MCKNIGHT CONSULTING GROUP

Database Performance Comparison: Aerospike and Redis

Aerospike
Redis

William McKnight
Jake Dolezal

JUNE 2025

Table of Contents

Executive Summary

This benchmark report compares the performance,cost-effectiveness, and resilience of Aerospike versus Redis open source across various data sizes (1 TB, 5 TB, and 10 TB) for both 70/30 read/write and 100% write workloads. The results demonstrate that Aerospike consistently outperforms Redis in terms of latency, throughput and cost efficiency. The findings of this benchmark provide valuable insights for organizations seeking to optimize their costs for large-scale data storage and management.

Key findings:

- Aerospike's infrastructure cost per transaction is between 5.2x and 9.5x lower than Redis's across workloads and data sizes.
- Aerospike delivered sub-millisecond p99 latencies ranging from 17% to 48% lower than Redis across workloads and data sizes.
- Aerospike delivered between 11% and 24% higher throughput across workloads and data sizes compared to Redis.
- Aerospike proved to be the most cost-effective option, with annual infrastructure costs between 78% to 87% lower than Redis across data sizes.
- Aerospike's resiliency tests showed its ability to maintain availability during disruptions. The system recovered quickly from node-level failures, such as restarting database processes, rebooting nodes, and shutting down and restarting nodes, under a continuous workload of 600,000 operations per second.

We also compared Aerospike and Redis' fault tolerance under continuous workload conditions through three resiliency tests. The results show Aerospike recovered faster in process restarts but slower in node shutdowns. Note that Aerospike employed stronger consistency guarantees during failover whereas Redis was vulnerable to losing acknowledged writes, even in "normal" failover scenarios. Aerospike experienced between 1-11% drop in throughput and the time to get back to previous throughput ranging from 3 minutes to 2 hours, versus Redis' 7-18% drop in throughput and time to get back to previous throughput of 38 minutes to 57 minutes.

These results suggest that Aerospike is a strong choice for applications requiring high performance, cost efficiency, and resilience particularly for large-scale data sets.

Benchmark Methods and Platforms

To evaluate and compare the performance characteristics, we designed a benchmarking framework focused on the top three critical dimensions for their workloads: latency, throughput, and cost-efficiency, plus resilience testing. These metrics reflect common priorities in real-time and high-throughput application scenarios, where data systems must deliver consistent performance at scale while maintaining operational affordability and uptime.

Test design

Our methodology is grounded in realistic usage patterns, simulating read-only and mixed workloads across varying data volumes. Each database was deployed and tested in a controlled environment that ensures consistency in hardware specifications, network conditions, and data models. To reflect the differences between in-memory and disk-backed use cases, we used out-of-the-box configurations to align with each system's deployment model and best practices recommended by vendors.

YCSB

The Yahoo! Cloud Serving Benchmark (YCSB) is a widely adopted open-source framework designed to facilitate the evaluation of the performance characteristics of modern cloud-based and NoSQL data stores. Originally developed by Yahoo! Research, YCSB provides a standardized methodology to measure key performance metrics, primarily latency and throughput, under various workloads that simulate real-world usage patterns. We cloned the YCSB GitHub repository.

To run the 5 TB and 10 TB tests, you need to slightly alter the core/src/main/java/site/ycsb/workloads/CoreWorkload.java file:

Line 367, change this:
 protected int recordcount
To this:
 protected long recordcount

Lines 434:435, change this:
  recordcount =
    Int.parseInt(…
To this:
  recordcount =
    Long.parseLong(…

Lines 448:451, change this:
  int insertstart =
    Int.parseInt(...
  int insertcount=

```
    Int.parseInt(...
To this:
  long insertstart =
    Long.parseLong(...
  long insertcount=
    Long.parseLong(...
```

And then recompile with Maven.

Because the `recordcount` property in those workloads is 3,333,333,333 and 6,666,666,667, respectively, and that is greater than the max value for a `Java Int` (2,147,483,647). Although we did not use them, the `insertstart` and `insertcount` parameters also depend on `recordcount` as well, and Java will complain if they are not `Long,` too.

Scale factor

We conducted our tests using three scale factors of pre-generated YCSB data: **1 TB**, **5 TB**, and **10 TB**. The schema is a single table (`usertable`) with the following YCSB configuration:

YCSB Parameter	Value
`fieldcount`	10
`fieldlength`	150
`fieldlengthdistribution`	constant

Thus, each record is 1.5KB. To create the 3 data sizes, we used the following YCSB parameters:

YCSB parameter	Scale	Value
`recordcount`	1 TB	666,666,667
	5 TB	3,333,333,333
	10 TB	6,666,666,667

Workloads

We tested two different workloads with different read-to-write ratios:

- **70/30** – 70% reads/30% writes (updates only, no inserts)
- **100/0** – 100% reads only

In YCSB, you can set the read/write ratio to simulate different workload scenarios. We chose the 70/30 read-heavy, and 100/0 read-only ratios to benchmark, as they represent different, yet typical, workload scenarios - typical transactional, high write activity, and read-intensive cases. These provide a comprehensive evaluation of the databases' performance, scalability, and efficiency.

To run these workloads, we used the following YCSB settings:

YCSB parameter	Scale	Value
readproportion	0.7	1.0
updateproportion	0.3	0
scanproportion	0	0
insertproportion	0	0
requestdistribution	uniform	uniform

We also used a high number of operations to ensure the workload ran long enough to assess the stability of the workload on each platform:

YCSB parameter	Scale	Value
operationcount	1 TB	100,000,000
	5 TB	500,000,000
	10 TB	1,000,000,000
hdrhistogram.percentiles	-	95, 99, 99.9

We ran these workloads and captured latencies (average, 95th, 99th, and 99.9th percentiles) and throughput (operations per second), as shown in the next section.

Systems under test

Our test involved the following systems:

Aerospike
Enterprise Edition
v.8.0.0.5

Redis
Community Edition
v.8.0.0

Additional configurations

While we installed and tested these platforms with an "out-of-the-box" configuration, we made a few minor configuration settings based on documented best practice for this type of use case:

Aerospike

- Used a replication factor of 2 (RF2)
- Disabled Transparent Hugepages in the OS
- Partitioned each NVMe SSD into four (4) equal sized partitions to improve I/O throughput

Redis

- Used a replication factor of 2 (RF2)
- Set append-only file (AOF) fsync[1] to every 1 sec to balances performance and durability and is recommended to be used when minimal data loss is acceptable in the event of a failure—this is to mimic the default behavior of Aerospike
- Disabled Transparent Hugepages in the OS
- Set vm.overcommit_memory=1 in the OS

Infrastructure

We also took into account the sizing implications of scaling each system under these workload demands. This includes infrastructure for self-managed Aerospike and Redis deployments. The goal is to provide a comprehensive view of the trade-offs between raw performance and cloud infrastructure requirements, enabling practitioners to make informed architectural decisions.

Aerospike

Since Aerospike recommends using fast NVMe solid-state drives (SSD), we chose EC2 instances that had an adequate amount of disk storage for the primary index using Aerospike's documented guidance[2].

