Aerospike | E-BOOK

# Benchmark: Aerospike vs. ScyllaDB

# Table of contents

# Executive summary

This benchmark evaluates the performance and cost efficiency of Aerospike and ScyllaDB when deployed to support 3 TB and 6 TB datasets under a mixed 70% read / 30% write workload. Each system was tested under two access patterns:

- A uniform distribution, designed to minimise cache effectiveness and expose storage-layer performance.
- A hotspot distribution, configured to achieve approximately 50% cache hit rate, in order to assess the impact of caching.

Across all tested scenarios, Aerospike consistently delivered higher sustained throughput, significantly lower latency, and more predictable performance than ScyllaDB. These results held as the dataset size doubled and under both cache-favorable and cache-unfavorable conditions.

## Key results summary

| Dataset | Workload | Aerospike throughput (ops/s) | ScyllaDB throughput (ops/s) | Aerospike P99 read (ms) | ScyllaDB P99 read (ms) | Aerospike P99 write (ms) | ScyllaDB P99 write (ms) |
|---|---|---|---|---|---|---|---|
| 3 TB | Uniform (low cache) | 487,865 | 196,801 | 0.92 | 17.53 | 0.90 | 3.50 |
| | Hotspot (~50% cache) | 632,659 | 261,544 | 0.98 | 15.30 | 1.20 | 5.20 |
| 6 TB | Uniform (low cache) | 973,824 | 308,614 | 0.98 | 32.51 | 1.65 | 6.71 |
| | Hotspot (~50% cache) | 1,205,757 | 449,102 | 0.99 | 15.70 | 1.82 | 4.47 |

## Principal findings

**Efficiency and cost**

- Aerospike was provisioned with approximately **one-third fewer infrastructure resources** than ScyllaDB in all tests, due primarily to differences in recommended replication factors.
- Despite this, Aerospike consistently outperformed ScyllaDB across throughput, latency, and stability metrics, resulting in substantially better performance per unit of provisioned capacity.

**Throughput**

- Aerospike sustained **2.5×–3× higher throughput** than ScyllaDB across all configurations.
  When cluster size doubled from 3 TB to 6 TB, Aerospike demonstrated near-linear scalability, while ScyllaDB exhibited sub-linear scaling.

**Latency**

- Aerospike maintained s**ub-millisecond P99 read latency** in every test, regardless of cache effectiveness.
  ScyllaDB's P99 read latency ranged from approximately 1**5 ms to over 30 ms**, with significantly higher variability.
- Aerospike's write latency closely mirrored its read latency, indicating efficient handling of mixed workloads.
- ScyllaDB's write latency was materially better than its read latency, suggesting it is comparatively more suited to write-heavy workloads, though still several milliseconds slower than Aerospike at P99.

**Predictability and tail latency**

- Aerospike exhibited **tightly bounded latency,** with minimal jitter and low peak-to-trough variation at both P99 and P999.
- ScyllaDB showed substantially greater variability, with long-tail latencies extending into the tens and, in some cases, hundreds of milliseconds.
- Aerospike's peak observed latency remained under **7 ms** across all tests, whereas ScyllaDB's exceeded **200 ms** in extreme cases.

**Effect of caching**

- Aerospike benefited modestly from caching in terms of throughput, but its latency remained effectively unchanged, demonstrating that its performance does not depend on cache hit rates.
- ScyllaDB showed measurable improvements with higher cache hit rates, particularly for read latency; however, even with caching enabled, it remained an order of magnitude slower than Aerospike running under cache-unfavorable conditions.

## Overall conclusion

Under the tested conditions, Aerospike demonstrated c**onsistently stronger performance characteristics** than ScyllaDB across throughput, latency, predictability, and scalability, while also requiring fewer infrastructure resources. These results suggest that Aerospike is particularly well-suited for latency-sensitive, high-throughput workloads at multi-terabyte scale, where predictable tail latency and efficient scaling are critical operational requirements.

As with any benchmark, alternative configurations and workload profiles may yield different results. Readers are encouraged to consider these findings alongside their own workload characteristics and to validate performance under representative conditions.

# Fairness and limitations

Our evaluation was conducted with a strong emphasis on fairness and methodological rigour, assessing both technologies from multiple technical angles and focusing exclusively on measurable behaviour rather than marketing claims.

We designed and executed the tests impartially, though one challenge was unavoidable: the two systems have fundamentally different architectures, making it impossible to construct a single test environment that suits both equally. To address this, we selected the configuration that best fits each technology, even when this required differences in the underlying infrastructure. The benchmark explains this rationale in detail and why we believe it offers the fairest basis for comparison.

We are confident in the accuracy and objectivity of the results; however, some margin of error is inevitable. Cloud instances can vary slightly from run to run, even within the same instance type, due to hardware wear, physical placement within an availability zone, and the presence of noisy neighbours on shared infrastructure. These variations are minor but unavoidable in any cloud-based benchmark.

That said, if new information emerges that materially changes the interpretation of either system's behaviour, we remain open to revisiting and refining our conclusions.

# Introduction

This benchmark compares two high-performance, horizontally scalable NoSQL databases:

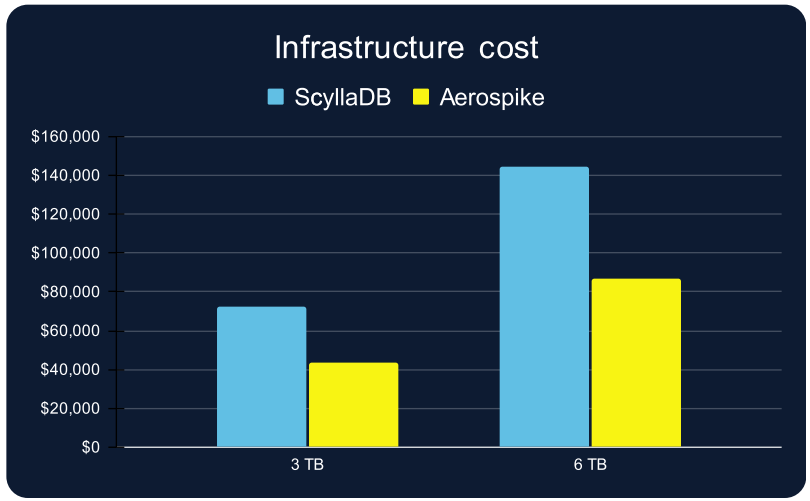
- **Aerospike** is known for delivering near-in-memory performance with highly predictable latency, even at P999 and beyond, despite not storing or caching record data in memory.
- **ScyllaDB** began as a C++ re-implementation of Apache Cassandra. It has since evolved into a next-generation, high-throughput, low-latency database that extends well beyond the original Cassandra ecosystem.

Both technologies are widely used in performance-sensitive and latency-critical environments, making a direct, data-driven comparison particularly valuable for organizations seeking to select the most suitable platform.

We first turn to the results. The subsequent sections describe the benchmark methodology and detail how the tests can be reproduced.

# Benchmark results

## Efficiency (Lower numbers are better)

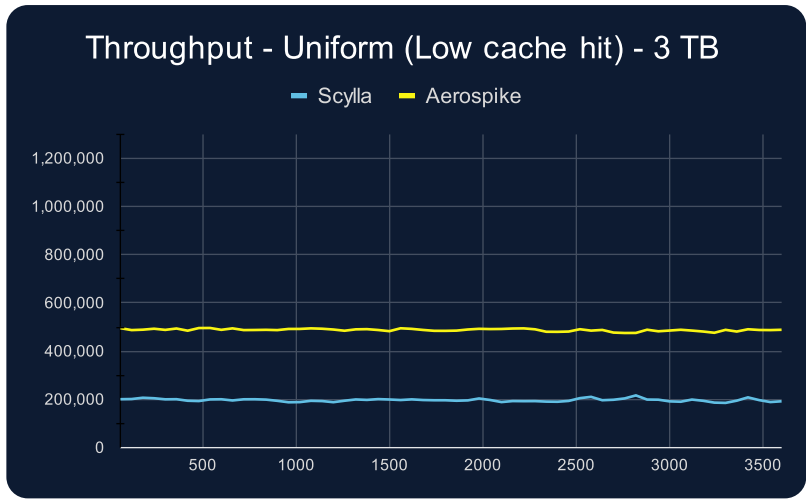



| | ScyllaDB | Aerospike |
|---|---|---|
| 3 TB | $72,165 | $43,293 |
| 6 TB | $144,330 | $86,585 |

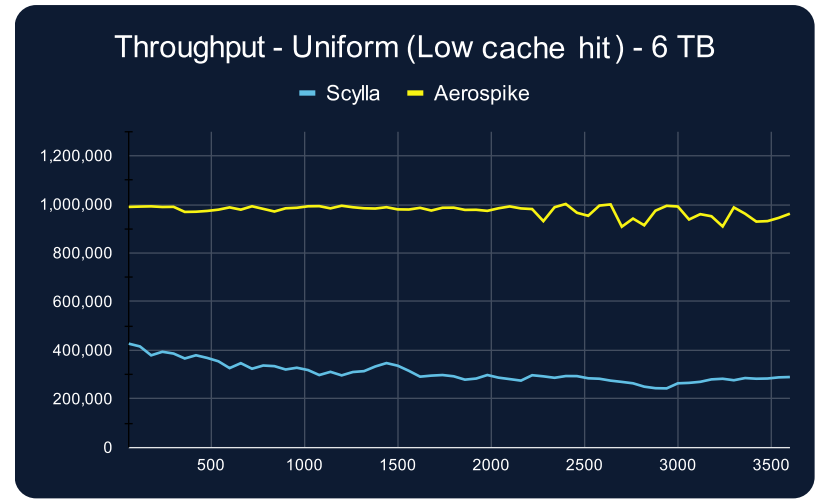
## Analysis

ScyllaDB requires roughly 50% more resources than Aerospike to store the same data at the same level of availability. When combined with Aerospike's ability to run on lower-cost instance types, this reduces overall infrastructure spend for Aerospike deployments to around 60% of ScyllaDB's.

