Solution Brief

# Feed Hungry ML Systems More Data, Faster & Efficiently

## Improving AI/ML Applications and Outcomes with a Modern Data Platform

### The Aerospike Difference for AI/ML

- Reduce time to execute Spark Jobs by leveraging Massive Parallelism in Spark and Aerospike
- Enables in-situ data exploration using SparkSQL and Presto eliminating compliance headaches by removing the need to copy data into multiple systems
- Creates a low latency inference or online training pipeline using Aerospike Connect for Spark and Aerospike 5

AI/ML systems have insatiable appetites for data. Machine learning models run better with more data, and the more iterations and the more training, tuning and validation you can do, the better your results. The challenges lie in data preparation (which is painful) and model creation and tuning as models are constantly evolving. Plus, you need an online system with streaming data and the need to make an inference in milliseconds. The problem is wanting to pull disparate signal data from sources from different countries and data centers in real-time. Sometimes the hardest part of AI/ML isn't the AI/ML, "it's the plumbing."

If you were to sum up the major challenges from the data science community, i.e. those responsible for creating and executing AI/ML models it would be:

- Need to build the most sophisticated model in the shortest time
- Training can go on for days/months
- Data prep is very I/O intensive due to complex transformations required
- As training goes on, people get unhappy

## The Aerospike Data Platform for AI/ML

The Aerospike data platform is designed to ingest large amounts of data in real-time for parallel processing while connecting to compute platforms as well as notebooks and ML packages.

Aerospike at its core is a multi-model NoSQL database that can load data into a Spark DataFrame in a massively parallel fashion. Once the data is in the Spark DataFrame, you can bring to bear any of the popular AI/ML libraries and Frameworks of your choice to create highly performant AI/ML models based on your use case. Furthermore, you can explore data in Aerospike using Jupyter or Apache Zeppelin notebooks. (As you may know, Spark has built in support for these.)

The beauty about this architecture is that it separates compute (Spark) and storage (Aerospike) so that you can right-size them independently to achieve lower TCO. And last but not least, you can deploy it anywhere with Kubernetes and Docker.

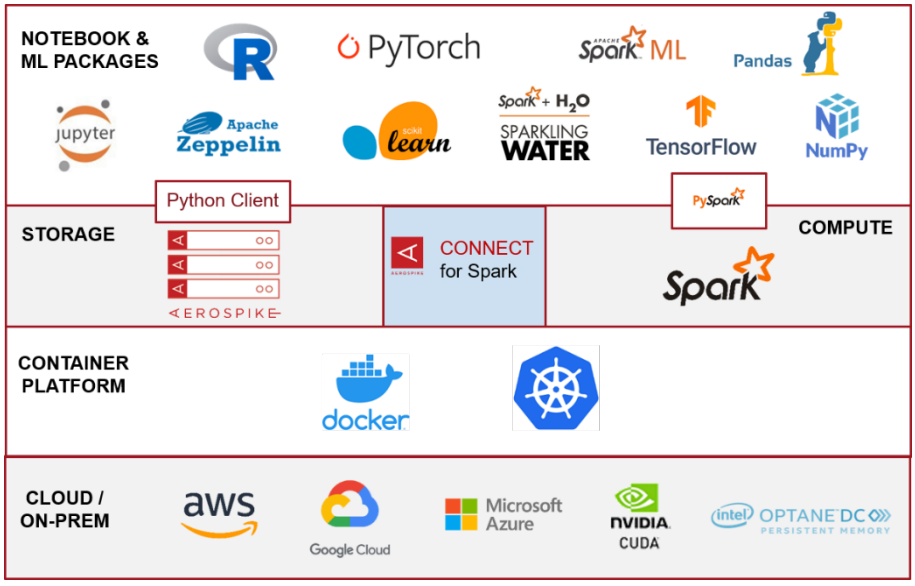



Figure 1: Aerospike Connect for AI/ML

## Aerospike Storage Architecture Provides Choice

Any storage architecture for AI/ML that you choose must satisfy low latency and high throughput reads and writes, while not blowing your budget at the same time. Aerospike has a hybrid-memory architecture that is optimized for in-memory, SSD, hybrid and all-flash. This is what enables Aerospike to speed up an AI/ML pipeline from a storage standpoint.

Aerospike works very closely with Hardware vendors such as HPE and Intel to make sure that we leverage their bleeding edge storage innovations.

