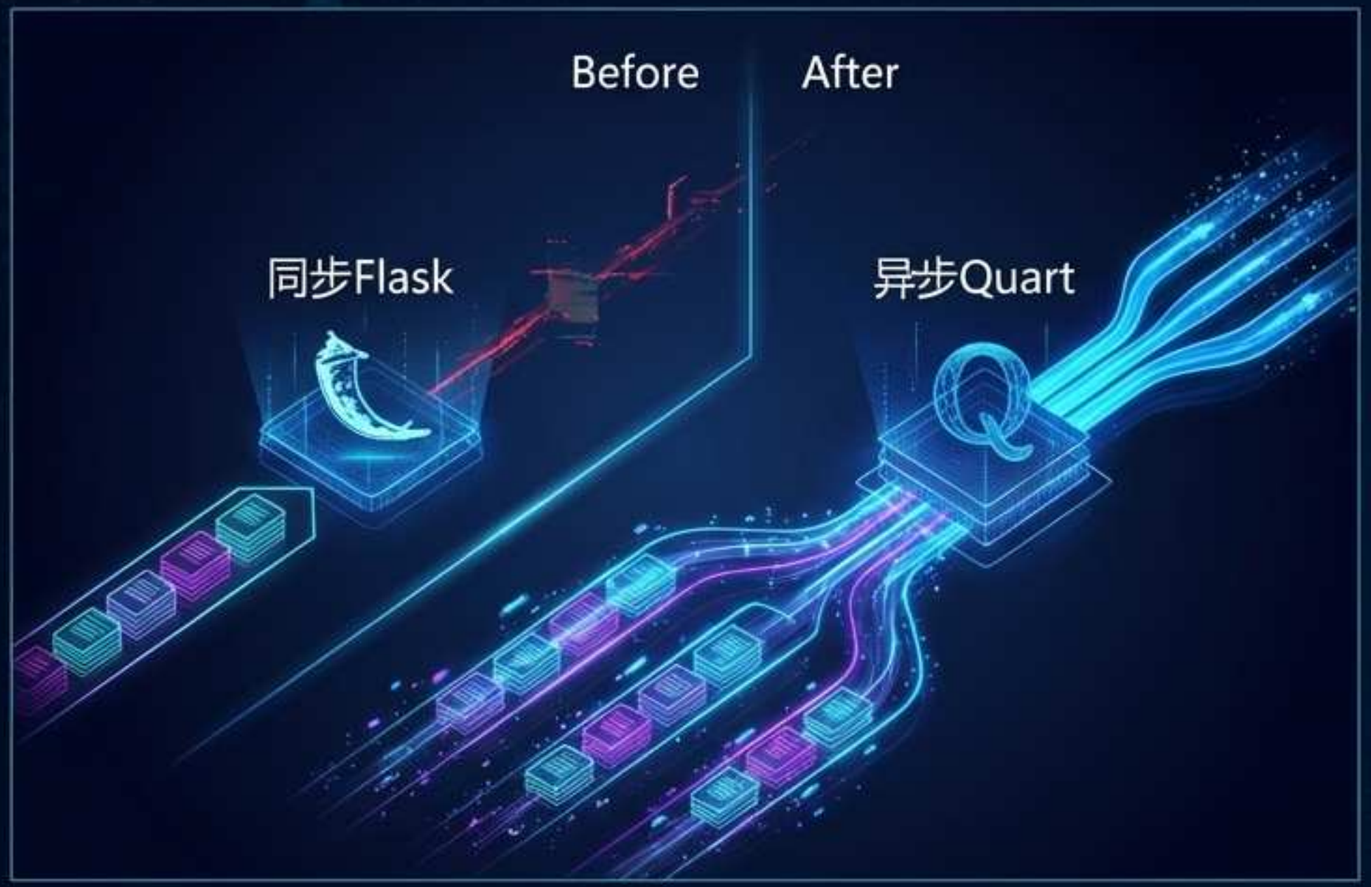
核心演进：

Dify v1.9.0将知识库摄取流程从一个固定的黑盒，升级为**可视化、模块化的知识库管道（Knowledge Pipeline）**。

架构优势：

- **可视化编排**：用户可通过拖拽方式自定义数据处理流程。
- **高可扩展性**：允许开发者添加自定义的数据处理节点。
- **端到端整合**：将数据工程与应用开发无缝集成在同一平台。

