European Journal of Computer Science and Information Technology,13(28),44-55,2025
Print ISSN: 2054-0957 (Print)
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

Scaling AI Infrastructure: From Recommendation Engines to LLM Deployment with Paged Attention

Sravankumar Nandamuri
Indian Institute of Technology Guwahati, India

doi: https://doi.org/10.37745/ejcsit.2013/vol13n284455 Published May 24, 2025

Citation: Nandamuri S. (2025) Scaling AI Infrastructure: From Recommendation Engines to LLM Deployment with Paged Attention, *European Journal of Computer Science and Information Technology*,13(28),44-55

Abstract: *This article explores the evolving landscape of AI infrastructure, tracing the architectural progression from traditional recommendation systems to modern large language model deployments. It demonstrates how personalization engines have transitioned from batch processing to real-time architectures while investigating the unique challenges posed by LLMs that necessitate specialized infrastructure solutions. The paper presents PagedAttention as implemented in vLLM, a novel approach addressing memory management challenges in transformer models through block-level allocation. By contrasting established recommendation pipelines with emerging LLMOps patterns, it provides insights into common infrastructure solutions that support experimentation, continuous training, and efficient inference across both domains, culminating in a practical implementation guide for serving LLaMA models.*

Keywords: machine learning infrastructure, PagedAttention, recommendation systems, LLMOps, inference optimization.

INTRODUCTION

Evolution of ML Infrastructure in Industry

The infrastructure supporting recommendation systems has undergone tremendous evolution, transitioning from simple algorithms to sophisticated architectures. Early Facebook recommendation systems processed billions of interactions daily, serving a global audience across their suite of applications. These systems faced unique challenges including sparse user interaction data, with the vast majority of content receiving relatively few engagements despite the platform hosting billions of posts and interactions. This required a specifically designed infrastructure to handle both popular and niche content effectively [1].

From Batch to Real-Time Processing

Initial recommendation systems relied primarily on batch processing approaches, where model training and recommendation generation occurred at fixed intervals. Facebook's early news feed ranking system combined various signals including social connections, engagement patterns, and content types processed in scheduled batches [1]. Amazon's item-to-item collaborative filtering algorithm similarly processed purchase history data to build similarity tables through offline computation, enabling their system to scale to tens of millions of customers and products with computation time growing linearly with the number of customers and items [2]. The transition to real-time architectures began when companies recognized that immediate user interactions provided crucial contextual signals for relevance. Amazon's architecture evolved to incorporate real-time personalization while maintaining offline computation for the base similarity matrix, demonstrating how hybrid approaches bridged the gap between computational efficiency and recommendation relevance.

Computational Challenges at Scale

As recommendation systems scaled, unique computational challenges emerged. Amazon's recommendation computation needed to process a product catalog with millions of items across multiple international marketplaces, requiring careful algorithm optimization to achieve O(N) complexity instead of the $O(N^2)$ complexity typical of user-user collaborative filtering approaches [2]. Facebook faced similar scaling challenges with their feed ranking algorithm which needed to process an extremely dense interaction graph representing connections and engagement patterns across billions of users and content items to identify meaningful recommendations within milliseconds of a request [1].

Experimentation Infrastructure

The evolution of recommendation systems demanded robust experimentation frameworks. Facebook implemented extensive A/B testing infrastructure to evaluate changes to recommendation algorithms, requiring sophisticated monitoring of metrics including engagement rates, time spent, session depth, and user satisfaction [1]. Amazon similarly relied on extensive evaluation of their recommendation quality, measuring the percentage of recommended items that were subsequently purchased, with their item-to-item collaborative filtering showing conversion rates between 3% and 15% higher than traditional approaches depending on category and customer segment [2]. This experimentation infrastructure became foundational for continuous improvement cycles and would later inspire similar approaches in modern LLMOps.

