VAST

White Paper

Beyond the Warehouse and the Lakehouse

A technical brief on VAST DataBase: the single platform for the analytical and operational workloads AI is producing today.

Table of Contents

Executive Summary 3

Where the Warehouse and the Lakehouse Fall Short 4

VAST DataBase Architecture 5

Performance Proof Points 7

How customers are deploying VAST DataBase 9

Economics of Consolidation 10

Closing 10

Executive Summary

For years, enterprise analytics followed a familiar progression. Data warehouses delivered high-performance analytical queries, but at significant cost and with limited retention, forcing organizations to carefully choose which data to model and store. Data lakes emerged to solve the retention problem by allowing raw data to be stored at massive scale in low-cost object storage, but analytical workloads still depended heavily on downstream warehouses for performance.

The lakehouse attempted to close that gap by allowing analytics directly against open table formats such as Iceberg and Delta Lake on object storage, reducing some of the ETL and duplication between lakes and warehouses. But as workloads scaled, many organizations continued relying on warehouses for performance-sensitive analytics, while operational systems, streaming infrastructure, vector retrieval, and analytical platforms remained separated across multiple synchronized systems.

AI workloads increasingly collapse those boundaries. Embedding pipelines, fraud analytics, operational AI systems, and retrieval workloads now combine continuous updates, low-latency lookups, and large analytical scans against the same datasets, often at the same time. The typical response is to stitch together warehouses, lakehouse tables, vector stores, streaming systems, operational databases, and synchronization pipelines simply to keep data fresh and queryable across multiple infrastructure tiers.

VAST DataBase is designed for this converged workload model. Rather than layering analytics over immutable files or separating operational and analytical systems into different platforms, VAST combines mutable analytical storage, streaming integration, vector retrieval, and large-scale analytics within the same shared-everything architecture. It preserves an open ecosystem (Spark, Trino, PyArrow, Iceberg-compatible reads) while enabling continuously updated analytical datasets at enterprise scale.

This paper walks through the VAST architecture, the workloads it enables, and the consolidation patterns customers are deploying today.

Where the Warehouse and the Lakehouse Fall Short

Each architecture was designed for a workload pattern that AI applications simply don't fit into.

The warehouse story

Modern cloud warehouses solved many of the scaling limitations of earlier MPP systems by decoupling compute and storage and simplifying ingestion from object storage. But they still largely preserve the architectural separation between operational data, streaming systems, and analytical infrastructure. AI workloads increasingly blur those boundaries, forcing organizations to maintain multiple synchronized systems to support transactional updates, retrieval, and analytics on the same data.

Even with lakehouse adoption, analytical, operational, and streaming workloads typically remain distributed across multiple systems. As AI workloads increasingly combine continuous updates, retrieval, and analytics on the same data, infrastructure complexity and compute costs grow with every synchronization layer added between them.

The lakehouse story

Lakehouses brought a much-needed solution to the issues of warehouse cost and rigidity: open table formats on top of a data lake let teams analyze data in place without having to move it into a warehouse first. However, three problems still show up at scale, especially for massive AI workloads:

- **Data quality:** As lakehouse environments grow, data quality and governance often become fragmented across pipelines, teams, and storage tiers, making it harder to ensure AI systems operate on fresh, trusted data.
- **Stale data:** Lakehouse freshness often depends on increasingly complex streaming and compaction workflows. As workloads become more dynamic, teams are forced to balance freshness, metadata overhead, query performance, and infrastructure complexity.
- **Not real-time enough for AI:** The immutable file model of object storage, combined with the metadata coordination required by lakehouse table formats, limits how quickly data can be mutated in place. Apache Iceberg adds ACID semantics on top of Parquet, but it does not change the underlying file model. As mutation frequency rises, maintaining fresh analytical views requires increasing coordination, compaction, and metadata overhead

Where AI breaks both

AI applications increasingly require continuously updated data, low-latency lookups, and analytical scans against the same dataset. Neither the warehouse nor the lakehouse were designed to do all three on the same dataset, and stitching them together with streaming systems and operational stores produces the complex infrastructure this paper argues against.

