White Paper

VAST DataBase:

Performance & Benchmarking

02/25 | Version 1.0.1

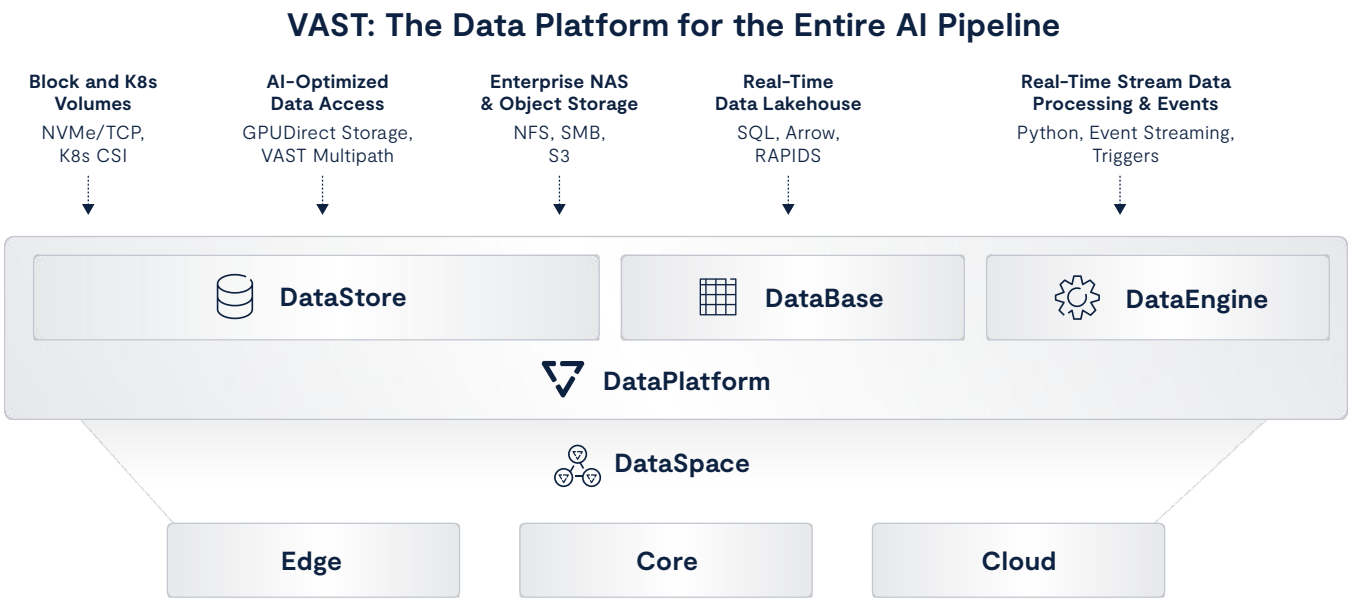
Table of Contents

Executive Summary

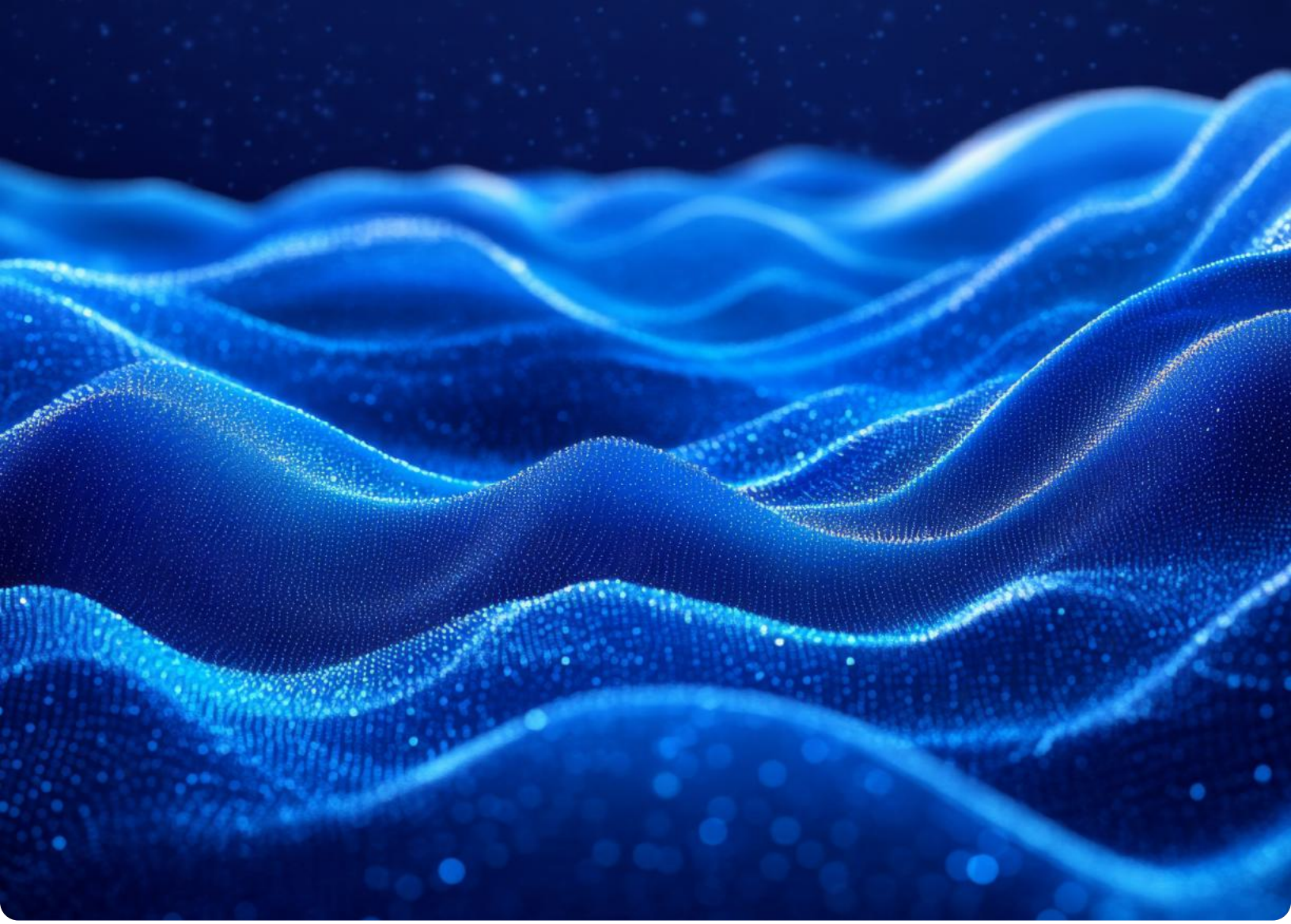
The data landscape is undergoing a significant transformation, driven by the increasing adoption of AI and advanced machine learning. As organizations seek to unlock the value of their data, they face many challenges. Traditional data infrastructure, often siloed and inflexible, struggles to meet the demands of modern data applications. As organizations move their workloads to the cloud, they seek solutions that can seamlessly integrate with their existing infrastructure and provide the scalability and performance required for AI and machine learning applications.

Recent advancements in the data ecosystem have resulted in a fragmented market for structured data, with specialized products addressing specific use cases. This fragmentation, combined with the push for composable data stacks, has increased complexity, driven up costs, and reduced efficiency. At the same time, the reliance on data lakes and advances in object storage have necessitated extensive data pipelines to move, transform, and prepare data for analytics. Open table formats like Apache Iceberg, Delta Lake, and Hudi were developed to bring structure to data lakes, promising to streamline these pipelines. However, in practice, they require extensive management and optimization for only modest performance gains. Even Iceberg, the most widely adopted open table format, demands routine maintenance and operational overhead to meet workload demands. Additionally, modern systems must seamlessly accommodate streaming workloads, further complicating the landscape with a proliferation of solutions—each with trade-offs in complexity, performance, and functionality.

VAST Data offers an alternative in its unified platform that addresses the challenges of modern data management. VAST simplifies operations and accelerates time to value by providing a single solution for a wide range of use cases and data workloads, including traditional data warehousing, analytics, AI/machine learning applications and streaming.



The VAST DataBase (VDB) is a flexible, high-performance database designed to meet the demanding needs of data in the age of AI. By focusing on best-in-class performance and scalability with its all-flash architecture, the VAST DataBase streamlines and accelerates the path to value for data while providing superior cost efficiency. This whitepaper presents the results of detailed performance benchmarking, demonstrating the database's exceptional capabilities in key areas that reflect modern data operations.



Findings

In this whitepaper we compare the VAST DataBase mostly to Apache Iceberg, a popular open table format that includes optimizations for object storage. Iceberg is flexible and can be seamlessly run in the cloud and in on-premises environments. Our goal is to assess the performance of VDB vis-a-vis object storage based systems (Iceberg being the table format) using standard warehousing benchmarks, high-stress filtering scenarios and transactional loads. Our findings are detailed in this writing, but here's the spoiler:

- VAST DataBase is 25% faster than Iceberg at executing TCP-DS benchmarks while using 30% less CPU to finish the task.
- VAST DataBase is **5-20 times faster** than modern object-based solutions at "finding needles in a haystack" operations while using 1/10 of the CPU, delivering small amounts of non-consecutive data at high throughput.
- Update and delete operations are up to **60 times faster** with VAST than object-based solutions due to the efficiency and design goals of VAST's architecture.

While this paper concerns itself with performance, it's important to note that the VAST architecture supplies enterprise-class scalability, data resilience and high-availability at an unbeatable TCO. The aphorism "cost, performance, reliability: choose two" no longer applies - VAST gives you all three.

Project Introduction

The VAST DataBase (VDB) is a modern hybrid transaction/analytical processing (HTAP) database that leverages the scalability, resilience, and availability of VAST infrastructure to realize a high-performance structured data system that can power almost any use case. It interacts with a variety of data processing and streaming engines, allowing them to offload fundamental activities best handled adjacent to the storage - scanning, filtering and projecting (search) - so the engines can do what they do best: analysis. This means you can use the right tool for the job, maintain total data coherency with other tools, while reducing the overall costs of analytics end-to-end.

