benchANT

Benchmarking MongoDB vs ScyllaDB: Performance, Scalability & Cost

Where ScyllaDB Beats MongoDB, and Why

> NoSQL databases promise to provide high performance and horizontal scalability for data-intensive workloads. In this report, we benchmark the performance and scalability of the market-leading general-purpose NoSQL database MongoDB and its performance-oriented challenger ScyllaDB.

- The benchmarking study is carried out against the DBaaS offers of each database vendor to ensure a fair comparison.
- The DBaaS clusters range from 3 to 18 nodes, classified in **three scaling sizes** that are comparably priced.
- The benchmarking study comprises **three workload types** that cover read-heavy, read-update and write-heavy application domains with data set sizes from 250GB to 10TB.
- The benchmarking study compares a total of **133 performance measurements** that range from throughput (per cost) to latencies to scalability.

ScyllaDB outperforms MongoDB in 132 of 133 measurements.

- For all the applied workloads, ScyllaDB provides **higher throughput (up to 20 times)** results compared to MongoDB.
- ScyllaDB achieves **P99 latencies below 10 ms** for insert, read and write operations for almost all scenarios. In contrast, MongoDB achieves P99 latencies below 10 ms only for certain read operations while the **MongoDB insert and update latencies are up 68 times higher** compared to ScyllaDB.
- ScyllaDB achieves up to **near linear scalability** while MongoDB shows less efficient horizontal scalability.
- The price-performance ratio clearly shows the strong advantage of ScyllaDB with **up to 19 times better price-performance ratio** depending on the workload and data set size.

In summary, this benchmarking study shows that ScyllaDB provides a great solution for applications that operate on data sets in the terabyte range and that require high (e.g, over 50K OPS) throughput while providing predictable low latency for read and write operations.

ABOUT THIS BENCHMARK

The NoSQL database landscape is continuously evolving. Over the last 15 years, it has already introduced many options and tradeoffs when it comes to selecting a high performance and scalable NoSQL database. In this report, we address the challenge of selecting a high performance database by evaluating two popular NoSQL databases: MongoDB, the market leading general purpose NoSQL database and ScyllaDB, the high-performance NoSQL database for large scale data. See our technical comparison for an in-depth analysis of MongoDB's and ScyllaDB's data model, query languages and distributed architecture.

For this project, we benchmarked both database technologies to get a detailed picture of their performance, price-performance and scalability capabilities under different workloads. For creating the workloads, we use the YCS, an open source and industry standard benchmark tool. Database benchmarking is often said to be non-transparent and to compare apples with pears. In order to address these challenges, this benchmark comparison is based on benchANT's scientifically proven Benchmarking-as-a-Service platform. The platform ensures a reproducible benchmark process (for more details, see the associated research papers on Mowgli and benchANT) which follows established guidelines for database benchmarking.

This benchmarking project was carried out by benchANT and was sponsored by ScyllaDB with the goal to provide a fair, transparent and reproducible comparison of both database technologies. For this purpose, all benchmarks were carried out on the database vendors' DBaaS offers, namely MongoDB Atlas and ScyllaDB Cloud, to ensure a comparable production ready database deployment. Further, the applied benchmarking tool was the standard Yahoo! Cloud Serving benchmark and all applied configuration options are exposed.

In order to ensure full transparency and also reproducibility of the presented results, all benchmark results are publicly available on GitHub. This data contains the raw performance measurements as well as additional metadata such DBaaS instance details and VM details for running the YCSB instances. The interested reader will be able to reproduce the results on their own even without the benchANT platform.

RESULTS OVERVIEW

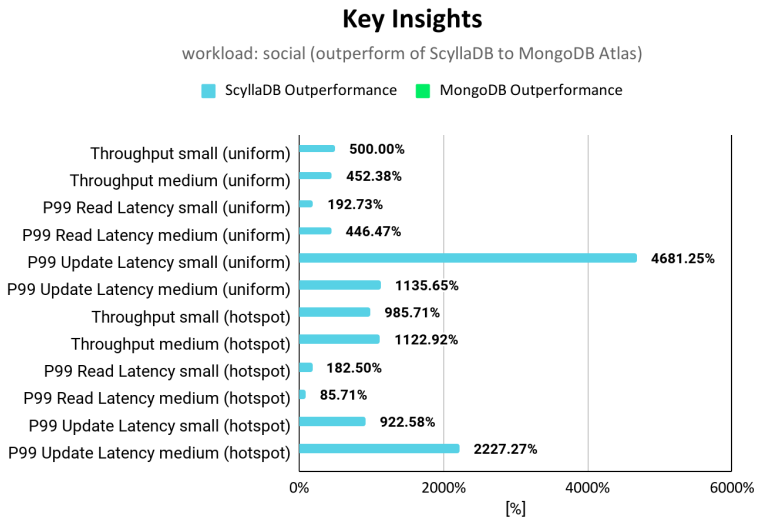
The complete benchmark covers three workloads: social, caching, and sensor.

- The **social** workload is based on the YCSB Workload B. It creates a read-heavy workload, with 95% read operations and 5% update operations. We use two shapes of this workload, which differ in terms of the request distribution patterns, namely uniform and hotspot distribution. These workloads are executed against the small database scaling size with a data set of 500GB and against the medium scaling size with a data set of 1TB.
- The **caching** workload is based on the YCSB Workload A. It creates a read-update workload, with 50% read operations and 50% update operations. The workload is executed in two versions, which differ in terms of the request distribution patterns (namely uniform and hotspot distribution). This workload is executed against the small database scaling size with a data set of 500GB, the medium scaling size with a data set of 1TB and a large scaling size with a data set of 10TB.
- The **sensor** workload is based on the YCSB and its default data model, but with an operation distribution of 90% insert operations and 10% read operations that simulate a real-world IoT application. The workload is executed with the latest request distribution patterns. This workload is executed against the small database scaling size with a data set of 250GB and against the medium scaling size with a data set of 500GB.

The following summary sections capture key insights as to how MongoDB and ScyllaDB compare across different workloads and database cluster sizes. Additional details on each type of testing performed, as well as detailed results for the caching workload benchmarks, are provided in the subsequent section. A detailed description of results for *all* workloads and configurations is provided on the benchANT site. Additionally, all raw results are publicly available on GitHub.

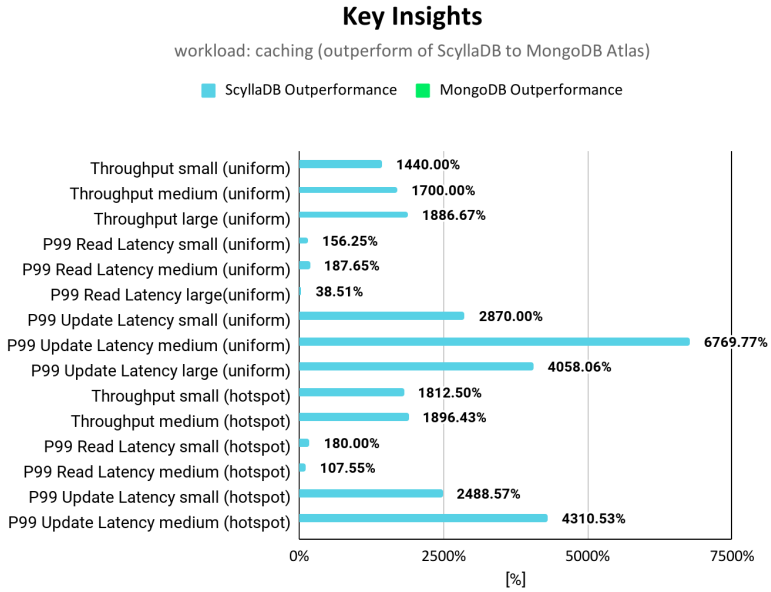
PERFORMANCE COMPARISON SUMMARY

For the **social workload**, ScyllaDB outperforms MongoDB with higher throughput and lower latency for all measured configurations of the social workload.analysis:



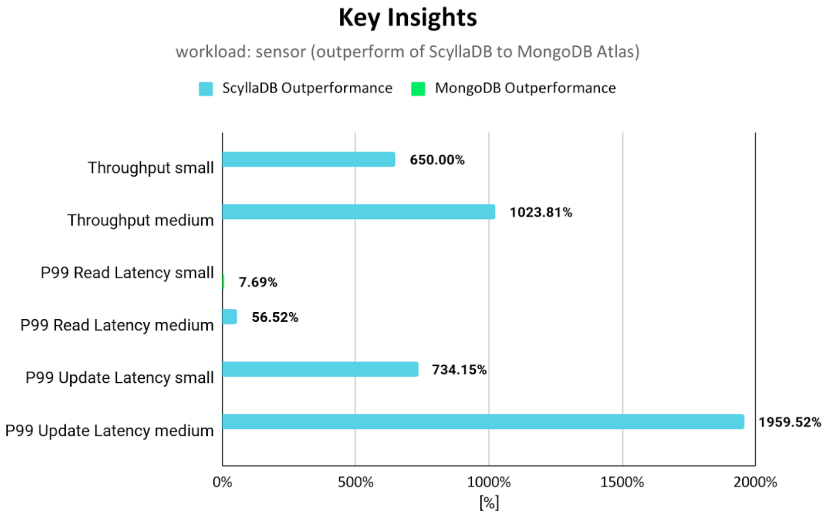
- ScyllaDB provides up to 12 times higher throughput
- ScyllaDB provides significantly lower (down to 47 times) update latencies compared to MongoDB
- ScyllaDB provides lower read latencies, down to 5 times

For the **caching workload**, ScyllaDB outperforms MongoDB with higher throughput and lower latency for all measured configurations of the caching workload.



