VAST

White Paper

11x Faster Vector Search at 91% Lower Cost

How VAST Eliminates the Hidden Bottleneck in Enterprise AI

Table of Contents

The Hidden Bottleneck in Your AI Pipeline **3**

Why Standalone Vector Databases Are the Wrong Architecture for Scaling Enterprise AI **4**

Staleness & Latency 4

Governance Complexity 4

VAST DataBase: Hierarchical Clustering Indexing for Hyperscale Vector Search **6**

DASE Architecture: The Foundation for Hyperscale Vector Search 6

Proprietary Indexing 7

Centralized Governance 8

Vector Search Performance: The 11x Benchmark **8**

High-Dimensional Retrieval at Scale 8

Continuous Ingest and Indexing at Enterprise Scale 8

Comparative Evaluation: VAST vs. Milvus **9**

The Case for RAG Infrastructure Consolidation **10**

Vector to Vector Cost Comparison 10

Total Cost of Ownership 11

The Hidden Bottleneck in your AI Pipeline

As enterprises build AI factories and move vector retrieval into production, the retrieval layer increasingly becomes core infrastructure, not simply a needed capability. Traditional data architectures struggle to meet the scale, throughput, and latency demands of modern AI workloads, often relying on multiple caching layers, data copies, and distributed services to compensate. They quickly reach limits in throughput and scalability when faced with trillions of records, millisecond latency requirements, and complex pipelines that mix transaction updates with queries. As a result, organizations have been forced to stitch together disparate systems, OLTP databases, data warehouses, file/object stores, vector databases, event brokers, pipeline orchestrators, and more, to handle AI data workflows.

This causes AI workload fragmentation. By bolting on vector store and search as a separate layer, segregated data architectures suffer from increasing costs, complexity, and sharding overhead. This presents organizations with today's painful RAG reality:

- Infrastructure complexity: Teams must manage multiple systems (object storage, a standalone vector database, a metadata catalog, pipeline orchestration, embedding, retrieval, and ranking models) before a single similarity search can execute.
- Limited query expressivity: Applications often need to perform hybrid joins combining similarity search on vectors with filtering on metadata. Most siloed vector stores cannot efficiently execute these unified queries without expensive data replication.
- Synchronization overhead: Changes to source data, plus the continuous flow of new incoming data, require frequent re-embedding and re-indexing, which quickly becomes a full-time engineering problem at scale.
- Governance exposure: Sensitive enterprise data is often replicated and/or stored in third-party cloud vector infrastructure, expanding the data governance perimeter and risk.
- Bloated costs: Standalone vector database licensing, query and egress fees, and increased operational and maintenance costs all drive up total cost of ownership.

Why Standalone Vector Databases are the Wrong Architecture for Scaling Enterprise AI

Dedicated vector databases were built when vector search was a niche capability; they were never designed to handle operational AI at scale. Standalone vector databases force AI engineers to balance recall, latency, and infrastructure efficiency due to three core structural constraints :

- In-memory index requirements: HNSW and similar graph-based ANN indexes must be held entirely in memory to achieve low-latency retrieval. As vector collections grow into the billions, memory residency becomes an expensive scaling ceiling.
- Sharding: As indexes exceed single-node memory, data must be split across multiple nodes, compounding latency by fanning out queries, detaching metadata from the vector system, and introducing shard routing and coordination overhead.
- Partitioning: As vector datasets grow, re-partitioning to rebalance system load requires index rebuilds that freeze data ingest and degrade recall, causing disruption within the RAG pipeline.



Staleness & Latency

Embeddings represent the semantic state of source data at the time they are generated. Any update to that source data, or arrival of new data, immediately makes the existing vector obsolete. If a document is changed but not re-embedded, the AI model will continue to reference the original, now-outdated meaning, leading to inference inaccuracies. Standard vector databases rely on re-embedding pipelines to address this, adding further infrastructure complexity and latency to the retrieval layer.

