VAST

White paper

Stream and Query in Real-Time

How VAST Event Broker Closes the Kafka-to-Analytics Gap

06/26 | Version 1.0

Table of Contents

The Kafka-to-Analytics Gap 3

How the Gap Forms 3

Why Broker Architecture Creates Complexity 4

The Shared-Nothing Architecture 4

The Cost of Scale 4

Why Is VAST Event Broker Different 5

VAST Event Broker: Architecture 6

A Native VAST DataEngine Capability 6

Disaggregated Shared-Everything Architecture 6

Topics as Tables 7

From Topics to Queryable Tables: JSON Expansion 8

Benchmark: VAST Event Broker Throughput 8

Methodology 8

Read/Write Workload 9

Write-Only and Read-Only Workloads 9

Vertical Use Cases 10

Summary and Migration Guide 11

Migration 11

The Kafka-to-Analytics Gap

VAST Event Broker closes the gap between event streaming and real-time analytics by making event topics immediately queryable the moment data arrives, without ETL pipelines, separate storage systems, or additional infrastructure. This paper explains why that architectural shift matters, how VAST achieves it, and what it means in practice for organizations running AI and real-time analytics workloads.

Kafka became the enterprise standard for event streaming because it provided a scalable and reliable way to move massive volumes of real-time data between systems. For years, that architecture worked well enough: events could be streamed into downstream storage and analytics platforms through separate ETL pipelines without creating significant operational friction. That separation no longer adequately meets the demands of modern workloads. AI inference pipelines, real-time fraud detection, personalization engines, and agentic automation all require event streams to be immediately queryable the moment data arrives. Increasingly, organizations also expect incoming events to trigger automated data and AI workflows in real time. As AI pipelines become more event-driven, the challenge is no longer just moving data faster, it is enabling event streams to become both immediately actionable and immediately analyzable.

How the Gap Forms

Under traditional streaming architectures, event data passes through multiple systems before becoming broadly queryable: from the event broker into stream processing, then into analytical storage, and finally into a query engine. Technologies such as Flink, Spark Streaming, and Kafka Streams can deliver real-time transformations on top of Kafka. These approaches, however, still require separate processing engines, additional infrastructure layers, and ongoing synchronization. As organizations scale AI pipelines and real-time workloads, this architectural separation creates operational complexity and infrastructure sprawl, even when each individual component is well-engineered. The consequences compound across several dimensions:



Analytics Latency: Streaming data passes through multiple processing, storage, and analytical layers before becoming broadly queryable, delaying time-to-insight.



Infrastructure Fragmentation: Event transport, stream processing, analytics platforms, and query engines are maintained as separate systems, increasing operational overhead.



Replication and Storage Costs: Kafka-style architectures replicate event data multiple times across brokers, while downstream pipelines often duplicate the same data again into object stores, lakehouses, or feature stores.



Operational Overhead: Large-scale streaming environments require ongoing management of brokers, partitions, consumer groups, scaling policies, and stream processing frameworks.



Cloud Cost Growth: Managed services such as Confluent Cloud and Amazon MSK can become expensive at high throughput and retention levels, particularly when combined with separate analytics and AI infrastructure.

Why Broker Architecture Creates Complexity

Kafka and related platforms rely on shared-nothing distributed architectures that use partitioning, replication, and inter-node coordination to achieve scale and durability. These mechanisms are effective for event transport, but as deployments grow, they consume increasing compute, storage, and network resources while adding operational overhead for scaling, rebalancing, and system management.

The Shared-Nothing Architecture

Traditional Kafka systems partition event topics across multiple brokers, with each partition maintaining its own event order and resilience provided by cross-broker replication. These coordination mechanisms serve important purposes: replication provides durability; partitioning enables distributed ownership and scale. The challenge at large scale is that they consume meaningful compute, storage, and network resources alongside application workloads. Common deployments often maintain three replicas, resulting in ongoing overhead from managing replica health, rebalancing partitions, and handling broker lifecycle events.