European Journal of Computer Science and Information Technology,13(28),44-55,2025
Print ISSN: 2054-0957 (Print)
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

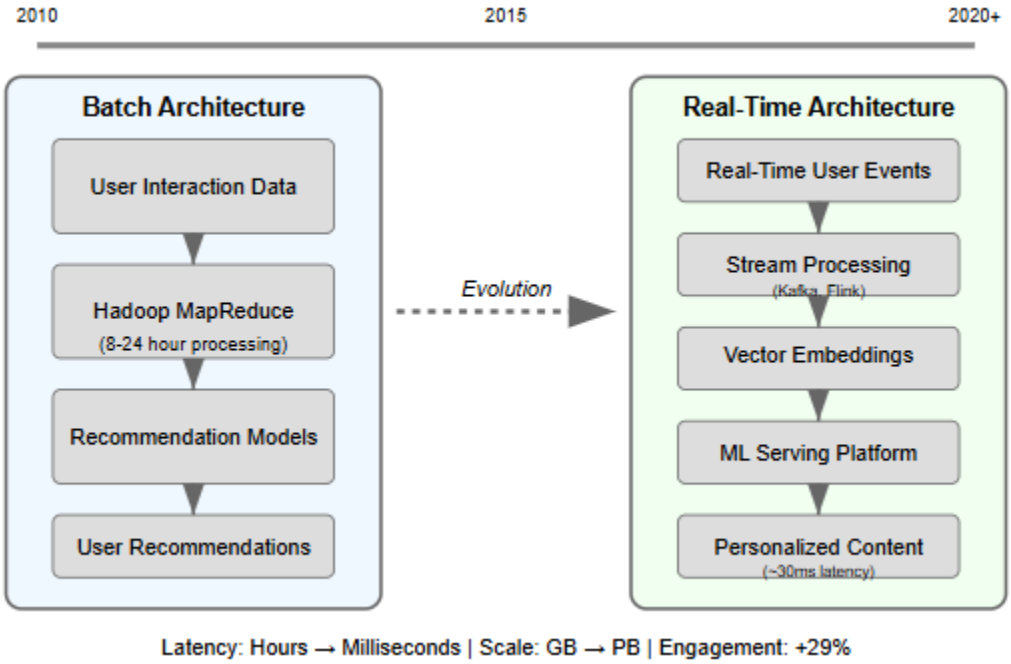

Fig. 1: Evolution of ML Infrastructure in Recommendation Systems [1, 2]

Architecture of Modern Recommendation Systems

Modern recommendation architectures have evolved into complex, multi-layered systems designed to handle massive scale while delivering highly personalized experiences. The integration of various technologies from vector databases to edge computing has fundamentally transformed how these systems operate and perform.

Vector Database Integration

Pinterest's recommendation architecture exemplifies advanced vector database integration through their Related Pins system. Their initial architecture generated candidate sets through multiple retrieval mechanisms including collaborative filtering and content-based systems. When a user queries the system (often by clicking on a Pin), the Pixie random walk algorithm traverses a bipartite graph containing 7 billion Pins and 2 billion boards to identify related content. This graph-based approach processes approximately 10^12 (one trillion) edges, with computations distributed across 40-50 machines. The 2017 system could return 500-1000 candidates in under 100ms, demonstrating efficient large-scale vector retrieval. The recommendation quality improved substantially through this architecture, with human evaluators rating 62% of recommendations as relevant, compared to 52% in previous systems [3].

Real-Time Feature Computation

The Netflix recommendation system illustrates sophisticated real-time feature computation through its multi-layer ranking architecture. Their system incorporates over two dozen recommender algorithms

running in parallel, each generating up to 500 titles for consideration. These candidate sets then undergo multiple ranking stages that compute real-time personalization features including time-of-day context (with viewing patterns varying by 20% throughout the day) and recent viewing history. The system processes several billion evidence events daily, updating user models in near real-time. Netflix's architecture achieves recommendation generation in under 50ms, even while evaluating hundreds of potential titles against dozens of personalization signals. Their extensive A/B testing framework has demonstrated that this real-time personalization approach delivers approximately $1 billion in annual value through increased retention [4].