Aerospike gives you the choice to select the storage architecture that suits your design:

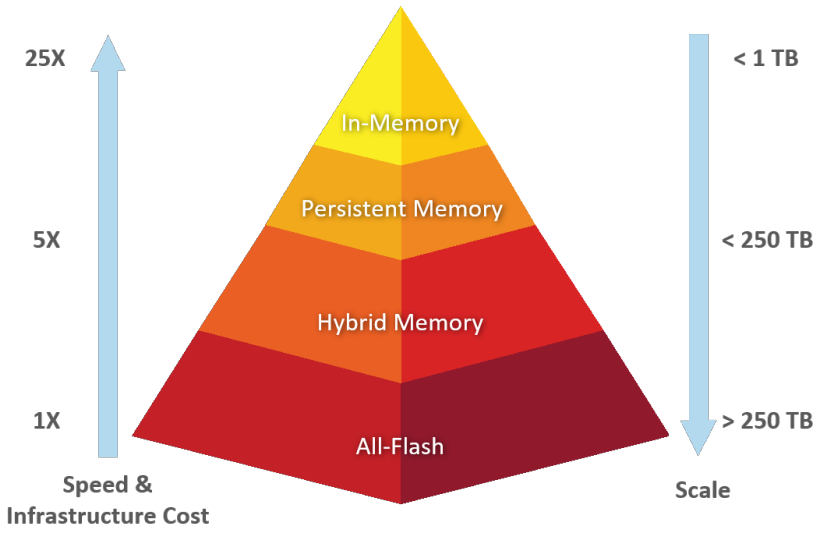



**Figure 2**: Aerospike Storage Architecture provides choice

- **In-Memory** – Storing both indexes and data in memory. While extremely fast, it tends to prove cost prohibitive at scale.
- **Persistent Memory (PMem)** - By storing both indexes and data in PMem, the indexes can be retained in persistent memory when the system is powered down, so Aerospike can typically be restarted in a matter of seconds to enable non-disruptive maintenance. Also, downtime is reduced, software updates and security patches can be performed more frequently, and redundancy and replication requirements can be met more easily, resulting in significantly lower TCO.
- **Hybrid memory** - Aerospike can store database indices in DRAM, and the data on SSD's, irrespective of the amount of data on an individual node. Thus, the data density per node for Aerospike is significantly higher than with other databases, and Aerospike preserves performance, while minimizing the cluster size.
- **All-Flash** – Similarly, you could choose All-flash for indexes and data which works well for a high number of objects (typically into the billions).

## Speeding up the Training Pipeline with Aerospike

Aerospike can help accelerate a training pipeline significantly. In short, Aerospike can serve as a System of Record and rapidly load Spark DataFrames with massively parallelism. Per Figure 3 below, Aerospike can serve as a System of Record, with the ability to store 100's terabytes up to petabytes of data.

1. Load the training data - This can be existing transactional and/or production data from your core system of record. Use Spark and the Aerospike Connect for Spark connector to load the data into Spark DataFrames in a massively parallel manner. (See subsequent section in this brief for more on parallelization.)

2. Data Prep and Exploration – Data preparation involves running Spark jobs for cleansing, enriching, transformations and pre-processing such as one-hot encoding, generalization, normalization, etc. Additionaly, third party data, can be also be added to production data. It is a very I/O intensive process.

   Data exploration is an important aspect of AI/ML pipeline. You must understand your data before you feed it to the AI/ML models (you can thus use Jupyter or Zeppelin notebooks to do so).

3. Create an AI/ML Pipeline – You will need to make the enriched or transformed data available to your AI/ML Platform via the familiar Spark APIs. The trained model is then served to the intended AI/ML application. Any Spark DataFrame-to-ML package conversions will take place here, as well.

   Although not shown in this diagram, you could use Aerospike as a statestore for your AI/ML pipeline to store any state information.

Model training as part of this pipeline involves selecting a Deep Learning or Machine Learning algorithm and feeding it with data, so that it can learn the underlying patterns.

Model Validation is to make sure that the model can predict accurately for data that was not a part of the training set.

Parameter Tuning is an iterative process and requires a lot of tuning of model parameters a.k.a. parameter tuning. Once the model is ready it is served via a REST endpoint.

