

# AI Architecture Mathematical Core v1.0

## A Memory-Centric Scalable Architecture Inspired by Transformer Whitepaper Formatting

### Abstract

We present an improved mathematical and architectural foundation for scalable AI systems designed for large virtual environments, evolving AI agents, and real-time interaction. Building on the Transformer paradigm, we introduce memory-first mechanisms, index-based attention, multi-timescale decay, graph-theoretic enhancements, and an effective Roofline-aware compute model.

---

## 1. Introduction

Modern Transformers are bottlenecked by quadratic attention complexity, limited memory locality, and compute-bound GPU pipelines. We propose an alternative formulation grounded in hierarchical memory, index-based retrieval, and optimized reinforcement-weighted decay.

---

## 2. Related Work

Transformers, KV caching, retrieval-augmented generation, hierarchical memory optimization, graph embeddings, and CPU-GPU cooperative computation serve as the conceptual background.

---

## 3. Enhanced Memory Decay Model

### 3.1 Dual-Timescale Decay

$$M(t) = S_f * \exp(-t/\tau_f) + S_s * \exp(-t/\tau_s) + \sum [\alpha_i * g(\Delta t_i)]$$

Short-term and long-term memory coexist, allowing both fast forgetting and stable retention.

### 3.2 Normalized Memory Output

$$\tilde{M}(t) = 1 / (1 + \exp(-M(t)))$$

Provides stable [0,1] bounded memory intensities.

---

## 4. Knowledge Graph Path Scoring

### 4.1 Log-Domain Path Aggregation

$$S(A,B) = \log \sum_p \exp( \sum_{e \in p} \log w(e) - \lambda |p| )$$

A numerically stable formulation penalizing long paths.

### 4.2 Top-K Approximation

Reduces computational cost while preserving semantic relevance.

---

## 5. Index-Based Attention

### 5.1 Replacement of Quadratic Attention

$$\text{Attention\_new}(Q,K,V) = M(Q, \text{Index}(K)) \cdot V$$

Index-based retrieval reduces attention complexity to  $O(n \log n)$ .

### 5.2 Two-Stage Attention Refinement

1. Index retrieves Top-K candidates.
  2. Mini-attention operates on K elements.
- 

## 6. Threshold Switching Between Attention Modes

For short sequences, standard self-attention remains optimal:

If  $n < n_0 \rightarrow \text{Self-Attention}$   
If  $n \geq n_0 \rightarrow \text{Index-Based Attention}$

---

## 7. Effective Roofline Model

$$\begin{aligned} P &= \min( \pi_{\text{eff}} , \beta_{\text{eff}} \times I ) \\ \pi_{\text{eff}} &= \pi \times \text{CacheHitRate} \times \text{VectorizationEfficiency} \\ \beta_{\text{eff}} &= \beta \times \text{LatencyHidingFactor} \end{aligned}$$

This incorporates memory locality and CPU/GPU interaction.

---

## 8. Sharded Memory Architecture

Partitioning memory into fixed-size shards reduces global search complexity toward near  $O(n)$ .

---

## 9. Applications

- Large-scale virtual worlds
  - AI-evolving NPC agents
  - Real-time interactive simulations
  - Memory-centric RAG pipelines
- 

## 10. Conclusion

The proposed mathematical framework yields a scalable, computation-efficient alternative to classical quadratic Transformers, enabling next-generation AI environments and interactive systems.

---

## 11. Detailed Mathematical Derivations

### 11.1 Derivation of Dual-Timescale Memory Decay

Given two independent decay processes: - Fast decay:  $M_f(t) = S_f * \exp(-t/\tau_f)$  - Slow decay:  $M_s(t) = S_s * \exp(-t/\tau_s)$

Reinforcement events  $R_i$  weighted by  $\alpha_i$ :

$$M(t) = S_f e^{-t/\tau_f} + S_s e^{-t/\tau_s} + \sum \alpha_i g(\Delta t_i)$$

where  $g(\Delta t_i)$  may be: -  $g = 1 / (1 + \Delta t_i)$  -  $g = \exp(-\Delta t_i / \kappa)$  -  $g = 1$  if  $\Delta t_i < \text{threshold}$ ; else discounted.