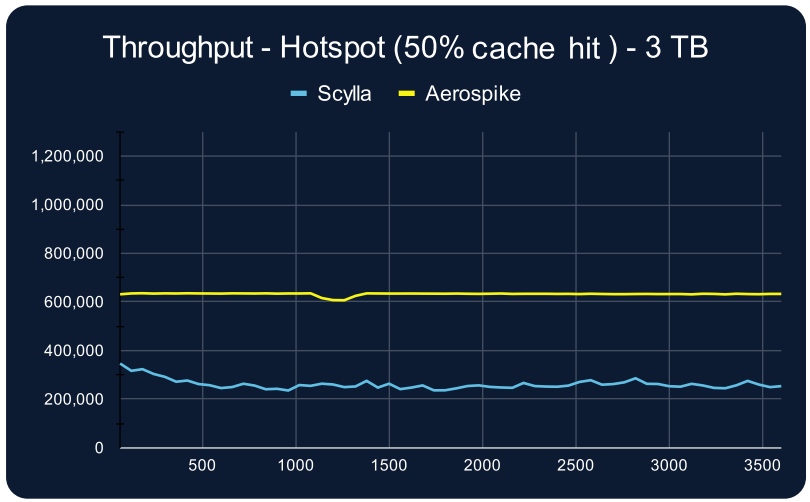
# Throughput (Higher numbers are better)



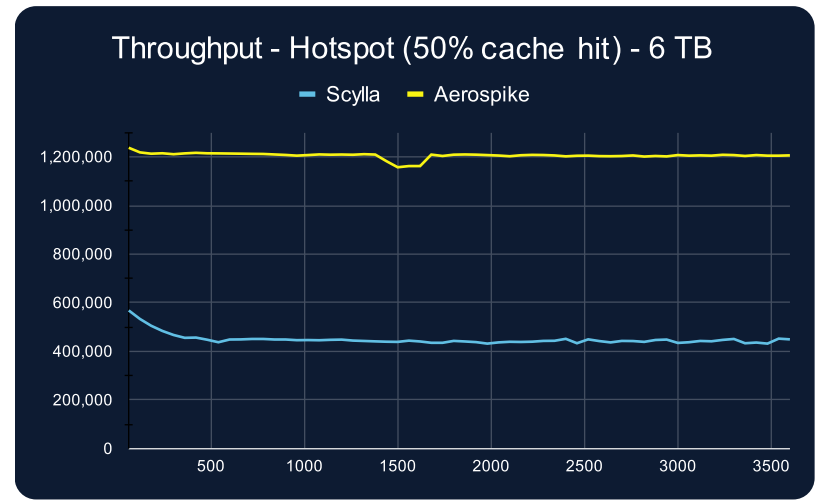

Average (Aerospike 488K ops/s, ScyllaDB, 197K ops/s)




Average (Aerospike 974K ops/s, ScyllaDB, 308K ops/s)




Average (Aerospike 633K ops/s, ScyllaDB, 262K ops/s)




Average (Aerospike 1,206K ops/s, ScyllaDB, 449K ops/s)

## 3 TB result

| | Uniform (Low cache hit) | | Hotspot (50% cache hit) | |
|---|---|---|---|---|
| | Scylla | Aerospike | Scylla | Aerospike |
| Average | 196,801 | 487,865 | 261,544 | 632,659 |
| Min | 185,348 | 475,275 | 236,040 | 607,238 |
| Max | 216,391 | 496,015 | 346,988 | 636,059 |
| Peak-to-trough variability (%) | 15.77% | 4.25% | 42.42% | 4.56% |
| Degradation | -0.03% | -0.01% | -0.25% | 0.00% |

## 6 TB result

| | Uniform (Low cache hit) | | Hotspot (50% cache hit) | |
|---|---|---|---|---|
| | Scylla | Aerospike | Scylla | Aerospike |
| Average | 308,614 | 973,824 | 449,102 | 1,205,757 |
| Min | 243,417 | 908,397 | 432,038 | 1,157,328 |
| Max | 427,366 | 1,001,909 | 568,546 | 1,237,507 |
| Peak-to-trough variability (%) | 59.60% | 9.60% | 30.40% | 6.65% |
| Degradation | -0.40% | -0.05% | -0.16% | -0.01% |

## Analysis

**Performance bounds:**

- Aerospike delivers 2.5–3× higher throughput than ScyllaDB.

**Predictability:**

- Aerospike delivers steady, consistent throughput, showing only 4.25% to 9.6% peak-to-trough variability.
- ScyllaDB's throughput fluctuates far more, showing 15.77% to 59.6% peak-to-trough variability.

**Efficiency:**

- Aerospike delivers higher and more consistent performance while using only two-thirds of the resources allocated to ScyllaDB.

**Scalability:**

- Aerospike exhibits near-perfect linear scalability, delivering a full 2× increase in throughput when hardware resources are doubled.
- ScyllaDB does not scale linearly, achieving only about a 1.6× increase under the same conditions.

**Degradation over time:**

- Aerospike's throughput remains essentially flat throughout the test, with less than a 0.05% difference between the first and last five minutes, indicating no performance degradation over time.
- ScyllaDB, by contrast, shows throughput degradation over time, with a 0.4% drop when comparing performance in the first and last five minutes.

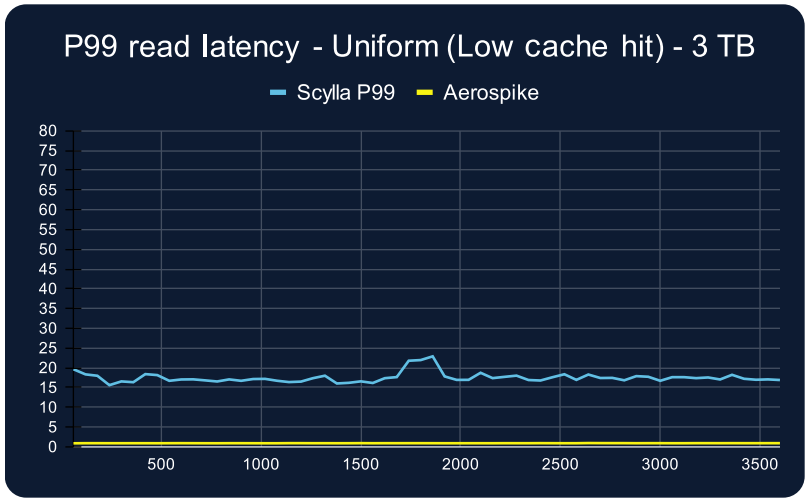
**Effect of caching:**

- Both systems achieve higher throughput when the workload lends itself to effective caching.
- Aerospike benefits less from caching, yet its performance without cache hits is still roughly 2× better than ScyllaDB's performance with a 50% cache hit rate. This clearly demonstrates that Aerospike's performance does not depend on the effectiveness of a caching layer.

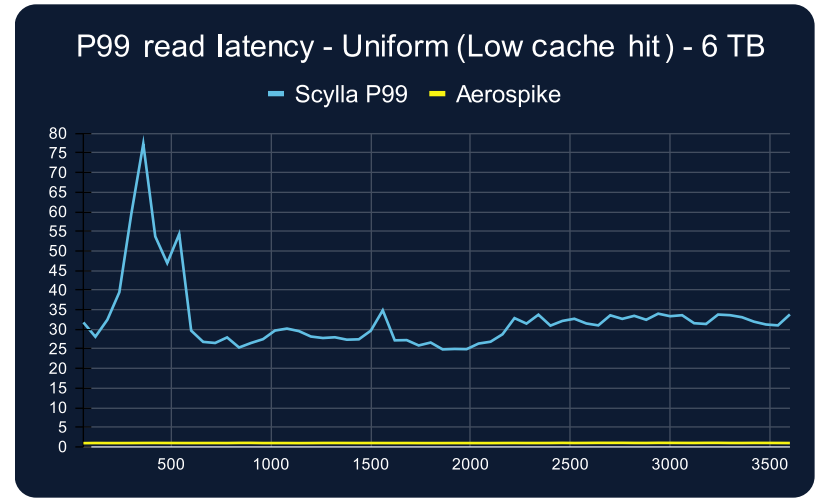
**Note:** In the 6 TB with Hotspot distribution test, we observed some throughput fluctuation, but this is not typical behaviour for Aerospike and was most likely caused by noisy-neighbour effects on the cloud infrastructure. We did not re-run this test, as ScyllaDB must have been equally susceptible to such conditions. We do not believe this variance has any material impact on the overall conclusions.

# Read latency (Lower numbers are better)

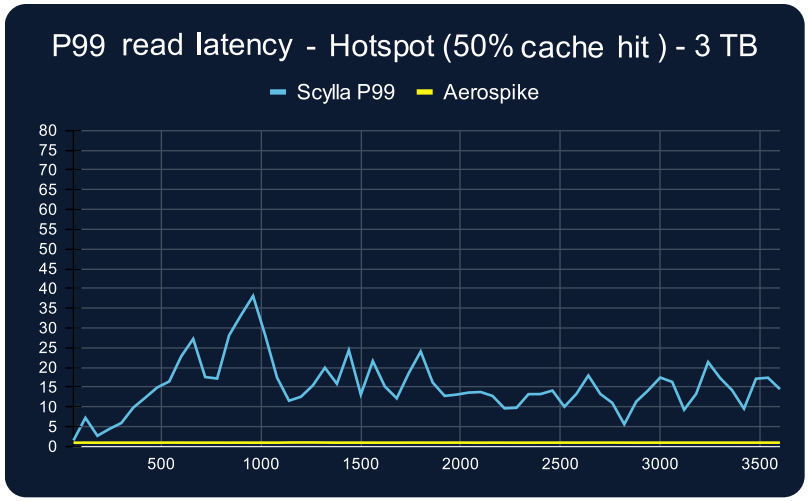
## P99 read latency graph



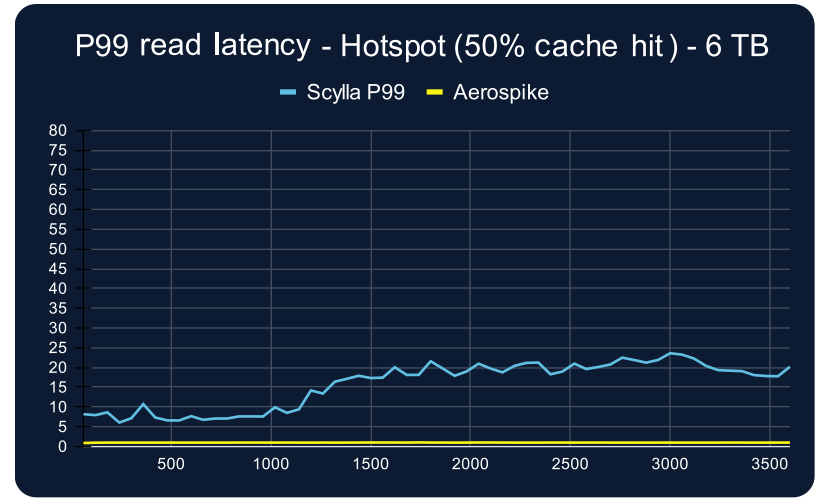

Average (Aerospike 0.92 ms, ScyllaDB 17.52 ms)




Average (Aerospike 0.98 ms, ScyllaDB 32.51 ms)

