ArcNeural

A Multi-Model Database for the Gen-AI Era

By Wu Min,

Fabarta Inc. & Hangzhou Dianzi University

Contents

1 Introduction & Motivation

- Generative AI Revolution
- Enterprise Needs in the Gen-AI Era
- Demand for Multi-Model Solutions

2 Design of ArcNeural

- System Overview
- Storage Layer
- MemEngine
- Vector Indexing
- GraphHTAP

3 Enterprise Applications

- Criminal Network Detection
- Chat to Graph
- Lessons Learned

4 Conclusion & Q&A

Generative AI Revolution

Transformative Impact

Over the past few years, Generative AI (Gen-AI) has dramatically transformed the technology landscape, particularly with the rise of Large Language Models (LLMs).

Industry-Wide Effects

The impact of Gen-AI is felt across industries: from AI-driven customer service (chatbots) to recommendation systems, personalized content creation, and automated decision-making.

Business Needs in the Gen-AI Era for Enterprise Customers

Unstructured Data (80%)	Semi-structured Data	Structured Data (20%)
Documents, emails, reports, etc.	JSON, XML, etc.	MySQL, Oracle, CSV

Currently, only 20% of enterprise data is structured, while 80% exists in the form of unstructured and semi-structured data (e.g., documents, emails, reports).

—— IDC

Existing Applications

- Document management systems
- Search engines for internal information
- Knowledge graphs

Generative AI Use Cases

- Internal Document Q&A
- Natural Language Querying
- Document Understanding, Summarization, and Generation

Intelligent Decision-Making

- Decision-making
- APIs or system-to-system integration

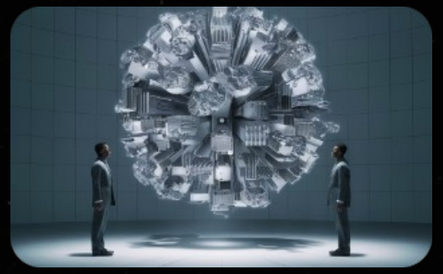
Demand for Real-Time Intelligent Decision-Making

Real-Time Data Processing

Enterprises are increasingly relying on AI for decision-making, driving a significant demand for real-time data processing capabilities.

Massive Data Volumes

The adoption of AI-driven intelligence requires databases to handle massive data volumes, including diverse data types and complex relationships.



Data-Driven Intelligence

The shift towards AI-driven intelligence necessitates databases that can support the efficient processing and analysis of vast datasets.

Problem & Motivation

Inefficiencies in Existing Databases

Traditional databases are optimized for structured, relational data, making them ill-suited for managing the multi-model data required by modern AI applications. They struggle to efficiently perform vector searches or capture complex relationships between data points.

Inefficiencies in RAG Systems

Retrieval Augmented Generation (RAG) systems, which leverage LLMs for more accurate responses, often suffer from inefficient contextual retrieval. Existing databases lack the capabilities to effectively index and retrieve information based on semantic context.

Demand for a New Approach

The need for a new generation of databases that can handle multi-model data, support efficient vector searches, and enable effective contextual retrieval is crucial for the advancement of AI applications. This new database should be tailored for the unique requirements of LLM-based systems.

Business Scenarios Requiring Multi-Model Solutions Beyond RAG

No.	Typical Problem	Recommended Approach	Explanation
1	Recommend tourist spots in Guangzhou	General LLM Capabilities	Standard large model response based on common knowledge
2	Company leave policy	Local Knowledge + RAG	Requires RAG to access specific internal documents
3	Compliance in financial reporting	Structured Data + RAG	Requires filtering structured data to ensure compliance
4	Sales ranking of product lines in North China for February	Structured Data + Large Model Query	Combination of structured data and AI-driven query for sales insights
5	Credit card applicants linked to a specific technology company	Large Model + Graph + Vector	Requires graph and vector searches to match and identify complex connections
6	Data lineage in supply chain information table	LLM + Graph	Investigate data lineage through graph queries
7	Analyze causes for revenue decline and suggest actions	Large Model + Agent + Small/Large Models	A complex scenario requiring a combination of models for decision-making insights

Graph

Vector

HTAP

Contents

1 Introduction & Motivation

- Generative AI Revolution
- Enterprise Needs in the Gen-AI Era
- Demand for Multi-Model Solutions

