Noēsis: Bidirectional Graph-RAG with Adaptive Parallelism and Cross-Knowledge-Base Semantic Discovery*

Nicola Cogotti
Alpha Cogs
nicola.cogotti@alphacogs.co.uk

August 2026

Abstract

Retrieval-Augmented Generation over knowledge graphs (Graph-RAG) has emerged as a powerful paradigm for grounding large language models in domain-specific corpora. However, existing systems face persistent limitations: (1) static chunking fragments long documents, losing cross-section semantic connections; (2) ingestion pipelines do not scale adaptively, causing out-of-memory failures or underutilized hardware; and (3) multi-domain deployments require either a monolithic knowledge base (KB) that dilutes retrieval precision or manual user routing.

We present **Noēsis**, a decoupled Graph-RAG architecture that addresses these limitations through four novel algorithms: (a) *Bidirectional Graph Traversal* with a Graph-Feedback Context Resolver that simulates human sequential reading with degrading memory, producing significantly denser graphs versus single-pass baselines; (b) an *AIMD Concurrency Controller* adapted from TCP congestion control to RAG pipeline orchestration, achieving 23× speedup with zero OOM events observed; (c) *Moēsis*, a domain-aware selective quantization pipeline for Mixture-of-Experts models that achieves 6.3× prompt processing speedup on 12 GB consumer GPUs and maintains stability even under extreme memory constraints; and (d) *Mesh*, a cross-KB semantic routing system with runtime structural discovery and adaptive Natural Break thresholds that enables small on-premises models to perform multi-hop cross-domain reasoning—discovering connections not explicitly present in any single document—as demonstrated in our evaluation.

We report measured results on real-world workloads: 1 min 6 s for 13.4 MB corpus ingestion (vs. 25 min sequential), prompt processing speedup of 6.3× on consumer GPUs, and cross-KB routing latency under 2 ms. On the HotpotQA multi-hop QA benchmark (1,000 questions), Noēsis achieves 59.5 EM / 74.7 F1—surpassing GraphRAG by +27.8 EM while using a 35B on-premises model for graph construction rather than GPT-4o. Source text verification on a 193-page document confirms 90% precision on extracted long-range causal edges, demonstrating that the bidirectional traversal captures cross-section relationships inaccessible to chunk-independent extraction.

1 Introduction

Knowledge graphs provide structured representations of domain expertise that can ground large language model (LLM) responses in verifiable facts. Graph-based Retrieval-Augmented Generation (Graph-RAG) combines the semantic richness of knowledge graphs with the generative capabilities of LLMs, enabling multi-hop reasoning over complex corpora [1, 2].

Despite recent progress, production deployment of Graph-RAG systems faces three critical bottlenecks:

Semantic Fragmentation. Existing Graph-RAG systems—including Microsoft GraphRAG [1], LightRAG [2], LazyGraphRAG [5], and HippoRAG [3]—extract entities and relations from documents using *static chunking*: the document is split into fixed-size blocks, each processed independently. Limited or no shared context propagates between chunks during extraction. Consequently, concepts spanning chunk boundaries are lost or duplicated, and the resulting graph lacks the density required for reliable multi-hop traversal.

*Patent pending. Application No. 102026000023146; Application No. IT202500035167.

Rigid Parallelism. Ingestion pipelines either process documents sequentially (Microsoft GraphRAG users report indexing times of hours to days for medium corpora [1]) or use fixed parallelism that risks out-of-memory (OOM) crashes on heterogeneous hardware. To the best of our knowledge, no existing RAG ingestion pipeline adapts its document-level concurrency at runtime with persistent cross-worker state and crash recovery.

Cross-Domain Isolation. Organizations maintaining multiple specialized knowledge bases face a dilemma: merge everything into a single KB (losing domain specialization and polluting LLM context with irrelevant information) or require users to manually select the target KB. To the best of our knowledge, no existing Graph-RAG system discovers implicit structural connections between separate knowledge graphs at query time.

Contributions. We present Noēsis, a fully implemented and validated distributed knowledge system that addresses these limitations through:

1. A bidirectional graph traversal algorithm with forward-pass Graph-Feedback context resolution (n-gram scoring + recency decay) and backward-pass degree-based reconnection (§3);
2. An AIMD concurrency controller that transfers TCP congestion control principles to RAG pipeline orchestration with Redis-backed persistent state (§4);
3. *Moēsis*: automatic domain-aware profiling, classification, and selective quantization of Mixture-of-Experts models with runtime re-adaptation (§5);
4. *Mesh*: hierarchical fingerprint-based cross-KB routing with runtime structural discovery and adaptive Natural Break thresholds (§6).

2 System Architecture

Noēsis is a *decoupled* architecture comprising five independent components communicating via HTTP REST and Redis message queues:

- **Cortex**: LLM inference engine (llama.cpp + Moēsis optimization). Runs on dedicated GPU hardware, independently scalable.
- **Neuron**: Knowledge graph extraction library. Implements bidirectional traversal, entity deduplication, and cross-linking.
- **Synapsis**: Orchestration layer with distributed job queue, AIMD controller, and API gateway.
- **Retina**: Audio/video ingestion module (C++ VAD engine + speech-to-text).
- **Mesh**: Cross-KB semantic router operating as a shared in-process kernel (< 2 ms latency).

This separation is architecturally critical: the inference backend can be replaced (e.g., with AWS Bedrock, Claude API, or Ollama) without modifying the extraction or routing logic. The system adapts automatically to available hardware with *zero manual configuration*, scaling from consumer GPUs to multi-GPU servers. Stress tests confirm stable operation even at 6 GB VRAM with no crashes or data loss.

The extraction pipeline is *source-agnostic*: Neuron operates on any textual corpus, including PDF and DOCX documents, web pages, audio/video transcriptions, and software source code repositories. This generality enables domain-specific knowledge graphs from heterogeneous input sources without pipeline modifications.