Data size	**1 TB**	**5 TB**	**10 TB**
Instance type	c7gd.4xlarge	i8g.4xlarge	i8g.4xlarge
vCPU per node	16	16	16
Memory per node	32 GB	128 GB	128 GB
SSD disk space per node	1 x 950 GB (NVMe)	1 x 3750 GB (AWS Nitro)	1 x 3750 GB (AWS Nitro)
Root disk size per node	gp2 20GB 100iops	gp2 20GB 100iops	gp2 20GB 100iops
Node count	5	7	14

Table 1: Cluster configurations, Aerospike

Redis

While Redis does not provide an explicit capacity planning guide, we used our field experience and documented best practice for in-memory stores. The formula to calculate the memory required to store our data set with Redis was:

Total memory (GB) = **Data size** (GB) x **Replication factor** (RF2) + **30% Overhead** (recommended by Redis[3])

[1] https://redis.io/docs/latest/operate/oss_and_stack/management/persistence/
[2] https://aerospike.com/docs/database/manage/planning/capacity
[3] https://redis.io/docs/latest/operate/rs/installing-upgrading/install/plan-deployment/hardware-requirements

In addition, we provisioned some adequately robust EBS disks for append-only file log persistence for Redis.

Data size	**1 TB**	**5 TB**	**10 TB**
Instance type	r6g.8xlarge	r6g.16xlarge	r6g.16xlarge
vCPU per node	32	64	64
Memory per node	256 GB	512 GB	512 GB
Total AOF disks	gp3 256GB 6,000 IOPS 125MB/s	gp3 256GB 6,000 IOPS 125MB/s	gp3 256GB 6,000 IOPS 125MB/s
Node count	10	23	46

Table 2: Cluster configurations, Redis

Benchmark Results

Performance latency

In YCSB tests, latency is a crucial metric because it directly impacts the user experience and system performance. **Lower latency indicates faster response times**, which is essential for:

- **Real-time applications**: Latency affects how quickly data is retrieved or updated, impacting applications that require immediate responses.
- **User experience**: Higher latency can lead to slower page loads, frustrated users, and decreased engagement.
- **System scalability**: As latency increases, systems may become bottlenecked, limiting their ability to handle increased traffic or workload.

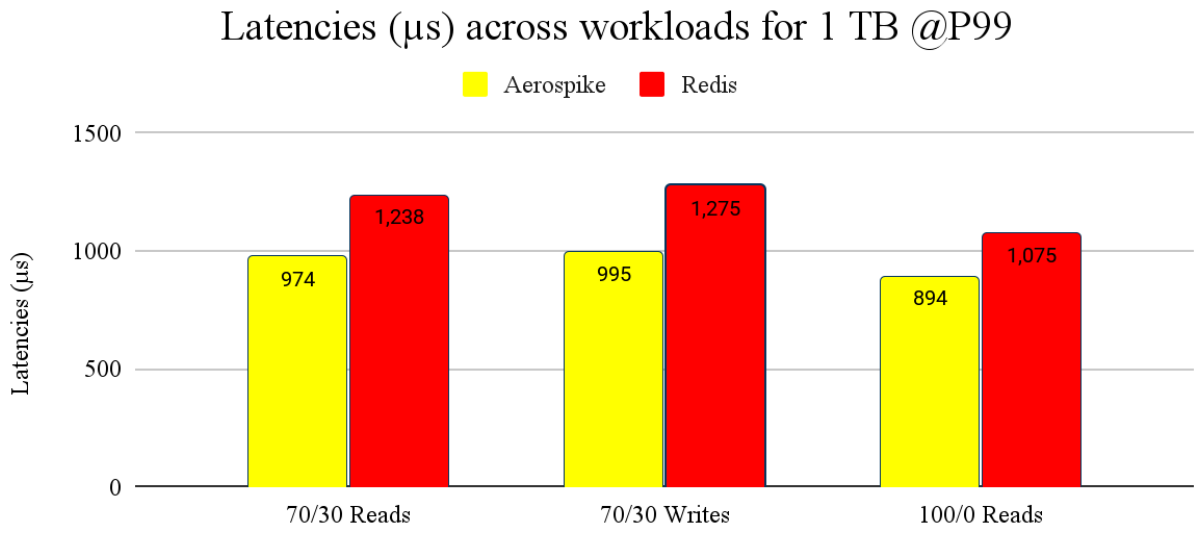

Figure 1: Latencies across workloads for 1 TB @P99

The performance comparison of Aerospike, Redis under two different workloads (70% reads/30% writes and 100% reads/0% writes) with 1 TB data size and RF2 replication factor reveals Aerospike performs the best, with the lowest latency across all workloads (974, 995 and 894 microseconds). Redis has the highest numbers. The results suggest Aerospike is optimized for efficient operations in these tests, handling reads and writes effectively.

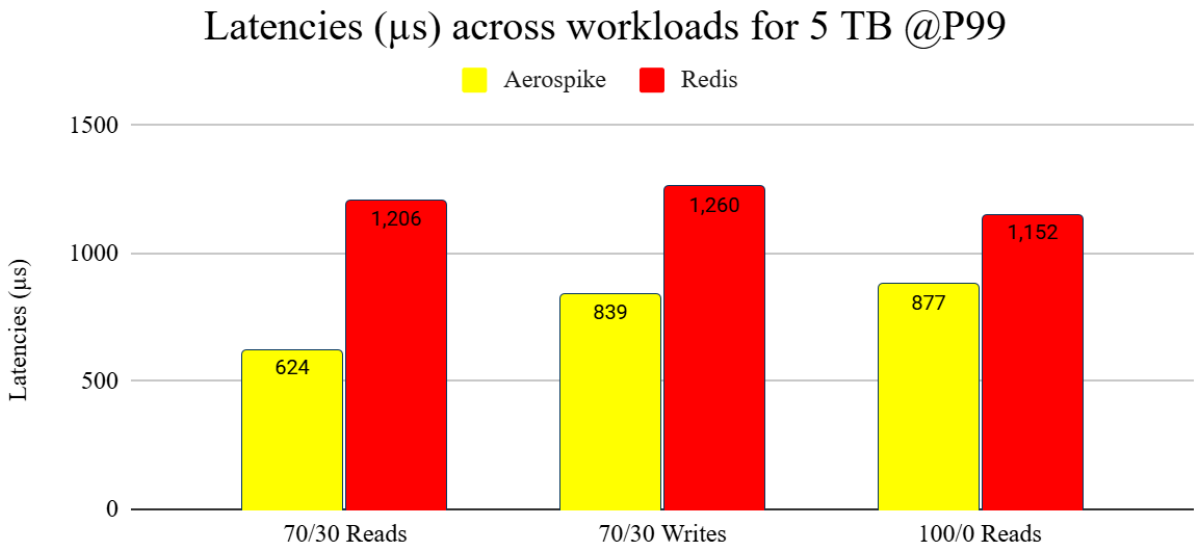

Figure 2: Latencies across workloads for 5 TB @P99

The performance comparison of Aerospike and Redis under two different workloads (70% reads/30% writes and 100% reads/0% writes) with 5 TB data size and RF2 replication factor shows Aerospike performs the best, with the lowest latency across all workloads (624, 839, and 877 microseconds). The results suggest Aerospike's efficiency and scalability hold up well with the increased 5 TB data size.

