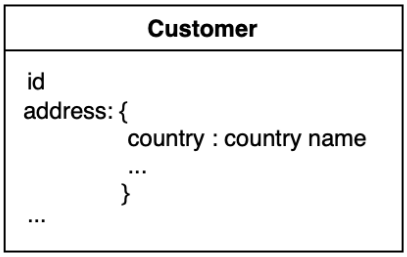

Figure 3.2.2 A graphic representation of the customer model

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

For MongoDB Atlas, we have sharded the collection by `_id` (see Appendix 6.1, "*MongoDB Atlas shard collection and indexes*").

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(kvTimeout).expiry(documentExpiry).durability(persistTo, replicateTo))`
MongoDB Query	`db.ycsb.find({_id: $1})`	`db.ycsb.update( { _id: $1 }, { $set: { fieldN: $2 } })`
DynamoDB API	`{ "TableName": "usertable", "Key": { "_id": "$1" }, "ConsistentRead": "false" }`	`{ "TableName": "usertable", "Key": {_id = {S: $1}}, "AttributeUpdates": { $2={Value: {S: $3} } }, "Action": "PUT" }`
Redis CLI	`HMGET $1 $2`	`HMSET $1 $2 $3`

4.1.3 Evaluation results

On each type of a cluster, Couchbase Capella significantly outperformed the other databases. 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, the other databases also demonstrated better performance. Each of the databases showed its best performance on an 18-node cluster. MongoDB Atlas achieved 58,290 ops/sec with a 8.13 ms latency, DynamoDB reached 109,350 ops/sec with a 5.1 ms latency, and Redis Enterprise Cloud was close to DynamoDB with 98,700 ops/sec and a 4.15 ms latency.

