ALTOROS™

2023 NoSQL DBaaS Performance:

Couchbase Capella vs. Amazon DynamoDB

Using Yahoo! Cloud Serving Benchmark, this report compares the throughput and latency of the popular databases as a service (DBaaS) across four business scenarios and four different cluster configurations.

By **Ivan Shryma**, Data Engineer, Altoros

Q2 2023

Table of contents

1. Executive summary

NoSQL encompasses a wide variety of database technologies that were developed in response to a rise in the volume of data and the frequency with which information is stored, accessed, and updated. In contrast, relational databases were not designed to cope with scalability and agility challenges that modern applications require. Furthermore, relational databases cannot take advantage of the affordable storage and processing power available in today's cloud environments. Meanwhile, new-generation NoSQL solutions help to achieve the highest levels of performance and uptime for modern application workloads. Finally, teams are more regularly seeking Database-as-a-Service (DBaaS) options to avoid having to invest increasing amounts of time and money in cluster support, deployment, and maintenance.

This report compares the performance results of four NoSQL databases as a service: **Couchbase Capella** and **Amazon DynamoDB**. The goal of this report is to measure the relative performance in terms of latency and throughput that each database can achieve. The evaluation was conducted on four different cluster configurations—3, 6, 9, and 18 nodes—as well as under four different workloads.

The first workload performed *update-heavy* activity, involving 50% reads and 50% updates of the data. The second workload was *read-only*, with 100% read operations.The third workload performed a *short-range scan* that involved 95% scans and 5% updates, where short ranges of records were queried instead of individual ones. Finally, the fourth workload was a query with a single filtering option to which an offset and a limit were applied.

As a default tool for evaluation consistency, we utilized the Yahoo! Cloud Serving Benchmark (YCSB)—an open-source specification and program suite for evaluating retrieval and maintenance capabilities of computer programs.

2. A testing environment

2.1 YCSB instance configuration

To provide verifiable results, the benchmark was performed on easily obtained Amazon Elastic Compute Cloud (EC2) instances. The YCSB client was deployed to 10 compute-optimized large instances. Each client instance of YCSB produced 40 threads. This means the total load on the database was 400 threads for each test.

Table 2.1 *A description of the Amazon EC2 instance deployed to the YCSB client*

Family	Compute-optimized
Type	c4.2xlarge
vCPUs	8
Memory (GiB)	15
EBS-optimized available	Yes
Network performance	High
Platform	64-bit
Operating system	Ubuntu 18.04 LTS
AWS region	us-east-1

(continued in the next page)

Table 2.1 *A description of the Amazon EC2 instance deployed to the YCSB client (continued)*

2.2 Couchbase Capella cluster configuration

Couchbase Capella is a fully managed Database as a Service. It combines the features of a key–value store allowing operations on single documents. The database also acts as a schemaless document store to access the documents by querying through SQL++ (SQL for JSON).

The Capella Control Panel includes a cluster sizing page, offering customers multiple options to choose from—such as instance sizes, configurations, and quantities. Couchbase Capella can also be tuned to deploy specific services to a single or several nodes in the cluster. The vendor calls this feature “Multi-Dimensional Scaling.”

Each node was configured to run the Data, Index, and Query services. The Data service is the most fundamental of all Couchbase services, providing access to data in memory and on disk. The Index service supports the creation of primary and global secondary indexes on items stored within Couchbase. The Query service supports the querying of data by means of SQL and relies on both the Index and Data services. Figure 2.2 shows the architecture of an example Capella cluster.

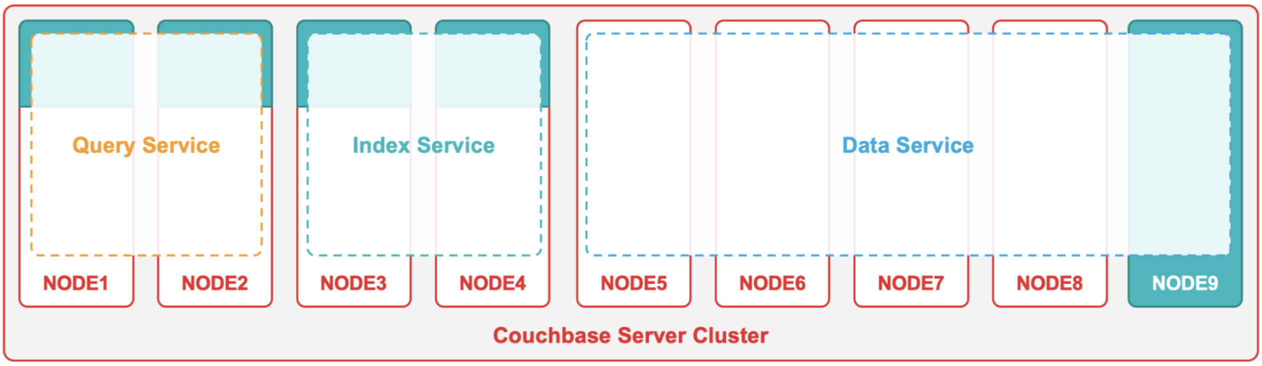

Figure 2.2 The architecture of a Couchbase Capella cluster (image credit)

After the cluster is deployed, data access should be configured by creating database credentials and granting the required access permissions. The test’s bucket was created with half of the available system memory allocated for it. In the report, we have used a new storage engine called Magma, which is designed to be highly performant for very large data sets that do not fit in memory.

The final step is to configure a list of allowed IPs on the control panel’s Connect tab, since Couchbase Capella allows clusters to connect to trusted IP addresses only.

Table 2.2 *Specification of a Couchbase Capella instance*

vCPUs	8
Memory (GB)	64
EBS storage (GB)	200
IOPS	5,700
AWS region	us-east-1

2.3 Amazon DynamoDB cluster configuration

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. All of the data is stored on solid-state drives (SSDs).

Amazon provides DynamoDB as a service. With this product, there is no need to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. The performance power of a cluster fully depends on the pricing model. It is hard to draw any architecture diagram, because DynamoDB initially was provided as a service.

With Amazon DynamoDB, users can configure read/write capacities for their tables. Users can also choose between two capacity modes for processing reads and writes:

- on-demand
- provisioned (default, free-tier eligible)

We chose the provisioned mode in order to specify the biggest number of reads and writes per second as individual settings. The read/write capacity is calculated against the cost. Because MongoDB had the highest operating costs of the tested vendors, we used those spend values as the capacity threshold for DynamoDB. The provisioned capacity can automatically scale in response to traffic changes.

For evaluation purposes, autoscaling was disabled to maintain parity with other databases and to limit costs. Unfortunately, under the provisioned mode, DynamoDB throws exceptions when read/write operations exceed the predetermined capacities. This resulted in failed operations in certain workloads. The failed operations may have been avoided if we had simply increased our investment in provisioned capacity to raise its ceiling. Nonetheless, this tells us that DynamoDB has issues with handling peak loads without autoscaling.

2.4 Operating costs

2.4.1 Couchbase Capella costs

The monthly billing report for running Couchbase Capella includes per instance–hour costs billed by the provider. Approximate monthly total for supporting a Capella cluster of specified configuration:

- 3 nodes amounted to around $2,642
- 6 nodes amounted to around $5,284
- 9 nodes amounted to around $7,926
- 18 nodes amounted to around $15,851

Note that charges in Couchbase Capella are billed in Couchbase Capella Credits.

