White Paper

AEROSPIKE

Query Aerospike real-time data at scale using SQL

Contents

Executive overview

It's easy to express but hard to achieve: give application programmers, data scientists, and business analysts access to high volumes of real-time operational data using familiar SQL tools without the high cost or high overhead. You'd want to enable them to integrate data easily from disparate systems, including commercial and open-source offerings, relational and non-relational platforms, and data sources deployed in the cloud or on-premises. Finally, you'd want to ensure the infrastructure can scale up or down as needs change, provide fast performance for mixed workloads, and keep operational costs low.

Aerospike is opening the world of real-time operational data to SQL programmers and users of popular SQL tools practically and cost-effectively. Known for its ultra-fast performance and ability to manage hundreds of gigabytes to petabytes of transactional data on small server footprints, Aerospike is empowering firms to analyze operational data in-place using industry-standard SQL. Doing so can cut infrastructure costs, speed the training of artificial intelligence (AI) and machine learning (ML) applications, and enable firms to act quickly as business climates change.

> *"Compared to the other solutions that we were evaluating, [what] made Aerospike so attractive was [that] its total cost of ownership, performance, and scale were all superior compared to any of the competitive offerings evaluated."*
>
> *Jason Yanowitz, EVP, CTO of Signal*

This paper explores how Aerospike delivers read/write SQL capabilities over its real-time operational data platform via three products

- Aerospike SQL Powered by Starburst
- Aerospike Connect for Presto-Trino
- Aerospike Connect for Spark

You'll learn how Aerospike's solutions separate compute and storage needs, enabling each layer to be scaled independently to avoid excess costs. You'll understand how Aerospike makes it easy to analyze operational data spanning systems of engagement at the edge and systems of record at the core using SQL even though the underlying data is schemaless.

You'll explore the unique features of Aerospike's solution designed to deliver exceptional performance on much smaller server footprints than other solutions. And you'll understand how Aerospike can be deployed to support popular SQL use cases, including interactive SQL queries in custom applications, batch processing of data transformations. and business analytics with tools from Tableau, Power BI, and others.

Technology background

Providing timely, efficient, and easy SQL access to real-time data has challenged firms for years, so it is helpful to understand current approaches to SQL operations over disparate data sources, including real-time data management platforms.

Federated databases and distributed SQL engines

Relational DBMS researchers and commercial vendors delivered early implementations of federated database technology decades ago. As shown in Fig. 1, these implementations typically centered around a SQL-based database server that would accept application requests for data residing in remote systems. Plus, these implementations would include an efficient plan for accessing that data, instruct subordinate components (e.g., data source-specific software connectors) to retrieve relevant portions of the data, integrate the interim result sets from multiple data sources (if needed), and return the final results to the requesting application. In addition, some implementations maintained metadata about underlying data sources for query optimization, and some even accepted local storage of user data.

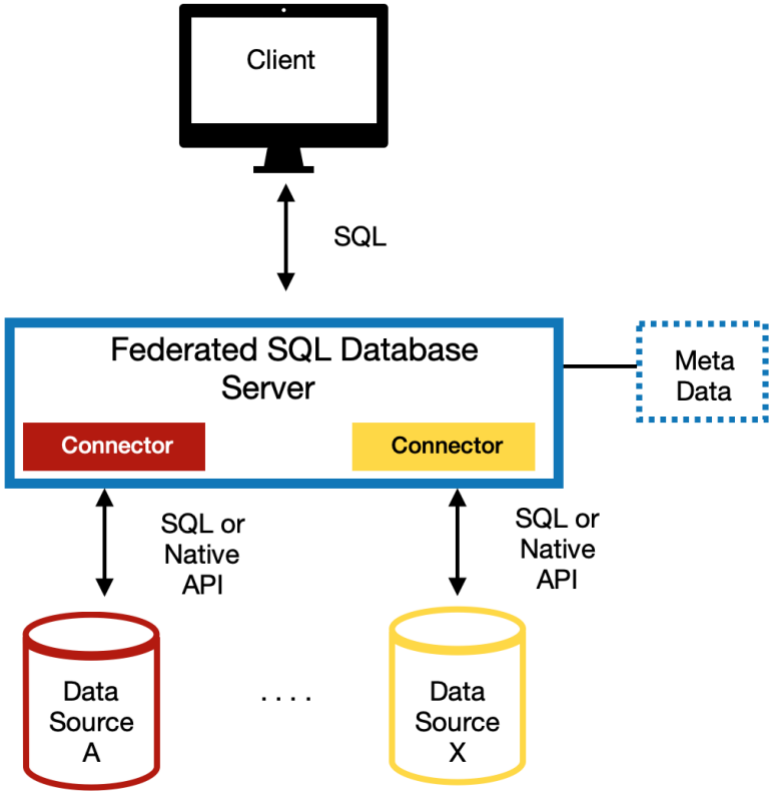

Figure 1: Federated database server architecture

The promise of using standard SQL to access physically distributed data appeals to many enterprises because critical business data is spread across multiple systems. However, problems with early federated relational database implementations soon emerged: performance was often

slow, scalability was poor, back-end data source support was limited, and platforms consumed substantial resources and were costly to operate.

As data volumes increase and non-relational data sources proliferate, demand for distributed SQL access to real-time data sources (including popular NoSQL systems) increase. Thus, the need for massively parallel SQL engines with high levels of scalability and lower costs. Starburst has emerged as one of the leading platforms for this purpose. Separation of compute and storage layers with flexible deployment options (on-premises or in the cloud) are the basic requirements of any long-term solution to this challenge.