Some common target use cases are:

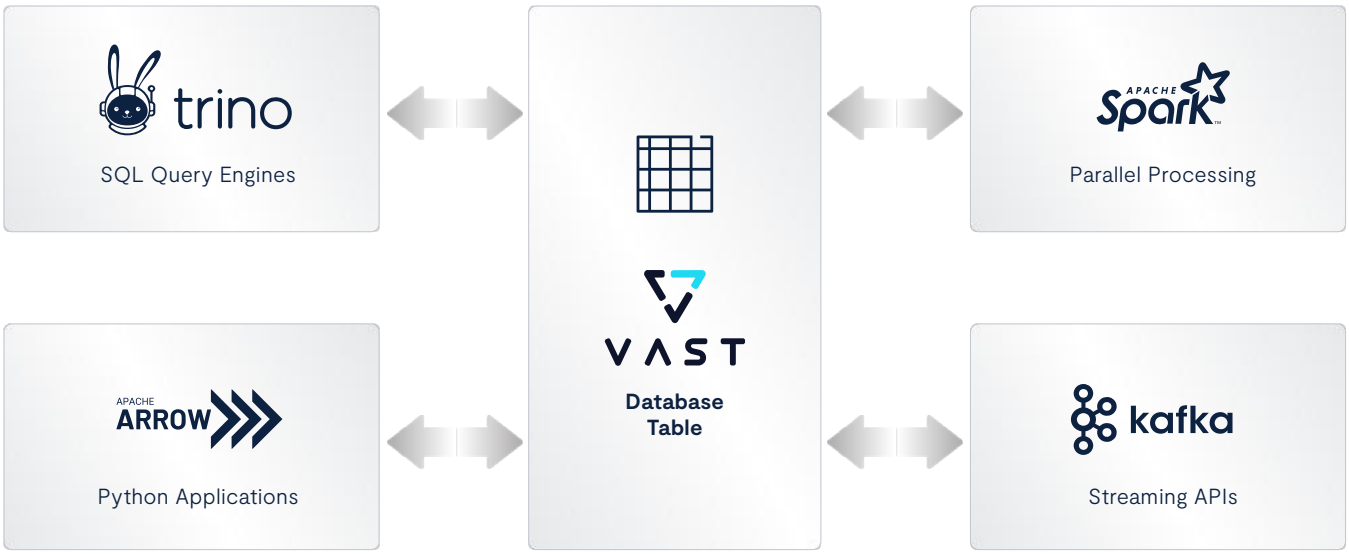
- Traditional “big data” processing, including data warehousing and reporting
- High transaction-rate data ingestion, real-time analysis and processing - common in ad tech, fin tech, web-scale and industrial verticals
- Log store, event, and packet capture for real-time analysis and processing in cybersecurity (SIEM)
- AI/ML data and metadata use cases

The VAST DataBase can serve a broad array of use cases because it advances the storage and retrieval of structured data on an all-flash architecture. Ultimately it allows you to use the proper tools for your job, be it for analysis or operations.

For instance, a high transaction rate IoT use case might:

- ingest data directly into a VAST DataBase table via Kafka APIs,
- while Spark is used to enrich the data set,
- while Trino is used to generate reports,
- all working on the same physical table at the same time.

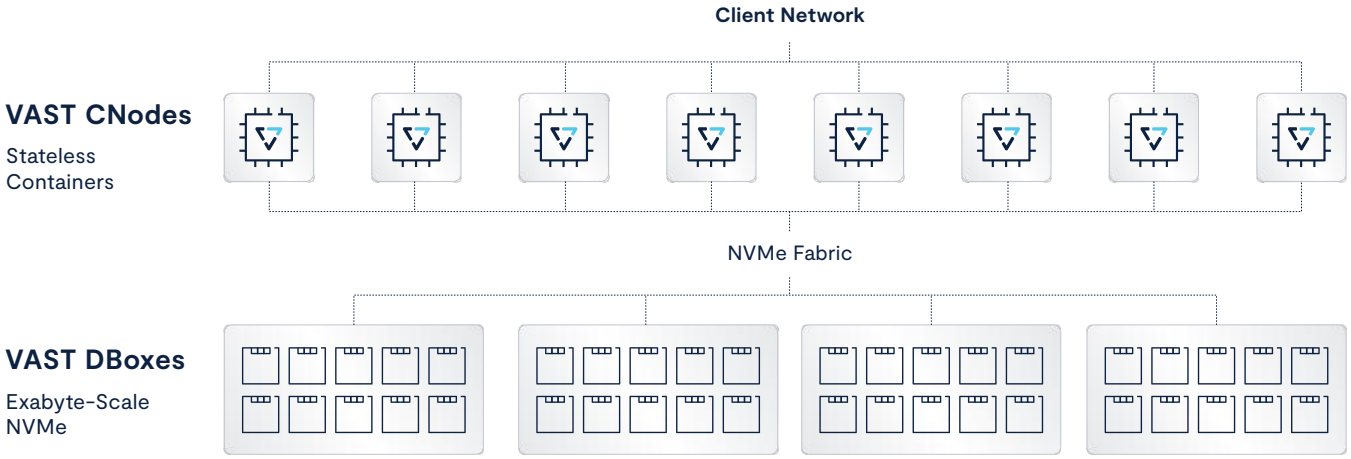
This can be done with read/write transaction-level coherence between processing clients: the VAST DataBase is ACID-compliant across tables and across processing engines.



Architecture

The DataBase leverages VAST's disaggregated shared-everything architecture (DASE), a system of tightly integrated storage enclosures and compute nodes, connected with a high-throughput, low-latency network (RDMA Ethernet or Infiniband). The architecture can be deployed as software on customer hardware or through VAST's hardware partners for an appliance experience. With DASE, VAST customers can expect:

- All-flash performance
- Independent scaling of compute and storage capacity
- Advanced data reduction: deduplication, object compression, and global compression
- Write-shaping to read-intensive media for lower cost per TB for raw storage
- Enterprise-class management and security features
- Low and zero-config ease-of-use



VAST's Disaggregated, Shared-Everything (DASE) Architecture

DASE architecture details can be found in the VAST Data Platform whitepaper.

Connectivity Ecosystem

VAST directly supports a number of popular data processing engines and protocols:

- **Kafka APIs:** VAST implements a high-performance Kafka interface (written in C++) for producing and subscribing to data streams. VAST does not repackage any open-source Kafka code.
- **Trino:** a high-performance, open-source, massively-parallel processing (MPP) engine for SQL that supports dozens of data sources, including VDB. It can handle data warehousing and ad-hoc querying & reporting.
- **Starburst:** A multi-user data analytics platform (using Trino as the core engine) with security and governance features appropriate for enterprises and governments.
- **Spark:** A common platform for parallel data processing.
- **Dremio:** A high-performance processing engine with security and governance features appropriate for enterprises and governments.
- **VAST Python SDK:** A Python interface that presents the VAST DataBase as PyArrow elements, allowing the Python ecosystem to be fully leveraged for AI/ML.

Together with these native integrations, a complete ecosystem of data management and visualization tools becomes available. If your tool can speak to Kafka, Trino, Dremio, Spark or can interface with Python, you can use it with the VAST DataBase.

Kafka

Kafka is unique in the above list because it is a protocol rather than a high level processing engine or lower level SDK. In case it wasn't clear: Kafka APIs can be used to stream directly to VAST DataBase tables without any intermediary agent. Resilience and scaling of Kafka storage is handled by VAST's underlying architecture, obviating the complex and fragile system of shards and replicas used by the reference Java implementation. In this architecture, Kafka becomes lightning fast, infinitely scalable, easy to manage, and worry-free.

Performance Benchmarking Methodology

Overview

The goal of this whitepaper is to present the performance of the VAST DataBase and compare it to an object storage backed format or data lake. We will be using Iceberg as the table format since it has become standard in the modern data stack. To ensure realistic and reliable performance benchmarks, we built two environments: one built on VAST DataBase using an on-prem VAST cluster, and one on object storage, leveraging Iceberg and built in AWS. Both clusters use Trino as the query engine. Trino is an open-source data federation layer that presents an SQL environment for a variety of data sources, including Iceberg on object storage and the VAST DataBase.

By using the same compute layer against the two different storage layers, with similar architectural elements, we are able to get as close as is reasonably possible with the two environments for an equitable comparison.

Performance Testing Environment

In this performance study we try to use consistent and equivalent hardware, software, and configurations in our benchmarking environments. Notes and statements:

Software

The VAST DataBase is a low-to-zero configuration solution, while Iceberg typically involves significant tuning for specific workloads. Efforts were made to optimize Iceberg by including features like the optimized Parquet reader with bloom filtering, using sensible options for Parquet file size (1GB) and selecting a fast compression algorithm (SNAPPY). No configuration or data optimizations were made on a per-workload basis.

Software versions are identical for all tests where applicable:

- Trino 429
- VAST 5.1

> **Use of Trino and Processing Engine Notes**
>
> Trino was leveraged as the engine for all tests as it has a modern Iceberg implementation and VAST natively supports it with a plugin. In recent versions, Trino has become a fast, consistent and stable performer making it an excellent choice for SQL data processing.

Data Arrangement

- There were no partitions used in any Iceberg test, and similarly no projections or partitionings were used on any VAST data
- No benchmark table data was sorted in any way for any test

Hardware

- AWS instances were selected to have CPU parity with on-premises resources, electing to error in AWS's favor on selection (AWS instances have faster CPUs).
- The VAST lab networking infrastructure is superior to AWS internal networking. AWS infrastructure cannot be fully known, it has a significant software element, and it will have region-to-region and availability zone differences.

Ultimately, there can never be absolute equivalency between environments but we have attempted to account for the factors within our control. Variability is a reality in cloud environments.

VAST Environment and Infrastructure Details

Performance tests with VAST DataBase were executed on the following hardware.