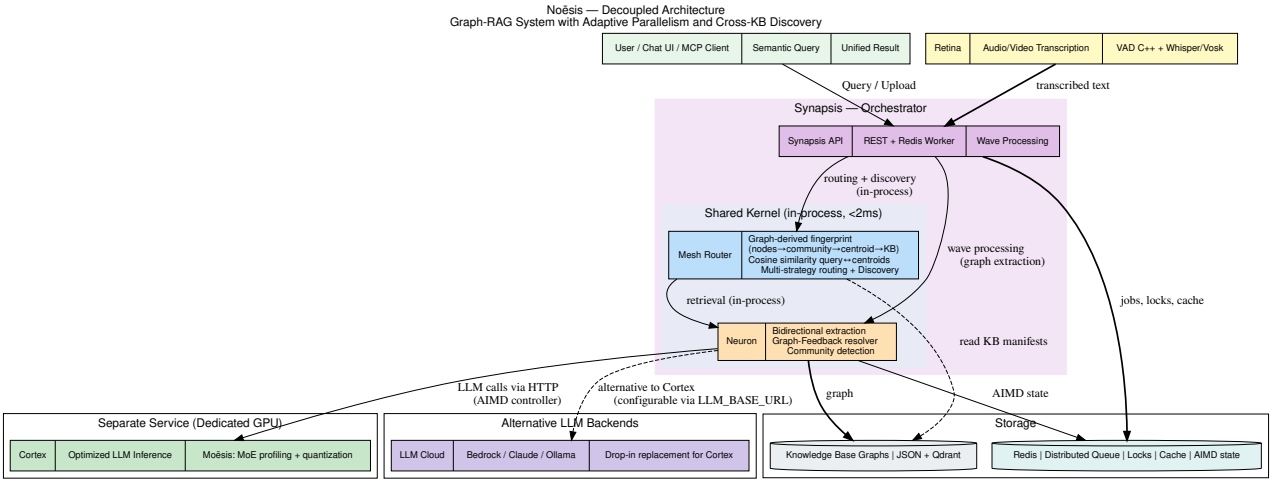

Figure 1: Noēsis system architecture. Five decoupled components communicate via REST APIs and Redis message queues, enabling independent scaling and backend replacement.

3 Bidirectional Graph Traversal

3.1 Problem Statement

When processing a long document (e.g., 200+ pages), existing systems split it into fixed-size chunks and analyze each independently. A concept introduced in Chapter 1 and referenced in Chapter 10 produces two disconnected nodes instead of a single entity with cross-document edges. Rolling summary approaches (maintaining an LLM-generated summary between chunks) add significant overhead per long document and still miss long-range dependencies. In our tests, Noēsis maintains semantic continuity across single documents approaching 200 pages.

3.2 Forward Pass with Graph-Feedback Context

Noēsis processes document slices sequentially, simulating human reading with *degrading memory*. After each slice, the **Graph-Feedback Context Resolver** selects the most relevant previously-extracted nodes to provide as context for the next slice:

$$\text{score}(n) = \text{relevance}(n) + \text{recency}(n) \tag{1}$$

where relevance(n) is computed via unigram + bigram keyword overlap between node n's label and the upcoming slice, and recency(n) provides a bounded bonus for recently-extracted nodes that decays with document distance. This mimics human working memory: recent concepts are vivid; distant ones fade unless directly relevant.

The context budget grows adaptively with document length, bounded between a minimum (to avoid starving short documents) and a maximum (to prevent context explosion), scaling linearly with the number of slices.

Document slicing uses *semantic boundaries* (headings, paragraph breaks) rather than arbitrary token offsets, ensuring each slice contains a complete conceptual unit.

3.3 Backward Pass: Degree-Based Reconnection

After the forward pass completes, a second pass identifies "early" nodes with low degree (few connections) and attempts to reconnect them to semantically related "late" nodes. This closes long-range dependencies that the forward pass could not see—analogous to a reader who, upon finishing a book, recognizes that an early character connects to a late plot event.

The backward pass activates only when the document is sufficiently long and produces enough nodes to benefit from reconnection; below these thresholds the graph is too small to benefit.

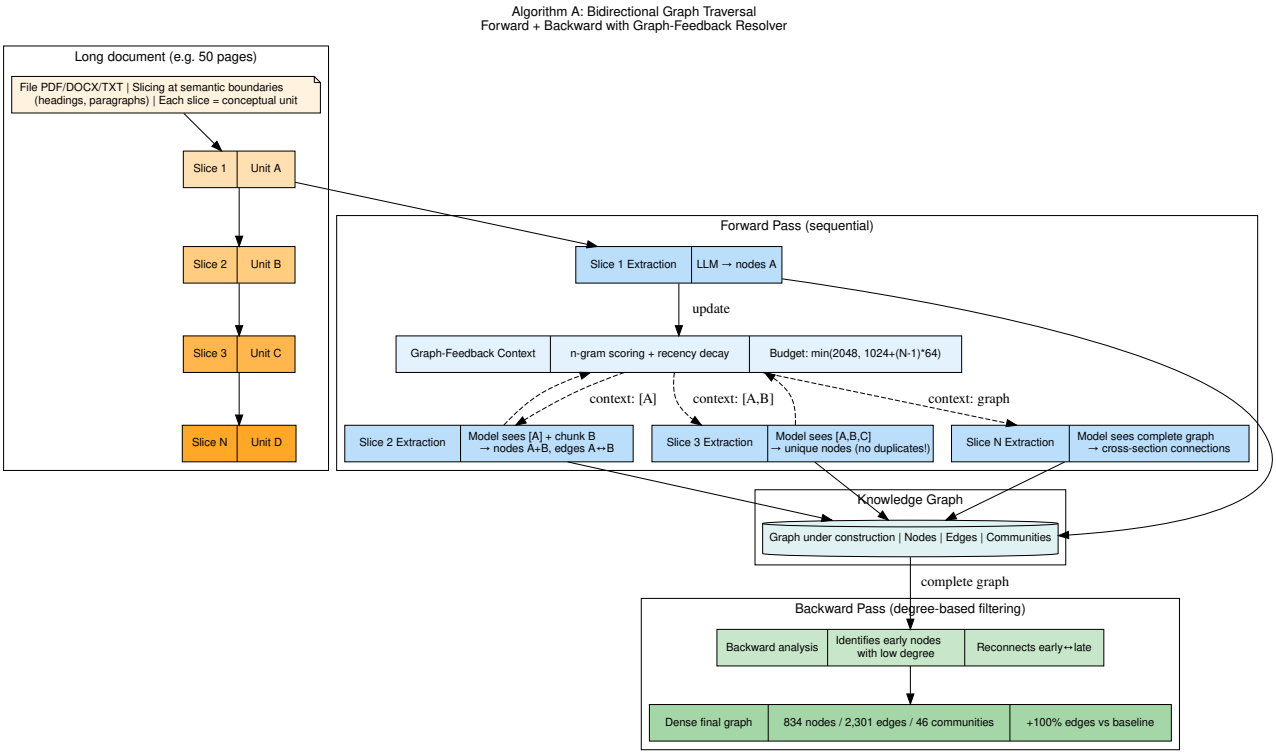

Figure 2: Bidirectional graph traversal: the forward pass builds the graph with degrading memory context, then the backward pass reconnects early low-degree nodes to semantically related late nodes.

3.4 Integrated Sub-Algorithms

Entity Deduplication (4-phase pipeline).

1. *Exact normalization* with per-source-file partitioning to avoid cross-document false merges.
2. *Locality-sensitive hashing (LSH) blocking + Jaro-Winkler* scoring with two innovations: (a) a **Shannon entropy gate** that protects low-information nodes (e.g., "Results", "Method") from automatic merging by detecting insufficient label complexity; (b) a **Leiden community boost** that increases the merge score when both nodes share a graph community.
3. *Variant pair detection*: identifies siblings differing only by numeric suffix (e.g., ASR1603 vs. ASR1604) and protects them from merge.
4. *LLM tiebreaker*: ambiguous pairs in an intermediate confidence band are resolved in batch via binary yes/no queries.