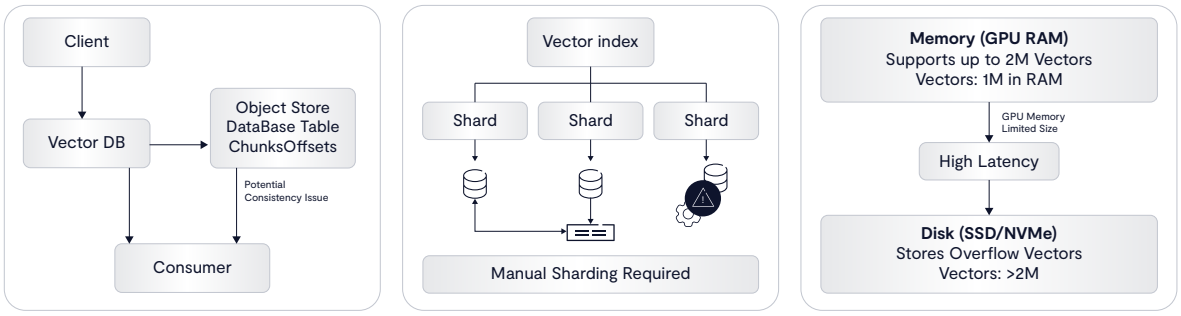


Governance Complexity

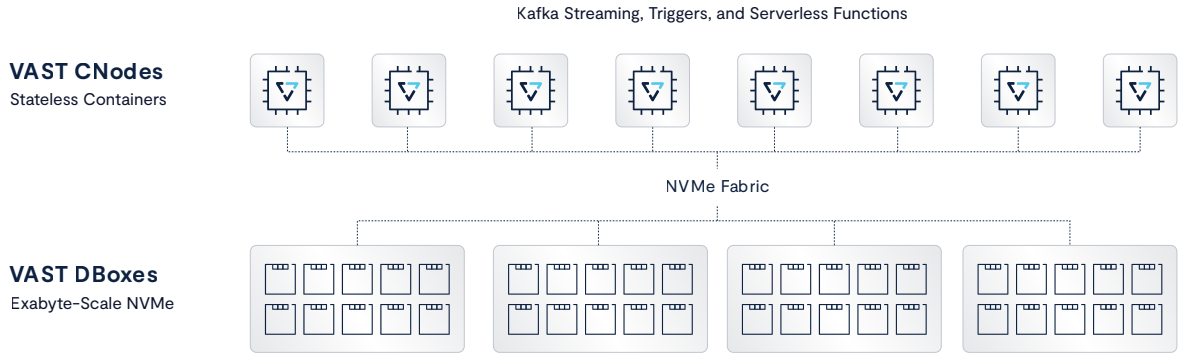
Vectorizing and storing enterprise data in a standalone vector database elevates governance risk by forcing teams to manage permissions across multiple systems, extending the organization's data governance perimeter. Access control duplication, audit logging, and data residency enforcement across systems add to the administrative burden and compliance exposure of maintaining separate vector infrastructure.