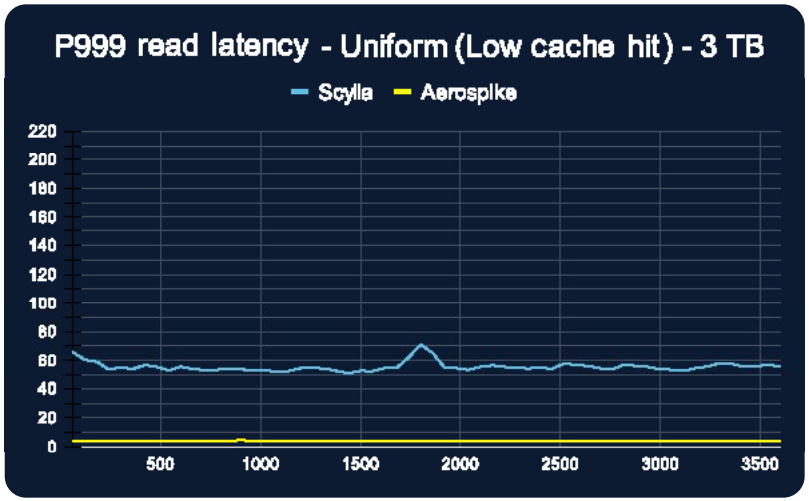



Average (Aerospike 0.98 ms, ScyllaDB 15.30 ms)

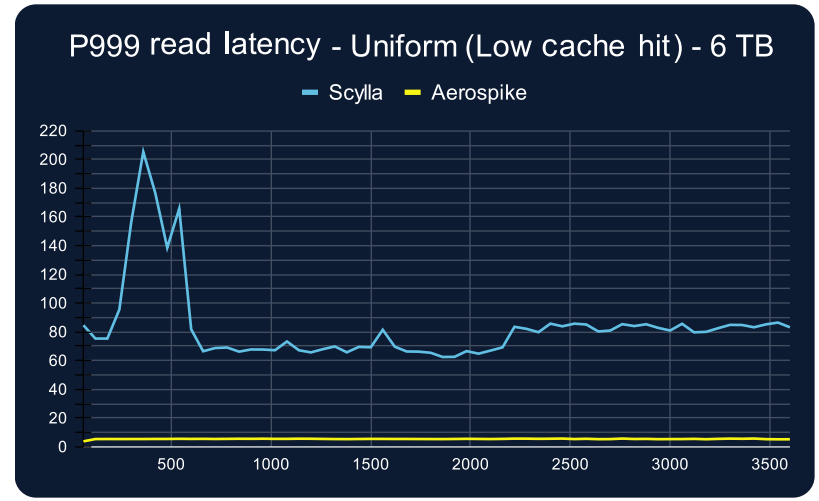



Average (Aerospike 0.99 ms, ScyllaDB 15.7 ms)

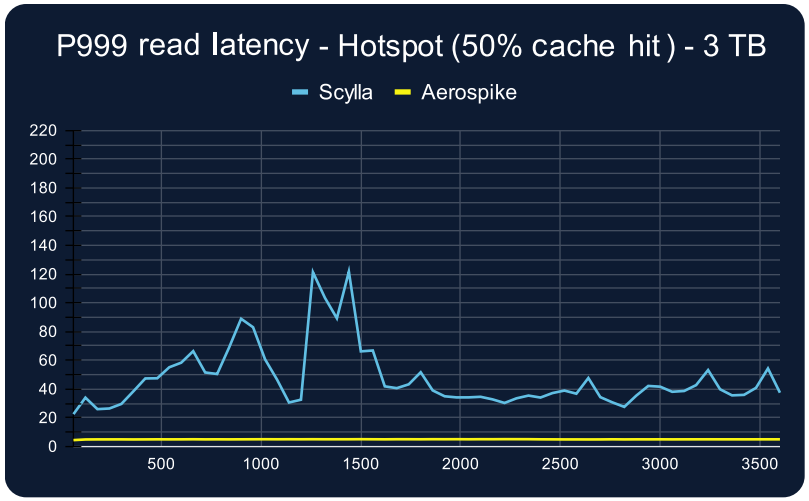
## P999 read latency graph



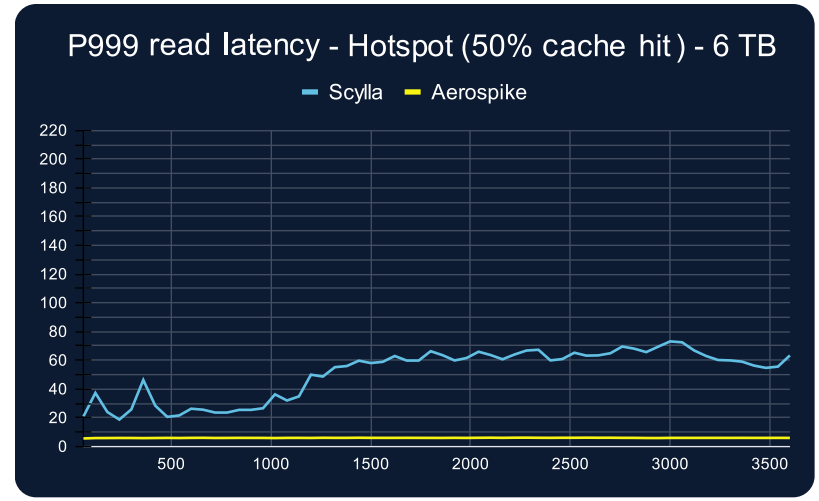

Average (Aerospike 4.42 ms, ScyllaDB 55.76 ms)




Average (Aerospike 5.49 ms, ScyllaDB 83.70 ms)




Average (Aerospike 5.0 ms, ScyllaDB 47.40 ms)




Average (Aerospike 6.0 ms, ScyllaDB 50.94 ms)

## P99 read latency

### 3 TB

| | Uniform (Low cache hit) | | Hotspot (50% cache hit) | |
|---|---|---|---|---|
| | Scylla | Aerospike | Scylla | Aerospike |
| Average (ms) | 17.53 | 0.92 | 15.30 | 0.98 |
| Min (ms) | 15.62 | 0.90 | 1.55 | 0.97 |
| Max (ms) | 22.90 | 0.96 | 38.10 | 1.02 |
| Peak-to-trough variability (ms) | 7.29 | 0.06 | 36.55 | 0.05 |
| Degradation (%) | 0.02% | -0.01% | -2.80% | -0.01% |

### 6 TB

| | Uniform (Low cache hit) | | Hotspot (50% cache hit) | |
|---|---|---|---|---|
| | Scylla | Aerospike | Scylla | Aerospike |
| Average (ms) | 32.51 | 0.98 | 15.70 | 0.99 |
| Min (ms) | 24.87 | 0.95 | 6.07 | 0.89 |
| Max (ms) | 77.35 | 1.03 | 23.61 | 1.04 |
| Peak-to-trough variability (ms) | 52.47 | 0.08 | 17.54 | 0.14 |
| Degradation (%) | 0.19% | -0.03% | -1.45% | -0.03% |

## P999 read latency

### 3 TB

| | Uniform (Low cache hit) | | Hotspot (50% cache hit) | |
|---|---|---|---|---|
| | Scylla | Aerospike | Scylla | Aerospike |
| Average (ms) | 55.76 | 4.42 | 47.40 | 5.01 |
| Min (ms) | 51.39 | 3.77 | 22.53 | 4.46 |
| Max (ms) | 71.01 | 4.62 | 121.74 | 5.12 |
| Peak-to-trough variability (ms) | 19.62 | 0.85 | 99.22 | 0.66 |
| Degradation (%) | 0.04% | -0.02% | -0.52% | -0.04% |

### 6 TB

| | Uniform (Low cache hit) | | Hotspot (50% cache hit) | |
|---|---|---|---|---|
| | Scylla | Aerospike | Scylla | Aerospike |
| Average (ms) | 83.70 | 5.49 | 50.94 | 6.01 |
| Min (ms) | 62.55 | 3.83 | 18.75 | 5.61 |
| Max (ms) | 205.30 | 5.78 | 73.17 | 6.16 |
| Peak-to-trough variability (ms) | 142.75 | 1.94 | 54.42 | 0.55 |
| Degradation (%) | 0.15% | -0.07% | -1.28% | -0.03% |

## Analysis

**Performance bounds:**

- Aerospike delivers sub-millisecond latency at P99 in all tests.
- ScyllaDB's latency is not tightly bound. The P99 latency fluctuates between 5 and 80 ms.

**Predictability:**

- Aerospike exhibits extremely low jitter, with P99 latency showing only 0.05–0.15 ms peak-to-trough variation across all tests.
- ScyllaDB's P99 latency fluctuates sharply, with peak-to-trough variation reaching up to 52.47 ms.

**Efficiency:**

- Aerospike delivers 16–33× better P99 read latency while using only one-third of the resources allocated to ScyllaDB.

**Scalability:**

- Aerospike's latency stays effectively unchanged when cluster size and load double.
- ScyllaDB fails to maintain linear latency scalability when cluster size and load double.

**Degradation over time:**

- Aerospike's latency remains flat throughout the entire test, with maximum degradation of less than 0.07%.
- ScyllaDB, by contrast, shows P99 latency degradation over time, with up to a 2.8% drop between the first and last five minutes.

**Effect of caching:**

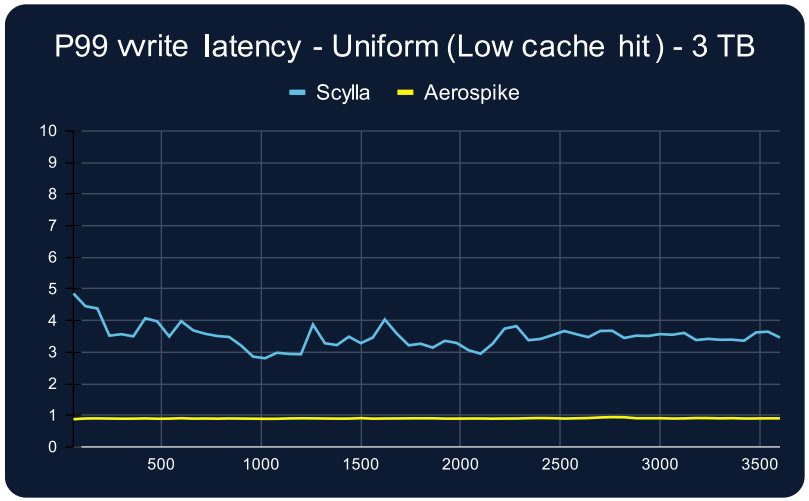
- Aerospike's sub-millisecond latency is independent of cache hit rate.
- ScyllaDB gains slight improvements from higher cache hit rates, but the gap between the two technologies remains large.