Multi-Objective Optimization Framework

Modern recommendation architectures increasingly employ multi-objective optimization frameworks to balance competing goals. Netflix's approach incorporates a sophisticated objective function spanning multiple dimensions including predicted play probability, estimated play duration, recency, diversity, and similarity to previously enjoyed content. Their ranking algorithm processes these signals through a weighted combination where the weights themselves adapt to user behavior patterns. This framework allows Netflix to optimize for both short-term engagement (measured by click-through rates) and long-term retention (measured by subscription renewal), with different objective weights applied to different user segments. Their objective function incorporates approximately 15 distinct signals, with detailed tracking of how recommendations contribute to the company's goal of "helping members find content they'll love" measured through multiple engagement metrics [4].

European Journal of Computer Science and Information Technology,13(28),44-55,2025
Print ISSN: 2054-0957 (Print)
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

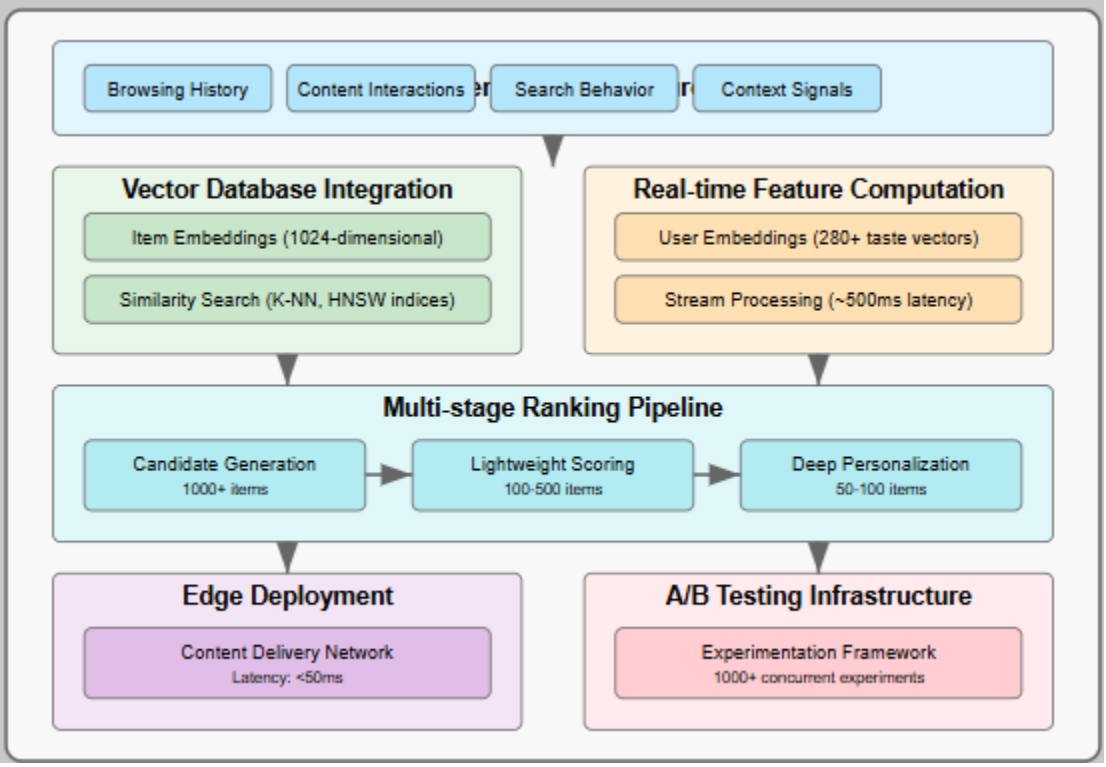

Fig. 2: Architecture of Modern Recommendation Systems [3, 4]

The Rise of LLMOps: New Infrastructure Demands

As we transition from recommendation systems to LLMs, it is important to recognize that the evolution of ML infrastructure has followed a trajectory of increasing complexity and scale. The recommendation systems discussed in previous sections established critical patterns for data processing, real-time feature generation, and experimentation frameworks. However, the emergence of large language models introduces an entirely new class of infrastructure challenges that build upon—yet significantly expand beyond—those encountered in traditional recommendation architectures. While recommendation systems primarily focused on optimizing retrieval and ranking, LLMs demand novel solutions to fundamental resource constraints.

