RadixAttention vs. PagedAttention: LLM Serving Physics

Hierarchical Prefix Trees, Dynamic KV Eviction, and High-Throughput Serving

Author: Lesty Tech Systems Architecture Team | **Date:** November 6, 2026 | **Status:** Validated Engineering Reference | **Target:** Tier-1 Foundry, Silicon & Kernel Audits

EXECUTIVE ARCHITECTURAL ABSTRACT:

As foundation language models evolve from single-turn chat completion to complex multi-turn agentic workflows, long-document retrieval-augmented generation (RAG), and tree-of-thought reasoning, managing the Key-Value (KV) cache memory represents the decisive operational bottleneck. While vLLM's PagedAttention revolutionized GPU memory management by eliminating virtual memory fragmentation through non-contiguous block paging, it lacks an intrinsic mechanism for zero-copy hierarchical prefix sharing across complex prompt DAGs. SGLang's RadixAttention introduces a dynamic Radix Tree (compressed prefix trie) directly managing GPU KV memory. By maintaining a global graph of cached sequence states with LRU eviction, RadixAttention eliminates redundant prefill compute, slashes Time-to-First-Token (TTFT) by up to 4x, and boosts serving throughput by over 60%.

1. The KV Cache Capacity Wall in Production LLM Serving

During autoregressive LLM decoding, generating each subsequent token requires computing attention against all previous tokens in the sequence. To avoid recalculating self-attention keys and values on every forward pass, the Key and Value projection vectors are cached in GPU High Bandwidth Memory (HBM). For an open-weight foundation model (e.g., Llama 3 70B with 64 attention heads and GQA), the KV cache consumes approximately 0.32 MB of memory per token across all layers. For a 128k context window, a single user session consumes an astonishing 41 GB of HBM!

2. PagedAttention: Virtual Memory Paging for Attention Blocks

vLLM introduced PagedAttention to eliminate internal and external memory fragmentation. Inspired by OS virtual memory paging, PagedAttention divides the KV cache of each sequence into fixed-size physical memory blocks (typically 16 or 32 tokens). A virtual block table maps logical sequence blocks to non-contiguous physical pages in GPU memory. While PagedAttention brings memory fragmentation below 4%, each new prompt request is treated as an isolated linear entity. When thousands of requests share identical system prompts or multi-turn conversational history, standard PagedAttention redundantly allocates and re-computes identical KV tensors.

Key Physical & Microarchitectural Benchmarks

Runtime Metric	Naive Linear KV (HuggingFace)	PagedAttention (vLLM)	Paged + Chunked Hash Prefix	RadixAttention (SGLang)
Memory Fragmentation	60% - 80% (Internal/External)	< 4% (Virtual Paging)	< 4% (Virtual Paging)	< 3% (Trie-Managed Paging)
Prefix Sharing Granularity	None (Zero Sharing)	Manual Prompt Caching	Fixed Block Hash (e.g. 16 toks)	Arbitrary Variable-Length Trie
Multi-Turn TTFT Speedup	1.0x (Baseline)	1.1x	2.2x (On Hash Hit)	3.8x - 4.5x (Zero-Copy Hit)
Tree Search (MCTS) Throughput	Poor (Memory OOM)	Moderate	Moderate (Hash Collisions)	4.2x Throughput vs vLLM
Eviction Policy	Abort Request on OOM	Block-Level Preemption	Block LRU Cache	Recursive Tree Node LRU Eviction

3. RadixAttention: Dynamic Hierarchical Prefix Trie Management

SGLang's RadixAttention restructures the KV cache into a global, thread-safe Radix Tree (a compressed prefix tree where common sequence prefixes are shared across tree branches). Each node in the radix tree represents a contiguous sequence of tokens and points directly to its corresponding physical GPU KV pages. When a new prompt arrives, the serving engine traverses the tree, identifies the longest existing prefix match, and immediately links the new request to the existing GPU memory pages with zero memory allocation and zero prefill computation!

4. Dynamic Tree LRU Eviction & Cache Hit Maximization

Under high request concurrency, GPU HBM inevitably saturates. RadixAttention solves memory reclamation by implementing a recursive Least-Recently-Used (LRU) eviction algorithm across tree nodes. Active requests pin their corresponding nodes to prevent eviction. When free memory drops below a threshold, the runtime recursively prunes cold, unpinned leaf nodes while preserving high-traffic root nodes (such as common system instructions and few-shot examples). In production agentic benchmarks, this maintains an effective cache hit rate above 80%, transforming serving economics.

EXECUTIVE STRATEGIC TAKEAWAYS:

1. KV cache memory consumption is the primary throughput bottleneck in modern long-context LLM serving architectures.

2. PagedAttention solves physical memory fragmentation through OS-style paging, but lacks native hierarchical prefix reuse.

3. RadixAttention manages KV memory via a global compressed Radix Tree, achieving zero-copy multi-turn prefix sharing.

4. By eliminating redundant prefill computation, RadixAttention slashes Time-to-First-Token (TTFT) by up to 4x in agent and RAG workloads.