2.4.2 Amazon DynamoDB costs

The pricing for Amazon DynamoDB under the provisioned mode is based on read/write capacities. For this report, there was no additional index or autoscaling. The read/write capacity was determined based on the monthly total costs for the MongoDB Atlas environment (got from previous report):

- 3 nodes amounted to around $5,026
- 6 nodes amounted to around $9,656
- 9 nodes amounted to around $14,292
- 18 nodes amounted to around $28,202

3. Workloads and tools

Database performance is defined by the speed at which a database processes basic operations. A basic operation is an action performed by a workload executor that drives multiple client threads. Each thread executes a sequential series of operations by making calls to a database interface layer both to load a database (the *load* phase) and to execute a workload (the *transaction* phase). The threads throttle the rate at which they generate requests, making it possible to directly control the load against the database. In addition, the threads measure latency, as well as the achieved throughput of their operations, and then report these measurements to the statistics module.

3.1 Workloads

The performance of each database was evaluated under the following workloads:

1) **Workload A.** Update heavily: 50% read and 50% update, request distribution is Zipfian.
2) **Workload C.** Read only: 100% read, request distribution is Zipfian.
3) **Workload E.** Scan short ranges: 95% scan and 5% update, request distribution is Uniform.

3.2 Tools

The YCSB client was used as a worker, consisting of the following components:

- a workload executor
- the YCSB client threads
- the extensions
- the statistics module
- the database connectors

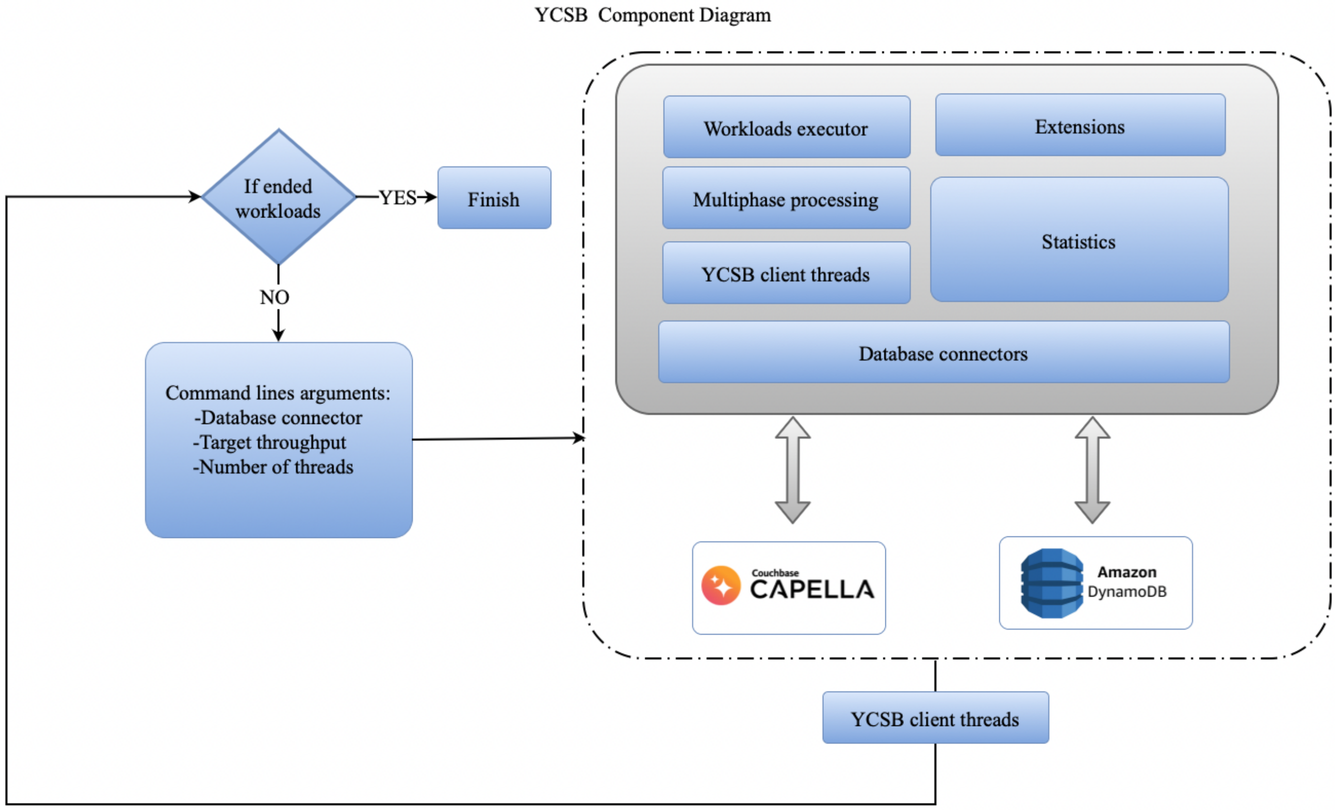

Figure 3.2.1 The components of the YCSB client

The workloads were tested under the following conditions:

- Data fits memory.
- Durability is false.
- Replication is set to “1,” signifying that just a single replica is available for each data set.

Workloads A, C, and E are standard workloads provided by YCSB. Default data models were used for these workloads.

4. YCSB benchmark results

4.1 Workload A: the update-heavy mode

4.1.1 Workload definition and model details

Workload A is an update-heavy workload that simulates typical actions of an e-commerce solution. This is a basic key–value workload. The scenario was executed with the following settings:

- The read/update ratio was 50%–50%.
- The Zipfian request distribution was used.
- The size of a data set was scaled in accordance with the cluster size: 25 million records (each 1 KB in size, consisting of 10 fields and a key) on a 3-node cluster, 50 million records on a 6-node cluster, 100 million records on a 9-node cluster, and 200 million records on a 18-node cluster.

Couchbase Capella stores data in buckets and collections, which are the logical groups of items—key–value pairs. vBuckets are physical partitions of the bucket data. By default, Capella creates a number of master vBuckets per bucket to store bucket data and evenly distribute vBuckets across all cluster nodes.

Querying with document keys is the most efficient method, since a query request is sent directly to a proper vBucket holding target documents. This approach does not require any index creation and is the fastest way to retrieve a document due to the key–value storage.

Amazon DynamoDB's read/write capacity for the workload was calculated through experiments. The chosen values have the best balance of read and write capacities based on cost. For each cluster, the following values were used.

- 3 nodes: 5,020 read and 9,390 write capacities
- 6 nodes: 6,600 read and 18,650 write capacities
- 9 nodes: 9,900 read and 27,575 write capacities
- 18 nodes: 25,000 read and 53,315 write capacities

4.1.2 Query

The following queries were used to perform Workload A.