Addressing the Limitations of Legacy Vector Based Systems

Before



After



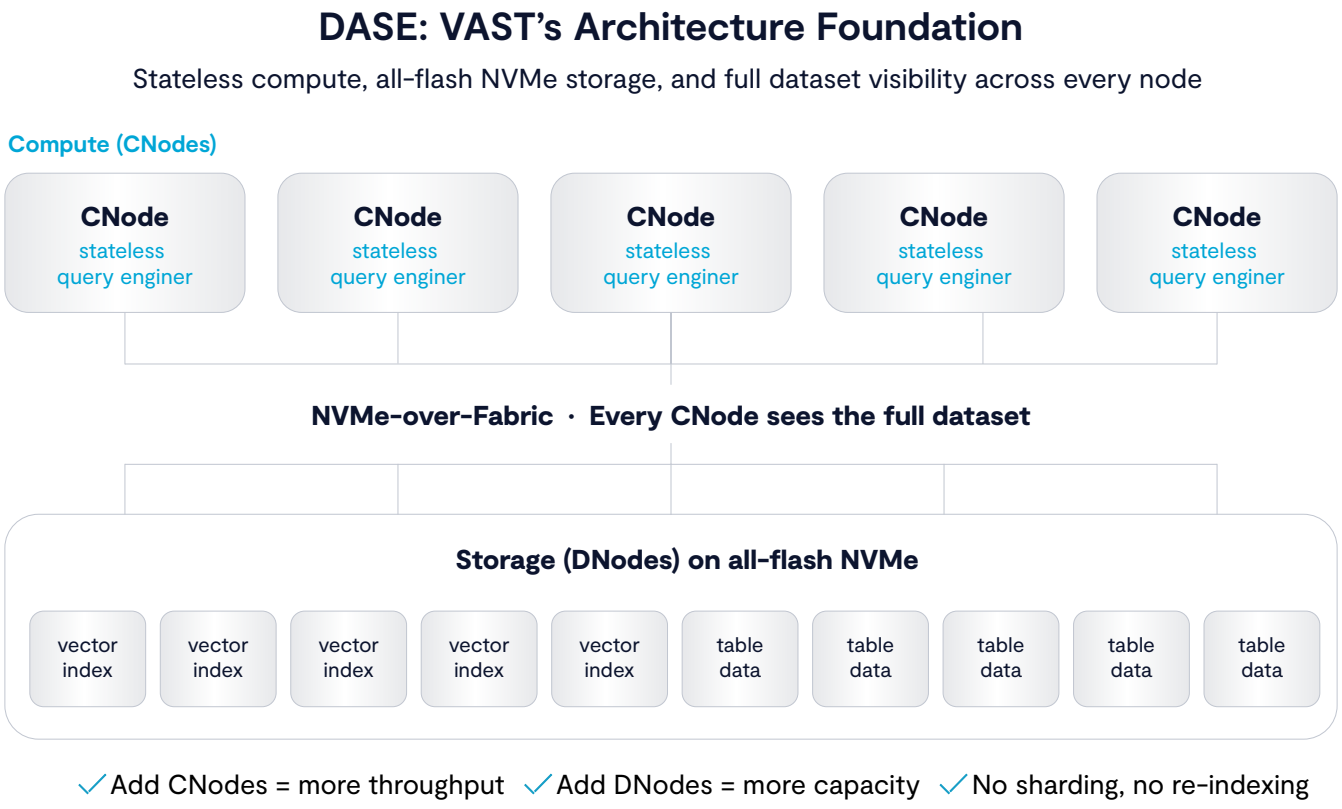
VAST DataBase: Hierarchical Clustering Indexing for Hyperscale Vector Search

Standalone vector databases were never designed to be primary data systems. The architectural assumptions underlying them, in-memory graph indexes, shard-based scaling, external synchronization pipelines, were built for a world where vector search was a niche capability. Those assumptions create the scaling ceilings, governance gaps, and operational overhead described above.

VAST builds vector storage and search natively into the VAST DataBase, the same system that holds structured data for SQL analytics, and on top of the VAST DataStore that manages the unstructured data the vectors are derived from. Within VAST AI OS, the vector isn't a replica in a separate system; it's a column in a table governed by the same permissions, queryable through the same engine, and visible in the same audit trail as every other data type on the platform. New vectors are ingested, clustered, and made available for search without requiring offline rebuilds, shard rebalancing, or separate indexing pipelines.