The Cost of Scale

Kafka scales effectively and is used to operate some of the largest streaming deployments in the world. The more pressing question for most organizations is what operational complexity and infrastructure cost are required to sustain that scale:

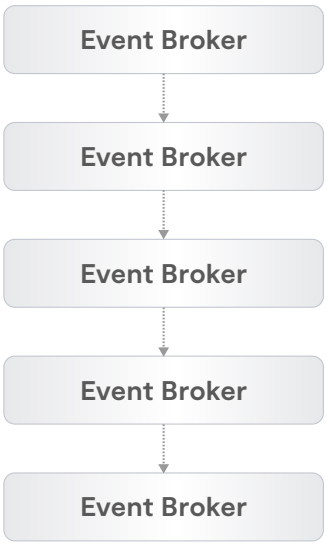
- **Queuing Latency:** Larger batch sizes improve throughput efficiency but introduce additional latency before events are processed.
- **Partition and Broker Management:** Scaling requires additional brokers, partition reassignment, and data redistribution that consume network and operational resources.
- **Rebalance Disruption:** Rebalances can temporarily affect throughput and increase latency while partitions and leadership assignments are redistributed.
- **Metadata Overhead:** Managing very large numbers of partitions increases the load on cluster coordination at the metadata layer.

The deeper challenge is structural: traditional architectures maintain a separation between event transport and analytical queryability. Closing that gap requires additional infrastructure: stream processors, analytical storage systems, and ETL pipelines, each introducing its own operational overhead and latency.

To address these constraints, VAST DataEngine provides the VAST Event Broker as a native platform capability. By consolidating event streaming, analytics, storage, compute, and orchestration into a single system, the VAST Event Broker makes event streams immediately queryable without requiring another streaming platform or additional infrastructure.

Why Is VAST Event Broker Different

Traditional Streaming Architecture

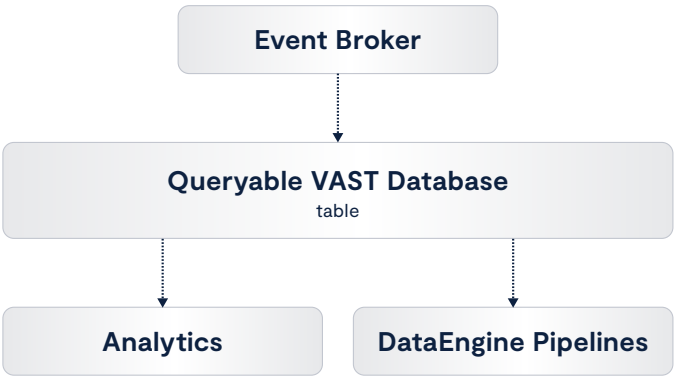


Multiple systems

Multiple copies of data

Multiple operational domains

VAST Event Broker Architecture



One platform

One copy of data

Immediate queryability event-driven execution

VAST Event Broker: Architecture

The VAST Event Broker is a Kafka-compatible API that allows applications to produce and consume messages using standard Kafka clients. It implements the Kafka wire protocol directly, enabling existing producers, consumers, and client libraries such as librdkafka, the Java SDK, Python kafka-python, Kafka Connect, and others to work without modification. As a result, it can serve as a drop-in replacement for systems that produce to or consume from Kafka APIs.

Unlike traditional streaming architectures that separate brokers from analytical storage, VAST Event Broker stores event streams directly in VAST DataBase. Topics are implemented as DataBase tables, providing a seamless protocol-level replacement in which existing producers and consumers require no code changes. By storing topics as tables and relying on VAST's underlying storage architecture for durability, VAST eliminates much of the broker-level replication overhead associated with traditional streaming platforms.