**Latency at extremes:**

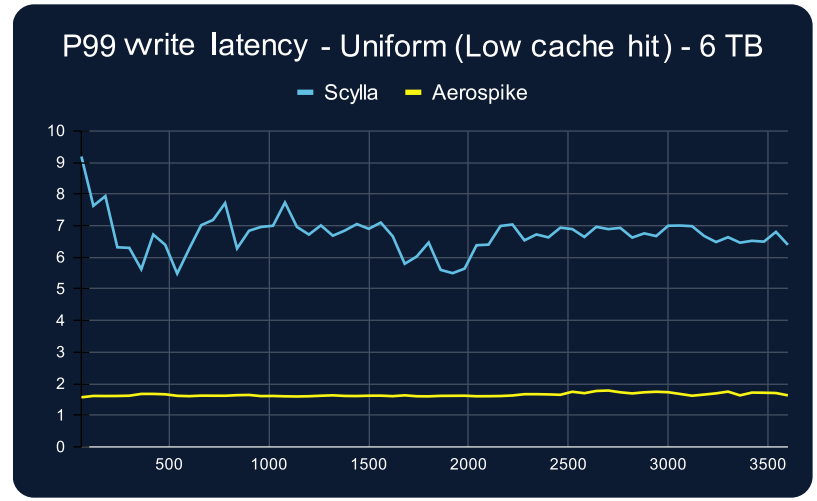
- Aerospike's P999 read latency increases only to about 6 ms, with minimal jitter. Its peak-to-trough variability remains below 1.94 ms.
- ScyllaDB's P999 latency averages 47–84 ms, with peak-to-trough variability reaching 142.75 ms.

# Write latency (Smaller numbers are better)

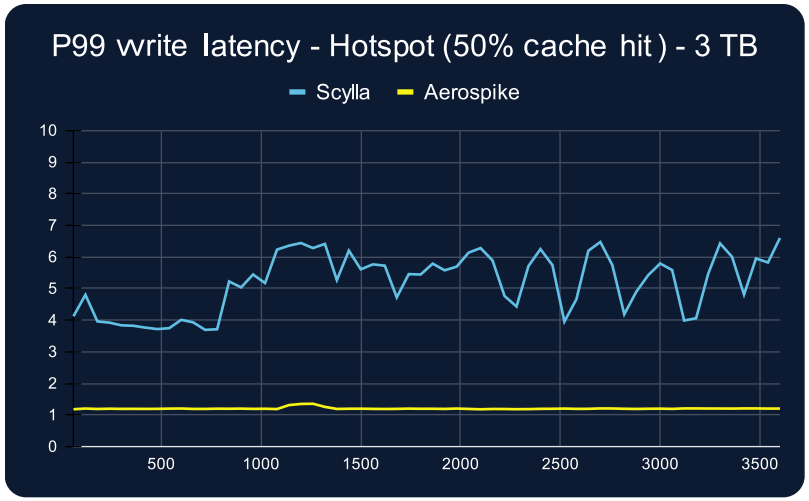
## P99 write latency



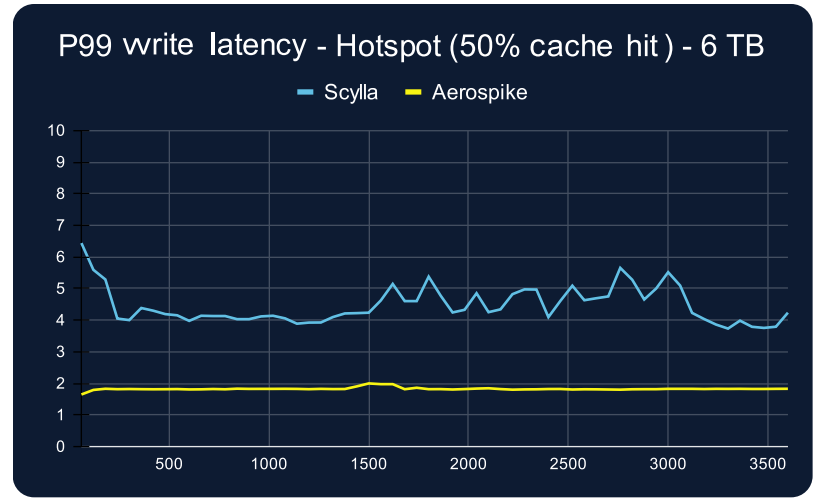

Average (Aerospike 0.9 ms, ScyllaDB 3.5 ms)




Average (Aerospike 1.65 ms, ScyllaDB 6.71 ms)

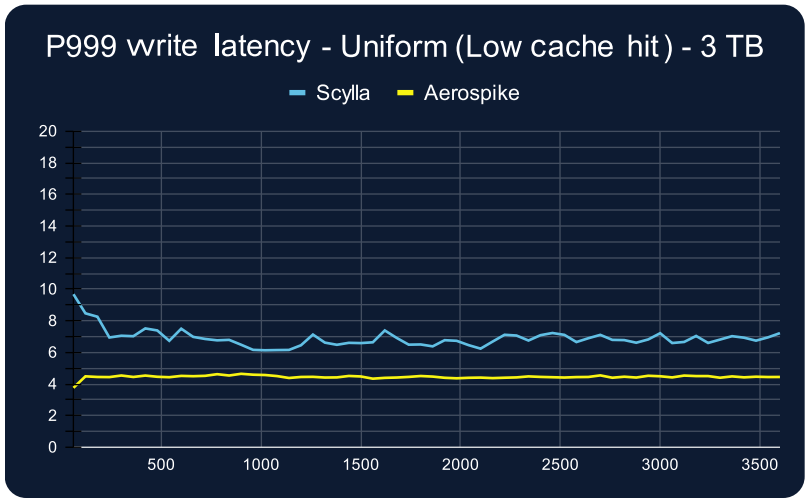



Average (Aerospike 1.2 ms, ScyllaDB 5.2 ms)

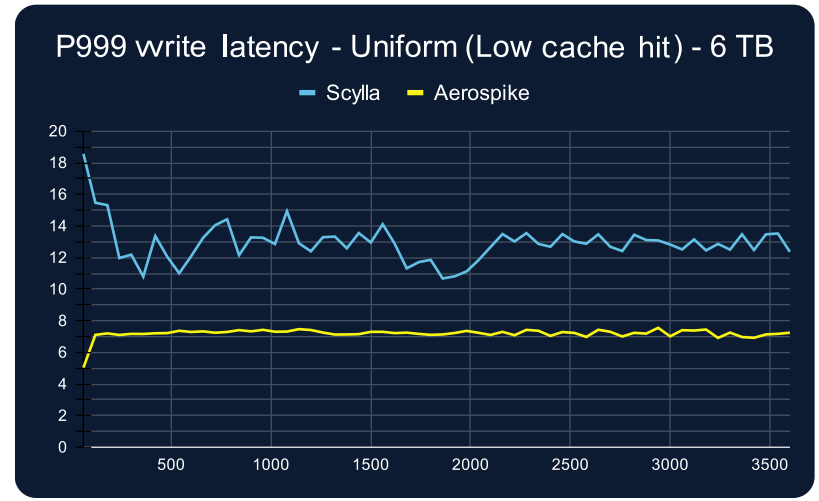



Average (Aerospike 1.82 ms, ScyllaDB 4.47 ms)

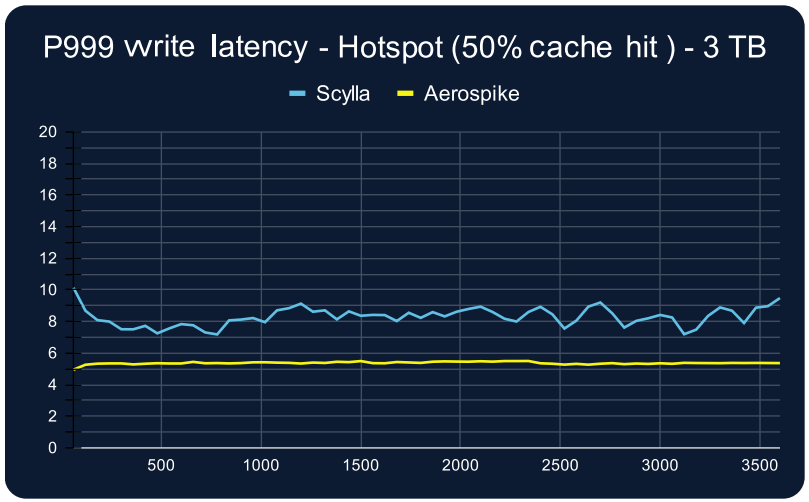
# P999 write latency



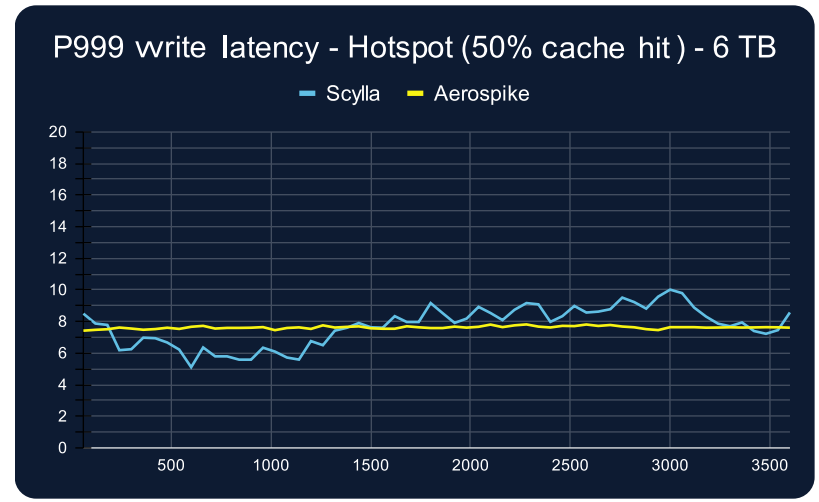

Average (Aerospike 4.45 ms, ScyllaDB 6.89 ms)




Average (Aerospike 7.19 ms, ScyllaDB 12.93 ms)




Average (Aerospike 5.37 ms, ScyllaDB 8.3 ms)




Average (Aerospike 7.62 ms, ScyllaDB 7.71 ms)

## P99 write latency

### 3 TB

| | Uniform (Low cache hit) | | Hotspot (50% cache hit) | |
|---|---|---|---|---|
| | Scylla | Aerospike | Scylla | Aerospike |
| Average (ms) | 3.50 | 0.90 | 5.20 | 1.20 |
| Min (ms) | 2.80 | 0.88 | 3.69 | 1.18 |
| Max (ms) | 4.85 | 0.94 | 6.60 | 1.35 |
| Peak-to-trough variability (ms) | 2.05 | 0.07 | 2.91 | 0.18 |
| Degradation (%) | 0.19% | -0.01% | -0.38% | -0.01% |