> *"Prior to Aerospike, we were running into challenges in terms of the cost of scaling. We moved to Aerospike for its hybrid memory architecture to leverage next-generation memory and SSDs to their fullest advantage."*
>
> *Sai Devabhaktuni, Sr Director of Engineering, PayPal*

SQL with NoSQL systems

SQL was developed to provide read/write access to tabular data structures managed by relational DBMSs, and it remains the world's most popular query language. Countless business applications and commercial tools depend on it. Its adoption by the American National Standards Institute (ANSI) and International Standards Organization (ISO) has contributed to application portability and longevity.

While relational databases have enjoyed a golden age over the past three decades, the steady increase in data volumes, new data types, new data models, and applications have made non-relational systems essential and increasingly common.

However, enterprises often want SQL access, particularly for discovery, real-time analytics, and reporting. Today, many NoSQL platforms support SQL through separate software components that present a relational or tabular view of data modeled in file systems, key-value stores, document stores, and so on. Typically, non-relational objects in the underlying data store are mapped to a tabular schema so SQL-based applications can manipulate data using familiar statements (e.g., SELECT). This mapping may be manual (requiring explicit instruction from a programmer or an administrator) or automated (inferred by a software component). Next, applications connect to a distributed SQL engine and issue SQL. Behind the scenes, the SQL engine optimizes and decomposes the query into subtasks to be handled by one or more underlying data source-specific connectors, which translate the operations into the native API of the target back-end data store. After completing its work, the data store returns the results to the connector, which feeds the data to the SQL engine to be integrated with interim results from other connectors, if needed. The final result is returned to the SQL client application.

As you might imagine, performance is influenced by the capabilities of the SQL query engine and the back-end store. If the SQL engine selects a poor data access strategy or the back-end store is

slow to process certain operations, the SQL client application can experience high data access latencies.

Various use cases for SQL access to NoSQL data carry performance and data consistency implications. For example, business analysts employing popular SQL-based business intelligence dashboards may be less tolerant of slow responses than IT specialists running batch analytical applications or initiating lengthy extract/transform/load (ETL) jobs. In addition, providing SQL access to NoSQL systems broadens the breadth of corporate and public data that applications can use to drive and refine critical business processes and decision-making.

Essential features

Enabling business analysts, data scientists, and application programmers to use SQL to access disparate, physically distributed data is challenging. It requires a well-designed software infrastructure to avoid performance problems, data availability issues, data inconsistencies, and high operational costs. Functional considerations are also essential, including the breadth of features and operations supported and the simplicity of deployment in the cloud, on-premises, or hybrid environments.

Practical questions for data architects and IT leaders include:

- **Use cases:** What are your SQL use cases, and how would you prioritize each? Consider the varying needs required by interactive ad hoc, complex, and batch queries and exploratory data analytics. How does the platform address these needs?
- **Scaling:** How well can the solution scale up and down to suit changing business requirements? To what extent can compute and storage layers be scaled separately, if needed?
- **Data sources:** What data sources are supported? How much setup, monitoring, and tuning are required at the SQL engine level and at each back end to achieve performance and availability targets?
- **SQL requirements:** What SQL capabilities are supported? What popular SQL tools and software libraries are supported? What SQL data types are supported? What specialized NoSQL data types are supported, and how are these mapped to SQL types?
- **Distributed processing:** What features are provided to ensure strong runtime performance? Consider both SQL query engine features and data source-specific features. For example, to what extent does each support parallelism and efficient use of machine resources?
- **SQL optimization:** Does the SQL engine employ cost-based query optimization and create intelligent data access plans that push down as much query processing as possible to the data source?
- **Data source optimization:** Can the data source feed statistical information to the query engine? Does the data source use native indexes, connection pooling, and other features to speed performance?

- **Infrastructure:** How does the infrastructure address data availability, data consistency, and security concerns?
- **Diagnostics:** What diagnostic capabilities are offered?

Let's explore how Aerospike, a leading NoSQL platform for managing real-time transactional data, supports SQL access.

Accessing Aerospike data with SQL

If you're new to Aerospike, it's a distributed multi-model NoSQL platform that provides high-speed predictable read/write access to billions of records in databases holding up to petabytes of real-time operational data. Firms in finance, retail, telecommunications, technology, and other industries use Aerospike to support systems of engagement and systems of record 24x7, often saving $1 - $10 million per application compared with other approaches. How? Aerospike delivers exceptional availability and runtime performance with dramatically smaller server footprints through deep exploitation of modern hardware, including multi-core processors with non-uniform memory access (NUMA), non-volatile memory extended (NVMe) Flash drives, persistent memory (PMem), network application device queues (ADQ), and more. Other features that distinguish Aerospike include its ability to automatically distribute data evenly across shared-nothing clusters, dynamically rebalance workloads, intelligently route queries to appropriate nodes for fast performance, and accommodate software upgrades and most cluster changes without downtime.

Aerospike can be deployed on-premises, in the cloud, or in hybrid environments. Firms worldwide use it in production as a system of record at the core, a system of engagement at the edge, and a global transactional system spanning multiple sites. Strong, immediate data consistency, self-managing and self-healing features, and flexible data replication technologies have helped drive Aerospike's increased usage worldwide. Figure 2 illustrates several ways Aerospike customers use its real-time data platform. For more details on Aerospike's internal design and key features, see Introducing Aerospike's architecture.