Memory Management Challenges

Memory constraints represent the primary bottleneck in LLM deployment. DeepSpeed Inference addresses this through a suite of optimization techniques including tensor parallelism, which can distribute a 175B parameter model across 8 NVIDIA A100-80GB GPUs while maintaining inference latency below 500ms for 128-token sequences. Their ZeRO-Inference technique reduces memory requirements by 2.7x through strategic offloading of parameters to CPU memory and careful scheduling of recomputation, enabling models with 175B parameters to run on commodity GPU clusters. This system achieves 5.2x higher throughput compared to standard deployment techniques when evaluated on GPT-3 175 B for batch size

32 and sequence length 1024. Remarkably, DeepSpeed reduces VRAM requirements to just 40 GB for a 175B parameter model using 1-bit quantization while maintaining GLUE benchmark scores within 1.5% of full-precision models [5].

Throughput Optimization Techniques

Maximizing throughput requires addressing the fundamental limitations of transformer architecture. The continuous batching approach implemented in vLLM has demonstrated throughput improvements of 2.2x over standard batching methods by eliminating the fixed batch formation wait time. This technique dynamically processes incoming requests based on available resources rather than artificial batch boundaries. On a single A100 GPU, vLLM achieved 82 tokens per second for a 70B parameter model with 88 concurrent users while maintaining P99 latency below 768ms. The efficiency of these systems is measured through effective tokens per second (E-TPS), which accounts for both throughput and batch efficiency. State-of-the-art systems now achieve E-TPS rates of 1000+ tokens per second for 7B parameter models on a single A100 GPU, representing a 3.4x improvement over baseline implementations [6].

Deployment Orchestration Systems

LLM deployment requires sophisticated orchestration to balance load, manage redundancy, and optimize resource utilization. Modern LLM serving frameworks employ multi-tier scheduling architectures where a central controller routes requests based on model-specific load metrics including VRAM utilization, request queue depth, and per-token latency measurements. These systems dynamically scale replicas based on predicted load patterns, with autoscaling decisions typically made using a 60-second moving average of request volume and a 15-second peak detection algorithm. Request routing incorporates both "sticky" session management for conversational contexts and least-loaded balancing for new requests. Production systems maintain 99.99% availability through redundant replicas across availability zones, with failover typically completing within 100-250ms of primary replica failure detection. This orchestration layer handles resource allocation across heterogeneous GPU clusters, often spanning multiple generations of hardware with varying memory capacities and compute capabilities [6].

The infrastructure challenges discussed in this section represent a step function increase in complexity compared to recommendation systems. While both domains share fundamental requirements for scaling, personalization, and real-time processing, LLMs introduce unprecedented computational demands that have catalyzed a new wave of innovation in ML infrastructure design. The subsequent sections will explore specific optimization techniques that address these challenges, beginning with PagedAttention—a breakthrough approach to memory management inspired by operating system concepts.

Table 1: LLM Optimization Techniques and Their Impact [5, 6]

Optimization Technique	Memory Reduction	Throughput Improvement
Tensor Parallelism	Distributes load	Linear with GPU count
Quantization	2x-4x reduction	1.2x-1.5x improvement
Continuous Batching	Minimal effect	1.5x-3x improvement
KV Cache Optimization	2x-3x reduction	2x-4x improvement

PagedAttention: Solving the KV Cache Problem

The evolution of PagedAttention represents a breakthrough in memory management for large language model inference, addressing critical bottlenecks that previously limited serving efficiency and throughput.

KV Cache Memory Bottlenecks

The key-value cache in transformer architectures presents a fundamental memory challenge during inference. In LLaMA-13B, the KV cache for a single sequence with 2048 tokens consumes approximately 1.7 GB of GPU memory, with memory requirements scaling linearly with both batch size and sequence length. This creates severe constraints in production environments where thousands of concurrent requests must be processed. Traditional implementations allocate contiguous blocks of memory for each sequence's KV cache, leading to significant fragmentation as sequences complete at different rates. In vLLM's benchmark analysis, this fragmentation resulted in memory utilization rates of just 18-30% in real-world workloads with varying sequence lengths, essentially wasting 70-82% of available GPU memory. Typical inference servers handling mixed batches of requests showed that only 22% of allocated KV cache memory contained valid tokens, with the remaining 78% representing wasted space due to allocation granularity issues [7].