Table 4.1.2 *Evaluated queries for Workload A*

	Read	Update
Couchbase Key–Value API	`collection.get(id, $2, getOptions().timeout(kvTimeout))`	`collection.upsert(id, content, upsertOptions().timeout(kvT`

		imeout).expiry(documentExpiry).durability(persistTo, replicateTo))

DynamoDB API	{
	    "TableName": "usertable",
	    "Key": {
	        "_id": "$1"
	    },
	    "ConsistentRead": "false"
	}
		{
		 "TableName": "usertable",
		 "Key": {_id = {S: $1}},
		 "AttributeUpdates": {
		   $2={Value: {S: $3} }
		   },
		 "Action": "PUT"
		}

4.1.3 Evaluation results

On each type of a cluster, Couchbase Capella significantly outperformed DynamoDB. On a 3-node cluster, it had a throughput of 232,050 ops/sec with a 2.67 ms latency. Couchbase Capella's performance improved all the way to an 18-node cluster, where it had a throughput of 423,580 ops/sec with less than a 1 ms latency.

As the cluster size increased, AWS DynamoDB also demonstrated better performance and showed the best performance on an 18-node cluster. DynamoDB reached 109,350 ops/sec with a 5.1 ms latency.

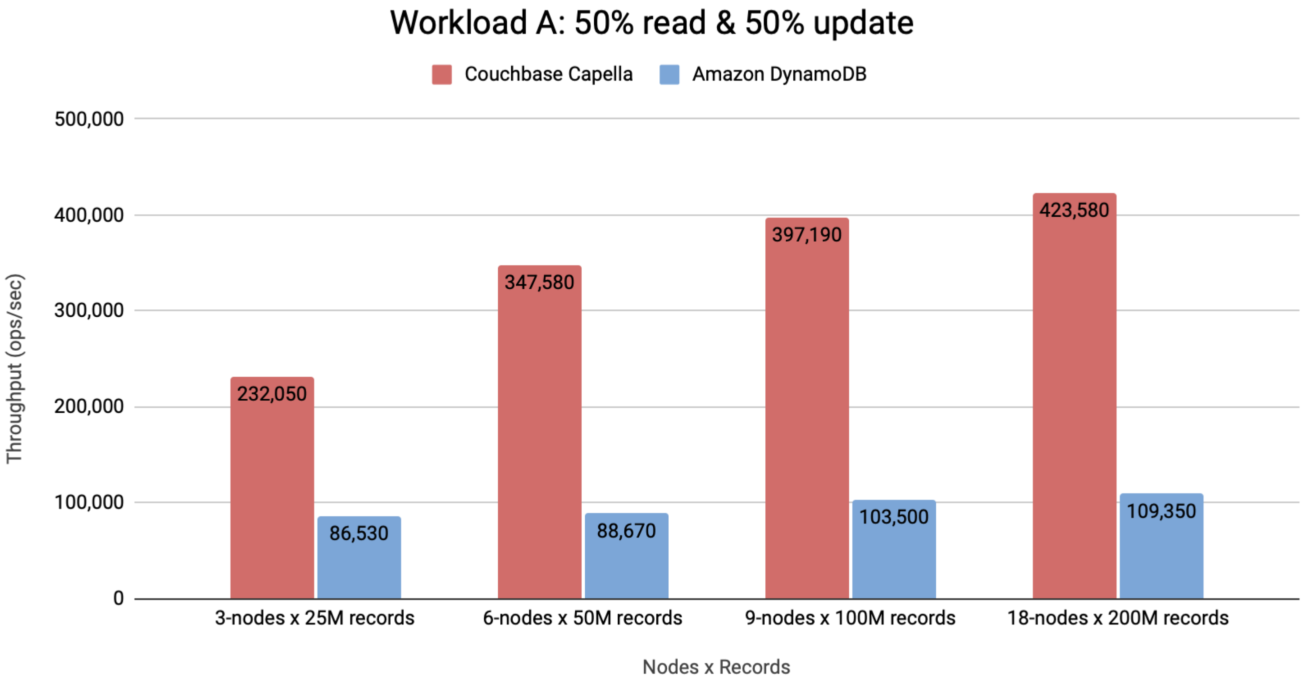

Figure 4.1.3.1 Throughput results under Workload A on 3-, 6-, 9-, and 18-node clusters

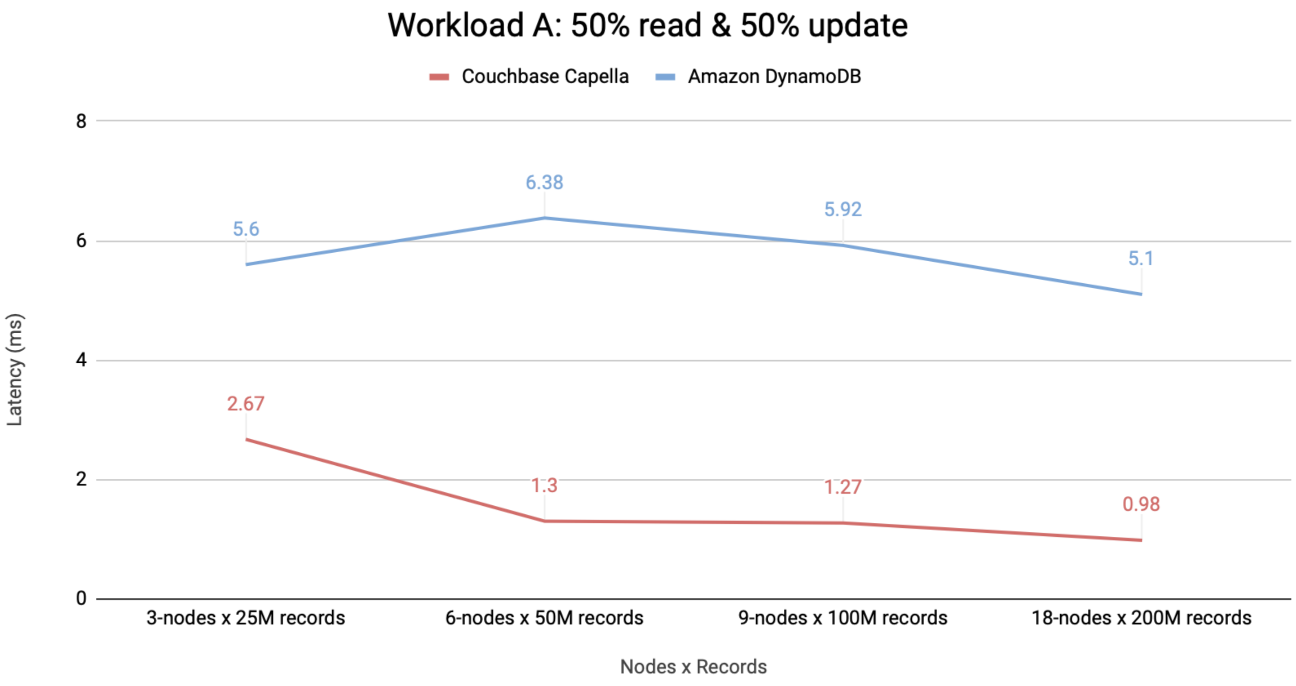

Figure 4.1.3.2 Latency results under Workload A on 3-, 6-, 9-, and 18-node clusters

Amazon DynamoDB produced unstable results due to a high number of failed operations. Across each type of cluster, Amazon DynamoDB had an average of 43–58% of failed operations, with only the 18-node cluster showing results with 25% of failed operations.