DASE Architecture: The Foundation for Hyperscale Vector Search

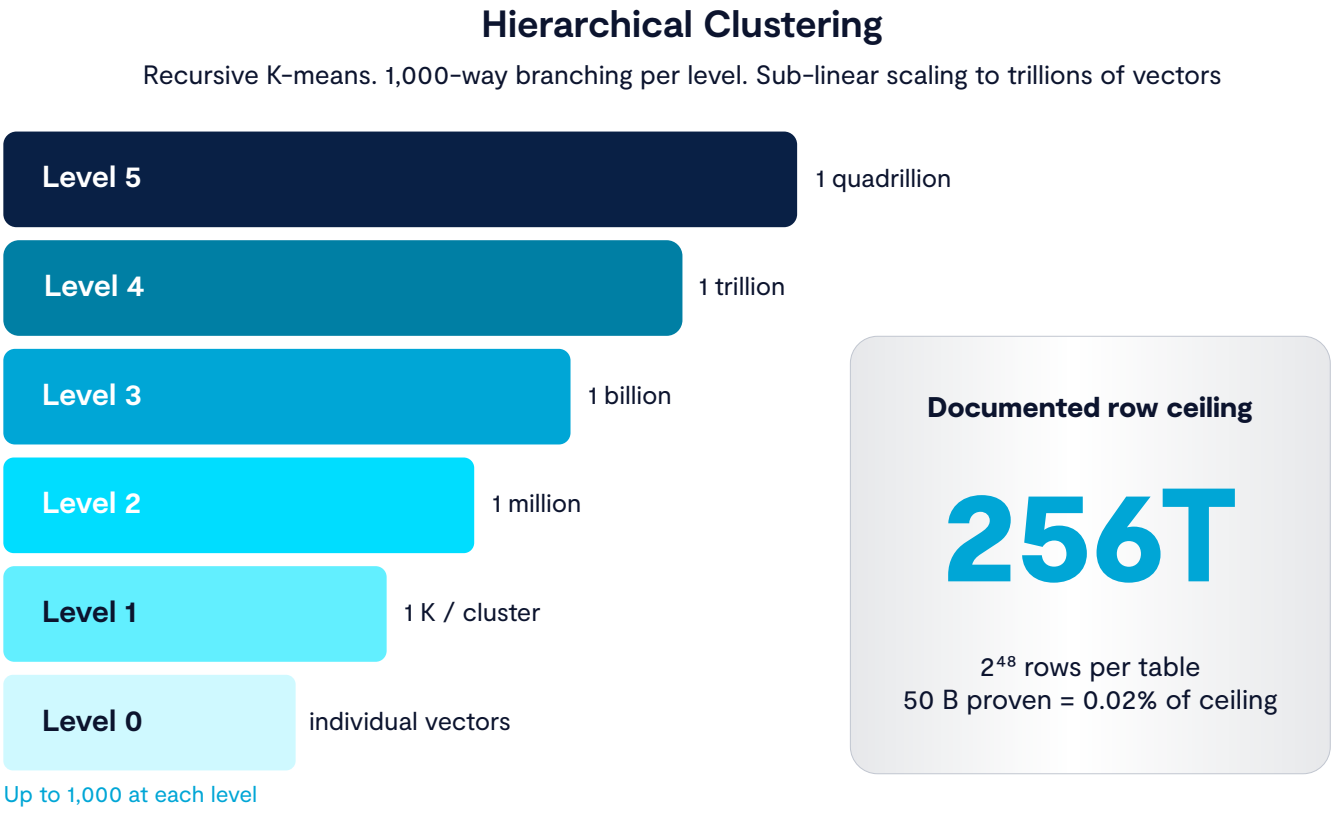
Most vector database performance problems are not algorithmic but structural. In a DASE cluster, compute nodes (CNodes) and storage nodes (DNodes) are fully separated. CNodes are stateless: they own no data and hold no partition assignments. Instead, every CNode connects to every DNode across the cluster via NVMe over Fabrics (NVMe-oF), delivering remote NVMe flash access at latencies approaching local storage. Because no CNode owns any partition, any CNode can execute any query against any vector without coordination, routing overhead, or cross-node data movement.



This has direct consequences for vector search performance. In a sharded system, a query fans out to every shard, each performs a local search, and results are merged at a coordinator; latency grows with shard count. With VAST, search cost is determined by tree depth, not fleet size. Scaling is equally clean: adding CNodes requires no data reorganization, and new nodes immediately participate in query execution across the full namespace. This is why VAST sustains over 1,000 QPS at 50 billion vectors on 8 CNodes.