### 6 TB

| | Uniform (Low cache hit) | | Hotspot (50% cache hit) | |
|---|---|---|---|---|
| | Scylla | Aerospike | Scylla | Aerospike |
| Average (ms) | 6.71 | 1.65 | 4.47 | 1.82 |
| Min (ms) | 5.48 | 1.57 | 3.73 | 1.65 |
| Max (ms) | 9.19 | 1.78 | 6.44 | 2.00 |
| Peak-to-trough variability (ms) | 3.71 | 0.21 | 2.71 | 0.35 |
| Degradation (%) | 0.14% | -0.05% | 0.30% | -0.03% |

## P999 write latency

### 3 TB

| | Uniform (Low cache hit) | | Hotspot (50% cache hit) | |
|---|---|---|---|---|
| | Scylla | Aerospike | Scylla | Aerospike |
| Average (ms) | 6.90 | 4.45 | 8.30 | 5.37 |
| Min (ms) | 6.13 | 3.76 | 7.18 | 4.94 |
| Max (ms) | 9.68 | 4.64 | 10.13 | 5.50 |
| Peak-to-trough variability (ms) | 3.55 | 0.89 | 2.96 | 0.55 |
| Degradation (%) | 0.16% | -0.03% | 0.01% | -0.03% |

### 6 TB

| | Uniform (Low cache hit) | | Hotspot (50% cache hit) | |
|---|---|---|---|---|
| | Scylla | Aerospike | Scylla | Aerospike |
| Average (ms) | 12.93 | 7.19 | 7.70 | 7.62 |
| Min (ms) | 10.67 | 5.03 | 5.10 | 7.41 |
| Max (ms) | 18.56 | 7.55 | 10.01 | 7.81 |
| Peak-to-trough variability (ms) | 7.89 | 2.52 | 4.91 | 0.40 |
| Degradation (%) | 0.13% | -0.05% | -0.05% | -0.01% |

## Analysis

**Performance bounds:**

- Aerospike's write latency remains roughly 2.4× to 4.3× better than ScyllaDB's across all tests.
- Aerospike's write latency also closely mirrors its read-latency profile, staying consistently under 2 ms at P99, which highlights its ability to handle mixed workloads efficiently.
- ScyllaDB's write performance is significantly better than its read performance, suggesting the system is more naturally suited to write-heavy workloads. Even so, it remains several milliseconds slower than Aerospike at P99.

**Predictability:**

- Aerospike exhibits extremely low jitter, with maximum peak-to-trough variability of only 0.35 ms at P99.
- ScyllaDB's write latency, while more stable than its read latency, still fluctuates noticeably, with peak-to-trough variability reaching 3.71 ms at P99.

**Efficiency:**

- Aerospike delivers 2.4× to 4.3× better P99 write latency while using only one-third of the resources allocated to ScyllaDB.

**Scalability:**

- Both systems show sub-linear scalability for write latency, but Aerospike is less impacted.

**Degradation over time:**

- Aerospike's latency remains flat throughout the entire test, with maximum degradation of less than 0.05%.
- ScyllaDB, by contrast, shows P99 latency degradation over time, with up to a 0.38% drop between the first and last five minutes.

**Effect of caching:**

- Caching cannot meaningfully affect write latency, so any variation between tests should be attributed to other factors.

**Latency at extremes:**

- Aerospike's write latency at P999 increases more noticeably than its read latency, which is expected since the write path involves additional network hops, making it more susceptible to tail-latency variation.
- ScyllaDB's extreme write latency is more contained than its extreme read latency.
- Aerospike remains roughly 50% faster at P999 in all but one test.
- In the 6 TB / 50% cache-hit test, ScyllaDB shows unusually strong P999 results matching Aerospike's performance. We do not know the reason, but such isolated anomalies are common in benchmarking and do not alter the overall trend.

# Benchmarking methodology

## Objectives

The objectives of the test are to measure and compare both technologies across the following dimensions:

- Hardware efficiency
- Throughput capacity
- Latency behaviours
- Scalability
- Performance stability and dependability

## Test partner - McKnight Consulting Group

These benchmarks were conducted in partnership with McKnight Consulting Group, whose extensive background in evaluating database technologies informed the execution of the tests. Aerospike designed the test methodology and authored this report; McKnight Consulting Group carried out the benchmark executions.

## Test tool

These benchmarks were conducted using the YCSB (Yahoo! Cloud Serving Benchmark) framework, a widely used open-source tool for assessing the performance of modern cloud and NoSQL data stores.

## Control variables

Control variables are the elements held constant to ensure that observed differences stem from the variable being tested. In database benchmarking, workload pressure and hardware are often chosen as control variables to enable a fair comparison between technologies. However, this approach can introduce bias: the selected hardware may suit one system's architecture better than the other, unintentionally giving it an advantage.

To remove this source of bias, we used cluster capacity, not hardware size, as a control variable. Each system was given a cluster sized to handle the target dataset with sufficient overhead. This lets each technology operate on hardware aligned with its design, enabling a fairer and more realistic comparison than forcing both onto a single hardware configuration.

## Dataset description

### Data schema

All records were stored as a single dataset within one keyspace/namespace.

The size of each record was set to be 2,000 bytes. Each record consists of 10 columns/bins, with each column sized at 200 bytes. The following configuration was used in YCSB to achieve this layout:

| YCSB parameter | Value |
|---|---|
| fieldcount | 10 |
| fieldlength | 200 |
| fieldlengthdistribution | constant |

## Data size

We ran the benchmark twice: once with each technology provisioned to store 1.5 billion records (3 TB of data), and again with 3 billion records (6 TB of data), ensuring both datasets could be handled comfortably in each configuration.

# Systems under test description

We used the latest available enterprise releases of both databases at the time of testing:

- Aerospike Enterprise Edition 8.1.0.1
- ScyllaDB Enterprise 2025.2.0

| | Record size | Record count | Data size |
|---|---|---|---|
| Benchmark 1 | 2,000 B | 1,500,000,000 | ~ 3 TB |
| Benchmark 2 | 2,000 B | 3,000,000,000 | ~ 6 TB |

## Server configuration

We used the default, out-of-the-box settings for both systems. While each technology offers tuning options that can optimize performance for specific workloads, as the workloads used in the test are fairly generic, the default configuration should provide a fair and representative baseline for comparison.

For ScyllaDB, we formatted the disks using the XFS file system, created a single keyspace with SimpleStrategy, and kept the default compaction settings.

For Aerospike, which can operate directly on raw block devices, we partitioned and zeroed the disks but did not format them with any file system.

## Client configuration

We used the default client configurations for both technologies.

ScyllaDB was configured with `LOCAL_QUORUM` for both reads and writes. Aerospike, by contrast, does not use a quorum model; its default is `COMMIT_ALL` for writes and ONE for reads, meaning a write is only acknowledged after it has been committed to all replicas defined by the replication factor, and the primary replica can then serve consistent reads.

In practice, this means that for writes, both ScyllaDB and Aerospike acknowledge the operation only after two nodes have committed it (RF=3 with `LOCAL_QUORUM` for ScyllaDB, RF=2 with COMMIT_ALL for Aerospike). For reads, however, ScyllaDB must contact two nodes to satisfy `LOCAL_QUORUM`, whereas Aerospike can read from a single primary.

**Note:** It might appear that ScyllaDB could be configured in a similar way (for example, RF=2 with ALL for writes and ONE for reads), but this would make the cluster unavailable after any single-node failure, as ALL could no longer be satisfied. Aerospike is architected differently: with RF=2 and its default policies, it offers consistency broadly comparable to `LOCAL_QUORUM` reads and writes in ScyllaDB, while remaining available for writes in the face of node failures.

## Selecting the optimal hardware configuration

We allocated the recommended amount of resources required to store the target dataset sizes of 3 and 6 TB, such that each database stored approximately 70% of its optimal total capacity, ensuring that neither technology was pushed into near-capacity behavior, which could otherwise skew the comparison.

### Replication factor

When sizing clusters for a target data capacity, we must determine the appropriate replication factor (RF) for each technology. This is not straightforward because Aerospike recommends RF=2, while ScyllaDB recommends RF=3 for achieving high availability and consistency.

We evaluated three approaches to align the replication factors:

1. Run Aerospike with RF=3:Aerospike fully supports RF=3, but this option presents two issues. First, it diminishes one of Aerospike's key advantages: achieving the same level of availability and consistency with only two replicas, rather than the three typically required by quorum-based databases. Second, Aerospike with RF=3 would actually provide higher availability than ScyllaDB with RF=3, since Aerospike can remain available even after a second node failure.
2. Run ScyllaDB with RF=2: This is not viable. ScyllaDB does not recommend RF=2 for workloads that require high availability and consistency, and it is rarely used in production for those workloads.
3. Use each system's recommended RF (Aerospike RF=2, ScyllaDB RF=3): Because ScyllaDB must maintain three copies of the data while Aerospike maintains two, and because we chose capacity as the control variable, ScyllaDB must be allocated 50% more hardware resources (CPU, RAM, storage, and network) to achieve the same usable capacity as Aerospike. Having extra resources is an advantage for ScyllaDB, but it also needs to do slightly more work than Aerospike on write requests.

Neither of the first two options is ideal, but the third provides the fairest and most realistic comparison. We therefore used RF=2 for Aerospike and RF=3 for ScyllaDB.

### Cloud provider

All benchmarks, including the database nodes and the YCSB clients, were run within a single availability zone in the AWS US West (Oregon) region.

### Instance types

As outlined in the Control Variables section, we aimed to avoid hardware bias by selecting the most appropriate instance types for each technology. Aerospike recommends ARM-based Graviton instances on AWS, as they offer comparable performance to Intel-based instances while being 10–15% cheaper.

At the time of this benchmark, we could not confirm whether ScyllaDB fully supported ARM-based processors, so we avoided using them. If ScyllaDB does support them, it could also benefit from the additional savings.

The instances used in the benchmarks were:

| | Instance name | CPU type | vCPU | Memory | Disk | Hourly rate |
|---|---|---|---|---|---|---|
| Aerospike | i4g.4xlarge | Arm | 16 | 128 GB | 3,750 GB | $1.235 |
| ScyllaDB | i4i.4xlarge | Intel x86-64 | 16 | 128 GB | 3,750 GB | $1.373 |

As shown, these two instance types are nearly identical, differing only in CPU architecture and price. Aerospike performs the same on both, but because i4g instances are cheaper, they are typically the preferred option.

## Sizing

To store 3 and 6 TB data sets, we used the following number of nodes for each technology:

| | | Instance type | Node count |
|---|---|---|---|
| 3 TB | Aerospike | i4g.4xlarge | 4 |
| | ScyllaDB | i4i.4xlarge | 6 |
| 6 TB | Aerospike | i4g.4xlarge | 8 |
| | ScyllaDB | i4i.4xlarge | 12 |

## Preparation stage

For each test, we began with a load phase to populate the database with the target number of records. Once the dataset was fully loaded, we waited for all compaction/defragmentation processes to complete.

