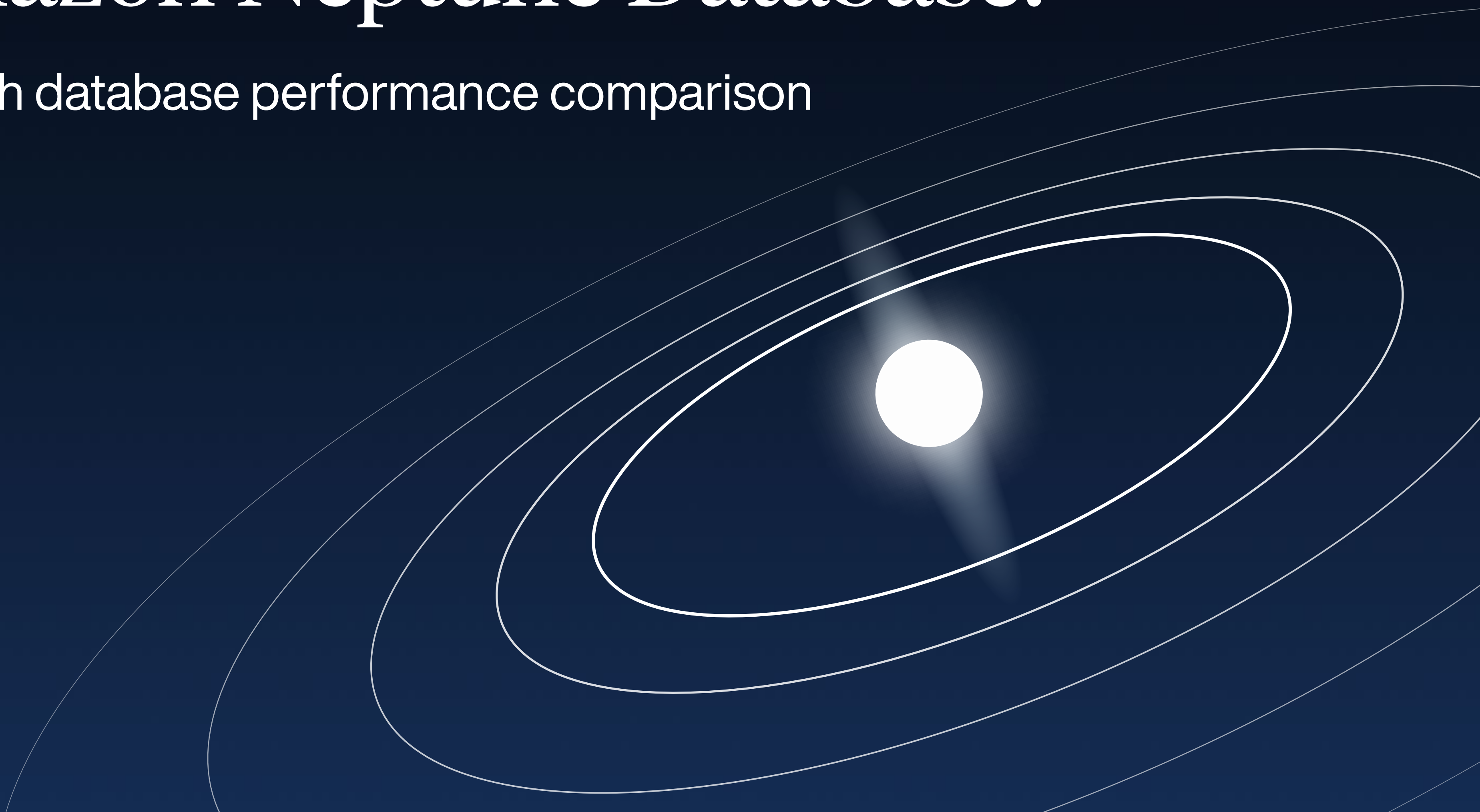
Aerospike | E-BOOK

Aerospike Graph vs. Amazon Neptune Database:

A graph database performance comparison



Executive summary

This benchmark compares **Aerospike Graph v3.0.0** and **Amazon Neptune v1.2.1.2** on a representative identity graph workload, using equivalent hardware and dataset size (scale factor 10M, 35 GB CSVs). The systems were evaluated across four dimensions: **latency, throughput, ingestion speed, and infrastructure cost.**

Aerospike Graph consistently outperformed Amazon Neptune in all metrics, demonstrating clear advantages for identity graph workloads that require fast, scalable, and cost-efficient operations.

We attempted to benchmark both systems on larger datasets (100M and 1B nodes), but **Amazon Neptune was unable to complete ingestion** for these sizes. As a result, this report focuses on the SF10M dataset, where both systems could be tested under the same conditions.

For performance at larger scales, refer to the Aerospike Graph scalability benchmark report, which includes results for SF10M, SF100M, and SF1B.

Key results at SF10M workload. Aerospike Graph provides:

- **Latency:** Up to **92% lower p99 read latency** and **87% lower p99 write latency.**
- **Throughput:** Up to **5x higher throughput**, sustained across concurrency levels.
- **Ingestion speed: 6x faster bulk ingestion** (2.5 hours vs. 16.2 hours)
- **Cost efficiency: 50% lower infrastructure** cost for the same workload.

These results are enabled by two key design elements. First, Aerospike Database's Hybrid Memory Architecture (HMA) provides a high-performance storage engine that the graph layer relies on for predictable low latency, high throughput, and fast retrieval of graph elements during traversals. Second, Aerospike Graph's separation of compute (Aerospike Graph Service) and storage (Aerospike Database) allows both layers to scale horizontally and independently. Together, these capabilities deliver real-time graph performance even at large data scales.

For organizations building identity graphs that demand low-latency resolution, high ingest rates, and real-time activation, Aerospike Graph is the clearly superior choice in both performance and cost.

Benchmarking framework

Our benchmarking framework (Tinkerbench) measures graph database query performance by executing representative Gremlin traversal operations that model real-world application patterns. Tinkerbench is an open-source tool developed by Aerospike and designed for Apache TinkerPop-based graph databases.

The benchmark captures the complete end-to-end operation lifecycle, including network round-trip time, server-side query processing, and result serialization. This measurement approach reflects actual application behavior, where queries execute as blocking operations, where each query waits for completion before proceeding to the next.

Tinkerbench evaluates performance by running multiple concurrent, sequential, and blocking queries across independent worker threads for a sustained duration. A persistent connection pool efficiently manages concurrent load while maintaining blocking semantics per thread.

