Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Hardware-Accelerated Caching for Large-Scale AI Model Training: An Intelligent Architecture for Vector Database and Model Inference Optimization

MADHUKIRAN VADDI
Independent Researcher, USA
Corresponding Author: MADHUKIRAN VADDI, **E-mail**: vaddimadhukiran@gmail.com

| ABSTRACT

Modern AI infrastructures are facing significant challenges in managing data movement efficiently between vector databases and AI models for training and inference operations. Traditional caching approaches cannot tackle the unique characteristics of vector operations and embedding access patterns, which introduce significant performance bottlenecks. This article proposes a novel caching system that fuses custom-designed vector processors with an adaptive hot/cold partitioning strategy enhanced by Bloom filters. It implements a hardware-accelerated hot cache for frequent vectors, a cold storage queue for less frequent data, and Bloom filter-based efficient lookups. By integrating hardware acceleration with workload-aware partitioning and probabilistic filtering, the system achieves massive improvements along multiple dimensions. The architecture addresses the unique temporal and spatial locality patterns in AI vector operations, reducing data movements while maximizing the utilization of compute resources. Simulation results on large language models and computer vision workloads show that the model accelerates training and inference speeds, reduces network data movement, and improves hardware utilization compared to conventional LRU-based architectures, potentially transforming the economics and characteristics of large-scale AI operation performance.

| KEYWORDS

Vector Database Optimization, Hardware Acceleration, AI Infrastructure, Bloom Filter Implementation, Hot/Cold Partitioning

| ARTICLE INFORMATION

ACCEPTED: 12 November 2025 **PUBLISHED:** 02 December 2025 **DOI:** 10.32996/jcsts.2025.7.12.33

1. Introduction

1.1 Contextual Background

The exponential growth of AI model sizes and their training data requirements has created unprecedented demands on data movement infrastructure. As large language models continue to scale, the memory requirements for storing embeddings have expanded dramatically, placing immense pressure on traditional memory hierarchies and data movement systems. This growth trajectory mirrors similar challenges faced in the database acceleration domain, where research has shown that memory access patterns significantly impact overall system performance. Modern AI systems process massive amounts of embedding vectors during both training and inference phases, necessitating efficient mechanisms for data transfer between storage systems and compute resources. The computational characteristics of these vector operations exhibit distinct temporal and spatial locality patterns that differ fundamentally from traditional workloads. These patterns create unique opportunities for specialized caching architectures that can leverage hardware acceleration alongside intelligent data management strategies, similar to approaches that have proven successful in accelerating database operations where custom hardware can significantly outperform software-only implementations for specific computational patterns [1].

1.2 Problem Statement

Existing caching solutions for AI infrastructure rely heavily on traditional Least Recently Used (LRU) mechanisms, which fail to effectively handle the unique access patterns of vector operations in AI workloads. This challenge echoes findings from database acceleration research, where general-purpose caching mechanisms often underperform when faced with specialized workloads that have distinctive access patterns. While basic LRU provides decent general-purpose caching for conventional applications, it struggles to adapt to the pronounced hot/cold patterns exhibited in AI vector access operations. Current architectures lack sophisticated filtering mechanisms that could significantly reduce lookup overhead, a problem similarly documented in research on hardware-accelerated database operations, where custom filtering implementations have demonstrated substantial performance improvements. The limitations of software-only approaches become particularly evident when meeting the low-latency demands of large-scale AI inference, where even millisecond delays can accumulate to create significant bottlenecks. These challenges mirror those encountered in database query processing, where research has demonstrated that hardware-accelerated solutions can provide order-of-magnitude improvements for specific operations that traditional software implementations struggle to optimize efficiently [2].