- Even a small 3-node ScyllaDB cluster performs better than a large 18-node MongoDB cluster
- ScyllaDB provides constantly higher throughput that increases with growing data sizes to up to 20 times
- ScyllaDB provides significantly better update latencies (down to 68 times) compared to MongoDB
- ScyllaDB read latencies are also lower for all scaling sizes and request distributions, down to 2.8 times

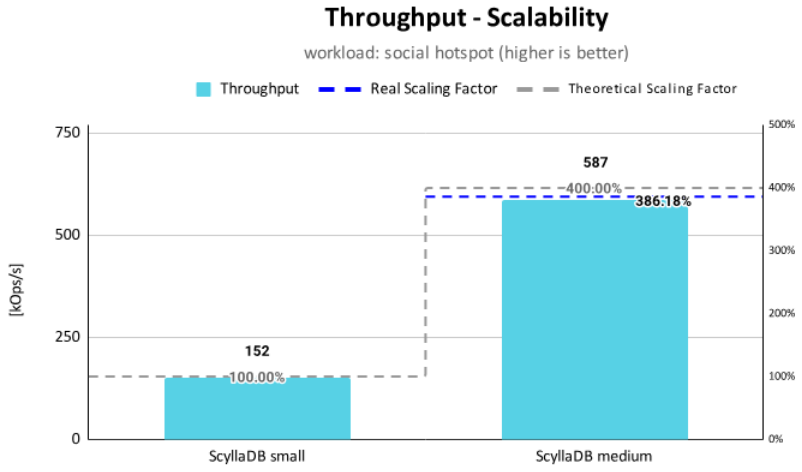
For the **sensor workload**, ScyllaDB outperforms MongoDB with higher throughput and lower latency results for the sensor workload except for the read latency in the small scaling size.

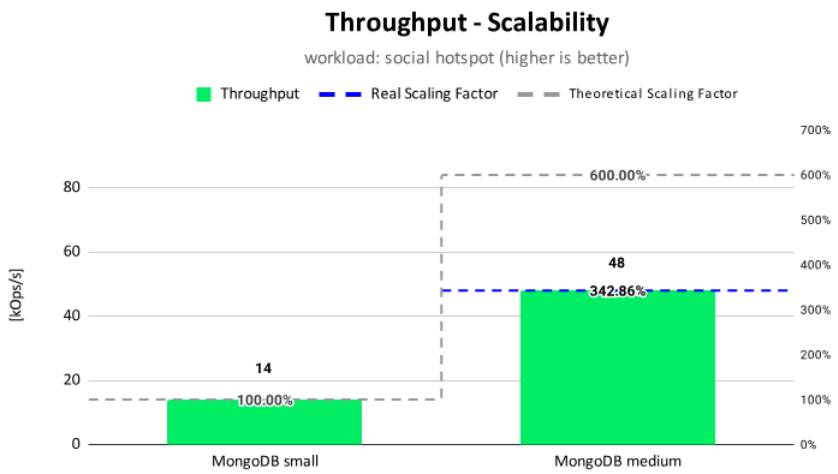


- ScyllaDB provides constantly higher throughput that increases with growing data sizes up to 19 times
- ScyllaDB provides lower (down to 20 times) update latency results compared to MongoDB
- MongoDB provides lower read latency for the small scaling size, but ScyllaDB provides lower read latencies for the medium scaling size

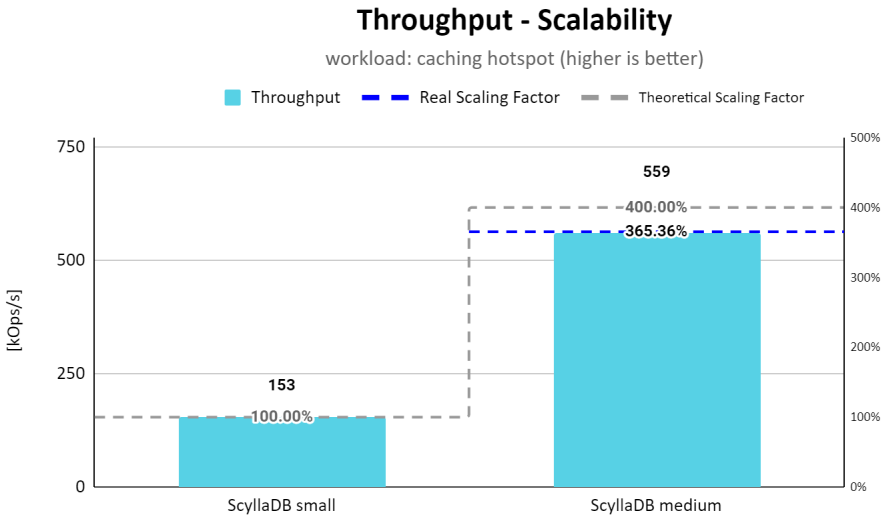
SCALABILITY COMPARISON SUMMARY

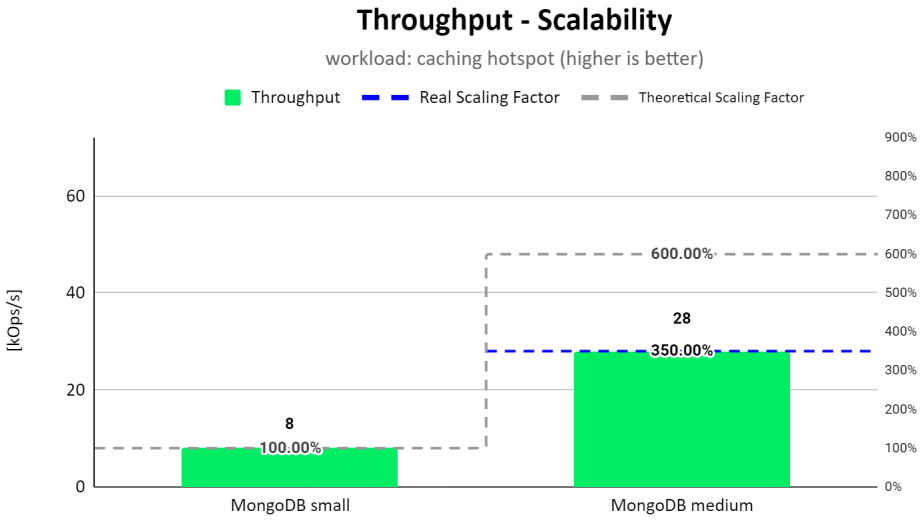
For the **social workload**, ScyllaDB achieves near linear scalability by achieving a throughput scalability of 386% (of the theoretically possible 400%). MongoDB achieves a scaling factor of 420% (of the theoretically possible 600%) for the uniform distribution and 342% (of the theoretically possible 600%) for the hotspot distribution.