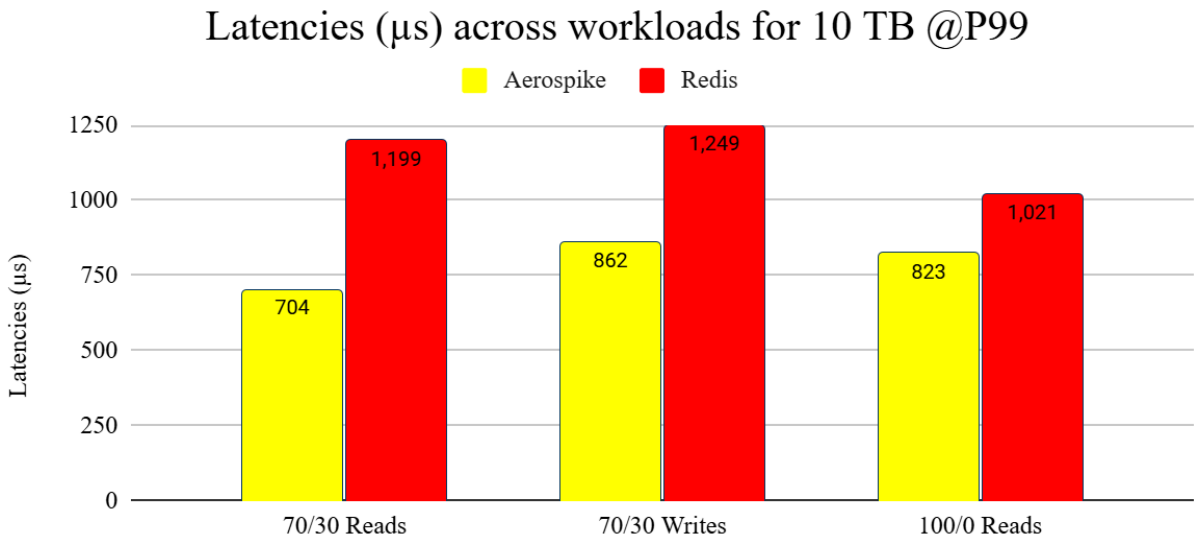

Figure 3: Latencies across workloads for 10 TB @P99

The established pattern continued at 10 TB, with Aerospike once again showing the least latency across all workloads. It is reasonable to conclude that this pattern would continue into higher volume.

Performance latency by workload and SLA

As performance by service level agreement (SLA) is often a deciding input to how systems are designed, these three charts exhibit each systems' relative latency by SLA.

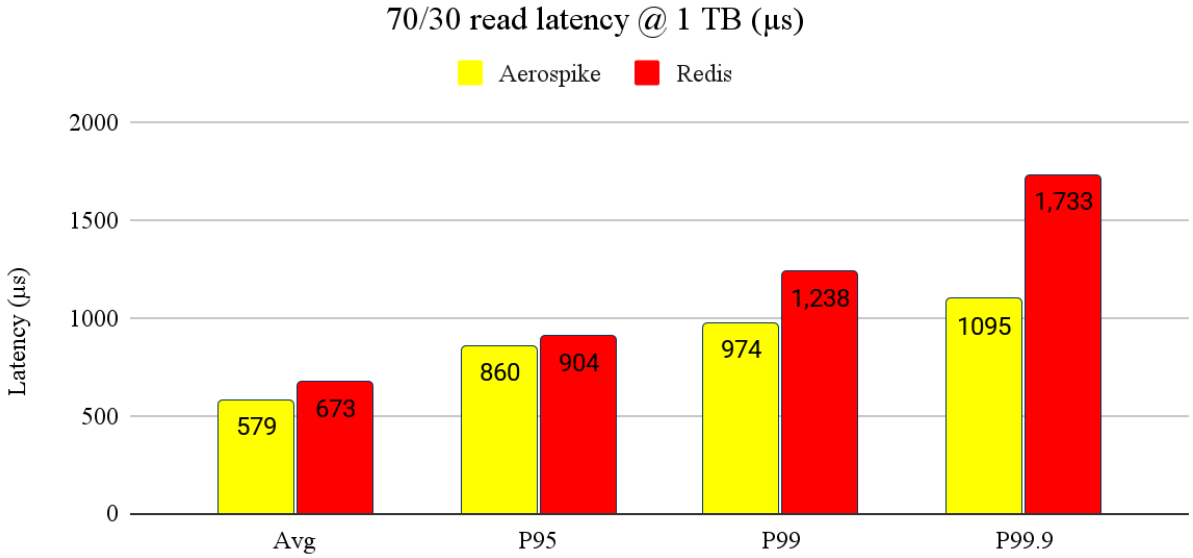

Figure 4: 70/30 read latency @ 1 TB

Aerospike's relatively low latency across SLAs is less than Redis for 1 TB for the 70% reads from the 70/30 workload. We see relatively similar behavior even with 30% writes of the 70/30 workload at 5 TB:

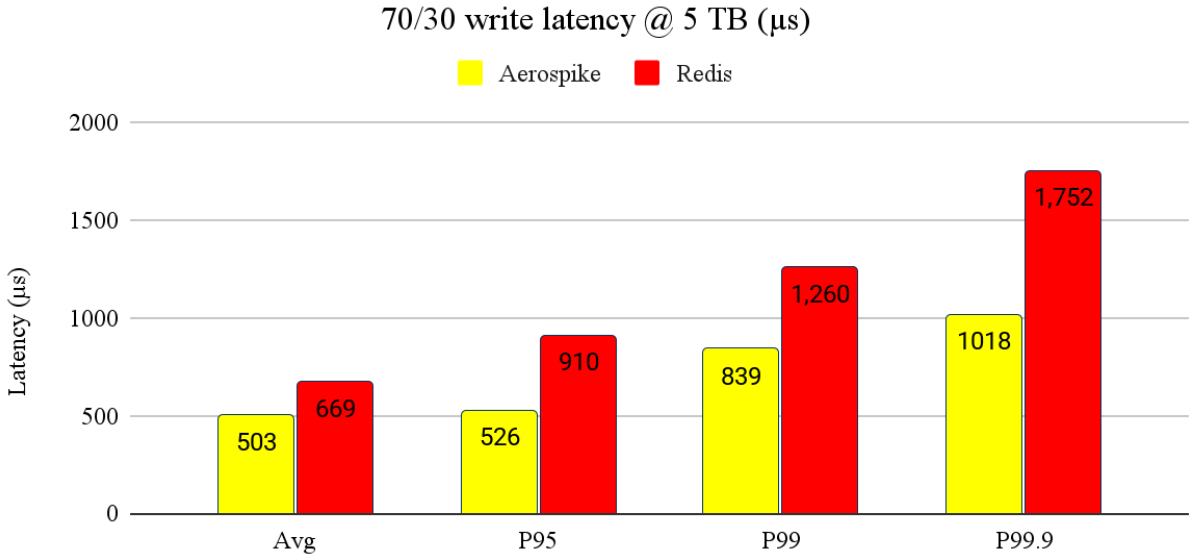

Figure 5: 70/30 write latency @ 5 TB

and for 100% reads from the 100/0 read-only workload latencies across SLA for 10 TB:

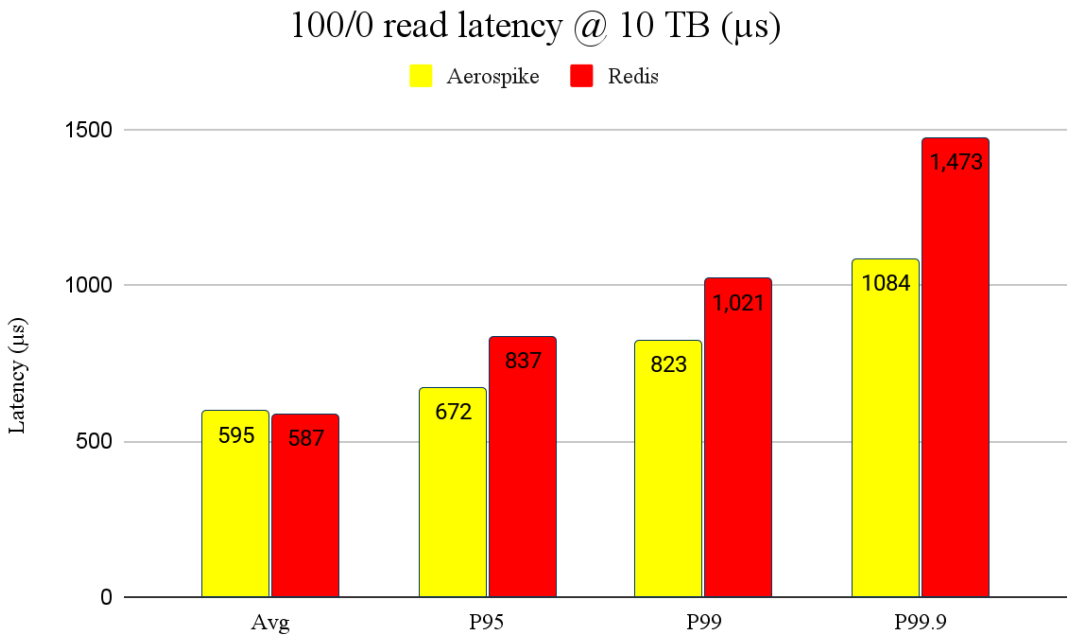

Figure 6: 100/0 read latency @ 10 TB

All workloads and data sizes are listed in the Appendix.

Performance throughput

In YCSB tests, throughput is a critical metric that measures the number of operations (e.g., reads, writes, updates) a database or system can handle per unit of time. In our case, its operations per second. **Higher throughput indicates better performance and scalability.**

Throughput is essential because it directly impacts:

- **System scalability**: Higher throughput enables systems to handle increased traffic, workload, or user demand.
- **Performance under load**: Throughput testing reveals how well a system performs under stress, helping identify potential bottlenecks.
- **Capacity planning**: Throughput metrics inform capacity planning, ensuring systems can handle expected workloads.

We measured throughput for the 70/30 reads/writes test and the read-only test because these workloads are more representative of typical use cases where read operations dominate, allowing us to evaluate the system's performance under realistic conditions and identify potential bottlenecks in read-heavy scenarios.

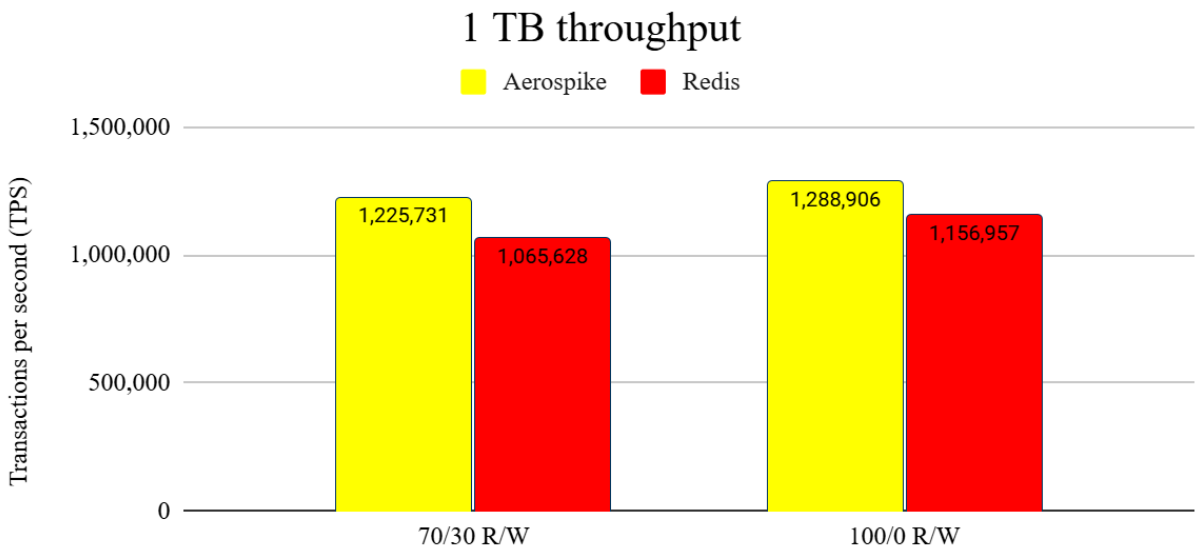

Figure 7: 1 TB throughput across workloads

The throughput comparison reveals Aerospike significantly outperforms Redis at 1 TB, with notably higher throughput in both 70/30 reads/writes (1,225,731) and 100/0 reads (1,288,906) workloads. Redis has lower throughput (1,065,628 and 1,156,957). Compared to Redis, Aerospike has approximately 13% and 10% more throughput per second for the respective workloads. Aerospike's high throughput suggests it's optimized for efficient operations, particularly in read-heavy scenarios.

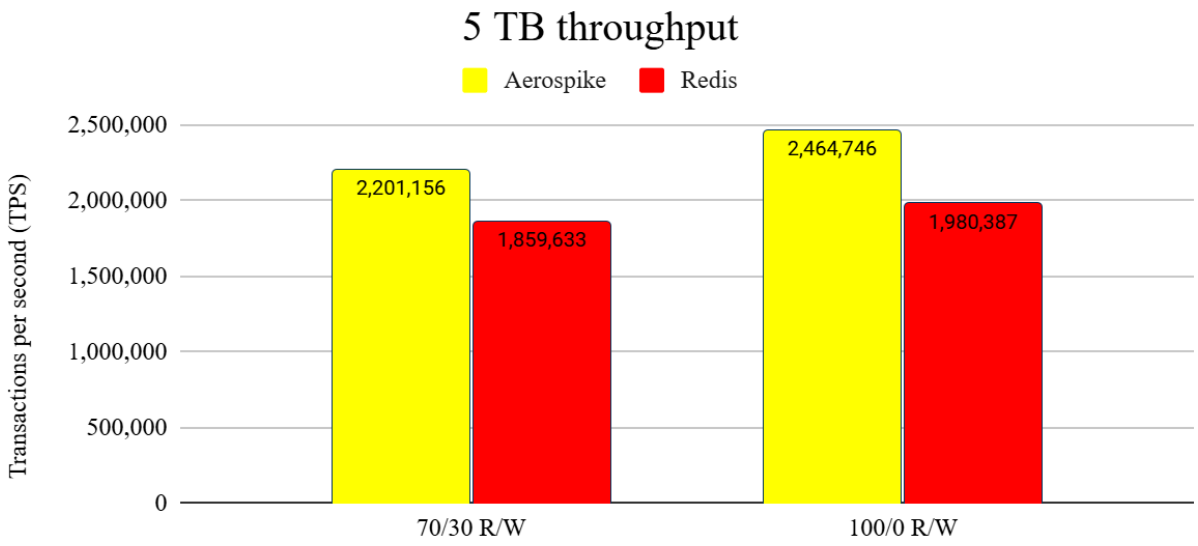

Figure 8: 5 TB throughput across workloads

The throughput comparison shows Aerospike outperforming Redis in both 70/30 reads/writes and 100/0 reads workloads with 5 TB data size and RF2 replication factor. Aerospike's

throughput remains very high at 2,201,156 and 2,464,746 TPS, while Redis shows relatively lower throughput at 1,859,633 and 1,980,387 TPS, indicating Aerospike's strong performance and scalability in handling datasets at 5 TB.