VAST DataBase Architecture

VAST DataBase is one capability of the VAST AI OS, a single platform that exposes structured, unstructured, and streaming data through a shared architecture. VAST DataStore, DataBase, and DataEngine are not separate products; they are capabilities of one system. That distinction matters because the operational cost of running a warehouse plus a lakehouse plus their supporting systems is, in practice, the cost of running and integrating multiple independent products.

Disaggregated Shared-Everything (DASE)

VAST Disaggregated Shared Everything (DASE) is the architecture underneath the platform, uniquely distinguishing VAST from shared-nothing analytical systems. Stateless compute nodes (CNodes) share global access to a single flash storage tier (DNodes) without requiring inter-node coordination on data ownership.

Compute and storage scale on independent axes: a workload that grows in concurrency does not require more storage, and a workload that grows in dataset volume does not require more compute. The tightly coupled warehouse cannot match this architecture model without overprovisioning.

Because nothing is partitioned across nodes in DASE - and every CNode can simultaneously see every DNode - queries execute against the same data without the east-west traffic that limits shared-nothing clusters. This east-west traffic in shared-nothing distributed databases grows quadratically with cluster size, which is the practical reason why these clusters get so fragile at scale. DASE replaces that pattern with shared-everything access, and the cluster behaves like a single computer rather than a federation of nodes.