2 Design of ArcNeural

- System Overview
- Storage Layer
- MemEngine
- Vector Indexing
- GraphHTAP

3 Enterprise Applications

- Criminal Network Detection
- Chat to Graph
- Lessons Learned

4 Conclusion & Q&A

Design of ArcNeural

1

System Overview

Comprehensive look at ArcNeural's architecture

2

Storage Layer

Efficient data persistence and retrieval

3

MemEngine

In-memory processing for high performance

4

Vector Indexing

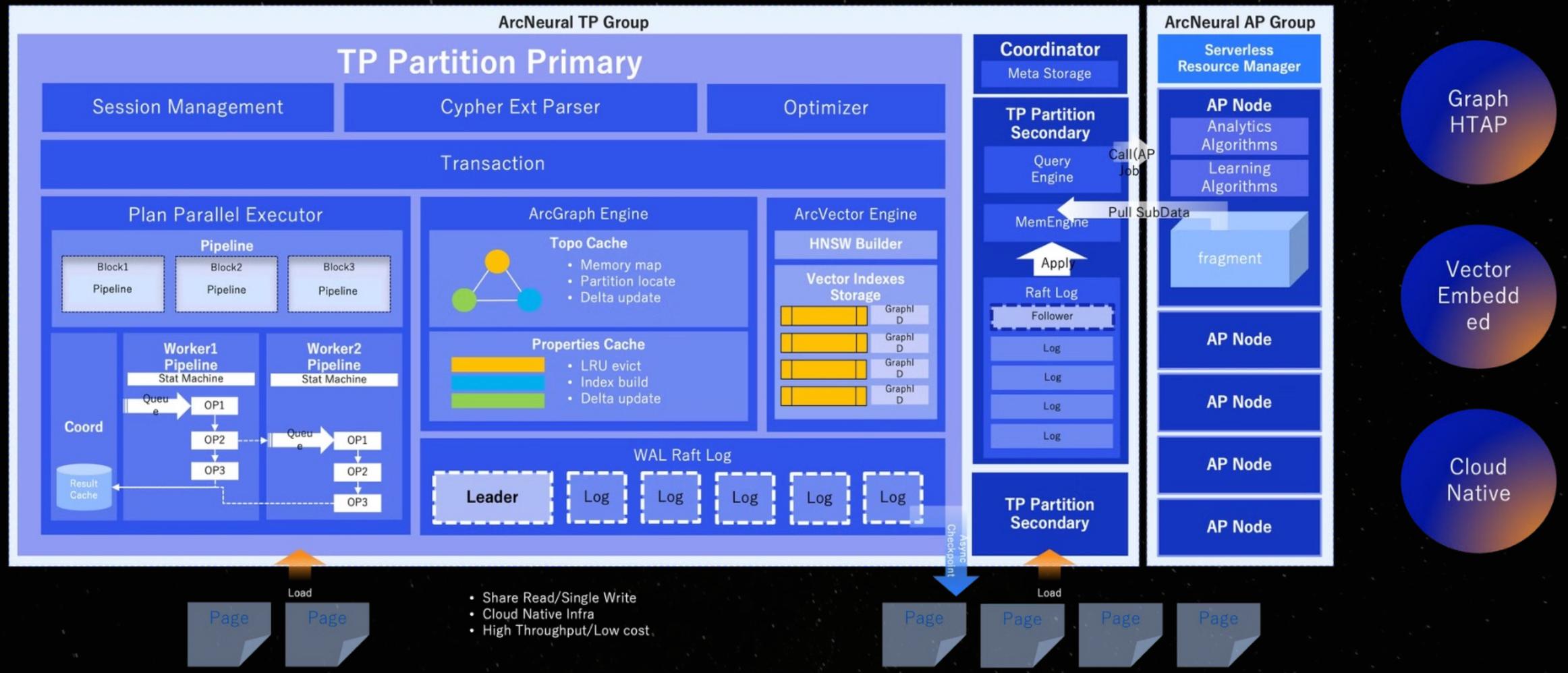
Advanced indexing for multi-dimensional data

5

GraphHTAP

Hybrid transactional and analytical processing for graph data

Design Overview of ArcNeural



ArcNeural TP Group

Includes TP Partition Primary, Cypher Ext Parser, Optimizer, Session Management, Plan Parallel Executor, and ArcGraph Engine

ArcNeural AP Group

Consists of AP Nodes with Analytics Algorithms and Learning Algorithms

Storage Layer

Incorporates MemEngine, ArcVector Engine, and various storage options

Key Features

Share Read/Single Write, Cloud Native Infra, High Throughput/Low cost, Vector Embedded, Graph HTAP, Cloud Native