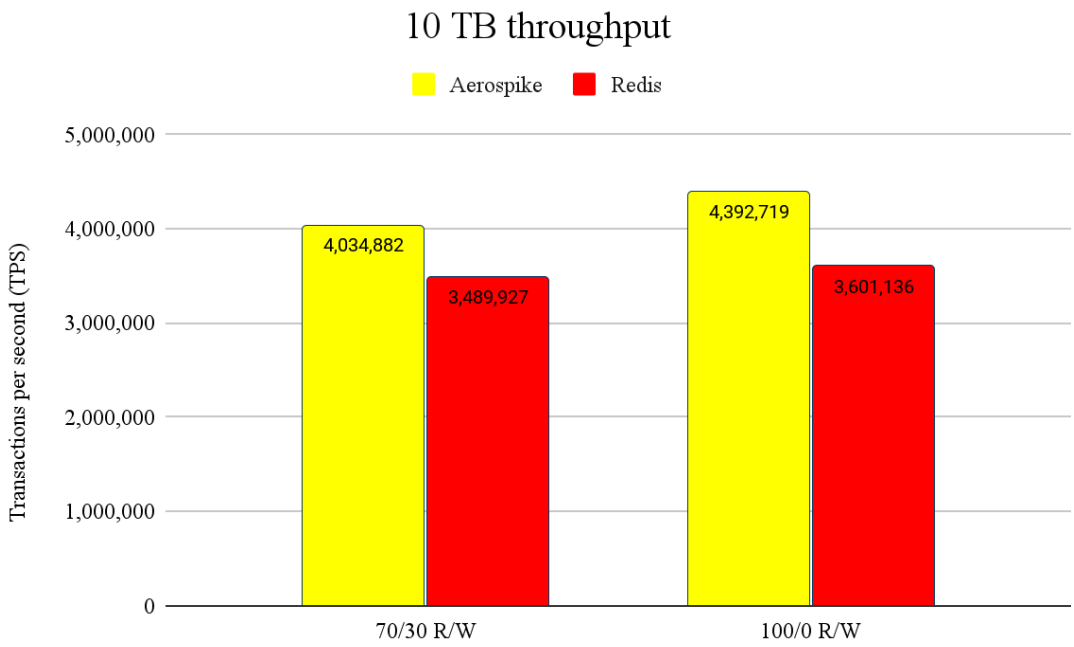

Figure 9: 10 TB throughput across workloads

The established pattern continued at 10 TB, with Aerospike once again showing the most throughput by far across all workloads. Aerospike outperforms Redis, with throughput per second approximately 14% and 18% greater than Redis. It is reasonable to conclude that this pattern would continue into higher volume.

Pricing

With these measured differences in latency and throughput, it might be expected that Aerospike would be higher in cost so we projected costs to an annual level to find out[4]. Data migration costs are not included for any of the systems tested.

We note specifically that the pricing used is 1 year upfront payment which for Aerospike and Redis, both of which were run using EC2 instances, is 1-year, all upfront and thus lower than on-demand pricing. This pricing model is the most fair to use in a benchmark because it reflects the actual costs incurred based on peak usage across all platforms, allowing for an accurate comparison of performance and cost efficiency.

[4] See https://aerospike.com/products/features-and-editions/ for Aerospike pricing guidance on their software costs.

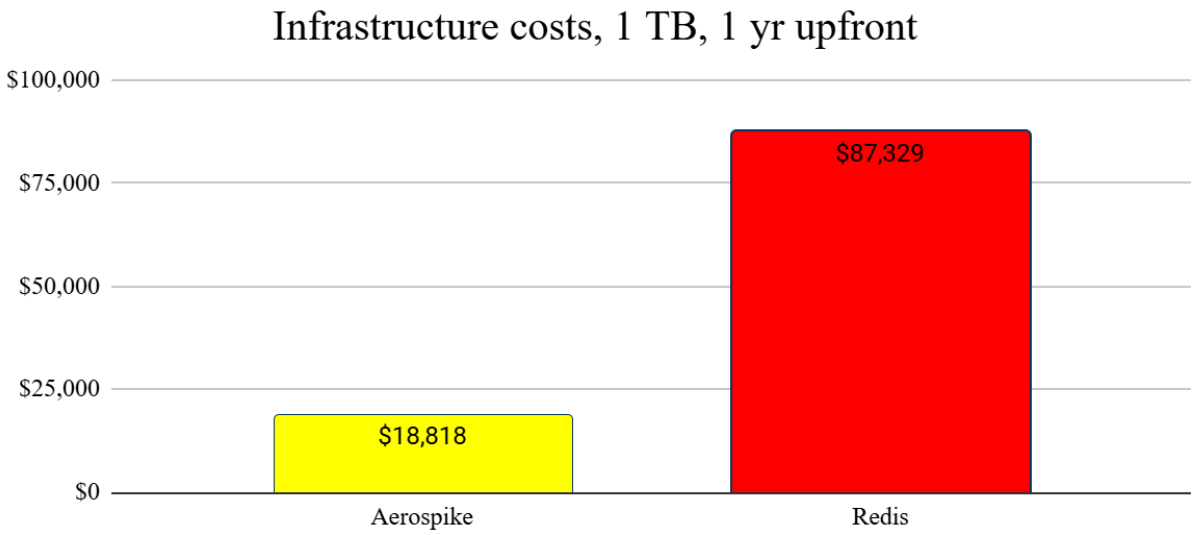

Figure 10: Infrastructure costs, 1 TB, 1 year upfront

Aerospike is the more cost-effective option for a 1 TB data size, with an annual cost of $18,818. Compared to Redis ($87,329), Aerospike offers significant savings of 78% making it the more economical choice.

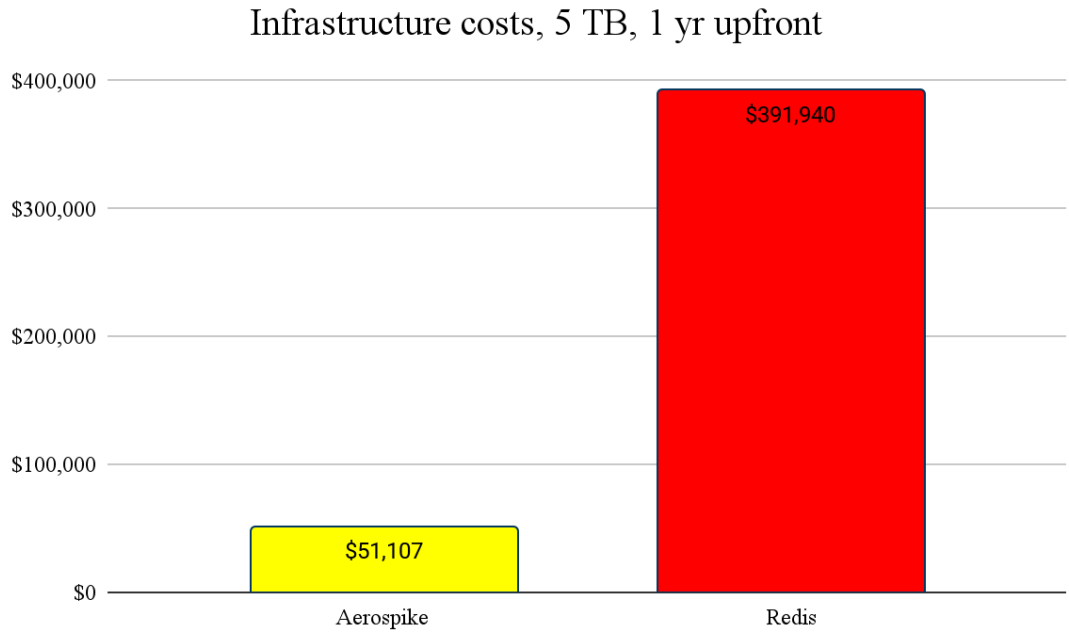

Figure 11: Infrastructure costs, 5 TB, 1 year upfront

The savings get more pronounced with the larger 5 TB data set compared to Redis. Aerospike offers 87% cost savings to Redis.