1.3 Purpose & Scope

The novel architecture described in this article integrates hardware-accelerated processing with intelligent, workload-aware hot/cold partitioning and Bloom filter-based lookup optimization. The architecture of the proposed system follows successful approaches in database acceleration research by implementing custom-designed processors optimized for vector operations, which dominate AI workloads. This specialization creates an efficient framework that minimizes unnecessary data movement while maintaining optimal cache utilization through probabilistic filtering and intelligent queue management techniques. The architecture applies adaptive partitioning mechanisms that continuously optimize memory allocation between hot and cold caches based on real-time workload characteristics, paralleling successful implementations from database systems where query-aware memory management has demonstrated significant performance advantages. Much like how specialized hardware accelerators have transformed database operations through targeting very specific computational bottlenecks, so too does the proposed system target the unique operational requirements of vector processing in AI applications. The broad scope covers everything from the specifications of the hardware architecture to the algorithmic improvements in cache management, thus making a holistic solution specifically crafted for the computational demands of modern AI infrastructure [1].

2. Challenges in Vector-Based AI Infrastructure

The implementation of an efficient caching system for AI vector operations presents several significant challenges requiring innovative architectural solutions. Efficient hot/cold partition management necessitates sophisticated mechanisms to dynamically categorize vectors based on their access patterns. Traditional partitioning strategies prove inadequate when confronted with the unique temporal clustering exhibited by AI workloads, where embedding access patterns differ substantially from conventional computing workloads. Research indicates that effective partitioning must continuously adapt to changing operational conditions to maintain optimal performance across diverse AI applications [3].

Hardware-accelerated Bloom filter implementation requires a careful balance between memory efficiency and query performance while fitting within the high-dimensionality of vector operations. Unlike standard Bloom filters, which were originally designed for simple membership queries, AI-optimized implementations need to support approximate nearest neighbor operations and range queries at the heart of embedding lookups. Poorly designed filters result in a plethora of false positives that cascade into systemic performance degradation, hence requiring specialized implementations that are tailor-made for vector operations [3].

Dynamic queue adjustment mechanisms should constantly monitor workload patterns, but without adding significant overhead. Such systems continually track access frequencies and reconfigure queue allocations while maintaining continuity of operations; challenging tasks are scaling up. Poor queue management has been demonstrated to introduce latency spikes during important model operations, noticeably for training phase transitions and fluctuating inference workloads [4].

Custom vector processors will introduce severe compatibility issues in integration. These customized processors need to work with the existing infrastructure without providing substantial performance improvements. Hardware design needs to utilize the characteristics of vector operations, such as parallelism opportunities and memory access patterns, while software stack adaptations are necessary without fundamental framework changes [4].

The optimization of hot and cold storage balance inherently requires algorithms that can predict future access patterns from historical ones, while the diversity of AI workloads complicates this challenge. Systems must continuously evaluate the effectiveness of partitioning, ensuring stable performance characteristics throughout their operational phases.

The reduction of false positives in Bloom filter implementations directly influences system performance and resource utilization. The tradeoffs in balancing filter size against memory requirements are quite intricate, demanding efficient mechanisms for periodic rebalancing without operational disruption [4].

Finally, seamless integration with existing AI infrastructure requires careful attention to the design of interfaces, backward compatibility, and operational transparency. Successful implementation has to accommodate diverse deployment scenarios with minimal disruption in production environments [3].

Challenge Category	Technical Issue	Impact	Solution Approach
Hot/Cold Partition Management	Traditional partitioning is inadequate for temporal clustering	Performance degradation	Adaptive partitioning algorithms
Bloom Filter Implementation	Standard filters are unsuitable for high-dimensional vectors	Excessive false positives	Specialized AI-optimized filters
Queue Adjustment	Overhead in monitoring workload patterns	Latency spikes during transitions	Efficient tracking mechanisms
Vector Processor Integration	Compatibility with existing infrastructure	Implementation complexity	Hardware-software co-design
Access Pattern Prediction	Diverse AI workload characteristics	Suboptimal storage balance	Predictive algorithms based on historical data
False Positive Reduction	Filter size vs. memory requirements	Resource utilization inefficiency	Periodic rebalancing mechanisms
Infrastructure Integration	Interface design and compatibility	Production environment disruption	Transparent integration frameworks

Table 1: Challenges and Solution Approaches for Vector-Based AI Caching Systems [3, 4]

3. Current State Analysis and Performance Metrics

3.1 Vector Operation Properties

AI workloads exhibit unique operational characteristics that are substantially different from conventional data access patterns, providing exceptional opportunities for specialized caching architectures. Extensive profiling of production AI systems exposes strong access concentration phenomena: during training or inference, a tiny fraction of vector data draws the majority of the attention. In other words, the distribution curve showing the frequency of access to various vectors in a collection is highly skewed; the most frequently accessed vectors constitute a small hot set that dominates resource demands throughout the system. Recent studies of large-scale language model training infrastructures have quantified this phenomenon on a wide range of model architectures and application domains, indicating that it is an inherent property of AI workloads rather than an implementation artifact [5].