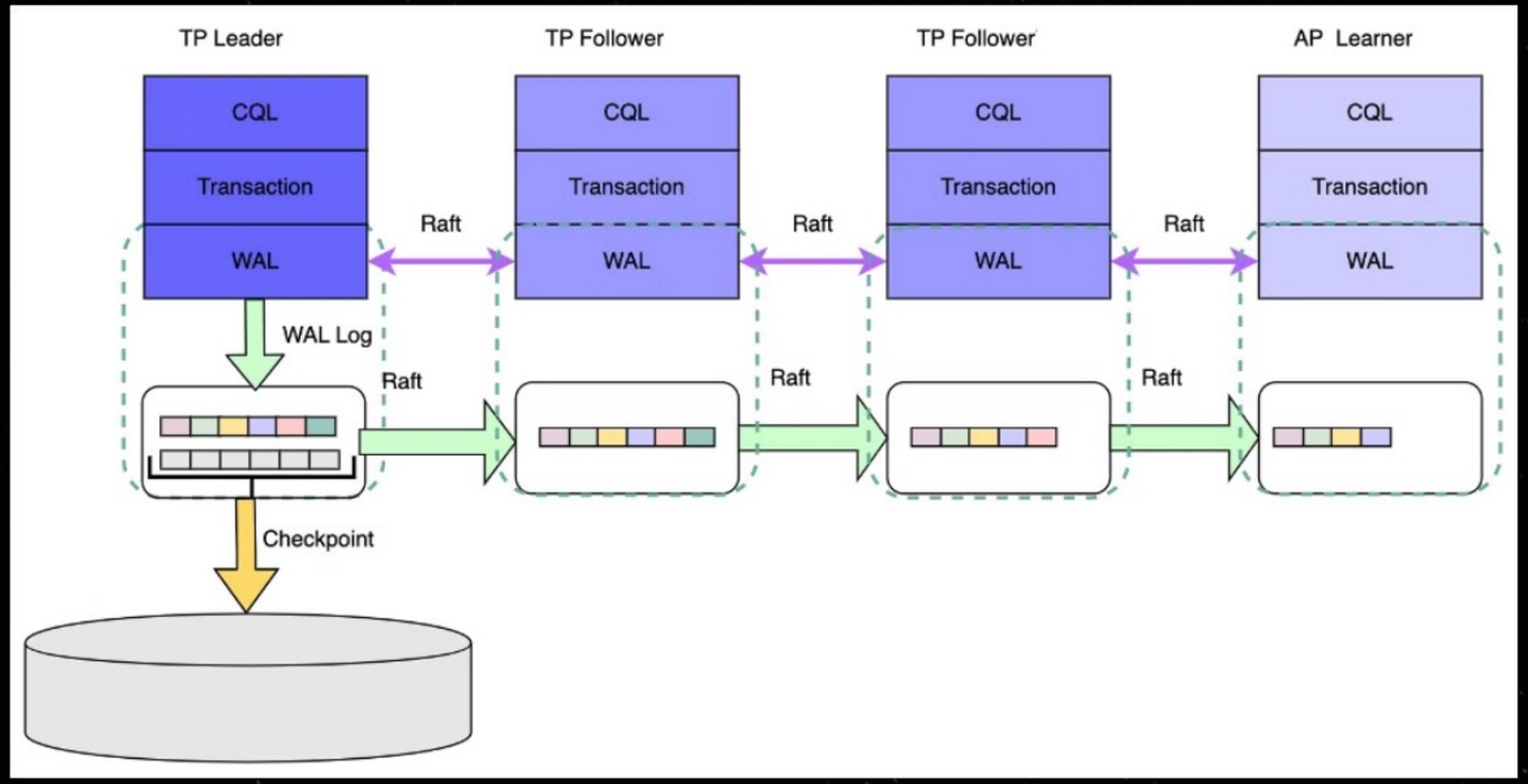
Storage: Persistent Layer

Log-as-Data

Write-Ahead Log (WAL) plays a central role in ensuring data consistency and persistence.

Persistent

- Single-node: RocksDB
- Distributed: TiKV
- Cloud-based storage: AWS S3, or OSS (Alibaba Cloud)



MemEngine: Graph Topology In-Memory Cache

Graph Topology & Attribute Data

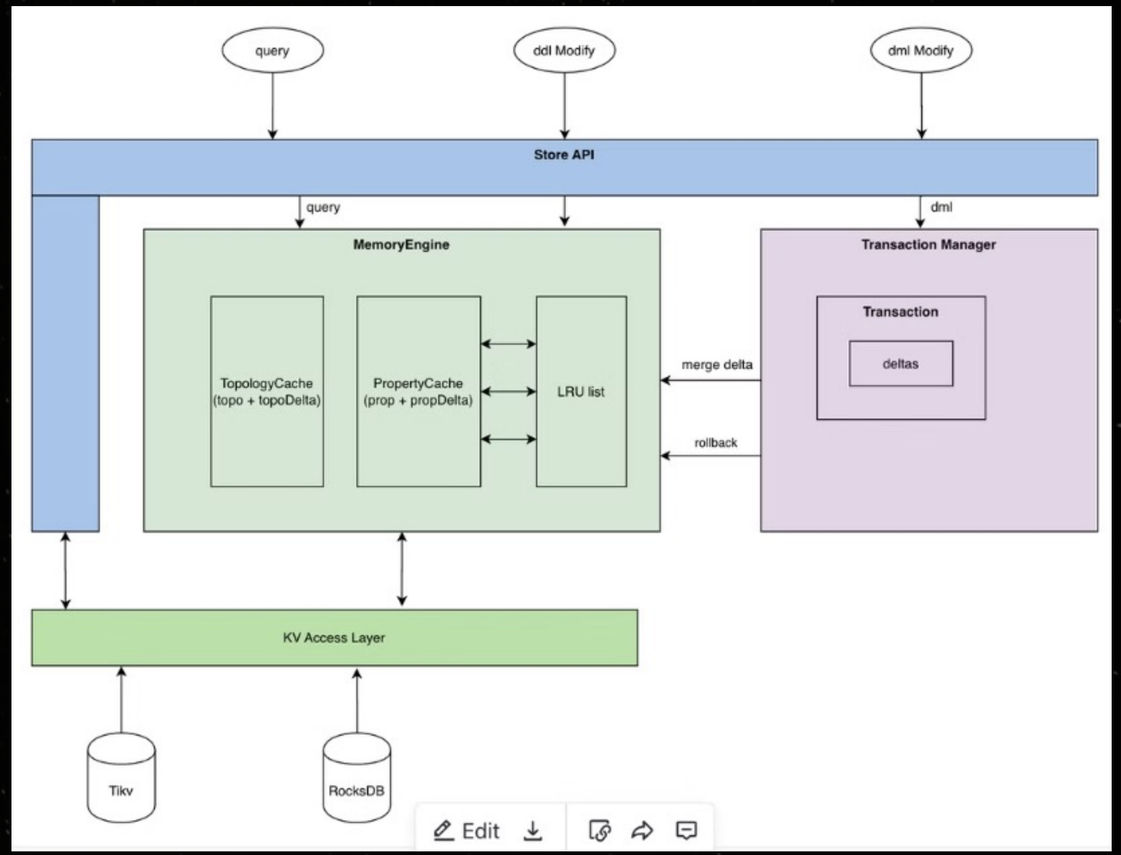
MemEngine stores *graph topology* and *attribute data* in memory.