Performance measurements are gathered across multiple test runs to account for system variability and JVM warmup. Tests run for a fixed duration with a specified number of concurrent threads. Latency is measured under sustained throughput, continuously recording and aggregating individual operation latencies to compute mean values with statistical bounds. This captures query latency under realistic sustained load, not isolated scenarios. Throughput is measured as aggregate operations per second (OPS) completed across all concurrent threads during the fixed time period.

Dataset

This benchmark uses an identity graph dataset that models identity resolution workflows, representing how multiple tracking identifiers (signals) are linked to canonical identity entities (GoldenEntities) through third-party identity providers (Partners). This workload pattern is representative of production identity graph applications used for cross-device targeting, fraud detection, and user identity resolution.

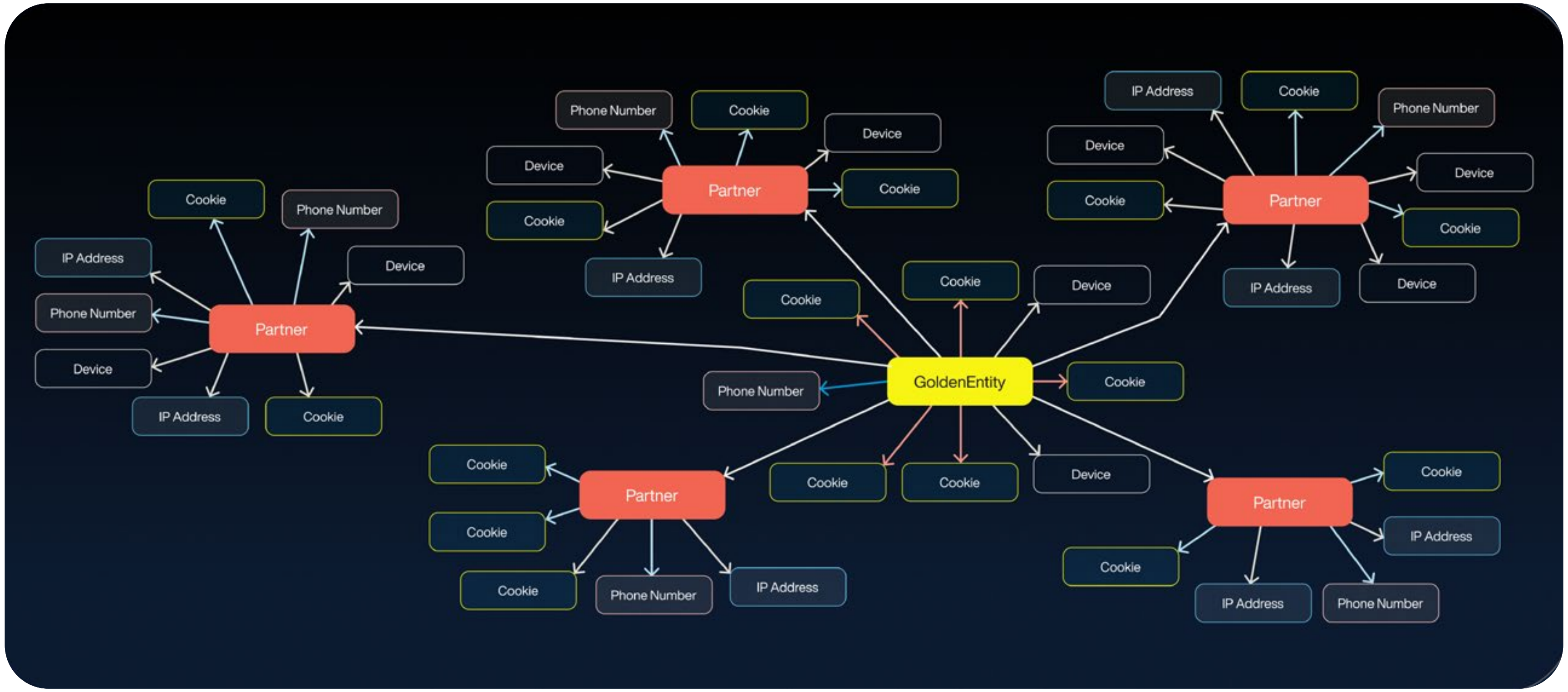
Graph schema

The dataset consists of the following vertex types:

- **GoldenEntity:** Canonical identity records representing unified user identities.
- **Partner:** Third-party identity provider entities (e.g., TDID, Tapad, UID2, Lotame).
- **Signal vertices:** Tracking identifiers including Device, IP Address, Account, Household, and Cookie vertices.

Edge relationships connect these entities:

- GoldenEntity to Signal relationships `(e.g., HAS_DEVICE, HAS_IP_ADDR)`
- GoldenEntity to Partner relationships `(HAS_PARTNER)`
- Partner to Signal relationships `(e.g., PROVIDED_DEVICE, PROVIDED_COOKIE, PROVIDED_IP_ADDR)`
- Partner to GoldenEntity relationships `(HAS_IDENTITY)`
- GoldenEntity to GoldenEntity relationships `(HAS_HOUSEHOLD)`



Scale factors

Three scale factors were evaluated:

- **SF10M:** 370 million vertices, 35 GB CSV source data
- **SF100M:** 3.7 billion vertices, 350 GB CSV source data
- **SF1B:** 37 billion vertices, 3.5 TB CSV source data

This benchmark report focuses on SF10M, as Amazon Neptune was unable to complete bulk loading of larger datasets within reasonable timeframes. Aerospike Graph successfully ingested and queried all three scale factors. The additional large-scale results are presented separately in the Aerospike Graph scalability benchmark report.

Workload

To ensure fair and comparable results across systems, both Aerospike Graph and Amazon Neptune were tested using the same workload using comparable hardware infrastructure: the same client machine, the same benchmark framework instance, and the same number of concurrent worker threads executing queries. This controlled comparison ensures that any performance differences observed are attributable to the database systems themselves rather than variations in test configuration or client-side execution environment.

Query workload

The benchmark suite executes two query categories:

- **Short read queries (SR):** Five read queries that traverse the graph to retrieve signals, devices, or partner information starting from a Device, GoldenEntity, or Partner vertex. These queries model common identity resolution patterns, such as finding all devices associated with a user or listing all tracking identifiers linked to a GoldenEntity.
- **Short write queries (SW):** Five write queries that create or modify graph relationships and properties. These include adding new partner subgraphs, updating timestamps, modifying GoldenEntity properties, and creating household relationships between entities.

Each query was executed with 32 concurrent worker threads, measuring latency from p50 to p999 and capturing throughput in OPS.