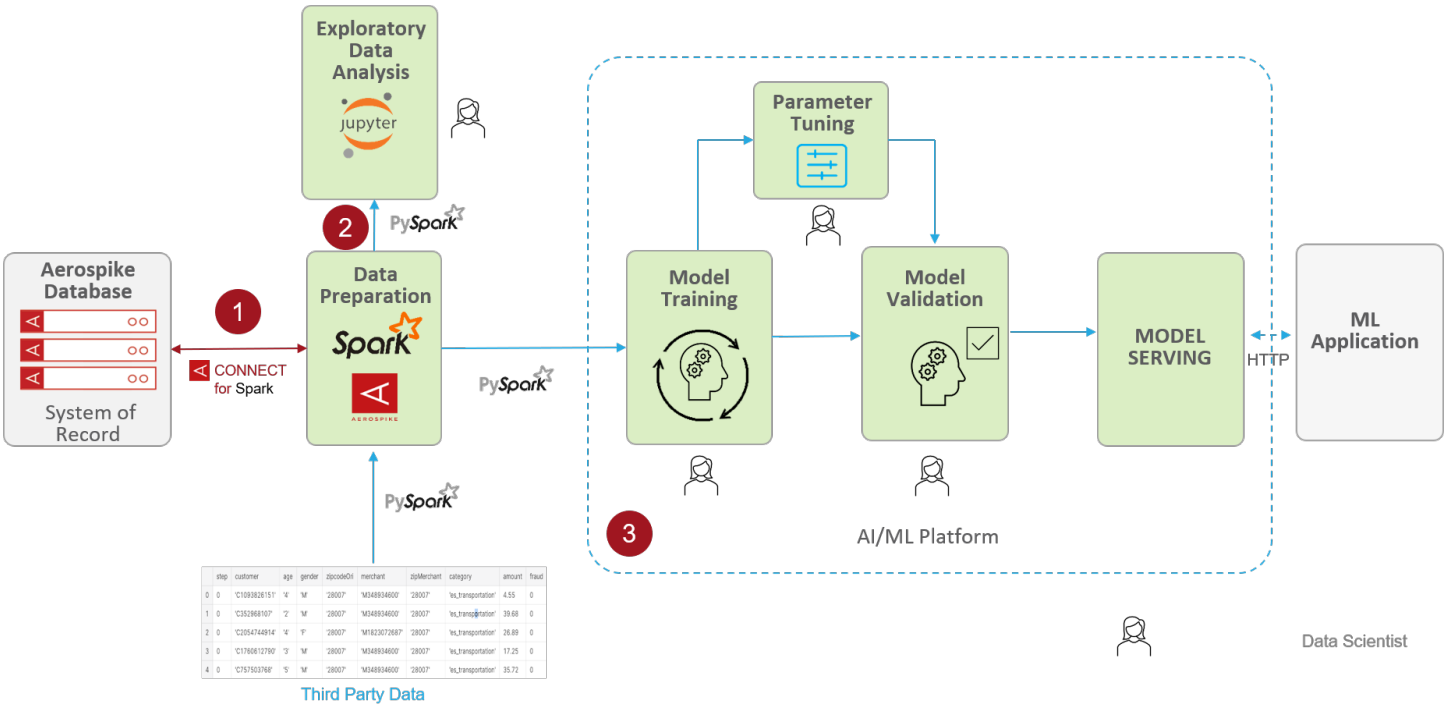



**Figure 3**: Aerospike massive parallelism accelerates the AI/ML training pipeline

## Speeding up the Inference Pipeline

Aerospike creates a very efficient, low latency inference pipeline with Aerospike. Aerospike can ingest data from disparate data sources into an edge system that is comprised of the Aerospike database using an Aerospike C client. (Clients for Go, Java, Scala, and other popular programming languages are available.)

There are multiple benefits of having an Aerospike system at the edge:

- **Providing backpressure** – Gaining the ability to slow down the ingest rate for millions of events per second so that the backend system that runs AI/ML or analytics workloads can catch up
- **Compliance** – Gain the ability to explore data that is ingested into your system by running queries at the edge for example to comply with e.g. GDPR, etc.
- **Filtering** – Filter-out only unnecessary data while preserving as much of the native dataset as possible
- **Span datacenters** – Aerospike allows users to ingest high velocity, high volume at the edge with our multi-site clustering technology which allows multiple sites to behave as a single cluster.

Along with the Aerospike database, Kafka can be used as a streaming source for Spark. Data is prepped/pre-processed in Spark and posted to an inference server. An AI/ML application then persists the predictions for use in the training pipeline.