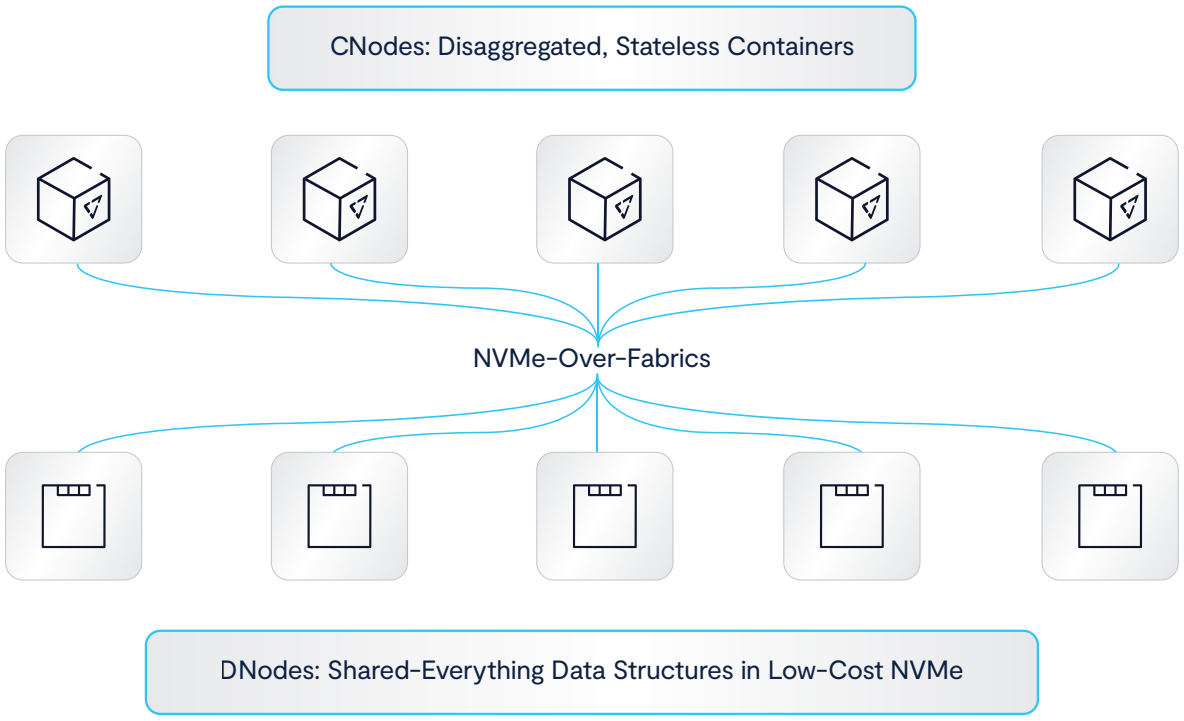


Figure 1. VAST Disaggregated Shared Everything (DASE) Architecture

Storage layout: small chunks, precise reads

In VAST DataBase, data ingests as rows into storage-class memory (SCM), where it is durable and immediately queryable. An asynchronous migration converts those rows into a columnar layout on flash in chunks roughly 32 KB in size, rather than the large 128 MB row groups typical of Parquet.

The chunk size matters most on the read path, which is where most of the customer value lives. Three benefits follow:

- **Smaller read amplification:** Predicate pushdown executes at the storage layer, and VAST returns only the columns and ranges that satisfy the query rather than full row groups that the engine then has to filter.
- **Higher performance:** Queries see less data on the wire and yield faster execution times, enabling substantially greater concurrency for distributed teams.
- **Lower compute consumption:** Less work per query at the engine layer translates directly into smaller compute footprints at the same workload volume.

On the write path, a row-level update touches only the affected chunk. There is no large file rewrite, and no compaction pipeline is required to amortize update cost. Open table formats, like Apache Iceberg and Delta Lake have to simulate mutability on top of immutable files; VAST DataBase does not.