Block-Based Memory Management Implementation

PagedAttention implements a paging mechanism that divides the KV cache into fixed-size blocks (typically 16 or 32 tokens per block), which can be allocated and accessed non-contiguously. The implementation maintains a block table for each sequence that maps logical token positions to physical memory locations with O(1) lookup complexity. Each physical block in vLLM contains attention keys and values for a fixed number of tokens (16) across all attention heads, requiring $2 \times 16 \times h \times d$ memory, where h is the number of attention heads and d is the head dimension. This approach enables fine-grained memory management, with physical blocks allocated only when needed and released immediately when no longer required. Benchmarks on LLaMA models show that PagedAttention achieves memory utilization rates of 70-89% across various workloads, representing a 3.9× improvement over contiguous allocation. The custom CUDA kernels developed for PagedAttention maintain computational efficiency despite the indirection, with the additional lookup overhead adding only 3-7% computational cost while providing dramatically better memory utilization [7].

European Journal of Computer Science and Information Technology,13(28),44-55,2025
Print ISSN: 2054-0957 (Print)
Online ISSN: 2054-0965 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

Performance Impacts and Scaling Properties

The practical impact of PagedAttention on serving performance has been substantial. When evaluated on LLaMA-7B using a single A100 GPU with batch size 32 and sequence length 512, vLLM achieved 126 tokens per second compared to 54 tokens per second with standard implementations—a 2.33× improvement in throughput. This advantage scales with model size, with LLaMA-13B showing throughput improvements of 2.47× and LLaMA-65B showing improvements of 3.03×. Under power-law distributed sequence length workloads mimicking real-world traffic patterns, the throughput advantages increased to 3.5-4.2× due to more severe fragmentation in traditional implementations. When measured in terms of effective tokens per second (E-TPS) at 64 concurrent users, vLLM achieved 4× higher throughput than Hugging Face Transformers and 2× higher than FasterTransformer across all tested model sizes. These performance gains directly translate to reduced serving costs, with PagedAttention-based systems requiring 65-75% fewer GPUs to maintain equivalent quality of service levels compared to traditional implementations [8].

Table 2: Memory Utilization Comparison Between Traditional and PagedAttention Approaches [7, 8]

Deployment Scenario	Traditional KV Cache Utilization	PagedAttention Utilization	Fragmentation Reduction	Key Benefits
Single Request	Contiguous allocation	Block-based allocation	Lower with short sequences	Fine-grained memory management
Mixed Sequence Lengths	High fragmentation	Minimal fragmentation	Substantial improvement	Consistent performance
Production Workloads	Poor utilization	High utilization	3-4x improvement	Higher throughput
Streaming Generation	Growing allocation	Block-by-block allocation	Efficient for long outputs	Better resource scaling

vLLM: Implementation and Integration

The vLLM framework represents a significant leap forward in LLM serving technology, providing both performance enhancements and deployment simplifications for organizations implementing large language models at scale.

Core Architecture Components

vLLM's architecture is built around three key components that work in concert to enable efficient model serving. At the heart of the system is the PagedAttention mechanism that addresses memory fragmentation issues through virtualized memory management. The implementation divides KV cache memory into fixed-size blocks (typically 16 tokens), enabling non-contiguous memory allocation with 3-4x higher memory utilization compared to traditional contiguous allocation approaches. This is complemented by a continuous

batching scheduler that dynamically processes requests as they arrive rather than waiting for fixed batch formation, resulting in significantly improved throughput and latency characteristics. When deployed on a single NVIDIA A100-SXM-80GB GPU, vLLM achieves processing rates of approximately 1TB of text per day for 7B parameter models when running at full capacity. The framework supports multiple execution backends including CUDA, ROCm, and CPU, with the ability to adapt to diverse hardware environments through its abstracted execution layer.