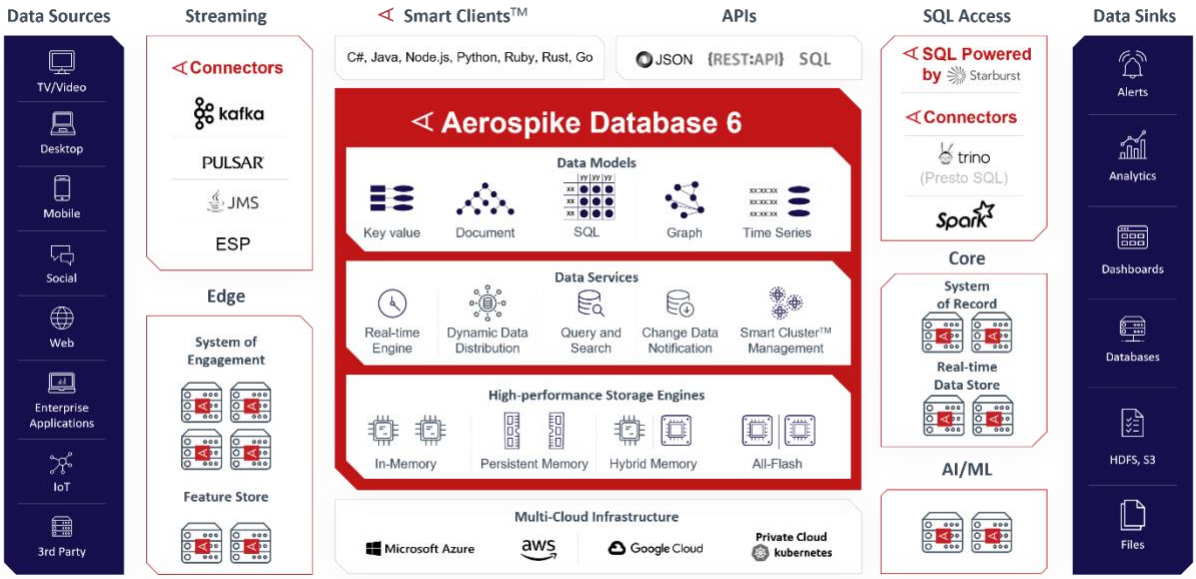

Figure 2: Aerospike Real-time Data Platform architecture

Accessing Aerospike with SQL enables customers to eliminate the expense and effort of moving Aerospike data to a separate SQL system for dashboarding, exploratory analytics, and other common SQL use cases. Furthermore, firms that rely on SQL-based systems for real-time data can now employ Aerospike as a back-end system, often cutting operational costs substantially due to Aerospike's comparatively small server footprint.

Let's explore how Aerospike delivers its SQL support.

Technology overview

Through connectors to Trino (PrestoSQL) and Apache Spark, Aerospike offers firms the ability to access one or multiple Aerospike clusters with ANSI SQL and integrate data from Aerospike with non-Aerospike systems. Furthermore, Aerospike has optimized its Trino and Spark integration for high performance, scalability, and functionality. As you'll see, Aerospike combines its massively parallel processing technology with the native parallel processing of these open source technologies to speed performance and support high levels of scalability that other back-end systems don't provide. Furthermore, Aerospike's connectors can push down filtering operations to Aerospike; this minimizes costly data transfers.

Before we delve further into the capabilities of these connectors, it's worth reviewing some usage scenarios for Trino and Spark. Trino is well-suited to supporting interactive analysis through ad hoc queries and popular business intelligence (BI) and reporting tools. Low data access latencies are essential to speeding analytics and delivering fast "time-to-insight." Furthermore, many popular BI tools -- such as Tableau, MicroStrategy, Looker, Qlik, and others -- include JDBC drivers that can be used with Trino, so deployment is straightforward. On the

other hand, batch SQL queries, ETL jobs, and exploratory data analytics associated with AI / ML applications often span massive data volumes. As a result, they require specialized libraries, including many that Trino doesn't support. Spark addresses such needs nicely and is often the preferred choice of data scientists and data engineers.

Presto, Trino, and Starburst SQL Engines

Presto was developed at Facebook to enable its analysts to issue interactive SQL queries against a multi-petabyte data warehouse stored in Apache Hadoop. Facebook released Presto as open-source in 2013; later, the Presto Software Foundation forked the code base into PrestoSQL, which was ultimately rebranded as Trino.

Key aspects of Trino include a wide range of available data source connectors, high levels of scalability, separation of compute and storage layers, support for ANSI SQL, cost-based optimization and other performance features, and deployment on-premises or in the cloud. For the remainder of this paper, we'll use Trino to denote both Trino and Presto solutions.

Aerospike has developed two products that leverage Trino: Aerospike SQL Powered by Starburst and Aerospike Connect for Presto-Trino.

Aerospike SQL Powered by Starburst

Aerospike SQL Powered by Starburst – also referred to as Aerospike SQL - is an integrated solution of Aerospike Connect for Presto-Trino and the Starburst Enterprise platform. Starburst Enterprise is a commercial implementation of Trino that provides a fully tested, supported, and hardened implementation of Trino. Aerospike Connect for Presto-Trino translates SQL queries from Trino/Starburst enterprise SQL engines. A complete discussion of how Aerospike Connect for Presto-Trino is included in the next section.