A Native VAST DataEngine Capability

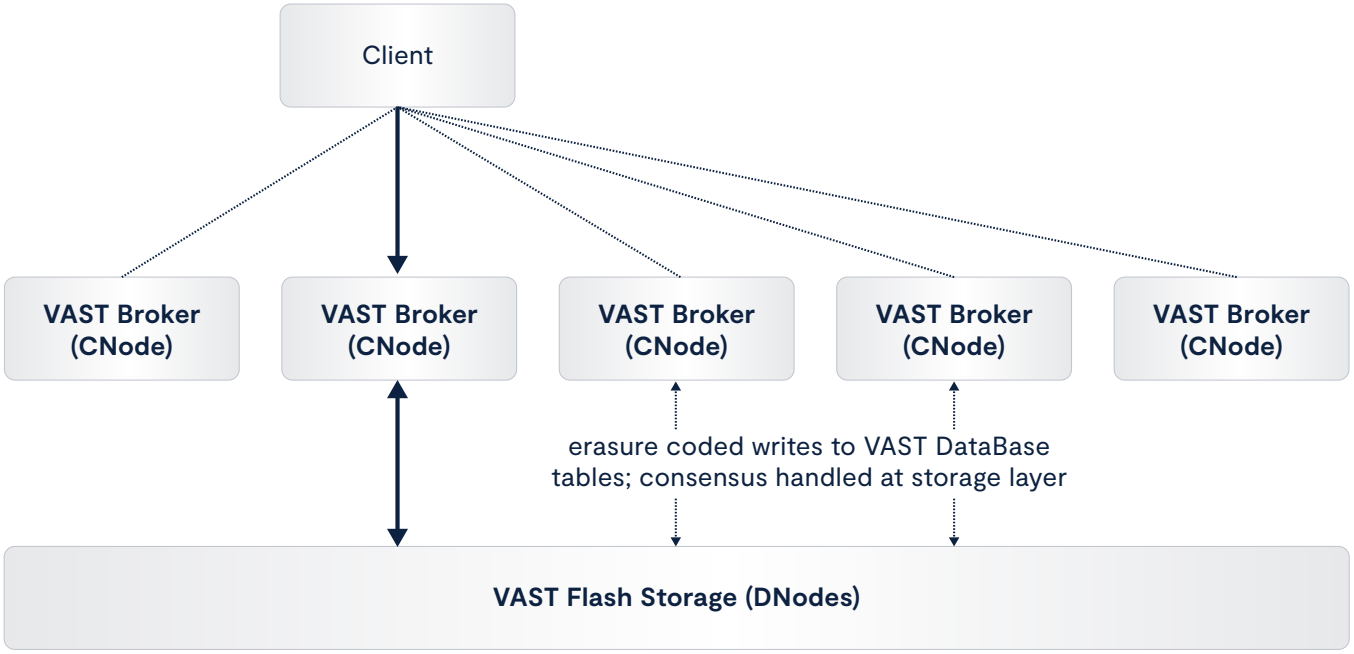
Built into the VAST DataEngine as a native capability, Event Broker converges streaming, analytics, storage, and compute into one platform. Event Broker serves as the event-driven foundation of VAST DataEngine. Incoming events can trigger serverless Python functions and orchestrated data or AI pipelines that perform enrichment, transformation, inference, vectorization, RAG updates, and other processing workflows automatically as data arrives. Those same event streams are simultaneously materialized as queryable VAST DataBase tables, allowing analytics, AI, and operational workloads to operate directly against fresh event data in real time.

- **Event-Driven Pipelines:** The Event Broker initiates autonomous data and AI pipelines for enrichment, transformation, inference, and orchestration immediately upon data arrival.
- **Serverless Processing:** DataEngine serverless Python functions execute custom logic in direct response to events, enabling organizations to transform data, generate embeddings, invoke AI models, update vector indexes, or automate downstream processes without separate infrastructure.
- **Real-Time Analytics:** Event Broker combines pub-sub streaming, real-time database ingestion, and event-driven triggers to power outcomes such as real-time anomaly detection and live feature engineering.