Adaptive Edge Collection

Handling Skewed Vertex Connectivity.

For vertices with a low degree: a vector like structure.

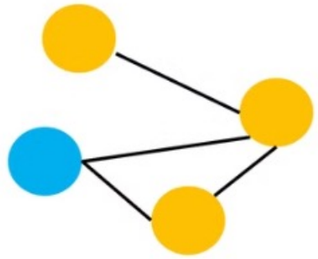
Vertices with a high degree: B-tree or a hash table.



Vector Indexing for High-Performance AI Workloads

Local storage uses memory-mapped files (*MMAP*) for HNSW indexing, while payload data is stored using *RocksDB*

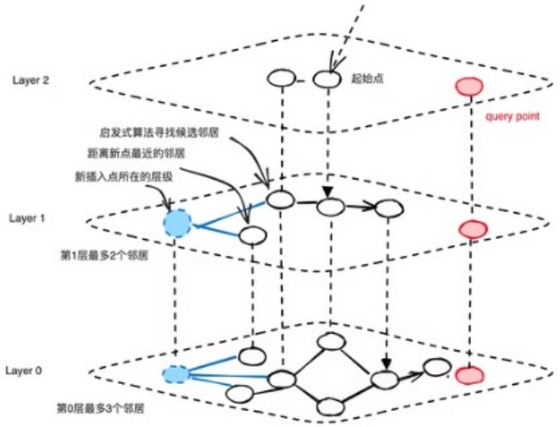
```
CREATE VERTEX IF NOT EXISTS Movie(
        PRIMARY KEY vid INT(64), info JSON
        content VECTOR(1536),
        VECTOR INDEX v_idx(content) HNSW
)
```



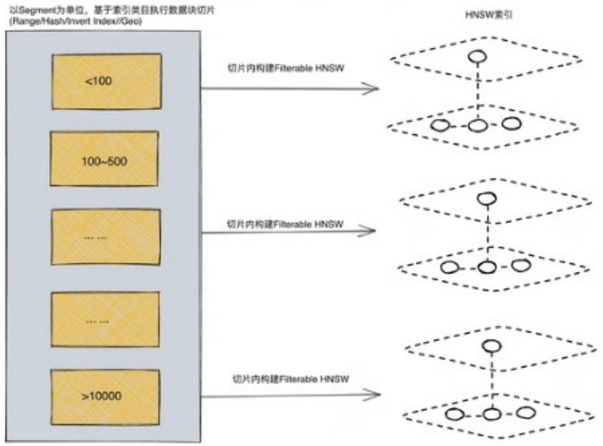
column	value
vid	0x110111
info	{title: “xxx” , year: 2024}
content	[0.9, 1.1……]

Arc HNSW Vector Index

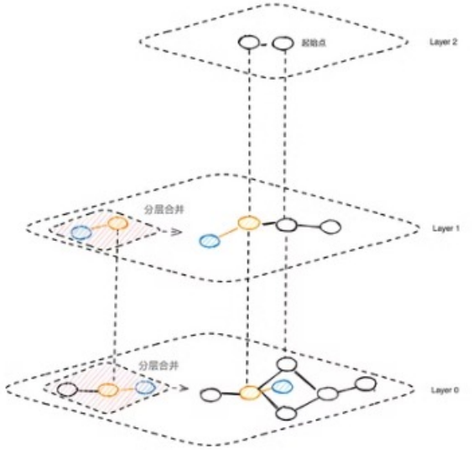
✓ SIMD Acceleration: 4x Performance Boost:



✓ Attribute Filtering: TopK by Field Attributes



✓ CRUD & ACID



Unified Graph Transactional and Analytical Processing

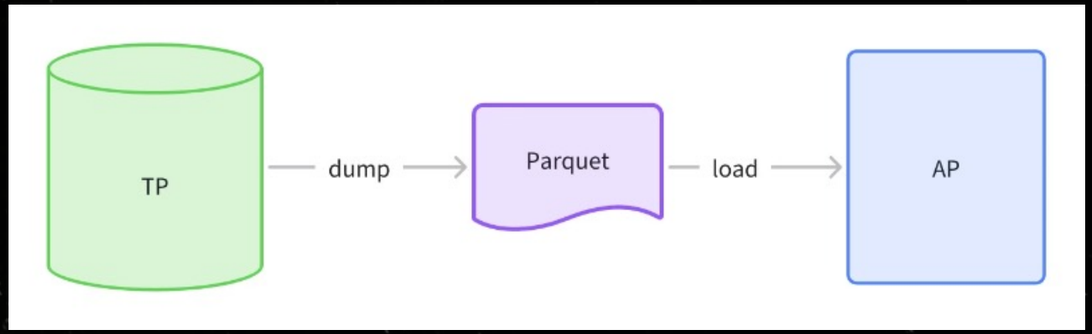
Unified Query Interface

- Cypher extend: A powerful query language that allows for graph data manipulation and analysis.
- Handle both real-time transactions and deep analytical workloads

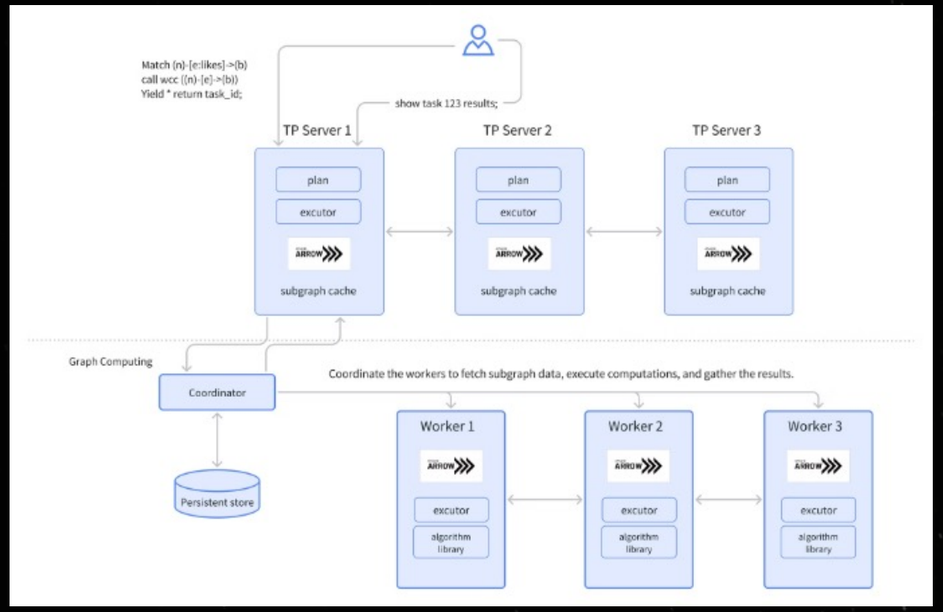
The TP layer performs semantic analysis to determine if an analytical (AP) operation is needed when a query request is received.

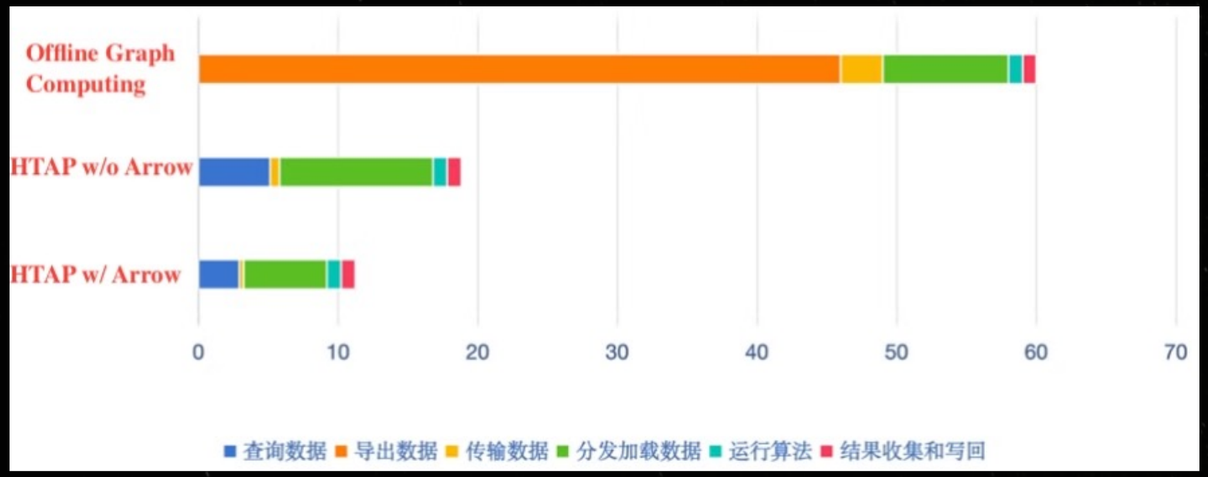
Example

```
-- Find the subgraph
MATCH (n)-[e:likes]->(b)
-- Call the PageRank algorithm on the specified
subgraph
CALL pagerank((n)-[e]->(b))
YIELD *  RETURN task_id; id;  --  Returned as an
asynchronous task.
```