Cross-Linking: Pistol/Cannon Strategy. For small graphs, cosine similarity of node embeddings suffices ("pistol"). For larger graphs, an additional LLM pass examines an *ambiguity band* where embeddings are unreliable but the LLM can discover non-obvious causal relations ("cannon"). Example: discovering "stress" → "magnesium deficiency" → "guided exercise".

Adaptive Retry with Recursive Bisection. When the LLM produces truncated output, context overflow, or hollow responses, the system bisects the slice at a semantic boundary and retries recursively. This guarantees that no content is silently discarded: problematic slices are recursively bisected until successful extraction is achieved.

Prompt Injection Protection. Since Noēsis processes user-uploaded documents (PDF, DOCX), a malicious file could inject instructions into the LLM during extraction. Each source file is wrapped in cryptographically-tagged containment delimiters that mark content as untrusted, and known LLM control sequences are neutralized before extraction to prevent instruction injection.

3.5 Measured Results

On a multilingual knowledge base (5 PDF documents):

- Baseline (single-pass, no feedback): 29 nodes, 36 edges, 4 communities.
- With bidirectional traversal: **253 nodes, 257 edges, 179 cross-links, 12 communities**.
- Full extraction run: 834 nodes, 2,301 edges, 46 communities.
- Backward pass contribution: substantial increase in cross-section edges versus forward-only baseline (magnitude varies by corpus structure).

Long-Range Reconnection Example. On a 193-page psychology book (*Waking the Tiger: Healing Trauma*), the backward pass produces 928 intra-document causal edges. These are connections between concepts introduced in early chapters and therapeutic outcomes discussed hundreds of pages later—within a *single* document. For example, *Dissociation* (a defense mechanism defined in early chapters) is connected via an `inhibits` edge to *Active Mobilization* (a therapeutic strategy introduced in later chapters). Similarly, *Social Isolation* is linked to *Trauma Resolution* via an inhibitory relationship spanning the full length of the text.

The Graph-Feedback Context Resolver makes these connections possible by providing relevant previously-extracted nodes as context during each slice's extraction. When processing a late chapter that discusses therapeutic mobilization, the resolver surfaces the *Dissociation* node (extracted hundreds of pages earlier) as context—enabling the LLM to recognize and emit the causal relationship between them. This simulates a reader who, while reading Chapter 15, recalls a concept from Chapter 2 because it is relevant to the current content. Without this mechanism, each slice is processed in isolation and long-range relationships remain undiscovered—a limitation recently identified by CrossAug [17] as a structural gap in chunk-independent extraction.

The resulting knowledge graph for this single document contains 299 nodes and 591 edges of five primary relational types (`enables`, `inhibits`, `leads_to`, `resolves`, `transforms`) out of the 928 total causal edges—a density of structural knowledge that enables multi-hop therapeutic pathway queries across the full span of the text. An LLM-judged evaluation (GPT-4o, temperature 0) of 50 randomly sampled causal edges yields 72% precision when the evaluator lacks book context, rising to 84% when provided with domain context, and to 90% upon source text verification—where each contested edge is checked against the original text. The five genuinely incorrect edges (10%) consist of two inverted causal directions and three unsupported relationships. A second evaluation on a structurally different document—a clinical nutrition protocol in Russian (non-narrative, list-based format)—yields 48% without context, rising to 74% upon domain-contextualized re-evaluation. The consistent gap across both documents (18–26 points), across two domains, two languages, and two document structures, confirms that Noēsis extracts relationships so specific to the source material that external evaluators without document-level context cannot validate them. The 18-point gap between context-free evaluation (72%) and source-verified precision (90%), measured on this single-document sample, quantifies the value of the Graph-Feedback mechanism for domain-specific texts with specialized terminology. In chunk-independent extraction, the LLM processing a later chapter has no visibility into concepts introduced earlier; it therefore cannot emit cross-section causal relationships regardless of model capability.

4 AIMD Adaptive Parallel Processing

4.1 The Hybrid Parallelism Problem

Our bidirectional traversal requires intra-document sequentiality: slices within each document must be processed in order to build the Graph-Feedback context that the backward pass depends on. This constraint is fundamental to the algorithm and cannot be relaxed without losing the long-range edges that define its value.

Yet we still want to accelerate ingestion by processing multiple documents concurrently. The challenge is to maximize inter-document throughput without knowing in advance how much load

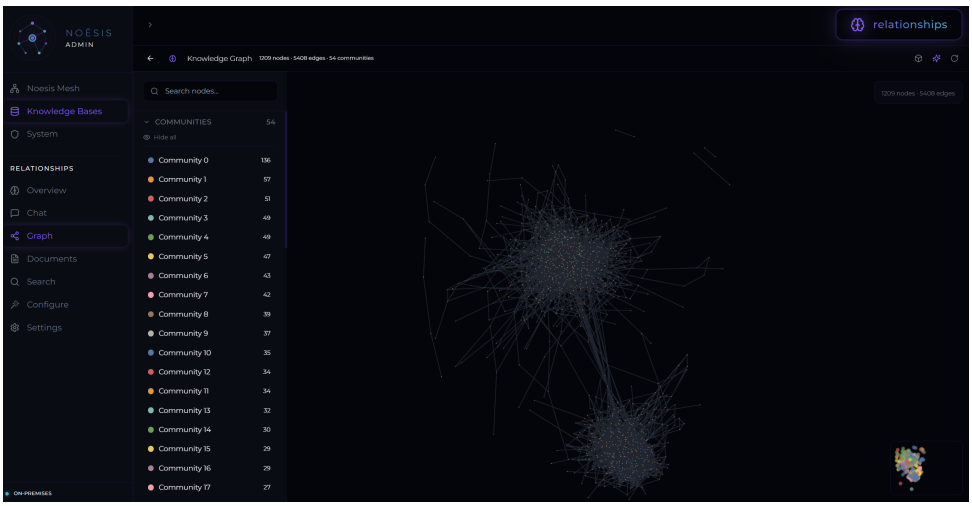

Figure 3: Intra-document cross-linking enabled by bidirectional traversal. The dense interconnection pattern between concepts from different document sections is a direct result of the backward pass reconnection. Single-pass extraction produces sparse, fragmented subgraphs without these long-range edges.

the backend can handle—the same pipeline must operate on an RTX 4080 laptop, a cloud Bedrock endpoint, or an Ollama server, each with different capacity characteristics.