Configuration

All tests were conducted on Amazon Web Services (AWS).

The infrastructure for each system was provisioned to reflect real-world production configurations:

Aerospike Graph:

- Aerospike Database: 2 × r5d.2xlarge instances
- Aerospike Graph Service: 1 × c6g.2xlarge instance

Amazon Neptune:

- Database cluster: 3 × r5d.2xlarge instances

Workload generator (Tinkerbench):

- 1 × c6g.2xlarge instance

Deployment and configuration: Aerospike was deployed with the primary index in RAM, and data was stored on NVMe SSDs using HMA mode. The replication factor (RF) was set to two to ensure fault tolerance. (Note: RF2 is a standard Aerospike practice.) Clusters were sized using Aerospike's sizing guidelines such that after the initial data load for each dataset size, 50% of the storage space was left [1] unused.

Amazon Neptune clusters were deployed via the AWS Management Console using equivalent instance types and configurations.

Bulk data ingestion:

- **Aerospike Graph:** Data was ingested from CSV files in Amazon S3 using the Aerospike Graph bulk loader, which runs on an Apache Spark EMR cluster to efficiently load large volumes of graph data from external sources.
- **Amazon Neptune:** Data was ingested from CSV files in Amazon S3 using the Amazon Neptune bulk loader.

[1] Capacity planning guide | Fragmentation Considerations

Key performance metrics

The following metrics capture the comparative performance characteristics of both systems under identical test conditions.

1. **Latency:** Measures the response time for read and write queries, reported at p50, p99, and p999 to represent typical and tail latencies. Lower latency with minimal variance indicates faster response times and more predictable performance.
2. **Throughput:** Represents the total number of OPS each system sustained across concurrent clients for a set of read queries. Higher throughput indicates greater capacity to handle concurrent queries.
3. **Data ingestion efficiency:** Evaluates the time required to bulk-load the SF10M dataset, measured in hours. Ingestion efficiency directly affects both the initial data population and the ongoing refreshes of large graph datasets.
4. **Cost efficiency:** Compares the costs required to support the SF10M workload under equivalent configurations. This includes both instance and license costs, showing the relative cost-effectiveness of each system for comparable workloads.

Results

This section presents the findings from our benchmarking study, focusing on key performance metrics that provide a framework for comparing the capabilities of Aerospike Graph and Amazon Neptune Database in handling large-scale identity graphs.

Latency and throughput

Latency is a crucial metric because it directly impacts user experience and overall system performance. Lower latency indicates faster response times, while minimal variance across percentiles reflects predictable performance, which is essential for real-time applications.

Aerospike Graph vs. Amazon Neptune Database read latency

The following charts illustrate the comparative read query latency of both systems. Figure 1 shows p99 latency for each test query. Subsequent charts display latency distributions by percentile to highlight performance consistency. All measurements were taken using 32 concurrent threads from the benchmark driver (Tinkerbench).

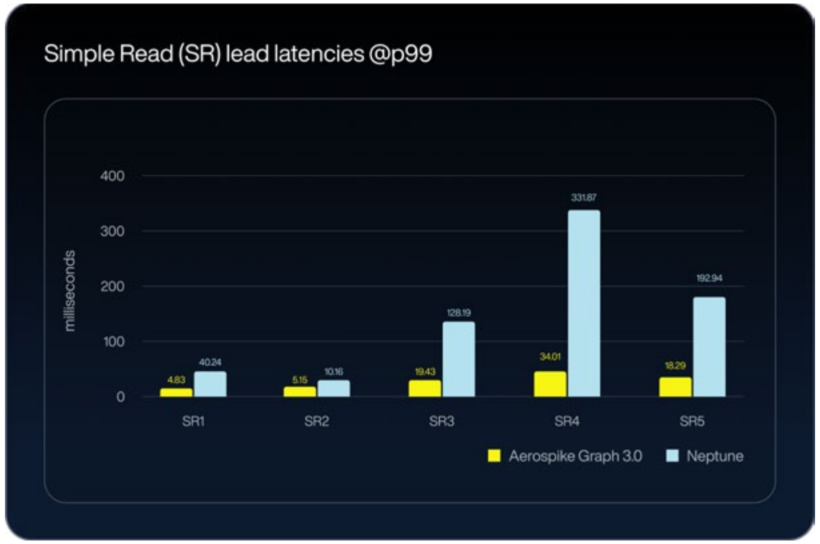


Fig. 1. Comparison of Aerospike Graph and Amazon Neptune Database for latency at p99 of five test read queries.

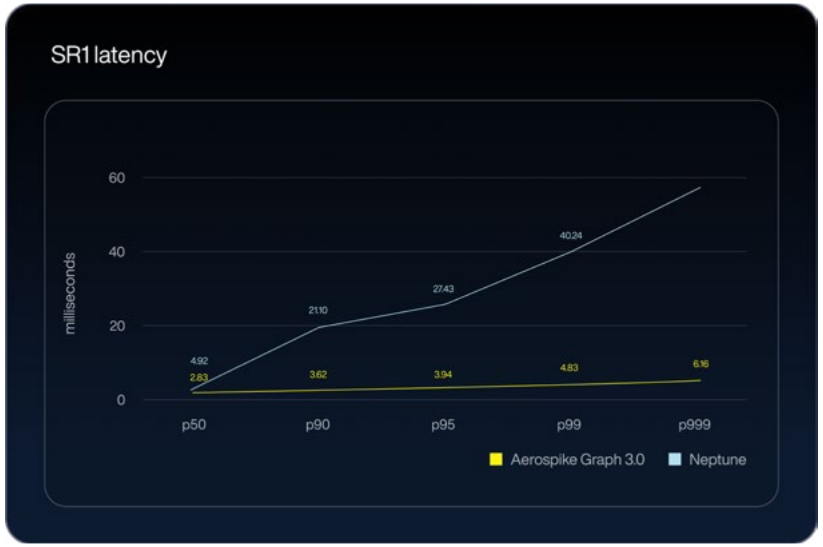


Fig. 2. Find all the partner vertices connected to a GoldenEntity.