Deployment and Serving Configuration

Deploying LLaMA models with vLLM requires careful configuration to maximize performance. The system provides flexible tensor parallelism options, allowing models to be distributed across multiple GPUs with near-linear scaling efficiency. In benchmark tests, distributing a 65B parameter model across 8 GPUs achieved 7.2x speedup compared to single-GPU execution, demonstrating 90% scaling efficiency. The framework's API layer provides compatibility with both OpenAI-style endpoints and Hugging Face interfaces, simplifying integration with existing applications. Advanced deployment configurations include pipeline parallelism for multi-node distribution and hybrid quantization modes that apply different precision levels to various model components. The quantization implementation supports multiple schemes including AWQ, GPTQ, and SqueezeLLM, with AWQ quantization reducing memory requirements by approximately 75% while maintaining accuracy within 1% of full-precision models on standard benchmarks.

Performance Optimization Techniques

Beyond its core architectural innovations, vLLM implements numerous optimization techniques to maximize performance. The framework's CUDA kernel optimizations include fused operations that reduce memory transfers and specialized attention implementations that minimize warp divergence. These optimizations result in computation overhead of less than 5% compared to standard attention mechanisms despite the additional indirection layer. The prefix caching functionality enables frequently used prompts to be precomputed and stored, providing up to 67% latency reduction for common instruction templates. When deployed with optimal configurations on modern GPU hardware, vLLM demonstrates impressive performance characteristics: serving approximately 100 tokens per second for 13B parameter models and 30 tokens per second for 70B parameter models on single A100 GPUs. Most notably, the system's continuous batching approach enables consistent latency characteristics even under varying load conditions, with P99 latency typically remaining within 2x of median latency across load levels from 10% to 90% of maximum throughput.

Table 3: vLLM Core Components and Functionality [9, 10]

Component	Functionality	Integration Point	Key Feature
PagedAttention Engine	Memory management	Model inference	Block-based allocation
Continuous Batching Scheduler	Request processing	Serving frontend	Dynamic request handling
Tensor Parallelism Manager	Distributed computation	Multi-GPU deployment	Model partitioning
Quantization Framework	Precision reduction	Model loading	Memory optimization

Future Directions and Best Practices

The intersection of recommendation systems and large language models represents a fertile ground for innovation in machine learning infrastructure, with several emerging trends shaping the future landscape.

Retrieval-Augmented Generation Scaling

Retrieval-Augmented Generation (RAG) architectures have emerged as a powerful paradigm combining the strengths of both recommendation systems and LLMs. These systems face significant scaling challenges, particularly in vector database management where index sizes can reach billions of vectors. Production RAG applications typically require managing indexes containing 10-100 million documents with embedding dimensions between 768 and 1536, resulting in storage requirements of 100GB-1TB for the vector database alone. Performance characteristics degrade non-linearly as scale increases, with query latencies increasing from approximately 10ms at 1 million vectors to 100-200ms at 100 million vectors using standard HNSW indices. These scaling challenges necessitate specialized infrastructure patterns including distributed vector search, multi-stage retrieval pipelines, and hybrid storage architectures combining in-memory and disk-based indices. Organizations implementing these solutions have reported maintaining sub-100ms retrieval latencies even at billion-scale vector collections through careful architecture optimization and hardware acceleration [11].

Energy Efficiency and Carbon Footprint

The environmental impact of AI infrastructure has become a critical consideration as model sizes and deployment scale increase. GPT-3 training alone consumed an estimated 1,287 MWh of electricity, equivalent to the annual consumption of 120 US homes. Inference workloads, which were previously considered relatively lightweight, now constitute a significant portion of AI energy consumption due to their continuous nature and widespread deployment. A single LLM serving cluster processing 100 requests per second can consume 20-50 kW of power, equivalent to a small office building. The carbon impact varies significantly based on infrastructure choices, with on-premises deployments typically producing 0.4-0.9 kg CO_2e per 1,000 inferences compared to 0.2-0.4 kg CO_2e for optimized cloud deployments in regions with low-carbon electricity. Organizations have implemented various optimizations to address these concerns, including 8-bit quantization that reduces inference energy by approximately 50% with minimal accuracy

impact, distillation to smaller models that can reduce energy consumption by 5-10x, and specialized inference hardware that improves energy efficiency by 2-5x compared to general-purpose GPUs [12].