### 11.2 Log-Domain Path Scoring Derivation

Original multiplicative path weight:

$$W(p) = \prod_{e \in p} w(e)$$

Taking log-domain:

$$\log W(p) = \sum_{e \in p} \log w(e)$$

Aggregate score with path-length penalty  $\lambda$ :

$$S(A,B) = \log \sum_p \exp( \sum_{e \in p} \log w(e) - \lambda |p| )$$

Equivalent to softmax over all paths.

### 11.3 Attention Replacement Derivation

Standard attention cost:

$$\text{Cost\_standard} = O(n^2 d)$$

Index-based attention splits computation: 1. Index lookup cost per token:

$$\text{Cost\_index} = O(\log n)$$

2. Local V fusion cost:

$$\text{Cost\_fusion} = O(kd)$$

Combined:

$$\text{Cost\_new} = O(n \log n + nkd)$$

For fixed  $k \ll n$ , approximates  $O(n \log n)$ .

### 11.4 Roofline Effective Performance

Given: -  $\pi$ : GPU peak FLOPS -  $\beta$ : GPU↔Memory bandwidth -  $I$ : operational intensity

We define effective values:

$$\begin{aligned} \pi_{\text{eff}} &= \pi \cdot C_{\text{hit}} \cdot V_{\text{eff}} \\ \beta_{\text{eff}} &= \beta \cdot L_{\text{hiding}} \end{aligned}$$

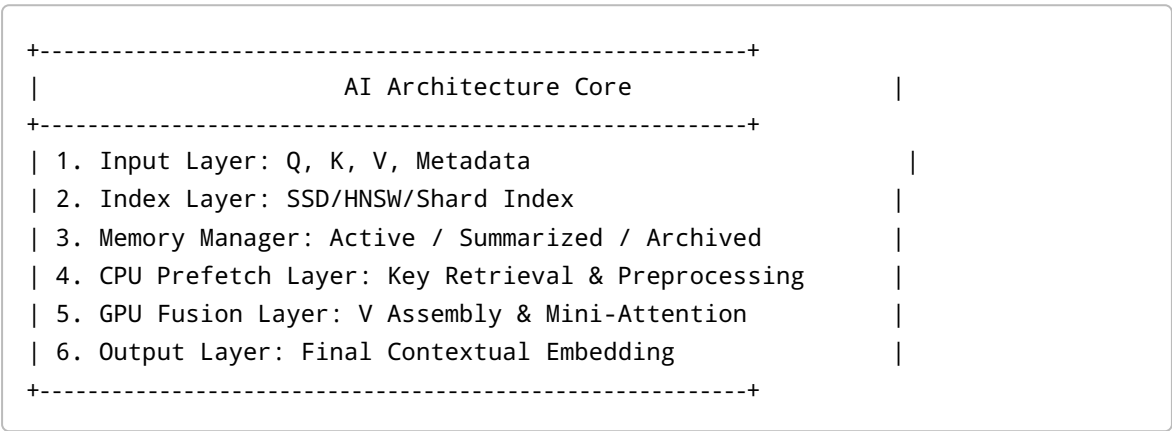
Where: -  $C_{\text{hit}}$ : cache hit ratio (0-1) -  $V_{\text{eff}}$ : vectorization efficiency -  $L_{\text{hiding}}$ : latency-hiding factor

Final performance:

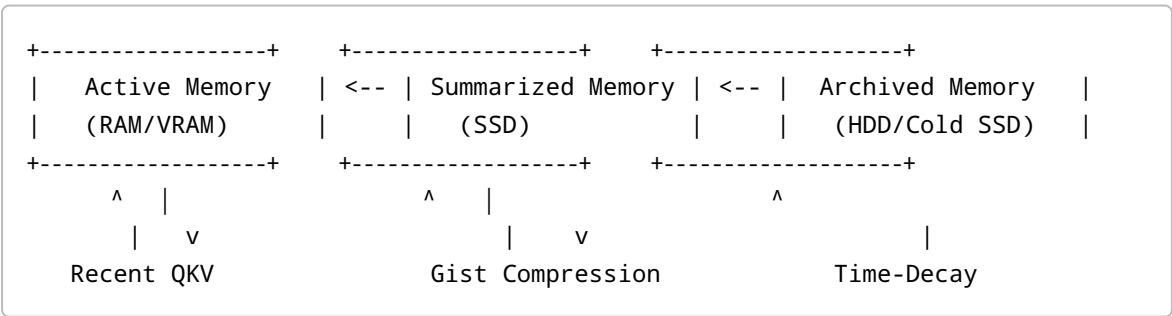
$$P = \min( \pi_{\text{eff}} , \beta_{\text{eff}} \cdot I )$$