Then we executed the benchmarking phase but withheld metric collection for the first hour to ensure that all caches were fully warmed.

## Measurements

To meet the benchmark objectives, once the one-hour warm-up period had completed, we measured the following metrics for each run over a 60-minute window, recording values at 60-second intervals. Neither of the first two options is ideal, but the third provides the fairest and most realistic comparison. We therefore used RF=2 for Aerospike and RF=3 for ScyllaDB.

### Latency

The client-observed latency was recorded at P50, P90, P99, and P999.

## Throughput

When measuring the throughput capacity of a system, it is important to identify the highest throughput it can sustain without causing a significant increase in latency. This typically requires knowing in advance what "good" performance looks like. For Aerospike, we knew that. For ScyllaDB, however, we needed to run multiple tests to determine the point at which increasing workload pressure no longer produced higher throughput and instead caused latency to rise as operations began to queue.

To identify the throughput level ScyllaDB could sustain, we executed multiple benchmark runs against the 3 TB cluster, gradually increasing the load until latency degradation became excessive. You can find the full details of these runs in Appendix IV.

For the 3 TB test, we used 840 threads for ScyllaDB and 256 threads for Aerospike. For the 6 TB tests, we doubled the thread counts to validate each system's linear scalability claims. However, we do not treat thread count as a meaningful performance metric. Because YCSB uses blocking APIs, the number of threads required to reach a given request rate is simply a function of each system's average response time.

To illustrate this, consider two systems: one with an average response time of 1 ms and another with 10 ms. With a single thread, the first system can achieve 1,000 TPS, while the second would need 10 threads to reach the same rate. This difference has nothing to do with their maximum throughput capacity; the first might cap out at 1,000 TPS, while the second could scale to 10,000 TPS.

# Workload description

We selected a 70% read / 30% write workload. Although this results in more reads than writes, it is by no means considered a read-heavy workload. We chose this ratio to apply slightly more pressure on reads, as in most real-time use cases, read latency has a greater impact on the user's experience of application performance. At the same time, we avoided making the workload predominantly read-heavy, as we did not want the results to be skewed towards a single access pattern.

## Write workload

Since we wanted to keep the total number of records constant, the write portion of the workload was implemented as record replacements rather than inserts. In both Aerospike and Scylla, a replace operation follows the same execution path as a standard write. However, it also introduces data fragmentation, which increases storage overhead and forces the database to work harder over time.

Because fragmentation naturally occurs in real-world deployments, we believe incorporating it into the benchmark makes the results more realistic and relevant, rather than artificially idealized.

## Read workload

ScyllaDB promotes its internal cache as a key mechanism for delivering high performance. As Aerospike does not rely on caching to deliver the best performance, we tested both technologies with read workloads with two different distributions:

- **Uniform:** Requests are evenly distributed across the entire dataset. Because access probability is uniform, cache hit rates are expected to be very low.

- **Hotspot:** A large portion of reads target a small subset of records, increasing the likelihood of cache hits.

In the hotspot tests, our goal was to achieve a high cache hit rate for ScyllaDB. However, this proved more challenging than expected. To achieve even a 50% cache hit rate, we had to direct 55% of read requests to just 1% of the dataset, a proportion that already seems unlikely in most practical scenarios.

## YCSB parameters

Below are the YCSB parameters used:

| YCSB parameter | Hotspot | Uniform |
|---|---|---|
| readproportion | 0.7 | 0.7 |
| updateproportion | 0.3 | 0.3 |
| scanproportion | 0 | 0 |
| insertproportion | 0 | 0 |
| requestdistribution | hotspot | uniform |
| hotspotdatafraction | 0.01 | |
| hotspotopnfraction | 0.55 | |

# Conclusion

This benchmark set out to provide a practical, data-driven comparison of Aerospike and ScyllaDB under realistic operating conditions. Both systems were evaluated at meaningful scale (3 TB and 6 TB datasets), under a mixed 70/30 read–write workload, using each platform's recommended replication and deployment model, and across access patterns that both minimise and favour caching.

Across all test scenarios, Aerospike demonstrated materially stronger performance characteristics than ScyllaDB. It consistently delivered 2.5–3× higher sustained throughput, sub-millisecond P99 read latency, and low, tightly bounded tail latency, even as dataset size and workload intensity doubled. These results held regardless of cache effectiveness, highlighting Aerospike's ability to deliver predictable performance directly from persistent storage rather than relying on cache hit rates.

ScyllaDB, by contrast, showed a much heavier dependence on caching to improve performance, particularly for read latency. While its write latency was notably better than its read latency, it remained several milliseconds slower than Aerospike at P99, and exhibited significantly higher variability and longer tail latencies. As dataset size increased, ScyllaDB did not preserve latency or throughput linearly, indicating less efficient scaling behaviour under the tested conditions.

Importantly, these performance outcomes were achieved by Aerospike while using approximately one-third fewer infrastructure resources. Even though ScyllaDB was provisioned with additional nodes to accommodate its higher replication factor, Aerospike still outperformed it across throughput, latency, predictability, and stability over time. This combination of higher performance and lower resource consumption translates directly into improved cost efficiency and operational simplicity at scale.

No benchmark can eliminate all sources of scepticism, and alternative configurations or workload profiles may yield different results. However, for latency-sensitive, high-throughput workloads operating at multi-terabyte scale, particularly those where predictable tail latency and efficient scaling are critical, the evidence from this study strongly supports Aerospike as the more capable and efficient platform.

Organizations evaluating these technologies are encouraged to validate these findings against their own workloads by testing their applications on Aerospike at console.aerospike.com.

Aerospike is also available as a self-managed solution: you can download the Community Edition to try a subset of features, or request a free 60-day Enterprise licence to evaluate the full capabilities of the platform.

# Appendix I: Context on a 2023 ScyllaDB benchmark referencing Aerospike

In 2023, ScyllaDB published a benchmark that compared its performance results against figures drawn from a previously published Aerospike benchmark. In that publication, ScyllaDB reported achieving higher throughput than the Aerospike configuration referenced, while also acknowledging higher P99 write latency relative to Aerospike.

While the numerical comparison itself was accurate with respect to the cited results, the broader context of the Aerospike configuration used as a reference is important for correctly interpreting the comparison.

The Aerospike benchmark results referenced by ScyllaDB were obtained using Aerospike All-Flash mode, a specific deployment configuration for storing all records and indices entirely on persistent storage, with no use of system memory. This configuration is designed to support extremely large datasets (including petabyte-scale deployments) at lower cost per terabyte, and prioritizes storage density and durability over maximum throughput.

As a result, All-Flash mode delivers lower throughput and higher latency than Aerospike's standard deployment model, which uses memory strategically (not as a cache) to accelerate access while still maintaining persistence on flash storage.

Because All-Flash mode represents the most storage-constrained configuration of Aerospike, it is not representative of Aerospike's typical performance envelope for latency-sensitive or high-throughput workloads. Therefore, comparisons against this configuration reflect a valid but narrow operating point rather than Aerospike's general performance characteristics.

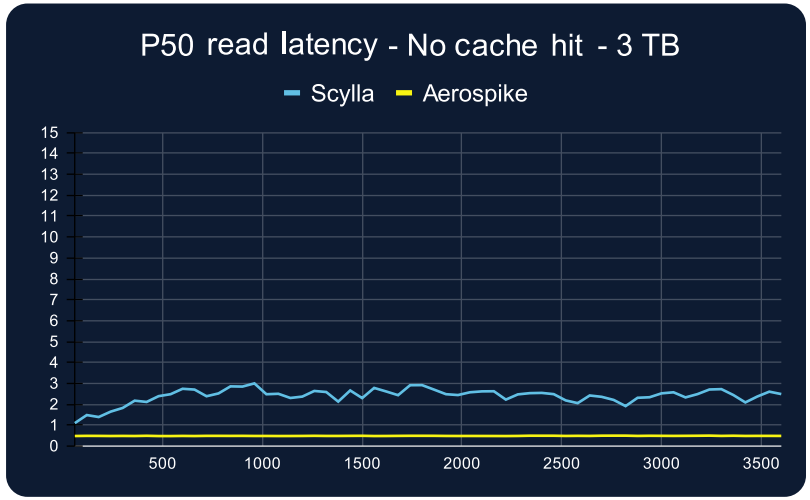
Even under these constrained conditions, the referenced Aerospike benchmark still demonstrated substantially lower tail latency than ScyllaDB, while delivering a majority of the reported throughput.

## Scope of this benchmark

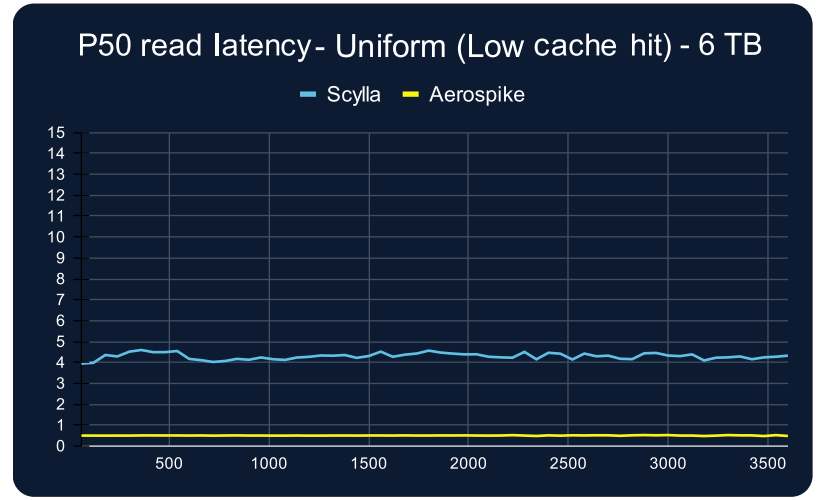
The benchmark presented in this report evaluates both Aerospike and ScyllaDB using configurations aligned with their recommended production deployment models for latency-sensitive workloads at multi-terabyte scale. Aerospike was tested using its standard architecture, which reflects how it is commonly deployed in performance-critical environments.

Therefore, the results in this study provide a more representative comparison of how both systems behave under realistic operating conditions, across mixed read/write workloads, varying cache effectiveness, and increasing scale.