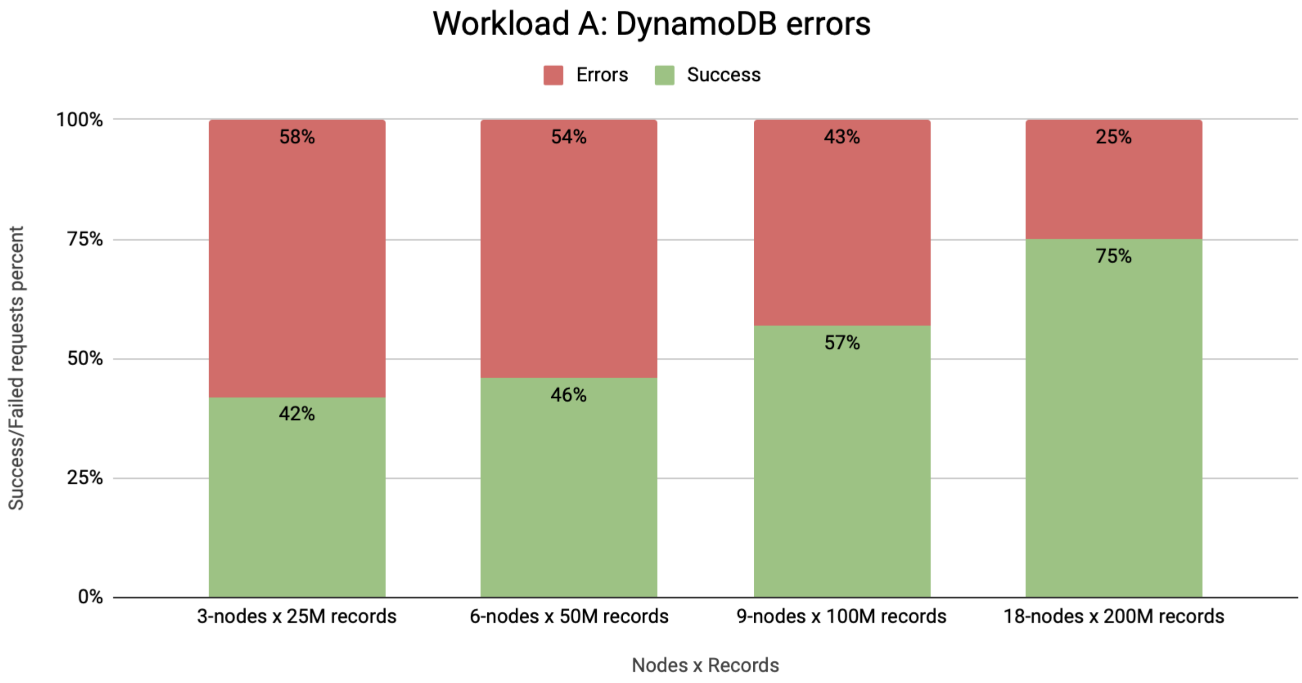

Figure 4.1.3.3 Results for DynamoDB under Workload A on 3-, 6-, 9-, and 18-node clusters

4.1.4 Summary

The throughput of each database grew constantly depending on the type of a cluster. Couchbase Capella achieved the throughput limit for each cluster type, but DynamoDB wasn't able to do it for read operations on an 18-node cluster. Couchbase Capella demonstrated high throughput growth and clearly outperformed Amazon DynamoDB on each type of a cluster.

Couchbase Capella stood out with a latency of about 1 ms on 6-, 9-, and 18-node clusters, and a latency of 2.67 ms on 3 nodes. Amazon DynamoDB had a latency of around 6 ms on every cluster type. Capella showed stable results without failed operations compared to Amazon DynamoDB.

4.2 Workload C: read-only

4.2.1 Workload definition and model details

Workload C is 100% read. The workload simulates user profile cache. The scenario was executed under the following settings:

- The read ratio was 100%.
- The Zipfian request distribution was used.
- The size of a data set was scaled in accordance with the cluster size: 25 million records (each 1 KB in size, consisting of 10 fields and a key) on a 3-node cluster, 50 million records on a 6-node cluster, 100 million records on a 9-node cluster, and 200 million records on a 18-node cluster.

4.2.2 Query

The following queries were used to perform *Workload C*.

Table 4.2.2 *Evaluated queries for Workload C*

	Read
Couchbase Key–Value API	`collection.get(id, $2, getOptions().timeout(kvTimeout))`

DynamoDB API	`{`
	`    "TableName": "usertable",`
	`    "Key": {`
	`            "_id": "$1"`
	`          },`
	`    "ConsistentRead": "false"`
	`}`

4.2.3 Evaluation results

In the workload, Couchbase Capella outperformed AWS DynamoDB. It showed growth of throughput before 18 nodes cluster, where happened drop.

Amazon DynamoDB was unable to reach the throughput limit, and the cluster had many unused “read units,” resulting in a performance that did not change with a larger cluster. Instead, performance slightly degraded due to the numerous additional partitions created by DynamoDB when increasing the number of “read units.” Thus, the results remained relatively stable with a throughput range of 141,080–149,900 “read units” and an average latency of 3.2 ms.

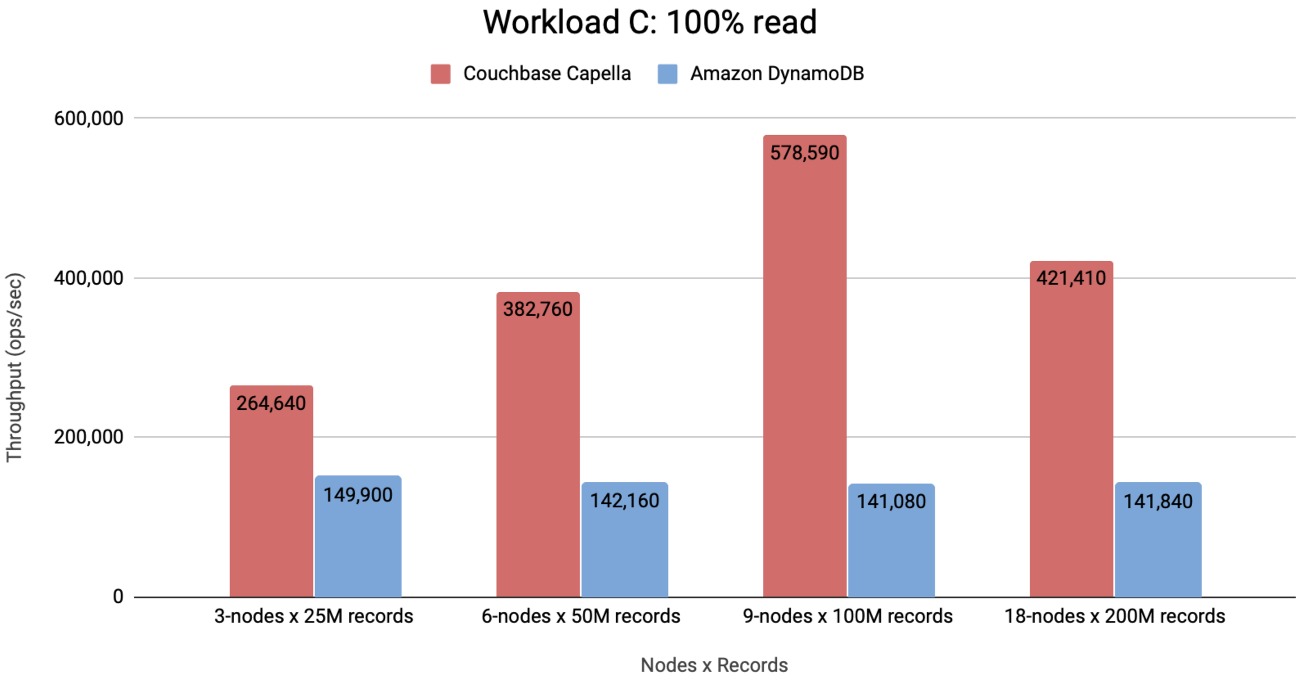

Figure 4.2.3.1 Throughput results under Workload C on 3-, 6-, 9-, and 18-node clusters