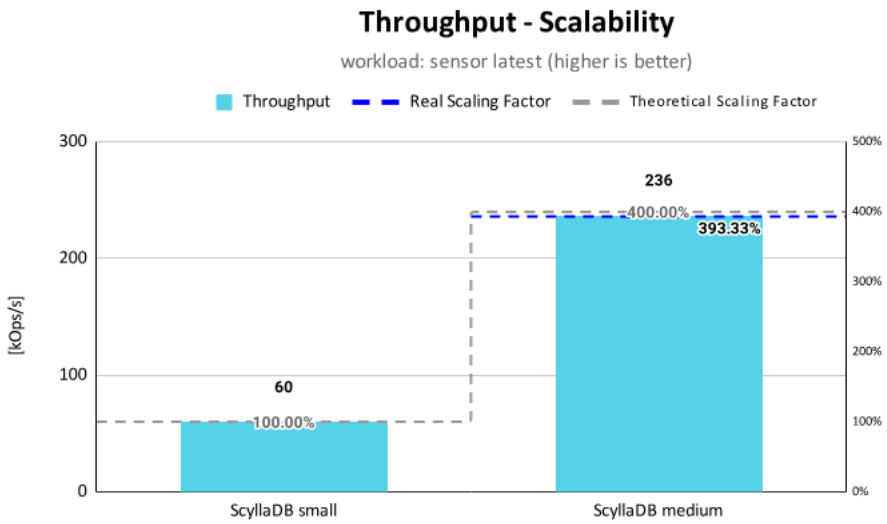


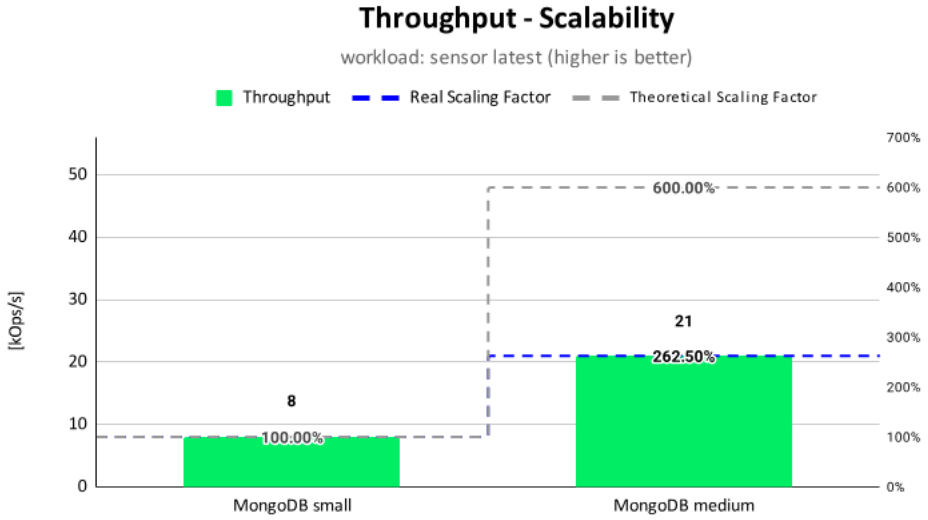
For the **caching workload**, ScyllaDB achieves near linear scalability across the tests. MongoDB achieves 340% of the theoretically possible 600% and 900% of the theoretically possible 2400%.





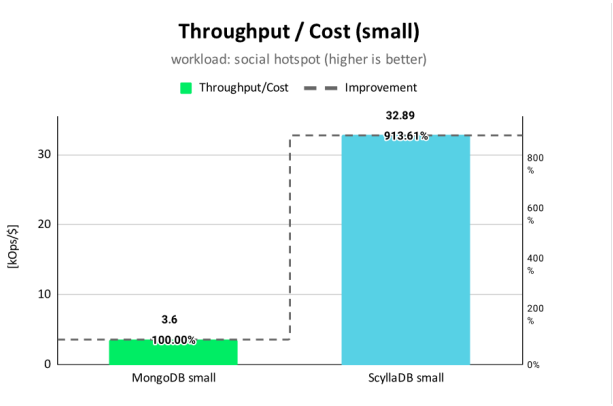
For the **sensor workload**, ScyllaDB is able to nearly achieve linear scalability with a throughput scalability of 393% of the theoretically possible 400%. MongoDB achieves a throughput scalability factor of 262% out of the theoretically possible 600%.

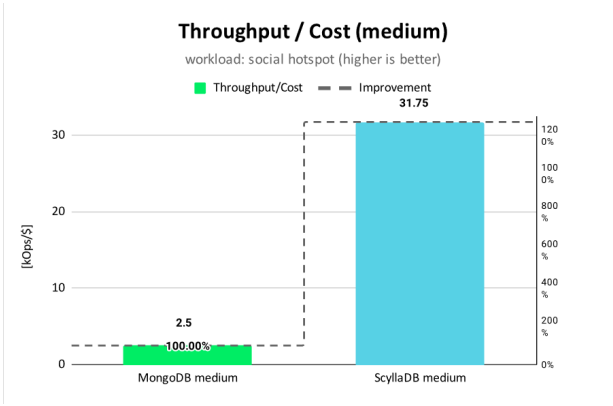




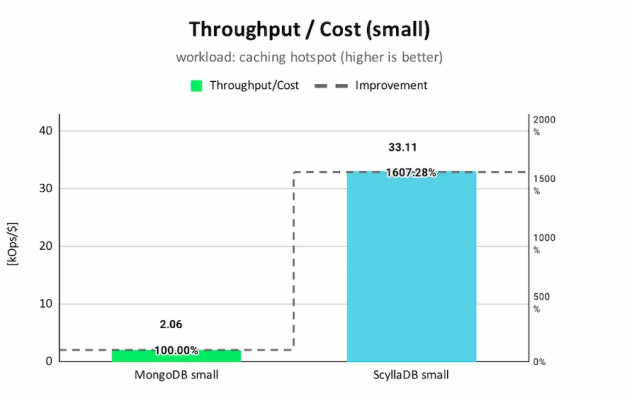
PRICE-PERFORMANCE RESULTS SUMMARY

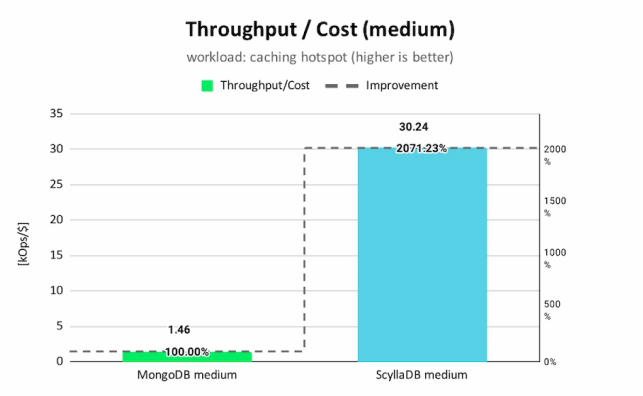
For the **social workload**, ScyllaDB provides five times more operations/$ compared to MongoDB Atlas for the small scaling size and 5.7 times more operations/$ for the medium scaling size. For the hotspot distribution, ScyllaDB provides 9 times more operations/$ for the small scaling size and 12.7 times more for the medium scaling size.



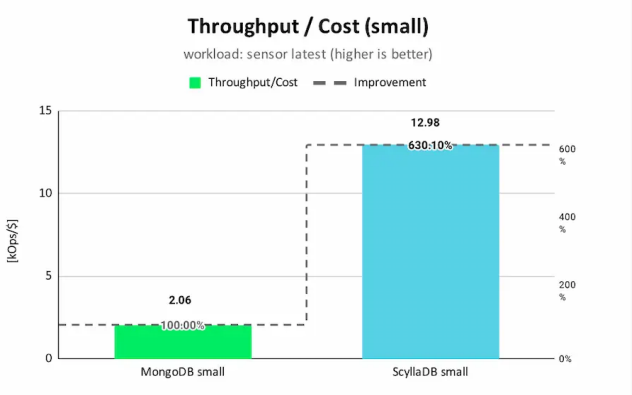


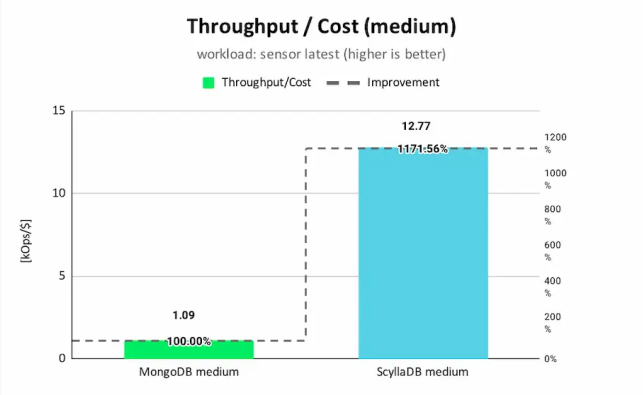
For the **caching workload**, ScyllaDB provides 12-16 times more operations/$ compared to MongoDB Atlas for the small scaling size and 18-20 times more operations/$ for the scaling sizes medium and large.