Proprietary Indexing

VAST's vector indexing uses a proprietary hierarchical clustering model. While traditional IVF organizes vectors into a single layer of clusters, VAST extends this into a recursive five-level hierarchy designed from the outset for trillion-scale retrieval, not stretched there from million-scale assumptions like HNSW or DiskANN. Each level groups up to approximately 1,000 centroids, expanding namespace capacity by a factor of 1,000 per level: L1 holds ~1K vectors through L5 ~1 quadrillion. Scaling adds depth to the tree, not shards or memory pressure.



Continuous Ingest Without Index Freezes

New vectors land in a non-clustered landing table and are organized into the hierarchy incrementally through background clustering. The background process runs K-means at K=1,000 on representative samples to form balanced clusters at each level, and reclustering is triggered automatically when cluster quality degrades. Quality is assessed using a combination of silhouette scoring and monitoring of regional drift in the vector space over time, with additional compaction triggered by excessive deletions. Because reclustering is local to each level and cluster, there are no global index rebuilds and no ingest freezes.

Two-Stage Search with Progressive Narrowing

Vector search executes in two stages. First, the query vector is compared against cluster centroids at each level using quantized distance calculations, accumulating a candidate pool of approximately 5,000 vectors. Second, the system fetches full-precision vectors for those candidates and re-ranks them by exact distance to produce the final top-K results. This design delivers fast approximate filtering at scale while preserving recall accuracy, sustaining 96–99% recall at 50 billion vectors on an 8-CNode cluster.

Vector Search with SQL Predicate Filtering

Because vectors are stored as a native column type alongside structured data in the VAST DataBase, metadata filtering is not a post-processing step; SQL predicates are evaluated as part of the query path against metadata columns in the same table as the vector. This eliminates the synchronization overhead, data duplication, and retrieval latency introduced when vector and metadata systems are separate.

Centralized Governance

When an enterprise generates embeddings from a restricted document and stores the resulting vectors in a separate vector database, the source document's access control metadata is typically not preserved within the retrieval layer. As a result, users with access to the vector index may retrieve semantically relevant content derived from documents they are not authorized to access.VAST eliminates this risk architecturally: vectors are stored as a native column in the same VAST DataBase table and inherits same access controls as the source data in the VAST DataStore. Further, it is subject to the same audit logging and data residency enforcement as any other data on the platform. There is no permission boundary to cross because there is no separate system.

Vector Search Performance: The 11x Benchmark

High-Dimensional Retrieval at Scale

We evaluated VAST's native vector architecture across datasets ranging from 10 billion to 50 billion vectors using both 128-dimensional and 1,024-dimensional embeddings to reflect modern retrieval workloads. At the 1-billion-vector scale, VAST delivered approximately 1,000 queries per second, more than 11× higher throughput than Milvus 2.6 running its disk-based AISAQ index under comparable conditions. Three findings were particularly significant:

- VAST scaled to 50 billion vectors on just 8 CNodes with no observed performance cliff.
- Recall remained consistently high (~96–99%) across both 128- and 1,024-dimensional embeddings.
- Performance remained stable as both dataset size and dimensionality increased.