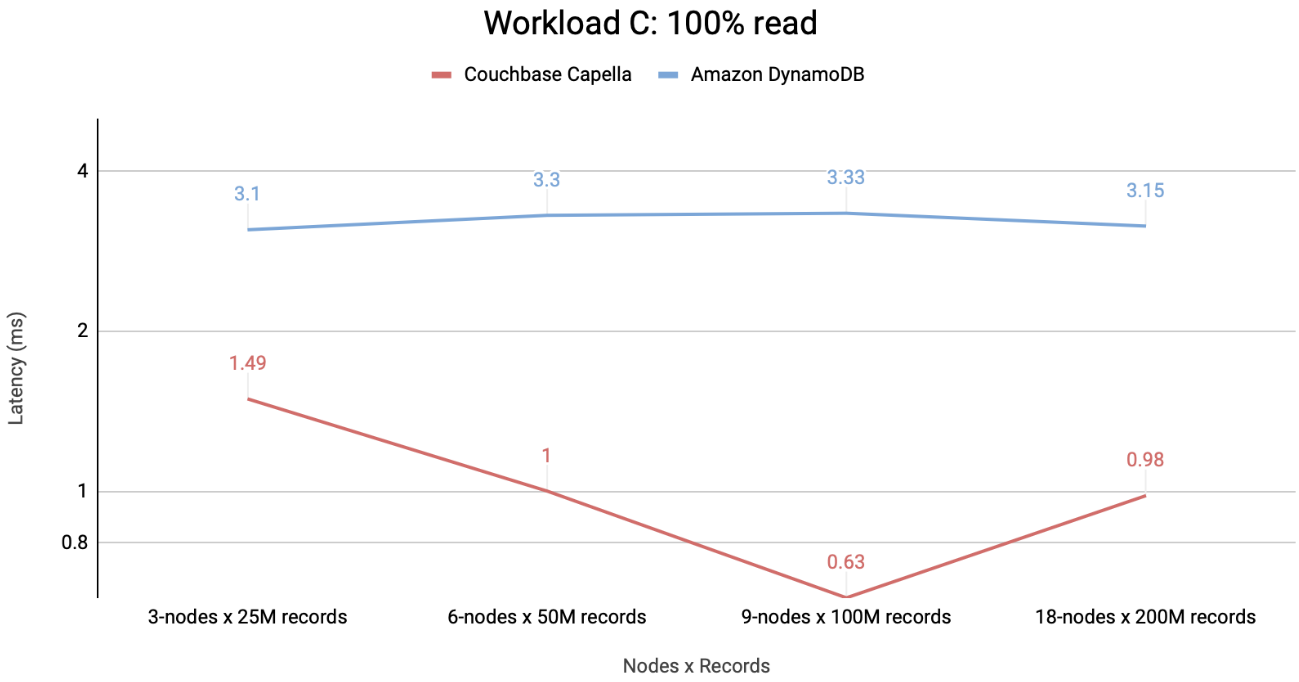

Figure 4.2.3.2 Latency results under Workload C on 3-, 6-, 9-, and 18-node clusters

4.2.4 Summary

Workload C, which consisted of simple read operations, produced interesting results for all the databases.

Couchbase Capella showed a steady and significant growth in performance up to the 18-node cluster, with a small drop in throughput and latency—from 578,590 ops/sec with a latency of 0.63 ms to 487,387 ops/sec with a latency of 0.98 ms. Nonetheless, this performance was still very good.

Amazon DynamoDB, unfortunately, hit its limit on 3 nodes and did not show significant changes with the other types of clusters.

4.3 Workload E: scanning short ranges

4.3.1 Workload definition and model details

Workload E is a short-range scan workload in which short ranges of records are queried instead of individual ones. This workload simulates threaded conversations, where each scan goes through the posts in a given thread (assuming the entries are clustered by ID). The scenario was executed under the following settings:

- The scan/update ratio was 95%–5%.
- The Zipfian request distribution was used.
- The size of a data set was scaled in accordance with the cluster size: 25 million records (each 1 KB in size, consisting of 10 fields and a key) on a 3-node cluster, 50 million records on a 6-node cluster, 100 million records on a 9-node cluster, and 200 million records on a 18-node cluster.
- The maximum scan length reached 100 records.
- Uniform was used as a scan length distribution.

In DynamoDB, the scan operation is required to use read capacity. There are no special tricks to speed up the scan operation. You can try using parallel scan, but read capacity will not change anyway. As long as read capacity is cheap, we were able to increase the default maximum count to 271,580 read operations. The capacities were chosen after a few experiments to get the best result. For each cluster, the following values were used:

- 3 nodes had 47,470 read and 900 write capacities.
- 6 nodes had 93,000 read and 1,360 write capacities.
- 9 nodes had 137,760 read and 2,000 write capacities.
- 18 nodes had 271,580 read and 4,000 write capacities.

4.3.2 Query

The following queries were used to perform Workload E.

Table 4.3.2 *Evaluated queries for Workload E*

	Scan	Update

Couchbase SQL++ / Key–Value API	`SELECT meta().id`
	`FROM ` `` `bucket` ``
	`WHERE meta().id >= $1`
	`ORDER BY meta().id`
	`LIMIT $2`
		`collection.upsert(id, content, upsertOptions().timeout(kvTimeout).expiry(documentExpiry).durability(persistTo, replicateTo))`

DynamoDB API	`{`
	`    "TableName": "usertable",`
	`    "Key": {`
	`            "_id": "$1"`
	`        },`
	`    "ConsistentRead": "false"`
	`}`
	
	`{TableName: usertable, AttributesToGet: [id]}`
		`{`
		` "TableName": "usertable",`
		` "Key": {_id = {S: $1}},`
		` "AttributeUpdates": {`
		`   $2={Value: {S: $3}}`
		` },`
		` "Action": "PUT"`
		`}`

4.3.3 Evaluation results

Amazon DynamoDB had the highest throughput on each type of cluster, with the best result of 110,080 ops/sec on an 18-node cluster. However, the lowest latency was achieved on a 3-node cluster with 4.89 ms. Except for the 18-node cluster, DynamoDB had the lowest latency. On 18 nodes, Couchbase Capella was the best with 3.84 ms, while having a throughput of 71,170 ops/sec.