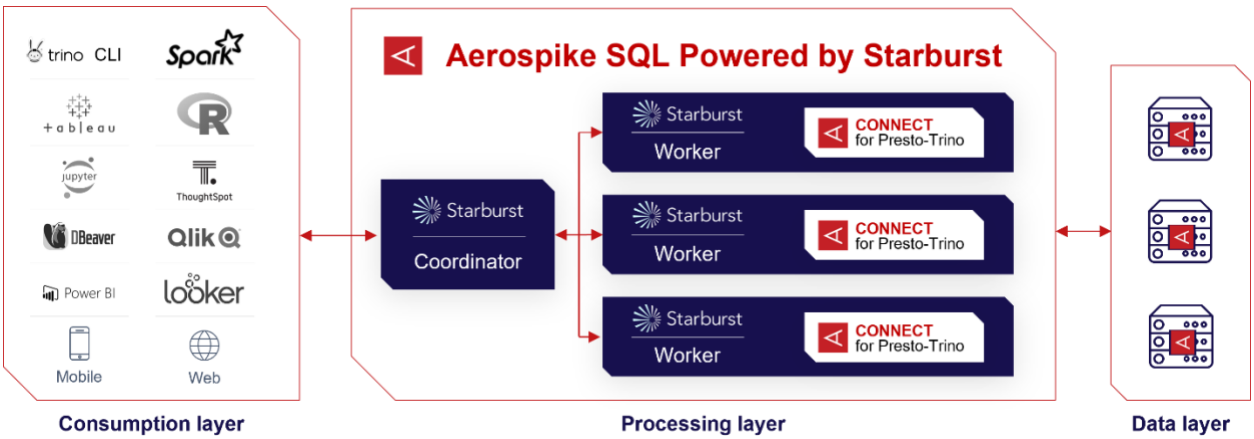

Figure 3: Aerospike SQL Powered by Starburst architecture

Aerospike SQL goes beyond the Aerospike Connect product in that it includes both Starburst Enterprise and Aerospike Connect for Presto-Trino as part of the installation, operation, and management of the environment. This provides the proven Starburst Enterprise SQL engine and all the technologies needed to process SQL queries in your Aerospike Database clusters, effectively adding ANSI SQL support to the Aerospike Real-time Data Platform.

Aerospike Connect for Presto-Trino

Aerospike Connect for Presto-Trino (sometimes called the Presto or Trino connector) enables firms to use ANSI SQL to read (SELECT) from or write (INSERT) data into Aerospike via popular BI tools and interactive queries, as shown in Fig. 4. Such workloads tend to be read-centric and demand high-speed data access. Aerospike's ultra-fast performance over high volumes of real-time data ensures the back-end system won't hinder SQL programmers and users of BI tools.

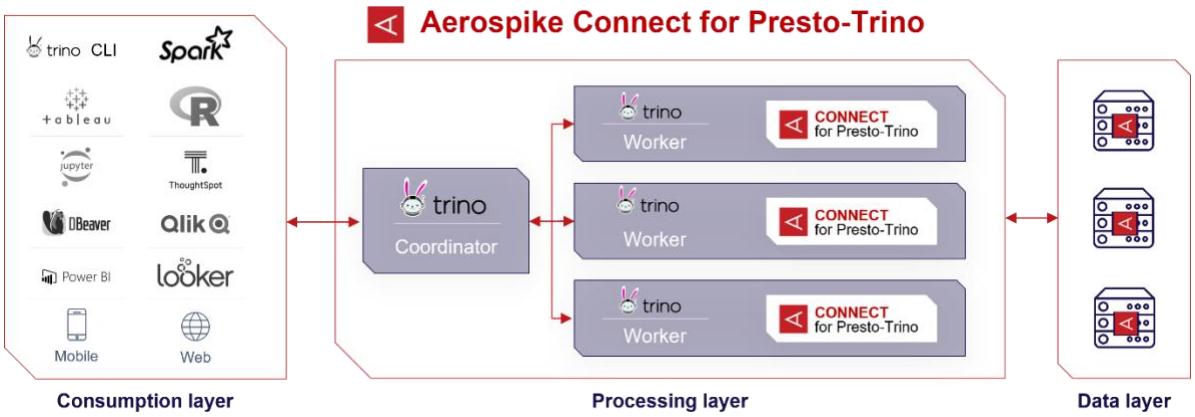

Figure 4: Using Trino with Aerospike

If you compare Figures 3 and 4, you will note that they are very similar. However, it is important to note that Aerospike SQL installs both Aerospike Connect and Starburst Enterprise, whereas Aerospike Connect is typically installed in existing Trino environments. In both cases, Aerospike Connect for Presto-Trino is installed in the Trino cluster worker nodes and interacts with the Trino coordinator.

	Aerospike Connect for Presto-Trino	Aerospike SQL Powered by Starburst
Included software	• Aerospike Connect for Presto-Trino	• Aerospike Connect for Presto-Trino • Starburst Enterprise Platform (SEP) • Starburst Admin
Trino cluster management	No - Trino installed and managed separately	Yes
Federated SQL queries across multiple databases	Yes	No – limited to Aerospike clusters only
Vendor support	Multiple	Single

Figure 5: Comparing Aerospike SQL and Aerospike Connect for Presto-Trino

How Trino Works with Aerospike

Let's explore the architecture of Aerospike's solution by walking through a simple query scenario involving a single Aerospike cluster. Referring to Fig. 4, imagine that a client application (at left) connects to the Trino/Starburst cluster and issues a SQL SELECT statement for data managed by Aerospike. The Trino/Starburst coordinator parses the query and develops a query plan using the Trino cost-based optimization (CBO) if configured to do so.

Unlike some data source connectors, Aerospike Connect provides Trino with statistics (row counts for tables) to help Trino's query engine intelligently determine an efficient query plan. The Trino coordinator divides its query plan among Trino workers, each of which interfaces with Aerospike. These Aerospike connectors initiate data partition scans in parallel and push down query predicates to the Aerospike server. Figure 6 below graphically details the process flow for Aerospike SQL and Aerospile Connect for Presto-Trino.

> *"Aerospike has done the unthinkable: they cut our server footprint by a factor of 6 while boosting our performance 300%. The resultant total cost of ownership is saving us over $1 million a year."*
>
> *Guy Almog, Head of IT Engineering, Playtika*

As a result, data retrieval is accelerated, and unnecessary data transfers are minimized. These connectors also load data returned from Aerospike's 4096 partitions into as many as ~2 billion Trino splits for the Trino workers to process. Finally, the Trino/Starburst coordinator fetches results from its workers and returns them to the client.