Disaggregated Shared-Everything Architecture

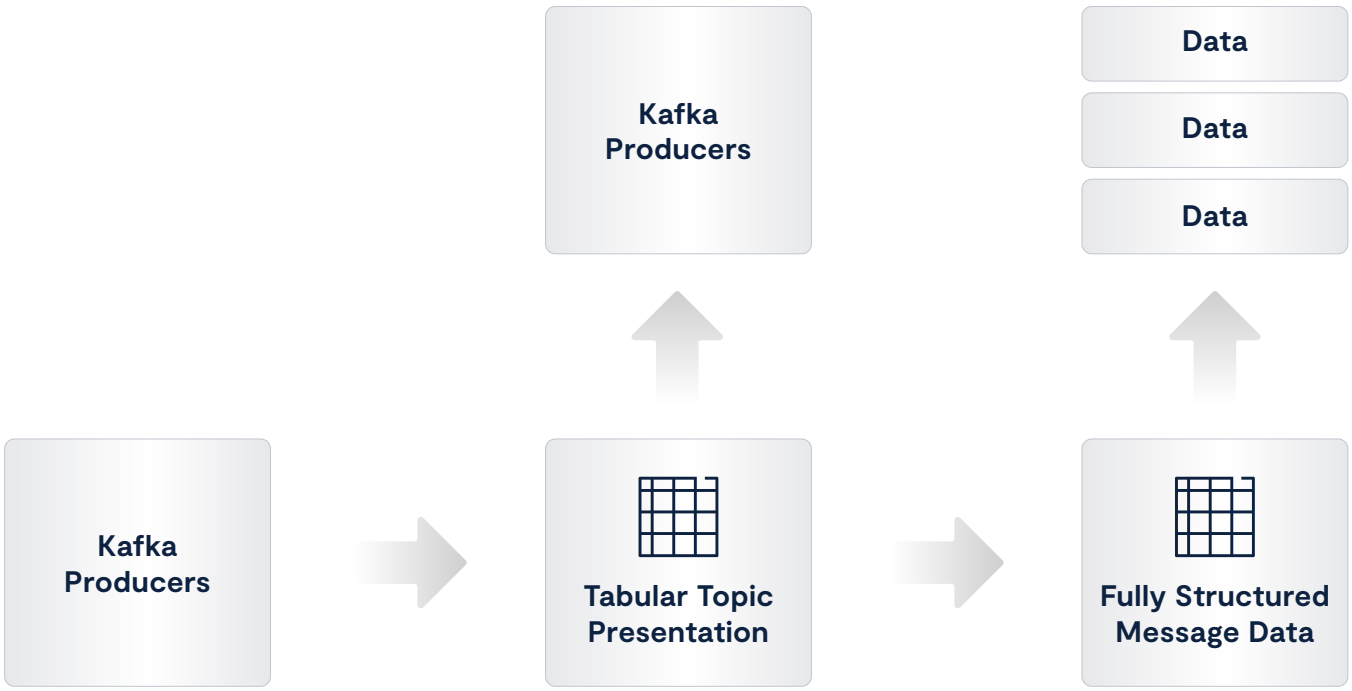
VAST DataEngine and Event Broker are built on VAST's Disaggregated Shared-Everything (DASE) architecture, which separates compute and storage so that broker compute nodes (CNodes) scale independently of storage capacity. This eliminates the need for broker-level partitioning and the inter-node coordination overhead that, at very large scale, creates latency and load imbalance in shared-nothing architectures.

With DASE, incoming messages write directly to DASE-backed flash storage, significantly reducing the replication traffic and consensus overhead of traditional Kafka clusters. Because durability is handled by VAST's storage layer rather than by redundant broker-level data copying, VAST Event Broker eliminates much of the replication overhead associated with traditional streaming platforms. The result is higher throughput and simplified operations. And because the storage layer handles resilience natively, those same event streams are immediately queryable for analytics and AI workloads, minimizing/reducing ETL steps.



Topics as Tables

In VAST Event Broker, each Kafka topic is implemented as a table in VAST DataBase. Messages produced through the Kafka API are transactionally persisted within that table, preserving the native Kafka message structure, including keys and values. Because topics are implemented directly as DataBase tables, event streams become immediately available across the VAST analytics ecosystem. SQL engines, DataEngine pipelines, AI workflows, and analytical applications can operate on fresh event data without requiring separate ETL pipelines. Topic tables also inherit the full capabilities of VAST DataBase, including ACID transactions, compression, encryption, erasure coding, and real-time upserts.



From Topics to Queryable Tables: JSON Expansion

When JSON messages arrive via the Kafka API, VAST DataBase automatically parses and expands those payloads into fully structured, queryable table columns without requiring a separate ETL pipeline, S3 staging layer, or Spark job. In a conventional streaming architecture, making event data broadly queryable requires a full processing pipeline: the event broker feeds stream processing, which writes to analytical storage, which is then queried by a separate query engine. With VAST Event Broker, that entire pipeline collapses, and events are stored directly as queryable DataBase tables the moment they arrive.

This is not simply faster event processing; it is a fundamentally different architecture that unifies the streaming and analytical layers. For workloads requiring additional query acceleration, VAST Sorted Tables can organize data according to common access patterns such as customer ID, event type, and geographic region, further reducing query latency while leaving the underlying topic tables unchanged.