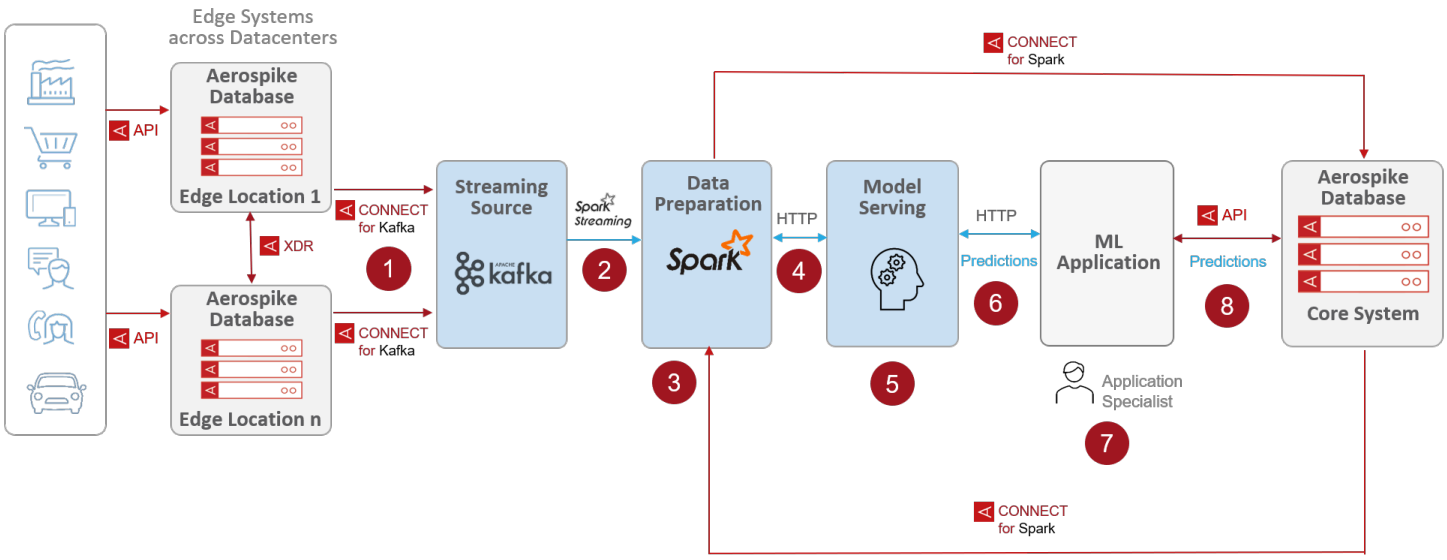



**Figure 4**: Aerospike Inference Pipeline

Per Figure 4, more specifically:

1. Kafka ingests data from Aerospike – via change notifications and then serves as the Streaming source for Spark (given that Spark works on a pull model)

2. Spark pulls event data from Kafka – as soon as they stream in thus avoiding any batching

3. Data preparation – includes cleansing, transformation, enrichment for both batch, or in this case, streaming data, where data is converted to a form that AI/ML models can understand. Depending on the volume of data involved, this may need massive parallelization. It is an I/O intensive process and requires an I/O performant database such as Aerospike.

4. Data Transmission via HTTP – whatever data you have prepped you send to the web server to invoke the model and execute the prediction. This is significant because ML models such as TensorFlow, for example, serve their models via a REST end point. Your Spark streaming application, after pre-processing the transaction would POST (HTTP) to the endpoint and the model would apply the ML model to the transaction.

5. Model Serving – once you have built the model it can be placed in the web server.

6. Inference Output – The model predicts (scoring) based on the input data stream or tensors. Inference can occur either at the edge or in the core system.

7. Application receipt of Score/Output – The application could be a webapp, which is where results are received. Note that applications could be asking the model the questions which would send the inference label to the end user, for example.

8. Persist the scores and write it to Aerospike – Aerospike uses cross-datacenter replication (XDR) to sync with the system of record for future training. To improve accuracy, a human such as a fraud analyst monitors the classification and manually labels them. Human intervention may not be required once the models perform consistently in production environment.

# Case Study: Massive Parallelism with Aerospike and Spark

A global Ad Tech customer was using Aerospike in conjunction with Spark and the Aerospike Connect for Spark pre-built integration for 360-degree profile for each user that visited the customer's digital asset.

In short, the customer experienced remarkable results, slashing the time it took their data science team to process Spark jobs from 12 hours down to 2.4.

Aerospike Connect for Spark has several features for achieving massive parallelism:

- **Scan by partitions** – Aerospike stores data evenly across 4,096 partitions that we can scan in parallel in Spark
- **Predicate pushdown**
- **Partition mapping** – Ability to map 4,096 Aerospike partitions against up to 32K Spark Partitions (which is configurable) for massively parallel reads.