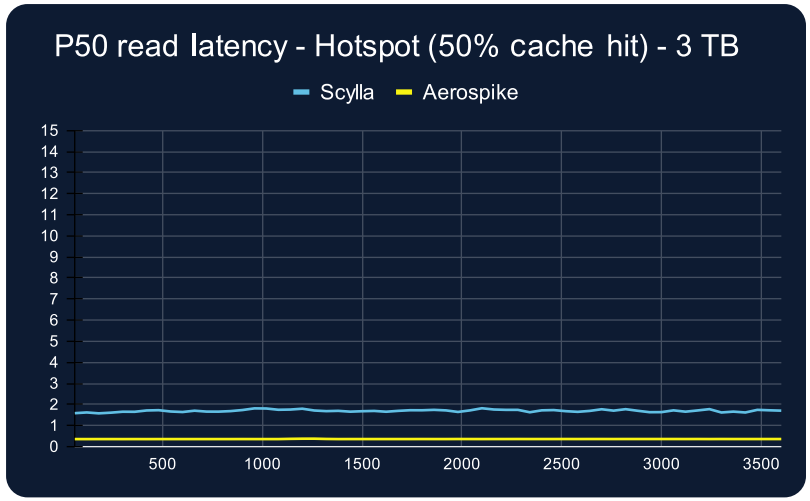
# Appendix II: Read latency (P50, P90)



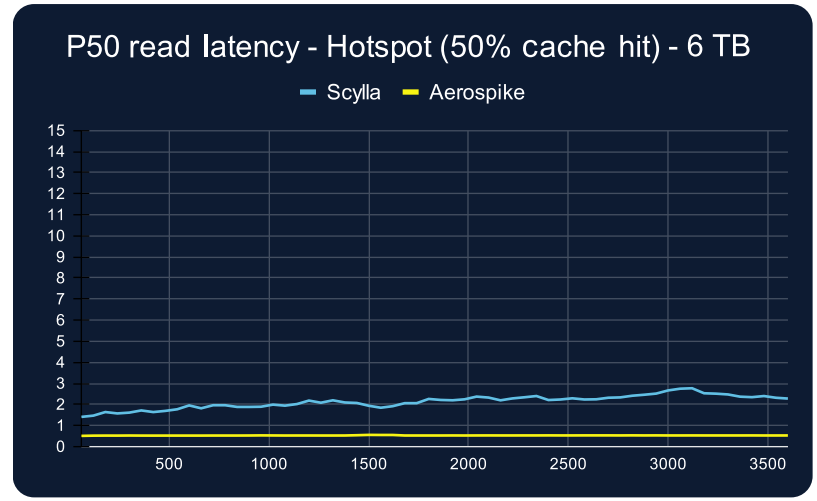

Average (Aerospike 0.48 ms, ScyllaDB 2.4 ms)



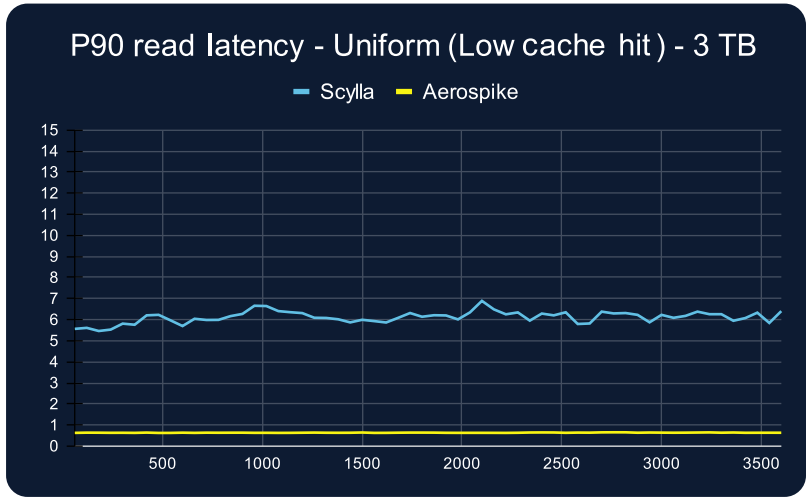

Average (Aerospike 0.50 ms, ScyllaDB 4.30 ms)



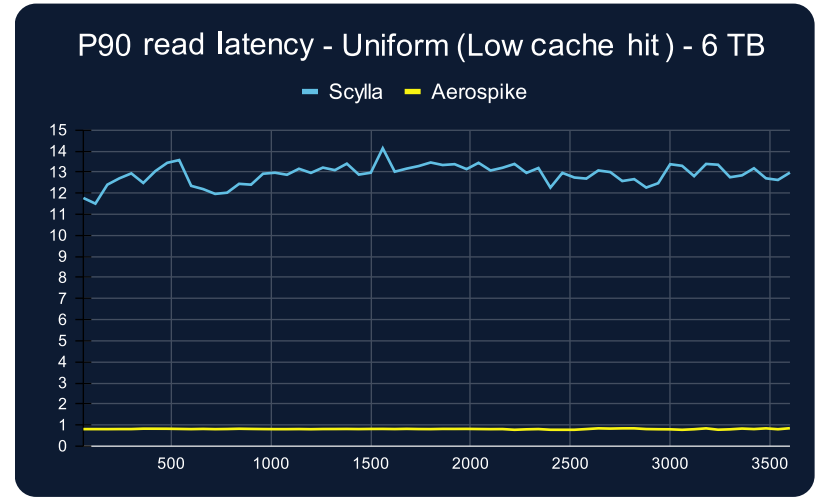

Average (Aerospike 0.35 ms, ScyllaDB 1.70 ms)



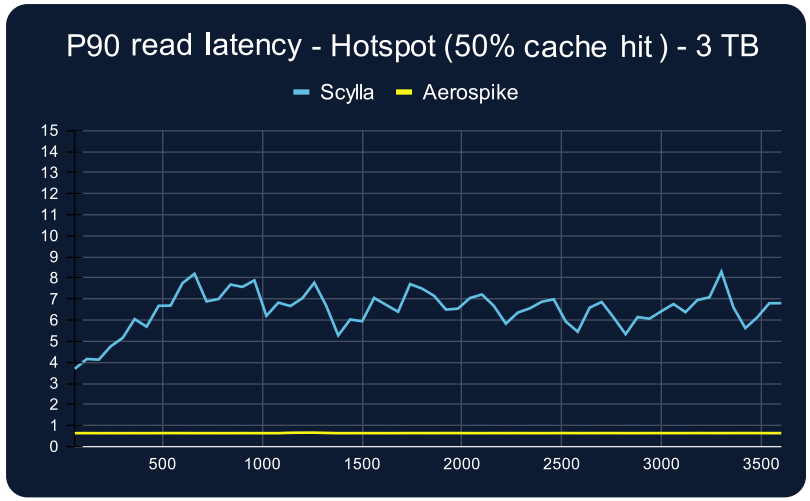

Average (Aerospike 0.53 ms, ScyllaDB 2.13 ms)



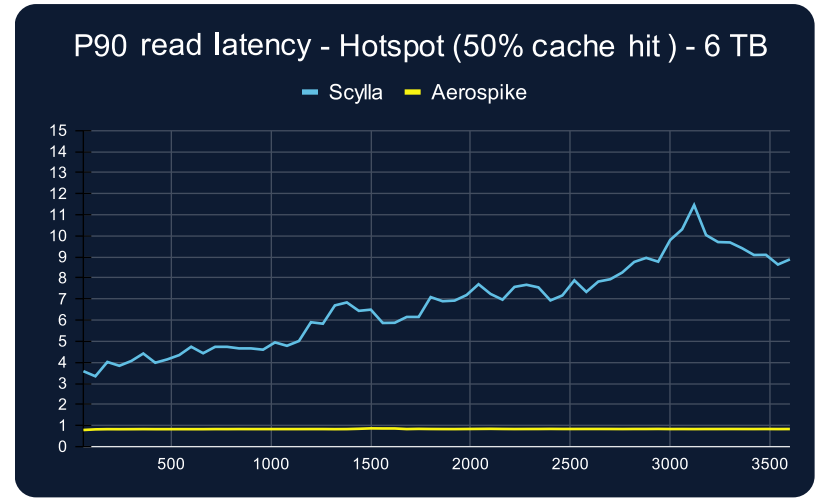

Average (Aerospike 0.64 ms, ScyllaDB 6.12 ms)




Average (Aerospike 0.81 ms, ScyllaDB 12.90 ms)

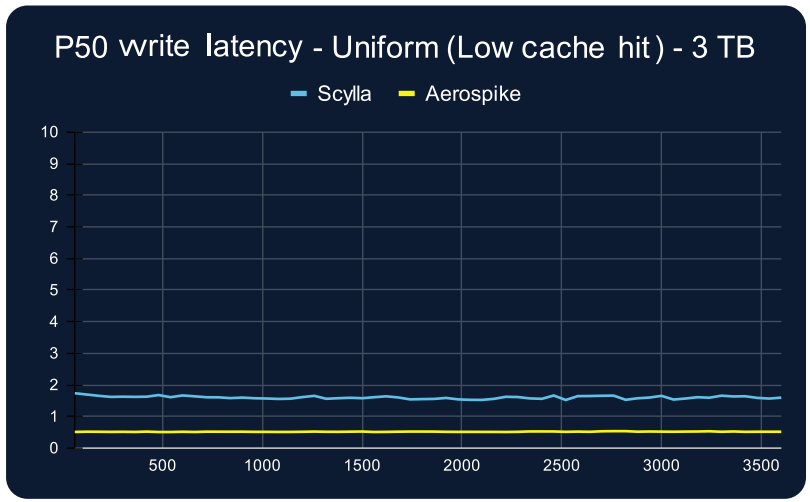



Average (Aerospike 0.64 ms, ScyllaDB 6.5 ms)

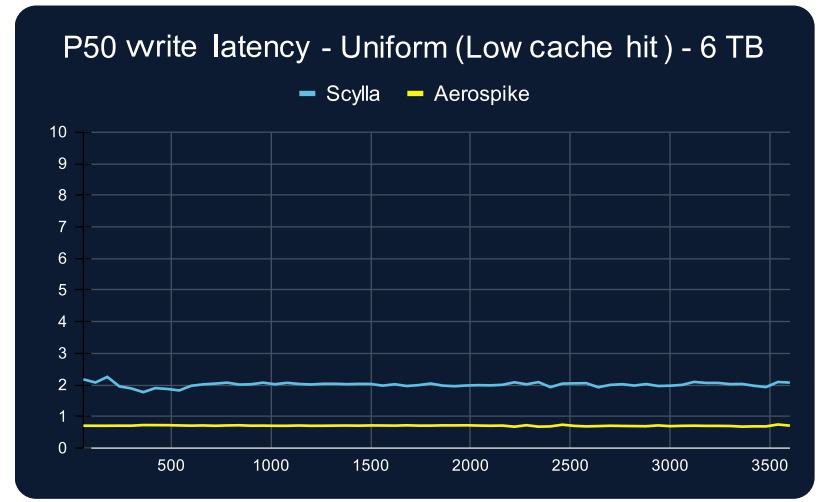



Average (Aerospike 0.83 ms, ScyllaDB 6.73 ms)