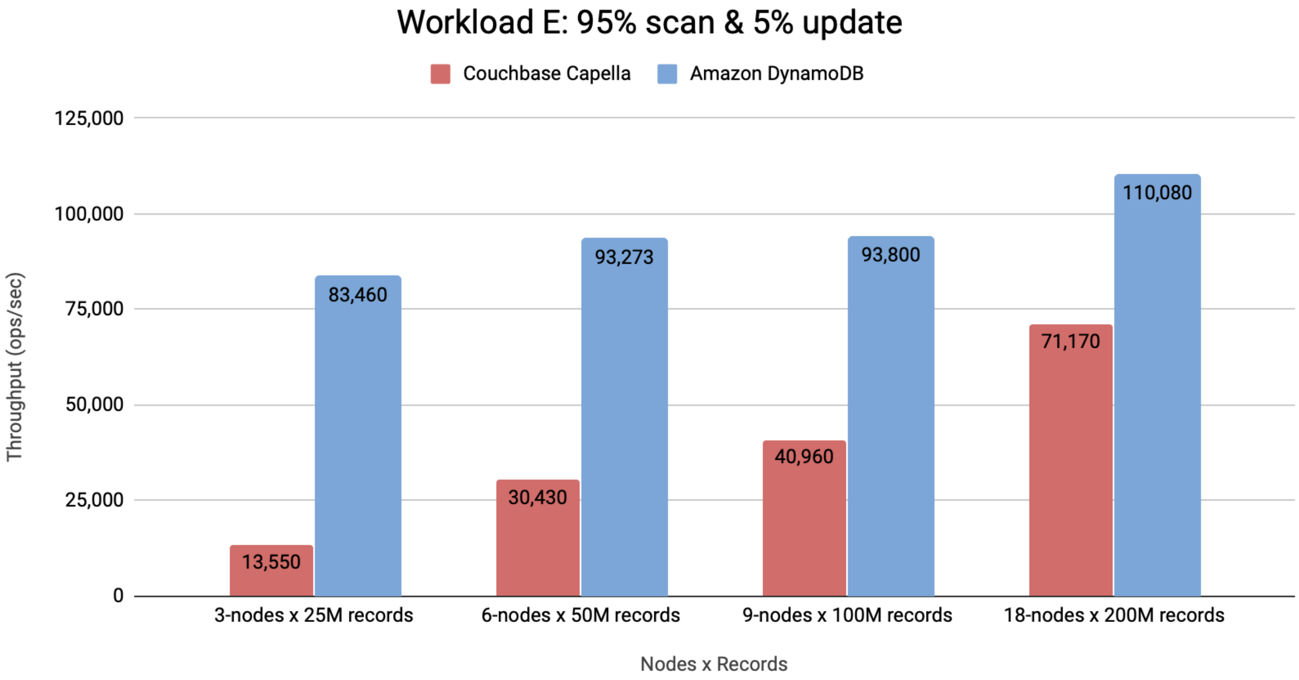

Figure 4.3.3.1 Throughput results under Workload E on 3-, 6-, 9-, and 18-node clusters

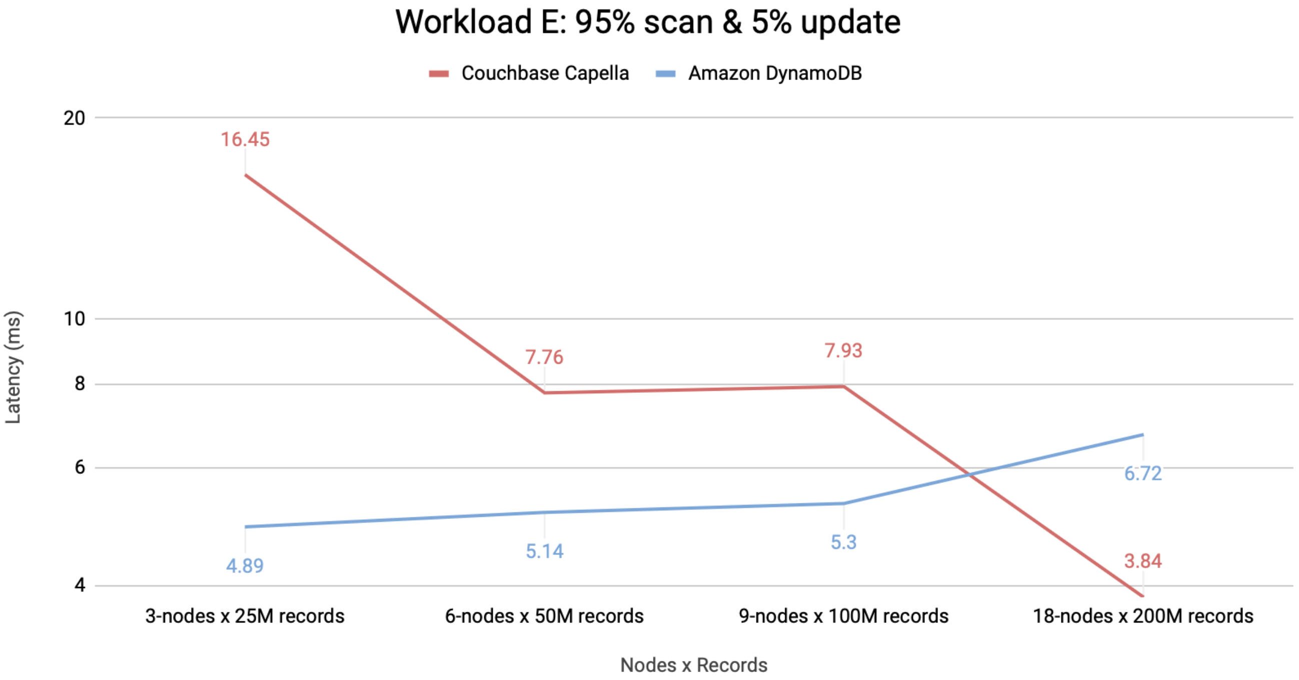

Figure 4.3.3.2 Latency results under Workload E on 3-, 6-, 9-, and 18-node clusters

In this workload, DynamoDB exhibited a high rate of failed operations, averaging around 50%. However, this percentage decreased slightly to 42% when running on an 18-node cluster. AWS recommends that customers have provisioned adequate capacity for their workload, or considered using exponential back-off functions, or changed to the on-demand mode, or increased the read and write throughput quotas to avoid these errors. All of which will either slow down the throughput, or increase its costs. Had we configured the DynamoDB test to avoid the errors generated from it hitting its financial ceiling, we believe that the database would cut its throughput in half and amplify its latencies.