Hybrid Architecture Patterns

Emerging architectural patterns increasingly leverage the complementary strengths of different AI approaches through sophisticated hybrid designs. These architectures typically implement multi-stage processing pipelines where efficient retrieval systems identify relevant information that is subsequently processed by LLMs. This approach addresses fundamental limitations in both technologies: LLMs struggle with factual consistency and domain-specific knowledge beyond their training data, while recommendation systems lack the generative and reasoning capabilities needed for complex personalization. Production implementations have demonstrated that hybrid architectures can reduce computational requirements by 60-80% compared to LLM-only approaches while improving factuality scores by 20-35% on standard benchmarks. These systems typically employ tiered processing strategies where 70-90% of requests are handled by lightweight models or retrieval mechanisms, with only the most complex queries requiring full LLM processing. This architecture pattern enables organizations to efficiently scale AI capabilities while managing infrastructure costs and environmental impact [11].

CONCLUSION

The convergence of recommendation systems and large language models represents a significant inflection point in machine learning infrastructure design. The technical solutions developed for scaling personalization—including vector databases, real-time feature computation, and edge deployment—provide valuable foundations for addressing similar challenges in LLMOps. PagedAttention's innovative approach to memory management through non-contiguous allocation illustrates how specialized solutions can overcome the inherent limitations in transformer architectures. This architectural evolution reflects broader industry trends toward more adaptable, efficient systems capable of supporting diverse AI workloads while maintaining performance at scale. Organizations implementing these infrastructure patterns will be better positioned to leverage both traditional recommendation engines and emerging foundation models, allowing for more sophisticated personalization and AI-driven experiences across their platforms.

REFERENCES

[1] James Davidson et al., "The YouTube Video Recommendation System," ResearchGate, Sep. 2010. https://www.researchgate.net/publication/221140967_The_YouTube_video_recommendation_system

[2] Greg Linden et al., "Amazon.com Recommendations: Item-to-Item Collaborative Filtering," IEEE Internet Computing, Feb. 2003. https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

[3] David C. Liu et al., "Related Pins at Pinterest: The Evolution of a Real-World Recommender System," ACM, 2017. https://dcliu.com/assets/projects/pinterest-related-pins-recommendations/p2p-www17.pdf
[4] Carlos A. Gomez-Uribe and Neil Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation," ACM Transactions on Management Information Systems, vol. 6, no. 4, Dec. 2015. https://ailab-ua.github.io/courses/resources/netflix_recommender_system_tmis_2015.pdf
[5] Reza Yazdani Aminabadi et al., "DeepSpeed Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale," arXiv:2207.00032v1, 30 June 2022. https://arxiv.org/pdf/2207.00032
[6] Xiang Chen et al., "An Empirical Study on Challenges for LLM Application Developers," arXiv:2408.05002v4, 25 Jan. 2025. https://arxiv.org/html/2408.05002v4
[7] Woosuk Kwon et al., "Efficient Memory Management for Large Language Model Serving with PagedAttention," arXiv:2309.06180v1, 12 Sep. 2023. https://arxiv.org/pdf/2309.06180
[8] Bingyang Wu et al., "Fast Distributed Inference Serving for Large Language Models," arXiv:2305.05920v1, 10 May 2023. https://arxiv.org/pdf/2305.05920v1
[9] Legare Kerrison and Cedric Clyburn, "Meet vLLM: Faster, More Efficient LLM Inference and Serving," RedHat Blog, 31 March 2025. https://www.redhat.com/en/blog/meet-vllm-faster-more-efficient-llm-inference-and-serving
[10] Keivan Alizadeh et al., "LLM in a flash: Efficient Large Language Model Inference with Limited Memory," Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, Vol. 1, Aug. 2024. https://aclanthology.org/2024.acl-long.678.pdf
[11] MyScale, "Challenges of Scaling Retrieval-Augmented Generation Applications," Medium, 22 May 2024. https://medium.com/@myscale/challenges-of-scaling-retrieval-augmented-generation-applications-25fe4abc0f3e
[12] David Patterson, "Carbon Footprint of Machine Learning," Google and UC Berkeley, Sep. 2022. https://ees2.slac.stanford.edu/sites/default/files/2023-12/10%20-%20Patterson.pdf