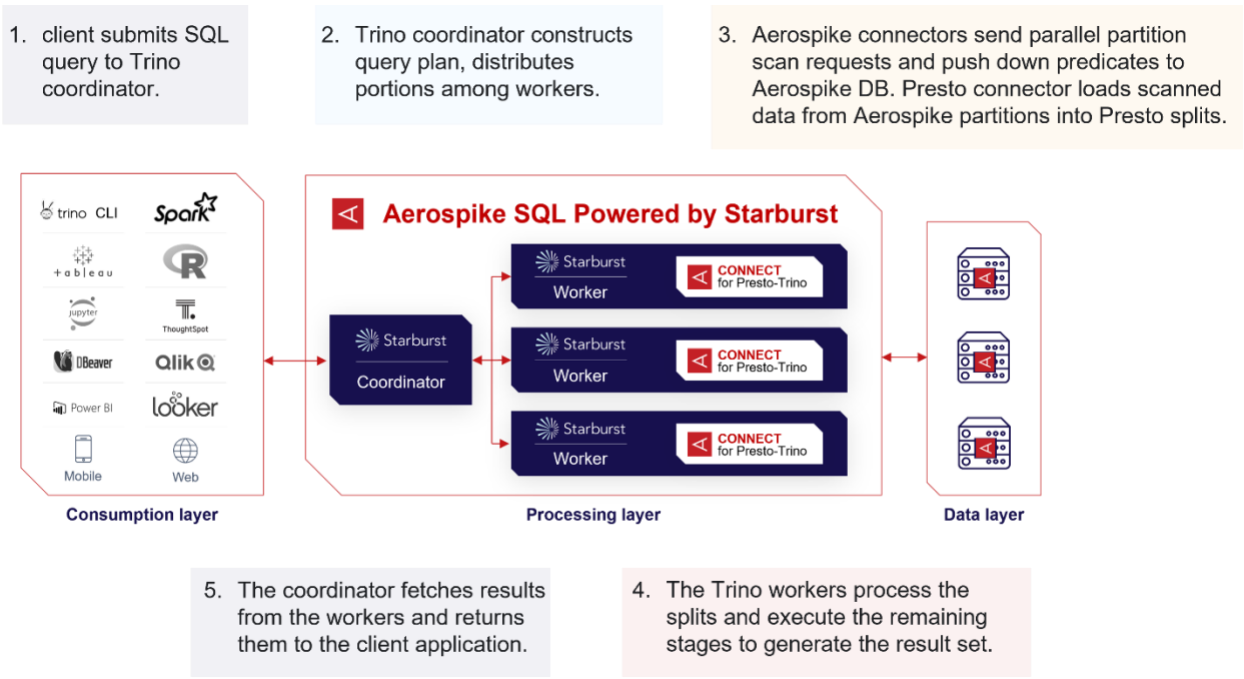

Figure 6: Process flow for Aerospike SQL/Aerospike Connect in the Trino/Starburst environment

Aerospike's approach leverages massive parallelism at the back end (Aerospike Database) and the Trino SQL engine layers. Users can scale each up or down independently to suit their needs. For example, adding or removing Trino nodes will not cause Aerospike to rebalance its workload or redistribute its data. The Aerospike server accommodates changes in its cluster without operator intervention. Aerospike also manages the addition or removal of nodes without service interruption and automatically rebalances data whenever a change in cluster status is detected. Upgrades to one or more Aerospike nodes don't require the cluster to be paused, nor any data rendered unavailable. For production users, SQL or otherwise, such capabilities afford 99.999% uptime and predictable real-time access to data.

Aerospike's Trino connector can be deployed to access multiple Aerospike clusters in different geographies or cloud regions if desired. In effect, applications can integrate data from multiple Aerospike servers, joining and filtering the results as desired. This can be particularly useful for federating queries across numerous edge-based systems.

A SQL application can read or write data to a globally distributed system of record managed by Aerospike (as shown below in Fig. 7). This is because a single Aerospike cluster can be configured to span multiple data centers or cloud regions with strong, immediate data consistency. The Aerospike Trino connector can be deployed on-premises or in the cloud, in either Docker or Kubernetes environments.

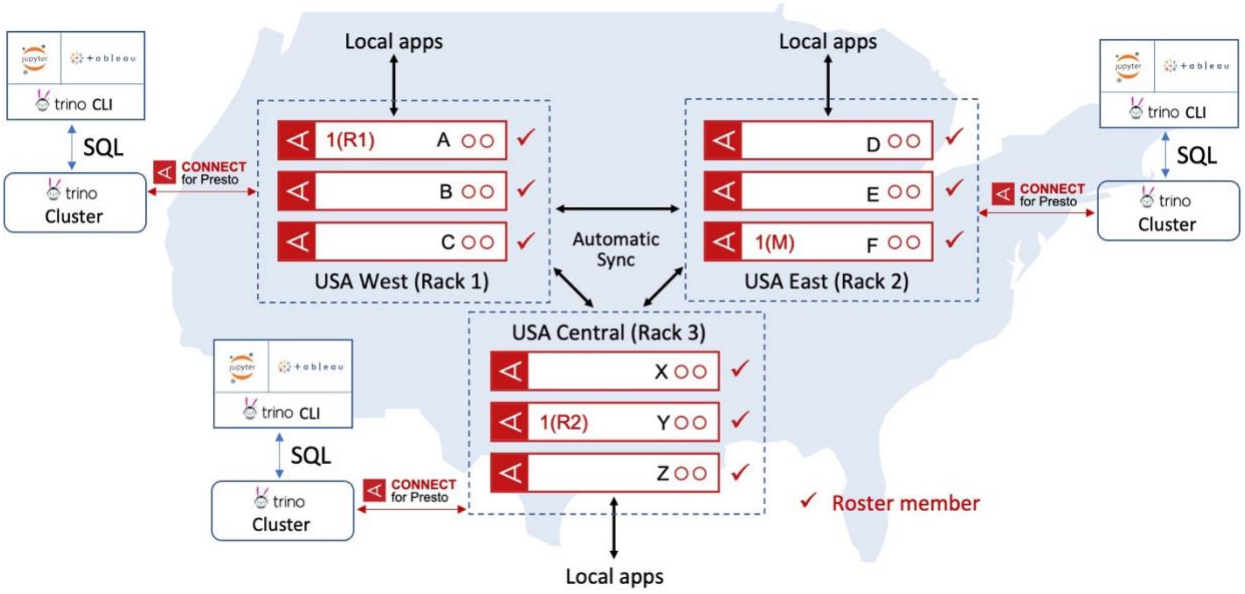

Figure 7: SQL access to a single Aerospike cluster that spans multiple sites (Conceptual view)