**Massive Parallelization**

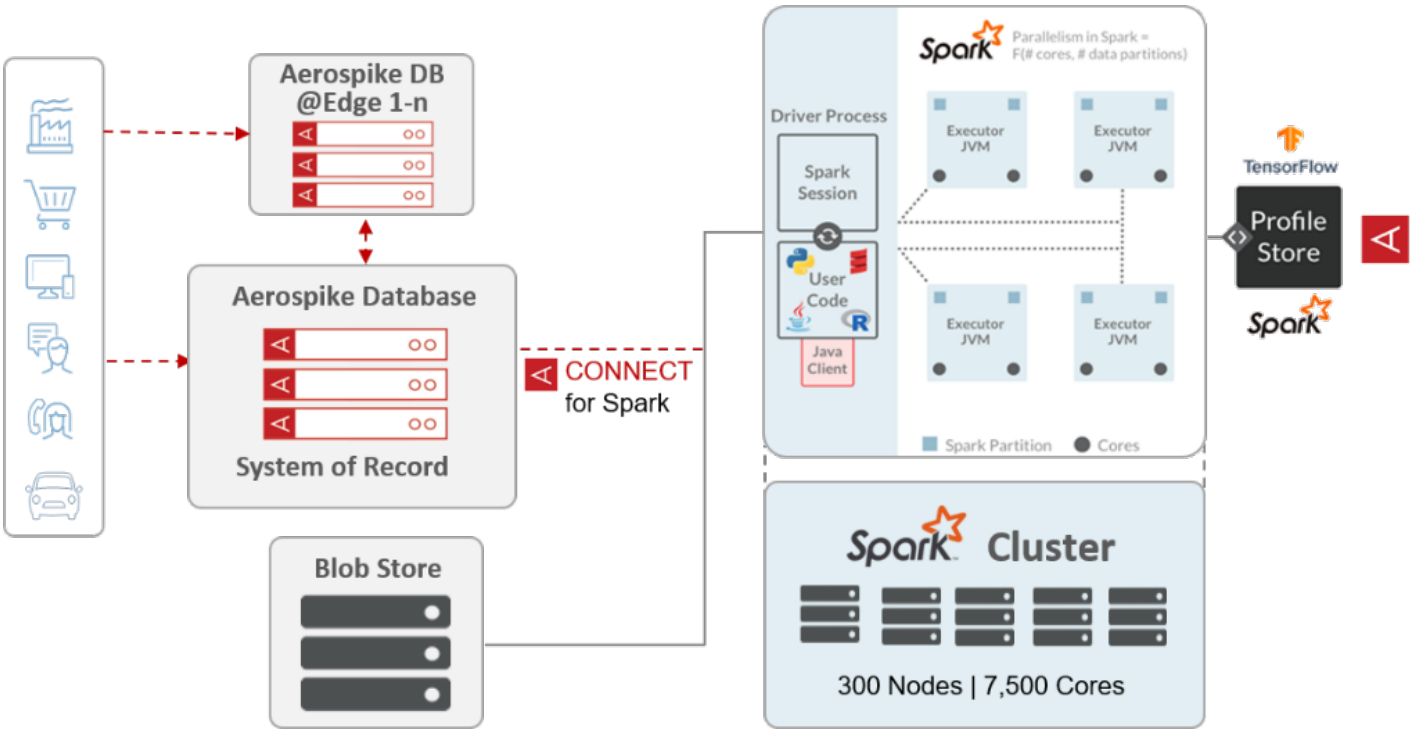
- ✓ 80% reduction in Spark Job Execution time
- ✓ Reduced training time
- ✓ Increase frequency of retraining

**Operational reliability at extreme scale**

- ✓ 13B Objects
- ✓ 150 TB unique data – multiple times a day

**Increased ROI**

- ✓ Only 33 Aerospike servers
- ✓ Increased utilization of Spark Cluster (300 nodes and 7,500 cores)




***"We were using custom code before which led to data quality issues and a complex data infrastructure. With Aerospike, we are processing Spark jobs that used to take 12 hours now in just 2.4."***