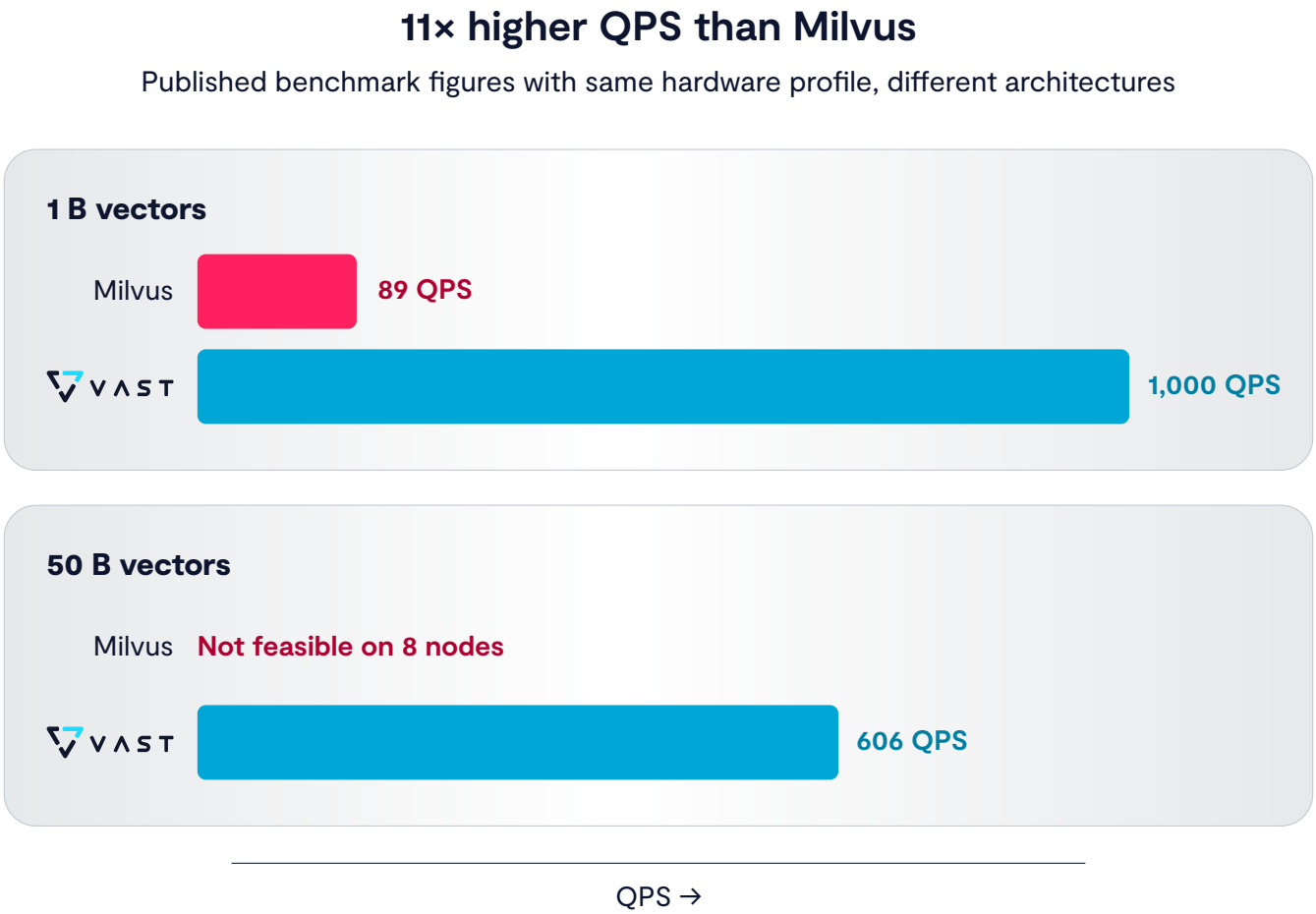
Continuous Ingest and Indexing at Enterprise Scale

High retrieval performance alone is insufficient for production AI systems. VAST was architected to support concurrent ingest, clustering, and retrieval at scale. New vectors are first written to a landing table, then incrementally incorporated into the hierarchical clustering index through background processing. This approach eliminates the need for index rebuilds, rebalancing operations, and maintenance windows.

A single clustering CNode can process approximately 1,000 1,024-dimensional vectors per second. In multi-node deployments, the architecture can ingest and index billions of new vectors per day while simultaneously serving retrieval workloads, enabling continuous operation without disrupting query performance.