To simplify SQL access, Aerospike's Trino connector uses heuristics to infer SQL schemas from an Aerospike database, relieving SQL programmers from the burden of doing so. Optionally, programmers can specify schemas for Aerospike data through JSON files. In either case, SQL users perceive Aerospike data as residing in tables just as they would with any relational DBMS. Furthermore, Aerospike also supports flexible schemas; this enables users to run SQL queries against a set with a multi-type record.

Aerospike's connector supports popular SQL data types and statements, including joins, range queries, and popular aggregate functions. The connector even supports popular Aerospike data types (maps, lists, and GeoJSON), mapping these to JSON for SQL programmers. Furthermore, Aerospike can ingest serialized JSON data using Aerospike's Document API; Trino users can then run SQL queries against this data using Trino's JSON functions and operators. Aerospike's Trino connector documentation provides further details on the connector's capabilities and usage examples.

For security, Aerospike recommends using transport layer security (TLS) for connection between Trino and Aerospike clusters. For authentication, Aerospike supports public key infrastructure (PKI), which enables central management of users, roles, and credentials. Alternatively, Aerospike environments configured for TLS can employ the Lightweight Directory Access Protocol (LDAP) for authentication. Finally, the Aerospike Connect for Presto-Trino supports audit trails for compliance and security purposes. See the product documentation for further details on security with the Trino connector.

Aerospike Connect for Spark and Spark SQL

Apache Spark has emerged as a leading open-source platform for AI/ML, streaming, and other applications. As shown in Fig. 8, Spark provides unified analytics for large-scale data processing. In addition, it features the Spark SQL query engine that exploits high-scale parallel processing to access disparate and distributed data sources, including popular NoSQL sources.

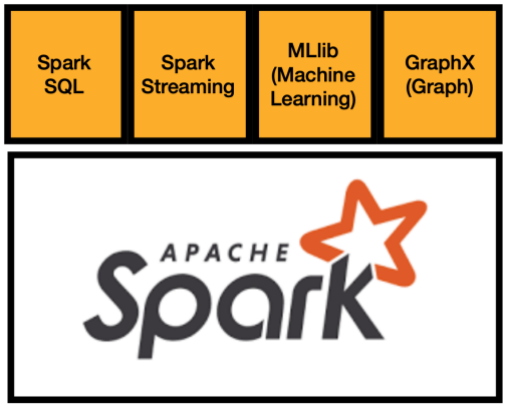

Figure 8: Major components of Apache Spark

Aerospike Connect for Spark enables programmers to use streaming and SQL APIs to read and write real-time data managed by Aerospike clusters. Doing so enables AI/ML and analytical use cases to leverage Aerospike in a Spark Streaming pipeline, integrate Aerospike data with other data sources supported by Spark, and exploit historical and streaming data in real-time to enhance business processes.

In many ways, Aerospike is an ideal storage choice for Spark due to its exceptional exploitation of native hardware and relatively small server footprints. This is particularly significant given the heavy memory and storage demands of AI/ML training workloads, many of which use Spark. Indeed, as Fig. 9 indicates, one study estimated that AI training workloads could require 6x the DRAM and 2x the SSDs of a standard cloud server, driving up costs considerably. Aerospike helps firms minimize such costs. As a recent petabyte-scale benchmark demonstrated, Aerospike delivers exceptional read/write performance on relatively small cloud clusters, cutting server footprint up to 80% and overall costs by $1 - $10 million per application.

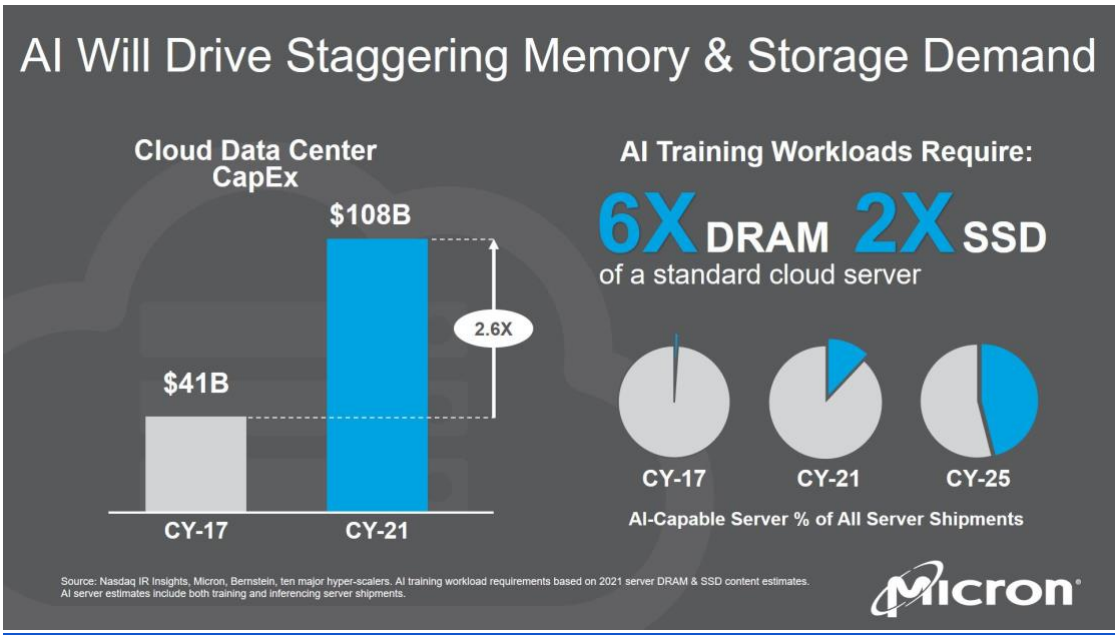

Figure 9: AI applications consume substantial memory and storage ***(Source: Nasdaq IR insights)***