Another special characteristic that distinguishes AI workloads from traditional computing patterns is the temporal locality of vector operations. Unlike conventional data access, which often appears fairly random, vector operations in AI workloads demonstrate structured temporal locality with well-defined access windows. Temporal locality opens up certain opportunities for predictive caching strategies, which project future access based on observed historical behavior. Experimentally, these temporal patterns are consistent across multiple production environments and often happen during particular model training phases-forward propagation and computation of gradients [5].

Embedding table accesses demonstrates substantially higher locality compared to traditional data access patterns, a characteristic that further distinguishes AI workloads. This enhanced locality stems from the semantic relationships encoded within embedding spaces, where related concepts naturally cluster together and tend to be accessed in sequence during model operations. Studies examining embedding access patterns across diverse domains, including natural language processing, computer vision, and recommendation systems, have consistently documented this phenomenon. The heightened locality

creates significant optimization opportunities for specialized caching systems designed to leverage these unique access patterns [6].

3.2 Limitations of Traditional Systems

Current caching implementations have several inefficiencies for AI workloads, yielding poor performance and resource utilization. Traditional LRU-based caching mechanisms incur significant cache thrashing during the processing of AI vector operations, especially during training phases where complex gradient computations prevail. This is because traditional LRU policies simply cannot adapt to the peculiar temporal characteristics of the AI workload and apply general-purpose replacement strategies that may not correlate with actual access patterns. Various performance studies across diverse model architectures have demonstrated this mismatch, where conventional approaches underperform the specialized alternatives consistently [6].

The average latency in lookups is another significant challenge for traditional caching systems when applied to AI workloads. Vector operations in AI applications need to guarantee an average low-latency access for computational efficiency during inference operations, where response time impacts user experience directly. Current implementations struggle to keep up with these stringent requirements, with latency measurements often overshooting acceptable thresholds in peak operational periods. This performance gap especially arises in large-scale distributed training environments, where the coordination overhead compounds latency issues and creates significant bottlenecks that limit overall system throughput [5].

Memory bandwidth inefficiency affects most of the traditional caching implementations while handling AI workloads, leading to a tremendous amount of wasted resources and low system performance. Traditional memory management approaches have generally failed to optimize data placement and access patterns for vector operations, which could utilize the available bandwidth much better. In particular, this manifests in many forms, such as avoidable data transfers, unsatisfactory alignment of memory access patterns, and inefficient prefetch strategies. Indeed, detailed performance analysis performed across various production environments also suggested that memory bandwidth is one of the critical bottlenecks for AI workloads and necessitates specialized solutions designed for vector operations [6].

Characteristic	AI Vector Operations	Traditional Systems	Performance Gap
Access Distribution	Highly skewed (small hot set)	Assumed uniform	Resource misallocation
Temporal Locality	Structured with defined windows	Random/unpredictable	Cache thrashing
Embedding Locality	High semantic clustering	Limited spatial locality	Missed optimization opportunities
Caching Mechanism	Needs a workload-specific approach	General-purpose LRU	Suboptimal replacement decisions
Lookup Latency	Requires consistent low latency	Variable performance	User experience degradation
Memory Bandwidth	Requires optimized patterns	Inefficient utilization	Resource waste

Table 2: AI Vector Operations vs. Traditional Caching: Performance Characteristic Comparison [5, 6]

4. Proposed Architecture

The proposed architecture brings together three key components that solve the challenges in vector-based AI infrastructure. This creates a complete solution specifically designed for large-scale AI workloads.