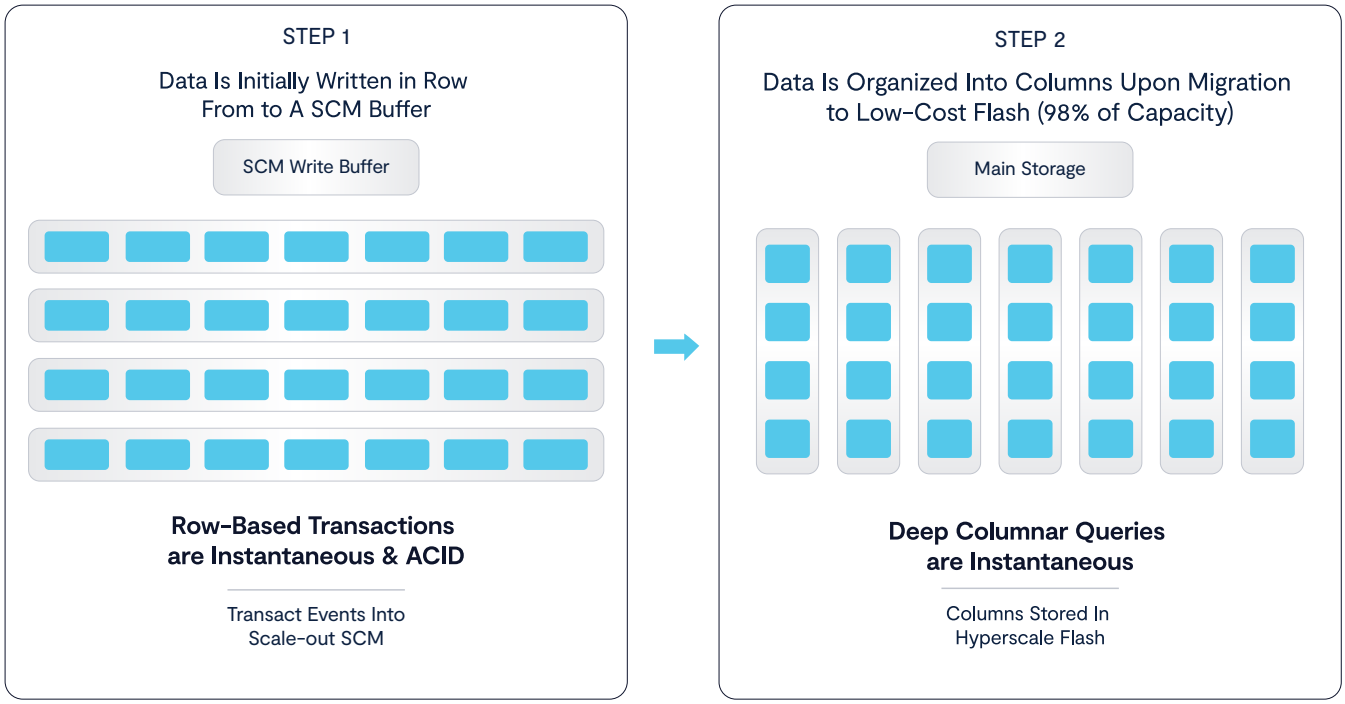


Figure 2. VAST DataBase storage layout.

Transactional semantics

VAST DataBase is fully ACID compliant with repeatable-read isolation and support for multi-statement transactions. The B-tree bitmap leaf blocks maintain multiple versions of each address tagged by snapshot timestamps, so concurrent transactions resolve to a consistent view without blocking on read locks. Snapshots, replication, row- and column-level security, and time travel all run against the same versioning machinery rather than being bolted on at the engine.

Sorted Tables and partitioning: tuning options on a system that already performs

Sorted Tables maintain a user-selected sort order continuously in the background. For workloads with high-cardinality access patterns such as point lookups or range scans on a sort key, they convert random I/O into sequential I/O and reduce lookup complexity to O(log n) over the sort hierarchy. Partitioning is similarly available where a join or filter pattern benefits from it. Both are elective features that customers can opt into for specific workload patterns; they are not required for VAST DataBase to perform well as an analytical database.

DataEngine: streaming, in-place pipelines, and managed engines

DataEngine is the compute capability of VAST AI OS. Four things sit underneath it:

- **The VAST Event Broker:** The Event Broker collapses streaming and analytics onto the same architecture. Kafka-compatible topics persist directly into VAST DataBase, so the streaming system and analytical store are no longer separate infrastructure tiers. There is no message bus to operate separately from the analytical store; the Kafka topic instantly materializes as a DataBase table.
- **Real-time event analytics from streaming sources:** Because topics instantly materialize as tables, streaming events land directly into structured, query-ready tables. Analysts can query the event stream the moment it arrives, and the same tables can be joined to historical archives without ETL between them. Both patterns are valuable, and the combined query against fresh and historical data is one of the more valuable workflows the platform enables.
- **DataEngine triggers and serverless Python functions:** Event triggers respond to topic or data-change activity, while Python functions execute inside the VAST runtime for real-time transformation, enrichment, and orchestration without separate streaming infrastructure
- **Managed and external engines:** VAST DataBase integrates with existing Spark, Trino, and Python frameworks through standard catalogs and APIs. Organizations can continue operating external compute environments or consolidate analytics infrastructure by using managed Spark and Trino services that run directly on VAST CNodes.

Performance Proof Points

Each benchmark below is reported relative to the architectural property it validates. The number is meaningful because the architecture predicts it.

Transactional updates at analytical scale

Lakehouse architectures were designed around immutable Parquet files on object storage. As workloads become more dynamic, including real-time RAG pipelines, fraud analytics, and continuously updated operational datasets, maintaining fresh analytical views requires increasingly complex buffering, compaction, and metadata coordination. VAST DataBase eliminates these constraints with a file-free transactional architecture that supports native row-level updates directly in its columnar store.

In VAST benchmarking, an Iceberg table updated through Trino MERGE at 500-row batches with a 20% per-column update probability sustained approximately 37 update transactions per second. The same workload against a VAST DataBase 1-billion-row table on a 3x1 cluster (three CNodes, one flash enclosure) executed through the vastdb Python SDK with parallel Arrow batches sustained approximately 8,000 transactions per second.