For the **sensor workload**, ScyllaDB provides 6-11 times more operations/$ compared to MongoDB Atlas. In both the caching and sensor workloads, MongoDB is able to scale the throughput with growing instance/cluster sizes, but the preserved operations/$ are decreasing.





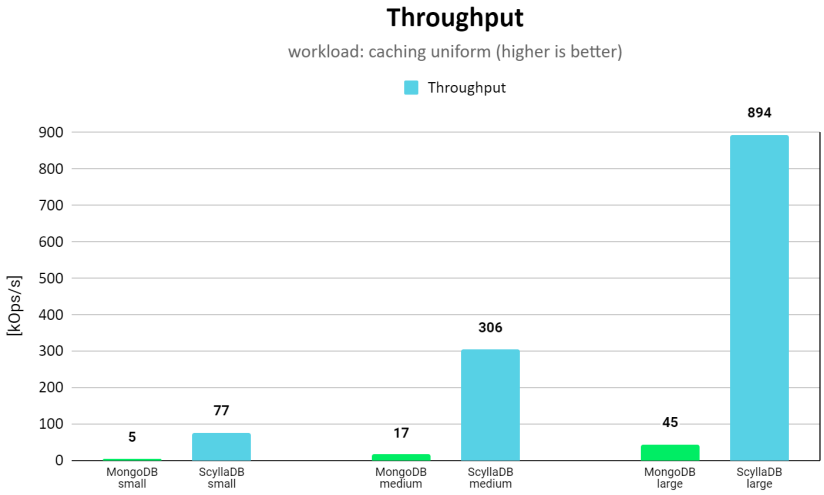
DEEP DIVE: CACHING WORKLOAD RESULTS

As stated earlier, the caching workload is based on the YCSB Workload A and creates a read-update workload, with 50% read operations and 50% update operations. In addition to the regular benchmark runtime of 30 minutes, a long-running benchmark over 12 hours is executed.

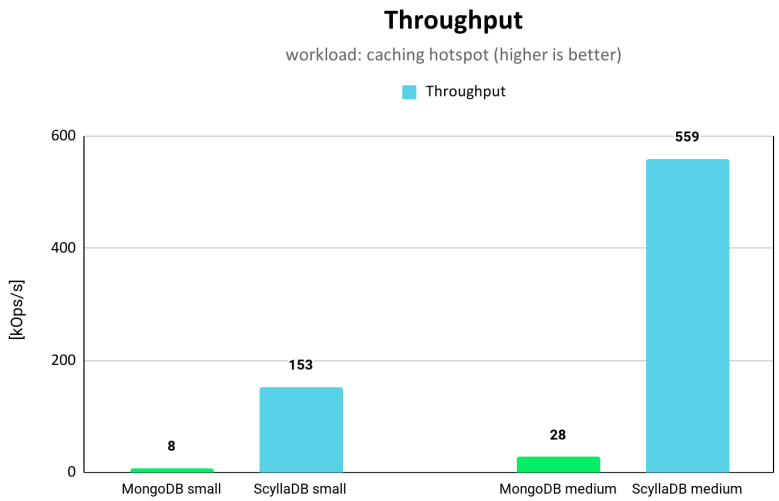
THROUGHPUT RESULTS

The throughput results for the caching workload with the uniform request distribution show that the small ScyllaDB cluster is able to serve 77 kOps/s with a cluster utilization of ~87% while the small MongoDB serves only 5 kOps/s under a comparable cluster utilization of 80-90%. For the medium cluster sizes, ScyllaDB achieves an average throughput of 306 kOps/s by ~89% cluster utilization and MongoDB 17 kOps/s. For the large cluster size, ScyllaDB achieves 894 kOps/s against 45 kOps/s of MongoDB.

Note that client side errors occurred when inserting the initial 10TB on MongoDB large; as a result, only 5TB of the specified 10TB were inserted. However, this does not affect the results of the caching workload because the applied YCSB version only operates on the key range 1 - 2.147.483.647 (INTEGER_MAX_VALUE); for more details, see the complete report on the benchANT site. This fact leads to an advantage for MongoDB because MongoDB's cache had only to deal with 2,100,000,000 accessed records (i.e. 2.1TB) while ScyllaDB's cache had to deal with the full 10,000,000,000 records (i.e. 10TB).



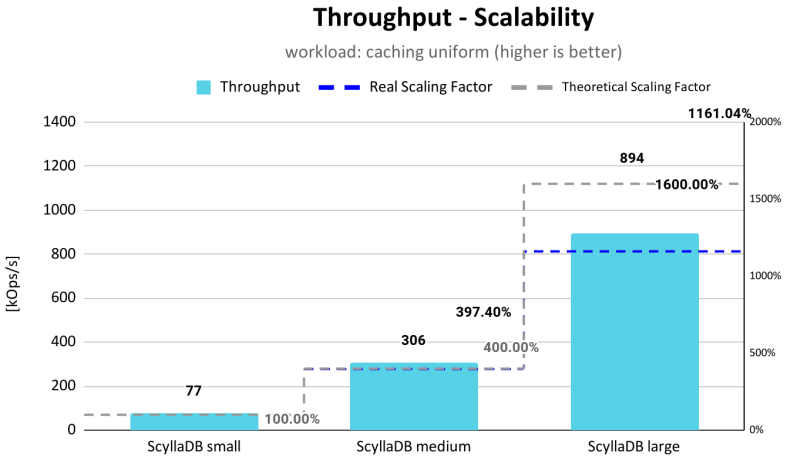
The caching workload with the hotspot distribution is only executed against the small and medium scaling size. The throughput results for the hotspot request distribution show a similar trend, but with higher throughput numbers since the data is mostly read from the cache. The small ScyllaDB cluster serves 153 kOps/s while the small MongoDB only serves 8 kOps/s. For the medium cluster sizes, ScyllaDB achieves an average throughput of 559 kOps/s and MongoDB achieves 28 kOps/s.



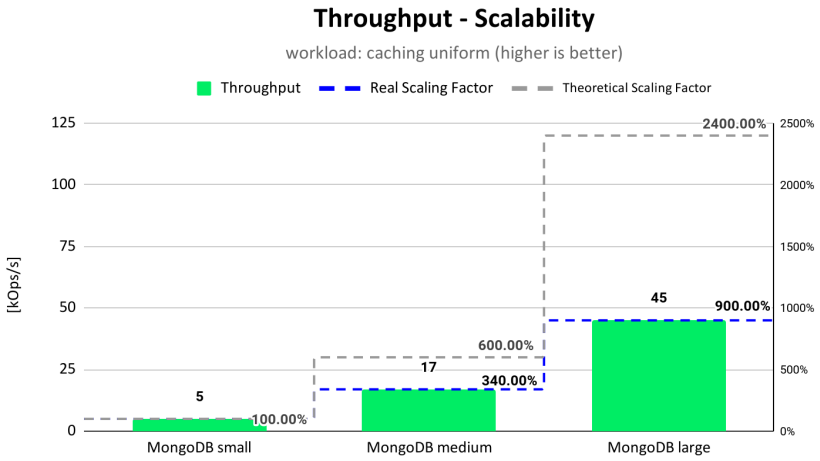
SCALABILITY RESULTS

The throughput results allow us to compare the theoretical throughput scalability with the actually achieved scalability. For ScyllaDB, the maximum theoretical scaling factor for throughput for the uniform distribution is 1600% when scaling from small to large. For MongoDB, the theoretical maximal throughput scaling factor is 2400% when scaling from small to large.

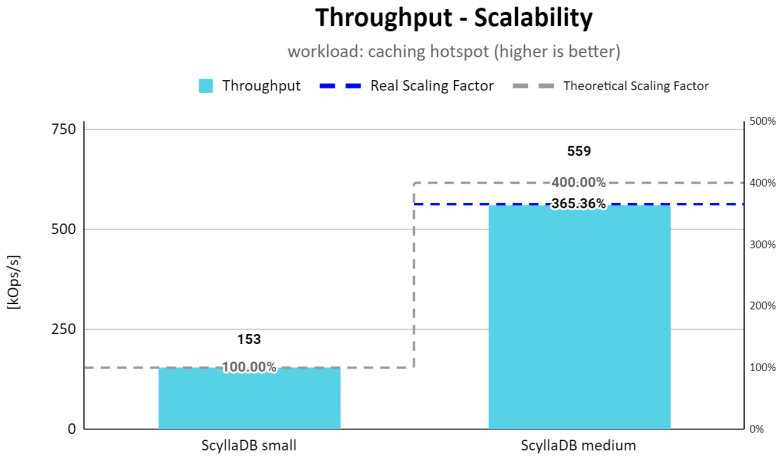
The ScyllaDB scalability results show that scaling from small to medium is very close to achieving linear scalability by achieving a throughput scalability of 397% of the theoretically possible 400%. Considering the maximal scaling factor from small to large, ScyllaDB achieves 1161% of the theoretical 1600%.