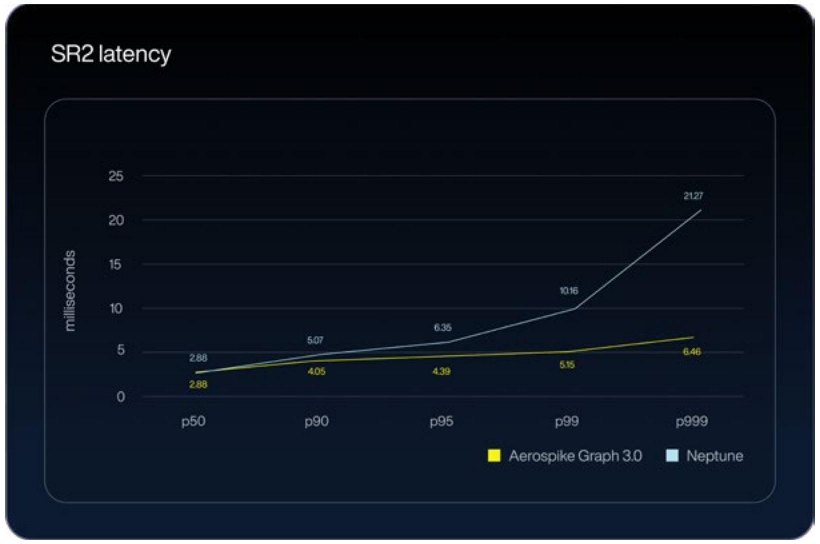


Fig. 3. Start with GoldenEntity and get all signals provided by a partner.

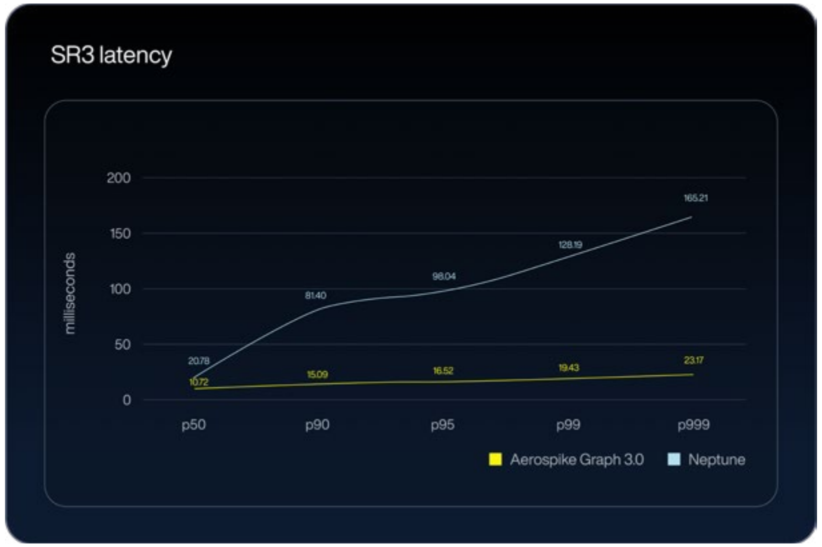


Fig. 4. Fetch all signals (including ones provided by partners) for a given GoldenEntity.

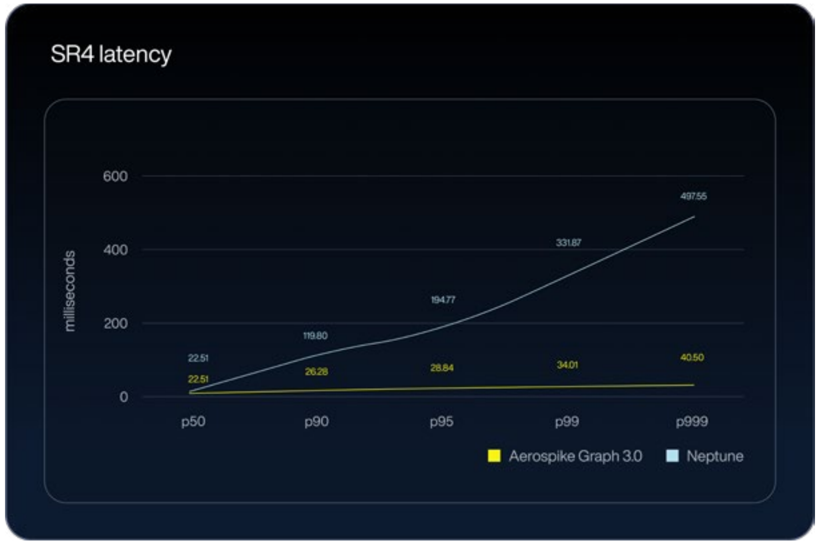


Fig. 5. List all tracking identifiers (cookies, device IDs, IP addresses, etc.) associated with a device

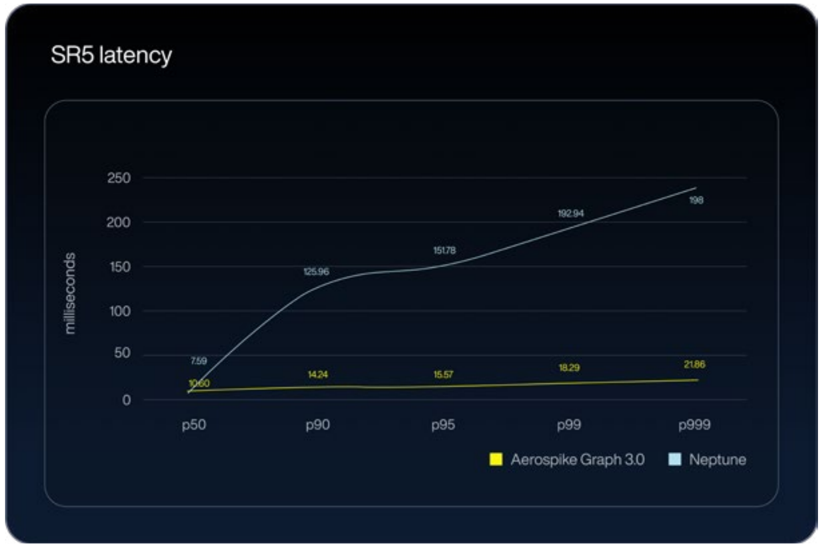


Fig. 6. Three-hop traversal to find connected devices for a GoldenEntity.

Aerospike Graph exhibited lower latency than Amazon Neptune Database across all percentiles and queries tested. At the critical p999 latency measurement, Aerospike Graph demonstrated significantly lower latency than Amazon Neptune Database across all tests.

Aerospike Graph read query latency shows minimal variance between percentiles, indicating consistent performance across typical (p50) and extreme (p99, p999) conditions. This consistency is important to ensure predictable performance and a consistent user experience.

Write latency

The following charts illustrate each system's comparative write query latency. The first chart shows write latency for each system by test query at p99. Subsequent charts show latency results for particular test queries organized by percentile, as performance according to service level agreements (SLAs) is frequently a critical factor in system design.