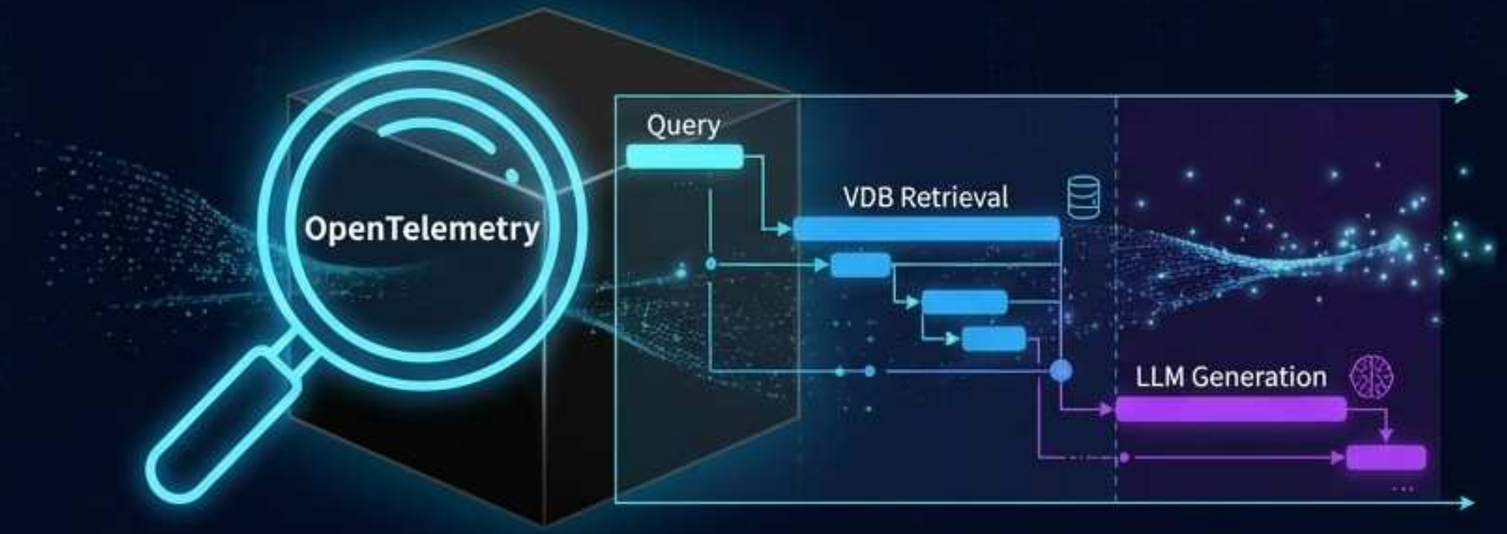
案例研究：RAGFlow的性能优化实践



- 性能目标：支持企业级高并发和高吞吐量的文档处理与问答场景。
- Infinity文档引擎：专为RAG设计的底层文档解析与管理引擎，优化IO和数据处理效率。
- 架构迁移：后端Web框架从同步的Flask迁移到异步的Quart，大幅提升了并发处理能力和系统响应速度。