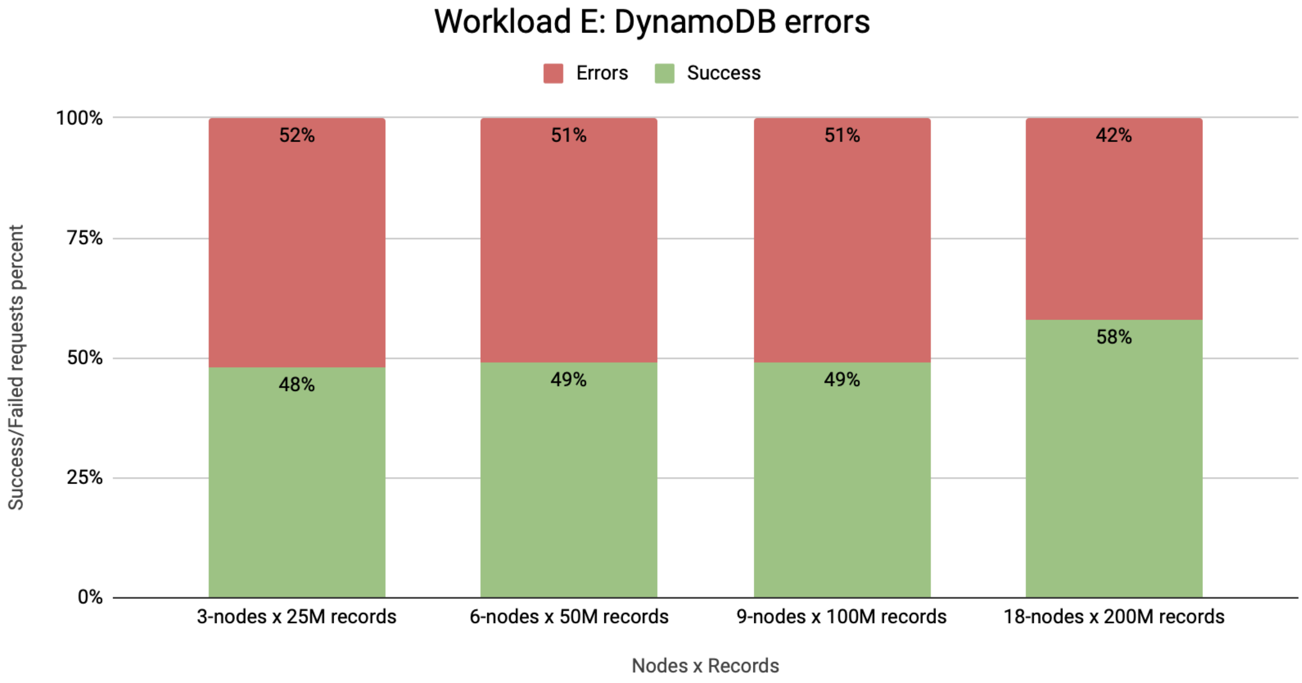

Figure 4.3.3.3 Results for DynamoDB under Workload E on 3-, 6-, 9-, and 18-node clusters

4.3.4 Summary

Under *Workload E*, DynamoDB demonstrated the best throughput results with a gradual increase in operations per second, but produced a high number of failed requests. The latency of DynamoDB increased from 4.89 ms on a 3-node cluster to 6.72 ms on 18 nodes.

As the number of nodes grew, Couchbase Capella exhibited good results in both latency and throughput without throwing errors. The latency was the lowest and most predictable across all the databases, decreasing from 16 ms on 3 nodes to 3.84 ms on an 18-node cluster. Additionally, the size of the cluster significantly impacted Couchbase Capella's throughput, which increased from 13,550 to 71,170 ops/sec, demonstrating more than a 5x improvement.

The cost per billion operations for this workload remained predictable and steady for Capella, as well, eventually coming in lower than DynamoDB at the highest scale (see Appendix).

5. Conclusion

Typically, no single database as a service is perfect for meeting all the requirements of a given scenario. Each solution has its advantages and disadvantages, which may become more or less important depending on the specific criteria. Despite this, DBaaS can help engineers to reduce the time needed for deployment, configuration, and support.

Though DBaaS solutions do not offer broad system tools for configurations, the databases have been optimally tuned for each workload. Therefore, configurations can be changed based on workloads.

Couchbase Capella performed better than AWS DynamoDB in Workload A and C Workload. It had good results in Workload E, where DynamoDB had the highest throughput with more than 40% of failed operations. In Workload C. Capella was good overall and showed that it is capable of performing any type of query with good performance.

Amazon DynamoDB is significantly different from Couchbase Capella, since it operates as a pure service without proper tuning available. Only two parameters can be changed: read and write capacities, which were calculated based on the cost of other databases for each workload. Unfortunately, DynamoDB produced a significant amount of failed requests.

6. Appendix

6.1 Indexes for the scan query

Couchbase indexes

```
CREATE PRIMARY INDEX ON `bucket` WITH {"num_replica":
NUMBER_OF_INDEX_NODES - 1}
```

