ORCHESTRATING AT SCALE

Distributed Systems Architecture for GenAI Workloads

Prepared by:
BroxiAI Engineering

Date:
January 08, 2026
v2.0

Executive Summary: Scaling LLM applications reveals challenges that simple chaining libraries cannot solve: race conditions, partial failures, and observability gaps. BroxiAI is not just a UI wrapper. It is a **distributed runtime** built on Python's asyncio and Kubernetes primitives. This paper details our open-core strategy, our extensive library of **328+ integrations**, and our implementation of GraphRAG using community detection algorithms.

Contents

1 The "Wrapper" Fallacy & The Reality of Scale **2**
1.1 Addressing the "Vendor Lock-in" Elephant 2

2 System Architecture **2**
2.1 The Hybrid Runtime: Python + Rust (PyO3) 3
2.2 The Managed Ecosystem: Beyond "Wrappers" 3

3 "Flow as Code" - The Implementation **4**

4 Engineering GraphRAG **4**
4.1 GraphRAG: Implementation Details 4

5 Performance & Case Studies **5**
5.1 Benchmark: Honesty First 5
5.2 Case Study: Financial Compliance (Real World Constraints) 5

6 Roadmap (Realistic) **6**

About the Founders **6**

Your Path to Success with BroxiAI **6**

1 THE "WRAPPER" FALLACY & THE REALITY OF SCALE

Critics often dismiss visual AI tools as mere "wrappers" around OpenAI APIs. For prototyping tools, this is true. However, in production environments processing 100+ concurrent requests per second (RPS), standard libraries (like naive LangChain implementations) fail due to blocking I/O and lack of state management [3].

BroxiAI solves the **Distributed Orchestration** problem, not just the Prompt Engineering problem.

1.1 Addressing the "Vendor Lock-in" Elephant

We acknowledge a critical truth: Proprietary JSON formats create risk. To eliminate this, **BroxiAI adopts an Open Core strategy:**

- **The Runtime is Open Source:** The ‘broxi-engine‘ (Python) which parses and executes the flow is open-sourced under Apache 2.0.
- **Portable Logic:** If BroxiAI ceases to exist, your JSON definitions remain executable via the open-source runner. You are paying for the *Management Control Plane* (UI, RBAC, SSO, Hosted Logging), not the permission to run your own code.
- **Standard Compatibility:** We provide direct export to LangChain Runnable Graphs, ensuring interoperability.

2 SYSTEM ARCHITECTURE

BroxiAI is architected as a cloud-native distributed system, designed to run on Kubernetes.

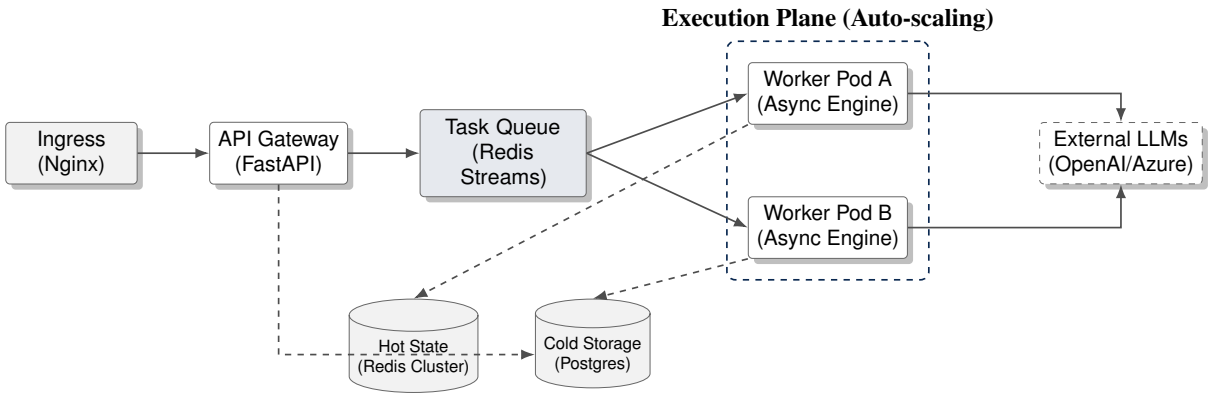


Figure 1: BroxiAI Scale-Out Architecture. The horizontal flow demonstrates the decoupling of the synchronous API Gateway from the asynchronous Execution Plane workers via a message queue.