Aerospike Graph exhibited lower write latency than Amazon Neptune Database across all test queries and percentiles.

Latency for both systems was measured using 32 concurrent worker threads.

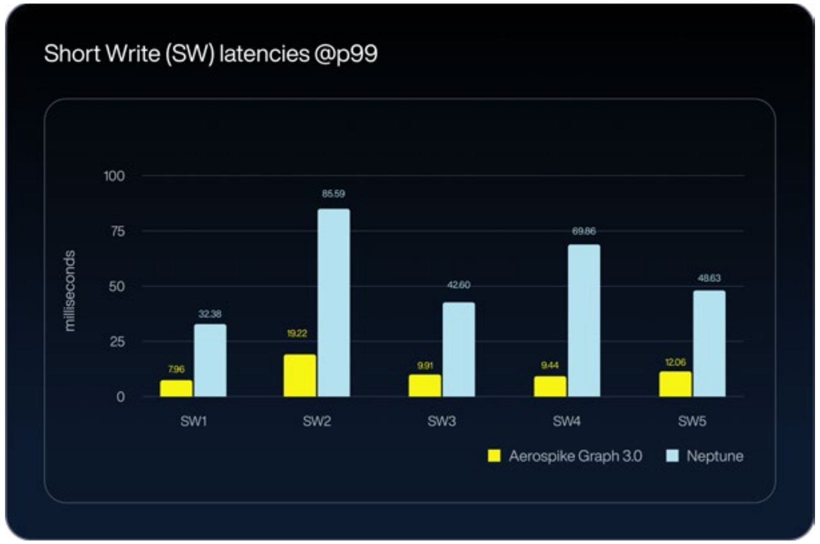


Fig. 7. Comparison of Aerospike Graph and Amazon Neptune Database for latency at p99 of five test write queries.

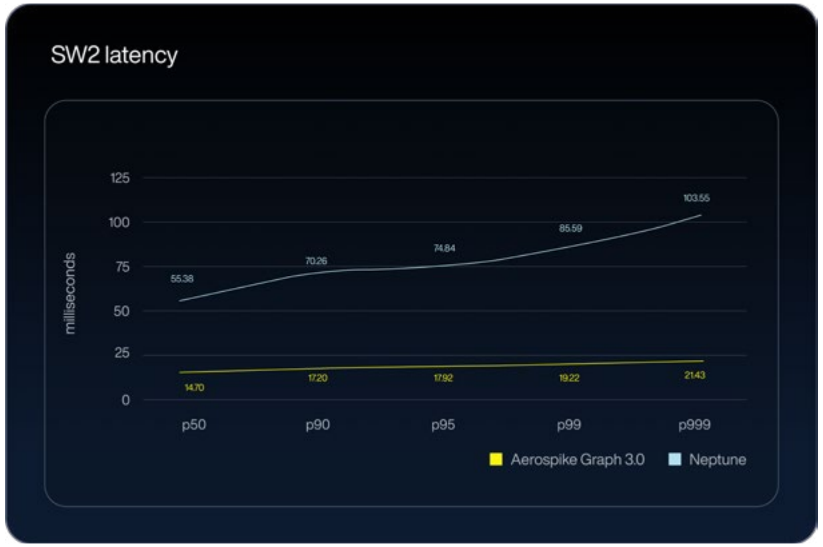


Fig. 8. Merge two GoldenEntities into a household

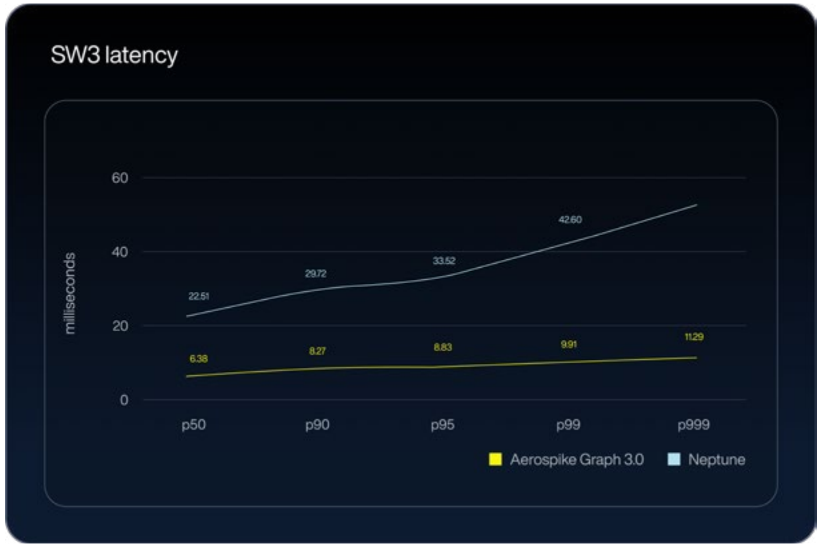


Fig. 10. Update the time stamps for signal vertices connected to a partner.

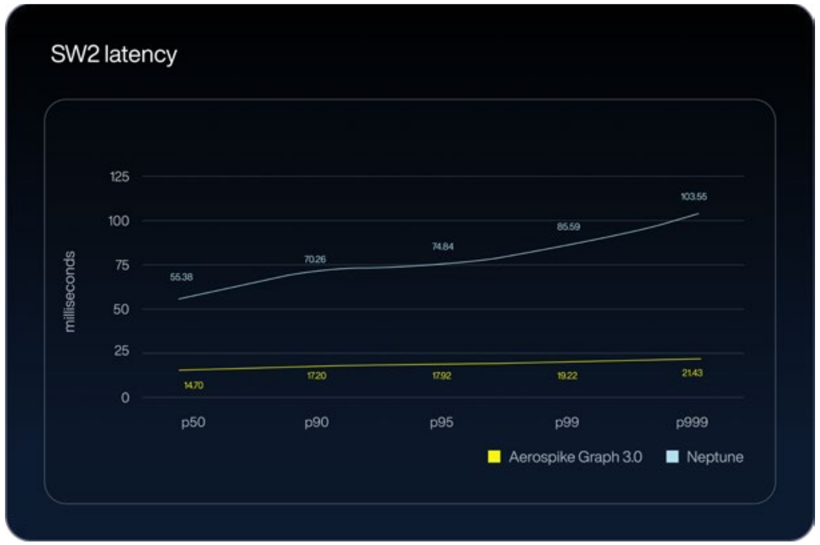


Fig. 9. Register a new partner and associate it with existing entities.