Storage System

Standard 3x1 VAST architecture:

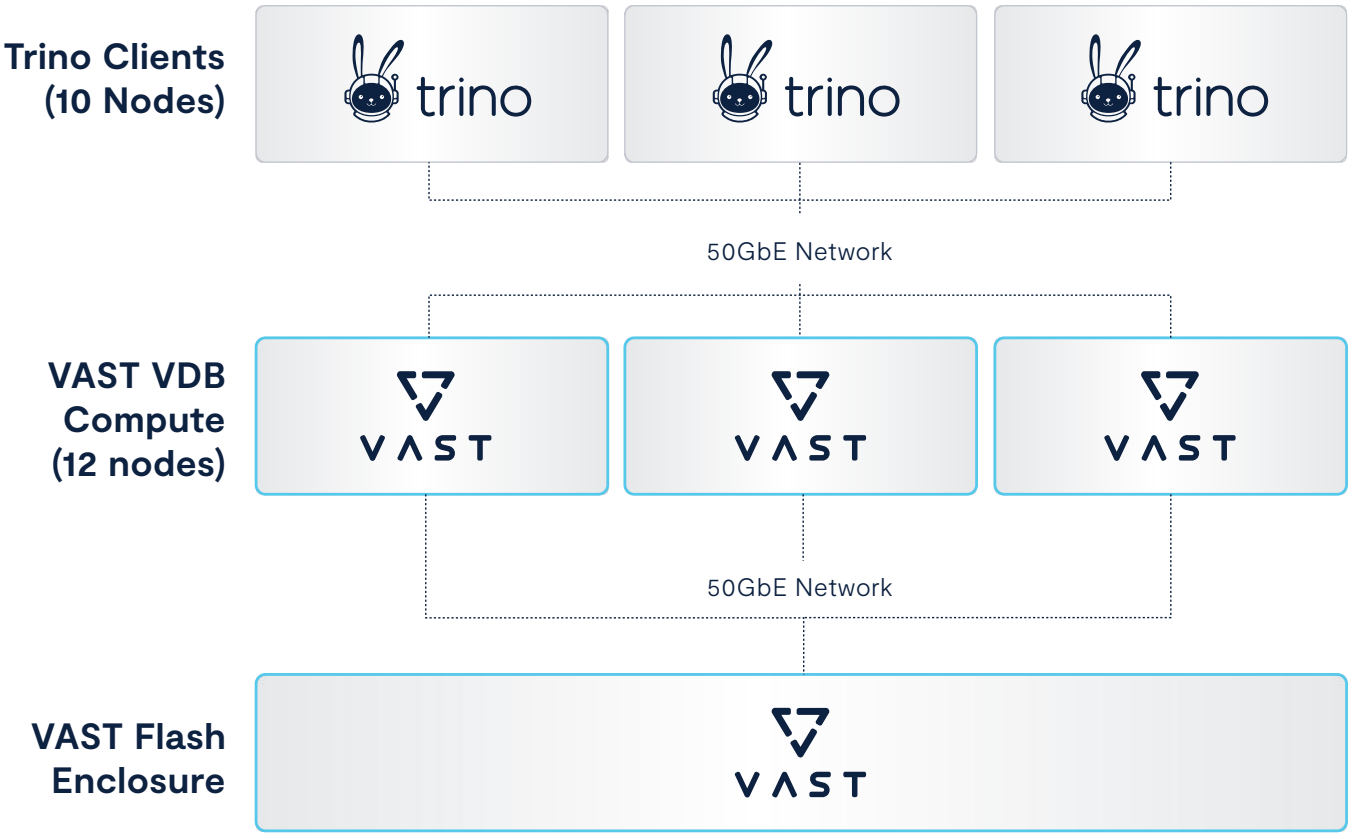
- VAST 5.1 with default install configuration
- 3 compute enclosures (12 nodes)
  - 2 x Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz (Ice Lake, 32 physical cores, 64 logical)
  - 256 GB RAM per node (~3TB total)
- 1 Ceres flash enclosure (AArch64)
  - 22 x 15.363 TB read-intensive NVMe
  - 8 x 800GB SCM

Client Systems

- Trino version 429
- 10 compute nodes
  - 2 x Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz (Ice Lake, 32 physical cores, 64 logical)
  - 256 GB RAM per node (~2.6TB total)

Network

- 50 GbE Client network
- 100 GbE Enclosure/compute fabric



AWS Environment and Infrastructure Details

Comparative analysis for the AWS environment was performed in EC2 using native in-region S3 for storage. For these tests, a single instance type was leveraged that closely resembles the systems used in the VAST environment described above (without resorting to high-cost specialized instances). These instances had faster CPUs with slower network connections than the VAST environment. The AWS architecture for testing is:

- Trino version 429
- 10 x M6i.16xlarge instance type
  - Platinum-class Ice Lake Intel processors
  - 64 vCPUs
  - 256 GB RAM
  - 25 GbE networking

Note on networking in AWS:

One method of increasing aggregate bandwidth between storage and compute nodes is to use a larger number of smaller instances. This ultimately reduces performance due to an increase in cross-talk between nodes. A reasonable trade-off was made by using the M6i.16xlarge instance type, which in our estimation is optimal for the intended benchmarks.

Tests and Methodologies

Summary

Our goal in the benchmarking tests was to emulate typical queries that would be performed in a variety of analytics scenarios. These scenarios represent compute-heavy analysis/warehousing, selective projections, a demonstration of the effects of updating/deleting data in a table, scaling effects and GPU acceleration effects.

- **TPC-DS**
  - sf1000 (1TB)
    - Concurrency: 1, 2 and 4
- **Filter benchmarks**
  - rf1000000 (1 trillion rows)
  - rf100000 (100 billion rows)
  - rf10000 (10 billion rows)
  - rf1000 (1 billion rows)
- **Data alteration benchmarks**
  - Update
  - Delete
- **Scaling comparisons**
  - Workloads on 8, 4 and 2 execution nodes
- **NVIDIA RAPIDS on Apache Spark**
  - Accelerated and unaccelerated VAST DataBase benchmarks
  - Accelerated and unaccelerated Hive benchmarks
  - Reference benchmarks for unaccelerated Iceberg

TPC-DS Benchmarks

TPC-DS is a widely accepted benchmark for measuring traditional data warehouse performance. It applies a workload that stresses CPU, memory and I/O subsystems. It is used to assess the ability of the database to perform various complex functions that are important to decision support.

TPC-DS consists of a set of tables that simulate a retail sales decision support system consisting of 25 tables of inventory, transaction, and support data. This data set comes in scale factors that align roughly with the logical size of the database in gigabytes. Scale factor 100 is a 100 GB dataset, scale factor 1000 is 1000 GB, and so on. A set of 99 pre-defined queries, executed in random order, comprises a query stream. Query streams can be executed at a concurrency factor where concurrency "1" is a single stream of queries, concurrency "2" is two simultaneous streams, and so on. A benchmark result is the time it takes to execute all of the query stream(s) at a given concurrency factor, on a data set of a given scale factor. Results below are on a 1 TB (sf1000) data-set at concurrency factors of 1, 2 and 4.

The 1TB scale factor was chosen because it is large enough to model a realistic use-case – billions of rows of transactions – while being small enough to test several concurrency factors with a reasonable amount of hardware.

Filter Benchmarks

A particularly common use case in modern analytics is to run highly filtered queries on a large volume of data. For example, rapid searching for specific records in a large log stream which is common in security information and event management (SIEM). Similarly, the ability to rapidly down-select on large amounts of uncurated data can enable real-time analytics on transaction data, skipping intermediate data engineering tasks that can delay time-to-insight and lose data fidelity.

To profile this type of capability on various platforms, a filtering benchmark was designed to measure the raw performance of scan/filter/project operations on standardized, unsorted data. Performance under these circumstances is affected by several factors to be measured in the workload.

- Bandwidth between the data and the compute performing the filtering operation
- Efficiency of comparison algorithms within the compute
- Data arrangement and accelerators in storage (row grouping, data structures, statistics, etc)

Data Generation

Data generation for filter benchmarks is done in a combined scale that includes a width factor (number of columns) and a row factor.

- The width factor determines the number of columns in a test table with a set ratio of data types that include int, int64, float, float64, boolean, and varchar data types.
- The row factor refers to how many millions of rows are generated (row-factor 1 = 1,000,000 rows).
- Additional factors can be set for sparsity and cardinality, however multiple factors for these parameters are not used in these benchmarks (coming soon).
- For these workloads, each row is generated with random data.
    - For integers and floats, a pseudo-random value is generated within the range of possible values for the bitwidth
    - For booleans a coin is flipped
    - For varchar there are two types of values:

        i. Values consisting of 6 randomly selected words from a list of 50,000 words

        ii. A single ID data type for each row that is a 16-character hex value in string form

The end result is that these tables can be queried for patterns and ranges in a way that predictably filters for specific amounts of data while keeping I/O random – sequential access occurs only as row groups and random chance allow.

Measurements

Once the data is generated, keyspace filters on 64-bit integers are used to request a percentage of the table rows, returning all columns. The results published here are grouped into two major sections. A result set that selects 0.01% of the table data per query, and a set that selects 0.001% of the table data.

- Filtration factor 0.0001 (0.01%) sits at a near-optimal point for Iceberg vis-a-vis VDB given available bandwidth and CPU.
- The second set, at filtration factor 0.00001 (0.001%), shows a narrower selection where network bandwidth restricts object-storage performance while VDB performs optimally.

Data Modification Benchmarks

The third and final set of benchmarks in this study focus on update and delete operations, wherein data is modified or removed. This is common in CDC operations, and prevalent in data pipelines. Simple record update performance is measured, alongside simple record delete performance. Unlike warehouses or lakehouses built on object storage, VAST can tolerate small updates without altering the performance characteristics of the target table over time. Object storage will incur a heavy write amplification penalty or an object accumulation penalty for minor updates. VDB data structures avoid cumulative impacts and a simple workload was devised to demonstrate these effects.