Full methodology appears in Transactional Lakehousing with the VAST DataBase.

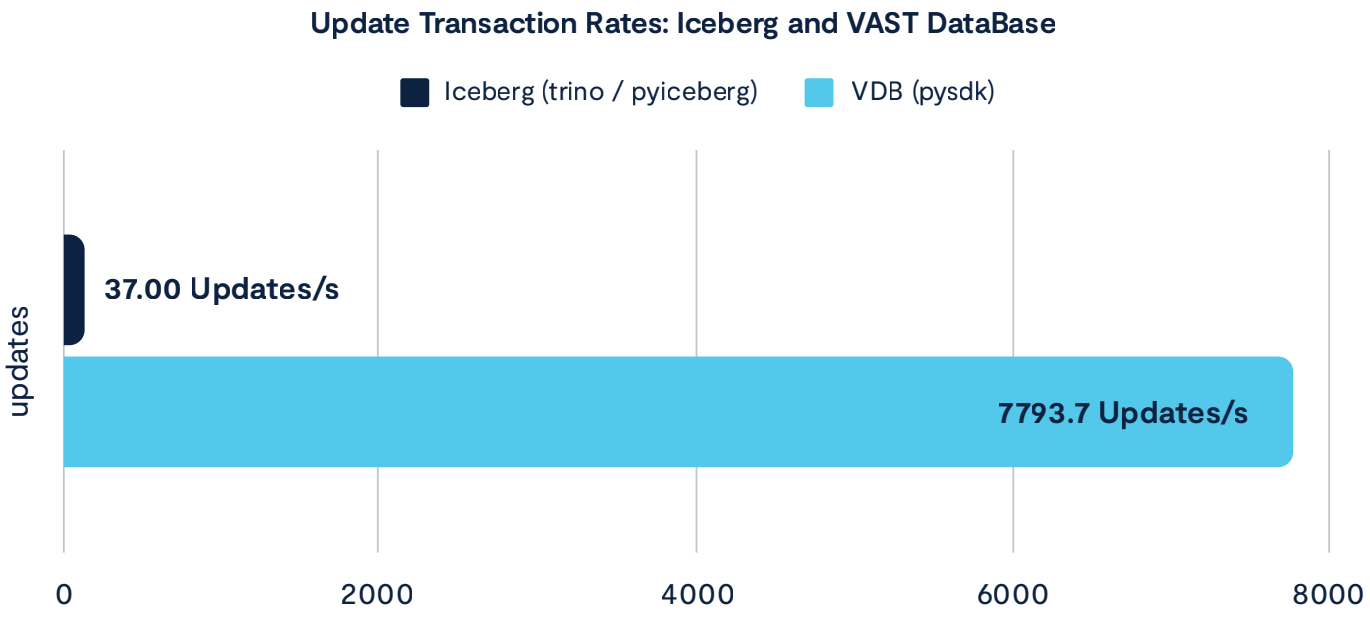


Figure 3. Update throughput, VAST DataBase versus Iceberg, 1-billion-row table.

The value behind these numbers is not merely the increase in raw transaction throughput: it's the fact that, with VAST, operational and analytical workloads can address the same table, on the same data, without an intermediate operational data store and the add-on synchronization pipeline that connects it to the warehouse or lakehouse.

Concurrent point lookups at analytical scale

Sorted Tables maintain a non-overlapping sort hierarchy that the storage layer uses to prune access at chunk granularity. Iceberg can simulate sort by partitioning on the candidate column, but the predicate is evaluated at the Parquet reader, which has to materialize row groups across the network before filtering.

In a point-query workload against a 10-billion-row table at 128 concurrent threads, lookup latency averaged approximately 120 ms per thread on VAST DataBase. The Iceberg comparison, partitioned into 1,024 buckets on the same column, averaged approximately 1.6 seconds per thread, roughly 13 times slower. The difference reflects where predicate evaluation happens: at the storage layer in VAST, and at the Parquet reader in Iceberg.