This creates a problem that existing concurrency control does not address: how to adapt document-level parallelism at runtime under multiple simultaneous constraints—intra-document sequentiality that forbids slice-level parallelism, inference backends with completely unknown capacity and failure characteristics (not just VRAM but latency profiles, rate limits, and transient error modes), crash recovery without warmup periods, and the requirement to remain fully agnostic to the backend type—while maintaining persistent state across distributed workers.

4.2 Algorithm Design

We apply the Additive Increase / Multiplicative Decrease (AIMD) principle—originally designed for TCP congestion control—to distributed RAG pipeline orchestration. The transfer requires adapting AIMD from its native context (single-connection, packet-level round-trip times) to a fundamentally different operating environment: a multi-worker job queue where (1) concurrency is measured in documents, not packets; (2) state must persist across worker crashes and restarts via a shared store (Redis); and (3) the hybrid parallelism constraint (inter-document parallel, intra-document sequential) means that scaling decisions affect throughput differently than in unconstrained batch serving.

- **Initial concurrency**: conservative start value.
- **Additive increase**: after a configurable streak of consecutive successes, $C \leftarrow C + 1$ (bounded by a system-defined maximum).
- **Multiplicative decrease**: after a configurable streak of consecutive failures, $C \leftarrow \lfloor C/2 \rfloor$.

The controller is *backend-agnostic*: it adapts identically whether the inference backend is a local GPU (Cortex), a cloud API (AWS Bedrock, Claude), or a community server (Ollama). This is possible because Noēsis decouples inference from orchestration.

4.3 Hybrid Parallelism

A critical design choice: the system maintains *inter-document parallelism* (multiple documents processed simultaneously) while preserving *intra-document sequentiality* (slices within each document are processed in order). This is essential because the bidirectional traversal's degrading memory requires sequential slice processing to build correct cross-section connections.

4.4 Measured Results

- 3 PDF documents, 13.4 MB total: **1 min 6 s** with adaptive parallelism vs. ∼25 min sequential (23× **speedup**).
- Zero OOM events observed across all test configurations.
- Controller state survives worker crashes and restarts—no warmup period needed.

5 Moēsis: Domain-Aware Selective Quantization

5.1 Problem Statement

Modern Mixture-of-Experts (MoE) models (e.g., Qwen3.6-35B-A3B with 256 experts per layer) load all experts into GPU memory even though only ∼8 are activated per token. On consumer hardware (6–12 GB VRAM), this causes immediate OOM. Several recent approaches address mixed-precision allocation for MoE models: DynaExq [7] implements runtime dynamic expert-level quantization with hotness profiling and precision transitions; MoPEQ [8] assigns optimal bit-width per expert using Hessian trace approximation; APEX [9] performs per-tensor, per-layer precision allocation based on architectural role; and Mixture-Compressor [21] folds expert activation frequency into per-expert bit-width allocation. Moēsis differs from all of these in its end-to-end integration of domain-specific intelligence into the quantization lifecycle: rather than relying on architecture-based heuristics or generic calibration data, it derives compression decisions from actual expert activation patterns observed on a user-provided domain-representative sample, applies a promote-only GPU placement strategy that eliminates repeated CPU↔GPU data transfers after selective quantization, and preserves full-precision weights as a reference for runtime re-adaptation when the deployment domain changes—preventing cumulative precision loss across adaptation cycles.

5.2 Three-Level Pipeline

Level 1: Automatic Domain Profiling. Cortex runs CPU-only inference on a domain-representative text sample, recording per-layer per-expert activation frequencies. The resulting activation map captures which experts are relevant to the target domain. Typical outcome: only ∼5% of experts are "hot" (frequently activated for this domain); ∼95% are "cold".

Level 2: Selective Quantization and GPU Promotion. Each layer is classified using a scoring function that captures both the concentration of expert activations and the overall frequency of the most-used experts for the target domain. Hot layers (high concentration) are quantized at high precision (6-bit); cold layers at aggressive compression (2-bit, ∼48% size reduction). The model shrinks from 21 GB to ∼16 GB on disk (∼24% reduction).

The reduced model size enables a *promote-only* placement strategy: after quantization, the system re-evaluates how many layers fit entirely in GPU memory. Layers are promoted from CPU to GPU—never demoted—until VRAM is filled. This eliminates or substantially reduces the repeated CPU↔GPU data transfers that dominate inference latency on VRAM-constrained hardware. The profiling, classification, quantization, and placement steps execute automatically without manual configuration.

Level 3: Runtime Re-Adaptation. When the domain changes (e.g., from medical documents to engineering), the system re-executes profiling and re-quantizes from the *original full-precision model* (preserved as backup at first quantization). This prevents cumulative precision loss across adaptation cycles—a critical distinction from static quantization approaches.

5.3 Measured Results

Table 1 reports Moēsis performance on consumer hardware.