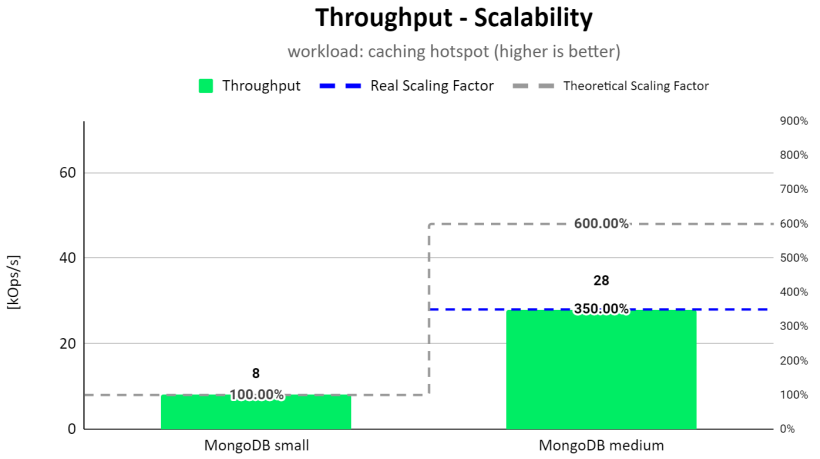
The MongoDB scalability results for the uniform distribution show that MongoDB scaled from small to medium achieves a throughput scalability of 340% of the theoretical 600%. Considering the maximal scaling factor from small to large, MongoDB achieves only 900% of the theoretically possible 2400%.



For the hotspot distribution, the small and medium cluster sizes are benchmarked. ScyllaDB achieves a throughput scalability of 365% of the theoretical 400%.



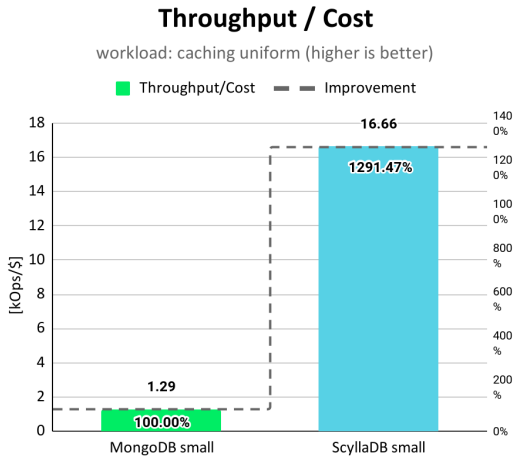
MongoDB achieves a throughput scalability of 350% of the theoretical 600%.

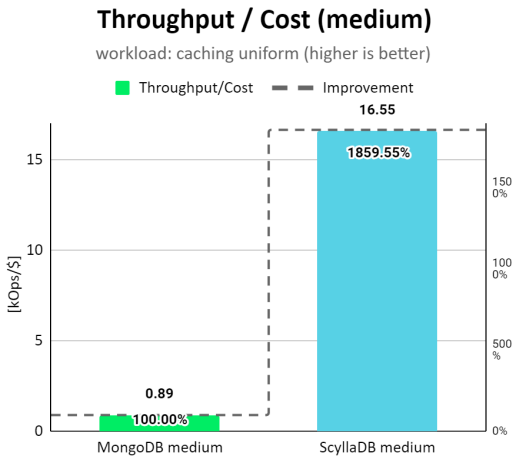


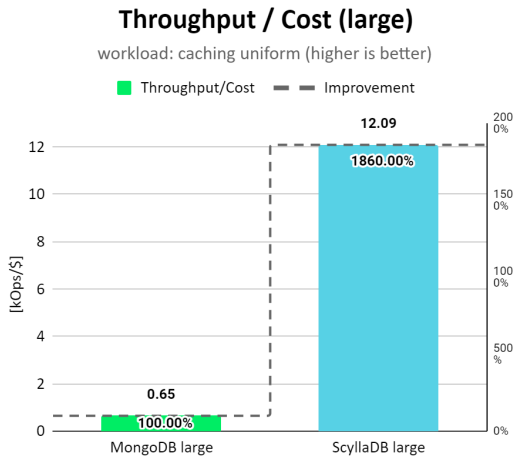
THROUGHPUT PER COST RESULTS

In order to compare the costs/month in relation to the provided throughput, we take the MongoDB Atlas throughput/$ as baseline (i.e. 100%) and compare it with the provided ScyllaDB Cloud throughput/$.

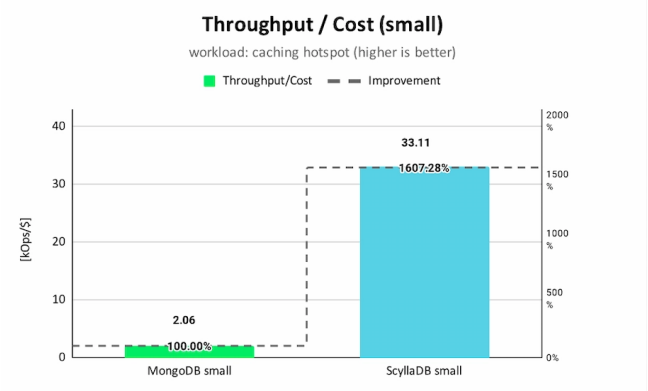
The results for the uniform distribution show that ScyllaDB provides 12 times more operations/$ compared to MongoDB Atlas for the small scaling size and 18 times more operations/$ for the scaling sizes medium and large.

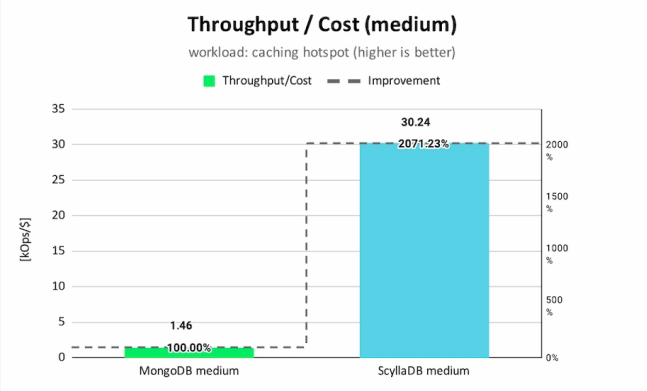






For the hotspot distribution, the results show a similar trend where ScyllaDB provides 16 times more operations/$ for the small scaling size and 20 times for the medium scaling size.