RAG的可观测性缺失：全链路追踪

黑盒困境：RAG系统的非确定性输出使其难以调试。当答案错误时，问题可能出在检索、增强或生成等任何环节。



新监控范式：传统监控（CPU、内存）不足，必须引入类似微服务治理的全链路追踪（Distributed Tracing）。

追踪目标：借助OpenTelemetry等标准，对LLM调用、检索步骤、上下文片段、工具执行等每一个环节进行结构化日志记录和审计。

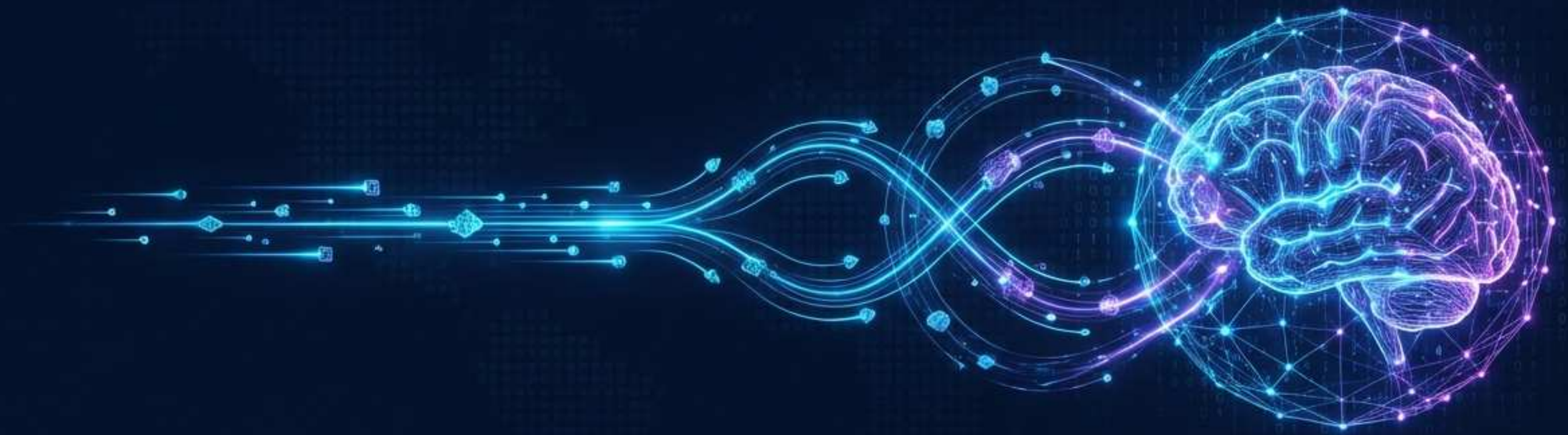
可观测性实现：托管服务与结构化日志

解决方案：利用AWS CloudWatch GenAI Observability等云原生托管服务，或集成LangSmith、Arize等第三方平台。

核心功能：将Agent执行循环中的每一个Span（模型调用、工具执行、上下文检索）自动捕获，并转化为可搜索、可回放的结构化日志。

架构价值：
实现对RAG系统行为的深度调试、性能分析、成本归因和合规审计，是企业级AI应用最后的、也是最关键的一环。

智能涌现：从RAG到行动的范式转变



RAG：信息检索的终点

Agent：智能行动的起点

推理（Reasoning）

规划（Planning）

行动（Action）

Agentic AI系统架构：涌现的四大支柱

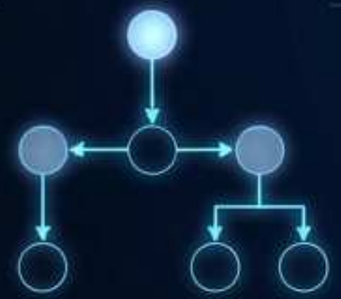
1. 推理引擎（Reasoning Engine）

以LLM为核心，负责决策、规划与反思。

2. 记忆系统（Memory System）

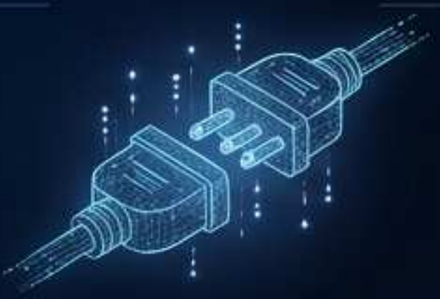
整合短期与长期记忆，实现情境感知与知识积累。

Agent Core



3. 编排模块（Orchestration Module）

控制任务执行流程，动态调用工具与知识。

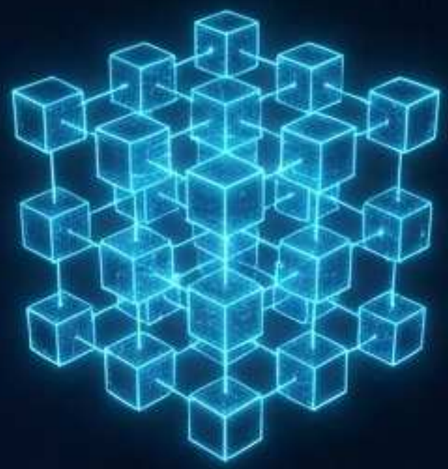


4. 工具接口（Tool Interface）

连接外部世界，执行代码、调用API、访问服务。

Agent的认知核心：分层记忆系统

信息交换与固化



短期记忆（STM - Short-Term Memory）

核心功能：
- 会话上下文（Session Context）
- 即时指令与用户输入

技术实现：上下文窗口（Context Window）

特性：易失性（Volatile）、高速访问、容量有限

长期记忆（LTM - Long-Term Memory）

核心功能：
- 向量化知识（Vectorized Knowledge）
- 历史经验与事实摘要

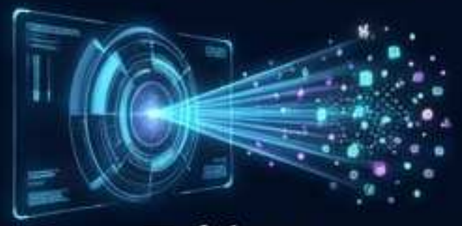