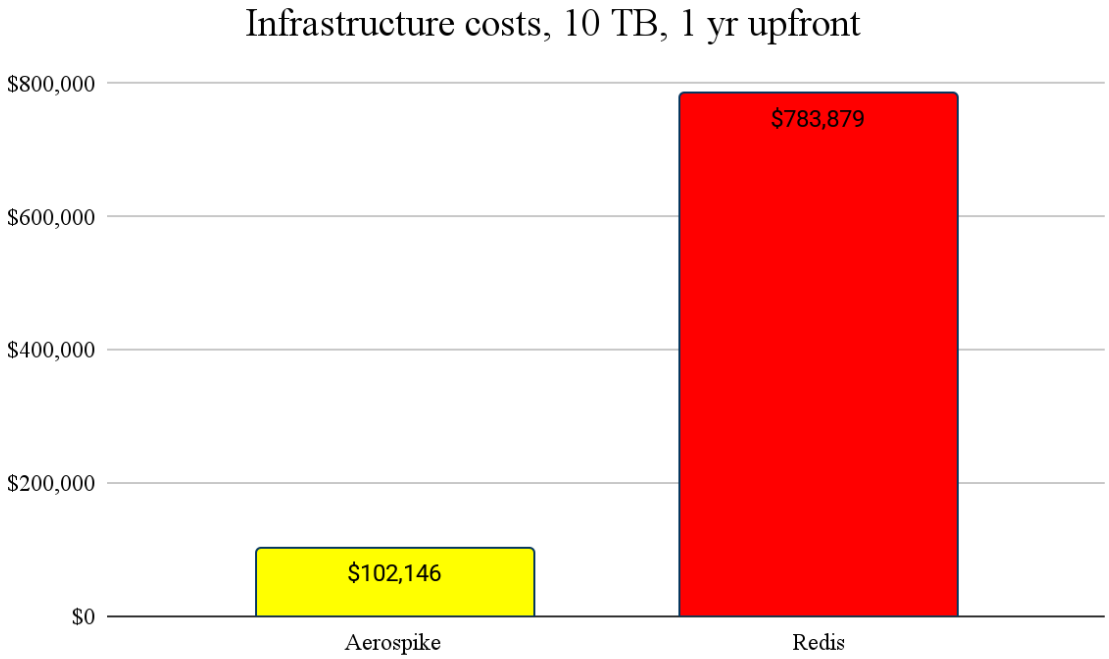

Figure 12: Infrastructure costs, 10 TB, 1 year upfront

We are not surprised to see that Aerospike is the more cost effective at the 10 TB level, as well. It is reasonable to assume this pattern would continue into larger data sets.

Resiliency testing

We also conducted three resiliency tests[5] that compares the fault tolerance of Aerospike and Redis under continuous workload conditions. To evaluate how each system handles node-level disruptions within a cluster, we subjected both databases to three controlled failure scenarios: 1) restarting the database process(es) on a single node, 2) rebooting the node entirely, and 3) shutting down and manually restarting the node after a delay. Throughout each test, we measured the impact on throughput and recorded the time it took for the affected node to fully rebalance and return to 100% of its pre-failure performance. The results highlight differences in failover behavior, recovery speed, and the ability of each system to maintain availability during the disruption of a scheduled maintenance event or an unplanned outage.

We repeated the 1 TB configurations from the latency tests and used the YCSB 70% reads/30% writes workload, but this time we held the throughput to near 600,000 operations per second to simulate a day-in-day-out normal workload, instead of a peak workload. Once the throughput stabilized, we conducted our resiliency tests.

Note that Aerospike employed stronger consistency guarantees during failover whereas Redis was vulnerable to losing acknowledged writes, even in "normal" failover scenarios. Aerospike

[5] Details are in the Appendix.

employed its default write commit setting of `COMMIT_ALL` while Redis did not use their `WAIT` command.

Note that Redis had an inherent advantage. Our 1 TB configuration of Redis had 10 nodes, while the Aerospike configuration had 5 nodes. Thus, the Redis test only impacted 10% of the overall data and 1/10th its compute, while the Aerospike test impacted 20% of its data and 1/5th its compute power. We could have reconfigured the test for a more equal data distribution; however, we realize that enterprises deploying these technologies in the field will likely have fewer Aerospike nodes than Redis nodes. We also realize that Redis could have been configured with a cluster of many, many smaller nodes—further diminishing the impact of a single node failure. However, the goal of this test is more to understand Aerospike's resiliency given its smaller infrastructure footprint.

Furthermore, in our observations of Redis, it seemed to recover its throughput levels in a linear fashion, evidenced by tracking the key counts restored in its keyspaces following the outage. We might assume the overall recovery time is linear to the number of keys or amount of data on each node. Thus, if we had lost 20% of the nodes/data in a Redis outage, it is possible the recovery time may have been twice as long (in addition to a greater throughput loss).