Senior Director, Data Science and Engineering
Top Global Ad Tech company

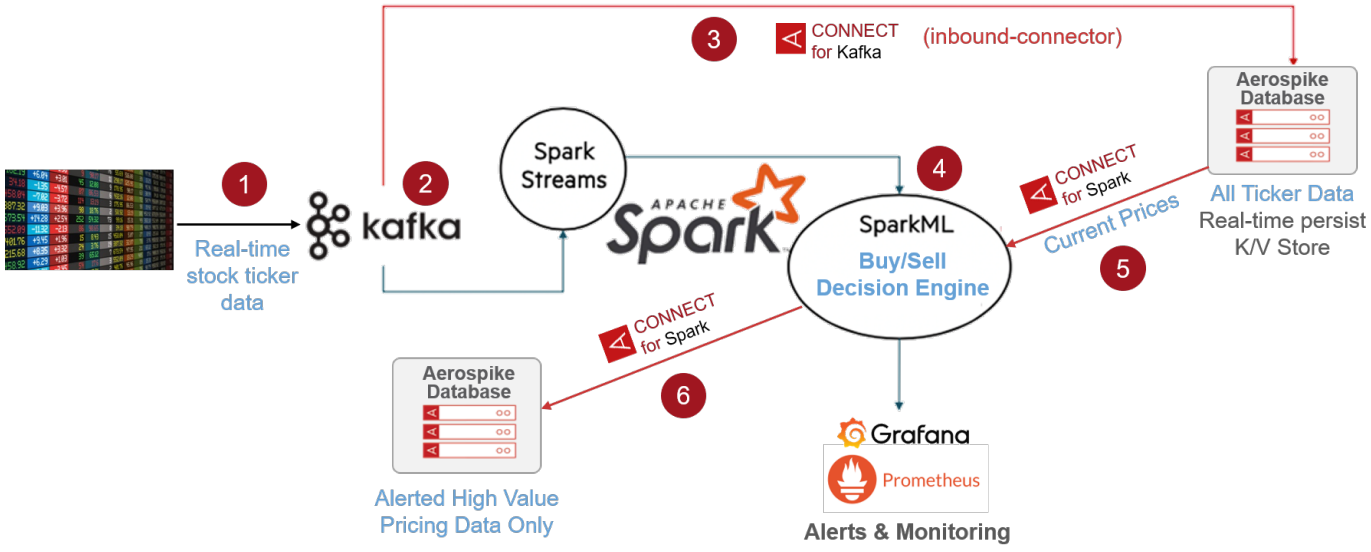
# Case Study: Real-Time Processing of Trading Data with Aerospike

Hewlett Packard Enterprise (HPE), an Aerospike strategic partner, deployed a High Frequency Trading solution with the Aerospike database, Aerospike Connect for Spark, and Aerospike Connect for Kafka pre-built integrations.

Hewlett Packard Enterprise

Per the below figure from left to right:

1. Live prices are streamed into the system through Kafka in JSON format
2. The Kafka cluster acts as a streaming source for Spark Streaming
3. Additionally, prices are streamed through the Kafka connect to an Aerospike instance that serves as a system of record.
4. In Spark, prices are processed individually through the Decision Engine and depending on the ticker a decision to buy or sell is made
5. To enable this decision process, current prices for each stock ticker are obtained from Aerospike. The decision engine uses Spark ML and other ML libraries with the following logic:
   - It checks whether the ticker price satisfies the condition to buy or sell.
   - Prices that do not satisfy the condition are simply ignored.
   - Prices that satisfy the condition (high-value prices) are notified via a real-time dashboard that was built using Prometheus and Grafana.
6. These high value prices are persisted in Aerospike for future reference via Spark APIs that use the Spark Connector.




***"Aerospike is second to none for ingesting and persisting millions of events per second… (Aerospike) allows me to do near-instantaneous machine learning on the data as it lands."***

Theresa Melvin Chief Architect of AI-Driven Big Data Solutions, HPE

AEROSPIKE

**About Aerospike**

Aerospike is trusted by leading enterprises around the world to help them build and deploy modern data architecture solutions with confidence. The Aerospike enterprise-grade non-relational database helps companies power mission critical, strategic operational applications that make digital transformation possible. Powered by a patented Hybrid Memory Architecture™ and autonomic cluster management, Aerospike is used by enterprises in the financial services, telecommunications, technology, retail, e-commerce, adtech, and online gaming industries and is well-suited for fraud prevention, digital payments, recommendation engines, real-time bidding and other applications that require extreme uptime, performance and scale. Aerospike customers include Adobe, Airtel, FlipKart, Kayak, Nielsen, and Snap. The company is headquartered in Mountain View, Calif.



2525 E Charleston Road, Mountain View, CA, 94043 | (408) 462-2376 | aerospike.com