技术实现：向量数据库（Vector Database）/ RAG

特性：持久化（Persistent）、语义检索、可扩展

RAG：Agent长期记忆的实现引擎

原始信息

历史对话、外部文档、经验数据



异步加工

LLM进行信息抽取、摘要、三元组提炼

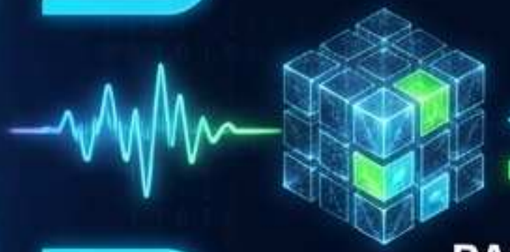
向量化存储

将语义事实存入向量数据库

RAG驱动

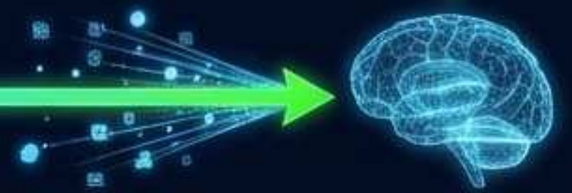
Agent查询

推理引擎产生检索需求



RAG检索

向量语义搜索，召回相关知识



情境增强

将检索到的知识注入提示词，供Agent决策

突破集成鸿沟：RAG作为Workflow的知识工具

工程实践：Dify工作流的逻辑控制与状态增强

- **超越线性链条：**引入循环、条件判断，实现复杂的任务逻辑。
- **增强LLM状态：**通过变量传递，让LLM在多步任务中保持记忆和上下文一致性。
- **实现复杂Agent行为：**如多轮澄清式问答、自动化报告生成等。

架构演进：RAGFlow Agent的规划与反思

RAGFlow v0.20.0 统一Agent与Workflow，引入Multi-Agent、规划与反思机制

连接世界：工具接口与统一网关

- **工具网关的核心职责：**
  - 标准化接入（Standardization）：将不同工具封装成统一接口。
  - 动态搜索（Discovery）：帮助Agent快速找到所需工具。
  - 集中治理（Governance）：统一管理认证、权限、监控和日志。

安全基石：Agent的沙盒环境与硬件级隔离

T11 RCE 威胁（远程代码执行）

- Agent执行不可信代码
- 访问恶意网页
- 操作敏感系统

解决方案：MicroVM 沙盒

- 核心技术：Firecracker / gVisor
- 实现机制：
  - 为每个任务或会话启动一个独立的微型虚拟机。
  - 提供硬件级资源隔离（CPU，内存，网络）。
  - 用完即焚（Ephemeral），环境干净，无状态残留。
- **优势：极致安全、秒级启动、高密度部署**

未来展望：RAG与Agent的终身学习

1．情境感知
（Context-Aware）

Agent利用RAG理解当前任务和历史对话。

2．跨会话记忆
（Cross-Session Memory）

Agent的记忆（LTM）在不同任务和会话中得以保留和积累。

3．终身学习
（Lifelong Learning）

Agent通过与世界持续互动，不断更新和优化其长期记忆库，实现能力的自我演进和智能的真正涌现。

向量化存储是Agent实现终身学习的记忆基石。

Mem0，Letta，LangMem