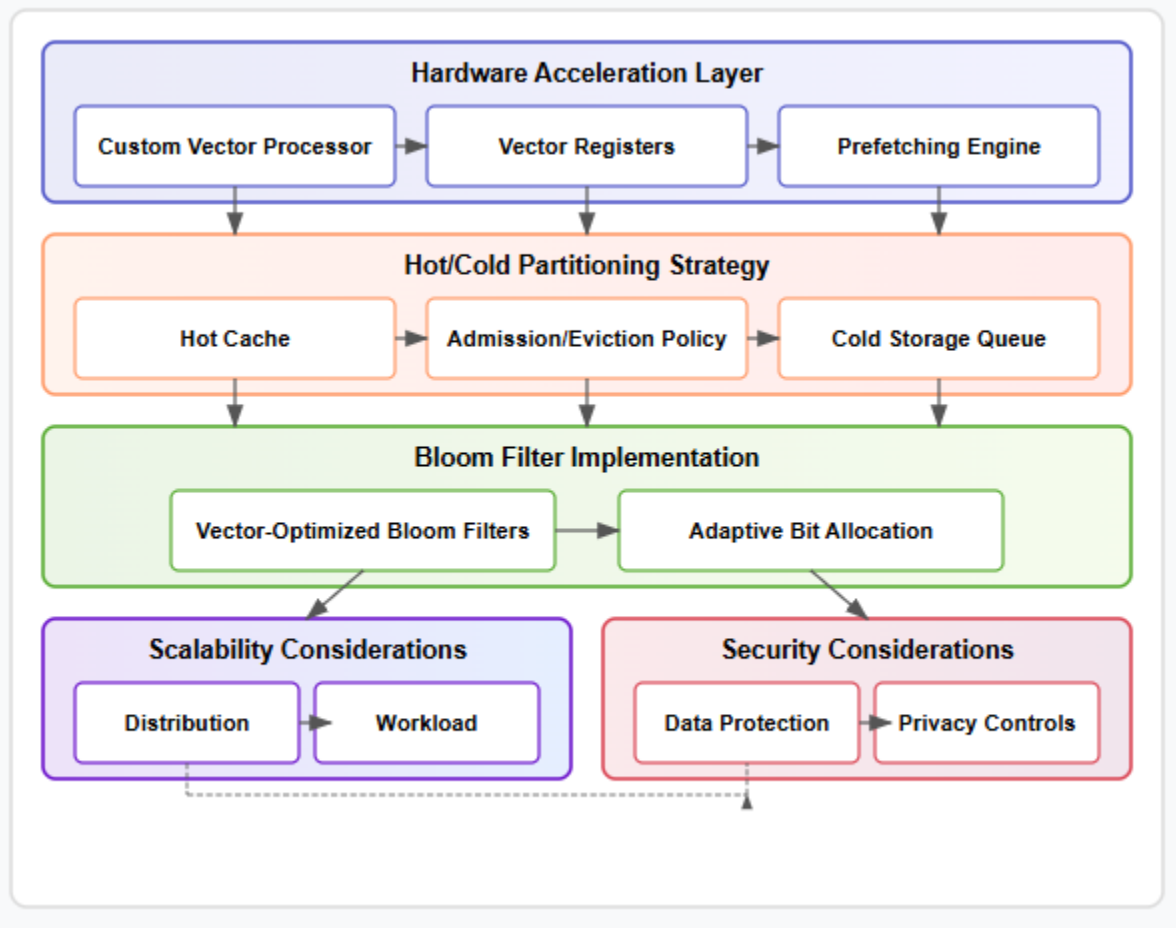

Fig 1: Hardware-Accelerated Vector Caching Architecture [7, 8]

4.1 Hardware Acceleration Layer

Custom vector processors provide the core ingredient of that acceleration layer and implement dedicated circuitry optimized for AI vector operations. These processors feature specialized instruction sets and execution units designed for high-dimensional vector manipulations and embedding lookups. Unlike general-purpose processors, these custom processors invest silicon resources in vector processing-related operations only, leading to substantial performance gains [7].

The hardware implementation includes dedicated vector registers of higher width than in conventional architectures; thus, this design enables efficient processing of high-dimensional embeddings without excessive memory transfers. Specialized functional units directly implement in hardware the usual vector operations, which eliminates overhead due to software implementation. In the processor pipeline, the stages for vector loading, computation, and storage have been optimized to minimize stalls and achieve maximum throughput.