Indeed, many firms have discovered that Aerospike is a powerful platform for AI/ML applications. Its architecture efficiently exploits the latest memory and storage technologies, speeding heavy data preparation work during the AI/ML training phase and reducing overall costs compared with traditional systems that don't exploit SSDs to the extent that Aerospike does. This is particularly important since training is a continual process rather than a one-time event due to frequently changing trends and patterns reflected in the data. Furthermore, data scientists often must experiment with hundreds or thousands of models before finalizing one for production use; speeding such efforts can result in significant business impact. For example, consider such time-critical use cases as fraud detection, risk analysis, etc. In addition, rapid rollouts of new AI-driven capabilities can result in substantial cost savings and improvements in customer service. Finally, Aerospike's ability to deliver predictable, ultra-fast performance for mixed workloads on small server footprints makes it an ideal platform for storing and managing real-time data needed by AI/ML applications.

So how can firms pair Aerospike with Spark? Fig. 10 illustrates a simple usage scenario, with Spark clients (at the bottom) connecting to Spark and accessing data from Aerospike that has been loaded into the Spark cluster through the Aerospike connector. Using DataFrames and DataSets (with Spark SQL) or Structured Spark Streaming APIs, programmers can interact with one or more Aerospike clusters in a familiar manner. For example, Python users can leverage PySpark to load Aerospike data into a DataFrame and then process the data with various open-source Python libraries, including Pandas and NumPy; AI/ML frameworks, such as Scikit-learn, PyTorch, TensorFlow, SparkML; and visualization libraries, such as Matplotlib.

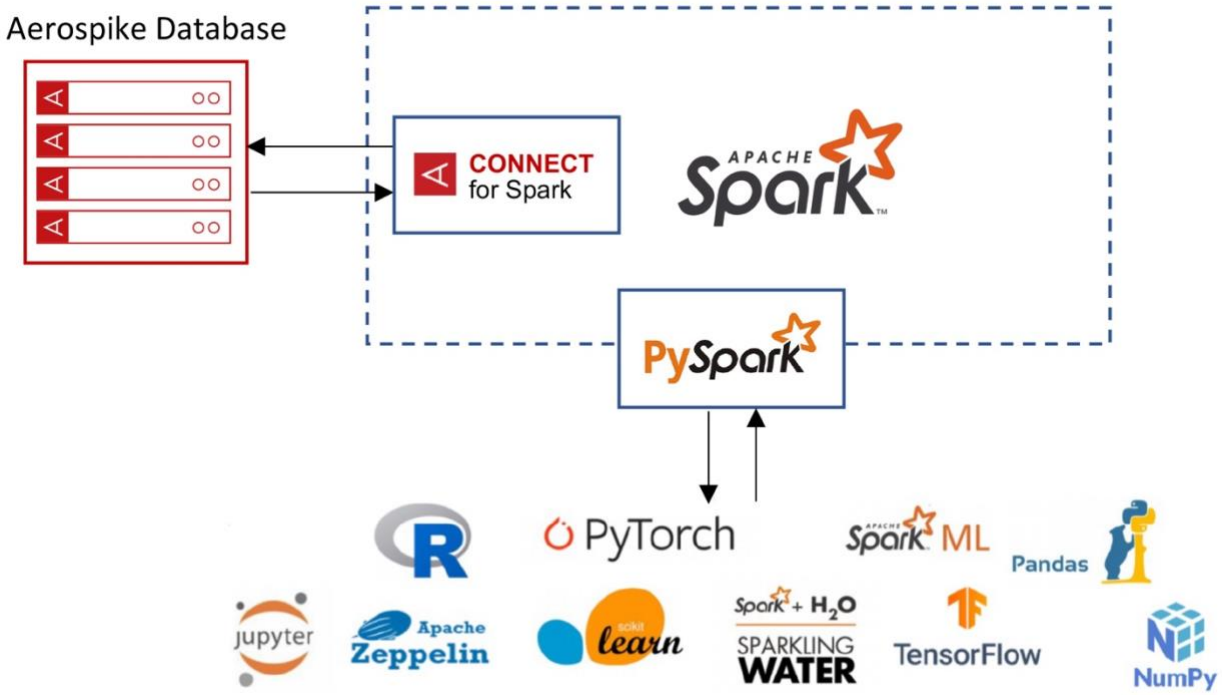

Figure 10: Sample use of Aerospike's Spark connector

Fig. 11 illustrates another usage scenario for Aerospike's Spark connector, in which streaming data (in this case, from Apache Kafka) can be processed and enriched through Spark's Streaming SQL API and ultimately written back to an Aerospike cluster. Aerospike's cross-data center replication (XDR) technology can feed data changes initiated by Aerospike client applications to Kafka via Aerospike's Kafka connector. It enables real-time business data collected by Aerospike to be combined with streaming data from other sources in a closed-loop real-time process.