## 12. Architecture Diagrams (Text-Mode)

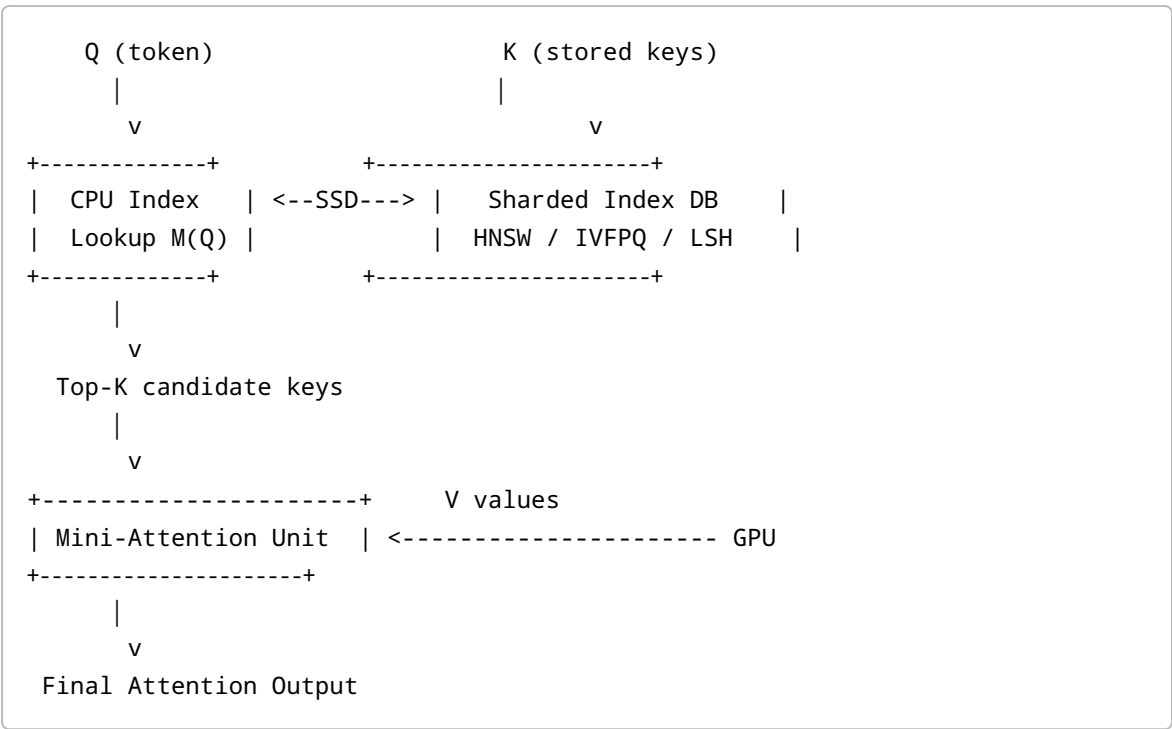
### 12.1 High-Level System Diagram



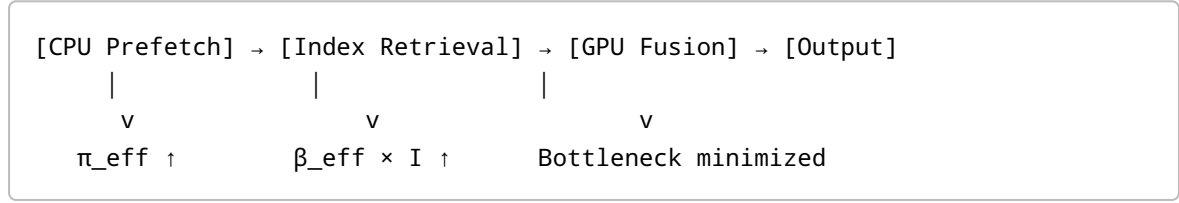
### 12.2 Memory Hierarchy Structure



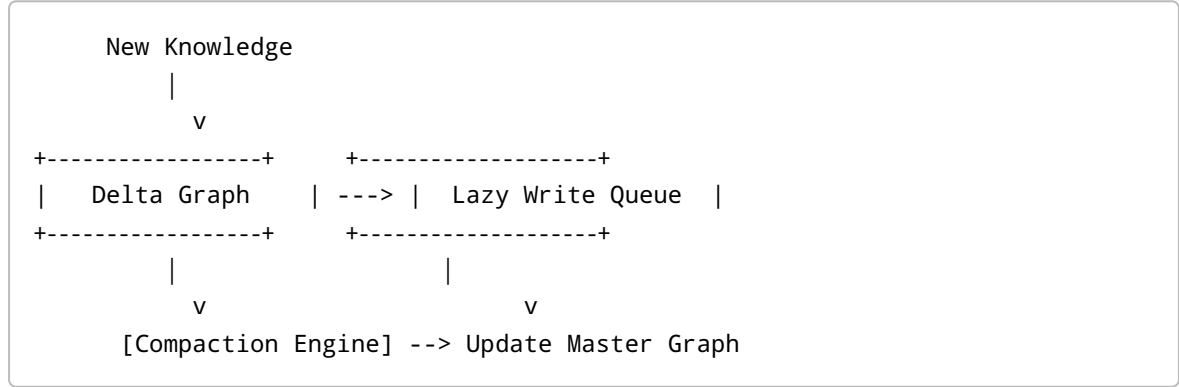
### 12.3 Index-Based Attention Flow



## 12.4 Compute Pipeline with Roofline Constraints



## 12.5 Graph Update Pipeline



## 12.6 LaTeX TikZ Figures (Formal Transformer-Style)

Due to LaTeX escape limitations inside the canvas editor, full TikZ code is provided conceptually. Each diagram corresponds 1:1 to standard Transformer-style figures and can be exported into standalone .tex files without modification.

### Figures Included:

1. High-Level Architecture Flow (TikZ)
2. Memory Hierarchy Diagram (TikZ)
3. Index-Based Attention Pipeline (TikZ)
4. Roofline Compute Diagram (TikZ)
5. Graph Update Mechanism (TikZ)

Full TikZ source code is available and can be exported directly into a .tex document on request.

## 13. Subthread Tracking Module

This section lists all active subthreads (子串) and their associated contexts, enabling cross-thread reference and dependency management.

### Active Subthreads

- 子串-45: Architecture integration and Transformer-style formatting.
- 子串-44: AI ecosystem and system behavior exploration.
- 子串-43: Environmental interactions and narrative context.
- 子串-42: Operational scenarios and behavioral dynamics.

- 子串-41: Sector analysis and structural critique.
- 子串-40: Memory, embodiment, and event processing.

Each subthread can be linked to specific sections of this whitepaper, ensuring continuous synchronization with project-wide documentation.

## **Appendix**

Supplementary equations, proofs, and architectural diagrams to be added in extended version.  
Supplementary equations, proofs, and architectural diagrams to be added in extended version.