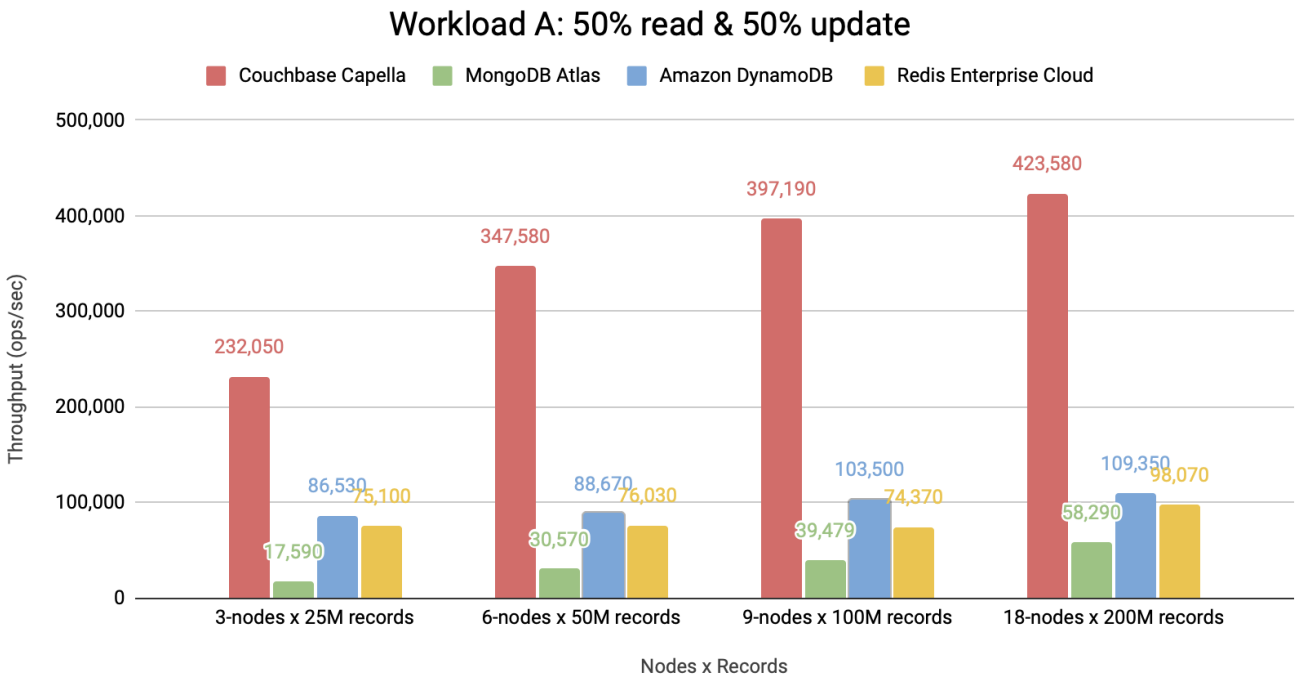

Figure 4.1.3.1 Throughput results under Workload A on 3-, 6-, 9-, and 18-node clusters

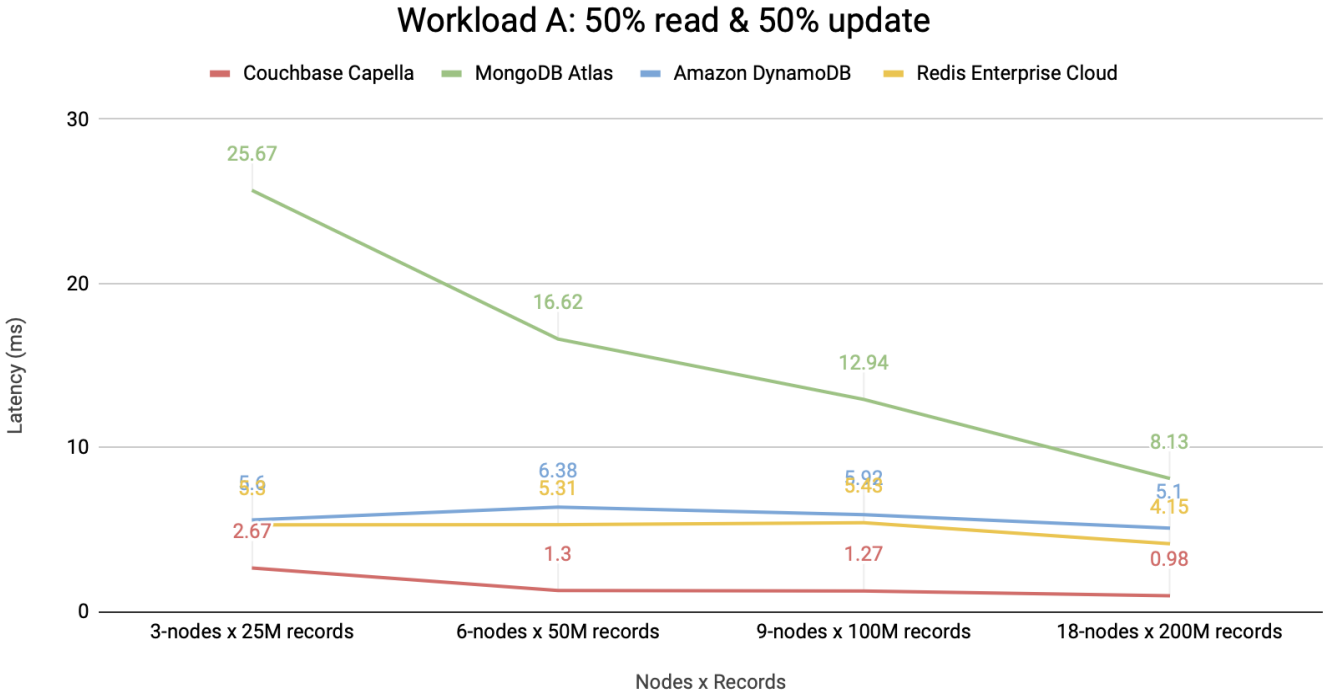

Figure 4.1.3.2 Latency results under Workload A on 3-, 6-, 9-, and 18-node clusters

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.