Native Graph Computation with Apache Arrow





1. Transfer Bottlenecks and overhead (local GraphAR file)

2. **Arrow** is a *columnar format*, ideal for handling large-scale

3. **same Apache Arrow format** used in *TP* is also directly compatible with the *Analytical Processing (AP)*, which Eliminats the Data Restructuring overhead.

Contents

1 Introduction & Motivation

- Generative AI Revolution
- Enterprise Needs in the Gen-AI Era
- Demand for Multi-Model Solutions

2 Design of ArcNeural

- System Overview
- Storage Layer
- MemEngine
- Vector Indexing
- GraphHTAP

3 Enterprise Applications

- Criminal Network Detection
- Chat to Graph
- Lessons Learned

4 Conclusion & Q&A

Criminal Network Detection in Encrypted Communications

Encrypted Platforms

Encrypted communication platforms like Telegram have become a hub for criminal activity.

Online Gambling

Criminal networks use platforms like Telegram to run illegal online gambling operations, often involving high-stakes bets and rigged games.

Pornography

Telegram is used to share illegal pornography, often involving child exploitation and other forms of exploitation.

Gun Trading

Criminal networks use Telegram to trade firearms illegally, often connecting buyers and sellers who operate outside the law.

Graph Analysis and Large Language Models: Solutions for Criminal Network Detection

Relationship Mapping

Revealing hidden connections between suspects by analyzing individual and group chat data.

Cross-Platform Analysis

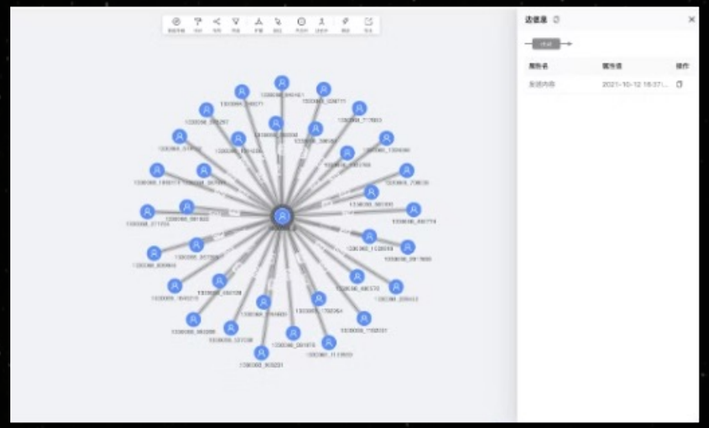
Identifying criminal networks operating across multiple platforms using shared identifiers.

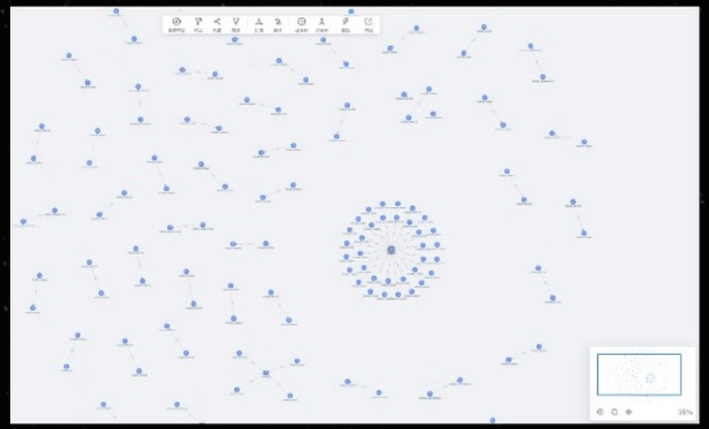
Behavior Analysis

Detecting suspicious behaviors and potential collaborators by examining interaction patterns.

Association Analysis

Pinpointing key individuals and analyzing links between suspicious accounts to establish stronger connections.





Large Language Models (LLMs) Enhance Graph

Content Analysis

Analyze communication data to flag potential criminal activities.

Key Information Extraction

Extract vital details like phone numbers, addresses, and card numbers from chat logs.

Lead Summarization

Summarize evidence to highlight key leads in a case.

Intelligent Interaction

Enable investigators to query the model for insights and connections.