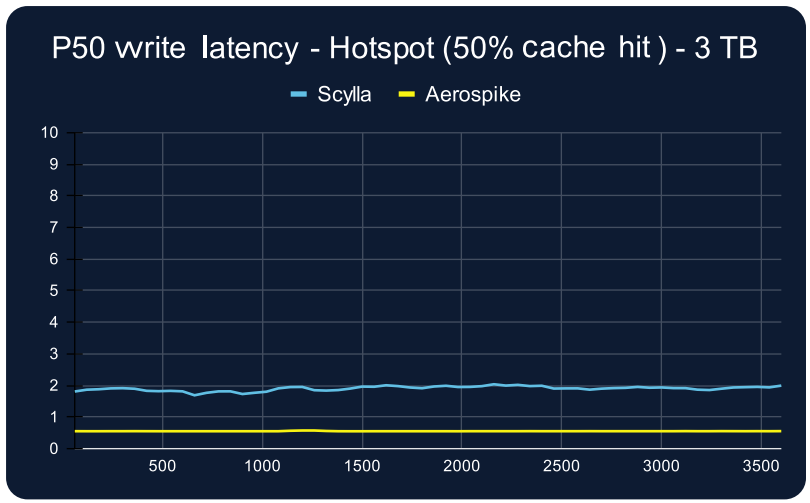
# Appendix III: Write latency (P50, P90)



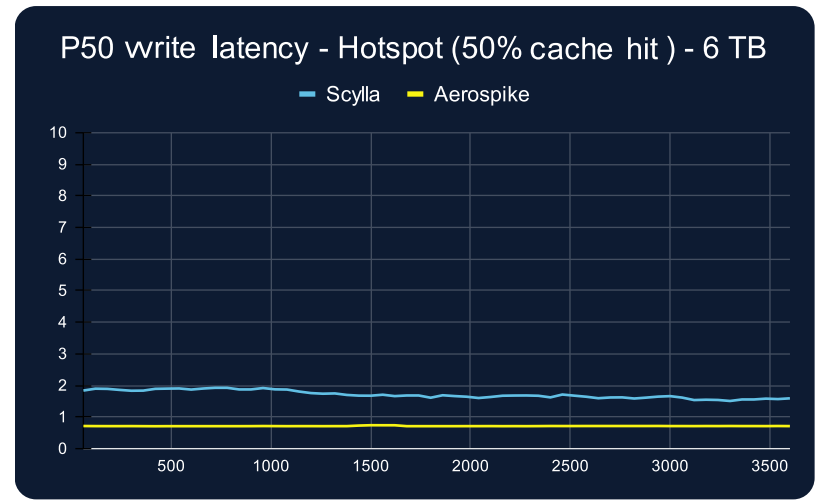

Average (Aerospike 0.52 ms, ScyllaDB 1.6 ms)



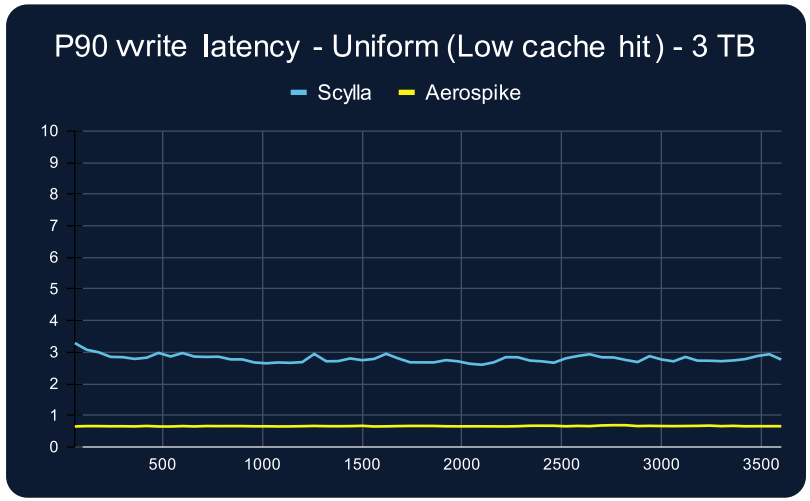

Average (Aerospike 0.70 ms, ScyllaDB 2.00 ms)



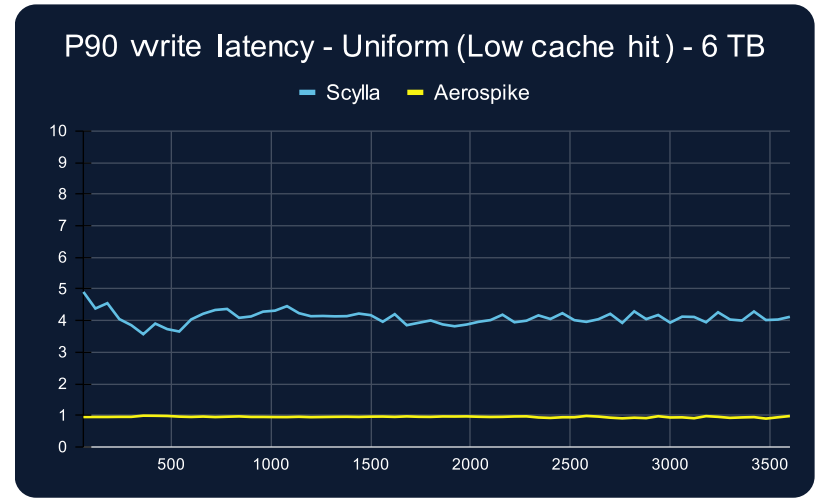

Average (Aerospike 0.55 ms, ScyllaDB 1.9 ms)



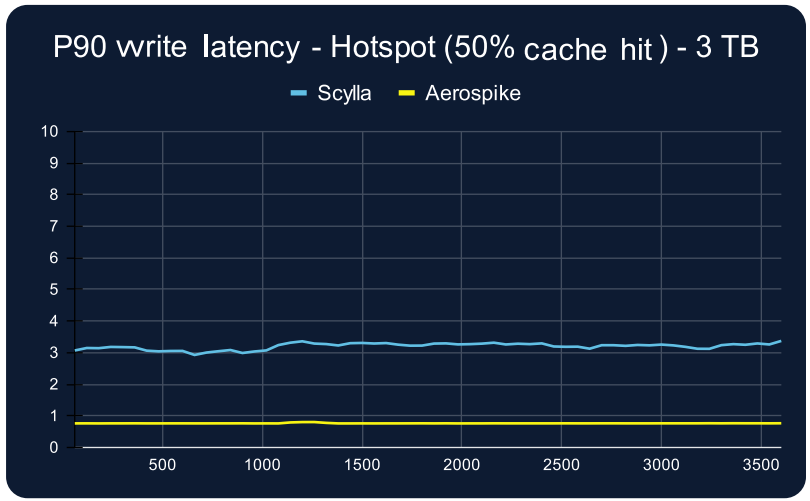

Average (Aerospike 0.71 ms, ScyllaDB 1.71 ms)



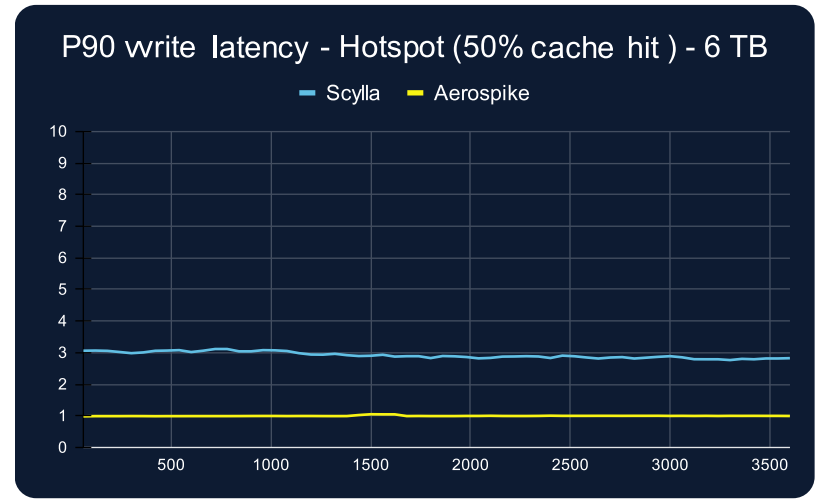

Average (Aerospike 0.66 ms, ScyllaDB 2.79 ms)




Average (Aerospike 0.94 ms, ScyllaDB 4.09 ms)




Average (Aerospike 0.76 ms, ScyllaDB 3.2 ms)




Average (Aerospike 0.99 ms, ScyllaDB 2.91 ms)

# Appendix IV: Determining ScyllaDB's maximum throughput

| Run | Hotspot Data Fraction | Hotspot Fraction | YCSB Client Count | Threads per Client | Total Threads | Cache-Hit Rate % | Average P99 Latency (ms) | Average Through-put (Ops/sec) |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.01 | 0.5 | 1 | 560 | 560 | 45 | 8 | 230 K |
| 2 | 0.01 | 0.5 | 1 | 616 | 616 | 45 | 8.5 | 230 K |
| 3 | 0.01 | 0.5 | 1 | 840 | 840 | 45 | 15 | 250 K |
| 4 | 0.01 | 0.5 | 1 | 560 | 1680 | 45 | 100 | 285 K |
| 5 | 0.01 | 0.55 | 1 | 840 | 840 | 50 | 15 | 260 K |

As the chart shows, throughput increases as the total number of threads is increased, but on the fourth run, the throughput increase is marginal, and latency increases exponentially. Therefore, we run the test with configurations of the run 5.

# Appendix V: About McKnight Consulting Group

Information management is all about enabling an organization to have data in the best place to successfully meet company goals. Mature data practices can integrate an entire organization across all core functions. Proper integration of that data facilitates the flow of information throughout the organization, which allows for better decisions made faster and with fewer errors. In short, well-done data can yield a better run company flush with real-time information and with less costs.

However, before those benefits can be realized, a company must go through the business transformation of an implementation and systems integration. For many who have been involved in those types of projects in the past – data warehousing, master data, big data, analytics - the path toward a successful implementation and integration can seem never-ending at times and almost unachievable. Not so with McKnight Consulting Group (MCG) as your integration partner, because MCG has successfully implemented data solutions for our clients for over a decade. We understand the critical importance of setting clear, realistic expectations up front and ensuring that time-to-value is achieved quickly.

MCG has helped over 100 clients with analytics, big data, master data management, and “all data” strategies and implementations across a variety of industries and worldwide locations. MCG offers flexible implementation methodologies that will fit the deployment model of your choice. The best methodologies, the best talent in the industry, and a leadership team committed to client success make MCG the right choice to help lead your project.

MCG, led by industry leader William McKnight, has deep data experience in a variety of industries that will enable your business to incorporate best practices while implementing leading technology. See www.mcknightcg.com.

# 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**