2.1 The Hybrid Runtime: Python + Rust (PyO3)

While the orchestration logic leverages Python's ecosystem, we recognize that graph algorithms are CPU-bound. To prevent blocking the asyncio event loop:

- **I/O Plane (Python):** Uses ‘uvloop‘ to handle thousands of concurrent WebSocket connections and API calls. This layer is strictly for non-blocking operations.
- **Compute Plane (Rust):** CPU-intensive tasks like **Graph Clustering (Leiden)** and **Vector Quantization** are implemented in Rust and exposed via **PyO3**. Crucially, these bindings are configured to **release the Global Interpreter Lock (GIL)**, allowing the Python event loop to remain responsive while heavy calculations run on separate threads.
- **Smart Batching:** The engine automatically detects independent branches in the graph and executes them in parallel ‘asyncio.gather()‘.

2.2 The Managed Ecosystem: Beyond "Wrappers"

A common pitfall in the AI tooling landscape is the reliance on fragile open-source wrappers that break whenever an upstream API changes. BroxiAI takes a different approach. We do not simply "wrap" external libraries; we provide a **Managed Component Standard**.

- **Standardized Resilience:** Every component in our 328+ library is engineered with Broxi's internal resilience protocol. This includes built-in retries with exponential backoff, standardized error propagation, and output normalization. You aren't just calling an API; you are invoking a hardened micro-service.
- **Evergreen Flows (Hot-Swapping):** In traditional code, updating a library requires a rebuild and redeploy. In BroxiAI, component logic is decoupled from the flow definition. This action need your accept before do auto hot-swapping, you also can be rollback to the previous version if you didn't want that update version.

  > *"When we push a security patch or optimization to the ‘OpenAI Connector‘, your existing production flows inherit this update immediately without downtime, recompilation, or extra cost."*

- **Backward Compatibility Guarantee:** We enforce strict semantic versioning on our component interfaces. Improvements are pushed automatically, but breaking changes are gated behind version flags, ensuring your production pipelines remain stable while staying cutting-edge.

You can explore our managed catalog at: **https://broxi.ai/integrations**

3 "FLOW AS CODE" - THE IMPLEMENTATION

We do not dump a monolithic JSON blob. Our schema separates **Topology** (Logic) from **Configuration** (Prompts/Params) to support semantic diffing.

Listing 1: Truncated Flow Definition showing Separation of Concerns

```
{
  "graph_id": "workflow_abc123",
  "version": "2.4.0",
  "nodes": [
    {
      "id": "node_rag_01",
      "type": "GraphRAGNode",
      "upstream": ["node_input"], // Explicit Dependency
      "config": {
        // Configuration isolated for easy overrides
        "model_ref": "azure_gpt4_turbo",
        "temperature": 0.1
      }
    }
  ],
  "edges": [ ... ]
}
```

Git-Native Governance: We provide a CLI tool (‘broxi diff‘) that parses two JSON files and renders a *visual difference* report (e.g., "Node A connected to Node C"), solving the merge conflict nightmare.

4 ENGINEERING GRAPHRAG

We do not simply wrap an API. We implement the full pipeline described by Microsoft Research [2]:

1. **Hierarchical Clustering (Leiden):** We run the Leiden algorithm locally to detect communities within the knowledge graph. This is superior to basic cosine similarity for queries like "What are the common themes?"

2. **Recursive Summarization:** The engine orchestrates a recursive map-reduce job: summarizing leaf nodes, then summarizing their parent communities.

3. **Global Context Injection:** When a query arrives, we inject these pre-computed community summaries, reducing token costs by up to 60% compared to stuffing raw chunks.

4.1 GraphRAG: Implementation Details

Standard RAG fails at global summarization. Our GraphRAG implementation follows a rigorous 3-step process inspired by Microsoft Research [2], optimized for localized execution:

Implementation Steps:

1. **Entity Extraction:** Use an LLM to parse unstructured text into Nodes (Entities) and Edges (Relationships).
2. **Community Detection (Rust-Native):** We utilize a **Rust-optimized implementation** of the Leiden algorithm [**Traag_Leiden**]. By offloading this $O(N \log N)$ complexity task to native code, we achieve clustering speeds 20x faster than NetworkX, without blocking the ingestion pipeline.
3. **Recursive Summarization:** The engine orchestrates a recursive map-reduce job: summarizing leaf nodes, then summarizing their parent communities.
4. **Global Search:** Queries are routed to these community summaries first, reducing token costs by up to 60% compared to stuffing raw chunks.