Update workload

The update workload is a simple test:

- A 10 million row target table is generated with random data implementing a keyspace exactly like the filtration benchmark above.
- The table is updated with a static value for one column using a predicate that selects roughly 100 records using a second column for filtering. ("`UPDATE [table] SET col1=42 WHERE col0 BETWEEN [val1] AND [val2]`")
- This update is executed 1000 times on different sections of the target table altering ~100 records at a time until 100,000 records have been updated.
- Each update operation is timed

This workload forces two things to happen on each iteration:

- Search for the location of applicable records (a filter operation)
- A minor update to a cell within the selected records

Any system using an object backend needs to:

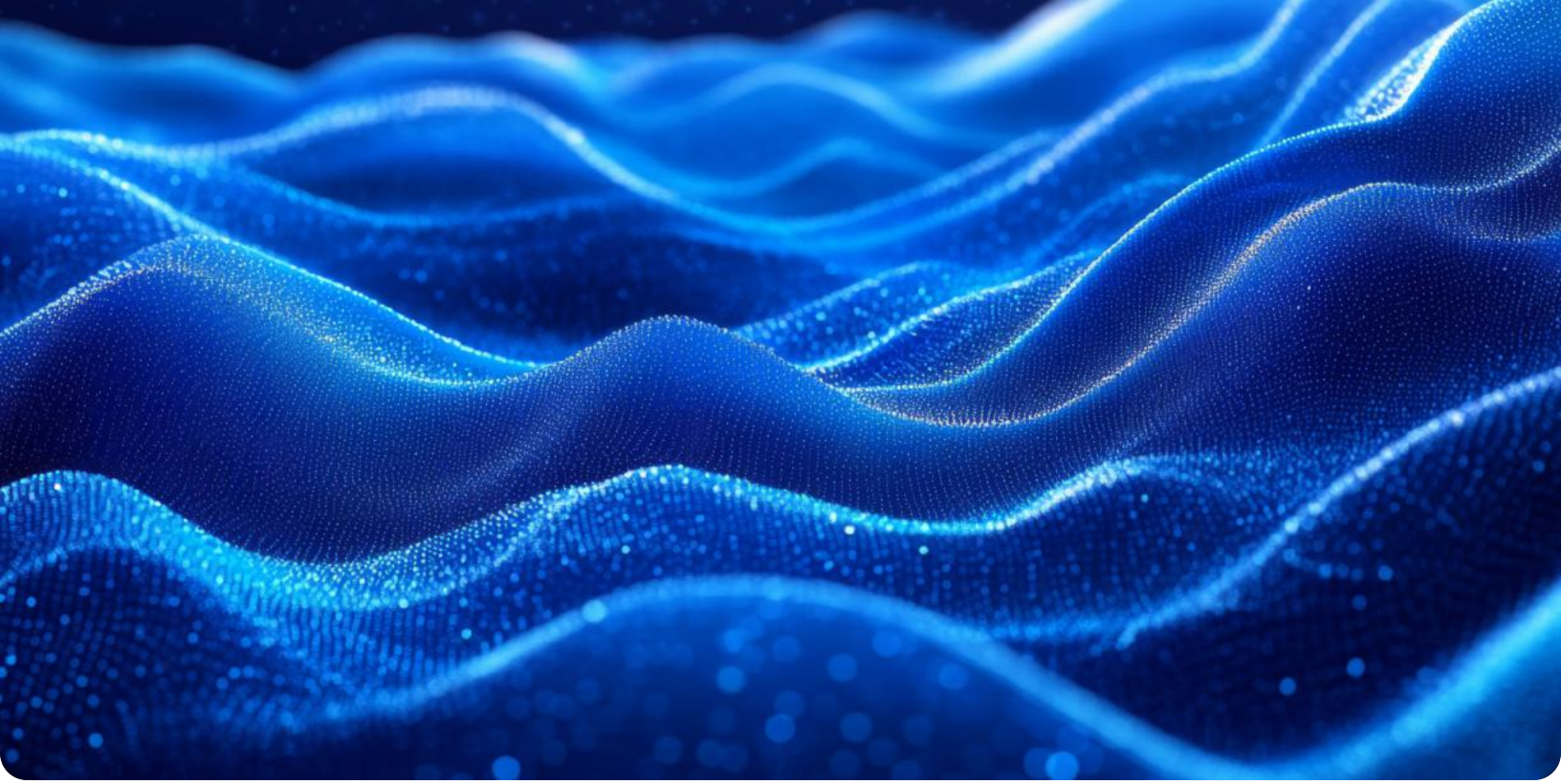
A. Rewrite entire container objects (normally hundreds of megabytes) for every affected container (usually Parquet), or

B. Create a new object with the altered data, masking the affected row(s) in the existing containers (this is the default behavior for Iceberg in Trino)

Option A would be entirely unworkable in this case.

For option B, small objects accumulate and the processing engine is forced to consider them for every subsequent update, thereby adding another object that needs its headers scanned. This ultimately slows down table performance until maintenance operations are run, which will rewrite the dataset to consolidate the objects. Write amplification is minimized in exchange for increased IOs and reduced read performance.

Things are simple for VAST. In storage, tables are virtualized in shallow B-tree structures and data blocks are stored in ~32KB chunks. When updates are made to the table, each affected row change will incur a write-amplification of no more than 32KB. This update is done in-line and maintenance operations don't need to be periodically triggered according to workload.



Delete Workload

Similar to the update workload, a delete workload is run for benchmarking with the following operation:

- A 10 million row target table is generated with random (high-cardinalty) data
- Records are deleted, roughly 100 records at a time, until 100,000 records have been removed
- Each delete operation is timed

This is very similar to the update benchmark workload above. On an object storage based system, each delete operation writes a file to "mask" the record(s) housed in existing Parquet objects. This results in object accumulation that forces more header scans, negatively impacting subsequent filter operations.

Once again, this is not how VAST DataBase deletes records; records are removed in-line with the operation without creating a complex network of tombstone objects that need to be scanned on subsequent queries.

Scaling Tests

A TPC-DS workload was executed under several hardware configurations to demonstrate the effects of scaling the compute and storage elements of the VAST test environment. This test demonstrates the asymmetrical impact on performance between VDB and Iceberg as the compute cluster size is reduced.

RAPIDS Tests

Deviating slightly from the rest of the workloads run in this paper, we're going to include a compelling benchmark run on Spark (rather than Trino) that combines VDB and NVIDIA RAPIDS. RAPIDS is a set of libraries that interface with GPUs via CUDA, an SDK for creating GPU accelerated applications. Part of RAPIDS comes as a plugin for Spark which, by extension, accelerates SparkSQL workloads. RAPIDS can offload a lot of compute tasks in analysis to GPUs in a way that is semi agnostic to the catalog in use. This means that it compounds the acceleration you get from VDB. RAPIDS speeds up computations, VDB reduces the data set needing processing. In short: they're two great tastes that taste great together.

The workload was run on Spark 5.4.1 on very modest hardware with 64 vCPUs, 1TB of memory and one NVIDIA T4 GPU.

Performance Results

Performance results from the described tests are summarized here. Included are data visualizations where appropriate, along with a brief analysis.

TPC-DS Benchmarking Results

The chart below shows the time to complete the TCP-DS benchmark at scale factor 1000 (1TB) with concurrencies of 1, 2 and 4 query streams in the VAST and AWS test environments (described in “Performance Testing Environment”).

Raw Benchmark Timings:

VAST DataBase

VAST DataBase TPC-DS Timings and CPU

Concurrency	Timing (sec)	CPU Seconds
1	588.72	253.1496
2	817.02	433.0206
4	1384.95	844.8195

Apache Iceberg on AWS/S3

Apache Iceberg TPC-DS Timings and CPU

Concurrency	Timing (sec)	CPU Seconds
1	696.41	306.4204
2	1093.06	644.9054
4	1909.31	1298.3308

Performance Deltas

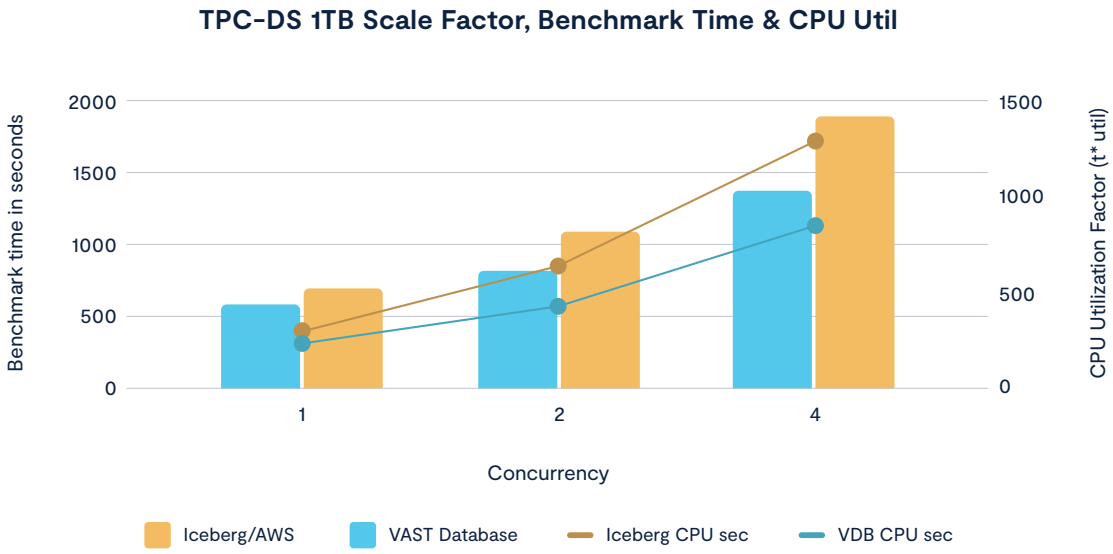
Timing and CPU Util. Differences

Concurrency	Timing diff	CPU util diff
1	15.46%	17.38%
2	25.25%	32.86%
4	27.46%	34.93%