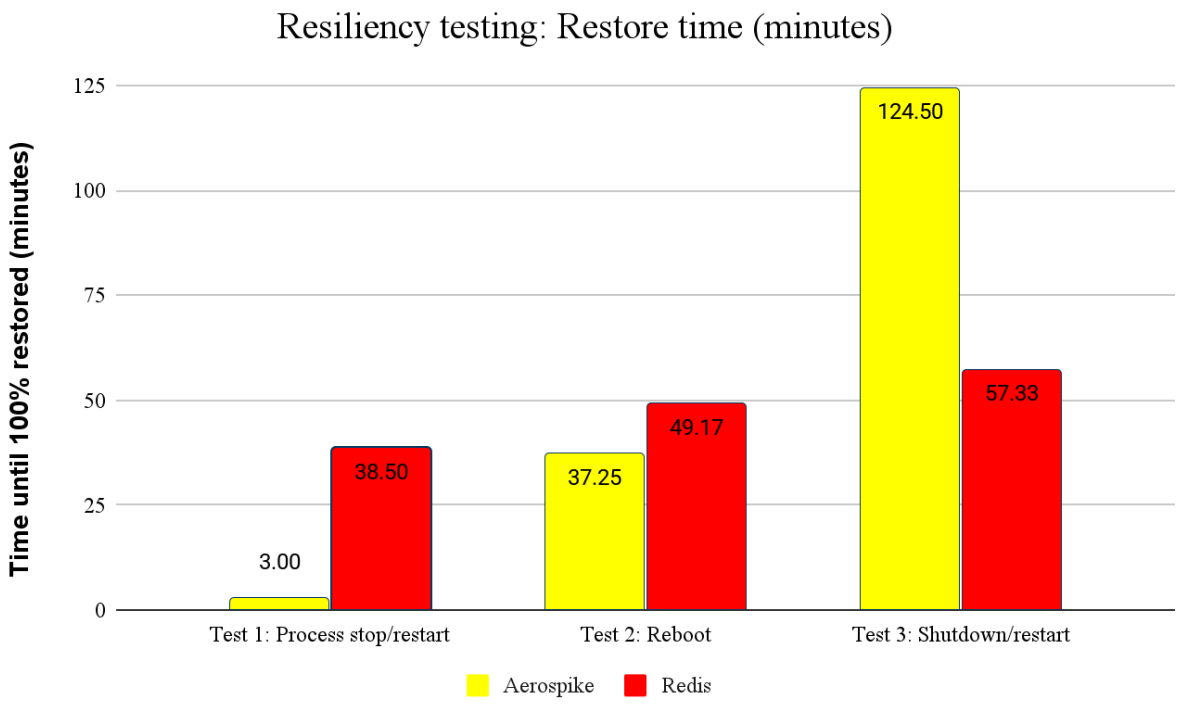

Figure 13: Resiliency testing: Restore time

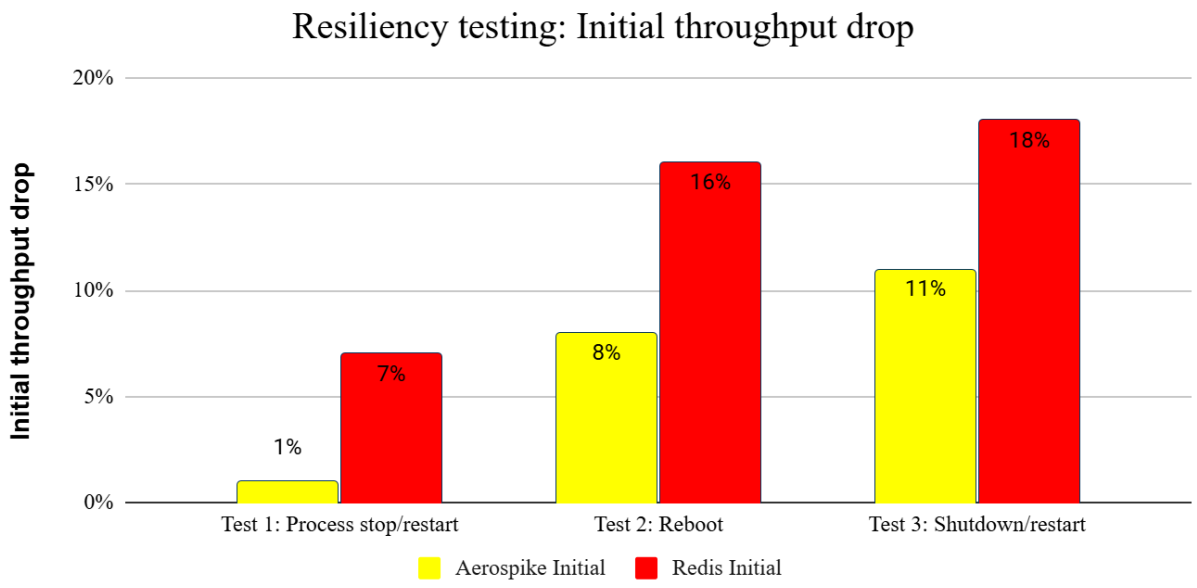

Figure 14: Resiliency testing: Initial throughput drop

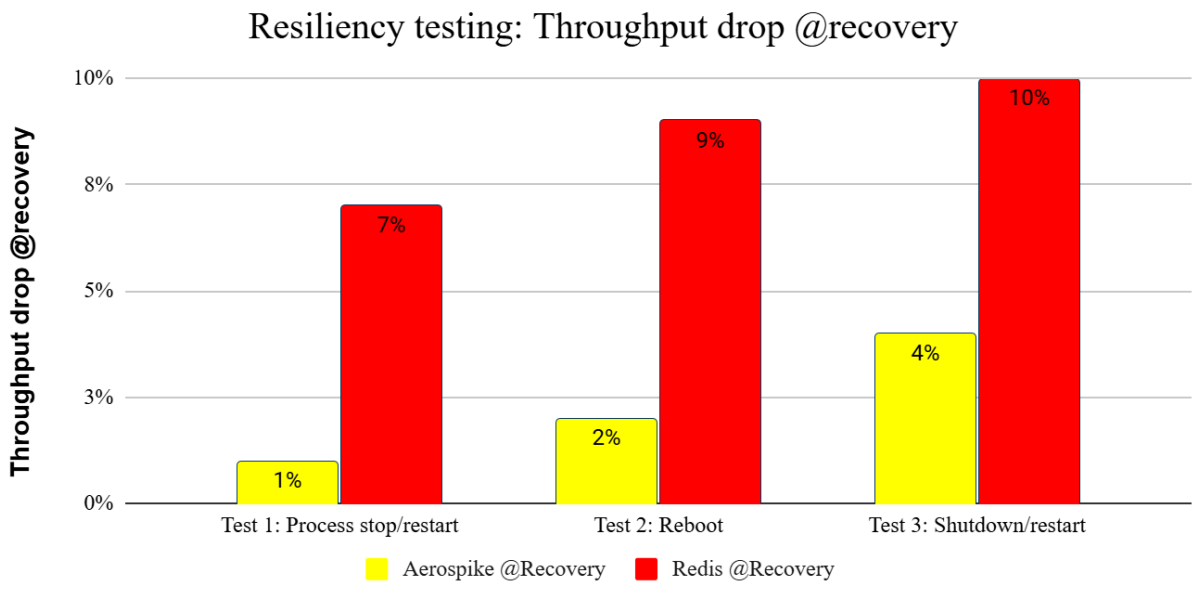

Figure 15: Resiliency testing: Throughput drop at recovery

Conclusion

Aerospike's superior performance, cost efficiency, and resilience as demonstrated in this report, have significant implications for organizations handling large-scale data. With its low latency and high throughput, Aerospike enables faster data processing than Redis open source, allowing businesses to gain insights and make decisions more quickly. This can be particularly beneficial for applications that require real-time data processing and analytics.

Also the cost-effectiveness of Aerospike can also lead to substantial savings on infrastructure costs, freeing up resources for innovation and growth. For organizations with rapidly growing data sets, Aerospike's performance and cost efficiency make it a very attractive option.

Appendix

Pricing and infrastructure details

Data Size: 1 TB		
Platform	**Aerospike**	**Redis**
Instance type	c7gd.4xlarge	r6g.8xlarge
vCPU per node	16	32
Memory per node (GB)	32	256
SSDs per node (GB)	1x950	-
Node count	5	10
Instance cost per hour, on demand	$0.7260	$1.613
Total per hour, all nodes	$3.630	$16.13
EBS disks/ DynamoDB storage	gp2 20GB 100iops	gp3 256GB 6000iops 125MB/s
Disk per month	$2.00	$35.48
Total disks/mo	$10.00	$354.80
Per month, on demand	$2,659.17	$12,128.24
Provisioned reads/sec	n/a	n/a
Provisioned writes/sec	n/a	n/a
Total upfront cost (1 yr reserved capacity)	n/a	n/a
Per year, on demand	$31,910	$145,539
1-Year, all upfront	$18,818	$87,329
Total cost (lower option)	**$18,818**	**$87,329**
Relative cost to Aerospike	-	*4.64x*

Data Size: 5 TB		
Platform	**Aerospike**	**Redis**
Instance type	i8g.4xlarge	r6g.16xlarge
vCPU per node	16	64
Memory per node (GB)	128	512
SSDs per node (GB)	1x3750	-
Node count	7	23
Instance cost per hour, on demand	$1.373	$3.226
Total per hour	$9.61	$74.19
EBS disks/ DynamoDB storage	gp2 20GB 100iops	gp3 256GB 6000iops 125MB/s
Disk per month	$2.000	$35.48
Total disks/mo	$14.00	$816.04
Per month, on demand	$7,026.21	$54,973.86
Provisioned reads/sec	n/a	n/a
Provisioned writes/sec	n/a	n/a
Total upfront cost (1 yr reserved capacity)	n/a	n/a
Per year, on demand	$84,314.52	$659,686.37
1-Year all upfront	$51,106.52	$391,939.60
Total cost (lower option)	**$51,107**	**$391,940**
Relative cost to Aerospike	-	*7.67x*