Benchmark: VAST Event Broker Throughput

VAST recently evaluated the throughput performance of VAST Event Broker using a three-node cluster running production VAST software. The significance of this benchmark extends beyond raw throughput: the same event streams were simultaneously available as queryable VAST DataBase tables and capable of triggering DataEngine-powered pipelines in real time; a capability traditional streaming benchmarks do not measure.

Methodology

The test cluster consisted of 3 CNodes with AMD Turin processors, fronting a single flash enclosure housing 22 × 15TB NVMes and 8 × 800GB SCM devices.

Producer Settings

- 1,024-byte messages; 4ms linger time; 1MB batch size; idempotence off (not currently supported)
- No durability or resilience settings were modified for testing. VAST provides data protection through its underlying architecture rather than through broker-level replication or unsafe write configurations.

Consumer Settings & Topic Configuration

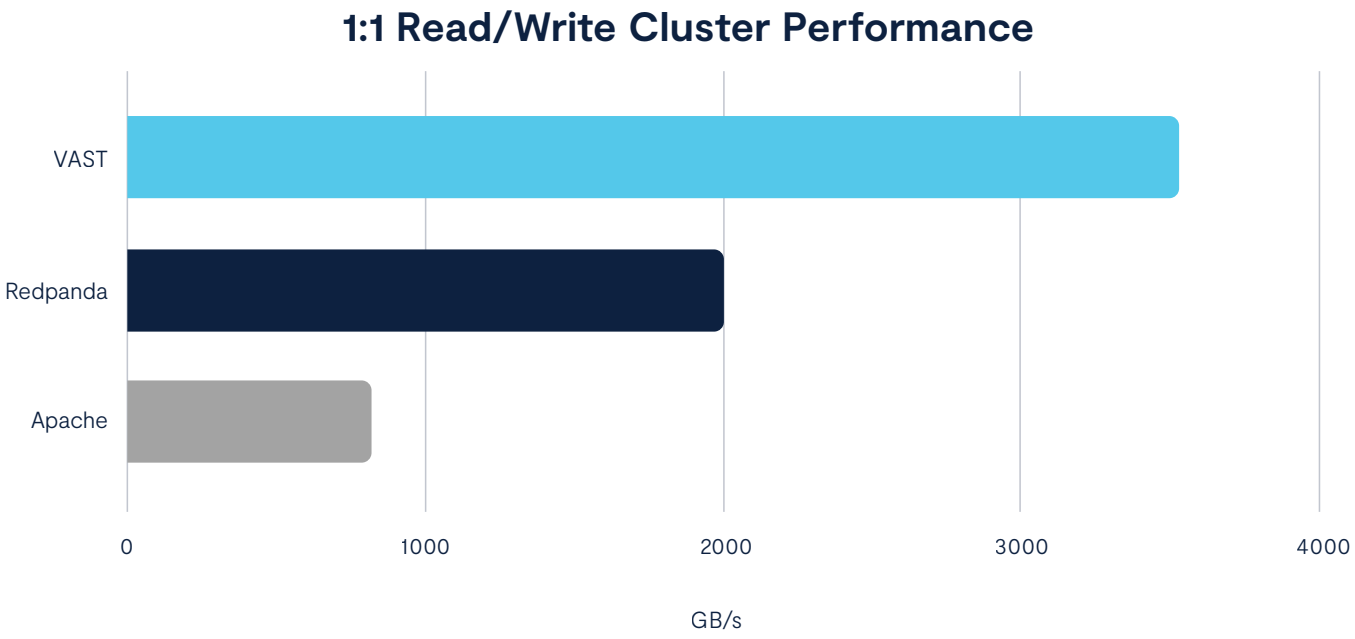
- Auto-commit off; consumers set to read from the earliest offset
- 192 partitions; message timestamps set to production time

Item	Quantity	Consumers
CNodes	3	2x AMD EPYC 9454P 48-Core Processor 376 GB RAM
DNodes	1	22x 15 TB NVMe 8x 800 GB SCM
Load Generators	2	22 x 15 TB NVMe SSDs 8 × 800 GB SCM (Storage Class Memory)
Networking	1 switch	2x 200 Gbps networking throughput

Three workload tests were conducted: a read/write workload (aggregate ingress and egress at maximum performance, held equal); a write-only workload (pure producer load); and a read-only workload (pure consumer load).