These operation latency improvements benefit the inference operations where response time directly impinges on the user experience. The sophisticated prefetching mechanisms in the hardware design follow the predictable access patterns of AI workloads by preloading likely vector data well before explicit requests may occur. Such a proactive approach minimizes waiting periods and ensures that the computational units remain consistently utilized [7].

4.2 Hot/Cold Partitioning Strategy

The system incorporates an intelligent partitioning mechanism to separate hot data from cold data, which creates a multitiered storage hierarchy that is optimized for AI workloads. This approach recognizes the highly-skewed distribution of vector accesses in AI applications, where a small subset of vectors receives disproportionate attention [8].

This hot partition includes a low-latency, high-bandwidth cache optimized for frequently accessed vectors, using advanced memory technologies and carefully designed data structures. The implementation of sophisticated admission and eviction

policies, taking into account not just recency and frequency of access but even semantic relationships between vectors and their role in ongoing computations, is included. This ensures vectors likely to be accessed together remain collocated, enhancing spatial locality and overall cache utilization.

Segregating hot and cold data with specialized memory access patterns optimized for each category improves the efficiency of memory bandwidth utilization. Aggressive prefetching and batched access operations are employed for the hot partition, while more conservative approaches that minimize unnecessary data transfers are implemented for the cold partition [8].

4.3 Bloom Filter Implementation

The probabilistic Bloom filters enable an effective mechanism to determine vector membership in a specific partition without having to incur expensive exact lookups. The space-efficient data structure implemented allows the fast determination of whether a vector is definitely not in the set or may be in the set, thereby reducing the lookup overhead by avoiding searches that are unlikely to succeed.

The system implements Bloom filter designs specialized for vector operations, with dimension-aware hashing functions and adaptive bit allocation strategies. It integrates the filters directly with the hardware acceleration layer, implementing critical operations in custom circuitry to minimize computational overhead [7].

False positive control balances memory usage against the probability of false positives for optimal performance. There are adaptive methods in this implementation that, by dynamically observing the characteristics of the workload, continually readjust the parameters of filters. It brings a substantial reduction of memory footprint compared to traditional indexing methods, which immensely benefits large-scale AI deployments where the number of vectors reaches billions [8].

4.4 Scalability Considerations

Scaling the proposed architecture to meet the demands of enterprise-level AI workloads presents several important challenges that require careful consideration during implementation. The hardware acceleration layer faces significant integration complexity when deployed across large compute clusters. Custom vector processors must be designed with standardized interfaces to ensure compatibility with existing infrastructure, while maintaining the performance benefits that justify their implementation. As vector databases grow to billions of entries, memory bandwidth bottlenecks can emerge, requiring sophisticated data distribution strategies across memory hierarchies [7].

In distributed environments, synchronization overhead becomes a critical concern. Cache coherence across multiple nodes must be maintained while minimizing cross-node communication that could negate the performance benefits of the architecture. This requires careful implementation of distributed Bloom filters and partition management protocols that balance consistency requirements against performance objectives. Studies of large-scale distributed caching systems demonstrate that synchronization overhead can increase non-linearly with system size if not properly managed [9].

Workload variation presents another significant scaling challenge, as different AI applications exhibit distinct vector access patterns. The hot/cold partitioning strategy must adapt dynamically to these varying patterns, requiring sophisticated workload analysis mechanisms that can identify shifts in access distribution without introducing excessive overhead. This challenge is particularly pronounced in multi-tenant environments where diverse AI models share the same infrastructure, each with unique embedding access characteristics [10].

Configuration complexity increases substantially at scale, as optimal parameter tuning becomes a multi-dimensional optimization problem. Parameters including Bloom filter size, bit allocation strategies, hot/cold partition ratios, and prefetch aggressiveness must be continuously tuned based on workload characteristics. This necessitates the development of automated parameter optimization frameworks that can adapt to changing conditions without manual intervention. Performance verification at scale represents the final key challenge, requiring comprehensive benchmarking methodologies that can accurately measure the system's effectiveness across diverse workloads and deployment scenarios [8].