Data Size: 10 TB		
Platform	**Aerospike**	**Redis**
Instance type	i8g.4xlarge	r6g.16xlarge
vCPU per node	16	64
Memory per node (GB)	128	512
SSDs per node (GB)	1x3750	-
Node count	14	46
Instance cost per hour, on demand	$1.373	$3.226
Total Per Month	$19.22	$148.38
EBS disks/ DynamoDB storage	gp2 20GB 100iops	gp3 256GB 6000iops 125MB/s
Disk per month	$1.60	$35.48
Total disks/mo	$22.40	$1,632.08
Per month, on demand	$14,052.42	$109,947.73
Provisioned reads/sec	n/a	n/a
Provisioned writes/sec	n/a	n/a
Total upfront cost (1 yr reserved capacity)	n/a	n/a
Per Year, on demand	$168,629.04	$1,319,372.76
1-Year all upfront	$102,145.85	$783,879.19
Total cost (lower option)	**$102,146**	**$783,879**
Relative cost to Aerospike	-	*7.67x*

Resiliency testing details

Aerospike	Redis
Resiliency Test #1:	**Resiliency Test #1:**
Stop Aerospike system process on 1 node, wait 30 seconds, restart process:	Shutdown Redis processes on 1 node, wait 30 seconds, restart processes:
sudo systemctl stop aerospike; sleep 30; sudo systemctl daemon-reload; sudo systemctl start aerospike	for p in 7000 7001 7002 7003 7004 7005 7006 7007 7008 7009 7010 7011 7012 7013 7014 7015; do redis-cli -p ${p} shutdown & done
Infra: instance-type=c7gd.4xlarge count=5	Infra: instance-type=r6g.8xlarge count=10
Workload: 1 TB 70/30 ~600K ops/sec	Workload: 1 TB 70/30 ~600K ops/sec
Stopped at: 2025-05-27 16:54:30 -30s	Stopped at: 2025-05-30 12:00:00 -30s
Restarted at: 2025-05-27 16:55:00 0s	Restarted at: 2025-05-30 12:00:30 0s
Node began processing traffic again at: 2025-05-27 16:55:25 25s	Node began processing traffic again at: 2025-05-30 12:00:32 2s
With only 1% reduction in overall throughput until 100% restored	With a 7% reduction in overall throughput until 100% restored
100% restored (back to previous throughput and 0 pending migrations) at: 2025-05-27 16:58:00 3m	100% restored (back to previous throughput and keys restored) at: 2025-05-30 12:39:00 38m30s

Aerospike	Redis
Resiliency Test #2:	**Resiliency Test #2:**
Reboot 1 node, wait 30 seconds, restart process:	Reboot 1 node, wait 30 seconds, restart processes:
sudo reboot; sleep 30; ssh; sudo systemctl start aerospike	sudo reboot; sleep 30; ssh; for p in 7000 7001 7002 7003 7004 7005 7006 7007 7008 7009 7010 7011 7012 7013 7014 7015; do time redis-server redis-cluster/${p}/redis.conf; done
Infra: instance-type=c7gd.4xlarge count=5	Infra: instance-type=r6g.8xlarge count=10
Workload: 1 TB 70/30 ~600K ops/sec	Workload: 1 TB 70/30 ~600K ops/sec
Reboot at: 2025-05-27 17:08:30 -30s	Reboot at: 2025-05-30 13:12:00 -30s
Restarted at: 2025-05-27 17:09:00 0s	Restarted at: 2025-05-30 13:12:30 0s
Node began processing traffic again at: 2025-05-27 17:18:20 9m20s	Node began processing traffic again at: 2025-05-30 13:14:50 2m20s
With an initial drop of 8% in throughput that improved to just 2% until 100% restored	With an initial drop of 16% in throughput that improved to 9% until 100% restored
100% restored (back to previous throughput and 0 pending migrations) at: 2025-05-27 17:46:15 37m15s	100% restored (back to previous throughput and keys restored) at: 2025-05-30 14:01:40 49m10s

Aerospike	Redis
Resiliency Test #3:	**Resiliency Test #3:**
Shutdown 1 node, wait 2 minutes, restart node, ssh, restart process:	Shutdown 1 node, wait 2 minutes, restart node, ssh, restart processes:
sudo shutdown 0; sleep 120; start; ssh; sudo systemctl start aerospike	sudo reboot; sleep 120; ssh; for p in 7000 7001 7002 7003 7004 7005 7006 7007 7008 7009 7010 7011 7012 7013 7014 7015; do time redis-server redis-cluster/${p}/redis.conf; done
Infra: instance-type=c7gd.4xlarge count=5	Infra: instance-type=r6g.8xlarge count=10
Workload: 1 TB 70/30 ~600K ops/sec	Workload: 1 TB 70/30 ~600K ops/sec
Shutdown at at: 2025-05-27 18:07:00 -2m	Shutdown at: 2025-05-30 14:26:00 -2m
Restarted process at: 2025-05-27 18:09:00 0s	Restarted at: 2025-05-30 14:28:00 0s
Node began processing traffic again at: 2025-05-27 18:10:50 1m50s	Node began processing traffic again at: 2025-05-30 14:36:10 8m10s
With an initial drop of 11% in throughput that improved to just 4% until 100% restored	With an initial drop of 18% in throughput that improved to 10% until 100% restored
100% restored (back to previous throughput and 0 pending migrations) at: 2025-05-27 20:13:30 2h04m30s	100% restored (back to previous throughput and keys restored) at: 2025-05-30 15:25:20 57m20s

About McKnight Consulting Group

Information Management is all about enabling an organization to have data in the best place to succeed to 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 that 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 makes 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 for mission-critical use cases and workloads, including machine learning, generative, and agentic AI. Aerospike powers millions of transactions per second with millisecond latency, at a fraction of the cost of other databases. Global leaders, including Adobe, Airtel, Barclays, Criteo, DBS Bank, Experian, HDFC Bank, PayPal, Sony Interactive Entertainment, and Wayfair rely on Aerospike for customer 360, fraud detection, real-time bidding, and other use cases. Headquartered in Mountain View, California, our offices are also located in London, Bangalore, and Tel Aviv.

Disclaimer

McKnight Consulting Group (MCG) runs all its tests to strict ethical standards. The results of the report are the objective and unbiased results of the application of queries to the simulations described in the report. The report clearly defines the selected criteria and process used to establish the field test. The report also clearly states the data set sizes, the platforms, the methods, etc. that were used. The reader is left to determine for themselves how to qualify the information for their individual needs. The report does not make any claims regarding third-party certification and presents the objective results received from the application of the process to the criteria as described in the report. The report strictly measures performance and cost and does not purport to evaluate other factors that potential customers may find relevant when making a purchase decision. This is a sponsored report. The client chose its configuration, while MCG chose the test, configured the database and testing application, and ran the tests. MCG also chose the most compatible configurations for the other tested platforms. Choosing compatible configurations is subject to judgment. The information necessary to replicate this test is included. Readers are encouraged to compile their own representative configuration and test it for themselves.