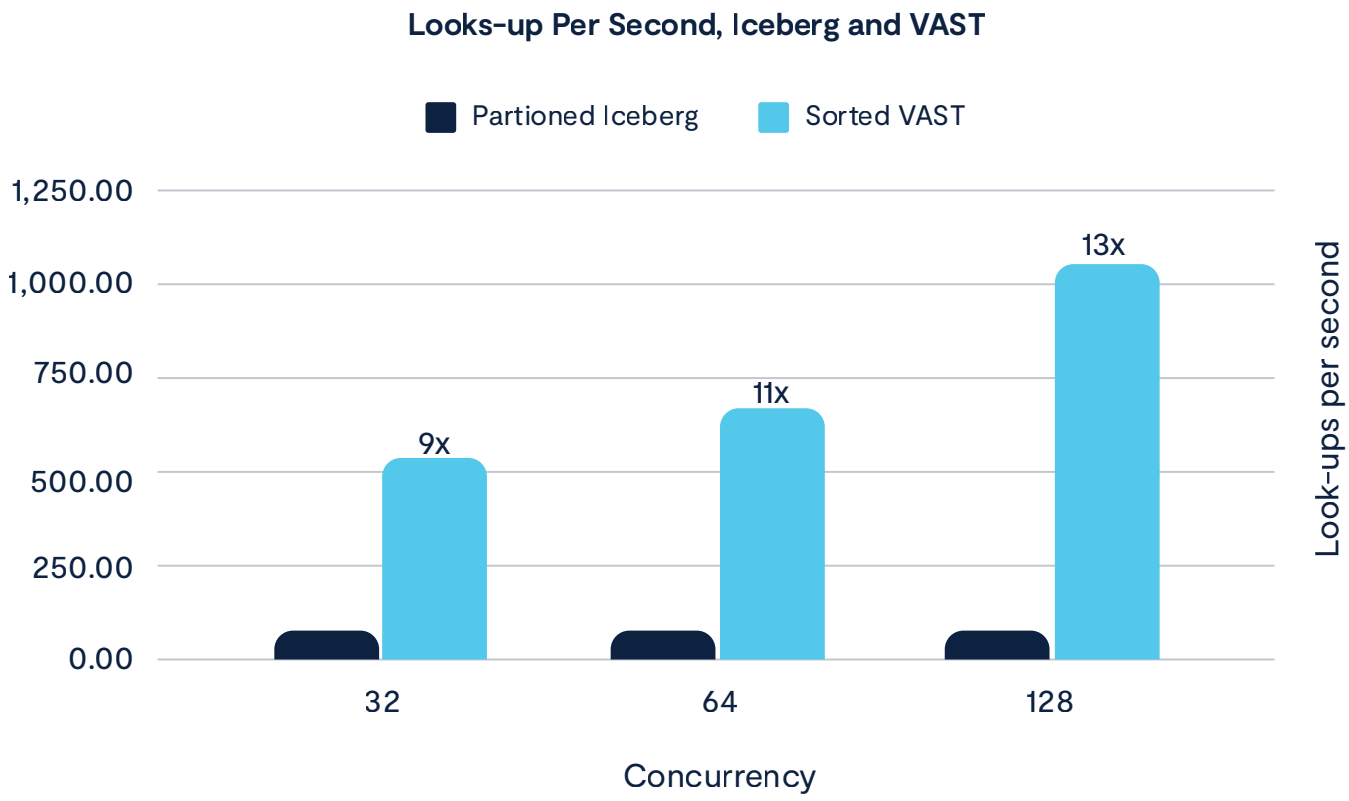


Figure 4. Point-lookup latency per thread at 1, 8, 32, and 128 concurrent threads, VAST DataBase versus Iceberg, 10-billion-row table.