Visualizations

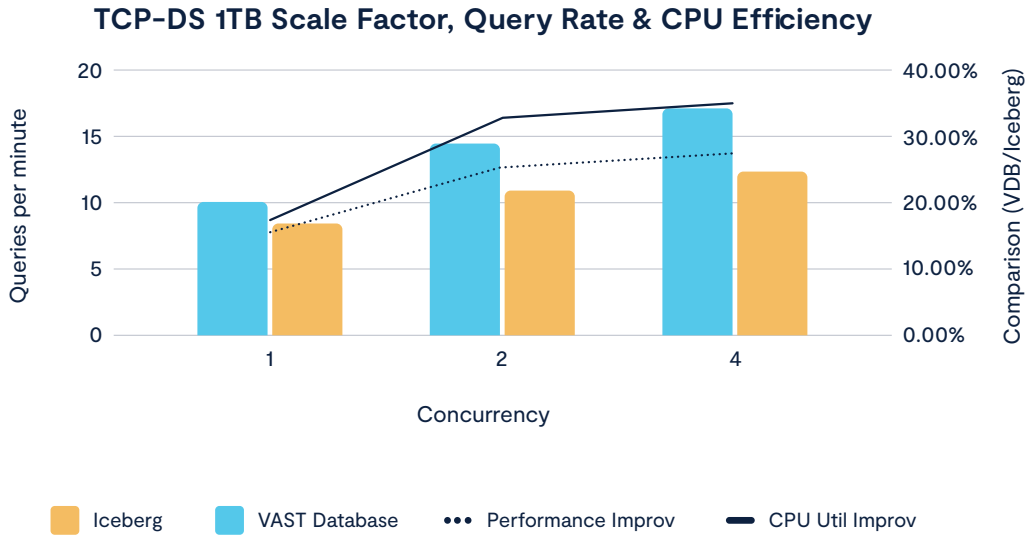
Benchmark timing and CPU utilization:

- The left y-axis shows time to complete the benchmark. Lower numbers are better numbers here, meaning less time to complete the benchmark.
- The right y-axis shows CPU-seconds consumed by the job. Lower numbers are better numbers here as well, meaning less CPU is used.



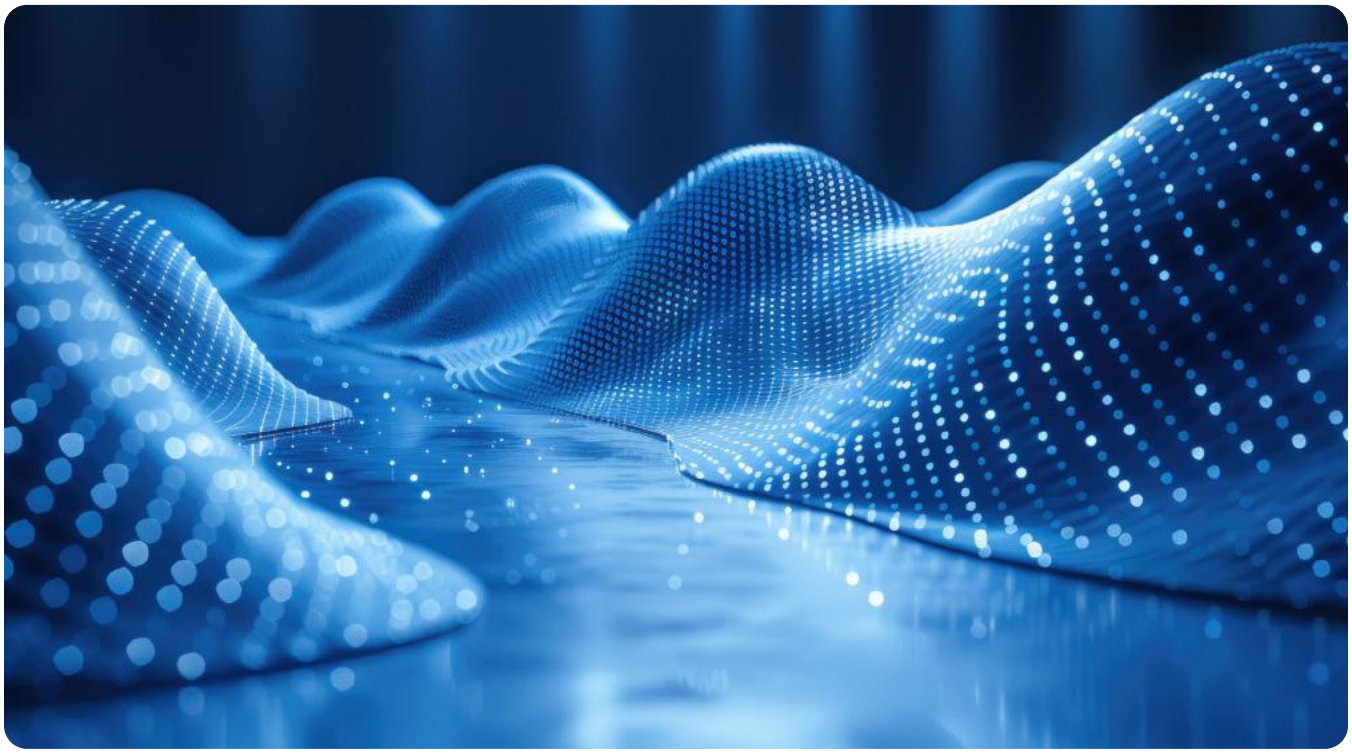
Queries per minute and improvement factors:

- The left y-axis shows the same timing data as above but presented as queries per minute. Higher numbers are better numbers here, meaning more queries are completed per time period.
- The right y-axis shows timing and CPU utilization improvements. Higher numbers are better here as well, meaning the performance and CPU utilization is a percentage factor better in VAST DataBase than in object storage.



Observations

- The VAST DataBase table format realizes significant performance improvements over object storage for warehousing use cases on data that is already curated and thus not highly selective. This is particularly true as the concurrency increases.
- CPU utilization per unit of work is 35% better with VAST DataBase at concurrency=4, indicating better handling of parallel requests as well as better utilization of hardware. In short, VAST is going to perform better in high-concurrency environments.



Filtering Queries

Filtering benchmark results are presented here across the following domains:

- Two filtration factors: 0.01% and 0.001% (under their respective sections)
- Four table length factors: 1, 10, 100 billion and 1 trillion, per table

Each filtration factor section presents:

- Raw figures
- Performance chart depicting rows per second with a difference multiple for series comparisons
- Chart depicting network throughput between engine and storage
- Chart depicting CPU utilization for benchmark runs

Benchmark results, 0.01% Selection

Filter results are expressed in rows-per-second processed by Trino for a filtration factor of 0.0001 (0.01%) on tables with scale factors ranging from 1 billion to 1 trillion rows.

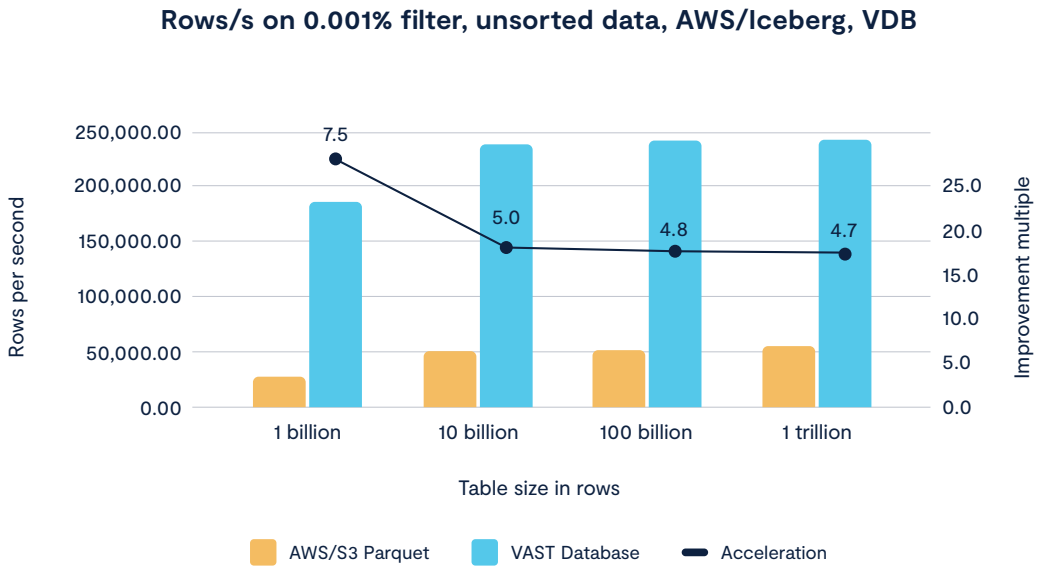
Data Throughput

Results on a predicate selecting 0.01% of 1, 10, 100 billion and 1 trillion row tables.

FIltering workload throughput in rows per second

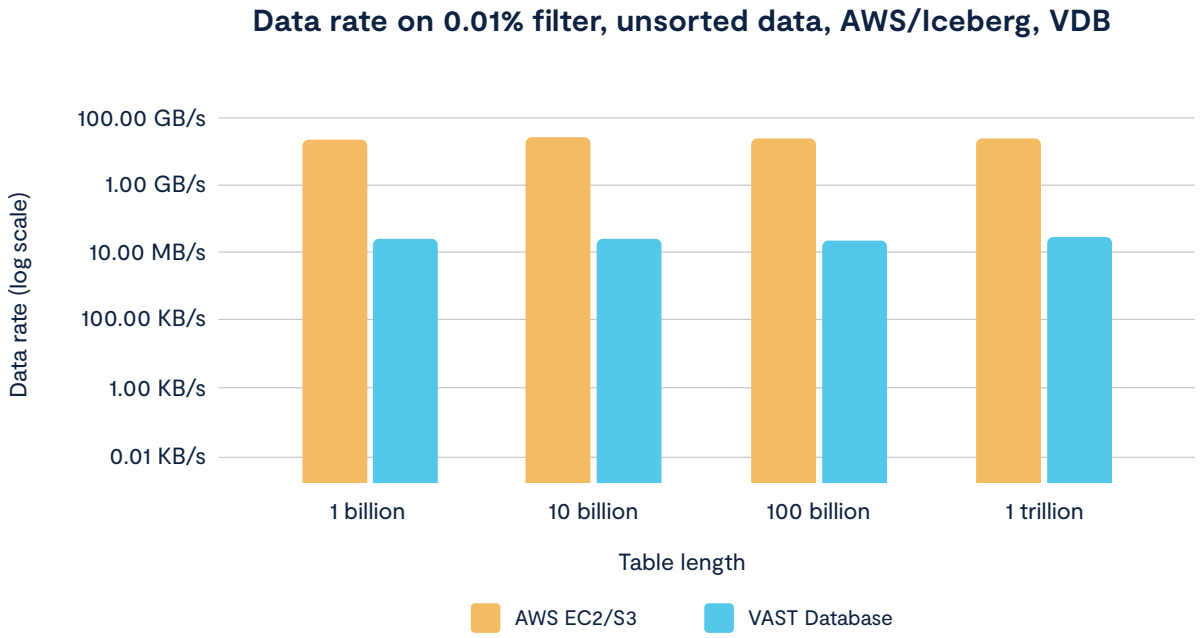
Table length	AWS EC2/S3	VAST DB	Acceleration
1 billion	24,963.62	186,979.01	7.5
10 billion	48,308.22	240,110.24	5.0
100 billion	51,453.67	245,128.50	4.8
1 trillion	51,980.24	245,353.99	4.7

- Throughput rate in rows per second on the left y-axis. Higher numbers indicate higher throughput.
- Rate difference multiples between Iceberg and VAST DataBase on the right y-axis (vast-rate/iceberg-rate).