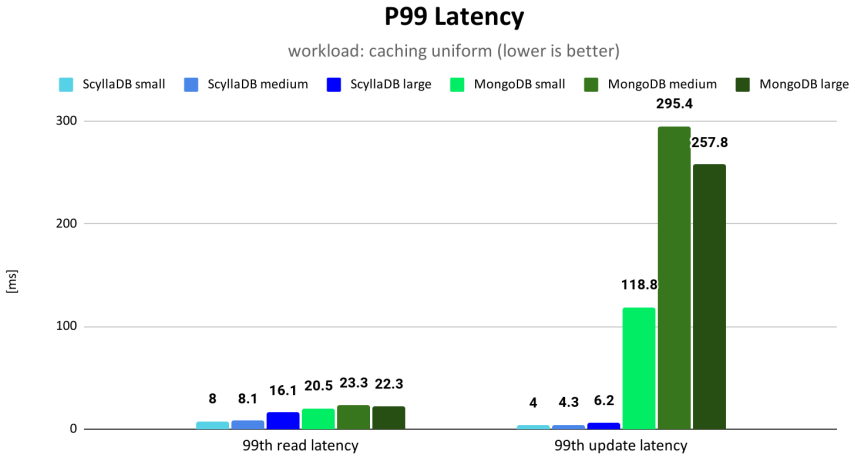




LATENCY RESULTS

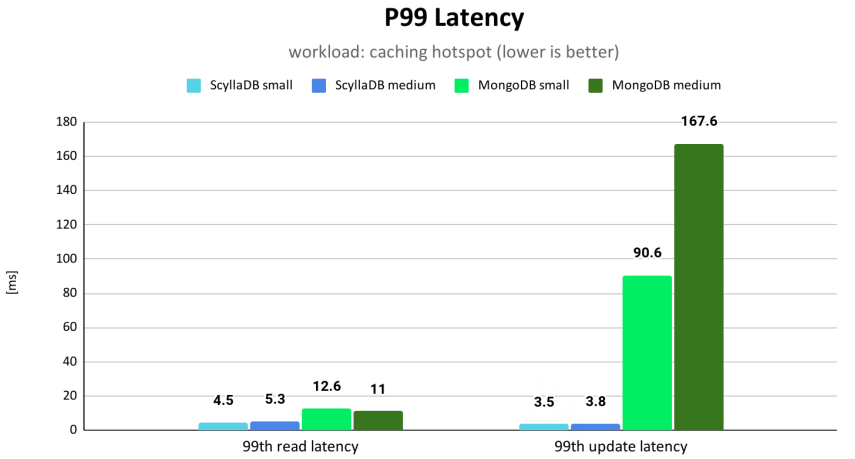
The P99 latency results for the uniform distribution show that ScyllaDB and MongoDB provide stable P99 read latencies. Yet, the values for ScyllaDB are constantly lower compared to the MongoDB latencies. An additional insight is that the ScyllaDB read latency doubles from medium to large (from 8.1 to 16.1 ms). The MongoDB latency decreases by 1 millisecond (from 23.3 to 22.3 ms), but still does not match the ScyllaDB latency.

For the update latencies, the results show a similar trend as for the social workload where ScyllaDB provides stable and low update latencies while MongoDB provides up to 73 times higher update latencies.



For the hotspot distribution, the results show a similar trend as for the uniform distribution. Both databases provide stable read latencies for the small and medium scaling size with ScyllaDB providing the lower latencies.

For updates, the ScyllaDB latencies are stable across the scaling sizes and slightly lower than for the uniform distribution. Compared to ScyllaDB, the MongoDB update latencies are 25 times higher for the small scaling size and 44 times higher for the medium scaling size respectively.

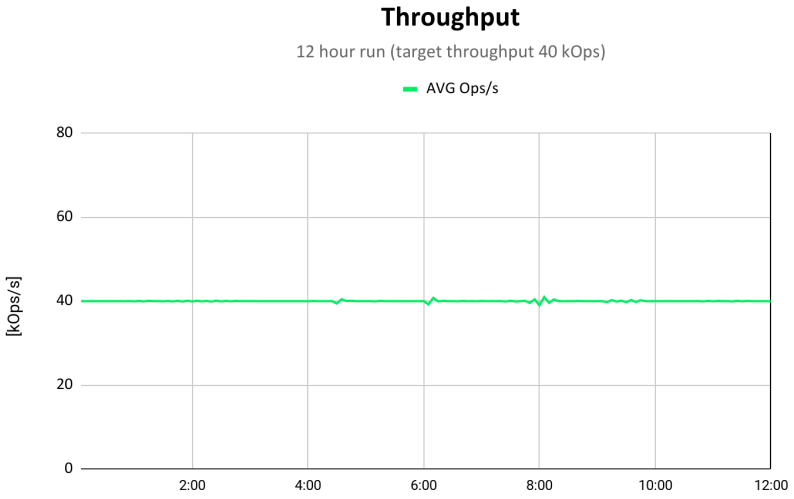


TECHNICAL NUGGET A - 12 HOUR BENCHMARK

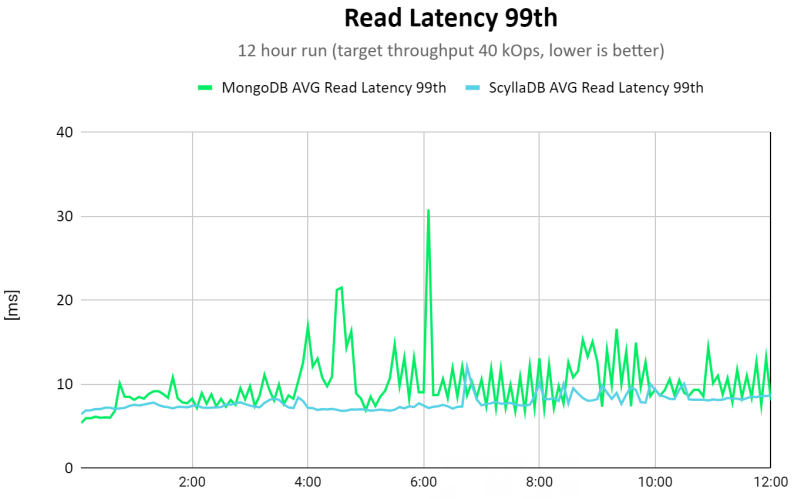
In addition to the default 30 minute benchmark run, we also select the scaling size large with the uniform distribution for a long-running benchmark of 12 hours.

For MongoDB, we select the determined 8 YCSB instances with 100 threads per YCSB instance and run the caching workload in uniform distribution for 12 hours with a target throughput of 40 kOps/s.

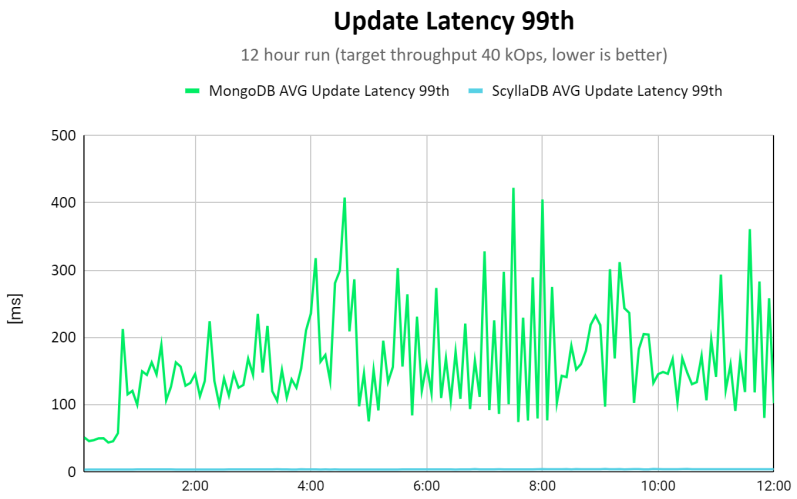
The throughput results show that MongoDB provides the 40 kOps/s constantly over time as expected.



The P99 read latencies over the 12 hours show some peaks in the latencies that reach 20ms and 30ms and an increase of spikes after 4 hours runtime. On average, the P99 read latency for the 12h run is 8.7 ms; for the regular 30 minutes run, it is 5.7 ms.

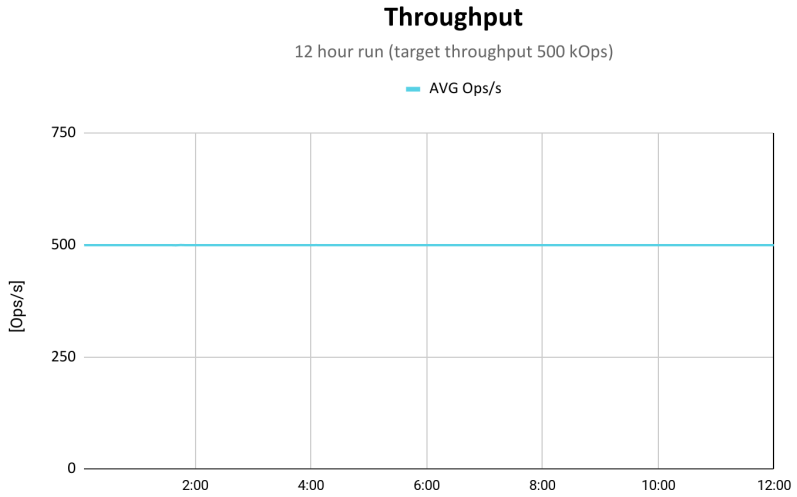


The P99 update latencies over the 12 hours show a spiky pattern over the entire 12 hours with peak latencies of 400 ms. On average, the P99 update latency for the 12h run is 163.8 ms while for the regular 30 minutes run it is 35.7 ms.