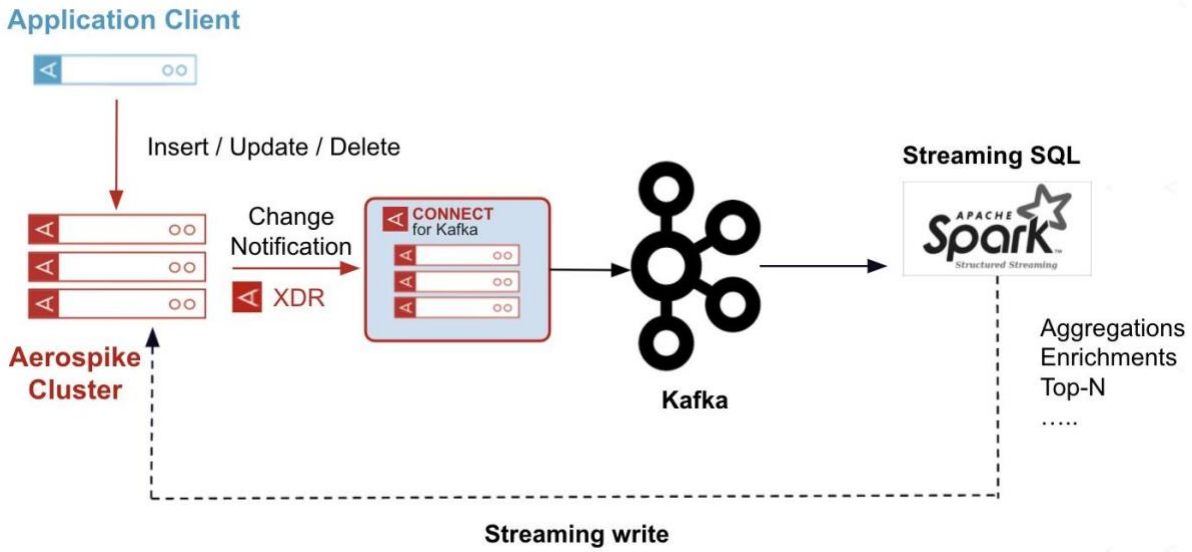

Figure 11: Combining Aerospike, Kafka, and Spark for real-time data processing

Like the Aerospike Trino connector, Aerospike Connect for Spark separates compute and storage layers for cost-efficient scalability. Furthermore, Aerospike's Spark solution employs predicate filtering, a scan-by-partition strategy, and mapping Aerospike partitions to Spark partitions for massive parallelization. In particular, users can employ up to 32K Spark partitions to read data from an Aerospike namespace; each namespace can store up to 32 billion records across 4096 partitions on a minimal server footprint. As a result, applications that were once impractical or too costly to implement are now feasible through the combination of Aerospike and Spark. Indeed, one global firm found that deploying Aerospike and Spark led to a 10x reduction in time-to-insight and helped the firm closed-loop pursue new business opportunities.

Aerospike Connect for Spark supports SQL INSERTs and SELECTs, schema inference, and pushdown of query predicates, including Aerospike Expressions, to reduce the scope of data scanned. Set indexes are also supported, enabling a Spark DataFrame to be mapped to an Aerospike set to promote an efficient set scan rather than requiring a full namespace scan. This is particularly useful for querying a small set of data in a large namespace, as recent performance benchmark tests indicate. Furthermore, Aerospike enables administrators to set quotas to limit Spark users so that no single Spark job consumes excess Aerospike resources. This prevents long-running or I/O intensive Spark jobs from unduly impacting other high-priority work.

Aerospike's Spark and Trino connectors can be deployed within the same data platform infrastructure to expand the breadth of analytics available. For example, the Aerospike Spark connector may be used to feed data from S3 or other cloud data sources to Aerospike in real-time. Such data can then be analyzed with BI tools or interactive SQL queries using Aerospike's Trino connector. Aerospike's low data access latencies and small server footprints frequently make such deployments attractive compared to alternate solutions.

Aerospike Connect for Spark is deployed by adding the connector's package to your application environment and later submitting the application to the Spark cluster for execution in a highly distributed manner. Users can deploy their Spark clusters on-premises or in the cloud, using Docker or Kubernetes as desired. As with Aerospike Connect for Presto-Trino, Aerospike Connect for Spark supports TLS, KPI, and LDAP for security. In addition, Aerospike's documentation provides further details on supported SQL statements and data types, performance features, configuration options, and tutorials.

Summary

Aerospike enables enterprises around the globe to query real-time operational data at scale using industry-standard SQL in a practical and cost-efficient manner. Through Aerospike SQL Aerospike Connect for Presto-Trino and Aerospike Connect for Spark, Aerospike empowers firms to read and write data to one or more Aerospike clusters using ANSI SQL. Firms can even combine this Aerospike data with data managed by popular commercial and open-source relational DBMSs, NoSQL systems, and file systems.

Known for its ultra-fast performance and ability to manage hundreds of gigabytes to petabytes of data on small server footprints, Aerospike is helping firms bring the world of real-time operational data to SQL programmers and users of popular SQL dashboards. This speeds time to insight, provides timely training (and retraining) of AI/ML applications, and enables firms to adapt quickly to a changing business climate. Furthermore, through exceptional resource efficiency and separation of compute and storage layers, Aerospike helps firms ensure that they can scale their infrastructures as needed without incurring high costs. Indeed, Aerospike's server footprints are often a tiny fraction of what alternate solutions require; its production users usually save $1 - $10 million per application as a result.

To learn more about Aerospike and its SQL support, contact sales@aerospike.com or visit the Aerospike website at www.aerospike.com.

About Aerospike

The Aerospike Real-time Data Platform enables organizations to act instantly across billions of transactions while reducing server footprint up to 80%. The Aerospike multi-cloud platform powers real-time applications with predictable sub-millisecond performance up to petabyte-scale with five-nines uptime with globally distributed, strongly consistent data. Applications built on the Aerospike Real-time Data Platform fight fraud, provide recommendations that dramatically increase shopping cart size, enable global digital payments, and deliver hyper-personalized user experiences to tens of millions of customers. Customers such as Airtel, Experian, Nielsen, PayPal, Snap, Wayfair and Yahoo rely on Aerospike as their data foundation for the future.

For more information, please visit www.aerospike.com.