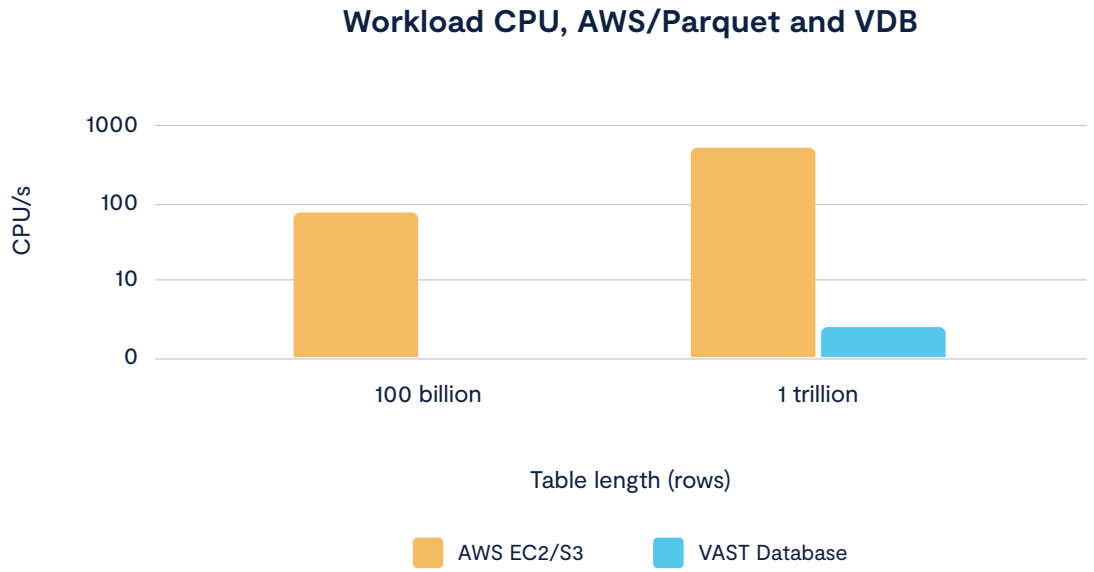
Network Data Rates

The left y-axis is expressed in bytes per second on a logarithmic scale. Lower numbers indicate less data being transferred across the wire. *Note that the scale is logarithmic for visibility and that the traffic generated when accessing VDB tables is fractional compared to object storage.*



CPU Utilization

CPU utilization was captured for benchmarks on 100 billion and 1 trillion row tables. The following chart shows the CPU seconds (per CPU average) used by Trino to complete the filter task.



Benchmark results, 0.001% Selection

Filter results are expressed in rows-per-second returned from the system for a filtration factor of 0.00001 (0.001%) on tables with scale factors ranging from 1 billion to 1 trillion rows.

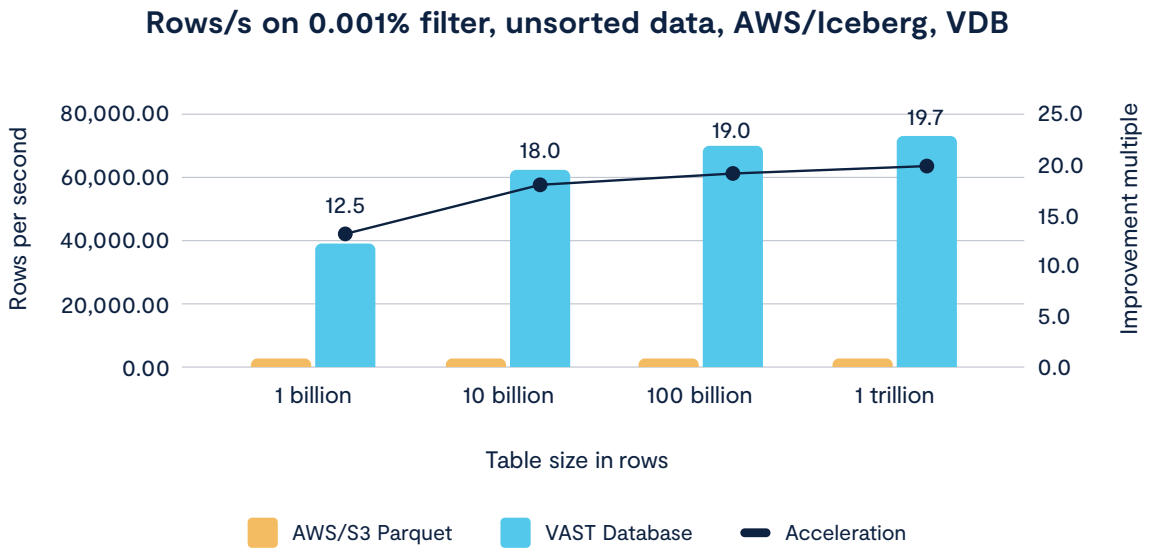
Data Throughput

Results on a predicate selecting 0.001% of 1, 10, 100 billion and 1 trillion row tables.

FIltering workload throughput in rows per second

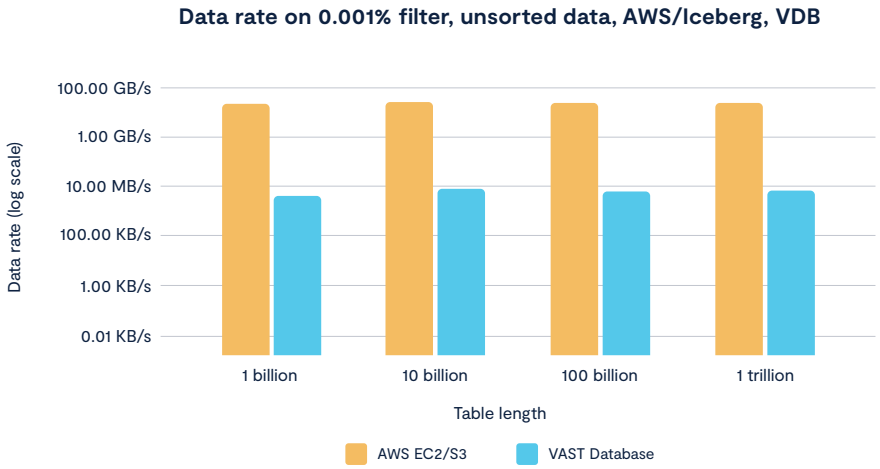
Table length	AWS EC2/S3	VAST DB	Acceleration
1 billion	3,108.68	38,835.59	12.5
10 billion	3,464.49	62,213.86	18.0
100 billion	3,691.31	70,169.36	19.0
1 trillion	3,734.14	73,426.46	19.7

- Throughput rate in rows per second on the left y-axis. Higher numbers indicate higher throughput.
- Acceleration factors (improvement in multiples) between Iceberg and VAST DataBase on the right y-axis. Higher numbers indicate higher multiples of performance in VAST DataBase as compared to Iceberg on AWS.



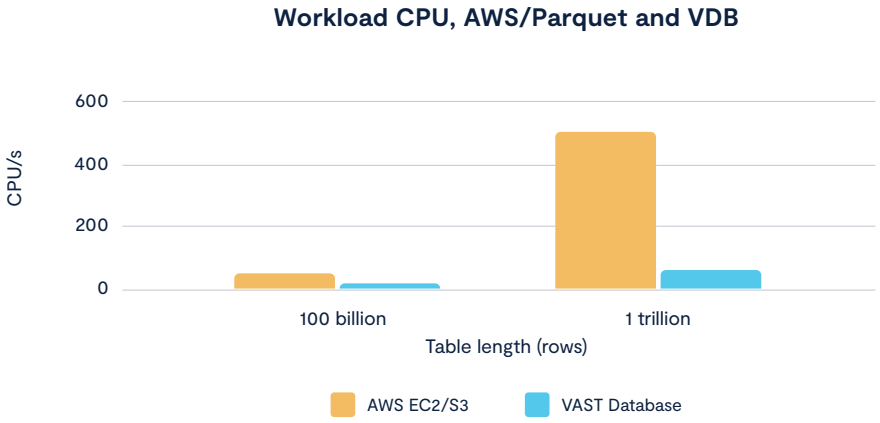
Network Data Rates

The same benchmark runs measuring the bandwidth between the storage system (S3, VAST) and compute cluster. The left y-axis is expressed in bytes per second on a logarithmic scale. Lower numbers indicate less data being transferred across the wire. *Note that the scale is logarithmic for visibility and that the traffic generated when accessing VDB tables is fractional compared to object storage.*



CPU Utilization

CPU utilization was captured for benchmarks on 100 billion and 1 trillion row tables. The following chart shows the CPU seconds (per CPU average) used by Trino to complete the filter task.



Filter Benchmark Observations

VAST DataBase exhibits profound performance gains compared to Iceberg on S3 when running highly selective (filtered) queries. This advantage manifests for 2 reasons.

1. When object storage-based table formats are used, processing engines must pull large quantities of data across standard networks to filter information. The use of sorted tables, partitioning and configuring to the workload (parquet file size, row group sizes, paging, etc) are techniques that can be used to deal with these problems in the real world, however there are trade-offs for each optimization.