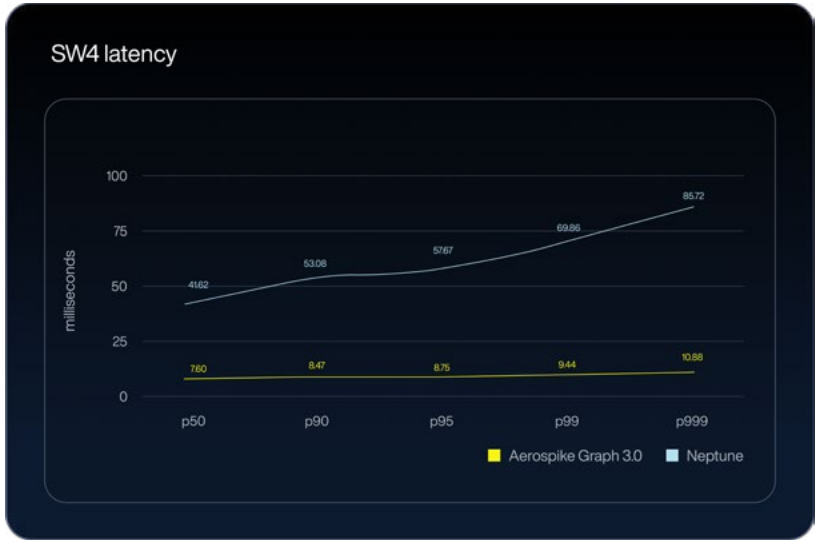


Fig. 11. Modify the properties of a GoldenEntity.

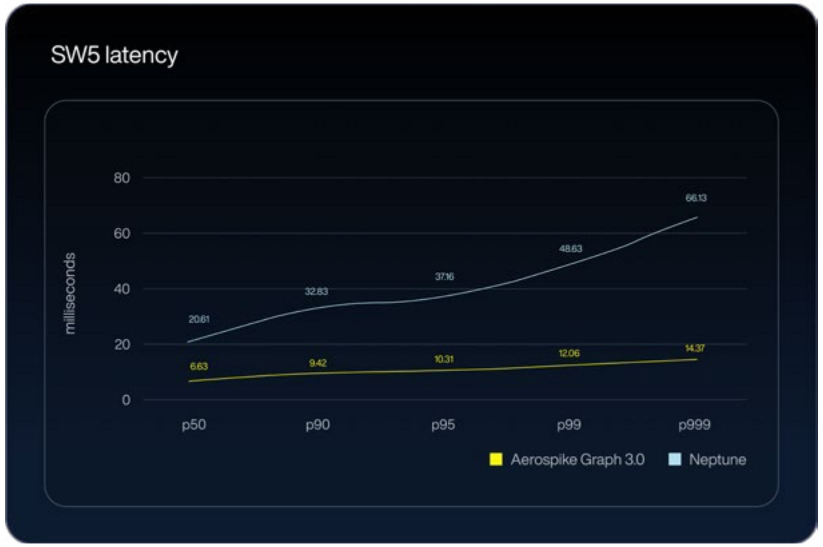


Fig. 12. Update the score property on edges to partners based on a specific type.

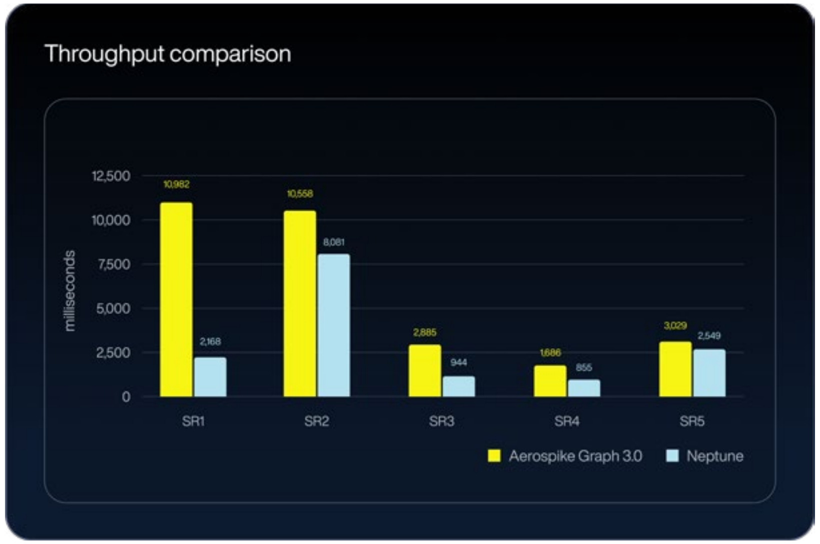


Fig. 13 Comparison of Aerospike Graph and Amazon Neptune Database for throughput of five test read queries.

Throughput

Under the same workload generator pressure (32 concurrent threads continuously executing queries), Aerospike Graph achieved significantly higher throughput than Amazon Neptune Database across all read queries. This higher throughput results from Aerospike Graph's lower query latency, allowing more operations to complete within the same time window despite identical load generation parameters.

The throughput comparison shows that Aerospike Graph achieved higher throughput than Amazon Neptune Database in all read scenarios. The greatest difference was observed for query SR1, where Aerospike Graph achieved 10,982 operations per second compared to Amazon Neptune Database's 2,168 operations per second, representing a 4x increase.

Bulk load / Ingestion performance and efficiency

Our tests evaluated the performance and efficiency of bulk data loading across different dataset sizes. Aerospike Graph successfully loaded all three scale factors (SF10M, SF100M, and SF1B). Amazon Neptune Database was only able to load the SF10M dataset within reasonable timeframes.

When we attempted to load the SF100M dataset using the Amazon Neptune bulk loader, the process failed after 26 hours. After resizing the Amazon Neptune Database writer instances to r5d.24xlarge, we attempted again and aborted after the initial bulk load speed indicated that ingestion would take over three weeks to complete. We did not attempt to ingest SF1B with Amazon Neptune Database, as we calculated that it would take approximately a year to complete. Please see Appendix A for a detailed description of the steps taken and calculations.

For performance at larger scales, refer to the Aerospike Graph scalability benchmark report, which includes results for SF10M, SF100M, and SF1B.

Bulk load performance

Scale factor	CSV size (GB)	Aerospike graph bulkload		Amazon Neptune bulkload	
		Infrastructure	Time (hours)	Infrastructure	Time (hours)
SF10M	35	EMR: 4@r5.2xlarge ASDB: 2@r5d.2xlarge	2.5	3 X r5d.2xlarge	16.2

Table. 1. Bulk load performance comparison for SF10M dataset.