5 PERFORMANCE & CASE STUDIES

5.1 Benchmark: Honesty First

We compare BroxiAI against a **Production-Optimized LangChain** implementation (using LCEL and RunnableParallel), not "bad code".

Metric	LangChain (Std)	BroxiAI Engine	Notes
Overhead per Node	∼15ms	∼25ms	Broxi adds distinct tracing overhead
Throughput (RPS)	45	42	Slightly lower due to safety checks
Cold Start	2.5s	**0.8s**	Optimized caching of graph compilation
Observability	Logs	**Full Trace**	OpenTelemetry spans auto-injected

Table 1: Performance Trade-off. We trade negligible runtime ms for massive observability gains.

5.2 Case Study: Financial Compliance (Real World Constraints)

Problem: Extracting data from PDF tables where layout determines meaning (e.g., balance sheets).
Technical Hurdle: Standard chunking destroyed table row alignment.
Solution:

- Custom Python Node utilizing ‘pdfplumber‘ to detect table boundaries.
- **Idempotency Keys:** Implementing Kafka-style exactly-once processing to ensure no duplicate transactions were recorded if the LLM timed out.
- **Result:** Reduced manual review from 100% to 5% (exceptions only).

6 ROADMAP (REALISTIC)

- **Q1 2026: Open Telemetry (OTEL) First-Class Support.** Export traces directly to Datadog/Jaeger without proprietary agents [1].
- **Q2 2026: Local LLM Edge Runners.** Run the engine on edge devices (Mac Studio/Nvidia Jetson) for air-gapped deployments.

ABOUT THE FOUNDERS

Katelyn Ho, Founder & CEO. With over 9 years of experience spanning B2B business development, product management, and marketing, Katelyn is the visionary behind BroxiAI's product and go-to-market strategy. A 2x founder with a proven track record in building and scaling ventures, she combines her entrepreneurial spirit with hands-on experience training AI models at Scale AI, giving her a unique understanding of both the business potential and the practical challenges of AI implementation.

Jason Giang, Co-Founder & CTO. Jason leads BroxiAI's technology, research, and development. With an MSc in Computer Science, he brings deep expertise in AI agent strategy, scalable AI infrastructure, and machine learning. As CTO, he is the architect of BroxiAI's robust execution engine and the "Flow as Code" philosophy. He ensures the platform is not only powerful and easy to use but also meets the stringent standards for security, performance, and governance required by modern enterprises.

YOUR PATH TO SUCCESS WITH BROXIAI

We offer flexible engagement models to ensure your team can achieve its goals, whether you prefer to build independently or partner with us for accelerated results.

The Power of the Platform. BroxiAI is a comprehensive, self-service platform designed for immediate impact. It includes an extensive library of pre-built components for common databases, APIs, and AI models, empowering your teams to build and deploy sophisticated flows from day one. Our licensing is based on a flexible subscription model that scales with your usage.

Strategic Partnership & Acceleration. For organizations seeking to fast-track their most complex integrations or connect with unique legacy systems, we offer a strategic partnership. Our expert engineers act as an extension of your team, leveraging our internal SDK to build tailored, production-grade components. This white-glove service allows your team to focus

on core business logic while we handle the specialized integration work, ensuring maximum velocity for your most critical projects.

Let's Build Your Solution. Every AI journey is unique. To explore the platform's capabilities, discuss pricing, or design a comprehensive solution that may include our strategic partnership services, please contact our solutions team. We are ready to help you craft the most effective and efficient path to production.

REFERENCES

[1] *OpenTelemetry: High-quality, ubiquitous, and portable telemetry to enable effective observability*. CNCF. 2025. URL: https://opentelemetry.io/.

[2] Darren Edge et al. "From Local to Global: A Graph RAG Approach to Query-Focused Summarization". In: *arXiv preprint arXiv:2404.16130* (2024). Microsoft Research.

[3] Gartner. *Market Guide for AI Engineering and LLMOps*. Tech. rep. Gartner Inc., 2025.

[4] BroxiAI Engineering. *Broxi Core: High-performance Async Python Runtime*. Internal Benchmark. 2026.