2. In contrast, VAST handles data selection and projection within the storage system on a small but highly performant backend network, using well designed data structures optimized for locating data in this environment. The processing engine quickly gets only what it needs avoiding bottlenecks on standard networks.

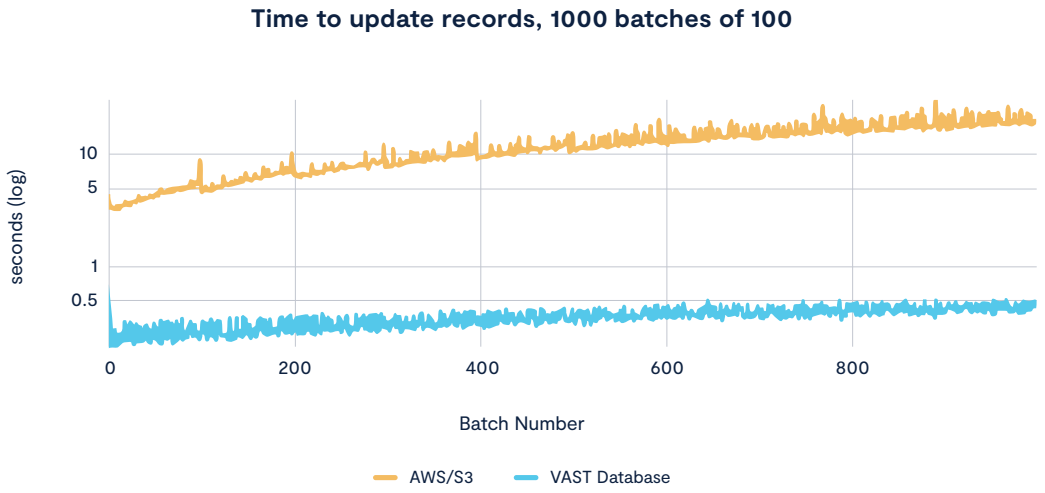
The use of two filtering factors in the above benchmarks is intended to demonstrate how much network throughput limits selective operations. This is a fundamental problem in the design of data presentations for BI, forcing the use of data engineering tasks for every situation in which you might want to use your data. Removing this bottleneck makes VDB five times faster than object for the 0.01% filtration factor to almost 20 times faster for 0.001%. Since VDB can rapidly down-select through large amounts of data, it can drastically simplify operational presentations. Or perhaps more importantly, it can drastically accelerate the development of operational presentations.

Data modification benchmarks

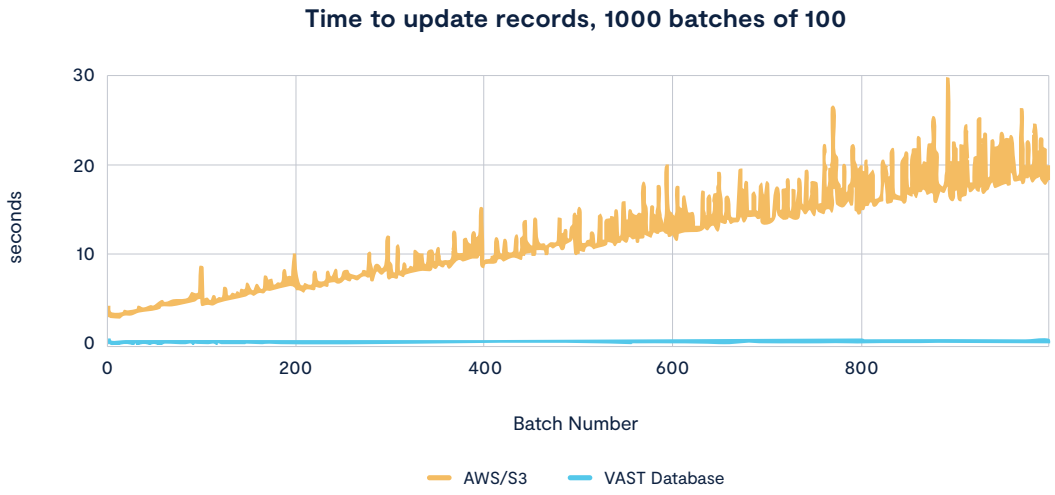
Data modification benchmarks include two very simple workloads where data is updated and deleted in serial operations (described in "Tests and Methodologies").

Update benchmark results

The update workload results below show the time (in seconds) it takes to update a block of 100 records over a 1000 iteration run. The y-axis is logarithmic to make the VAST and AWS/S3 results visible on the same chart.



The same data is presented here on a linear y-axis to visualize actual performance differences.



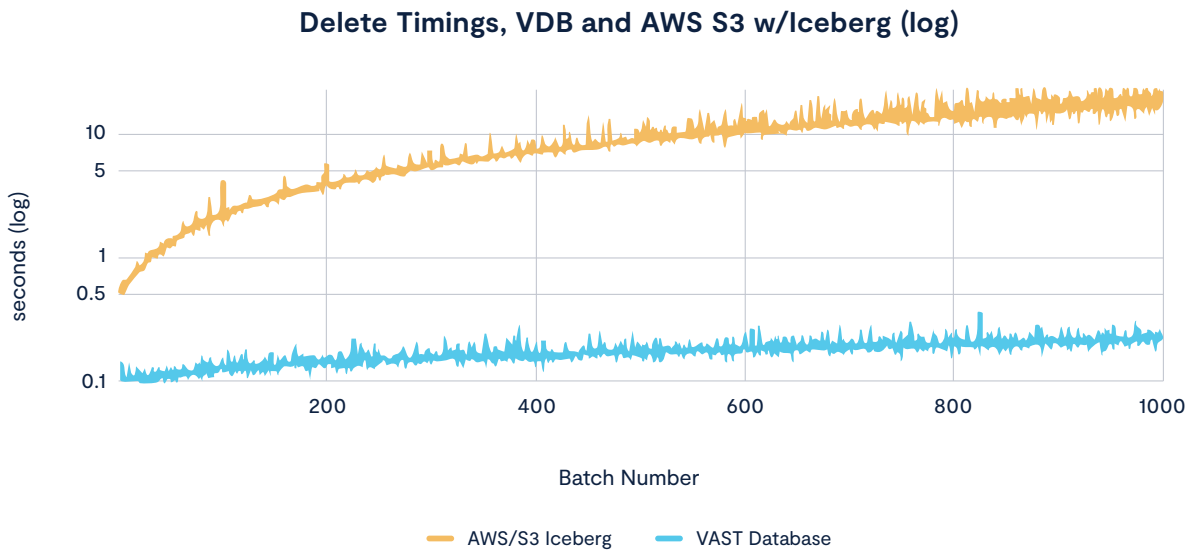
Observations

These results are simple to understand. As discussed above, an object storage environment incurs a latency penalty for each update to begin with. For Iceberg, updates create new objects in storage for each transaction. The engine is then forced to consider more objects in each subsequent "WHERE" clause and there is a linear increase in update time.

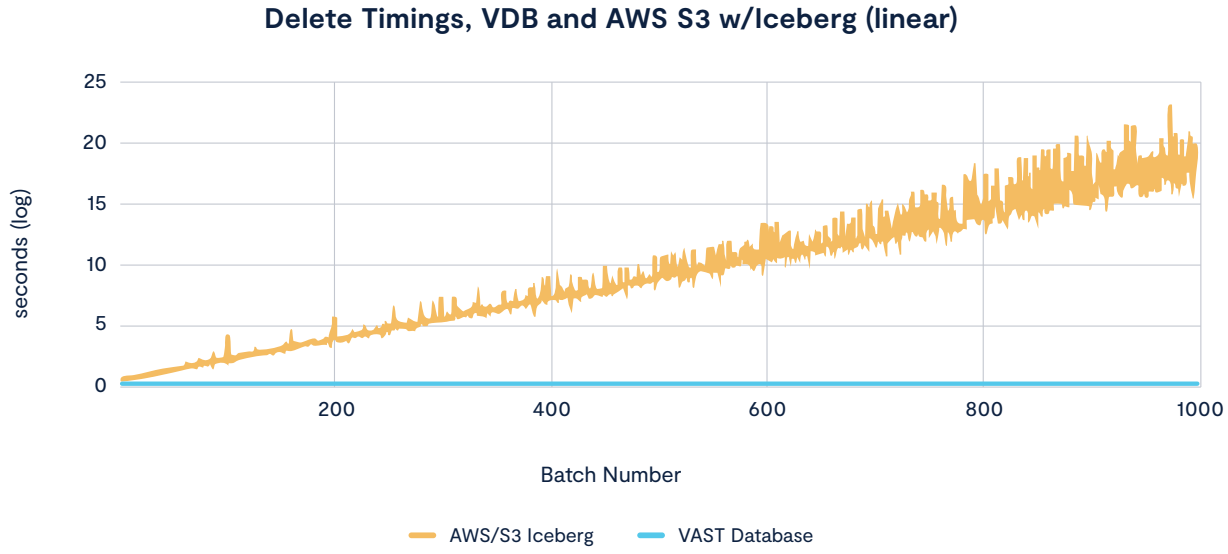
The VAST results show a very slight increase in latency as records accumulate, the rate is fractional compared to object storage and the effect diminishes over time.

Delete benchmark results

Delete benchmark results are presented using the same methodology as the update figures. This shows time (in seconds) to complete each delete operation, continuing the workload out to 1000 of these operations. Again, the y-axis is on a log scale to make both results visible on the same chart.



The same data is presented here on a linear y-axis to visualize actual performance differences.