ASDB is Aerospike Database. EMR cluster instances are used only during bulk load operations and are terminated after ingestion completes. Costs are calculated using AWS on-demand pricing in the us-east-1 region.

Multiple factors influence bulk load performance, including dataset size, cluster capacity, and worker configuration. However, the fundamental difference in bulk load architectures between the two systems explains the significant scalability gap observed.

Architectural differences

Aerospike Graph bulk loader: Aerospike Graph bulk loader uses Apache Spark, a distributed processing framework that enables horizontal scaling. The loader partitions input data across multiple Spark workers, which process data chunks in parallel and write directly to the Aerospike Database cluster. This distributed architecture allows ingestion throughput to scale linearly with the number of Spark workers. As more workers are added, more data can be processed concurrently. The independent compute and storage layers in Aerospike Graph's architecture also enable ingestion to proceed without impacting query performance, as compute resources can be allocated separately for ingestion workloads.

Amazon Neptune bulk loader: Amazon Neptune bulk loader reads data from Amazon S3 and processes it through Neptune's cluster architecture. Neptune uses a single primary writer instance architecture, where all write operations, including bulk loads, must route through the primary writer instance. While Neptune supports multiple read replicas for query scaling, these replicas do not participate in write operations. This single-writer architecture creates a throughput bottleneck that cannot be alleviated by adding more instances or scaling resources. As dataset size increases, the primary writer becomes the limiting factor, and ingestion throughput does not scale proportionally with cluster size.

Scalability implications

The distributed nature of Aerospike Graph bulk loader enables near-linear scaling. Adding Spark workers allows proportionally more data to be ingested in the same timeframe. This explains Aerospike Graph's ability to ingest SF100M (3.7 billion vertices) in eight hours and SF1B (37 billion vertices) in 31.8 hours, maintaining consistent throughput rates as the dataset size increases.

In contrast, Neptune's single-writer architecture creates a fixed throughput ceiling. As the dataset size increases from SF10M to SF100M to SF1B, ingestion times grow exponentially because the primary writer cannot process data faster, regardless of cluster configuration. This architectural limitation explains why Neptune could not complete the SF100M and SF1B dataset ingestion within reasonable timeframes, despite upscaling to larger instance types.

Infrastructure costs

This section evaluates the infrastructure costs for Aerospike Graph and Amazon Neptune Database for the SF10M dataset.

Cost comparison methodology:

The cost comparison includes infrastructure and license costs. For Aerospike Graph, this covers compute, storage, and license costs, but excludes operational management, since Aerospike Graph is currently self-managed. For Amazon Neptune, costs are based on fully managed service pricing, which bundles infrastructure, licenses, and operational management.

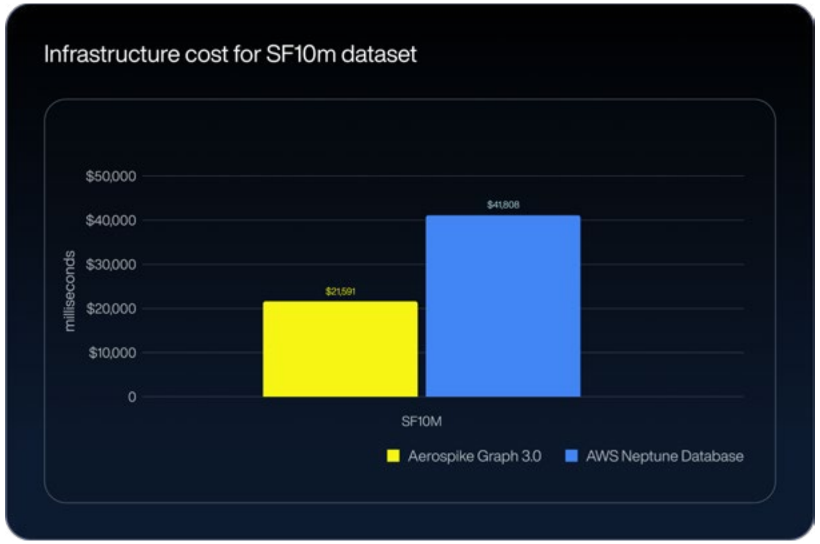


Fig. 14. Aerospike Graph can be deployed at 50% lower infrastructure cost compared to Amazon Neptune for the same dataset size.

The results demonstrate that Aerospike Graph can be deployed for a quarter of the infrastructure cost compared to Amazon Neptune for the same SF10M (35 GB CSV) dataset and workload profile.

Benchmark summary

This benchmark study evaluated Aerospike Graph and Amazon Neptune Database, comparing their capabilities in handling large-scale identity graphs with respect to scalability, performance, and cost efficiency.

Our study highlights three key advantages of Aerospike Graph over Amazon Neptune Database:

Predictable query latencies: Aerospike Graph delivered between 48% and 92% lower latency at p99 for read queries and between 75% and 87% lower latency at p99 for write queries.

Predictable performance: Aerospike Graph exhibited predictable performance with minimal variance in latency between p50 (typical case) and p999 (most stringent), ensuring consistent and predictable performance at scale. Amazon Neptune Database demonstrated greater variance, with an average variance of 9,127.0 ms vs. 16.2 ms, than Aerospike Graph between p50 and p999 in all tests.

Higher throughput: Aerospike Graph also delivered between 16% and 344% higher throughput than Amazon Neptune Database.

Efficient bulk loading: Aerospike Graph ingested the SF10M dataset more than six times faster than Amazon Neptune Database (2.5 hours vs. 16.2 hours). Whereas Aerospike Graph ingested SF100M in eight hours and SF1B in 31.8 hours, Amazon Neptune Database failed to ingest these datasets in a timely manner, and we estimate that the SF100M would have taken a month to load, while the SF1B would have taken over a year.

These results demonstrate that Aerospike Graph can effectively store and query at lower latency, higher throughput, and lower cost than Amazon Neptune Database. Aerospike Graph read query latency shows minimal performance degradation even at extreme percentiles (p99 and p999). This consistency is vital for the high-fan-out workloads typical of graph use cases, where the slowest response dictates the overall experience.

Amazon Neptune Database uses a single write master, limiting write throughput despite multiple read replicas. This bottleneck impacts high-volume, real-time data ingestion. Both platforms offer scalable compute and storage. Neptune automatically scales storage, while Aerospike provides vertical and horizontal scaling for cost control. Aerospike achieves durability and high availability with fewer replicas (RF2 vs. Neptune's RF3), offering unlimited replicas and better resource efficiency, reducing operational costs and energy.