Comparative Evaluation: VAST vs. Milvus

We compared VAST against Milvus 2.6 using its disk-based AISAQ backend at the 1-billion-vector scale. Compared to Milvus, VAST demonstrated 11× higher query throughput, 91% lower infrastructure cost per 1,000 queries, and stable retrieval behavior at significantly larger dataset scale. While VAST maintained stable performance at 50B vectors, Milvus throughput degraded significantly as dataset size increased from 1B to 5B vectors on the same hardware configuration.



The Case for RAG Infrastructure Consolidation

Vector to Vector Cost Comparison

In the recent competitive performance evaluation against Milvus 2.6 using its disk-based AISAQ backend, VAST delivered a 91% reduction in cost per 1,000 queries at a 50-billion-vector namespace (128 dimensions).

Cost factor	Milvus (in-memory)	Milvus (disk / AISAQ)	VAST DataBase
Node config	128 CPU, 3TB DRAM	128 CPU, 384GB DRAM	128 CPU, 384GB DRAM
Node count (50B vectors)	~11	~9 (est. ~80 at 50B)	8
Hardware cost per node	~$130,000	~$25,000	~$25,000
DRAM cost for index (50B)	$980,000	$448,000	$6,400 (NVMe flash)
Cost per 1,000 queries	**$0.047**	**$0.033**	**$0.003**
VAST savings vs Milvus disk	—	—	91% lower

Methodology: Cost per query derived from hardware acquisition cost to reach equivalent performance at 50 billion vectors, 128-dimensional embeddings. Milvus in-memory requires ~3TB DRAM per node at ~$14/GB. Milvus AISAQ requires an estimated 10x node count increase to sustain throughput at 50B scale, extrapolated from observed 1B-to-5B scaling behavior. VAST sustained >1,000 QPS on 8 standard CNodes (256GB RAM, 64 cores) with storage on NVMe flash at ~$0.20/GB. Power cost modeled at $0.13/kWh. VAST latency and recall figures measured at 50B vectors; Milvus could not be evaluated at equivalent scale on the same hardware.

Beyond hardware, key architectural differences drive additional TCO savings:

- Memory management overhead eliminated: VAST's memory-bounded architecture removes the need to continuously tune in-memory indexes, pre-allocate capacity, and rebalance as data grows.
- Sharding complexity removed: VAST's shared-everything DASE model scales linearly without data reorganization or application-layer changes.
- Deduplication reduces storage at scale: VAST's inline deduplication capability applies to vector workloads; a unique infrastructure-level efficiency not available in purpose-built vector databases.
- Compliance and governance overhead reduced: Consolidating on VAST eliminates the need to duplicate access controls, audit logging, and data residency enforcement across systems.

Total Cost of Ownership

Beyond the vector store and retrieval layer, many other factors influence the total cost of ownership for an entire RAG pipeline from object storage to pipeline orchestration and agentic AI deployment. Consolidating these services with VAST AI OS generates significant savings for ongoing AI system operations.

AI Process	VAST AI OS Component
Object storage	VAST DataStore
Vector store & retrieval	VAST DataBase
Event-driven compute	VAST DataEngine
Automated pipelines	VAST InsightEngine & VAST DataEngine
Agent development & deployment	VAST AgentEngine
Global access management	VAST DataSpace

Vector retrieval is quickly becoming foundational infrastructure for analytics and AI workloads, but most standalone vector systems were never designed for enterprise-scale retrieval, continuously updated datasets, or hybrid analytical workflows.

VAST DataBase was built differently. By combining native vector retrieval, transactional structured data, streaming ingestion, and shared-everything architecture on a single platform, VAST eliminates the synchronization layers, shard coordination overhead, and fragmented infrastructure that increasingly limit conventional vector architectures at scale.

Whether you are evaluating large-scale semantic search, multimodal retrieval, operational AI pipelines, or hybrid vector and analytical workloads, VAST can help you assess the architectural tradeoffs of your current environment and benchmark against real production-scale workloads.

©2026 VAST Data, Inc. All rights reserved. All trademarks belong to their respective owners.

06/ 2026 | 11474