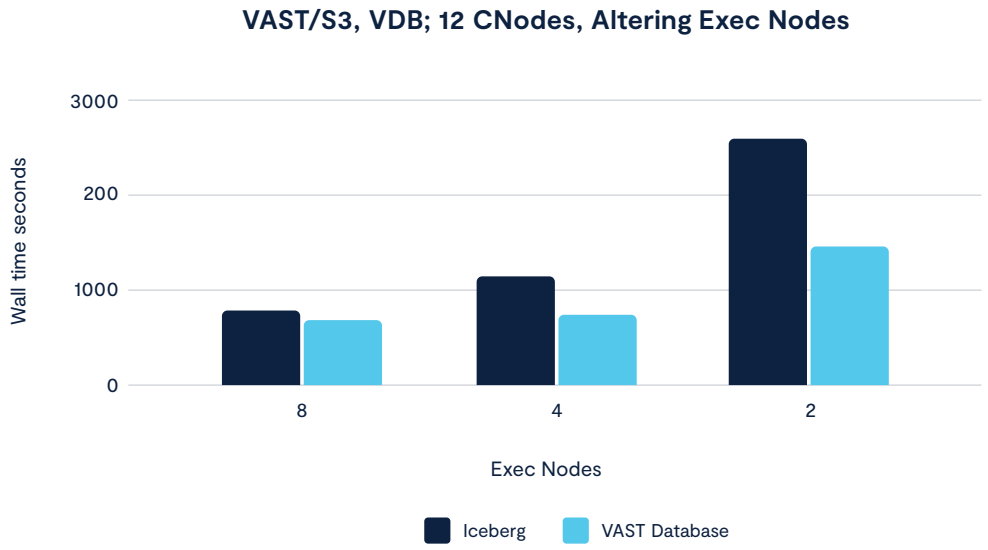


Observations

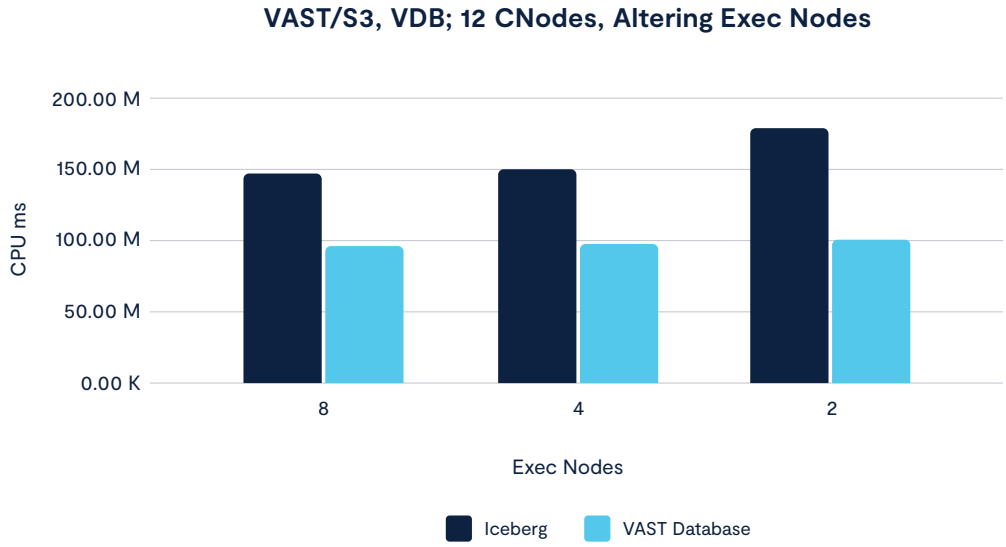
These results are very similar to the update results because they articulate the same differences in the underlying architectures. Iceberg creates tombstone objects in storage forcing their consideration in subsequent WHERE clauses, creating a degradation in performance. There are some superficial similarities in VAST, where tombstone objects are briefly placed in storage-class memory until they are reconciled by the system. However these objects are efficiently handled and reconciled in the background with negligible performance impact and without having to schedule clean-up that is tuned with the workload.

Scaling Factors

This test runs a 1TB TPC-DS workload on the VAST cluster described in "Performance Testing Environment" above, but runs the workload on 8, 4 and then 2 Trino workers.



These are the same workload tests, but viewed as total CPU seconds consumed by the jobs:



Observations

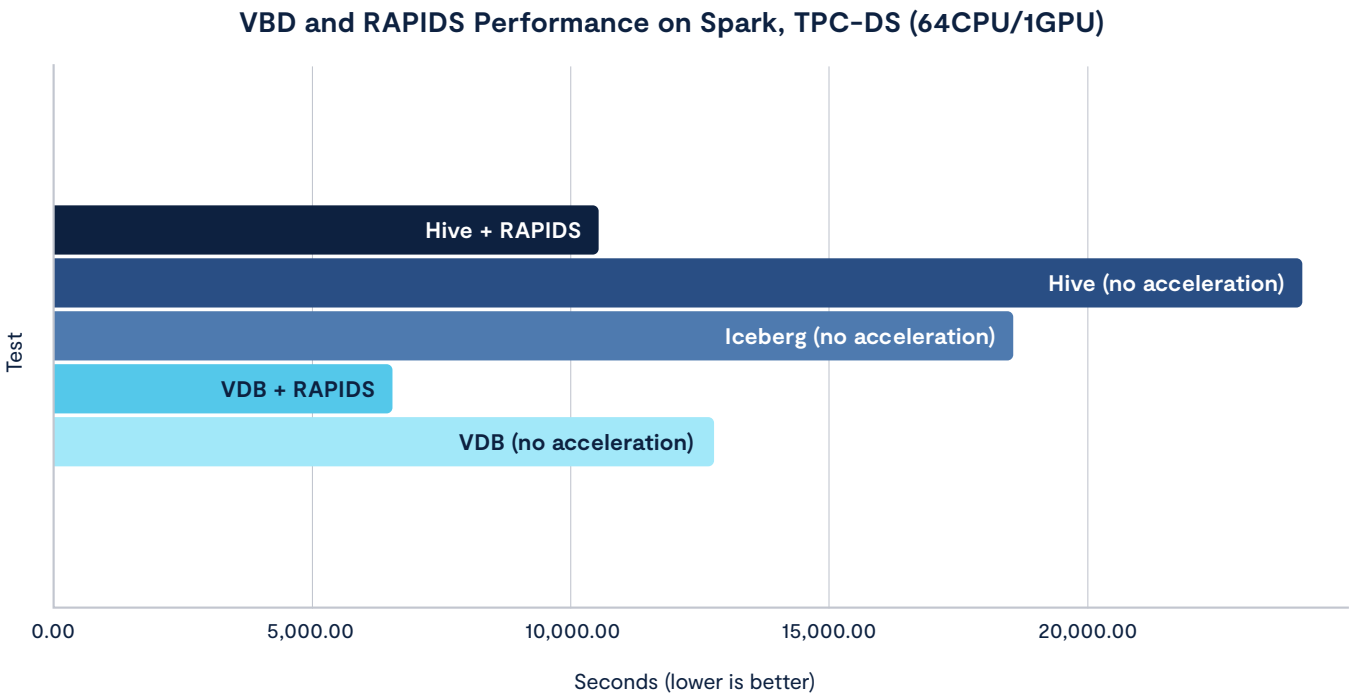
This test shows that a reduction in compute squeezes the Iceberg wall time disproportionately higher than it does to VDB. Viewed as CPU consumption, it becomes clear that it's not only taking longer but it's actually running less efficiently. While VDB enjoys a consistent compute benefit due to filtering on storage. CPU time actually increases for the same amount of work for the Iceberg tests. Owing to memory stress on the compute cluster but also a reduction in the efficiency of Iceberg's file pruning routines when using small numbers of nodes. Yet another factor to take into account when designing workloads for Iceberg.

NVIDIA RAPIDS

This chart includes several benchmarks timings for reference. These are 1TB TPC-DS benchmarks running on SparkSQL with or without RAPIDS acceleration:

- A hive/parquet baseline benchmark showing an unaccelerated workload
- An Iceberg benchmark to make sure a modern lakehouse format is represented for comparison.
- Hive with RAPIDS acceleration
- An unaccelerated VDB benchmark
- A RAPIDS-accelerated VDB benchmark

Benchmark Results



Observations

VAST DataBase shows a 40% boost in performance versus NVIDIA's implementation for hive. It does this without any code changes or relative configuration changes, proving the extensibility of RAPIDS but also the careful design of VAST's plugin. RAPIDS cannot support this workload on Iceberg tables - work is being done to fix this. However if we extrapolate the gains, VDB will still show a roughly 20-30% advantage over Iceberg in this environment.

Conclusion

Reliable benchmarks are an essential part of understanding database performance and capabilities. By testing the execution speed and resource utilization of standard benchmark frameworks along-side high-stress workloads, we are able to quantify important performance characteristics. To summarize, the VAST DataBase:

- Consistently outperforms Apache Iceberg by 30% in concurrent warehousing workloads
- Is 10-20 times faster than Iceberg at highly-selective workloads
- Exhibits 10 to 50 times the performance of Iceberg for charge data operations

Benchmarking frameworks are an important first step in understanding how data processing works at scale. While performance and CPU usage are critical metrics to measure and validate, they are not the only factors to consider. The end-to-end time-to-value of data is the ultimate measure of success for data initiatives. With the infinite data pipeline configurations available today, comparing pipeline performance can easily become a quagmire. It is therefore important to consider the lifecycle of data, from source to consumer, and account for the specific use cases when building a data architecture.

Note:

Test harnesses in support of this paper used VAST 5.1 software. The current GA release is 5.3 which includes enhancements that improve performance across all of the workloads reported here. We are looking forward to an update to this paper with VAST version 5.4 which will include benchmarks for our Kafka API implementation, fully sorted tables & projections, compression-in-flight and more.

Even with a best-in-class data warehouse with incredible performance and efficiency, the complexity of ingestion and consumption of data into and out of that data warehouse can greatly hinder innovation with data. Therefore, the effort required to build, manage, and maintain complex data pipelines must also be considered in addition to the specific technologies used. Every replication of a dataset along a pipeline will add risk, complexity and latency to that pipeline.

The VAST DataBase is a key element of the VAST Data Platform: software architected specifically to streamline the data pipeline. By providing a database that combines transactional and analytical capabilities with superior query performance, the VAST DataBase targets both traditional and cutting-edge use cases like BI and AI. VAST envisions a data ecosystem that removes the complexity of overly composable data stacks, with the VAST DataBase playing the central role for all structured data.

Interested in learning more? **Contact us today.**