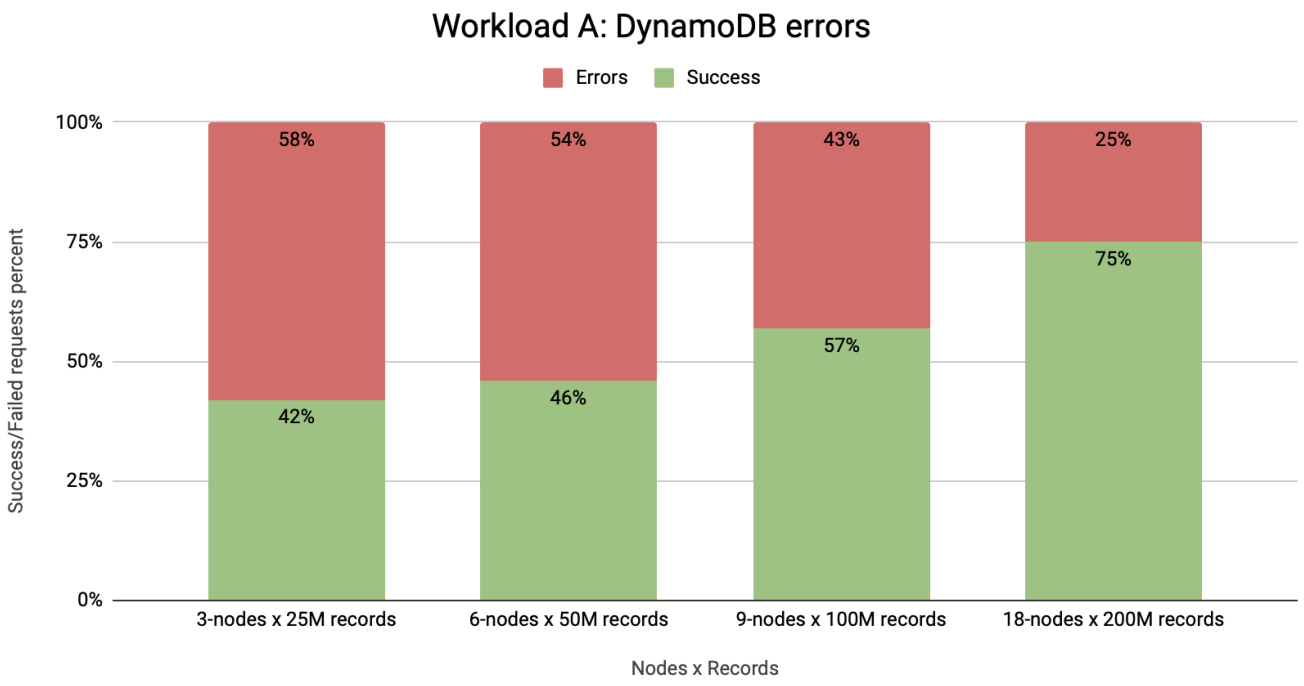

Figure 4.1.3.3 Results for DynamoDB under Workload A on 3-, 6-, 9-, and 18-node clusters

4.1.4 Summary

The throughput of each database grew constantly depending on the type of a cluster. All databases achieved the throughput limit for each cluster type, except DynamoDB for read operations on an 18-node cluster. Couchbase Capella demonstrated high throughput growth and clearly outperformed MongoDB Atlas, Amazon DynamoDB, and Redis Enterprise Cloud 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. Redis Enterprise Cloud had a stable latency of about 5 ms, which decreased to 4.15 ms on an 18-node cluster. It also showed significant improvement in throughput only on an 18-node cluster. The latency of MongoDB Atlas decreased continuously from 25.67 ms on 3 nodes to 8 ms on an 18-node cluster. However, among other databases, MongoDB had worse results. Nonetheless, it demonstrated the most growth with a cluster's expansion from 3 to 18 nodes, with a 3x reduction in latency and a 3x increase in throughput. All the databases 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.