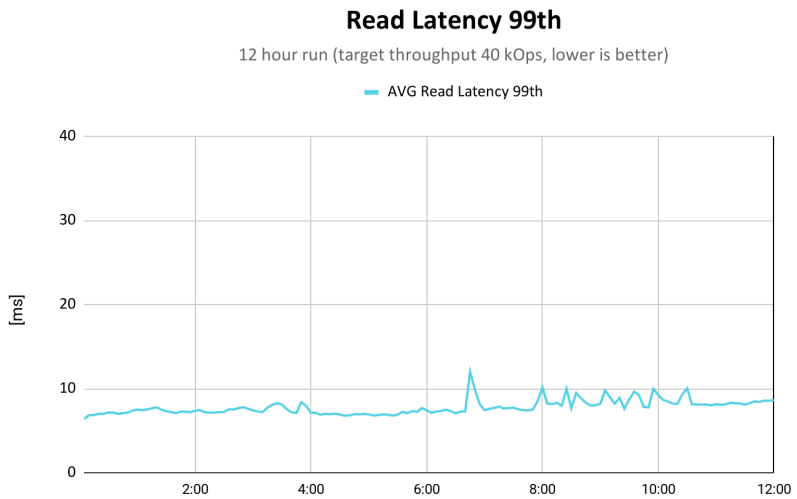


For ScyllaDB, we select the determined 16 YCSB instances with 200 threads per YCSB instance and run the caching workload in uniform distribution for 12 hours with a target throughput of 500 kOps/s.

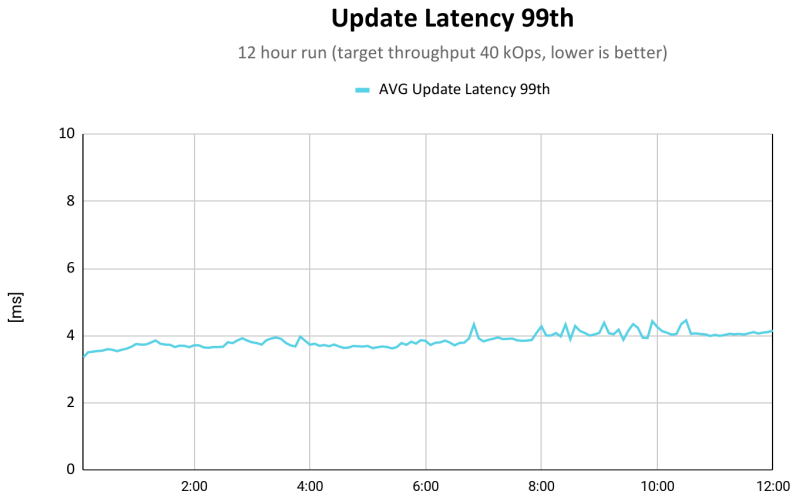
The throughput results show that ScyllaDB provides the 500 kOps/s constantly over time as expected.

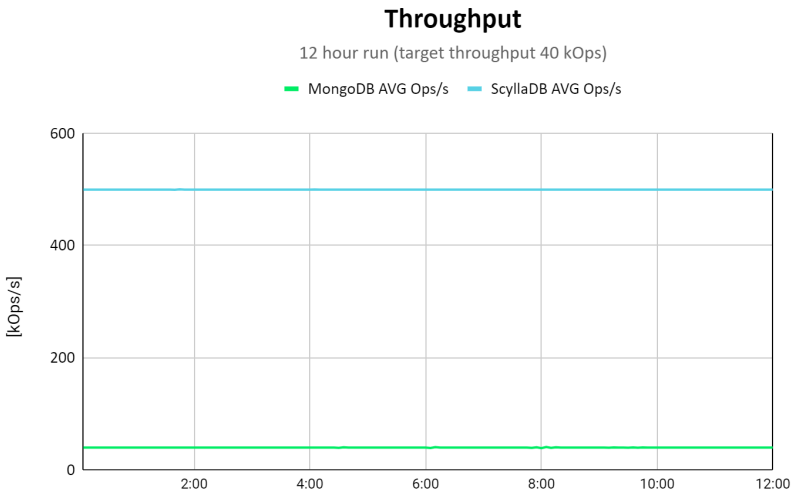


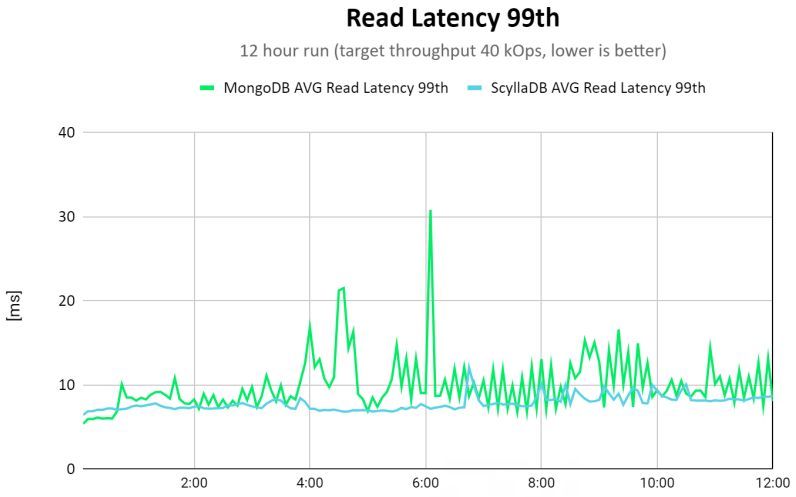
The P99 read latencies over the 12 hours stay constantly below 10 ms except for one peak of 12 ms. On average, the P99 read latency for the 12h run is 7.8 ms.

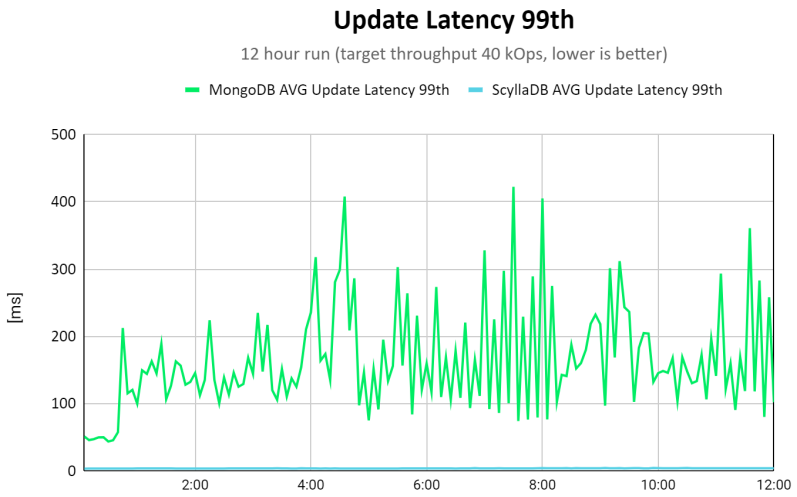


The P99 update latencies over the 12 hours show a stable pattern over the entire 12 hours with an average P99 latency of 3.9 ms.





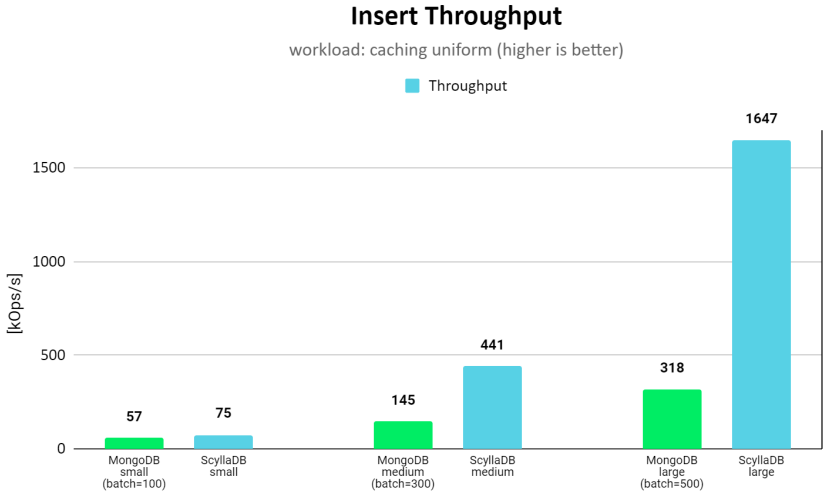




TECHNICAL NUGGET B - INSERT PERFORMANCE

In addition to the three defined workloads, we also measured the plain insert performance for the small scaling size (500 GB), medium scaling size (1 TB) and large scaling size (10 TB) into MongoDB and ScyllaDB. It needs to be emphasized that batch inserts were enabled for MongoDB but not for ScyllaDB (since YCSB does not support it for ScyllaDB).