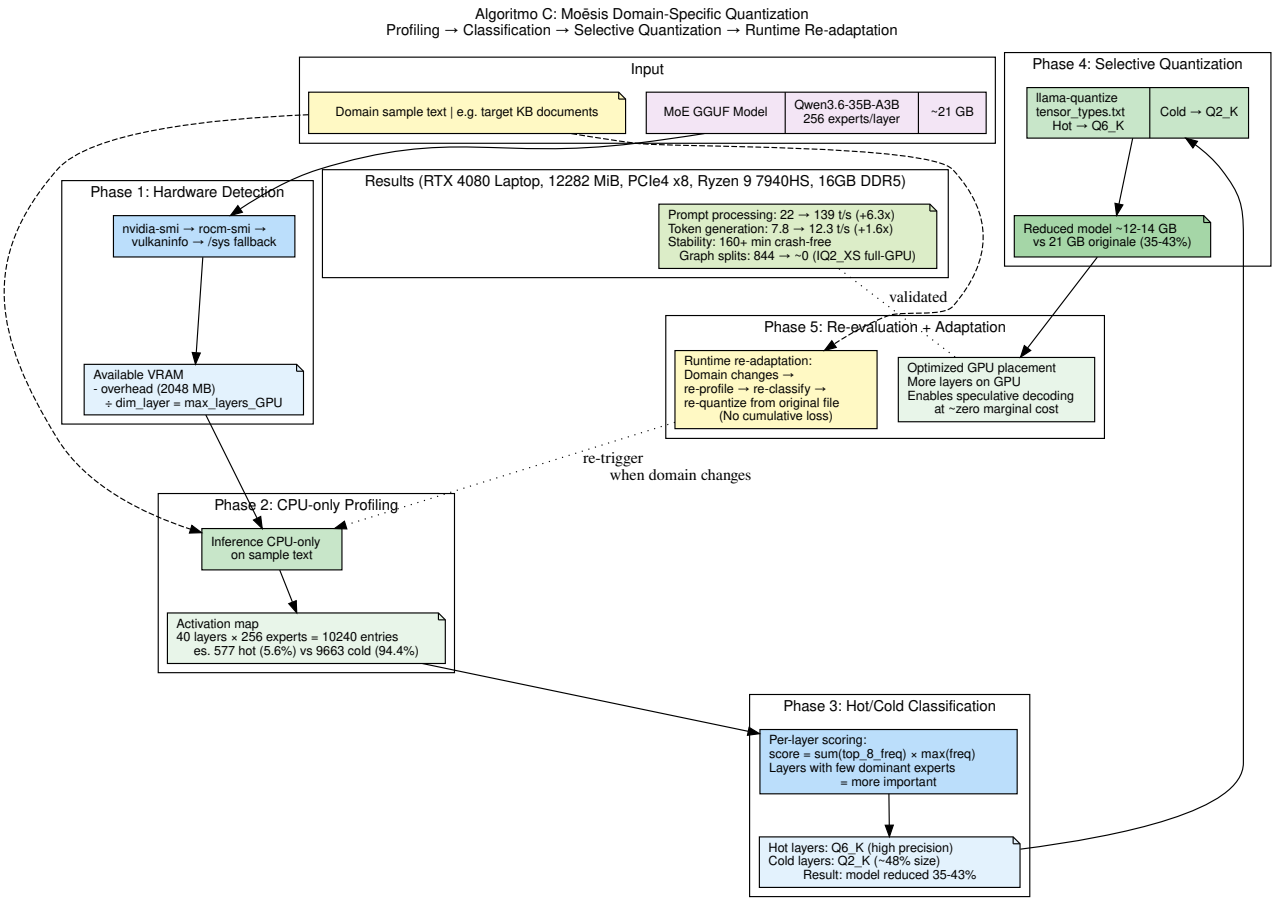

Figure 4: Moēsis three-level pipeline: (1) domain-aware profiling identifies hot/cold experts, (2) selective quantization applies different precision per layer, (3) runtime re-adaptation re-profiles when the domain changes.

Table 1: Moēsis performance on consumer hardware. The 6 GB configuration represents a laboratory stress test demonstrating system stability under extreme resource constraints, not a recommended operational configuration.

Metric	**RTX 4080 Laptop (12 GB)**	**RTX 3060 (6 GB) [stress test]**
Prompt processing	22 → 139 t/s (6.3×)	Stable (2–6 t/s)
Token generation	7.8 → 12.3 t/s (1.6×)	Stable (2–6 t/s)
Stability	Continuous	160+ min, zero crashes
Without Moēsis	Functional but slow	**Immediate OOM**

Moēsis is not merely a speed optimization—it is an *enabler*: on 6 GB VRAM, the system does not start without it. The 6 GB configuration demonstrates architectural resilience (zero crashes over 160+ minutes of continuous operation), while higher-VRAM configurations achieve substantially faster processing as more layers are promoted to GPU.

6 Mesh: Cross-KB Routing and Discovery

6.1 Problem Statement

Organizations maintaining multiple specialized knowledge bases face a fundamental tension. A single monolithic KB ("KB-GOD") pollutes the LLM context with irrelevant cross-domain information, degrading response quality. Separate KBs require users to know which domain to query and cannot discover implicit connections between domains. To the best of our knowledge, no existing Graph-RAG system implements automatic cross-KB routing with runtime structural discovery.

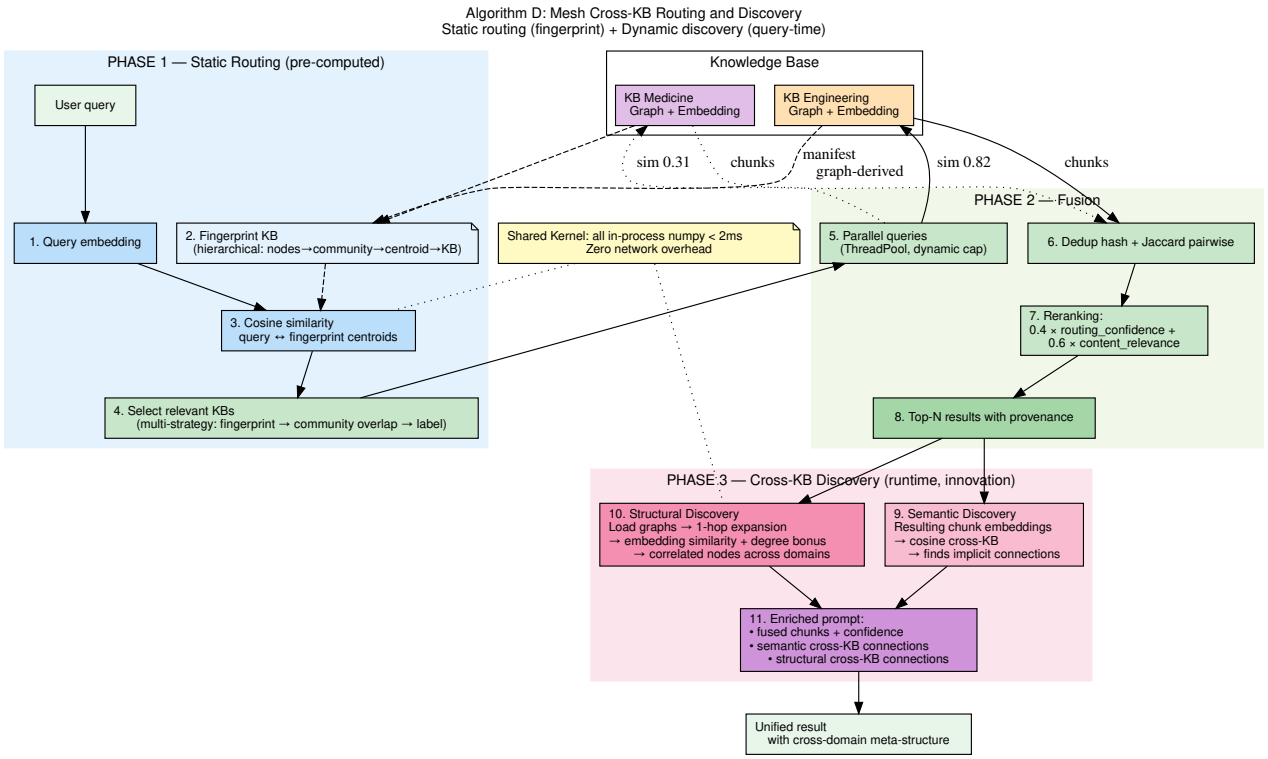

Figure 5: Mesh cross-KB routing: queries are routed to relevant knowledge bases via hierarchical fingerprint matching, then runtime structural discovery identifies emergent connections across separate graphs.

6.2 Hierarchical Fingerprint Routing

Each KB exports a semantic fingerprint derived from its graph structure:

$$\mathbf{fp}_{\text{KB}} = \text{L2-norm}\left(\sum_{c \in \mathcal{C}} w_c \cdot \boldsymbol{\mu}_c\right) \tag{2}$$

where $\mathcal{C}$ is the set of Leiden communities, w_c is the community weight, and $\boldsymbol{\mu}_c$ is the mean embedding of community c's member nodes.