How to Chat to Graph? Option1: Text2Cypher

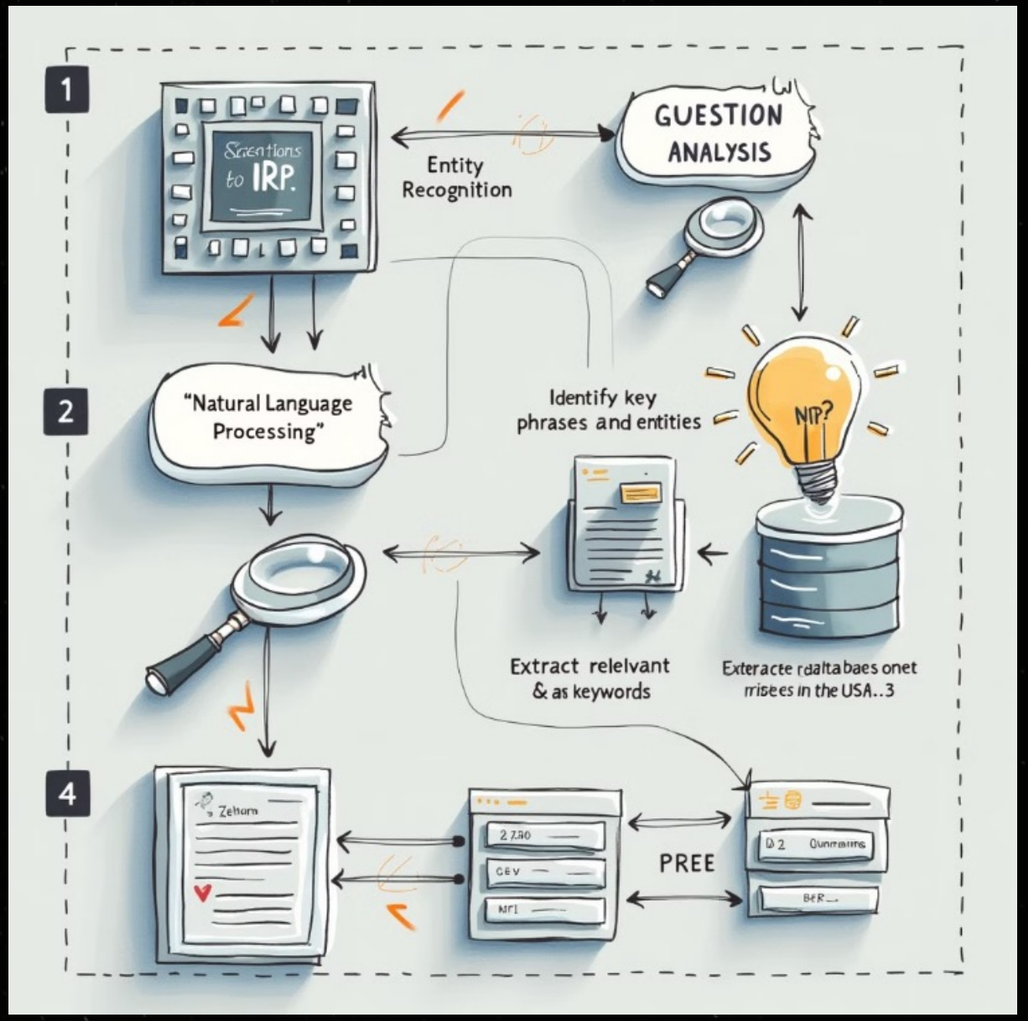
Direct Cypher query generation that translates natural language questions into a Cypher query for ArcNeural.

1. A user asks an investigative question in plain text (e.g., "Who are the collaborators of {user_id}?").
2. The system processes the text and identifies the appropriate graph traversal and features needed.
3. A Cypher query is generated to retrieve the answer from the graph.

Example Query

Question: "What is the IP address of user {user_id}?"

```
MATCH (u:User {id: '{user_id}'})
RETURN u.ip_address AS IP_Address
```



This approach may not handle complex reasoning or multi-step operations well. Some Cypher sentences (e.g., Having clause) are not implemented

Option 2: GraphAgent

1 **Understanding the Question**

The agent figures out what graph operations are needed.

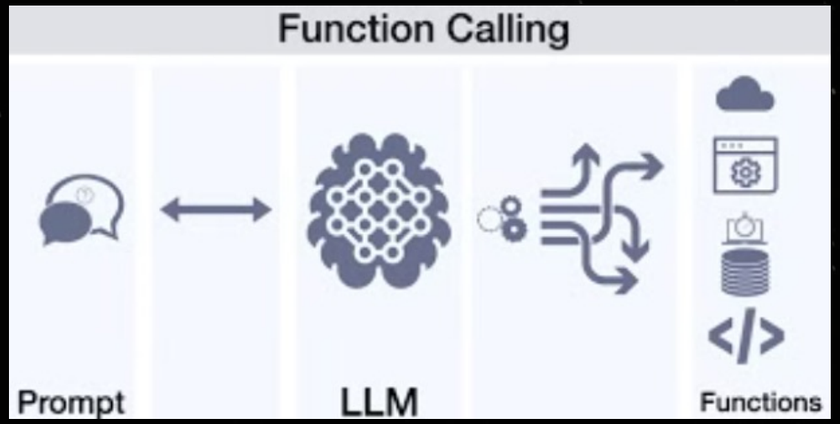
2 **Creating a Plan**

The agent makes a plan of steps to complete the task (e.g., get nodes, check connections, count connections).