Read/Write Workload

During the read/write workload, VAST Event Broker sustained 3.54 GB/s (3,540 MB/s) of aggregate streaming throughput across the 3-node cluster, the scenario that most closely reflects real-world operating conditions. Redpanda reaches approximately 2 GB/s (2,000 MB/s) on a comparable 3-node AWS cluster; Apache Kafka sustained approximately 825 MB/s across 3 nodes in equivalent lab tests. VAST Event Broker's result represents approximately 1.7× to 4.3× greater throughput than these leading platforms on the same workload profile.



Unlike traditional streaming benchmarks that measure event transport in isolation, VAST Event Broker delivered these throughput levels while simultaneously making event streams available as queryable VAST DataBase tables and capable of triggering DataEngine-powered pipelines in real time.

Write-Only and Read-Only Workloads

In the write-only workload, the 3-node cluster sustained burst traffic of approximately 17 million messages per second for over thirty minutes, totaling more than 30TB of burst message traffic; sustained write throughput held at 4.75 million messages per second. In the read-only workload, VAST Event Broker sustained more than 12 million messages per second, achieving 12.57 GB/s of aggregate bandwidth across the cluster. These discrete read and write results are directly relevant to industrial and engineering environments, where high-velocity telemetry must be ingested rapidly during collection windows and replayed or analyzed as a separate downstream step, a workflow that benefits from strong read and write performance independently, and that VAST supports without requiring separate storage systems for ingestion and analysis.

Vertical Use Cases

VAST Event Broker's combination of high-throughput ingestion, immediate queryability, and event-driven pipeline execution addresses demanding real-time workloads across industries.



Financial Services: Real-Time Fraud Detection

Fraud detection models require access to transaction events the moment they occur. With VAST Event Broker, transaction events are immediately queryable and can simultaneously trigger DataEngine inference pipelines—enabling fraud scoring against live data without pipeline delay.



AI and Machine Learning: Feature Engineering

Real-time feature engineering depends on fresh event data. VAST Event Broker makes events immediately queryable at ingestion, enabling feature stores and training pipelines to operate against current data rather than data that is minutes or hours old.



Industrial and IoT: Telemetry Ingestion

Industrial environments generate high-velocity telemetry that must be ingested rapidly and analyzed as a separate downstream step. VAST's high write throughput handles burst ingestion efficiently while immediate queryability supports downstream analysis without separate storage systems.



Operational Intelligence: Real-Time Observability

Operations teams increasingly need to query event streams in real time to detect anomalies and investigate incidents. VAST Event Broker's topic-as-table model makes operational event streams directly queryable with SQL, reducing the time between event occurrence and analytical insight.

Summary and Migration Guide

Modern event-driven architectures must deliver value across three layers simultaneously: event transport, real-time analytics, and event-driven execution. VAST Event Broker addresses all three within a single platform:



Consolidated Infrastructure:
VAST Event Broker eliminates the external Kafka cluster and embeds event broker functionality directly within the VAST platform, removing the need to deploy, monitor, or support separate streaming infrastructure.



Real-Time Analytics:
Streaming data is immediately queryable as SQL tables without an ETL pipeline, enabling real-time operational analytics and event detection on live data.



Event-Driven Execution:
VAST Event Broker works with VAST DataEngine to power autonomous, event-driven pipelines that enable organizations to act on data in real time.

Migration

For organizations with an existing Kafka implementation, migration to VAST Event Broker requires changing broker endpoints rather than rewriting producers or consumers. A representative workload can be evaluated in a matter of days.

The recommended starting point is replacing a single Kafka topic or cluster; VAST does not require a full migration to demonstrate meaningful value. Relevant metrics to track during evaluation include read/write throughput, per-broker cost, and streaming-to-analytics latency for a representative event type. VAST's event streaming team can recommend appropriate sizing, configurations, and workload paths for your environment.

For twenty years, organizations treated event streaming, analytics, and operational systems as separate architectural domains. AI workloads increasingly require all three to operate on the same data in real time. VAST Event Broker represents a different approach. Event streams become immediately queryable data while simultaneously serving as triggers for DataEngine-powered pipelines and AI workflows. The result is a unified platform for event transport, real-time analytics, and event-driven execution.

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

06/25