Query routing uses a multi-strategy approach with fallback:

1. *Primary*: cosine similarity between query embedding and KB fingerprints.
2. *Secondary*: community-level semantic overlap.
3. *Tertiary*: lexical label matching (ensures routing even without an active embedding model).

Selected KBs are queried in parallel via a thread pool. Results undergo 5-phase fusion: extraction → confidence scoring (weighted combination of routing affinity and content relevance) → Jaccard n-gram deduplication → reranking → top-k cap.

6.3 Runtime Structural Discovery

After initial retrieval, Mesh discovers emergent connections between KBs through two mechanisms:

Chunk-Level Semantic Discovery. Computes embeddings of retrieved chunks and identifies cross-KB pairs with high similarity that were not pre-computed in any manifest.

Node-Level Structural Discovery. Loads graphs of selected KBs, identifies query-relevant nodes, expands to 1-hop neighbors, and compares nodes across KBs using embedding similarity with a structural bonus based on graph role similarity (incorporating node degree and community membership).

6.4 Natural Break Adaptive Threshold

Instead of a fixed global similarity threshold, Mesh computes the optimal threshold for each specific query:

1. Compute all pairwise similarities between chunks/nodes of different KBs.
2. Sort in descending order and compute first differences (slopes).
3. Detect the "knee": the index where the slope exceeds a statistically-derived threshold based on the distribution's mean and standard deviation.
4. Fallback: if no clear knee exists, use a high percentile as threshold.
5. Safety clamp: the adaptive threshold never falls below a fixed domain-calibrated baseline.

Behavior. For semantically close KBs (e.g., two domains sharing overlapping terminology), the threshold lowers automatically, discovering real connections that a fixed threshold would filter. For distant KBs (e.g., domains with no shared vocabulary), it remains high, filtering noise.

6.5 Measured Results

Evaluation example. Query: "How can stress create hormonal dysfunctions and affect relationships? What methods mitigate negative effects?"
KBs involved: three domain-specific knowledge bases covering health, behavioral science, and interpersonal dynamics.
Model: Gemma 4 E2B (2.3B effective parameters, designed for smartphones, <2 GB RAM).

- **Without** Mesh discovery: Fragmented response with explicit disclaimers about incomplete information. Model admits inability to connect domains.
- **With** Mesh discovery + Natural Break: Organic, confident response connecting cortisol → endocrine system → estrogen/progesterone → emotional regulation → couple communication. Proposes structured strategies (individual: mindfulness; couple: empathic listening, physical connection → oxytocin). Zero disclaimers.

The emergent connection "physical contact → oxytocin → couple bonding" exists in no single document—it was discovered by Mesh at query time by comparing nodes across separate graphs. A 2.3B-parameter model achieves multi-hop cross-domain reasoning that the same model cannot perform without Mesh—demonstrating that the architecture, not model scale, enables emergent discovery.

Critically, Mesh is equally effective at *rejecting* irrelevant connections: a knowledge base built from a software codebase correctly produces zero cross-KB connections with semantically unrelated domains, demonstrating that the routing is selective rather than indiscriminate—avoiding the context pollution that degrades monolithic approaches.

Shared Kernel Architecture. Routing and discovery execute *within* the API process itself (in-process NumPy operations), not as external service calls. This Shared Kernel pattern eliminates network round-trips entirely, achieving routing latency <2 ms. The same in-process execution applies to runtime structural discovery—enabling real-time cross-KB reasoning without the latency penalty of microservice architectures.

7 Evaluation

Table 2 summarizes the capabilities introduced by Noēsis relative to common design patterns in existing Graph-RAG systems.

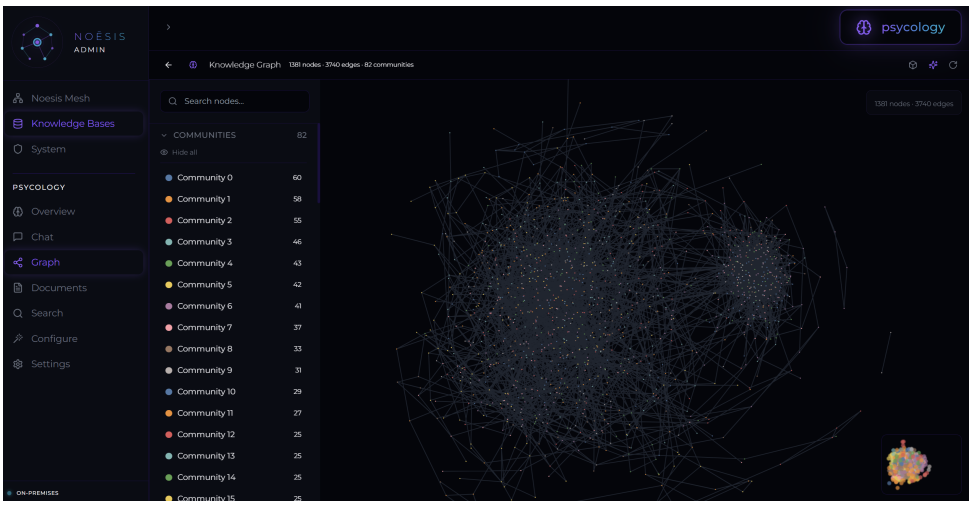

Figure 6: Two-dimensional visualization of a knowledge graph generated by Noēsis (1,381 nodes, 3,740 edges, 82 communities). Node colors indicate Leiden communities; edge thickness reflects relationship strength.

Table 2: Capabilities of Noēsis versus common patterns in existing Graph-RAG approaches†.

Capability	**Existing Approaches**	**Noēsis**
Cross-chunk context during extraction	Rare or limited	✓
Adaptive ingestion parallelism	Not observed	✓
Independently scalable inference backend	Configurable backends, not decoupled services	✓
MoE-aware selective quantization	Architecture-based or generic calibration; static	✓
Cross-KB routing (< 2 ms)	Not addressed	✓
Cross-KB structural discovery (runtime)	Not addressed	✓
Adaptive similarity thresholds	Fixed thresholds	✓
Multi-pass entity deduplication	Basic or none	✓
Extraction-time prompt injection containment	Not addressed	✓
6 GB stress test $\rightarrow$ multi-GPU scaling	Typically requires >16 GB	✓

†*Reflects common patterns observed in published documentation of systems including GraphRAG, LightRAG, LazyGraphRAG, HippoRAG, DynaExq, MoPEQ, APEX, and RAGFlow as of August 2026. Individual systems may address some capabilities through extensions or plugins not reflected here.*

7.1 Multi-Hop QA Benchmark

To evaluate retrieval quality on an established multi-hop reasoning task, we benchmark Noēsis on the HotpotQA validation set (distractor setting) [18], following the same protocol and subset size (1,000 questions) used by HippoRAG [3] and StepChain [19].