The following results show that for the small scaling size, the achieved insert throughput is on a comparable level. However, for the larger data sets of the medium and large scaling sizes, ScyllaDB achieves a 3 times higher insert throughput for the medium size benchmark. For the large-scale benchmark, MongoDB was not able to fully ingest the full 10TB of data due to client side errors, resulting in only 5TB inserted data. Yet, ScyllaDB outperforms MongoDB by a factor of 5.



CONCLUSION

The complete benchmarking study comprises 133 performance and scalability measurements that compare MongoDB against ScyllaDB. The results show that ScyllaDB outperforms MongoDB for 132 of the 133 measurements.

For all of the applied workloads, namely caching, social and sensor, ScyllaDB provides higher throughput (up to 20 times) and better throughput scalability results compared to MongoDB. Regarding the latency results, ScyllaDB achieves P99 latencies below 10 ms for insert, read and update operations for almost all scenarios. In contrast, MongoDB only achieves P99 latencies below 10 ms for certain read operations while the insert and update latencies are up 68 times higher compared to ScyllaDB. These results validate the claim that ScyllaDB's distributed architecture is able to provide predictable performance at scale (as explained in the benchANT technical comparison of MongoDB vs ScyllaDB.

The scalability results show that both database technologies scale horizontally with growing workloads. However, ScyllaDB achieves nearly linear scalability while MongoDB shows a less efficient horizontal scalability. The ScyllaDB results were to a certain degree expected based on its multi-primary distributed architecture while a near linear scalability is still an outstanding result. Also for MongoDB, the less efficient scalability results are expected due to the different distributed architecture (as explained in the benchANT technical comparison of MongoDB vs ScyllaDB).

When it comes to price/performance, the results show a clear advantage for ScyllaDB with up to 19 times better price/performance ratio depending on the workload and data set size. In consequence, achieving comparable performance to ScyllaDB would require a significantly larger and more expensive MongoDB Atlas cluster.

In summary, this benchmarking study shows that ScyllaDB provides a great solution for applications that operate on data sets in the terabytes range and that require high throughput (e.g., over 50K OPS) and predictable low latency for read and write operations. While this study does not consider the performance impact of advanced data models (e.g. time series or vectors) or complex operation types (e.g.aggregates or scans) which are subject to future benchmark studies. But also for these aspects, the current results show that carrying out an in-depth benchmark before selecting a database technology will help to select a database that significantly lowers costs and prevents future performance problems.

APPENDIX: BENCHMARK SETUP, PROCESS, AND LIMITS

DBAAS SETUP

As explained in our technical comparison, MongoDB and ScyllaDB follow different distributed architecture approaches. Consequently, a fair comparison can only be achieved by selecting comparable cluster sizes in terms of costs/month or total compute power. Our comparison selects comparably priced cluster sizes with comparable instance types, having the goal to compare the provided performance (throughput and latency) as well as scalability for three cluster sizes under growing workloads.

The following table describes the selected database cluster sizes to be compared and classified into the scaling sizes small, medium and large. All benchmarks are run on AWS in the us-west region and the prices are based on the us-west-1 (N. California) region, which means that the DBaaS instances and the VMs running the benchmark instances are deployed in the same region. VPC peering is not activated for MongoDB or ScyllaDB. For MongoDB all benchmarks were run again version 6.0, for ScyllaDB against 2022.2. The period in which the benchmarks were carried out was March to June 2023.

DBaaS Scaling	DBaaS	Cluster Type	Version	Instance Type	Instance Specs	Total Data Nodes	Storage Capacity	Monthly Costs
small	MongoDB Atlas	replica set	6.0	M60_NVME	8 vCPU / 61 GB RAM	3	1.6 TB	$3,880
	ScyllaDB Cloud	N/A	2022.2.0	i4i.2xlarge	8 vCPU / 64 GB RAM	3	1.3 TB	$4,620
medium	MongoDB Atlas	sharded	6.0	M80_NVME	16 vCPU / 122 GB RAM	9	4.8 TB	$19,180
	ScyllaDB Cloud	N/A	2022.2.0	i4i.4xlarge	16 vCPU / 128 GB RAM	6	5.2 TB	$17,870
large	MongoDB Atlas	sharded	6.0	M200_NVME	32 vCPU / 244 GB RAM	18	18.6 TB	$69,120
	ScyllaDB Cloud	N/A	2022.2.0	i4i.8xlarge	32 vCPU / 256 GB RAM	12	20.9 TB	$73,932

WORKLOAD SETUP

In order to simulate realistic workloads, we use YCSB in the latest available version 0.18.0-SNAPSHOT from the original GitHub repository. Based on YCSB, we define three workloads that map to real world use cases. The key parameters of each workload are shown in the following table, and the full set of applied parameters is available in the GitHub repository.

Workload	Distribution	Reads [%]	Updates [%]	Inserts [%]	Data Size small/medium/large [TB]	Record Size [KB]
caching (YCSB A)	uniform & hotspot	50	50	0	0.5/1/10	1
social (YCSB B)	uniform & hotspot	95	5	0	0.5/1	1
sensor	latest	10	0	90	0.25/0.5	1

The caching and social workloads are executed with two different request patterns: The uniform request distribution simulates a close-to-zero cache hit ratio workload, and the hotspot distribution simulates an almost 100% cache hit workload.

All benchmarks are defined to run with a comparable client consistency. For MongoDB, the client consistency `writeConcern=majority` and readPreference=primary is applied. For ScyllaDB, `writeConsistency=QUORUM` and `readConsistency=QUORUM` are used. For more details on the client consistencies, read some of the technical nuggets below and also, feel free to read our detailed comparison of the two databases. In addition, we have also analyzed the performance impact of weaker consistency settings for the social and caching workload.

BENCHMARK PROCESS

In order to achieve a realistic utilization of the benchmarked database, each workload is scaled with the target database cluster (i.e. small, medium and large). For this, the data set size, the number of YCSB instances and the number of client threads is scaled accordingly to achieve 70-80% load with a stable throughput on the target database cluster.

Each benchmark run is carried out by the benchANT platform that handles the deployment of the required number of EC2 c5.4xlarge VMs with 16 vCores and 32GB RAM to run the YCSB instances, deployment of the DBaaS instances and orchestration of the LOAD and RUN phases of YCSB. After loading the data into the DB, the cluster is given a 15-minute stabilization time before starting the RUN phase executing the actual workload. In addition, we configured one workload to run for 12 hours to ensure the benchmark results are also valid for long running workloads.

For additional details on the benchmarking methodology (for example, how we identified the optimal throughput per database and scaling size), see the full report on the benchANT site.

LIMITATIONS OF THE COMPARISON

The goal of this benchmark comparison focuses on performance and scalability in relation to the costs. It is by no means an all-encompassing guide on when to select MongoDB or ScyllaDB. Yet, by combining the insights of the technical comparison with the results of this benchmark article, the reader gets a comprehensive decision base.

YCSB is a great benchmarking tool for relative performance comparisons. However, when it comes to measuring absolute latencies under steady throughput, it is affected by the coordinated omission problem. The latest release of the YCSB introduces an approach to address this problem. Yet, during the benchmark dry runs, it turned out that this feature is not working as expected (unrealistic high latencies were reported).

In the early (and also later) days of cloud computing, the performance of cloud environments was known to be volatile. This required experimenters to repeat experiments several times at different times of the day. Only then were they able to gather statistically meaningful results. Recent studies such as the one by Scheuner and Leitner show that this has changed. AWS provides particularly stable service quality. Due to that, all experiments presented here were executed once.

About Dr. Daniel Seybold and benchANT

Dr. Daniel Seybold is the co-founder and CTO of benchANT, a company that specializes in benchmarking and performance testing of databases. Daniel started his career as a researcher with a focus on distributed systems and databases. He has extensive experience in the field of database performance testing and has been working with NoSQL databases such as MongoDB, Cassandra and ScyllaDB for more than a decade. During his academic career he published over 20 papers on cloud and database performance related topics on renowned scientific conferences and completed his PhD with the thesis An automation-based approach for reproducible evaluations of distributed DBMS on elastic infrastructures.

These research results are the technical foundation of benchANT which pursues the goal of supporting organizations in selecting the right database for their use case. The company is a spin-off of the University of Ulm, Germany.

benchant.com