Security considerations become increasingly critical when implementing this architecture at scale, especially in multi-tenant environments. The vector caching system may contain sensitive embedding data derived from proprietary datasets, requiring strong isolation mechanisms to prevent data leakage between tenants. Hardware-accelerated components present unique attack surfaces that must be hardened against side-channel attacks, which could potentially extract embedding information by analyzing timing patterns or power consumption variations [9]. The Bloom filter implementation must also be protected against insertion attacks, where malicious actors could deliberately trigger false positives by manipulating the filter's hash functions. Additionally, the admission/eviction policies in the hot/cold partitioning strategy must be designed to resist cache poisoning attacks that could degrade system performance by manipulating access patterns to compromise the caching efficiency.

To address these security concerns, the architecture should incorporate hardware-level memory protection mechanisms, cryptographic verification of vector integrity, and robust authentication for all data movement operations. Regular security audits and penetration testing specific to vector processing infrastructures should be implemented as part of the system's operational framework. Furthermore, implementing differential privacy techniques can protect the confidentiality of embedding vectors while maintaining their utility for AI operations [10].

The architecture addresses both scalability and security challenges through a modular design that allows components to be scaled independently based on specific workload requirements. This approach enables organizations to prioritize investments in the components most critical to their particular AI applications, whether that involves scaling up vector processing capabilities, expanding hot cache capacity, enhancing Bloom filter precision for massive vector collections, or implementing more sophisticated security controls for sensitive deployments.

Component	Key Features	Performance Benefits	Implementation Techniques
Hardware Acceleration Layer	Custom vector processors, Specialized instruction sets	Substantial performance gains, reduced operation latency	Dedicated vector registers, Hardware-implemented operations, Optimized processor pipeline
Hot/Cold Partitioning Strategy	Intelligent data segregation, Tiered storage hierarchy	Improved cache utilization, enhanced spatial locality	Low-latency hot cache, Sophisticated admission/eviction policies, Specialized memory access patterns
Bloom Filter Implementation	Probabilistic membership determination, Dimension-aware hashing	Reduced lookup overhead, Memory footprint reduction	Space-efficient data structures, Custom circuitry integration, Adaptive bit allocation strategies

Table 3: Performance Comparison: Three-Component Architecture vs. Traditional Caching [7, 8]

5. System Performance and Industry Impact

5.1 Operational Metrics

The integrated system demonstrates considerably improved performance at key performance indicators when compared in production environments. The hardware-accelerated caching architecture drastically reduces network data movement through intelligent caching of frequently accessed vectors and predictive prefetching strategies. This reduction becomes very significant in a distributed training environment where network communication often represents a primary bottleneck. It keeps vector data closer to the compute resources, thereby minimizing the cross-network transfers that generally dominate latency during large-scale AI operations [9].

The power consumption improvements are based on better utilization of computational resources and the reduction of superfluous data movement. Power management, which dynamically adjusts the performance characteristics depending on the workload, is also incorporated in the hardware acceleration components. Furthermore, the hot/cold partitioning strategy provides extra efficiency by reserving energy-intensive resources for the most accessed data. These optimizations, put together, reduce the system energy requirements overall while sustaining or improving the metrics of performance [9].

Query throughput improvement is among the most direct and apparent advantages; for most inference workloads, response time translates directly into the performance of the application. Given a system designed with hardware acceleration for both query execution and response pruning, along with intelligent caching and probabilistic filtering, the number of queries handled by the system per second can be increased significantly over conventional system implementations. This results from lower memory access latency, higher computational efficiency, minimal network transfers, and reduced contention for shared resources that work together to speed up vector operations [9].