6.2 Pricing

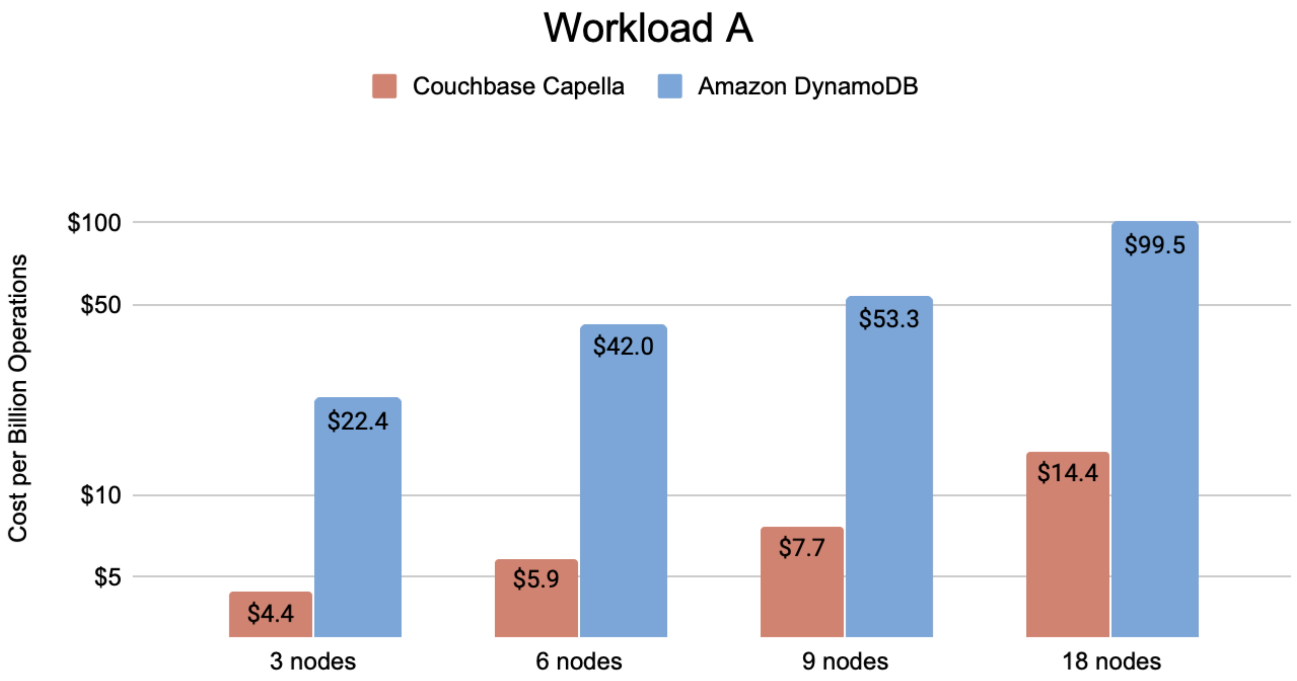

Figure 6.4.1 Cost per billion operations for Workload A

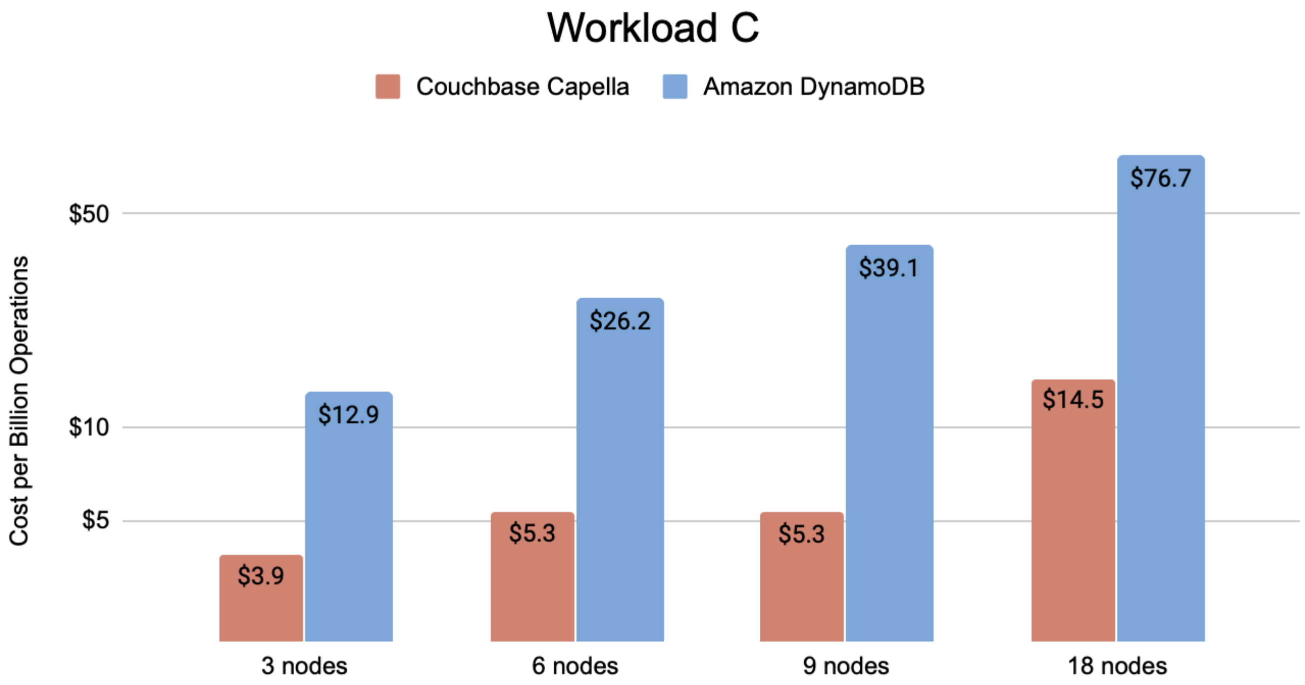

Figure 6.4.2 Cost per billion operations for Workload C

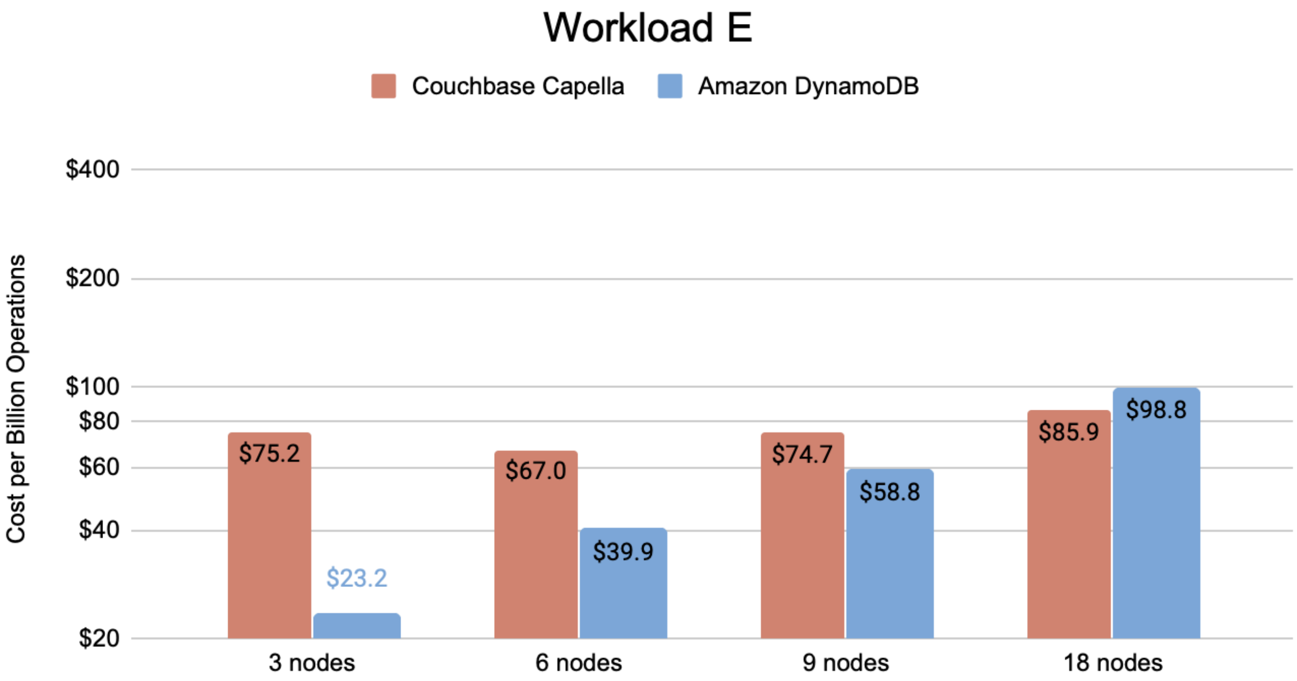

Figure 6.4.3 Cost per billion operations for Workload E

7. About the author

Ivan Shyrma is Data Engineer at Altoros with extensive hands-on experience in high-load, scalable applications and web services development. Ivan has worked as a full-stack engineer for several years designing durable distributed systems. He is able to create complex architecture solutions, adopt systems for production use, and is keen on resolving any engineering problems.

Altoros is an experienced IT services provider that helps enterprises to increase operational efficiency and accelerate the delivery of innovative products by shortening time to market. Relying on the power of cloud automation, microservices, AI/ML, and industry knowledge, our customers are able to get a sustainable competitive advantage. For more, please visit www.altoros.com.

Want more?

To download other research papers and articles like that:

- check out our resources page
- subscribe to the blog
- or follow @altoros for daily updates

Feel free to contact us if you'd like to discuss your project.