The customer value here is that low-latency lookups and large analytical scans can run against the same table, eliminating the dual-write pattern that lakehouse-plus-operational-store architectures require.

TPC-DS analytical performance on open table formats

On the TPC-DS benchmark, VAST DataBase ran approximately 30% faster than the same workload against Iceberg on Amazon S3 on comparable hardware. For customers evaluating cloud-warehouse migration paths or lakehouse consolidation, this is the relevant baseline: open table format performance comparable to or better than the cloud-warehouse alternative.

How customers are deploying VAST DataBase

Four workloads dominate VAST DataBase deployments.

Modernizing the data warehouse

Legacy on-premises and cloud warehouses hitting concurrency, capacity, and cost ceilings are among the most common migration sources. DASE removes the tight coupling between compute and storage, allowing each to scale independently so concurrency growth does not require proportional storage expansion. Existing Spark, Trino, and PyArrow workflows continue to operate against VAST DataBase, making migration a query-layer reconnection rather than a workload rewrite

Simplifying the lakehouse

Iceberg and Delta Lake deployments accumulate operational overhead that was not in the original business case: compaction scheduling, small-file management, manifest coordination, and the engineering capacity those impose. With VAST DataBase, there is no compaction to schedule, no small-file remediation cycle, and no external compute cluster.

ETL simplification and cloud repatriation

Unpredictable cloud egress, non-linear compute pricing, and multi-system ETL chains have increased the operational and economic complexity of modern analytics architectures. DataEngine triggers replace many external ETL workflows with in-place transformation, while the Event Broker collapses streaming and landing pipelines into the platform itself. VAST AI OS can be deployed on-premises, in private cloud environments, or with hyperscale cloud integrations, allowing organizations to reduce infrastructure sprawl, data movement, and operational overhead across deployment models.

Real-time operational analytics

Telemetry, retail transactions, advertising logs, and network events all require the same thing: live ingest and analytical access against the same dataset at high concurrency. Conventional implementations stitch together Kafka clusters, operational stores, lakehouse tables, and synchronization pipelines. With VAST, streaming topics persist directly into queryable DataBase tables, while transformations can execute through DataEngine, Spark, Trino, or existing analytical frameworks operating against the same underlying data.

Economics of Consolidation

The cost story for VAST DataBase is less about per-unit pricing than about how many systems an organization is operating. A modern AI analytics stack typically consists of a data lake, an event broker, an operational data store, external compute clusters, a data warehouse, and the ETL and integration layers between them. Each tier carries its own licensing, monitoring, security model, infrastructure footprint, and engineering overhead, while the cost of integrating and moving data between them grows with scale and concurrency.

VAST AI OS consolidates these tiers onto a single shared-everything platform. DataStore covers the data lake, DataBase unifies operational and analytical data, and DataEngine handles streaming, orchestration, and compute workflows. The result is not only fewer infrastructure tiers, but also lower hardware footprint, reduced power and cooling requirements, less data movement, and substantially lower operational complexity. At full consolidation, organizations can reduce total cost of ownership by approximately 60% while simplifying management across on-premises, private cloud, and hyperscale deployments.

Closing

The data warehouse and the lakehouse were each the right answer for their workload era. AI applications expose the outdated assumption underneath them both: that transactional and analytical workloads can be separated and served by different systems. VAST DataBase is the alternative when that assumption no longer holds, providing one platform that handles continuously updated data, low-latency lookups, and large analytical scans on the same data.

The patterns this paper describes (operational data stores consolidated onto analytical surfaces, streaming pipelines replaced by trigger functions, warehouse and lakehouse functions running against a single namespace) are how customers are using VAST DataBase to retire the warehouse-plus-lakehouse infrastructure that AI workloads have made complex and expensive to maintain.

©2026 VAST Data, Inc. All rights reserved. All trademarks belong to their respective owners.
6/26 | 10753