Setup. The Noēsis knowledge graph is constructed by Qwen3.6-35B-A3B running on-premises using domain-appropriate extraction prompts with no task-specific optimization or training. Answer generation is performed by GPT-4o. Baseline systems—GraphRAG, HopRAG, and StepChain—use GPT-4o for *both* graph construction and answer generation, representing a substantially more expensive configuration for the extraction phase. BGE dense retrieval uses bge-large-en-v1.5 embeddings with GPT-4o for answer generation. Additionally, Noēsis retrieves k=10 chunks per query, whereas baseline systems in [19] use k=20 passages—a retrieval budget disadvantage of 50% for our system that makes the achieved scores more notable.

Results. Table 3 reports Exact Match (EM) and F1 scores. All baseline numbers are taken from the StepChain paper [19].

Table 3: Multi-hop QA performance on HotpotQA (distractor, 1,000 questions). Baselines from [19]. † denotes systems using GPT-4o for both graph construction and answer generation.

System	EM	F1
GraphRAG† [1]	31.70	42.74
BGE dense + GPT-4o	47.60	60.36
Noēsis + GPT-4o	**59.50**	**74.74**
HopRAG† [20]	62.00	76.06
StepChain† [19]	66.70	79.50

Noēsis achieves 59.50 EM / 74.74 F1, surpassing both GraphRAG (+27.8 EM) and dense retrieval (+11.9 EM), while using a 35B on-premises model for the computationally expensive graph construction phase rather than GPT-4o. Notably, HotpotQA paragraphs average ∼4 sentences—below the threshold at which Noēsis's bidirectional traversal and backward pass activate. The system's primary architectural advantage (long-range cross-section reconnection) is structurally inactive on this benchmark, yet it still achieves 96% of HopRAG's performance. The remaining gap to StepChain (−7.2 EM) and HopRAG (−2.5 EM) reflects primarily the difference in graph construction model capacity (35B on-premises vs. GPT-4o).

Ablation: Architecture vs. Model Scale. To isolate the contribution of the retrieval architecture from the answer-generation LLM, we replace GPT-4o with Gemma 4 E2B (2.3B effective parameters, designed for smartphones, <2 GB RAM). Results are shown in Table 4.

Table 4: Ablation: effect of answer-generation model on HotpotQA. The Noēsis graph (built by Qwen3.6-35B-A3B) is identical across both configurations.

Configuration	EM	F1
BGE dense + GPT-4o	47.60	60.36
Noēsis + Gemma 4 E2B (2.3B)	47.20	60.30
Noēsis + GPT-4o	59.50	74.74

Noēsis with a 2.3B-parameter answer model (EM=47.20, F1=60.30) matches the performance of BGE dense retrieval with GPT-4o (EM=47.60, F1=60.36)—a model approximately 100× larger. This suggests that the retrieval architecture contributes approximately 80% of the overall QA performance, with LLM reasoning capability accounting for the remaining ∼20%. The graph structure delivers sufficiently precise context that even a small model can produce correct answers.

Ablation: Retrieval Budget. To verify whether the k=10 retrieval budget disadvantages Noēsis relative to baselines operating at k=20, we re-run the full 1,000-question evaluation with k=20 chunks. Results: EM=60.90, F1=76.12. Doubling the retrieval budget yields only +1.4 EM, indicating that graph-guided retrieval already captures the relevant multi-hop evidence at lower k. At matched budget (k=20), Noēsis matches HopRAG in F1 (76.12 vs. 76.06) while using a 35B on-premises model for graph construction rather than GPT-4o.

7.2 Source Code Understanding

To demonstrate the source-agnostic nature of the architecture, we evaluate Noēsis on a software source code corpus (55 Python files). The system produces a knowledge graph of 978 nodes and 3,824 edges with density comparable to document-based KBs of similar corpus size.

Query Evaluation. Query: *"Trace the execution flow of a mesh chat query through the system."*

The system correctly identifies the complete 8-step execution pipeline spanning 5+ modules, from HTTP endpoint through orchestration, parallel KB queries, result fusion, cross-KB discovery, to streaming response. This demonstrates that Noēsis delivers equivalent reasoning quality on source code as on natural language documents, without pipeline modifications—confirming the source-agnostic design claimed in §2.

8 Related Work

Graph-RAG Systems. Microsoft GraphRAG [1] introduced community-based summarization but uses static chunking and sequential processing. LightRAG [2] reduces indexing cost but does not address cross-document continuity. LazyGraphRAG [5] eliminates pre-summarization but still chunks independently. HippoRAG [3] models hippocampal memory for retrieval and extracts simplified triples (OpenIE), but does not build dense inter-document graph structures. CrossAug [17] addresses the same underlying problem—missing cross-chunk relations—through a complementary approach: a GNN-guided post-extraction augmentation step that identifies high-scoring regions for LLM-based relation completion. Our bidirectional traversal differs by propagating previously-extracted graph structure as context *during* the extraction process itself, combined with a backward reconnection pass that closes long-range dependencies.

LLM Concurrency Control. CONCUR [4] applies AIMD-based admission control to regulate the number of active agents in LLM batch inference, operating at the GPU KV-cache level within a single serving engine. HiveMind [13] uses AIMD backpressure as part of an HTTP proxy for coordinating concurrent LLM agent workloads. These systems operate on unconstrained batch serving where any request can be processed in parallel with any other.

Our controller addresses a fundamentally different problem. The bidirectional traversal algorithm requires intra-document sequentiality: slices within a single document must be processed in order to build the Graph-Feedback context that the backward pass depends on. This constraint is intrinsic to the extraction method and cannot be relaxed. The concurrency controller must therefore operate on *document-level* granularity (not token, not KV-cache, not agent), adapt to backends with unknown capacity (consumer GPU, cloud API, or local Ollama—no prior specification required), maintain persistent state across distributed workers via a crash-resilient shared store, and resume operation without warmup after worker failure. To the best of our knowledge, no surveyed system combines these properties.

MoE Optimization. Recent work on Mixture-of-Experts quantization has explored multiple approaches to mixed-precision allocation. DynaExq [7] implements a runtime system with hotness-aware precision profiling, non-blocking precision transitions, and fragmentation-free memory pooling for memory-constrained GPU inference. MoPEQ [8] assigns optimal bit-width per expert using Hessian trace approximation. APEX [9] performs per-tensor, per-layer precision allocation based on architectural role and layer sensitivity. Mixture-Compressor [21] folds expert activation frequency into per-expert bit-width allocation. FIDDLER [10] profiles expert popularity for CPU-GPU placement but does not compress the model. HybriMoE [11] optimizes CPU-GPU scheduling on the kTransformers framework. ExpertFlow [12] optimizes expert caching for inference without model compression. imatrix supports domain-specific calibration data but still quantizes all layers uniformly.