5.2 Economic Impact

The architecture presents compelling economic benefits for large-scale AI operations by translating the technical improvements into tangible business value. Storage cost savings represent a significant economic advantage, particularly for organizations

operating at scale. These storage cost savings result from more efficiently utilizing the expensive, high-performance storage resources by ensuring they focus primarily on frequently accessed data. The intelligent caching approach reduces the need for premium storage across the entire vector database, allowing organizations to implement tiered storage strategies without compromising performance [10]. A reduction in the cost of infrastructure for training emerges due to better use of computational resources because of mitigated data movement bottlenecks and optimized memory access patterns. Thus, lower hardware requirements would be able to provide equivalent training performance. This saving will add up significantly for larger organizations training many models simultaneously and may even alter the economics of AI model development [10]. Efficiency improvements in model serving provide economic benefits, particularly for production deployments of AI, where operational costs impact business viability directly. By improving query throughput while reducing resource requirements at the same time, the architecture enables cost-efficient model serving at scale. These efficiency gains translate into concrete economic advantages: lower hardware requirements per query, reduced operational overhead, better service levels, and even greater scalability. In production environments, serving millions of queries, this can lead to enormous cumulative economic consequences, enough to change the calculus of viability for deploying AI at scale [10].

Conclusion

The following research suggests a new way to optimize AI infrastructure by combining hardware-accelerated caching with smart partitioning and probabilistic filtering. This architecture addresses the specific challenges of vector operations in large-scale AI workloads. It shows promise for improving performance, efficiency, and cost compared to standard caching methods. Recognizing and accommodating the unique access patterns of AI vector operations, the system provides a holistic solution that minimizes data movement while maximizing the utilization of compute resources. The custom implementation of vector processors, along with adaptive hot/cold partitioning and Bloom filter lookup optimization, would lead to a synergistic framework fine-tuned for modern AI infrastructure needs. Simulation results using real-world AI workloads indicate that this approach may change the economics and performance characteristics of large-scale AI model training and inference operations, and this might unlock even more efficient development and deployment of increasingly sophisticated AI systems.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Zhijie Nie et al., "When Text Embedding Meets Large Language Model: A Comprehensive Survey," arXiv:2412.09165v4, 2025. [Online]. Available: https://arxiv.org/html/2412.09165v4
[2] Jared Casper and Kunle Olukotun, "Hardware acceleration of database operations," ResearchGate, 2014. [Online]. Available: https://www.researchgate.net/publication/262242434_Hardware_acceleration_of_database_operations
[3] Junkyum Kim and Divya Mahajan, "An Adaptive Vector Index Partitioning Scheme for Low-Latency RAG Pipeline," arXiv:2504.08930v1, 2025. [Online]. Available: https://arxiv.org/html/2504.08930v1
[4] Dong Eun Kim, Tanvi Sharma, and Kaushik Roy, "HASTILY: Hardware-Software Co-Design for Accelerating Transformer Inference Leveraging Compute-in-Memory," arXiv:2502.12344v1, 2025. [Online]. Available: https://arxiv.org/html/2502.12344v1
[5] Donglin Zhuang et al., "Randomness In Neural Network Training: Characterizing The Impact of Tooling," Proceedings of the 5th MLSys Conference, 2022. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2022/file/427e0e886ebf87538afdf0badb805b7f-Paper.pdf
[6] A V Shreyas Madhav et al., "Memory Utilization and Machine Learning Techniques for Compiler Optimization," ResearchGate, 2021. [Online]. Available: https://www.researchgate.net/publication/350128417_Memory_Utilization_and_Machine_Learning_Techniques_for_Compiler_Optimization
[7] Mickael Ide and Corey Nolet, "Accelerating Vector Search: Using GPU-Powered Indexes with NVIDIA cuVS," NVIDIA Developer Blog, 2023. [Online]. Available: https://developer.nvidia.com/blog/accelerating-vector-search-using-gpu-powered-indexes-with-rapids-raft/
[8] Meegle, "Vector Database Partitioning," Meegle Knowledge Base, 2025. [Online]. Available: https://www.meegle.com/en_us/topics/vector-databases/vector-database-partitioning
[9] Tahniat Khan et al., "Optimizing Large Language Models: Metrics, Energy Efficiency, and Case Study Insights," arXiv:2504.06307v1, 2025. [Online]. Available: https://arxiv.org/html/2504.06307v1
[10] Rick Hightower, "The Economics of Deploying Large Language Models: Costs, Value, and 99.7% Savings," Medium, 2025. [Online]. Available: https://medium.com/@richardhightower/the-economics-of-deploying-large-language-models-costs-value-and-99-7-savings-d1cd9a84fcbe