Conclusion

This benchmark demonstrates that, on a real-world identity graph workload, Aerospike Graph delivers:

● LATENCY

92% lower tail latency

87% lower p99 write latency

● QUERY THROUGHPUT

5x higher

● BULK LOADING

6x faster of a 370 million-vertex dataset

● INGESTION

3.7B vertices in eight hours

37B vertices in under 32 hours

● INFRASTRUCTURE COST

~50% lower

These differences are not incremental; they are architectural. Amazon Neptune Database's single-writer design imposes a hard ceiling on write and ingestion scalability that has not been removed in subsequent releases.

Neptune remains a solid, fully managed choice for smaller graphs, read-heavy workloads, or teams that prioritize AWS ecosystem convenience over raw performance and cost. For everything else at scale, Aerospike Graph is the proven, higher-performing, lower-cost alternative.

Amazon Neptune Database, as a fully managed AWS service, provides operational convenience for customers already invested in the AWS ecosystem. It handles patching, backups, and high availability automatically, with integration into AWS IAM, CloudWatch, and VPCs. However, its elasticity is constrained by AWS's managed scaling policies, and it can be less predictable under high-volume, transactional loads requiring low latency.

Aerospike Graph, in contrast, is designed for maximum real-time operational agility in any environment. It inherits Aerospike's proven low latency, horizontally scalable key-value foundation, enabling millisecond-level graph traversals at petabyte scale. It gives operators more control over deployment, enabling choice between on-premises, hybrid, or multi-cloud deployments, while maintaining consistent performance and infrastructure cost control with independently scalable compute and storage layers.

Appendix A

Item A: Successful loading of SF10M

```
{
status: "200 OK",
payload: {
feedCount: [
{
LOAD_COMPLETED: 39832
}
],
overallStatus: {
fullUri: "s3://identity-benchmark/SF10M",
runNumber: 1,
retryNumber: 0,
status: "LOAD_COMPLETED",
totalTimeSpent: 58258,
startTime: 1742590707,
totalRecords: 2235778690,
totalDuplicates: 0,
parsingErrors: 0,
datatypeMismatchErrors: 0,
insertErrors: 0
}
}
}
```

Item B: Failed SF100M load - 26 hours into load and 0 progress.

Initially, we tried to load this with eight r5d.4xlarge, but after 26 hours, this load had gotten nowhere.

```
{
status: "200 OK",
payload: {
feedCount: [
{
LOAD_NOT_STARTED: 395999
}
],
overallStatus: {
fullUri: "s3://identity-benchmark/SF100M",
runNumber: 1,
retryNumber: 0,
status: "LOAD_IN_PROGRESS",
totalTimeSpent: 94700,
startTime: 1742928945,
totalRecords: 0,
totalDuplicates: 0,
parsingErrors: 0,
datatypeMismatchErrors: 0,
```

Appendix A

```
insertErrors: 0
},
errors: {
startIndex: 0,
endIndex: 0,
loadId: "aadf2504-e859-4fdd-b6e2-61a26f215e0d",
errorLogs: []
}
}
}
```

Item C: Failed SF100M load - 16 hours into loading the cookies vertices.

We then tried re-running, passing the folders in one vertex label at a time.

After about 16 hours, it was projected to take another 13 hours, so ~30 hours per folder, and we have nine folders.

In total, we expected this to take about 10 days to load vertices, which usually takes a fraction of the time that edges take, so this would likely take nearly a month to load.

```
{
status: "200 OK",
payload: {
feedCount: [
{
LOAD_NOT_STARTED: 67343
},
{
LOAD_IN_PROGRESS: 20
},
{
LOAD_COMPLETED: 80654
}
],
overallStatus: {
fullUri: "s3://identity-benchmark/SF100M/vertices/
Cookie",
runNumber: 1,
retryNumber: 0,
status: "LOAD_IN_PROGRESS",
totalTimeSpent: 57689,
startTime: 1743032977,
totalRecords: 4032840000,
totalDuplicates: 0,
parsingErrors: 0,
datatypeMismatchErrors: 0,
insertErrors: 0
},
```

Appendix A

```
errors: {
startIndex: 0,
endIndex: 0,
loadId: "2f8d9e9f-0d5c-4d20-9359-f12cd427c455",
errorLogs: []
}
}
}
loadId: "aadf2504-e859-4fdd-b6e2-61a26f215e0d",
errorLogs: []
}
}
}
```

Item D: Failed SF100M load - Using db.r5d.24xlarge instead

To exhaust options, we tried upping the writer instance to the largest available instance (r5d.24xlarge).

We were able to load the cookie vertices; this took 17 hours, which is faster than the ~30 hours per folder we saw with the r5d.4xlarge. However, it would still take more than six days to load the vertices, and then likely more than two weeks to load the edges (being generous with that).

Because of this, we decided to abandon the SF100M experiment and the SF1B experiment.

```
{
status: "200 OK",
payload: {
feedCount: [
{
LOAD_COMPLETED: 148017
}
],
overallStatus: {
fullUri: "s3://identity-benchmark/SF100M/vertices/
Cookie",
runNumber: 1,
retryNumber: 0,
status: "LOAD_COMPLETED",
totalTimeSpent: 63567,
startTime: 1743096756,
totalRecords: 7400002365,
totalDuplicates: 0,
parsingErrors: 0,
datatypeMismatchErrors: 0,
insertErrors: 0
}
}
}
```

Appendix B

Tinkerbench Configuration

```
graph.client.maxConnectionPoolSize=32
graph.client.minConnectionPoolSize=32
graph.client.maxInProcessPerConnection=8
graph.client.maxSimultaneousUsagePerConnection=8
graph.client.minSimultaneousUsagePerConnection=8
benchmark.measurementForks=1
benchmark.measurementIterations=100
benchmark.measurementTime=60
benchmark.measurementTimeout=1000
benchmark.measurementThreads=32 / 64 ← Adjust as
needed
benchmark.idBufferSize=500000
benchmark.mode=sample <---- For latency
benchmark.mode=throughput <---- For Throughput - 1 &
Throughtput - 2}
```

About Aerospike

Aerospike is the real-time database built for infinite scale, speed, and savings. Our customers are ready for what's next with the lowest latency and the highest throughput data platform. Cloud and AI-forward, we empower leading organizations like Adobe, Airtel, Criteo, DBS Bank, Experian, PayPal, Snap, and Sony Interactive Entertainment. Headquartered in Mountain View, California, our offices include London, Bangalore, and Tel Aviv.

For more information, please visit https://www.aerospike.com.

2440 W. El Camino Real, Suite 100, Mountain View, CA 94040 | (408) 462-2376