Moēsis differs from all of the above in its end-to-end integration of domain-specific intelligence into the quantization lifecycle: it derives compression decisions from actual expert activation patterns observed on a user-provided domain-representative sample, applies a promote-only GPU placement strategy that eliminates repeated CPU↔GPU transfers after selective quantization, and preserves full-precision weights as a reference for runtime re-adaptation when the deployment domain changes—preventing cumulative precision loss across adaptation cycles.

Multi-KB Routing. R1-Router [14] trains an LLM via reinforcement learning to decide when and where to retrieve from multiple KBs during step-wise reasoning. DAKS [15] performs KB routing with budgeted retrieval and alignment graphs for cross-KB evidence fusion. HydraRAG [16] combines graph topology with tri-factor cross-source verification. Adaptive-k [6] uses largest-gap detection for retrieval quantity selection—a different problem from connection discovery thresholds. These systems route queries to relevant KBs but do not discover *emergent structural connections* between separate knowledge graphs at query time. Mesh differs by performing runtime comparison of graph nodes and chunks across KBs using adaptive Natural Break thresholds—discovering connections that exist in no single KB's index.

9 Limitations and Future Work

The measured speedup values (23× for AIMD parallelism, 6.3× for Moēsis prompt processing) depend on the specific hardware and corpus tested. On different corpora or with different backend configurations, the absolute values will vary, although we expect the qualitative trends to hold.

The evaluation presented in this paper spans a standard multi-hop QA benchmark (HotpotQA), two long documents in different languages and formats, a software codebase, and three cross-domain knowledge bases. Broader evaluation across additional benchmarks and domains would further characterize the system's boundaries.

The Mesh cross-KB routing evaluation demonstrates the mechanism on a representative query across three domain-specific KBs. Characterizing performance across a wider range of query types and KB configurations

remains ongoing work.

10 Conclusion

We have presented Noēsis, a fully implemented Graph-RAG system that introduces four algorithmic innovations addressing persistent limitations of existing approaches. The bidirectional traversal with Graph-Feedback context produces knowledge graphs with densities significantly exceeding those of independent chunking, achieving 90% source-verified precision on long-range causal edges extracted from a 193-page document. The AIMD concurrency controller adapts document-level parallelism at runtime under the constraint of intra-document sequentiality—a requirement intrinsic to the bidirectional traversal algorithm—enabling safe, adaptive throughput across heterogeneous hardware without manual configuration. Moēsis makes MoE models more viable on consumer GPUs through domain-aware selective quantization with runtime re-adaptation. And Mesh enables automatic cross-domain reasoning without knowledge base fusion, achieving emergent multi-hop discovery on small on-premises models.

The system has been fully implemented and extensively tested on a corpus exceeding 60 documents and 170 MB across multiple knowledge bases—including PDF, DOCX, a complete software codebase, and single documents approaching 200 pages—processing documents in multiple languages (including non-Latin scripts) and heterogeneous source types. Quantitative evaluation on HotpotQA (§7.1) demonstrates competitive multi-hop retrieval quality, surpassing GraphRAG by +27.8 EM while using a 35B on-premises model for graph construction.

References

[1] D. Edge, H. Trinh, N. Cheng, et al. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[2] Z. Guo, L. Zhao, et al. LightRAG: Simple and Fast Retrieval-Augmented Generation. *EMNLP*, 2025.

[3] B. Gutierrez, Y. Yang, et al. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. *arXiv preprint arXiv:2405.14831*, 2024.

[4] Q. Chen et al. CONCUR: High-Throughput Agentic Batch Inference of LLM via Congestion-Based Concurrency Control. *arXiv preprint arXiv:2601.22705*, 2026.

[5] Microsoft Research. LazyGraphRAG: Setting a new standard for quality and cost. *Microsoft Research Blog*, 2024. `https://www.microsoft.com/en-us/research/blog/lazygraphrag-setting-a-new-standard-for-quality-and-cost/`

[6] Adaptive-k Authors. Efficient Context Selection for Long-Context QA. *arXiv preprint arXiv:2506.08479*, 2025.

[7] DynaExq Authors. DynaExq: Dynamic Expert Quantization for Scalable Mixture-of-Experts Inference. *arXiv preprint arXiv:2511.15015*, 2025.

[8] K. T. Chitty-Venkata, J. Ye, M. Emani. MoPEQ: Mixture of Mixed Precision Quantized Experts. *Proceedings of the IEEE/CVF ICCV Workshops*, 2025.

[9] E. Di Giacinto, R. Palethorpe. APEX: Adaptive Precision for Expert Models. Technical Report, LocalAI, March 2026. `https://github.com/localai-org/apex-quant`

[10] FIDDLER Authors. FIDDLER: CPU-GPU Orchestration for Fast Inference of Mixture-of-Experts Models. *ICLR*, 2025.

[11] HybriMoE Authors. HybriMoE: Hybrid CPU-GPU Scheduling and Cache Management for Efficient MoE Inference. *arXiv preprint arXiv:2504.05897*, 2025.

[12] ExpertFlow Authors. ExpertFlow: Optimized Expert Activation and Token Allocation for Efficient Mixture-of-Experts Inference. *arXiv preprint arXiv:2410.17954*, 2024.

[13] HiveMind Authors. HiveMind: HTTP Proxy with AIMD Backpressure for Concurrent LLM Agent Workloads. *arXiv preprint arXiv:2604.17111*, 2026.

[14] C. Peng, Z. Xu, Z. Liu, et al. R1-Router: Learning to Route Queries Across Knowledge Bases for Step-wise Retrieval-Augmented Reasoning. *arXiv preprint arXiv:2505.22095v1*, 2025.

[15] DAKS Authors. Traceable Cross-Source RAG for Chinese Tibetan Medicine Question Answering. *arXiv preprint arXiv:2602.05195*, 2026.

[16] HydraRAG Authors. HydraRAG: Structured Cross-Source Enhanced Large Language Model Reasoning. *arXiv preprint arXiv:2505.17464*, 2025.

[17] CrossAug Authors. CrossAug: GNN-Guided Cross-Chunk Graph Augmentation for Graph-RAG. *arXiv preprint arXiv:2605.28004*, 2026.

[18] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, C. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. *EMNLP*, 2018.

[19] T. Ni, X. Yuan, W. Zhang, S. Li, K. Wu, R. P. Liu, W. Ni. StepChain GraphRAG: Reasoning Over Knowledge Graphs for Multi-Hop Question Answering. *arXiv preprint arXiv:2510.02827*, 2025.

[20] H. Liu et al. HopRAG: Multi-Hop Reasoning for Logic-Aware Retrieval-Augmented Generation. *ACL Findings*, 2025.

[21] Mixture-Compressor Authors. Mixture-Compressor: Expert-Aware Quantization for Mixture-of-Experts Models. *arXiv preprint*, 2024.