3 **Executing APIs**

The agent uses several APIs to get and analyze data from the graph.

Optimize workflow by employing agents for **function calling**.



NodeDegree: Gets the number of connections for a node.

NodeFeature: Gets a specific feature of a node (e.g., IP address).

NodesFeature: Gets features for multiple nodes.

NeighbourCheck: Checks if two nodes are connected.

RetrieveNode: Gets a node using its identifier.

RetrieveNodeWithType: Gets a node of a specific type (e.g., User, Group).

```
plan_prompt = ( "Let's first understand the
problem and devise a plan to solve the
problem." f"Each step is a interaction with
Graph, the schema of Graph is {schema}"
...
"At the end of your plan, say
'<END_OF_PLAN>'" )
```

Acts as a method when **Text2Cypher** fails to generate the correct Cypher query.

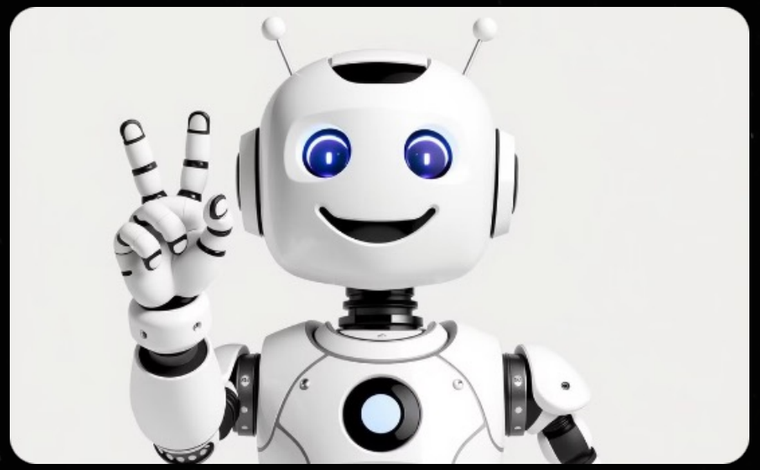
Limitations:

Slower Performance: Significantly slower than Text2Cypher because it involves generating an execution plan and calling APIs step-by-step.

Execution Overhead: The execution context is longer, requiring more resources and time, especially when handling large or complex graph structures.

More Complexity: Requires careful coordination of multiple API calls, which can introduce complexity in implementation.

Types of Graph Queries



Easy

These questions can be answered by looking up the feature/degree of only one node or travel on the graph within one hop.

For example, “What is the IP address of the user with ID {user_id}? or "Who are the members of the group with ID {group_id}?"

Medium

These questions require reasoning on the graphs for more than one hop and involve returning the feature/degree of nodes.

For example, “Who are the most connected members within the criminal group with ID {group_id}?”or "Who are the key influencers in a criminal network connected to user {user_id}?".

Hard

These questions cannot be directly answered by looking up the graph, but the graph can be useful by providing informative context.

For example, “What communication patterns suggest that {user_id} may be hiding their identity?” or **“What are the suspicious activity trends among users in {group_id}?"**

Lessons Learned from Building Graph-AI Systems

1 Graph Questions Are Challenging for LLMs

State-of-the-art LLMs are needed for graph understanding. Break down tasks for better results.

2 Design an Effective Workflow

Use data labeling, a clear graph schema, and community detection. Create meaningful features (node degrees, centrality measures).

3 Chain-of-Thought (CoT) and Few-Shot Learning are Valuable

Use Chain-of-Thought (CoT) prompting and Dynamic Few-Shot Learning (examples: Natural Language → CoT → Cypher/API)

4 Leverage LLMs for Insights, Not Perfect Answers

Don't rely on LLMs for absolute accuracy. Use them to generate useful insights and initial leads, saving analysts time.

5 Self-Consistency Enhances Robustness

LLMs can produce inconsistent results. To address this, generate multiple solutions for the same query, ensuring more reliable outcomes.

Conclusion

Unified Multi-Model Data Handling

ArcNeural seamlessly integrates **structured**, **unstructured**, and **semi-structured** data, allowing businesses to work with a wide range of data types within a single platform.

AI-Optimized Performance

The combination of **vector search**, **graph-based HTAP**, and **LLM-enhanced analytics** makes ArcNeural ideal for businesses looking to gain **faster insights** and make **smarter decisions** in real time.

Tailored for Complex, Real-Time Applications

Whether it’s **criminal network detection**, **fraud analysis**, or **enterprise AI solutions**, ArcNeural provides the scalability, performance, and flexibility required to address today's most complex data challenges.

Q&A